OOP

Nesneye yönelik programlamada (OOP-**Object Oriented Programing**) nesnelerin özellikleri ve metotları vardır.Özellikler bir nesne ile ilgili bilgileri tutan alanlardır.

ÖRNEK: televizyon bir nesne olsun.

Renk,bilgi ve şuankikanal televizyonun bir özelliğidir.

Televizyon.renk:=”renkli” veya Televizyon.renk:=”siyah-beyaz”

İf **Televizyon.renk:**=”renkli” then **Televizyon.bilgi**:=”bu televizyon renklidir”
Else **televizyon.bilgi**:=”bu televizyon renkli değil”

Kanaldeğiştir bir metot olur.

Televizyon.şuankikanal:=”atv” olsun.

Kanal değiştirmek için kumandaya basarsak “kumanda aletinin düğmesine basıldı” olayı oluşur.Bu olay kanal değiştir metodunu tetikler ve kanal değişir.

Özellikler bir nesne ile ilgili bilgilerin tutulduğu alanlardır.

Metotlar ise OOP’de nesneyi ilgilendiren prosedür ve fonksiyonlara verilen isimlerdir. Metot her nesnenin kendi özelliklerinden aldığı bilgileri işleyerek bir sonuca giden fonksiyon ve prosedürdür.

OLE

OLE, windows altındaki uygulamaların birbiri ile bilgi alış verişini sağlayan güçlü bir yöntemdir. Bu yöntemle uygulamalar arasındaki nesnelerin birbiri ile bağlantı kurması ve birbirlerindeki nesneleri kullanabilmeleri mümkün olmaktadır. OLE kullandığımızda, uygulamalarımız içinde ikinci uygulamaya ait bir nesne kullanırken, istediğimiz zaman bu ikinci uygulamayı programımız içinden çalıştırabilir ve bu nesne üzerinde bazı değişiklikler yaparak son haliyle bu nesneyi saklayabiliriz.

1- GİRİŞ

Delphi 5 üç paket halinde bulunmaktadır.

1- Standart
2- Professinal
3- Enterprise

KURULUM: İlgili CD den delphi butonunu tıklamak ile başlar. Kurulum için üç ayrı tipten biri seçilerek gerçekleştirilir.

1- Typical: En çok kullanılan tüm yönleri ile yüklenir.

2- Compact: En kısıtlı hali ile yüklenir. Yer az ise tercih edilir.

3- Costum: Bileşenlerin hangileri kurulacak ise seçim şansı vardır. Seçilen modüllere göre kurulum sağlanır.

Delphi 5; 6 adet bileşene sahiptir. Bunlar:

1- Program files
2- Shared files
3- BDE
4- SQL links
5- Database desktop
6- CORBA Support

Bileşenlerin seçiminden sonra kurulacak veri tabanı seçilir. İşlem devam ettirilir. Seçme işlemleri sonunda INSTALL seçeneği ile kurma işlemi gerçekleştirilir.

1.1- Delphi 5'deki Yenilikler

Delphi5'te değişiklikten çok yenilik vardır. Gelişen teknolojilere paralel olarak delphi5'e yeni bir çok bileşen eklenmiştir veya eskilerde düzenlemeler yapılmıştır. Bu değişiklikler Object Inspector'da, Project Manager'de, AUTO-Hotkey özelliği'nde, Auto-Create Form option'da, Proje Explorer'da, Web sayfaları oluşturma v.b. modullerde olmuştur.

1.2- Delphi 5 Genel Özellikler

- Delphi 5 kendisine dil olarak Object Pascal'ı seçmiştir.
- Inprise (Borland) 'ın component (bileşen) teknolojisini kullanır.
- 32 bitlik derleyiciye sahiptir.
- Win 95, win 98, Win 2000, Windows NT altında çalışan ve bu ortamlarda çalışacak programlar üretilebilen uygulama geliştirme aracıdır.
- Borland'ın Delphi ve C++ kodlarını alarak EXE üreten aracı her iki dilide birbirine oldukça yaklaştırmıştır. Bu sayede Delphi'de yazılan kodları C++ 'da görmek mümkündür.
- En güçlü yanlarından biride Windows API fonksiyonlarının tümüne rahatlıkla ulaşıyor olmasıdır.Bu sayede güçlü ve hızlı programlar geliştirilebilmektedir.
- Delphi 5 176 klasör, 2204 dosya ile 138 MB yer kaplamaktadır.

Bu özelliklerinden dolayı görsel dil olarak çok yaygın olarak kullanılmaktadır.

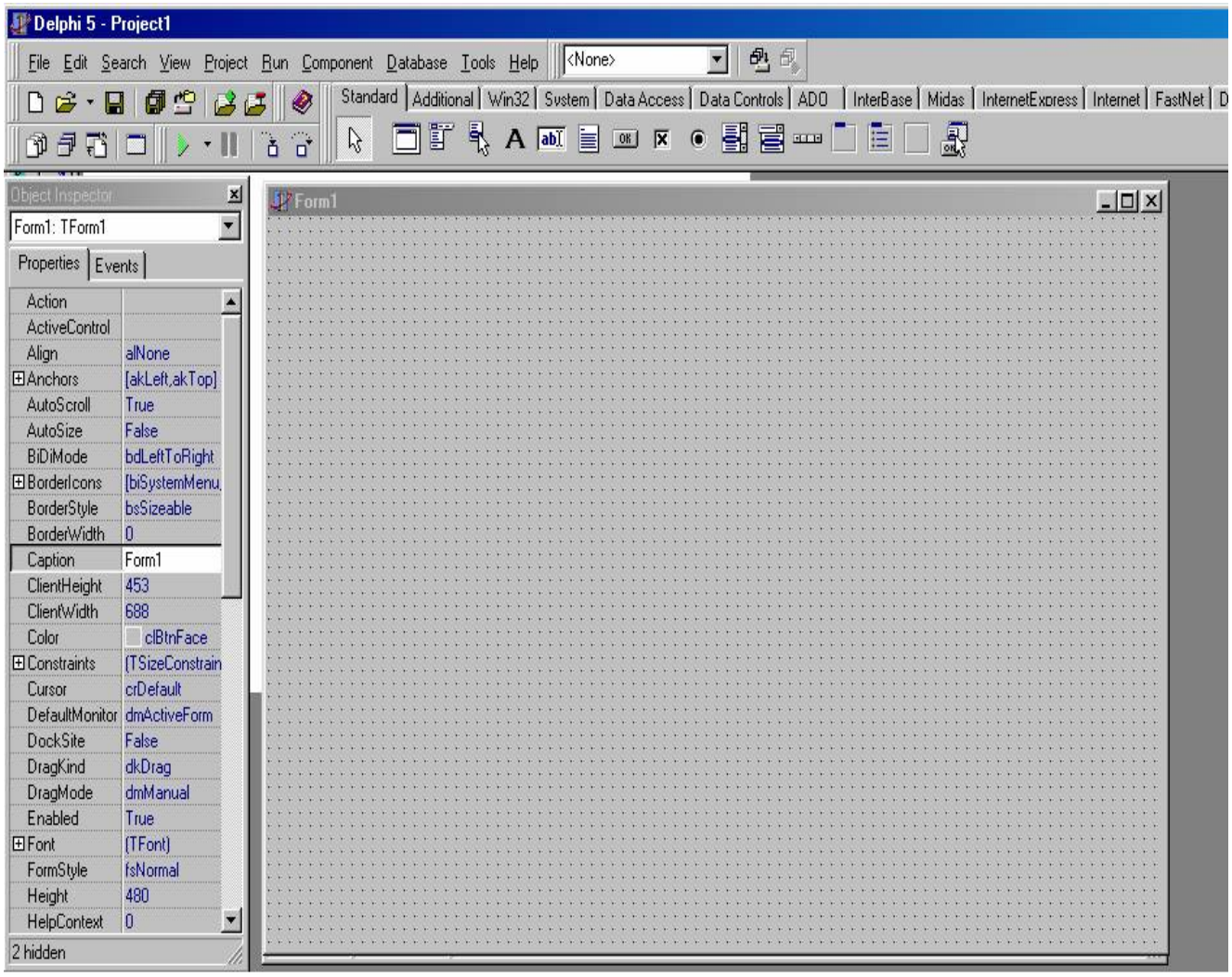
1.3- Delphi'ye Giriş ve Çıkış İşlemi

Delphi'ye giriş için Başlat menüsünden, Programlar seçeneğinden Delphi bulunur çalıştırılır veya masa üstüne oluşturulmuş herhangibir kısayol var ise çalıştırılır. Bu işlem otomatik olarak bizi Delphi çalışma ortamına getirir.

Delphi'den çıkış için Çalışma ortamında bulunan Menülerden File seçilir, bunun seçeneklerinden bazıları çıkış işlemini gerçekleştirir. Bu seçenekler çık, sakla çık v.b. olabilir.
Bu işlemde bizi windows ortamına geri gönderir.

1.4- Delphi 5 Uygulama Geliştirme Ortamı

Delphi çalıştırıldığında karşımıza çıkan aşağıdaki görünümde bir çalışma ortamı gelir.



Delphi çalıştırıldığında karşımıza çıkan ekrana ***IDE (Integrated Development Environment-Tümleşik Uygulama Geliştirme Ortamı)*** adı verilir. IDE adının verilmesi;İ

- Uygulama geliştirmenin,
- Derlemenin,
- Hata ayıklamanın,
- Çalıştırmanın tek ortamda yapılıyor olmasıdır.

IDE 4 bölümden oluşur. Bunlar;

1. Ana Pencere (MENÜ+BİLEŞENLER+HIZLI ÇALIŞMA ORTAMI)
2. Nesne İnceleyicisi (OBJECT INSPECTOR)

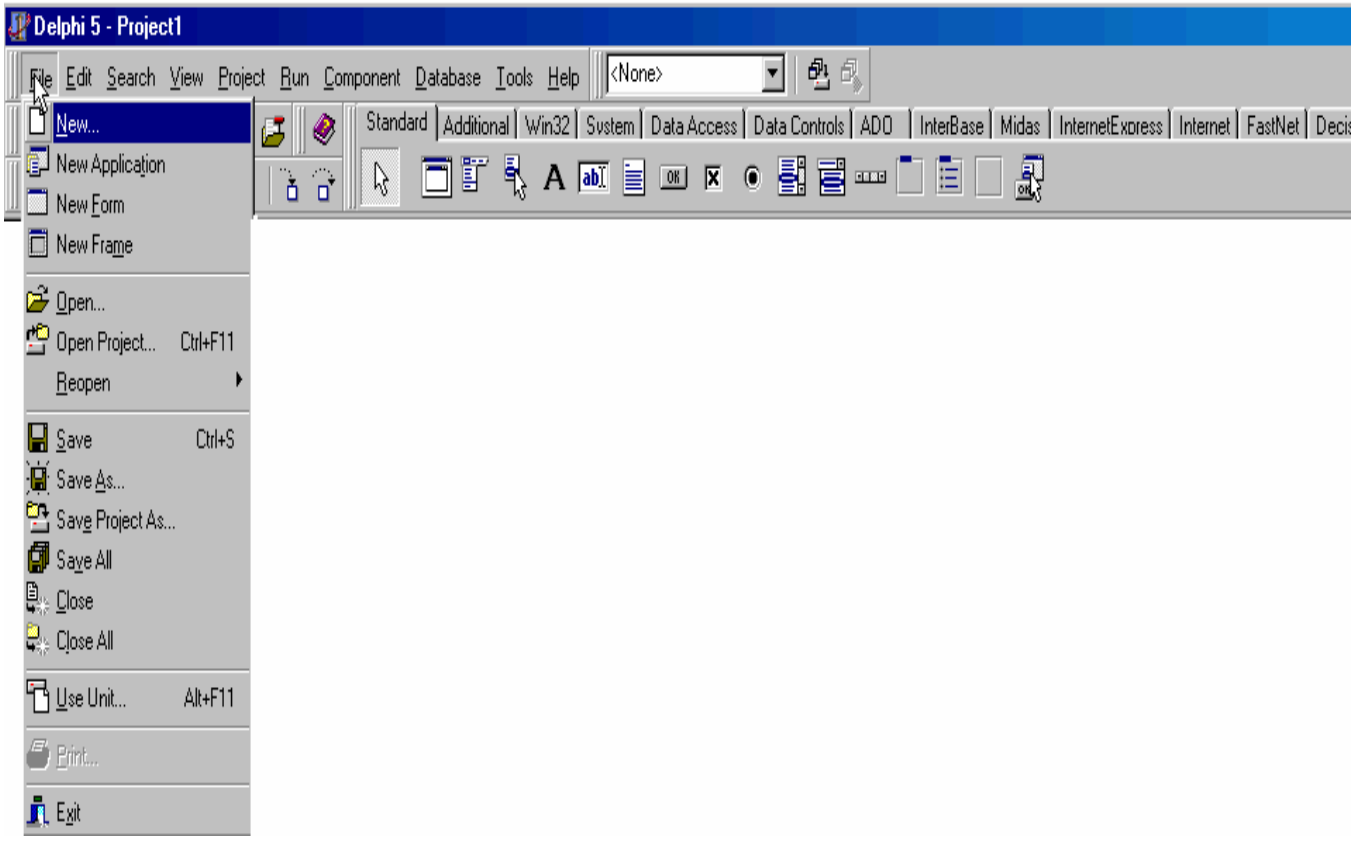
3. Kod Editörü (UNIT)
4. Form Tasarımcısı (FORM)

ANA PENCERE: En üst kısımda gördüğümüz penceredir. Kendi içinde 3 ana bölümden oluşur. Bunlar ;

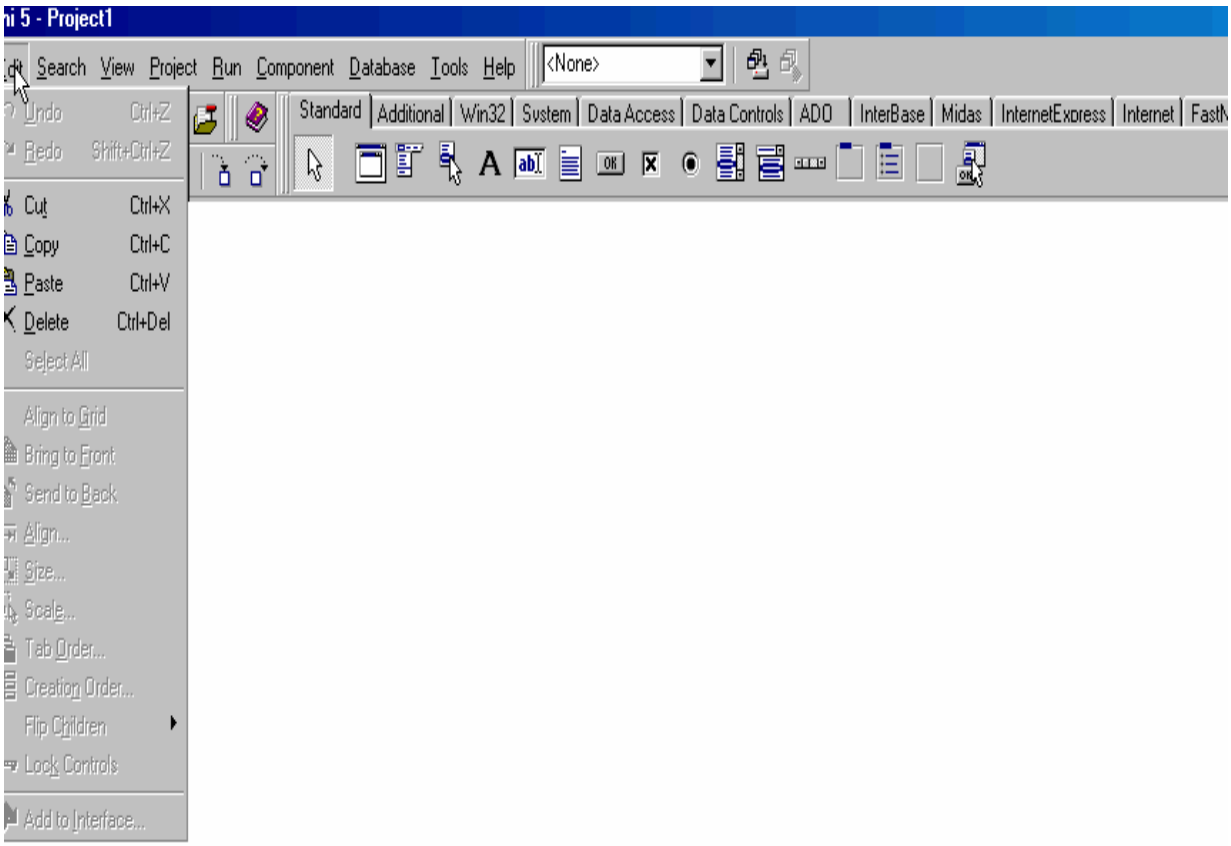
- Ana Menü
- Speed Bar
- Bileşen Sekmeleridir.

Ana menü: Burada Windows ortamında çalışan programlardan alıştığımız klasik menülerden biri bulunmaktadır. En üst sıradadır. Her menü seçeneğinin farklı işlevleri vardır. Seçenekler ise; File, Edit, View, Project, Run, Component, Database, Tools ve Help'ten oluşmaktadır.

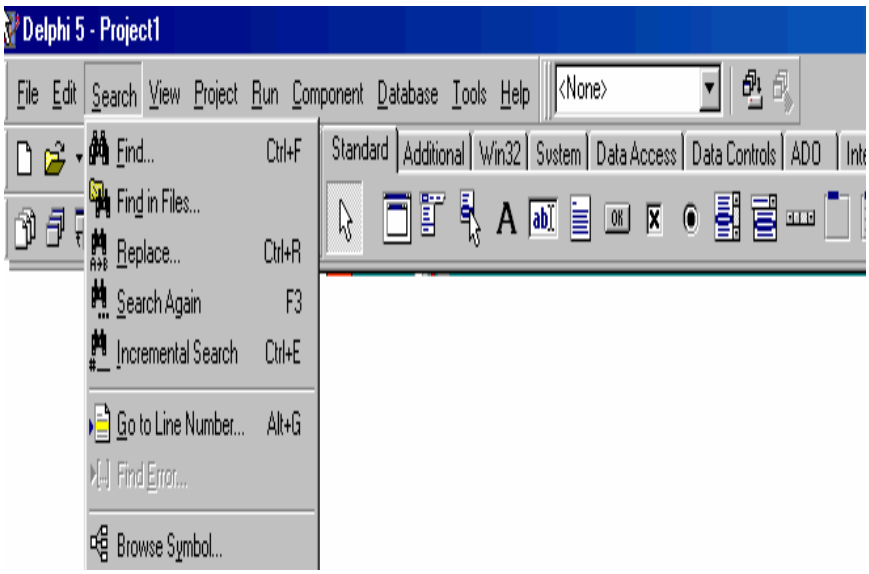
File seçeneği: Burada dosya işlemleri yürütülür. Ekran görünümü aşağıdaki şekildedir.



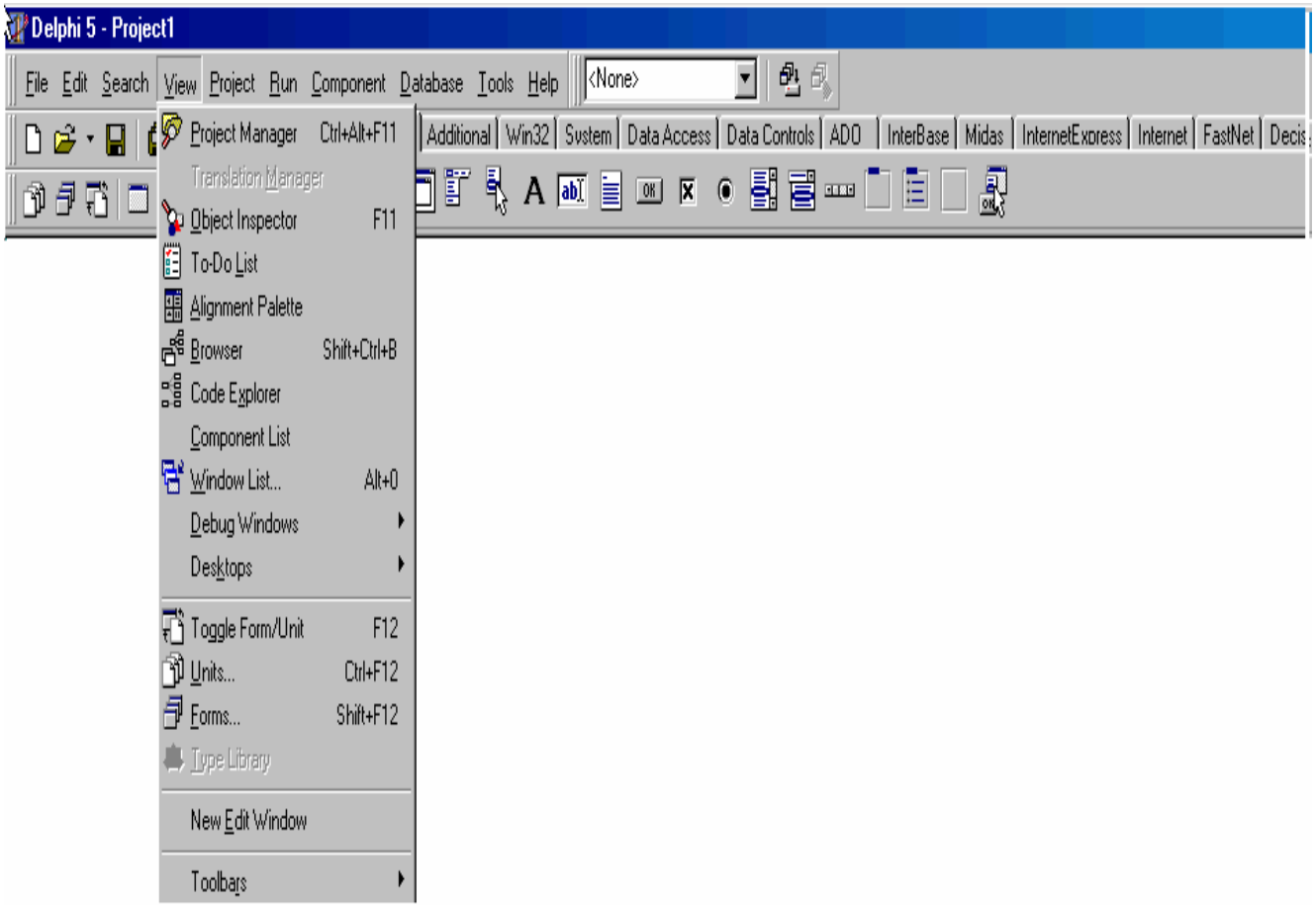
Edit seçeneği: Burada düzeltme işleri yani kesme, kopyalama, silme v.b. işlemler yürütülür.



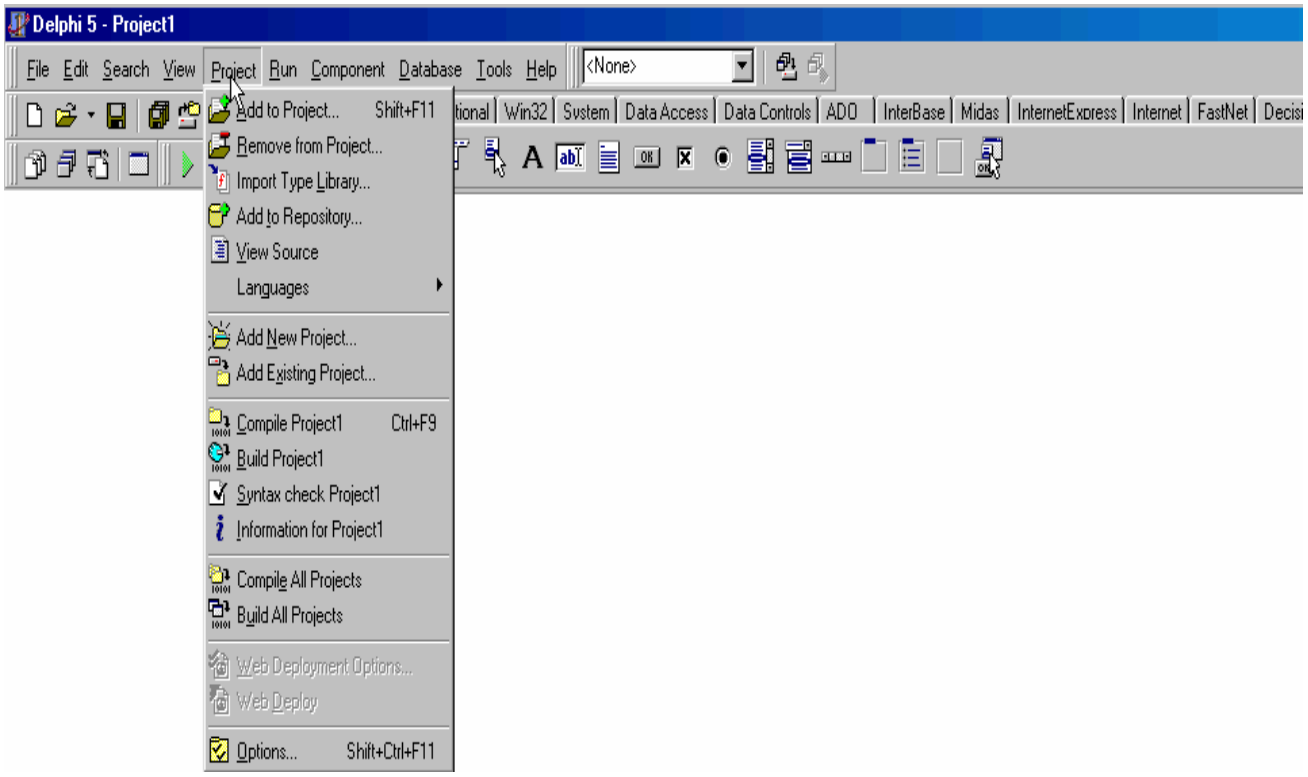
<u>Search seçeneği:</u> Bu seçenekte arama, yeniden düzenleme v.b. işlemler yapılır.



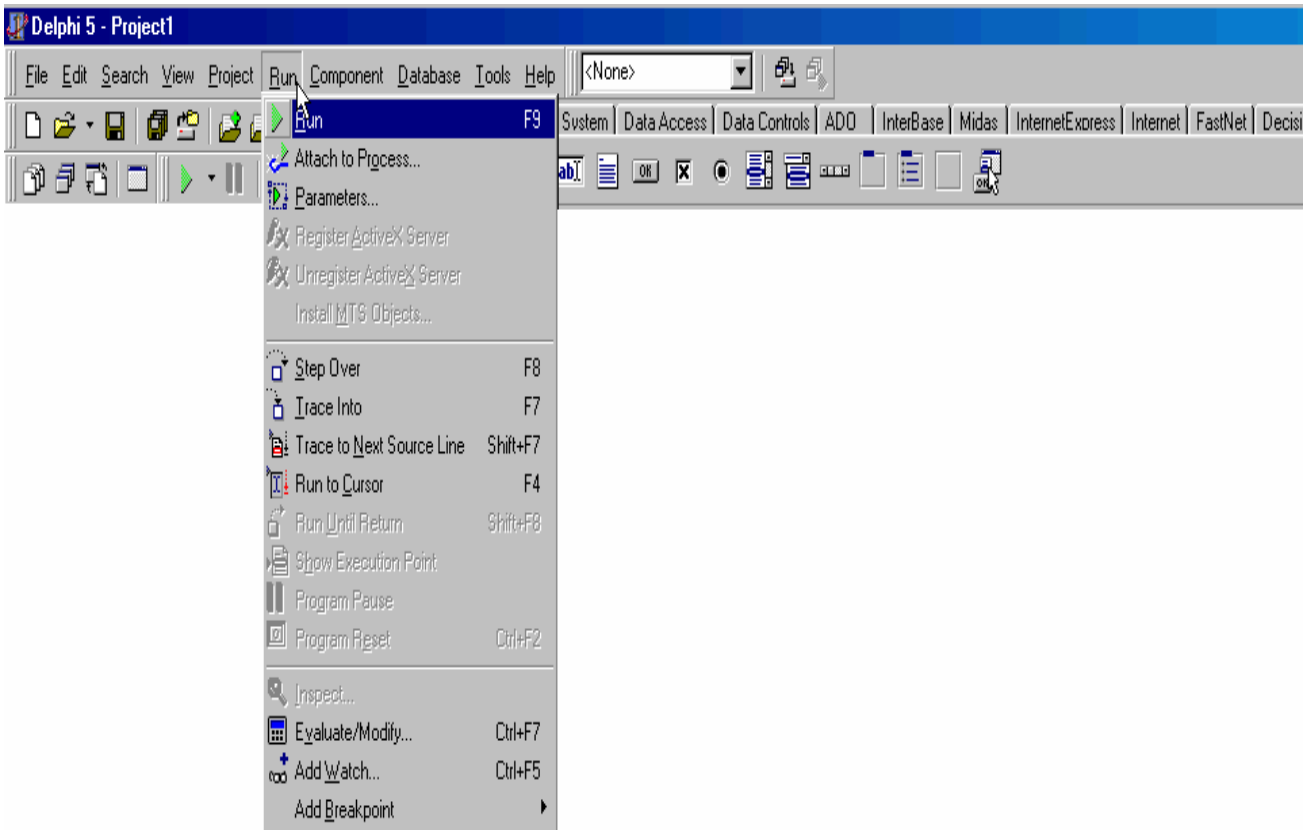
View seçeneği: Burada projede kullanılacak araçlar seçilir.



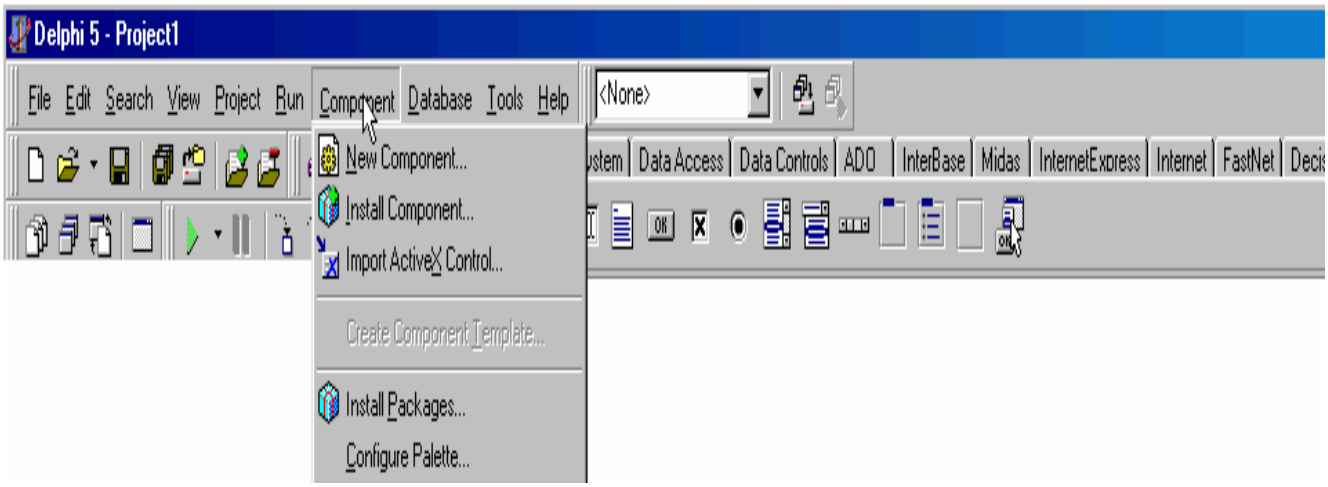
Project seçeneği: Proje üzerinde işlemler yapmak için kullanılır.



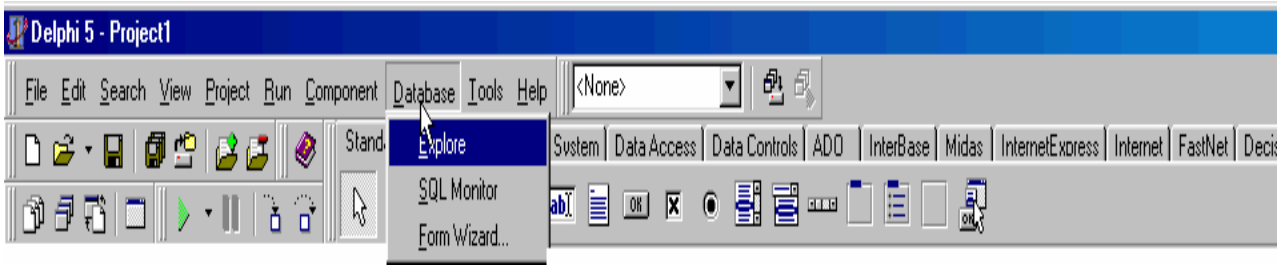
Run seçeneği: Projeyi çalıştırma işlemler yapmak için kullanılır.



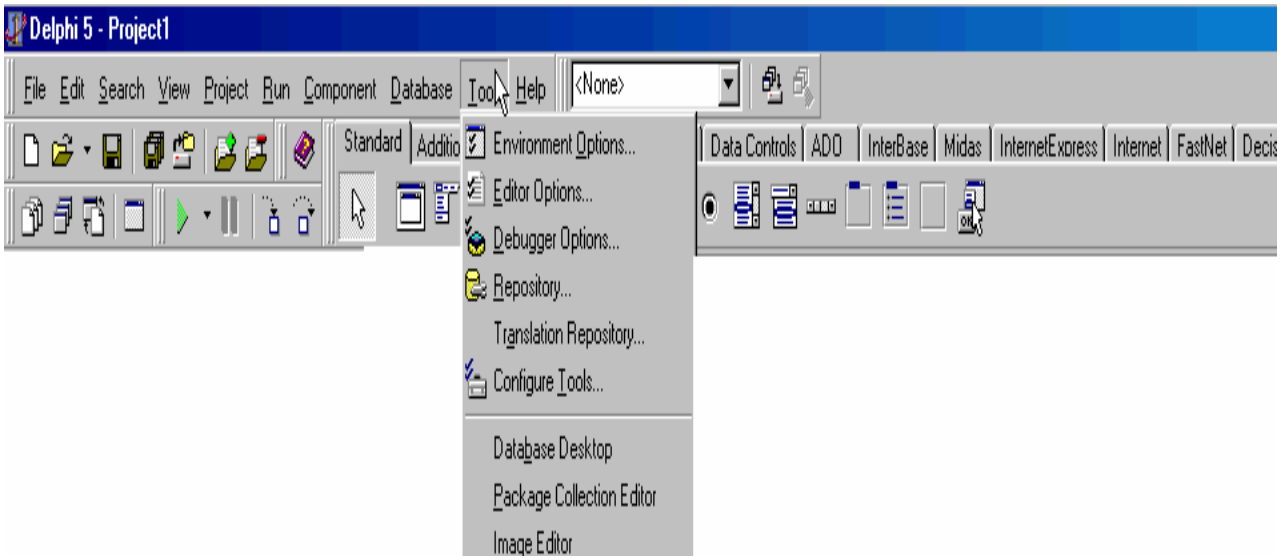
Component seçeneği: Projede kullanılacak olan componentler için kullanılır.



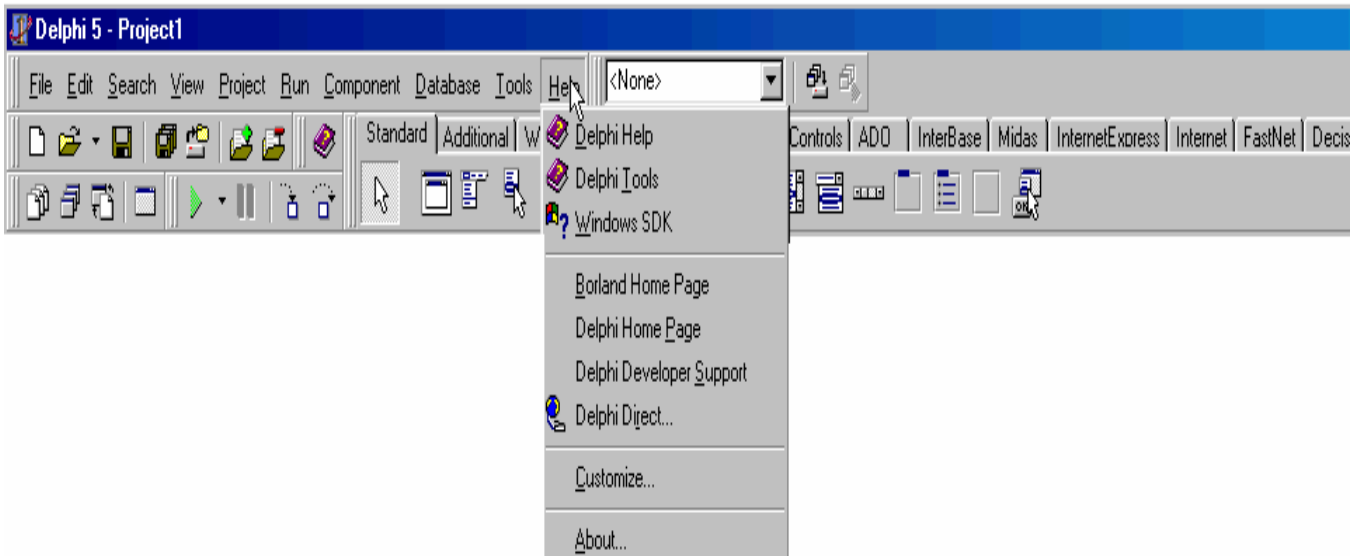
Database seçeneği: Veri tabanı işlemleri için kullanılır.



Tools seçeneği: Projede kullanılan araçlar üzerinde işlemler yapılan seçenektir.



Help seçeneği: Yardım seçeneğidir.



Speed Bar: Uygulama geliştirme esnasında sık kullanılan nesneler buraya yerleştirilmiştir. Programcı isterse buraya yeni nesneler ekleyebilir veya çıkarabilir.

Bileşen Sekmeleri: Bu sekmeler Delphi'nin temelini oluşturan ,işlemleri kolaylaştıran bileşenlerin(Component) bulunduğu yerdir. Programcı isterse buraya yeni bileşenler ekleyebilir veya çıkarabilir.

NESNE İNCELEYİCİSİ (OBJECT INSPECTOR): IDE'nin solundaki dikey penceredir. **Properties** (özellikler) ve ***Events*** (olaylar) sekmesinden oluşur.

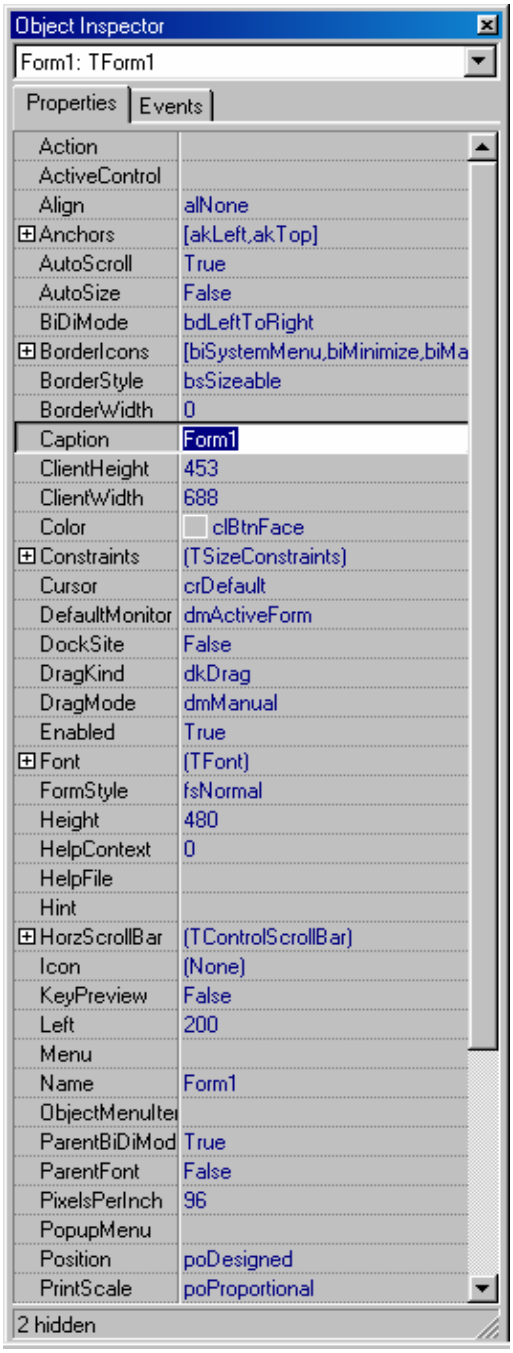
Delphi çalıştırıldığında otomatik olarak ekrana gelir. Formun veya formun içindeki nesnelerin özelliklerini bulundurur. Bu pencerede tüm özellikler alfabetik sıradadır. İstenirse seçenekler özelliklerine göre listelenebilir. Bunun için ilgili pencere üzerinde iken sağ tuş tıklanır**. Arrange** seçeneği kullanılır. Buradaki seçeneklerden **by category** seçilir.

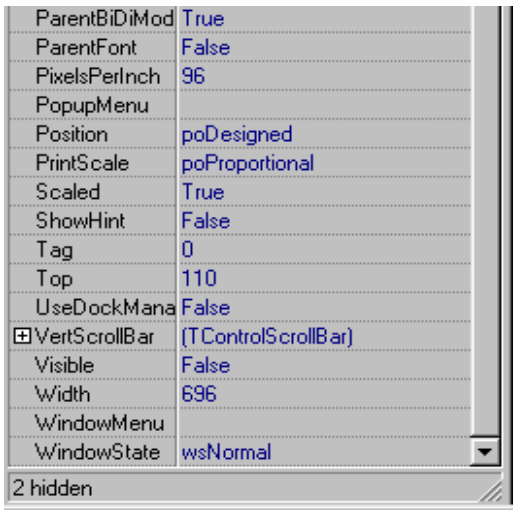
Yine sağ tuş tıklanarak view seçeneği seçildiğinde nesnelerin tüm özelliklerinin görüntülenmesini istemiyorsak ilgili işaretini iptal ederiz.

View menüsünden görüntülenmesini istediğimiz veya istemediğimiz özellikleri on/off yapabileceğimiz gibi aynı menüden All, toggle, none seçeneklerinide kullanabiliriz.

All : Tüm özellikleri listeler.
Toggle: O sırada görüntülenen özellikleri on/off yapar.
None: Tüm özellikleri görüntüden kaldırır.
Object ınspector penceresini kapatmak için kısa yoldan **hide** seçilebilir. Ters işlemi için **view** seçeneği tekrar on yapılır.

Olaya dayalı programlama (Event-Driven Programming) görsel uygulama geliştirme ortamları ve nesneye yönelik programlama ile gelişme göstermiştir





FORM TASARIMCISI: Formlar üzerine delphi bileşenleri koyabileceğimiz pencerelerdir. Her Delphi programı en az bir adet formdan oluşur. Bileşen (component) paketindeki bileşenleri sürükle bırak özelliği ile forma yerleştirebiliriz. Form üzerinde yerlerini değiştirebiliriz. Bileşenleri kullanma Delphi ile programlamanın temel taşlarındandır.

İlk açılışta içi boş olarak ***form1*** adı ile görüntülenir. İstenirse adı değiştirilebilir. Kullanımını biraz tanıyacak olursak; sol üst köşede form denetimini sağlayacak olan denetim menüsü açılır. Sağ üst köşede kapatma, kaplama, küçültme işlemleri yürütülür. Kullanıcı tarafından dizayn edilir.

KOD EDİTÖRÜ PENCERESİ: Kodların yazıldığı dilin adı Object Pascal'dır. Bir form oluşturulduğunda kodları yazmak için bir modül (unit) oluşturulur.Bu modül Delphi ilk başlatıldığında projeye dahil edilen ve form1 penceresinin altında kalan penceredir. UNIT1.PAS adı ile pascal kodu içerir. Unit adı istenirse kullanıcı tarafından değiştirilebilir. Bu pencere düzey olarak iki bölümden oluşur. Soldaki seçenekleri kullanarak unitin başka bir yerine geçilebilir, unit hakkında bilgi edinilebilir.

Burada; format, variable/constanst ve uses seçenekleri vardır.Uses ile unite dahil edilen bazı unitler izlenebilir.. Bu pencere kapat düğmesi ile kapatılabilir. Tekrar açılması için view menüsündeki Code explorer seçilir. Buradaki kod satırları otomatik olarak hazırlanır.

1.5- Project Manager (Proje Yöneticisi)

Uygulama geliştirmede en önemli sorun form ve modüllerin artması ile birinden diğerine geçişlerin zorlaşmasıdır.

Bunun için; View menüsünden Project Manager seçeneğinden tüm modüllerin listesi görüntülenir. Buradan seçim yapılabilir. Ayrıca burada yeni bir modül eklenebilir, silinebilir, çift tıklama ile çalıştırılabilir.

1.6- Proje Dosyaları

Proje geliştirildiğinde dizininizde farklı uzantılı dosyalar oluşur. Bunları şöyle özetleyebiliriz.

Ana proje dosyası: **.DPR** uzantılıdır. Oluşturulan EXE dosya adını bu dosyadan alır.Bu dosyada projede kullanılan formların, modüllerin listesidir. İlk oluşturulan dosyadır.

Grafik form dosyası: **.DFM** uzantılıdır. Çalışılan form ile ilgili tanımları tutar.Görsel özellikler kayıt edilir.

Pascal Modülleri: .PAS uzantılıdır. Object pascal kodunu içerir.

Proje seçenek dosyası: **.OPT** uzantılıdır. Proje ile ilgili opsiyonların tutulduğu dosyalardır. Her proje için bir tanedir.

Derleyici kaynak dosyası: .RES uzantılıdır.Uygulamaya ait simge (icon) gibi dış kaynakların tutulduğu dosyalardır.

Yedek proje dosyası: **.~DP** uzantılıdır. Proje 2.kez kayıt olduğunda oluşur. Son saklamadan önceki halini tutar.

Yedek modül dosyası: **.~PA** uzantılıdır. Unit 2. kez kayıt edildiğinde oluşur. Kodların son saklamadan önceki halini bulundurur.

Yedek grafik dosyası: **.~DF** uzantılıdır. .DFM uzantılı dosyayı 2. kez kayıt edildiğinde oluşur.

Masaüstü seçenek dosyası: **.DSK** uzantılıdır.Ana menüde Tools menüsünden Options seçeneği ile çalışma ortamında değişiklik yapılmış ise oluşur.

Derlenmiş modül: **.DCU** uzantılıdır. .PAS uzantılı dosya derlendikten sonra oluşur.

Tip kütüphaneleri (type library): .TLB uzantılıdır. .COM nesneleri için Delphi tarafından oluşturulur.

Tüm bunların dışında zaman zaman daha farklı dosyalar ilede karşılaşmak mümkündür. Bunlardan bazıları şunlardır:

.EXE : Derlenmiş dosya
.WMF, .BMP : Resim dosyaları
.ICO simge dosyalrı
.HLP : yardım dosyalırı
.HTM : İnternet Web sayfalarında kullanılan dosyalar
.CFG : proje seçeneklerini içerir
.DOF : Proje seneklerinin mevcut ayarlarını içerir.
.OCX : ActiveX denetimi veya formları içeren özel .DLL dosyalarıdır.
.DSM : Tüm tarayıcı sembollerini içerir.
.DLL : Çalıştırılabilir dosyaların bir başka sürümüdür.

2- FORMLAR VE BİLEŞENLER:

Formlar, görsel uygulamanın temelini oluşturur. Bir takım nesneler formlar üzerine yerleştirilir. Bu nesnelerein fare ile etkileşimi ile ortaya bir takım sonuçlar çıkar. Bu sonuç nesnelere bir takım kodların eklenmesi ile oluşur.

Bileşenler (component) Delphi'nin formlar üzerine yerleştirebildiği nesnelerin her biridir. Bileşenler sayfasından seçilir. Forma yerleştirilir. Daha sonra istenirse özellikleri ile oynanabilir. Başka kodlar ile ilişkilendirilebilir.

Bileşenler panelinde pek çok sayıda bileşen vardır. Bunlar gruplandırılarak sekmeler dağıtılmıştır.Bunları aşağıdaki şekilde özetleyebiliriz.

Standart : En fazla kullanılan nesneler buradadır. Standart olan bileşenleri kapsar.

Additional : Window tabanlı her uygulamada görülmez. Tshape ve Timage gibi grafik tabanlı bileşenleri bulundurur.

Win 32: 32 bit Windows ile birlikte gelen bazı bileşenleri kullanabilmek için delphi tarafından hazırlanan bileşenleri bulundurur.

Data Access: Bu bileşenler yardımı ile sorgulama, veritabanı, tablo bağlantıları v....b. işlemler yapılır.

Data Controls: Veritabanındaki kayıtları ekrandaki bileşenler ile ilişkilendirir.

Midas: DCOM COBRA standardında program geliştirmek için gereken bileşenleri bulundurur.

Win 3.1: Windows 3.1 ortamında kullanılacak bileşenleri bulundurur.

Dialogs: Dosya aç, sakla, yazıcı ayarları gibi Windows'ta sık karşılaşılan bileşenleri bulundurur.

System: Windows sisteminin bazı özellikleri ile ilgilki bileşenler vardır. DDE ve OLE'ye ait bileşenleri bulundurur.

Qreport: Görsel rapor geliştirme bileşenlerini bulundurur.

ActiveX:Başka programlama dilleri ile oluşturulan ActiveX bileşenlerine ait örnekleri bulundurur.

Desicision Cube: OLAP fonksiyonları ile ilgili bileşenleri bulundurur.

Samples: Örnek bileşenleri bulundurur.

Araç Çubuklarında Kısa Bir Gezinti:

Altı adet araç çubuğundan bahsetmiştik. Bunları kısaca açıklayacak olursak;

1- **STANDART:** Sol tarafta menü çubuğu altındaki 7 adet butondan oluşur. Bu butonlar ile yeni bir çalışma başlatılabilir, eski çalışmalardan biri seçilebilir, çalışma kayıt edilebilir, proje açılabilir, ekleme v.b. işlemler yapılabilir.

2- **VIEW :** Bu buton ile unit ve form görüntüleme, unıt ve form arasında gidip gelme, yeni bir form açma gibi işlemler yürülebilir.

3- **DEBUG :** Burada ise çalıştırma, bölüm bölüm çalıştırma, adım adım izleme gibi işlemler yürütülür.

4- **CUSTOM :** Help butonunu on/off yapmak için kullanılır.

5- **DESKTOP :** Masa üstüne saklamak için gerekli butonun on/off olması için kullanılır.

6- **COMPENENT PALETTE :** Program geliştirmak için kullanılan component butonlarının on/off olmasını sağlar. Burası 18 adet sekmeden oluşmuştur. Hangi sekme aktif hale getirilirse onunla ilgili butonlar alt satırında görüntülenir.

Projeleri Çalıştırmak:

Üzerinde çalışılan projeyi çalıştırmak için Run menüsünden **Run** komutu veya **F9** tuşu kullanılır.

Çalışmayı sona erdirmek için Form1'in kapat düğmesi tıklanır veya Run menüsünden program **Reset** kullanılır. Program reset çalışmayı geçici olarak durdurur.

Delphi Pencerelerinin Organizasyonu :

Üzerinde çalışılan pencerelerin yerlerini değiştirmek, kaydırmak, simge durumuna getirmek mümkündür.

Dockable seçeneği (kısayol sağ tıklama) project manager ve pencerelerinin iç içe durabilmeleri için kullanılır. **On/Off** şeklinde çalışır.

Object inspectore penceresinin başka bir pencere altında kalmasını istemiyor veya pencerenin sürekli en üstte olması isteniyor ise kısayol menüsünden **stay on top** komutu verilir.

Araç Çubuklarında Değişiklik Yapmak:

Bu işlem için;

- Araç çubuğu üzerinde iken sağ tuş tıklanır. (6 adet araç çubuğu listesi gelir.)

- Buradan **Customize** seçilir.

- Customize ekranında **toolbars, commands, optins** seçenekleri vardır.

- Araç çubuklarında değişiklik için **commands** butonu tıklanır. Burada komutlar gruplara ayrılarak yerleştirilmiştir.

- **Catagories** ve **Commands** seçiminden sonra seçilen buton araç çubuğuna eklenir.

- Butonlar üzerinde fareyi tutarsak ilgili butonun komut karşılığı ve kısayol tanımı görüntülenir. Bunları iptal etmek için yine **Customize** ekranındaki **Options** sekmesinde bulunan **show tooltips, show shortcut keys on tooltips** onay kutularının iptal edilmesi gerekir.

Masa Üstü Ayarlarını Kayıtetmek:

Bu işlem için **Current Desktop** butonu tıklanır. Saklanacak masaüstü için isim verilir.

3- STANDARD SAYFASI BİLEŞENLERİ

Delphi'de görsel uygulama geliştirme ortamının temel birimleri bileşenlerdir. Bu bileşenler kullanılarak uygulama geliştirebilir. Standard sayfasında bulunan bileşenler şunlardır:

- TButton bileşeni
- TEdit bileşeni
- Tlabel bileşeni
- TlistBox bileşeni
- TcomboBox bileşeni
- Tmemo bileşeni
- TcheckBox bileşeni
- TradioButton bileşeni
- TgroupBox bileşeni
- Tpanel bileşeni
- TscrollBox bileşeni
- TmainMenu bileşeni
- TpopUp bileşeni

Tbutton Bileşeni: Bu bileşen sayesinde form üzerine düğmeler yerleştirebilir ve bu düğmelere programın çalışması sırasında basıldığı zaman gerçekleşmesi istenilen işlemler bağlanabilir.. Düğmede görüntülenecek yazı **Caption** özelliğine yazılır.

Örnek: Forma yerleştirdiğimiz bir butona basıldığında İyi çalışmalar ifadesi görüntülensin.
Bu işlem için oluşturacağımız forma bir adet Tbutton bileşeni ekleyelim. İlgili prosedurünü aşağıdaki şekilde ifade edelim.

```
Begin
        Showmessage('iyi çalışmalar');
End;
```

TEdit Bileşeni: Bu bileşen ile dışardan bilgi girişi yapılır.

TLabel Bileşeni: Form üzerine statik yazılar yazmak için kullanılır. Tlabelin Caption adında bir özelliği vardır.Bu özelliğin içine yazılan yazı aynen görüntülenir.

TListBox Bileşeni: Form üzerinde kullanıcılara herhangibir listenin gösterilmesi veya bu listeden herhangi bir elemanın seçilmesi istendiği zaman kullanılır. TlistBox form üzerine yerleştirildikten sonra görüntülenecek olan liste elemanları bileşenin Items özelliği içinde yerleştirilir.Uygulama çalışırken de listeye yeni elemanlar girilebilir veya silinebilir.

TcomboBox Bileşeni: TlistBox bileşenine benzer.Yerden kazanmak gerektiğinde veya kullanıcının dikkatini belli bir eleman üzerine çekmek gerektiğinde kullanılır. Burada da Items özelliğine liste elemanları yazılabilir.

Tmemo bileşeni: Bu bileşen ekran üzerinde sınırları belirlenmiş bir alan üzerinde yazı yazılmasını veya değişiklik yapılmasını sağlar. Bu sınıfa ait nesnelerin Lines adında bir özelliği vardır. Bu özellik Tstring tipindedir.

TcheckBox Bileşeni: Bu bileşen daha çok evet-hayır, Var-yok, Kadın-erkek gibi cevabı iki seçenekli olan soruların cevaplamasında kullanıcıların bu bileşen üzerine bir işaret koyması yada boş bırakması ile anlam kazanır.Bileşen üzerinde bir kez tıklandığında bileşen işaretlenmiş olacak tekrar tıklandığında işaret kalkmış olacaktır.

RadioButton bileşeni: Kullanıcıların birden fazla seçenekten yalnız birini işaretlemeleri için kullanılır. Uygulamada minimum iki tane olursa anlam kazanır. Örneğin;mezuniyet(ılk-orta-lise),yabancı dil(İngilizce-İtalyanca-Fransızca) gibi çok seçenekli özelliklerde kullanılır.

TradioGroup bileşeni: Tradiobutton bileşenleri bir araya gelerek daha pratik biçimde kullanılabilmeleri için geliştirilmiştir. Bu bileşeninde Items adında bir özelliği vardır. Bu özellik Tstring tipindedir. Her satıra bir bilgi gelecek şekilde secenekler yazılabilir. Items içine yazılan her bilgi sıfırdan başlayarak numarandırılmaktadır.

TgroupBox bileşeni: Form üzerine yerleştirilen bileşenleri kullanıcıya daha derli-toplu vermek için kullanılır. Özellikleri kısıtlıdır. Çok büyük uygulamalarda hafızanın optimize edilmesi için kullanılır.

Tpanel bileşeni: TgroupBox bileşeni gibi kullanılabilir. Ancak bunların olanakları daha geniştir.Uygulama geliştirilirken bileşenleri form üzerine direkt yerleştirmektense, paneller üzerine yerleştirilmesi daha sonraki düzeltmelerde büyük kolaylık sağlar.

TscrollBar bileşeni: Bu bileşende maksimum ve minumum özelliklere bir değer atanır. Daha sonra forma eklenen scrollbar üzerindeki düğme sağa ve sola hareket ettirilir.Bu sırada OnChance olayı oluşmaktadır. Bu noktada TscrollBar bileşeninin Position özelliği orantılı olarak bize min ve max arasında hangi noktada olduğumuzu gösterir.

TmainMenu bileşeni: Uygulamalarda kullanılan menuleri oluşturmak için kullanılır. Form üzerine Tmainmenu bileşeni yerleştirilir.Üzerinde çift tıklanarak menü tasarımcısı açılır. Menü ifadeleri yerleştirilir.

4- PROJE YÖNETİCİSİ VE PROJE GRUPLARI

4.1- Proje Yöneticisi (Project Maneger):

Geliştirilecek uygulamalar birden fazla dosyadan oluşacaktır.En azından modüllerin saklandığı pascal dosyaları,form dosyaları,çalışabilir dosya v.b. tüm bunlar proje olarak adlandırılır.Projeler büyüdükçe bunların sayısı ve kapsamıda büyüyecektir.Programcı için dosyaları seçmek ve ilave yapmak birinden diğerine geçmek gibi işlemler zorlaşacaktır.

Delphi bu amaçla proje yöneticisini devreye sokmuştur. Bu modül projeyi oluşturan kaynakların listesini ağaç şeklinde gösterir. Bu ağaç üzerinde istenen dosya tıklanarak geçiş yapılabilir yani üzerinde çalışılabilir.

Proje yönetisinin kullanımı için Ana menüden, View seçeneği seçilir sonra Project Maneger seçilir.Karşımıza 2 bölümden oluşan (1. projeleri tanıtan ağaç yapı.2.bu proje dosyalarının diskteki yeri) ekran gelir. Burada ilgili birim seçilip tıklanarak işlemler yürütülür.

4.2- Proje Grubu:

Yönetici Delphinin önceki sürümlerinde de vardır. Ancak Delphi 5'te daha farklı hale getirildi. Ayrıca proje grubu oluşturuldu. Proje grubu büyük uygulamalarda işe yaramaktadır.

Proje grubu ile delphi'de birden fazla proje açmak ve bunları takip etme mümkündür.Proje yöneticisi ile farklı projeleri görmek ve seçmek mümkündür.

Proje grubuna yeni bir proje eklemek için; Project yöneticisi ekrenından New seçilir. Gelen pencereden Application simgesi tıklanır. Yeni proje eklenmiş olacaktır.

Proje Grubu Kaynak Kodu; Proje grubunun kendiside disk üzerinde bir dosyadır. Bunun da kayıt edilmesi gerekir. Kayıt edilmesi için Proje yöneticisi üzerinde iken sağ tıklanır gelen menüden save project group yada save project group as seçilir**. .BPG** uzantılı olarak kayıt edilmiş olur.

Proje grubu aslında birden fazla projenin birbiri ile bağlantılı olduğunu gösteren bir konfigürasyon dosyasıdır. İçeriğini görmek için Proje yöneticisinde iken sağ tıklanır gelen menüden View Project Group Source seçeneği seçilir. Kodlar görüntülenmiş olur.

4.3- Batch File Operasyonları:

Delphi uygulamalarında bazen belli işlerin otomatik olarak yaptırılmasını isteyebiliriz. Yani toplu komut işleyen dosyalar oluşturabiliriz.

Text dosyası oluşturmak için; Ana menüden **File** seçeneği, **New** seçeneği, **Text** seçenekleri kullanılır. Karşımıza gelen ekranda istenilen text dosyası oluşturulabilir ve **.TXT** uzantısı ile saklanır.

Batch dosyası oluşturmak için;

1. Ana menü, **View** seçeneğinden, **Project Manager** ile proje aktif hale getirilir.
2. File menüsünden, **New** ve sonra **Batch File** seçenekleri girilir.
3. Proje yöneticisi tarafından 2. bir proje oluşturulmuştur.
4. Oluşturulan yeni proje üzerinde sağ tıklanır.
5. Gelen menüden **Edit/Option** seçilir.
6. **Batch File Option** ekranı gelir.Bu ekranda komutlar yazılır.
7. Dosya oluturulmuştur.

Batch Dosyasını Çalıştırmak için;

1. **View** menüsünden, **Project Manager** seçilir.
2. Burada **Project1** üzerinde tıklanır, **Active** yapılır.
3. Ana menüden,Project seçilir ve **Build All Projects** seçenekleri seçilir.
4. Batch dosyasının çalışıp görevini yerine getirdiği görülür.

4.4- Build Sooner ve Build Later Seçenekleri:

Delphi yukardan aşağıya ağaç üzerinde bulundukları sırada derler. Programcı hangi projenin önce derlenmesini istiyorsa ağaç üzerinde yukarıya alması gerekir. Projeyi ağaç üzerinde yukarıya almak için ; yukarı alınacak proje üzerinde iken sağ tıklanır, gelen menüden **Build Sooner** seçeneği seçilir. Böylece proje yukarı çekilmiş olur. Ters işlemi için yani projeyi ağaç üzerinde aşağıya almak için proje üzerinde iken sağ tuş tıklanır, gelen menüden **Build Later** seçilir.

Proje yöneticisi ister büyük ister küçük uygulamalarda kullanılsın programcılar için vazgeçilmeyecek çözümlerden biridir. Özellikle birbiri ile ilgili çalışan birden fazla .EXE nin kontrol edilmesi yada oluşturulan EXE'ler ile birtakım DLL'lerin oluşturulması sırasında daha kolay testlerin yapılabilmesi proje yöneticisi ile rahatlıkla kontrol edilebilmektedir.

Delphi 5'de proje yöneticisindeki en önemli yeniliklerden birisi Windows gezgini ile birlikte çalışmasıdır.Diğer bir yenilikte üst tarafta tüm projelerin listesini veren bir TcomboBox bileşenidir.

4.5- MDI ve SDI Uygulamaları:

MDI (Multiple Documant Interface) uygulamalarda, bir ana ve bu ana pencere içinde açılabilen birçok çocuk pencere vardır. Çocuk pencereler ana pencerenin sınırları dışına çıkamazlar.Onun içinde hareket ettirilebilirler.MDI uygulamalarda çocuk pencereler ekranı kaplayabilir,simge durumuna küçültülebilir, kendi aralarında düzenlenebilir.MDI uygulamalar iki yada daha fazla formdan oluşur.MDI uygulamaları geliştirmek için oluşturulan formun **FormStyle** özelliği ile uğraşmak gerekir.Bu uygulamalara örnek Excel ve Word programlarıdır.

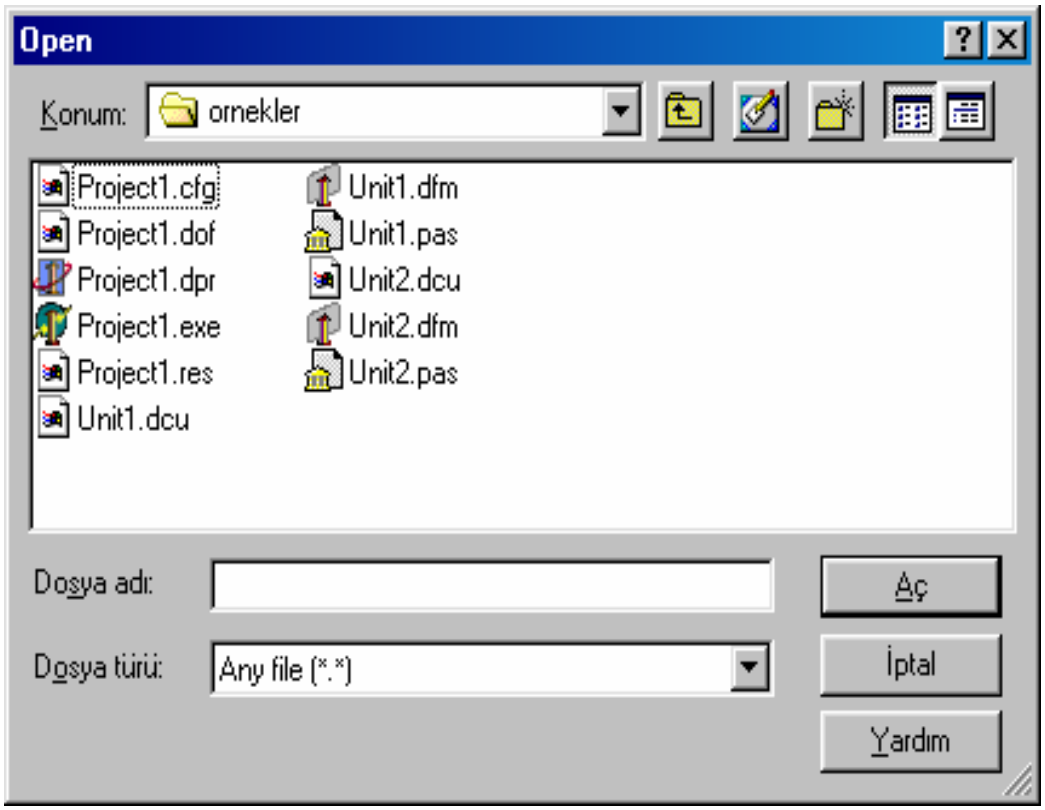
SDI (Single Documant Interface) uygulamalar tek pencereden oluşan uygulamalardır. Ancak SDI uygulamalarda birden fazla pencerede bulunabilir. Bunlar ana ve çocuk pencereler değildir. Her pencere kendi başına hareket edebilir. Her biri ekranı kaplayabilir veya simge durumuna küçültülebilir. Ancak bu tür uygulamalarda ana pencere vardır ve kapatıldığında tüm pencerelerde kapatılır. Buna örnek olarak Delphi uygulama geliştirme ekranı verilebilir. Dört ayrı ekran vardır. Yalnız ana pencere kapatıldığında pencerelerin tümü birden kapatılır.

Projeleri kayıt etmek: File menüsünden **Save** , **Save As** veya **Save Project As** komutlarında biri ile olur.

Save Project As komutu ile projeye daha önce dahil edilmiş her unit için bir diyalog kutusu gelir.

Projeleri açmak: File menüsünden **Open Project** veya **Reopnen** kullanılır.

Örnek bir proje hazırlandığında aşağıdaki şekilde birden fazla dosya olarak klasöre yerleşir.



Proje sonlandırılması:

Projelerin sonlandırılması için **halt** deyimi kullanılır.

Örnek:

```
...
begin
        halt
end;
....
```

5- DELPHI ve FORMLAR

Formlar Delphi'de en önemli nesnelerinden biridir. Geliştirilen her uygulama formlar üzerine oturtulmuştur.

5.1- Formlar Ve Özellikleri:

Formlar Delphi'nin en önemli bileşenleridir. Form işlemleri 3 bölümde toplanabilir.

1. Özellikler (Properties)
2. Olaylar (Event)
3. Formlara uygulanabilecek metotlar

Form kapatıldığında ve projeden çıkıldığında bellekten tüm değişkenler silinir. Form hakkında bilgi içeren değişkenlere özellikler denir.

Form özelliklerinin çoğu tasarım anında değiştirilebilir. Bazı özellikler vardırki çalışma anında değiştirmek anlamlıdır. Örneğin çalışma anında formu kullanılamaz hale getirmek için kullanılan **ENABLE** özelliğini tasarım anında değiştirmek anlamlı değildir. İçeri yalnızca tasarım anında değişen özellikler **Object Inspector'**da bulunmaz.

Windows ortamında her işlem pencereler yardımı yürütülmektedir. Formlar kendileri bir nesne olmakla beraber içlerinde de bir çok nesne bulundururlar. Delphi'de bir proje başlatıldığında karşımıza boş bir form çıkar. Uygulamanın özelliğine göre kullanıcı isterse birden fazla form ile çalışabilir. Ancak ilk form her açılışta ilk görüntülenendir. Açılış formu değiştirilebilir.

Tasarlama anında formu oluşturan nesneleri ve özellikleri rahatça görebiliriz ve değiştirebiliriz. Ancak bazı özellikler vardır ki tasarım anında görülmez, uygulama anında işleme girer.

5.2- Birden Fazla Bileşenin Büyüklüğünü Ayarlamak:

Form üzerinde bulunan birden fazla bileşenin büyüklüğünü ayarlamak için;

1. Büyüklüğü ayarlanacak bileşenler seçilir,
2. Bileşenlerden biri üzerinde iken sağ tuş tıklanır,
3. Gelen menüden "**size**" seçilir,
4. iki sütundan oluşan (width,height) menü gelir,

   En ve boy ayarlaması yapılır.Bu menüde bulunan seçenekler;

   No Chance: Bileşenin büyüklüğünü değiştirme

   Shrink To Smallest: Seçilen bileşenin enini/boyunu içlerinden en küçük olana göre ayarla

   Grow To Largest: Seçilen bileşenlerin enini/boyunu içlerinden en büyük olana göre ayarla

   Width/Height: Seçilen bileşenlerin enini/boyunu Tedit bileşeni içindeki rakam eşit olacak şekilde ayarla

5. seçilen özelliğe göre bileşenler boyutlandırılır.

5.3- Birden Fazla Bileşenin Hizalarının Ayarlanması:

Form üzerindeki bileşenleri hizalanması için;

1. Hizalanacak bileşenler seçilir,
2. Bileşenlerden biri üzerinde iken sağ tuş tıklanır,
3. Gelen menüden "**Align**" seçilir,
4. İki sütundan oluşan (Horizontal/Vertical) menü gelir,
   Bu menüde bulunan seçenekler;

   No Chance: Bileşenlerin hizalarını değiştirme
   Left Sides: Bileşenleri sol kenarları aynı hizada olacak şekilde ayarla
   Centers: Bileşenlerin merkezlerini aynı hizaya getir
   Right Sides: Bileşenleri sağ kenarları aynı hizada olacak şekilde ayarla
   Space Equally: Bileşenleri aralarında eşit boşluk olacak şekilde hizala
   Center in Window: Bileşenlerin tümünü formun ortasına gelecek şekilde hizala
   Tops: Bileşenleri üst kenarları aynı hizada olacak şekilde ayarla
   Bottoms: Bileşenleri alt kenarları aynı hizada olacak şekilde ayarla

5. Seçilen özelliğe göre bileşenler hizalanır.

5.4- Nesneler Ve Formlar:

Her nesnenin bir takım özelliği vardır ve bir takım görevleri yerine getirirler. Nesneler tek olmasına rağmen sınıflar (Class) genel anlam taşır. Örneğin Ayşe'nin arabası nesne, arabalar sınıf oluşturur.

Sınıf (Class) : aynı özellik ve göreve sahip nesnelerden oluşur.
Nesne (Object): Bir sınıfa ait bir tek örnektir.

Delphi'de formlar **Tform** adlı bir sınıfa ait nesnelerdir.Bunlar bir çok özelliklere sahiptir. Ayrıca formların başka form ve nesneler ile etkileşime geçmesini sağlayacak **metotlar (procedur ve fonksiyonlar)** vardır.

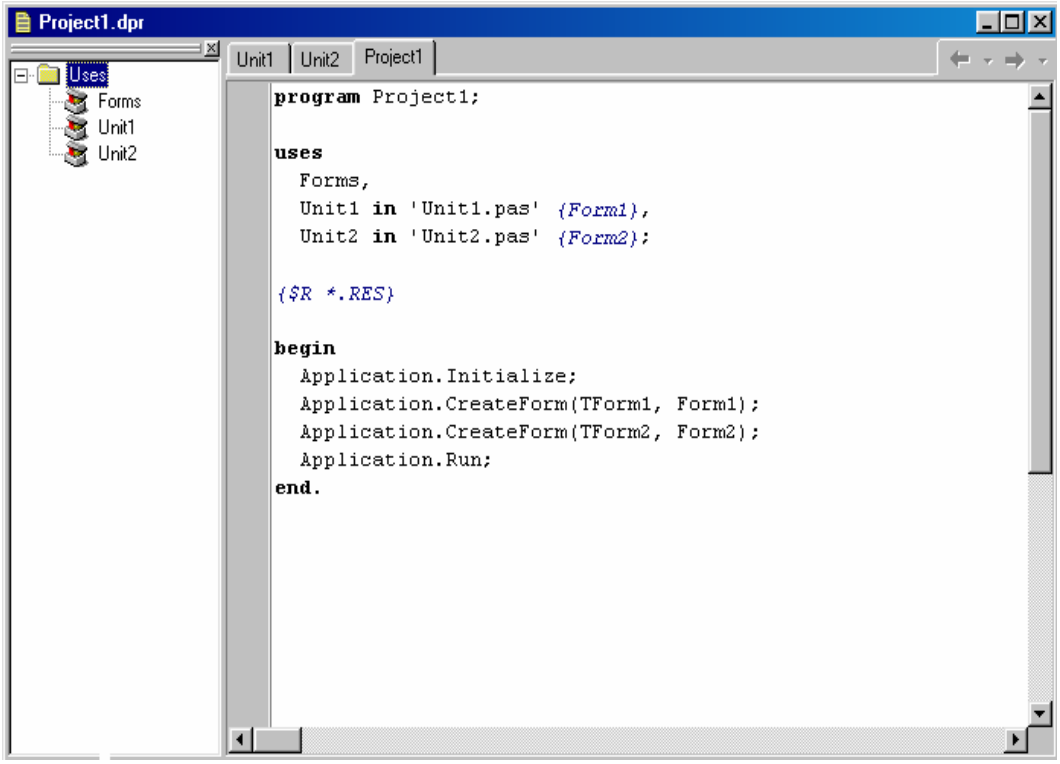
5.5- Uygulamalarda Birden Fazla Form Kullanmak:

Delphi uygulamalarında birden fazla form kullanılır. Formlar için otomatik olarak oluşturulan her bir modüle ilgili kodları bulundurur. İlk açılan form otomatik olarak default formdur. Kullanıcı bunu isterse değiştirebilir. Bu işlem için;

1. Ana pencerede bulunan **Project** menüsü secilir,
2. Bu menüden **Properties** seçilir,
3. Gelen pencereden açılış formu belirlenir.

Formlara butonlar koyarak birinden diğerine geçiş sağlanabilir. Bu işlem için ilgili butonun procedurü aşağıdaki şekilde düzenlenir.

```
Begin
   Formn.Show;
End;
```



5.6- Model Ve Modeless Pencereler:

Bazı durumlarda bir formdan başka bir form çağrıldığında ,çağırdığımız form kapanmadan diğerine geçilmesi istenmez.Bu tür formlar **Modal** olarak adlandırılır. Formun modal kabul edilmesi için iki işlem yapılması gerekir. Bunlar;

1. **Modal Form** çağrılırken prosedürü aşağıdaki şekilde ifade edilir.

```
Begin
   Form3.ShowModal;
End;
```

2. Kapat için seçilen buton prosedürü aşağıdaki şekilde ifade edilir.

```
Begin
   close;
End;
```

Model Pencereler: Üzerinde işlem tamamlanmadan başka pencereye geçiş izni vermeyenlerdir.

Modeless Pencereler: Üzerindeki işlemler tamamlanmadan diğer pencerelere geçiş izni verenlerdir.

Denetim Düğmesi ve Menüsü

Formun sol üst köşedeki denetim menüsüde simge durumuna küçültme ,ekranı kaplama işlemi ile paraleldir.

Denetim menüsünün proje de bulunması istenmiyorsa Object Inspector'dan **BordeIcons** alt özelliği olan **bisystemmenü** özelliği **false** yapılır.Aynı işlem kod yazarakta elde edilebilir.

```
Begin
        BordeIcons:=BorderIcons+[bisystemmenü];
End;
```

5.7- Çalışma Anında Formun Boyutlarını Fare İle Değiştirmek :

Proje çalışırken formun boyutlarını değiştirme işinin iptal edilmesi isteniyorsa özelliklerde, **Borderstyle** seçeneği düzenlenir. Bunun alt menüleri;

Bssizeable: Default olan değerdir.
Bsdialog:Küçült,ekranı kapla düğmelerinin maxsimize minimize özelliğine bakılmaksızın iptal eder.
Bsnone:Çalışma anında formun etrafı çizgisiz olsun isteniyorsa, hem düğmeler hem başlık iptal edilir.
Bssingle: Formun boyutları çalışma anında değişsin istenmiyorsa kullanılır.
Bssizetoolwine: Formda başlık çubuğu yüksekliği azalır.Ekranı kapla,simge durumu düğmeleri kaldırır.
Bstoolwindow: Çalışma anında boyutların değiştirilmesi istenmiyorsa kullanılır.

5.8- Form Başlığı Ve Form Adı :

Object Inspecter'da **Caption** ile eş zamanlı form başlığı yazılır. Çalışma anında form başlığında değişiklik yapılabilir. Aşağıdaki kod ile çalışma anında form üzerine tıklama ile başlık değişir.

```
Begin
        Form1.caption:='yeni başlık';
End;
```

Proje içinde geçerli olan proje adı vardır. Bunu değiştirmek için form özelliklerinde **Name** bilgisi değiştirilir.

5.9- Formun Boyutlarını Belirlemek :

Boyutlar **width** ve **hight** değişkenlerinde tutulur. 800x600 modunda çalışırken ,açılan formun genişliği 544 ,yüksekliği 375 pixeldir. Bu değerlerde değişiklik yapılabileceği gibi kod ile de değişiklik yapılabilir.

```
Begin
        Form1.width:=Form1.width-20;
        Form1.hight:=Form1.hight-20;
End;
```

Bu iki özellikten başka **ClientWidth** ve **ClientHight** özellikleride vardır. Burada net boyutlar saklanır. Hight özelliğinden başlık çubuğu ve kenar çizgisi çıkarılırsa ClientHight elde edilir. Bu ise Hight özelliğinden yaklaşık 8 pixel küçüktür.

5.10- Forma Kaydırma Çubukları Eklemek :

Nesnelerin hepsi görünmüyorsa kaydırma çubukları eklenebilir.Bunun için;
Autoscroll özelliği true veya false yapılır. (false; kaydırma çubuklarının kendiliğinden devreye girmesini önler.) Kaydırma çubuklarında değişiklik için **Horzscrollbar**, **Vertscrollbar** özellikleri kullanılır.

5.11- Formun Masa Üstünde Yerini Belirleme :

Formun ekrandaki konumu left, top özellikleri ile belirlenir. Form ekranı kapladığında left ve top 0 (sıfır) dır.

5.12- Font Kullanımı :

Yazım için defoult değer 8 punto ve font Ms sans serif dir.Font özellikleri için 7 alt seçeneğinden biri kullanılabilir. Bunlar;
Charset,color,height,name,pitch,size,style'dir.
Charset: Font için dil veya karakter kümesi seçimi yapılır.
Color: Forma yazılacak bilgi rengi.
Hight: Satır yüksekliği belirlenir.
Size: Punto değişikliği yapılır.
Picth: Fpdefoult ifadesini içeriyorsa hight ve size birbirini etkiler.Tersi isteniyorsa kullanılır. Fpfixed değeri aktarılır.
Name: Fontu belirler.
Style: Fontun altında olup fsbold,fsunderline,fsstrikeout seçeneklerini bulundurur.
Zemin rengi seçimi color seçeneği ile gerçekleşir.

5.13- Fare Şeklini Değiştirmek :

Bu işlem için özelliklerden **cursor** seçeneği kullanılır. Burada 17 farklı değer vardır. **Crdefoult** sabitliği temsil eder.

5.14- Formları Kullanılamaz Hale Getirmek :

Bu işlem için forma ait **Enable** özelliği kullanılır. Bu durumda true,false değerinden biri alınır. Durum false ise form üzerinde işlem yapılamaz.

5.15- Hint Özelliği :

Buton üzerinde iken açıklama yaptırtmak için kullanılır. Bunu için sırayla iki işlem gerçekleştirilir.

1) Showhint özelliğine true aktarılır.

2) Açıklama metni girilir.

5.16- Active Control Özelliği :

Nesnelerin sıraları taborder ile belli olur. 1.nesne 0, 2.nesne 1, 3.nesne 2 v.d. değerini alır. Proje ilk çalıştırıldığında cursor enküçük taborder değerine sahip olan nesneye konumlanır. **Activecontrol** 'da sırayla bunlar görüntülenir. Burada hangisi seçilirse ilk o aktif olur.

5.17- Nesneleri Ön Ve Arka Plana Almak :

Forma eklenen nesnelerin bazıları ön bazıları arka plana atılabilir. Gerektikçe kullanılabilir.Geriye atmak için **sendtoback**, öne almak için **bringtofront** deyimleri kullanılır.

```
Begin
Memo1.sendtoback;
End;
```

```
Begin
Memo1.bringtofront;
End;
```

5.18- Formlara Simge Durumuna Küçültme Ve Ekranı Kapla Özelliği Verme :

Object Inspector, **BorderIcons** özelliğinde alt menünden true false değerlerini değiştirmekle olur. Aynı işlem program kodu yazarakta gerçekleştirilebilir.

Örnek:

```
Begin
        Form1BorderIcons:=Form1BorderIcons-[Biminimize];
End;
```

```
Begin
        Form1BorderIcons:=Form1BorderIcons+[Biminimize];
        Form1BorderIcons:=Form1BorderIcons+[Bimaximize];
End;
```

5.19- Formların Aktif Ve Pasif Yapılması :

Projede kullanılan tek bir form ise bellekle ilgili herhangibir sorun çıkmaz. Proje sonlandırıldığında form bellekten silinir.

Projede birden fazla form kullanılabilir. Formu ekranda görüntülemek için **Show** metodu kullanılır. Formlar show metodu ile ekrana gelirken activite olayı oluşur. Formun pasif duruma getirilmesi ise deactive olayını oluşturur.

5.20- Formları Bellekten Çıkarma :

Tek formlu proje çalıştırıldığında projenin formu DPR uzantılı dosyadaki createform() metodu ile belleğe yüklenir. Bu arada hem show hemde active olayları oluşur.Proje sona erdirildiğinde form hem kapatılır,hemde bellekten silinir.Bu arada oluşan olay Destroy olayıdır.

Formun kapatılması sırasında Onclose olayı oluşur.Close ile kapatılan form show metodu ile tekrar geri getirilebilir.

5.21- Çalışma Anında Formların Yediden Boyutlandırılması :

Çalışma anında formların genişlik ve yüksekliğinde değişiklik yapılabilir. Bu işlemde resize olayı oluşur. Çalışma anında boyutlarda değişiklik yapılıp yapılamayacağı formlara ait **Borderstyle** değişkeninin içine bağlıdır.

Formlarla İlgili Olaylar (EVENTS)

Form ve burada kullanılan nesneler için tanımlı olan birkaç olay mutlaka vardır.
Formlar için tanımlanmış olaylar **Object Inspector** penceresinden **Events** sekmesinde görülür. Seçili nesneye göre tanımlı olaylar listesi farklılık gösterir. Önceden tanımlı olaylardan biri meydana geldiğinde yapılmak istenen işlemler **UNIT.Pas** uzantılı dosya içinde yordam içinde hazırlanır.

Örnek: Oncreate,Onclick, Onclose, Onexit v.b. birer olaydır.

Bir Tuşa Basma Olayı

Çalışma anında klavyenin yön ve kontrol tuşları dışında (alt,control,shift) herhangibir tuşa basıldığında Onkeypress olayı oluşur. Hazırlanan yordama basılan tusun kendisi parametre olarak

gönderilir. Keypress olayının gerçekleşebilmesi için form üzerinde düğme ve metin kutusu gibi nesneler olmamalıdır. Keypress'i temsil eden yordam key adlı parametre alır.

Örnek:

```
Begin
        Canvas.textout(40,40,key+' tusuna basıldı');
End;
```

Örnek çalıştırıldığında A tusuna basılırsa; 'A tusuna basıldı' mesajı görüntülenir.

Keydown Keyup Olayları

Keydown tuşa basıldığı andaki olayı, Keyup tuş serbest bırakıldığı andaki olayı gerçekleştirir.Bu olaylar dışardan 3. bir parametre almaktadır.

```
Procedure Tform1.FormkeyDown{sender:Tobject;
                        Var key:word;
                         Shift:Tshiftstate};
Begin
Showmessage(chr(key)+'tusuna basıldı');.
End;
```

Basılan tuşun **ASCII** karşılığı key değişkenine aktarılır. 3.parametreye ise bu olay oluştuğu zaman shift, ctr,alt tuşlarının birinin basılı olup olmadığı konusunda bilgi aktarılır.

Keydown sırasında shift tuşu basılı ise shift değilse boşluk içerir.
Ctr basılı ise ssctr değilse boşluk
Alt basılı ise ssalt değilse boşluk içerir.

```
Procedure Tform1.FormkeyDown{sender:Tobject;
        Var key:word; Shift:Tshiftstate};
Begin
If ((shift=[ssalt]) and (key=VK_F1)) then messagedlg('alt+F1  tusuna
bastınız', mtinformation,[mbOK],0);
End;
```

NOT: F1'e basıldığında key parametresi VK_F1 değerini alır.
F2'ye basıldığında key parametresi VK_F2 değerini alır.
.....
F10'a basıldığında key parametresi VK_F10 değerini alır.

Mouse Down Olayı

Çalışma anında form üzerinde iken farenin herhangi bir tuşuna basılması halinde oluşur. Bu olayı temsil eden Formmousedown yordamıdır. Bu yordam 4 parametreye sahiptir.

```
Procedure Tform1.FormkeyDown{sender:Tobject;
        Button:Tmousebutton;
                Shift:Tshiftstate;
                        X,Y:integer);
Begin
```

Button adlı 2.parametrede mousedown olayının oluştuğu sırada farenin hangi tuşuna basıldığı konusunda bilgi bulunur. Eğer çalışma anında fare işareti formun üzerinde iken, farenin sol tuşuna basılarak olay oluşturulursa button parametresine mbleft, sağ ile mbright, ikisine birden basıldığında mbmiddle değeri aktarılır.

```
Procedure Tform1.FormkeyDown{sender:Tobject;
        Button:Tmousebutton;
                Shift:Tshiftstate;
                        X,Y:integer);
Begin
if button= mbleft then showmessage('farenin soluna basıldı');
End;
```

Shift adlı 3.parametreye, mousedown olayı oluştuğunda shift,alt,ctr tuşlarından birinin basılı olup olmadığının bilgisi tasınır. Burada da yine ssalt, ssshift, ssctr değerleri aktarılır.

X,Y adlı parametrelere olay oluştuğunda farenin form üzerindeki yeri pixel olarak aktarılır.Yer değiştikçe kendiliğinden güncellenir

MoveUp Olayı

Farenin tuşundan el kalkınca gerçekleşecek olay içindir.

MouseMove Olayı

Farenin form üzerinde yeri değişince olay gerçekleşir.Tıklamaya gerek yoktur.

FORMLARA UYGULANAN METOTLAR

Show Ve ShowModal Metotları

Projede birden fazla form var ise bunların görüntülenmesi gerektiğinde show metodu kullanılır.

```
Begin
        Form2.show;
End;
```

Önceden pasif durumda olan veya **hide** ile gizlenen formlar **show** ile aktifleşir.

Birden fazla form ile çalışırken biri kapatılmadan diğerine geçilebilir. Bunu önlemek için **Show Modal** metodu kullanılır.

```
Begin
        Form2.showmodal;
End;
```

Showmodal metodu ile ekrana getirilen form kapatılmadan o proje dahilinde o sırada ekranda olan başka bir forma geçiş yapmak mümkün değildir. Ancak showmodal metodu ile ekrana gelen bir form var ise başka bir formu show veya showmodal metodu ile ekrana getirilebilir.

Refresh Metodu

Textout() veya başka bir teknikle formun üzerine yazılanları veya çizilenleri silmek için **refresh** metodundan yararlanılır. Bu metotda formdaki nesneler etkilenmez. Temizlenecek form aktif değilse, refresh ile formun adıda belirtilir.

```
Begin
Form1.ReFresh;
End;
```

Close Hide Free Metotları

Close metodu ile ekranda olan formlar kapatılmaktadır.Projede birden fazla form var ise close ile form adı da belirtilmelidir. Kapatılan form başlangıç formu ise projeyide sonlandırır.

```
Begin
   Form2.close;
End;
```

Hide Metodu

Formları gizler(masa üstünden kaldırır) Bir formun tamamı için **HIDE** kullanılabileceği gibi içindeki bir nesne içinde kullanılabilir.

```
Begin
   Form2.Hide;
End;
```

Close veya **hide** metodu ile kapatılan formlar bellekte yer işgal etmeye devam ederler.Eğer bellekteki yeride boşaltmak istiyorsak formları **free** metodu ile kapatmalıyız.Free ile silinen formu show ile ekrana getirmek istersek programın çalışması kırılır.Eğer tekrar kullanmak gerekirse CreateFormu() ile tekrar oluşturulmalıdır.

```
Begin
   Form2.close;
End;
```

```
Begin
   Form2.Free;
End;
```

```
Begin
   Application.CreateForm(Tform2,Form2);
End;
```

LineTo Ve MoveTo Metodu:

Lineto ile belirtilen formun yüzeyine çizgi çizilir. Genel ifadesi;

Lineto(x,y) şeklindedir.

ışardan iki parametre alır. Belirtilen koordinat sonlandırma koordinatıdır. Başlangıçta sol üst köşeden başlar.

Bitiş değeri x.,y pixel cinsinden verilir. Başlangıç sol üst köşe istenmiyorsa **moveto()** ile çizim yapılır.
Çizgi kalınlığını değiştirmek için (default 1 pixel) pen nesnesinin width özelliği kullanılır.

Renk değiştirmek için **pen** nesnesinin **color** özelliği kullanılır.Bu işlemde renk ifadesinin önüne "cl" harfleri gelir.

Ayrıca **mouseup** ve **mousedown**'u kullanarakFormun Üzerine çizgi çizilebilir.

Çizgidışında serbest çizim yapmak için **mousedown** ve **mouse move** olayları kullanılır.

Elips Metodu

İstenilen forma istenilen çap,renk ve kalınlıkta çember ve elipsler çizer.

Elipse(başlamaX,başlamaY,Yükseklik,Genişlik)

Regtangle Metodu

İstenilen boyutta kare ve dikdörtgen çizilmesini sağlar.

Regtangle(başlamaX,başlamaY,genişlik,Yükseklik)

Form Özellikleri

Formlarda kullanılan özellikleri aşağıdaki şekilde listeleyebiliriz.
BorderStyle : Programın çalışması sırasında formun davranışı belirlenir.

Caption :
ClientHeight:
ClientWidth:
Color:
Constraints:
Ctl3D:
Cursor:
DockSite:
DragKind:
Enabled:
Font:
FormStyle:
Height:
HelpContext:
Hint:
HorzScrollBar:
Icon:
KeyPriew:
Left:
Menü:
Name:
ObjectMenuItem:
PixelsPerInch:
PopUpMenu:
Position:
PrintScale:
Scaled:
ShowHint:
Tag:
Top:
VertScrollBar:
Visible:
Width:
WindowMenu:
WindowState:

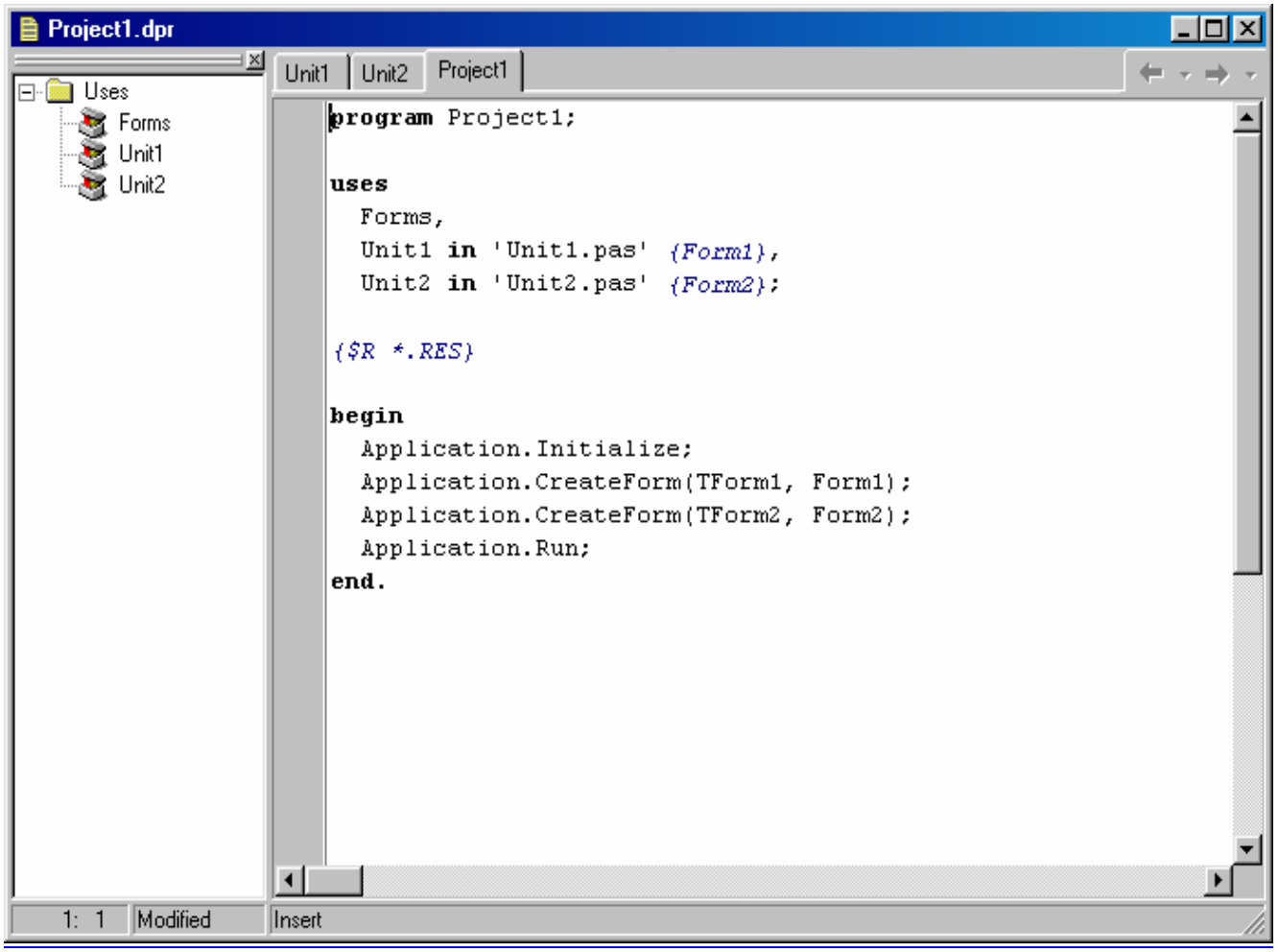
6- PROGRAMLAMAYA GİRİŞ

Delphi'de program kodları .**DPR** uzantısı ile saklanır. Program kodunu görmek için **Project** menüsünden **View Source** seçeneği kullanılır.Bu seçenek kullanıldığında ekrana program kodu içeren project1.dpr penceresi gelir. Bu pencere iki kısımdan oluşur. Code explorer adı verilen soldaki kısım kodları içeren unit, project1 sekmelerinden oluşan sağdaki kısım.

Code explorer penceresi kapatılabilir, yeri değiştirilebilir, büyüklüğü değiştirilebilir ve eski haline döndürülebilir.

Delphi tarafından projeler için hazırlanan kod satırı 10 satırdır.

Delphi projelerinde uses deyimi ile forms adındaki hazır unit, proje hazırlandığı zaman unit1'e dahil edilmekte harddiskte de **.Pas** uzantısı ile kayıt edilmektedir.



File menüsünden **New Application** komutu ile yeni bir proje başlatılabilir.

Kod içeren unit1'den forma geçmek için veya tersi için view menüsünden **toggle form/unit** seçeneği kullanılır. Aynı işlem için view menüsünden **project manager** komutu ile project manager ekranı gelir. Burada unit'e çift tıklama yapılırsa kod ekranı , form'a çift tıklanırsa form ekranı gelir.

Unitlerde

interface adlı kısımda uses deyimi ile unitte kullanılacak hazır unitler belirlenmektedir.

Type deyimi ile başlatılan bölümde yeni tipler tanımlanmaktadır.

Delphi'de formları tanımlamak için kullanılan nesne tipi **Tform** olarak adlandırılır.

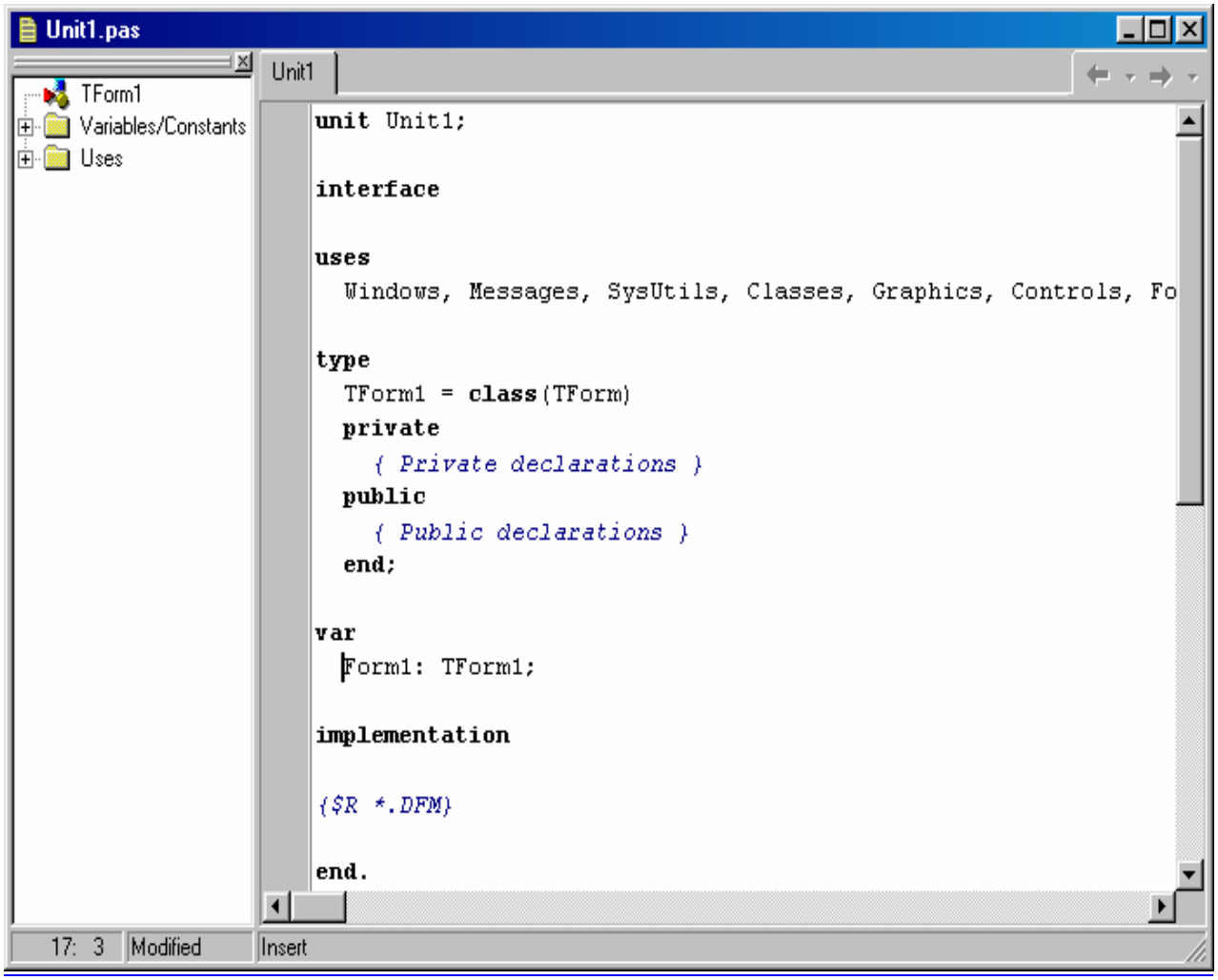
```
Tform1 = class(Tform)
```

Var deyimi ile unite dahilinde kullanılacak olan değişkenler tanımlanır.

```
Form1 = Tform1;
```

Implementation deyimi ile unit'in asıl program satırına geçilir.

Unit'in son satırı **end**. Deyimi ile biter.



Örnek unit görünümü

6.1- Projelerde Değişiklik Yapmak Ve Çalıştırmak:

Proje çalıştırıldığında ilk form aktif olur ve belleğe yüklenir.

Olay güdümlü (event-driven) programlama dillerinde çalışma anında herhangi bir olay gerçekleştiği zaman ilk olarak gerçekleşen olay ile ilgili aynı ada sahip yordamın olup olmadığı kontrol edilir ve varsa otomatik olarak işletilir.

Delphi'de projenin çalıştırılması halinde formu temsil eden pencerenin hazırlanıp masaüstüne getirilmesi olayına **Formcreate** adı verilir. Bu esnada aynı isme sahip unit varsa otomatik olarak çalışır.Dolayısı ile unit'e formcreate() adında bir yordam dahil edilirse proje çalıştırılırken bu yordam otomatik olarak çalışır. Açılışta ilk yapılması istenen işler bu yordama yazılabilir.

Form üzerinde iken çift tıklanırsa (ki bu iş menülerden de yapılabilir.) Yordam otomatik olarak açılır.

Form yüzeyine sabit yazı yazdırmak için **textout()** deyimi kullanılır.

Textout(n,m,'ifade')

Nesneler üzerinde işlem yapmaya yönelik kullanılan (textout() v.b.) deyimlere **method** adı verilir.

Form yüzeyine **canvas** adı verilir. Canvas'ı etkileyecek textout() metodu 3 parametre alır.

Canvas.textout(10,15,'bilgisayar');

```
Procedure  Tform .formcreate(sender:Tobject);
Begin
        Show;
        Canvas.textout(20,20,'bilgisayar');

end;
```

Show: Hangi pencerenin çıkış ortamı olarak kullanılacağını gösterir.

Proje çalışmasını sona erdirmek için **Run** menüsünden **Program Reset** seçeneği seçilir.
Yazılan metne özellik kazandırılmak istenirse **Canvas** İfadesi seçenekleri ekrana getirir.
Kazandırılacak özellikle ilgili seçenek buradan seçilir.
Kazandırılacak özelliklerden bazıları şunlardır:

Destroy
Arc
Brushcopy
Chord
Copyrect
Draw
Drawfocusrect.....

InputBox() Fonksiyonu :

Inputbox() aktif form veya pencereden bağımsız bir diyalog kutusu içinde dışardan bilgi girişini sağlar.

Kullanım şekli:

```
Değişken:=Inputbox(diyalog kutusu başlığı,metin kutusu başlığı,varsayılan bilgi);
```

Örnek:

```
Ad:=inputbox('bilgi girişi','adınız','');
```

Showmessage() Deyimi :

Showmessage() deyimi ile istenilen bilgi diyalog kutusu içinde görüntülenir. Ekrana yazılır.

Kullanım şekli:

```
Showmessage(Değişken)
```

Örnek:

```
Showmessage(adres)
```

- Dışardan alınan bilginin atandığı değişken unit1.pas içinde var bildiri deyimi ile tanımlanmalıdır.

Messagebox() kullanımı :

Bu metot ile ekrana diyalog kutusu içinde mesaj gelir ve seçim yapma imkanı sunar.

Messagebox application nesnesine yönelik olarak iş yapar.Bu nedenle metotdan önce application nesnesinin ismi verilir.Bunun için (unitte başka yordam hazırlamak için);

penceresinden yararlanılır. Burada **events** sekmesinde **onclick** seçeneğinde çift tıklanırsa **formclick** adlı yordam unit'te otomatik olarak aşağıdaki şekilde hazırlanır.

```
Procedure Tform1.formclick(sender:Tobject);
Begin
..............
end;
```

Yordamın içine yazılanlar form çalıştırıldığında otomatik olarak işletilir

Örnek:

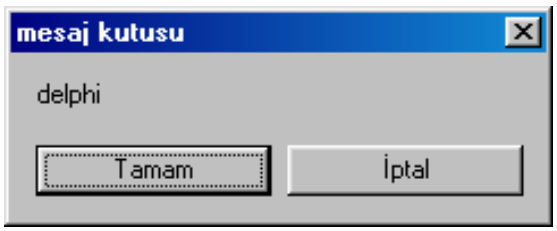
```
Procedure Tform1.formclick(sender:Tobject);
Begin
Application.messagebox('delphi','mesaj kutusu',
                        mb_okcancel + mb_defbotton1);
 end;
```

Burada üç adet parametre kullanılmaktadır.

1. parametre : mesajın kendisi
2. parametre : diyalog kutusunun başlığı
3. parametre : diyalog kutusunda bulunacak düğmelerdir.

Örnek:

```
unit Unit1;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Fo
type
  TForm1 = class(TForm)
    procedure FormClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormClick(Sender: TObject);
var
        tus:integer;
begin
        tus:=application.messagebox('delphi','mesaj kutusu',
                 mb_okcancel+mb_defbutton1);
        if tus=1 then
                showmessage('tamam tıklandı') ;
        if tus=2 then
                showmessage('iptal tıklandı');
end;

end.
```



Diyalog kutusu ekranda iken tamam tıklanırsa **messagebox()** metodu 1 değerini ,İptal tıklanırsa 2 değerini döndürür. Geriye döndürülen bu değeri bir değişkene aktarmak gerekir.

Aktarılan değişkenide procedure içinde tanımlamak gerekir.

Messagebox() metodu ile kullanılan diyalog kutularına her istenen düğme konamaz.

Program tarafından geri gönderilen değerler aşagıdaki şekildedir.

DÜĞME	DEĞER	İFADE
Ok	1	IDOK
Cancel	2	IDCANCEL
Abort	3	IDABORT
Retry	4	IDRETRY
Ignore	5	IDIGNORE
Yes	6	IDYES
No	7	IDNO

Program Kodu Yazılan Pencerede Yapılan Bazı İşlemler :

İçeriği ekranda görüntülenen unit için 2.kod penceresi açmak gerekirse **view** menüsünde **newedit window** seçeneği kullanılır. Yada kısayol seçeneği ile (sağ tıklama) aynı işler gerçekleştirilir.

Üzerinde çalışılan proje birden fazla unit'e sahip ise her unit kod penceresinde bir sekme ile gösterilir.

Project menüsünde iken **viewsource** komutu ile DPR uzantılı proje dosyasının içeriği görüntülenirse kod penceresinde proje dosyası içeriisinde bir sekme hazırlanır
.

Cod sayfalarını kapatmak için kısa yoldan **close page** seçilir. Cod sayfasında değişiklik istenmiyorsa **readonly** kullanılır.

Unit ve Formları Yazdırmak :

Bu işlem için file menüsünden print seçeneği kullanılır. Aktif olan unit veya form yazdırılır.

KONTROLLER VE NESNELER

Delphi'de programlama da gerekli olan işlemler için önceden hazırlanmış program modülleri vardır. Bu modüllerin her birine **kontrol** veya **bileşen (component)** adı verilir. Yapılması gereken projeye ilgili bileşeni dahil etmektir. Dahil edilen her bileşen nesne özelliği kazanır. Dolayısı ile her türlü işlem yapılır.

penceresinde seçili olan nesne özellikleri hakkında bilgi bulunmaktadır.

Dışardan girişleri **Inputbox()** fonksiyonu , sabit bilgi yazdırma **textout()** fonksiyonu ile gerçekleşir. Ayrıca sabit bilgi yazdırma işlemi **label** ile de gerçekleşir.

Program yazarken kullanılan her nesne veya bileşen **component palette** araç çubuğunda düğme ile temsil edilir. Bu kontroller fonksiyonlarına göre 18 ayrı gruba ayrılmıştır.

Label nesnesinin büyüklüğü ve yeri değiştirilebilir , silinebilir.

Projelerin çalışmasını sonlandırmak için **Run** menüsünden **program reset** kullanılır.

Program Kodu Yazarak Nesnelerin Özelliklerinde Değişiklik Yapmak

Genel değişiklikler için penceresi kullanılır.Aynı fonksiyonlar kod yazarakta gerçekleştirilebilir. Bunun için kod yazılan pencereyi (unit) görüntülemek gerekir. Kod penceresini görüntülemek farklı yollardan sağlanabilir.

1- Form üzerinde iken çift tıklama yapmak.
2- **View** menüsünden toggle Form/unit seçeneğini seçmek.

örnek:

```
Procedure Tpencere.formcreate(sender:Tobject);
Begin
        Label1.caption:='bilgisayar';
End;
End.
```

Bu arada nesne adını yazıp . (nokta) işareti konduktan sonra beklenirse özellikler listelenir.Kod yazılabileceği gibi listeden seçilerekte yapılabilir.

Örnek:

Form başlığı değiştirilsin isteniyorsa ;

```
Pencere.caption:="giriş ekranı";
```

şeklinde kod kullanılabilir.

Metin Kutusu (edit) Nesnesi :

Delphi'de metin kutularına **edit** adı verilir. Component palette ab düğmesi bu işlem için kullanılır.

Dışardan metin girmek için kullanılacak olan edit nesnesinin (metin kutusunun) başlangıç değerinin boşluk olarak değerlendirilmesi gerekir. Metin kutusu nesnelerinin içerdikleri bilgilerde **text özelliği** bulunmaktadır.

Edit nesnesi kullanıldığı sürece otomatik olarak herbiri edit1,edit2,edit3 v.b. şeklinde isimlendirilir.Programcı isterse bu isimleri değiştirebilir.

Edit nesnesi girilirken değişken isimlerinin görünmemesi içinde penceresinde özellikler sütununda text ifadesinin karşılığı boşaltılmalıdır.

Events (olaylar) :

Projenin çalıştırılmasından sonraki her adım (formun yüklenmesi,fonk.tuş kullanımı v.b.) bir olaydır.

Object inspector'dan **events** tıklanırsa olarlar listesi gelir. İncelenebilir, tanımlama yapılabilir. Delphi'de formlar için önceden tanımlanmış 33 olay vardır. Formun otomatik olarak oluşturulması olayına **oncreate** adı verilir.

Örnek: Form çalışırken üzerine tıklandığında "tıklama yaptınız " mesajını versin.

İlgili unite yordam eklememiz gerekir. Yordam adı mesajver () olsun. Bu yordamın program satırındaki show mesaj deyimi ile ekrana mesaj verelim. Unitlere eklenen yordamlar Type bildirim kısmında tanımlanmalıdır. Unitte kullanım aşağıdaki şekildedir.

```
.....
type
  ........
   procedure mesajver(sender:Tobject);
   ....
...........
implementation
..........
procedure Tform.mesajver(sender:Tobject);
begin
        showmessage('tıklama yaptınız');
end;
....
```

Ayrıca event seçeneklerinden onclick karşılığınada mesajver yordam adı yazılmalıdır.

KeyPress Olayı :

Çalışma esnasında klavyenin herhangi bir tuşuna basılması durumunda **onkeypress** olayı gerçekleşir.

Örnek: edit1'den girilen karakter her ne ise entere basıldığı anda ekrana görüntülenir.

```
Procedure Tform.edit1keypress(sender:Tobject;varkey:char);
Begin
        if ord(key)=13 then
        begin
                showmessage(edit1.txt);

        end;
end;
```

Düğmeler ve button kontrolü :

Component palette araç çubuğundaki her düğme bir kontrolü temsil eder. Kullanmak için; button nesnesini temsil eden düğme forma eklenir. Caption özelliğinden button1 sabit mesajı değiştirilebilir.

Örnek:

Caption	Çıkış

Kullanılan ismin önünde veya içinde **&** işareti kullanılabilir. Bu ise sonraki karakteri altı çizili olmasını sağlar.

Özetle; delphi de pencere veya form hazırlamak için kod yazmak yerine New Form kullanılabilir. Form oluşurken ilgili unit hazırlanmakta,form bilgisini içeren DFM uzantılı dosya hard diske kayıt edilmekte,Yeni formdan projeyi haberdar etmek için DPR uzantılı proje dosyasına iki satır eklemektedir.

Aktif bir formdan başka bir forma müdahale edebilmek için söz konusu forma ait unit'in aktif forma ait unit'e uses deyimi ile dahil edilmesi gerekir.

Nesneleri Kilitlemek :

Forma yerleştirilen nesneler edit menüsündeki **lock controls** komutu ile kilitlenir. Formdaki nesnelerin boyları belli yüzde ile büyütülebilir veya küçültülebilir. Bunun için **edit** menüsündekin **scale** seçeneği seçilir.Bu seçim sonucu **scaling factor** metin kutusu gelir.Düşünülen yüzde değeri buraya yazılır.

7- FORMLAR VE FORM ÜZERİNDE BİLEŞENLER

Formlar Ve Üzerine Yerleştirilen Bileşenler Arasındaki İlişkiler Ve *.DFM Dosyalar

Form üzerine bileşen yerleştirildiğinde ,Delphi aslında çalışılan modülle aynı adda ama ***.DFM** uzantılı bir dosyaya, form üzerine yerleştirilen bileşenler hakkında bazı bilgiler yazmaktadır. Yani bu dosya bir anlamda konfigurasyon dosyasıdır.

Form üzerinde sağ tuş tıklanarak gelen menüden View As Text seçeneği seçilirse ***.DFM** uzantılı dosyanın içeriği görüntülenir. Aynı dosyaya file menüsü open seçeneğini kullanarak da ulaşılabilir. Text modundan forma dönmek için tekrar sağ tus tıklanır ve gelen ekrandan view as Form seçeneği seçilmelidir.

***.DFM** uzantılı dosya üzerinde yapılan değişikliklerde forma üzerine aktarılır.

Text Formatında DFM Uzantılı Dosyalar :

DFM uzantılı form dosyaları delphi 5 ile gelen bir özellikten dolayı text dosya olarakta kayıt edilebilir. Sonuçta word gibi bir metin düzenleyici tarafından açılabilir ve hatta üzerinde yapılan değişiklikler delphi'ye otomatik olarak aktarılabilir. Ancak bunun için form dosyasını kayıt etmeden önce form üzerinde sağ tıklanarak gelen menüden **Text DFM** seçeneği seçilmelidir. Aynı seçeneğe **Tools** menüsünün **Environment options** menüsünden **New Forms As Text** onay kutusu işaretlenmiş olmalıdır. Bu özellikte kayıt edilen *.DFM uzantılı dosya word ile açılabilir örneğin caption değiştirilsin ve kayıt edilsin. Delphi açıldığında bu değişiklikleri kabul edecek ve değişiklikleri yapacaktır.

Formalar Üzerinde Gerçekleşen Olaylar :

OnActive: Form aktif olduğu anda oluşur.

OnDeActive: Bu olay üzerinde çalıştığımız formun dışındaki bir forma tıkladığımız zaman , az önce çalıştığımız formun üzerinde devreye girer.

OnCreate: Bu olay form ilk yaratılırken oluşur. Yani form ekranda görünmeden olay oluşur.

OnClose: Bu olay formumuz kapandığı anda oluşur. Ama form hafızadan silinmez. Alanında silinmesi isteniyorsa OnDestroy olayı gerçekleşir.

OnDestroy: Bu olay form hafızada serbest bırakılması esnasında oluşur.

OnShow: Form ekranda görünmeden hemen önce oluşur.

OnPaint: Formun tamamı veya bir bölümü ekranda görüntülenmek üzere iken oluşur.

Formların Kalıtımı :

Proje geliştirme aşamasında formların genel özellikleri aynı olsun istenir yada bu konuya dikkat edilir. Örneğin her formda aynı tanımların yapılması gerekiyor ise bu bir kere yapılır. Diğer formlar bundan kalıtım (inheritance) mantığı ile türetebiliriz.

Bunun için ;

- Aktif proje varsa kapatılır (**File/close all**),
- **File/New** ile yeni bir form oluşturulur,
- Form hazırlanır,
- Form isim**.Pas** adı ile saklanır,
- Form Üzerinde sağ tıklanır,Gelen menüden **Add To Repository** seçilir,
- Gelen ekrandan title kısmına formun adı (isim.pas) yazılı,
- Pages yazılı kısımdaki listeden **Forms** seçeneğine girilir,
- **Ok** basılır
- Yeni olusturulan form kod deposuna eklenmiştir.

Nesne Deposu (Object Repository) :

Kod deposune eklenen formlar çok ise bunları kullanmak zaman alacaktır. Oysa bir çok formda kullanılacak temel özellikleri kendinde barındıran yukardaki gibi bir formu Delphi'nin nesne deposune **(Object Repository)** eklediğimizde , istediğimiz zaman bu depodan temel özellikleri önceden belirlenmiş bir formu alabilir ve üzerine yeni bileşenler yerleştirebilir veya değiştirebiliriz. Böyle çalışma hem hızı artırır hemde işlemlerin kolaylığını artırır.

Nesne deposune eklenen formu kullanmak için;

- Ana menüden **File/Close All** ile açık modüller kapatılır,
- Ana menüden **File/ New Application** seçilir, (uygulama oluşturulur)
- Form üzerinde çift tıklanır;

Frame Bileşenleri :

Frame'ler delphi 5 ile gelen bir özelliktir. Görsel arabirimler yaratırken ;aynı görsel arabirimin birden fazla oluşturulması ve kullanılması gerekebilir. Böyle durumlarda programcı bu tekrarlanan alanları frame olarak (form düzenler gibi) hazırlar. Saklar ve gerektiği zamanda bu tanımlı alanları formlarına yerleştirerek kullanılır.

Frame oluşturmak için;

- yeni bir uygulama başlatılır,
- Ana menüden **File/New Frame** seçilir,
- Frame ile karşılaşılır,(form oluşturma gibi),
- Frame düzenlenir, (nesneler yerleştirilir),
- Saklanır,(normal unit gibi),
- Bileşen paletine yerleştirilir (frame üzerinde sağ tıklanır/**Add To Palette**)
- Gelen menüden ilk alan bileşene verilen ad yazılır;
- Bileşenin hangi sayfada görünmesi isteniyorsa işaretlenir;

Frame kullanmak için;

- Asıl Form üzerinde iken;

- Bileşenler sekmesi **Templates** sekmesine gidilir,
- Buradan, oluşturulan frame forma aktarılır.

Frame üzerinde değişiklik yapmak;

Değişiklik 2 şekilde gerçekleştirilebilir

1. Gerçek form üzerine yerleştirilen frame üzerinde istenilen değişiklik yapılır. Bundan sadece üzerine yerleştiği form etkilenir.

2. Hazırlanıp bağımsız form gibi saklanan frame'in kendisi çağrılır. Düzeltmeler yapılır. Bu durumda o frame'i kullanan tüm formlarda da aynı değişiklik gerçekleşir.

Ancak Frame üzerinde her metot çalışmaz. Örneğin üzerine yerleştirilen buton ile formu kapatmak istersek derleme aşamasında hata verir. Zira **kapat (close)** metodu forma ait bir metotdur frame ait değildir. Yinede frame üzerinden formu kapatmak istersek **close** deyimi yerine aşağıdaki ifade bütününü kullanmak gerekir.

```
Procedure TfisFrame.buttonclick(sender:Tobject);
Var
Frm:Tcomponent;
Begin
Frm:=self.getParentComponent;
While not (frm is form)
  Do form:=Frm.GetParentComponent;
(frm as Tfrm).close;
end;
```

Constraint Özelliği :

Delphi bu özellik ile hem uygulamanın çalışma anında hemde tasarım anında bileşenleri ulaşabileceği **max min** büyüklüklere sınır koymak için kullanılır. Bu özellikte; mak. Genişlik ,yükseklik ve min. Yükseklik ve genişlik vardır.

Anchor Özelliği :

Tüm görsel bileşenlerde olan bir özelliktir. Formlar minumun büyüklüğe sahip olsa bile üzerindeki bazı bileşenlerin görünmesi isteniyorsa kullanılır.

8- EDİT KONTROLÜ VE METİN KUTULARI

MaxLength özelliği : Bir metin kutusuna en fazla 32 KB kadar metin girilebilir.Kullanıcı bunu sınırlayabilir. Bu ise **MaxLength** özelliği ile belirlenir.Varsayım olarak değer 0 dır. Metin kutusuna girilen bilginin uzunluğu **GetTextLen** özelliğinden yararlanılır.

```
Begin
        Edit2.Text:=IntToStr(Edit1.GetTextLen);
End;
```

TabStop Özelliği:

Metin kutularına kullanıcı müdahalesini engellemek için kullanılır. Herhangi bir nedenle edit nesnesinin atlatılması isteniyorsa kullanılır. Varsayım olarak **TabStop** özelliği **True** değerini bulundurur. Özelliklerden bu değer False yapılırsa çalışma anında ilgili edit alanı atlatılmış olur.

PasswordChar Özelliği:

Şifre girişlerinde de metin kutularından yararlanılır. Forma eklenen edit alanı şifre girişi için kullanılacaksa İlgili edit alanının **PasswordChar** özelliği kullanılır. Varsayım olarak ‘ **#0** ‘ karakterlerini içerir. Şifre olarak kullanılacak karakter sayısı ‘0’ rakamı ile yer değiştirilir. Örneğin 5 karakterlik şifre için #5 değerini alır. Varsayım olarak girilen karakterler ne olursa olsun düşey çizgi **‘|’** ile gösterilir. Bu karakteride değiştirmek mümkündür.

```
Procedure Tform1.FormCreate(Sender:Tobject);
Begin
        Edit1.PasswordChar:='*';
        Edit1.text:='';
```

```
Begin
   if edit1.text := 'konya' then showmessage
('bildiniz')
      else showmessage('bilemediniz!');
```

AutoSelect Özelliği:

Daha önce girilmiş bir metin kutusunun üzerinde gidildiğinde metin kutusunun bütün içeriği otomatik olarak seçili duruma getirilmektedir. Bu özelliğe **False** değeri aktarılırsa seçili durum iptal edilmiş olur. (**shift+tab** tuşları ile sonuç kontrol edilebilir.)

AutoSize özelliği:

Formlara eklenen metin kutularının yükseklikleri seçilen font ve puntoya bağlı olarak otomatik olarak belirlenir. Bu özelliğe **False** değeri aktarılırsa bu özellik iptal edilmiş olur.

CharCase özelliği: Delphi varsayım olarak metin kutusuna girilen bilgiyi değiştirmez. Girilen bilgilerin büyük harfe çevrilmesini istiyorsak bu özelliğe **ecuppercase** değeri aktarılır. Girilen bilgilerin küçük harfe çevrilmesini istiyorsak bu özelliğe **eclowercase** değeri aktarılır. (küçük i harfi büyük harfe çevrilirken delphi yanılır.)

ReadOnly Özelliği:

Metin kutusu aracılığı ile ekrandaki bilgide değişiklik yapılması istenmiyorsa kullanılır. Bunun için özelliğe **true** değeri aktarılır.

FormatCurr() fonksiyonu:

Biçimleme işlemlerinde kullanılır.

```
Procedure Tform1.ButtonClick(Sender:Tobject);
Var
        Tutar:currency;
Begin
        Edit2.text:=formatcurr('#,###',tutar)'';
        Tutar:=strtocurr(edit1.text);
```

Formatdate() ve **formatfloat()** fonksiyonlarıda **formatcurr()** fonksiyonu gibi kullanılır.

Maskedit kontrolü:

Belli tip ve uzunluktaki bilgileri girerken kullanılır. Bunu kontrol eden düğme Additional sekmesinde bulunur. İlgili nesne forma yerleştirilir. Maskedit1 adını alır. Kullanıcıyı dogru ve istenilen miktarda bilgi girmeye zorlaması amacı ile kullanılır. Bu nedenle maskedit nesnesi için önceden

maske hazırlanması gerekir. Bunun için maskedit1 nesnesi seçili iken özelliklerde editmask özelliği çağrılır. Karşımıza edit mask editörü gelir.Buradan hazır maskelerden biri seçilebileceği gibi kullanıcı kendiside maske hazırlayabilir. Seçme veya hazırlama işleminden sonra diyalog kutusu kapatılır. Metin kutusu maskeye uygun bilgi girişini kabul eder.

Maske Hazırlarken Kullanabilecek Karakterler Ve Özellikleri

KARAKTER	*İŞLEVİ*
>	Küçük harfleri büyük harfe çevirir.
<	Büyük harfleri küçük harfe çevirir.
L	Yalnızca harf girilmesine izin verir. Girilmesi zorunludur.
A	Harf ve rakam girilmesine izin verir. Girilmesi zorunludur.
a	Harf ve rakam girilmesine izin verir. Girilmesi zorunlu değildir.
0	Yalnızca rakam girilmesine izin verir.
#	Rakam, + , - gibi karakterlerin girişine izin verir.
/	Tarih ayıracı.
;	Maskeleri birbirinden ayırmak için kullanılır.
-	Boşluk yerine kullanılır.

9- MEMO KONTROLÜ:

Standart sekmesinde bulunan **Memo** nesnesi birden fazla bölünecek kadar fazla bilgi varsa kullanılır. Memo nesnesinin form üzerinde yerini belirlerken memo nesnesine ait **align** özelliğinden yararlanılır. Özelliklerden Align özelliğine **alBottom** değeri aktarılırsa memo nesnesi formun alt kenarına yanaştırılır.

Memo1 nesnesine varsayım olarak aktarılan bilgiyi silmek veya başka bir bilgi aktarmak için özelliklerden **Lines** özelliği kullanılır.Lines özelliği seçilirse String list editör diyalog kutusu gelir. Buradan içerik silinebilir, düzeltilebilir veya yeniden girilebilir.

Bilgi girişi sırasında satır doldukça enter tuşuna basılmaksızın bir alt satıra geçer. Bu geçişi ise **WordWrap** özelliği sağlar. Bu özellik **false** yapılırsa enter tuşuna basılması gerekir.

Enter tuşuna basmayı da engellemek istiyorsak **WnatReturns** özelliğine **False** değeri aktarılmalıdır.

Bilgi girişi sırasında oluşan görünümün düzenlenmesi için **Alignment** özelliği seçeneklerinden biri kullanılabilir.

Girilen bilgilerin uzunluğu konusunda sınırlama gerekiyorsa **MaxLength** Özelliği kullanılır. Varsayım olarak 0 bulunduğu sürece sınırlama kabul etmez.

Çalışma anında başka bir dosyanın içeriği lines özelliğine aktarılabilir. Bunun için **LoadFromFile()** metodu kullanılır.

```
Procedure Tform1.FormClick(Sender:Tobject);
Begin
        Memo1.Lines.LoadFromFile('c:/autoexe
c.bat');
End;
```

Memoya girilen bilgiler text özelliğine aktarılır. Showmessage ile görüntülenebilir.

Memoya aktarılan bilgiler ayrılan alana sığmaz ise kaydırma çubukları eklenebilir.Bunun için **ScrollBars** özelliğinden yararlanılır.Varsayım olarak **ssNone** değerini içerir. Buraya **Both** atanırsa yatay ve düşey kaydırma çubukları eklenir.

SaveToFile metodu ile memo nesnesinin içeriği diske kayıt yapılabilir. Kaydedilen dosya not defteri ile açılabilir.

```
Procedure Tform1.ButtonClick(Sender:Tobject);
Begin
        Memo1.Lines.SaveToFile('c:\deneme.txt
');
End;
```

Count özelliği memo nesnesinin satır sayısını bulundurur.

```
Begin
        Edit1.text:=ınttostr(memo1.lines.count);
End;
```

Line() özelliğine parametre vererek memonun herhangi bir satırı başka bir alanda kullanılabilir. İlk satır numarası 0 kabul edilir.

```
Procedure Tform1.Aktarclick(Sender:Tobject);
Begin
         Edit1.Text:=memo1.lines[2];
End;
```

Add metodu memo alanının en sonuna satır ekler.

```
Procedure Tform1.ekleclick(Sender:Tobject);
Begin
         Memo1.lines.add(edit1.text);
End;
```

Insert metodu memonun herhangi bir satırına eklemek yapmak için kullanılır.

```
Procedure Tform1.ekleclick(Sender:Tobject);
Begin
        Memo1.lines.insert(strtoint(edit2.text),edi
t1.text);
End;
```

Delete metodu memonun herhangi bir satırına silmek için kullanılır.

```
Procedure Tform1.ekleclick(Sender:Tobject);
Begin
        Memo1.lines.delete(strtoint(edit2.text));
End;
```

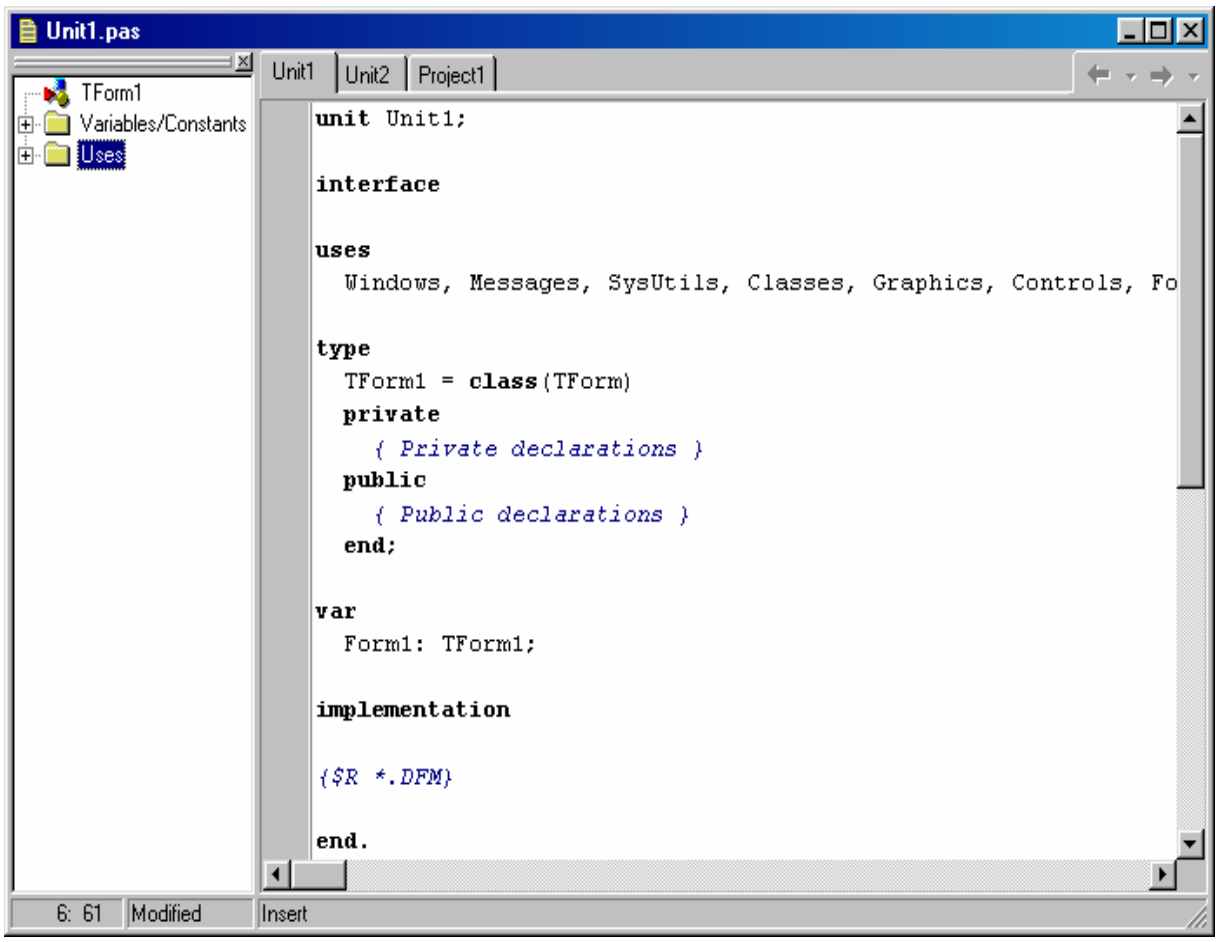
Memo nesnesinin değişik yerlerinde değişik fontlar kullanılmak isteniyorsa **RichEdit** kontrölü kullanılır. Bu kontrol **win32** sekmesinde bulunur. RichEdit nesnesi forma eklenirse buraya girilen bilgiler memo bilgisi gibi lines özelliğinde saklanır. Fontlarda düzenleme için forma font diyalog kutusu bağlanabilir. Çalışma anında kullanılır. Başka yöntemlerde tercih edilebilir.

```
Procedure Tform1.buttonclick(Sender:Tobject);
Begin
    Fontdiyalog1.Execute;

RichEdit1.SelAttributes.Assign(Fontdiyalog.Fon
t);
End;
```

10- DEĞİŞKEN YAPISI VE TANIMLARI

Projede kullanılacak değişkenlerin önceden tanımlanması gerekir.

Delphi tarafından otomatik olarak hazırlanan unitin görünümü aşağıdaki şekildedir.



Uses deyimi ile unitin gerek duyduğu tüm hazır unitler yüklenir.

Type deyimi tip tanımlamalarını içerir.

Private deyimi yordamlar bulundurur.

Public deyimi yordamlar bulundurur.

Var bildiri deyimi unit dahilinde kullanılcak değişkenler tanımlanır. Tanımlanan bu değişkenler istenen yordamlarda kullanılır.

Unit unit1;

Interface

Uses

Windows. Message, Sysutils, Classes, Graphics

Type

Tform1 = class(Tform)

Private

{private declarations}

public

{public declaration}

Örnek:

```
Procedure Tform1.formcreate(sender;Tobject);

Begin
        Ad     := 'ahmet'
        Soyad:= 'tasar'
End;
Prosedure tform.form1click(sender;tobject);

Begin
        Showmessage(ad + soyad);
End;
```

Eğer sadece üzerinde işlem yaptığımız yordamda kullanmak üzere değişken tanımlamak istiyorsak. Bunlar yordam içinde de tanımlanabilir.

```
Procedure Tform1.formcreate(sender;Tobject);
        Ad     :string[10];
        Soyad:string[15];

Begin
        Ad     := 'ahmet'
        Soyad:= 'tasar'
End;
Prosedure tform.form1click(sender;tobject);

Begin
        Showmessage(ad + soyad);
End;
```

Yordam içinde tanımlanan değişkenler, o yordamdan çıkıldığında bellekten silinirler.Değişken tanımlama anında aynı zamanda aktarma işlemide yapılabilir. Ancak bu özellik **var** bloğunda tanımlanan değişkenler için geçerlidir. Yordam içindeki değişken tanımında geçersizdir.

Örnek:

```
Ad:string[10]='alper';
```

Char Bilgi Tipi:

Tek karakterlik bilgileri geçici olarak bellekte saklamak gerektiğinde kullanılır. Char tipindeki değişkenlerde ASCII karakter kümesindeki 256 karakterden birisi saklanabilir.

Örnek:

```
Var
Evethayır:char;
```

Karakter tipi değişkenlere aktarılan ASCII kümesindeki bilgiler ekran veya yazıcıdan görünen karakter değildir. Bir etkiyi temsil ederler.Bu tip bilgileri char tipindeki değişkenlere aktarma imkanı olmadığından chr() fonksiyonundan yararlanılır.

Char tipindeki değişkende saklanan bilginin ASCII kodunu öğrenmek istiyorsak **Ord()** fonksiyonu kullanılır.Char tipine sahip değişkene sabit bilgi atanabilir.Bilgi 'ifade' şeklinde olmalıdır.

örnek:

```
Var
Cevap:char;
Durum:char;
Begin
        Cevap:='e';
        Durum:=chr(69);
        Edit1.text:=cevap;
        Edit1.text:=durum;
End;
```

String Bilgi Tipi:

Birden fazla karakteri bellekte saklamak için kullanılır.Değişken adı ile birlikte maksimum eleman sayısı verilir.

```
Prosedure tform.form1click(sender;tobject);

var
        Ad     :string[10];
        Soyad:string[15];

Begin
        Ad     := 'ahmet'
        Soyad:= 'tasar'
Edit1.text:=ad;
Edit2.text:=soyad;
End;
```

String tipi değişkene karakter bilgi yazılır.

Sayısal veya tarih tipi bilgiyi string tipe aktarma gereği duyulursa **inttostr()** tip değiştirme fonksiyonu kullanılır. Tersi bir dönüşüm içinde **strtoint ()** fonksiyonu kullanılır.

Boolen Bilgi Tipi:

Bu tip değişkenler bellekte 1 byte yer kaplarlar.Yalnızca true, false değerini alabilirler.

Örnek:

```
Var
Evethayır:boolean;
Dogruyanlis:boolean;
```

```
Procedure Tfrom1.formclick(sender:tobject);

Var
Sonuc   :boolean;
Final   :integer;

Begin
  Final:=strtoint(inputbox('final notunu giriniz:', 'not
girişi',''));
Sonuc:=final>60;
if sonuc = true then
        Showmessage('gecer');
if sonuc = false then
        Showmessage('kalır');
End;
```

Integer Bilgi Tipi:

Tamsayı değerler için kullanılır.Delphide tamsayı değişken için integer tipten başka byte,shortint, longint, word deyimleride kullanılır.

DEĞİŞKEN TİPİ	ARALIĞI	
integer	-32768	+32767
Byte	0	255
Longint	-2.147.483.648	+2.147.483.647
Shortint	-128	+127

Real Veri Tipi:

Ondalık sayılar için kullanılır. Bu tip sayılar için real deyiminden başka single, double, extend, comp deyimleride kullanılır.

DİZİ DEĞİŞKENLER

Aynı özelliğe sahip birden fazla değişkeni bir seferde tanımlamak için **array** adı verilen dizi değişkenler kullanılır.

Kullanımı: **var**
İsim:array[ilk..son eleman] of tip

Örnek:

```
Var
Ogrenci:array[1..20] of integer
```

Dizi elemanı gerekli yerlerde indisi ile kullanılır.

Örnek:

```
Ogrenci[1]=strtoint(inputbox('not giriniz','not girişi',''))
```

VARİANT DEĞİŞKENLER

Variant bildiri deyimi ile tanımlanan değişkenlere herhangi tipteki bilgiler aktarılabilir.

Örnek:

```
Var
Ad:variant;
Begin
    Ad:='konya';
     Ad:=713.017;
End;
```

Variant tipi değişkene aktarılan bilginin tipini öğrenmek için **vartype()** fonksiyonu kullanılır.

Örnek:

```
Var
Ad:variant;
tip:variant;
Begin
    Ad:=743;
     Tip:=vartype(ad);
     Showmessage(tip);
End;
```

Vartype() fonksiyonu geriye sayı gönderir. Bu sayılar ve anlamları aşağıdaki tabloda olduğu gibidir.

GÖNDERİLEN SAYI	ANLAMI
0	EMPTY
1	NULL
2	SMALLİNT
3	INTEGER
4	SINGLE
5	DOUBLE
6	CURRENCY
7	DATE
8	OLESTR
9	DISPATCH
10	ERROR
11	BOOLEAN

SABİT TANIMLAMAK

Değişkenin içeriği değiştirilmesin isteniyorsa yeni değişken tanımlamak yerine **Const** deyimi ile sabit tanımlanır. Bu durumlarda da tip belirtmeye gerek yoktur.

Örnek:

```
Const
    Hafta=7;
    Yıl=365;
     Derleyici='delphi';
Begin
     Showmessage(derleyici);
End;
```

11- OBJECT PASCAL PROGRAMLAMA DİLİ (I)

Delphi uygulama geliştirme dili olarak **Object Pascalı** seçmiştir. Borland Pascal 7.0 'ın bir devamı niteliğinde olan Object Pascal Delphi ile oldukça güçlü hale gelmiştir.

Sabitler: Program içinde belli bir değer birden fazla kullanılıyorsa bu değeri gerekli olan her yerde tekrar tekrar yazmak yerine bir kere tanımlamak daha uygun olur. Bu tanımlama ile programın okunulabilirliği artırıldığı gibi belli zamanlarda programın güncellemesi kolaylaşır. Yani bir değişiklik gerektiğinde sadece tanım değiştirilir. Hata yapma riski azalır.

Object Pascal'da sabitler **Const** ile başlayan bölümde tanımlanır.

Örnek:

```
Const
bolumadı='BİLGİSAYAR PROGRAMCILIĞI';
Boladıuzunluk=Length(bolumadı);
Saatdakika=60;
Gundakika=24*saatdakika;
Kdv=12
```

Const bölümünde sayısal veya alfabetik ifade tanımlanabileceği gibi tanımlanan değeri bir formülle kullanabiliriz.Ancak değişken kullanamayız. Zira değişkenin değeri programın çalışması sırasında hesaplanır. Oysa sabitler **.EXE** oluşmadan önce derleme sırasında sabitin yerine asıl değerin konması ile oluşmaktadır. Sabitlere eşitlenecek pascal fonksiyonları da vardır. Bunlar;

Abs,
Addr,
Chr,
Hi,
High,
Length,
Lo,
Low,
Odd,
Ord,
Pred,
Round,
SizeOff,
Succ,
Swap,
Trunc

Şeklinde ifade edilebilir.

Değişkenler: Pascal değişkenleri tipleri önceden belirlenen , içlerinde belli değeri tutan ve değere programatik olarak ulaşılan hafıza alanlarıdır. Değişkenlerin dört önemli özelliği vardır.

Değişkenin adı: Değişkeni tanımlar ona ulaşmayı sağlar.

Değişkenin Tipi: Değişkenein ne türde değer alabileceğini tanımlar.

Değeri: Değişkenin o an için hafızada bulundurduğu değeridir.

Hafızadaki alanı: Hafıza adresi olarakta adlandırılabilir. Görsel programlamada genel olarak hafıza adreslerine pek gerek yoktur.

Değişkenlerin Deklarasyonu: Değişkenler Object Pascal'da **Var** deyimi ile başlayan blok içinde yazılır.

Örnek:

```
Var
İsim, soyad :string;
Yas:byte;
Maas:integer;
Telefon:string;
```

Değişken deklare edildiğinde ,derleyici hafızada o değişkenin tutabileceği bir alan oluşturur. Ancak bu aşamada değişkenin değeri belli değildir. Program içinde değişkenlere değer atanacaktır. Değer atama operatörü **:=** işareti ile gerçekleşir.

Örnek:

```
isim:='Hasan';
Soyad := 'Kara';
Yas:=23;
Boy:=180;
```

Değişken Tipleri: Genelde Object Pascal'da kullanılan değişken tipleri şunlardır:

- Basit değişken tipi
- String değişken tipi
- Yapısal değişken tipi
- İşaretçi (pointer) değişken tipi
- Prosedür değişken tipi
- Variant değişken tipi

Fundamental Ve Generic Değişken Tipleri:

Object Pascal önceden tanımlanmış bir dizi değişken tipi ile gelmektedir. Bu değişken tipleri Fundamental ve generic olmak üzere iki grupta toplanır. Fundamental değişken tipinin alabileceği değer kümesi ve formatı işletim sistemi ve **CPU**'dan bağımsız olup Object Pascal'ın değişik sürümleri için değişmezler. Ancak generic değişken tiplerinin alabileceği değer kümesi ve formatı işletim sistemi ve CPU'ya göre farklılık gösterebilir. Object pascal'da bu iki değişken tipinin oluşmasını sağlayacak önceden tanımlanmış değişken tipleri ,integer, char, string değişken tipleridir.

Integer Değişken Tipi: Tam sayılar için kullanılır. Değişik katagorideki integer tipleri hafızada değişik büyüklük kaplar. Integer değişken tipi fundamental ve generic olmak üzere iki kısımda incelenir.

Fundamental Integer Değişken Tipi		
Tip	**Alabileceği değerler**	**Kapladığı alan (BYTE)**
ShortInt	-128 127	1
SmallInt	-32,768 32,767	2
LongInt	-2,147,483,648 2,147,483,647	4
Byte	0 255	1
Word	0 65,535	2

Int64	-9,223,372,036,857,775,808....9,223,372,036,857,775,807	8

Generic Integer Değişken Tipi (32 bitlik obj.pascal)		
Tip	**Alabileceği değerler**	**Kapladığı alan (BYTE)**
Integer	-2,147,483,648 2,147,483,647	4
Cardinal	0 2,147,483,647	4

Int64 Değişken Tipi: Bu değişken tipi Delphi 4 ile birlikte gelmiştir. Sayıların büyüklüğü artığından dolayı ihtiyaç olmuştur. Örneğin önceki versiyonlarda integer değer 2,147,483,647 rakamından büyük rakam kullanılamıyordu. Oysa int64 değişken tipi 8 byte yeri olan değişkendir ve -9,223,372,036,857,775,808....9,223,372,036,857,775,807 sayı aralığında geniş bir yelpazeyi kapsar.

Real48 Tipi Değişken: Bu değişken tipide Delphi 4 ile birlikte gelmiştir. Önceki sürümlerde kullanılan 6 byte'lık real değişken tipine karşı gelmektedir. Önceki versiyonlar ile uyum problemi varsa kullanılmalıdır. Aksi halde delphi ile birlikte gelen real değişken tipi intel uyumlu olup hafızada 8 byte yer kaplamaktadır.

Real48 ve Real Tipinin Alabileceği Değerler		
Real48	2.9x10^-39	1.7x10^38
Real	5.0x10^-324	1.7x10^308

Boolen Değişken Tipi: Boolen değişkenler mantıksal değerlerdir. Object Pascal programlama dilinde değişik büyüklükte yer kaplayan birden fazla **Boolen** değişken tipi vardır.

Tip	Kapladığı Alan (Byte)
Boolen	1
ByteBoolen	1
WordBoolen	2
LongBoolen	4

Real Değişken Tipi: Bu değişkende kesirli rakamlar tutulur.

Tip	Alabileceği Değerler	Kapladığı Alan (Byte)
Real48	2.9x10^-39 ... 1.7x10^38	6
Single	1.5x10^-45 ... 3.410^38	4
Double	5.0x10^-324 ... 1.7x10^308	8
Real	5.0x10^-324 ... 1.7x10^38	8
Extended	3.4x10^-4932 ... 1.1x10^4932	10
Comp	-2^63+1 ... 2^63-1	8
Currency	-922,3376,203,685,477.58808 922,3376,203,685,477.58808	8

Char Değişken Tipi: Bu değişken tipinde sadece tek karakter tutmak mümkündür. Object Pascal fundamental ve generic olmak üzere iki karakter tipini desteklemektedir. Fundamental karakter tipi **AnsiChar** ve **WideChar** olmak üzere iki tanedir. Ansichar değişken tipi Ansi karakter setini destekleyen 1 byte uzunluğunda değişken tipidir. WideChar ise Unicode karaker setini destekleyen 2 byte uzunluğunda değişken tipidir. WideChar tipinin ilk 256 karakteri ANSI ile eşleniktir. Generic

karakter değişkenin tipi Char'dır . WideChar genellikle karmaşık harfleri olan diller için kullanılacak olan sembolleri içerir.

String Değişken Tipi: Aslında Char karakter tipinin katarını bulundurur. Temelde **ShortString** ve **LongString** olmak üzere iki türdür. (Ayrıca WideString tipide mevcuttur.) ShortString Pascal dilini desteklemek amacı ile vardır ve en fazla 255 karakter bulundurur. LongString ise hafızadaki boş yer kadar uzunluk tutabilir. Hiçbir şey belirtilmediği sürece String değişken tipi LongString kabul edilmektedir. Programcı değişken için uzunlukta belirleyebilir.

Delphi' de string değişkenler ShortString, AnsiString(LongString), WideString, Pchar olarak kullanılabilir.

ShortString: Bu türe sahip değişkenler maksimum 255 uzunluğa sahip olabilirler.

```
Var
X : ShortString;
Begin
  X := 'Delphi';
End;
```

x değişkeni hafızada #6,'D','e','l','p','h','i' şeklinde saklanır

ShortString tipine ait değişkenlerin belli bir harfi o harfin indeks numarası ile elde edebilir. Yani Shortstring tipindeki dekişkenler indeksi 1 ile başlayan karakter dizileridir. Yukardaki örnekte x[3] bize "l"harfini x[1] "D" harfini verir. X[0] ise 6 rakamını verir. Bu ise stringin uzunluğudur ve 0. byte'da saklanır.

Pchar: Bu karakter tipi ile string uzunluktaki maksimum 255 karakterlik olan uzunluk aşılabilir. Pchar dekişkenler özellikle Delphi ile C,C++ gibi programlama dilleri arasında parametre geçişleri yaparken yada WindowsAPI fonksiyonlarını çağırırken sık sık kullanılırlar. Bu tipteki değişkenlerin içinde bulunan değerin sonunda bir sonlandırıcı (null karakter) vardır. Bu karakter sayesinde değişkenin içindeki değerin nerede son bulduğu anlaşılır.

Örnek:

```
Var
X : PChar;
Begin
  X := 'Delphi';
End;
```

x değişkeni hafızada 'D','e','l','p','h','i',#0 şeklinde saklanır.

Pchar değişken tipi aslında bir pointer (hafıza adresi) dir. Delphi modülü içinde aşağıdaki şekilde tanımlanır.

```
Type
Pchar = ^Char;
```

AnsiString: Delphi 2.0 ile birlikte gelmiştir. Bu tipe ait değişkenler 2 Gb'a kadar yer tutabilirler. Aslında bunlarda birer hafıza değişkenidir , sonlarında sıfır sonlandırıcı işareti vardır. AnsiString aynı zamanda longString değişkendir. Hafızaya dinamik olarak yerleşirler. Longstring ve Pchar arasında da uyum vardır. Birbirlerine çevrilebilirler.

ShortString, AnsiString, String: String kelimesi hem **AnsiString** hemde **ShortString** yerine kullanılmaktadır. Ana menüden, Project, Options seçenekleri seçildiğinde gelen ekrandan compiler sekmesi seçilsin. Burada Hihg String denilen yeri işaretli ise programda string olarak tanımlanan her değişken AnsiString kabul edilecektir. Burası işaretli değilse tüm String değişkenler ShortString kabul edilecektir.

{$H} direktifi:** Programda Project,options seçimi ile yapılan stringin türünü belirleme program içinde de gerçekleşebilir. Program içinde herhangi bir yerde **{$H+} ifadesini kullanmışsak deklare edilen stringler AnsiString yerine geçmektedir. **{$H-}** ifadesinden sonra deklare edilen stringler ise Short String yerine geçmektedir.

Type Deklarasyon Bloğu: Object Pascal programlama dili yeni data tipleri tanımlamak için gerekli ortamı vermektedir. Bu yeni data tipleri programın type deklarasyon bloğunda gerçekleşir.

Genel ifadesi

```
Type
Yenitip1=<tip tanımı>
Yenitip2=<tip tanımı>
```

Şeklindedir. Tip bir kez oluşturulduktan sonra değişkenlerin tipi bu oluşturulan tiplerden biri olarak verilebilir.

```
Var
Değişken1:yenitip1;
Değişken2:yenitip2;
```

Örnek:

```
Type
Indeks=1..255;
String20=String[20];
```

Sıralı Tipler: Sıralı tipler kullanıcı tanımlı, basit ve anlaşılırlık için kullanılabilecek tip tanımlarına verilebilecek örneklerden biridir.

Genel ifadesi;

```
Type
YeniTip=(değer1, değer2,.............,değerN);
```

Şeklindedir.

Örnek:

```
Type
Gunler=(pazartesi,Salı,Çarşamba,Perşembe,Cuma,cumartesi,Pazar);
Renkler=(sarı,kırmızı,mavi,yeşil,siyah,beyaz);

Var
Gun:Gunler;
```

Gun:=Salı; şeklinde bir atama yapılabilir.

- Ayrıca verilen değerler kendi içinde sıralıdır. Yani Pazartesi<Salı mantıksal ifadesi doğru olarak anlaşılacaktır.

- Sıralı tiplerde değerlerin yerleşme sıraları onların sıra(ordinal) numarasını verir. (Pazartesi 0, Salı 1, Çarşamba 2 gibi) Bu numaralara **Ord()** fonksiyonu ile ulaşılabilir.

Örnek:

```
Gun:=carsamba;
Showmessage(inttostr(ord(gun)));
```

Bu örnek sonucu ekranda 02 yazılacaktır.

Variant Değişkenler: Variant bildiri deyimi ile tanımlanan değişkenlere herhangi tipteki bilgiler aktarılabilir.

Örnek:

```
Var
Ad:variant;
Begin
   Ad:='konya';
    Ad:=713.017;
End;
```

Variant tipi değişkene aktarılan bilginin tipini öğrenmek için **vartype()** fonksiyonu kullanılır.

Örnek:

```
Var
Ad:variant;
tip:variant;
Begin
   Ad:=743;
    Tip:=vartype(ad);
    Showmessage(tip);
End;
```

Vartype() fonksiyonu geriye sayı gönderir. Bu sayılar ve anlamları aşağıdaki tabloda olduğu gibidir.

GÖNDERİLEN	ANLAMI

SAYI	
0	EMPTY
1	NULL
2	SMALLİNT
3	INTEGER
4	SINGLE
5	DOUBLE
6	CURRENCY
7	DATE
8	OLESTR
9	DISPATCH
10	ERROR
11	BOOLEAN

Diziler: Dizi ayni tipe sahip bir çok değerin tutalabildiği bilgi yapılarıdır. Object Pascal'da diziler 1,2,3 veya N boyutlu olabilirler.

Genel ifadesi;

```
Type
Diziadı1=array[Indextanımı] of birbaşkatip;
Diziadı2=array[Indextanımı1,Indextanımı2] of birbaşkatip;
Diziadı3=array[Indextanımı1,Indextanımı2,Indextanımı3] of birbaşkatip
```

Şeklindedir.

Örnek:

```
Type
Maasdizisi=array[1..12] of integer;
Gunler=(Pazartesi,Salı,Carsamba,Persembe,Cuma,Cumartesi,Pazar);
Gundizi=array[Pazartesi..Cuma] of string;
```

Kayıt (Record) Veri Tipi: Değişik veri tipine sahip elemanların bir araya gelerek oluşturdukları kapsüle edilmiş değişken tipleridir. Elemanlarının her birine alan adı verilir.

Genel ifadesi

```
Type
Tipadı=record
Alan1:alantipi1;
Alan2:alantipi2;
...............
alanN:alantipiN;
end;
```

Şeklindedir.

Örnek:

```
Type
ogrencitipi=record
Adi:string[20];
Soyadi : string[20];
Yas:integer;
```

KONTROL YAPILARI

Object Pascal'da kontrol yapıları, koşullu ve döngü yapıları olmak üzere iki kısımdan oluşur.

If Deyimi: Seçim ifadesi olarakta adlandırılabilir.

Genel ifadesi

```
if koşul then begin
                .........
                end
        else begin
                ..........
                end;
```

Şeklindedir.

Örnek:

```
if hizsure< 10 then begin
                ikramiye:=maas * 2;
                end
        else begin
                ikramiye:=maas*3;
                end;
```

Koşulun olumlu ve olumsuz durumda yapılacak iş birden fazla ise işler begin ve end deyimleri arasına yazılır.Aksi takdirde begin ve end bloğuna gerek yoktur.

Case Deyimi: İç içe bir çok if kullanılması gerektiğinde kullanılır.

Genel ifadesi:

```
Case menusecenek of
        1:procedure01;
        2:procedure02;
        3:procedure03;
        4:procedure04;
        else 5:procedure05;
end;
```

Şeklindedir.

While Döngüsü: Koşul sağlandığı sürece döngü çalışır.

Örnek:

```
 i:=1;
While i<=10
        Do begin
                Showmessage(inttostr(i));
                inc(i);
        end;
```

Repeat Until Döngüsü: Koşul ne olursa olsun repeat until anahtar kelimeleri arasındaki bölüm bir kez çalıştırılmakta sonra şarta bakılmaktadır.

Genel ifadesi;

```
 Repeat
.........................
until koşul;
```

Şeklindedir.

Örnek:

```
 i:=01;
Repeat
        Showmessage(inttostr(i));
        İnc(i);
Until i==11;
```

For Döngüsü: For döngüsünde başlangıç bitiş arasında tanımlanan sayaç yardımı ile belirtilen işler tekrarlanır.

Genel ifadesi;

```
 For sayac:=baslamadegeri to bitisdegeri do
begin
        ...............................
end;
```

şeklindedir.

Örnek:

```
 For i:=01 to 10 do
begin
        Showmessage(inttostr(i));
End;
```

Döngü azalan biçimde olması isteniyorsa ;

```
 For sayac:=sondeger downto ilkdeger do
begin
        ..............................
end;
```

Şeklinde ifade edilmelidir.

Break İfadesi: Bu deyim ile çalışmakta olan bir döngü kırılıp döngü dışına çıkarılabilir.

Örnek:

```
 i:=01;
while true
do begin
        showmessage(inttostr(i));
        inc(i);
        if i> 10 then break;
end;
```

Continue İfadesi: Bu komut sayesinde komutun altında kalan satırlar çalıştırılmadan döngünün en başına gidilebilir.

Örnek:

```
 i:=00;
while i<=10
do begin
        inc(i);
        if i=5 continue;
                showmessage(inttostr(i));
end;
```

ÖRNEK TANIMLAR

Örnek1:

```
Procedure Tform1.formcreate(sender;Tobject);

Begin
        Ad     := 'ahmet'
        Soyad:= 'tasar'
End;
Prosedure tform.form1click(sender;tobject);

Begin
        Showmessage(ad + soyad);
End;
```

Örnek2:

```
Var
Cevap:char;
Durum:char;
Begin
        Cevap:='e';
        Durum:=chr(69);
        Edit1.text:=cevap;
        Edit1.text:=durum;
End;
```

Örnek3:

```
Procedure Tfrom1.formclick(sender:tobject);

Var
Sonuc   :boolean;
Final    :integer;

Begin
  Final:=strtoint(inputbox('final notunu giriniz:', 'not girişi',''));
        Sonuc:=final>60;
if sonuc = true then
        Showmessage('gecer');
if sonuc = false then
        Showmessage('kalır');
End;
```

.

12- ALFABETİK ,SAYISAL VE TARİH FONKSİYONLARI

Length(): Alfabetik bir dizinin uzunluğunu bulur. Dizinin kendisi veya bulunduğu değişken olabilir.

Kullanım Şekli: değişken=Length('ifade');
Değişken=Length(değişken);

Copy(): Alfasayısal ifadenin veya değişkenin soldan itibaren istenilen kadarını kopyalar.

Kullanım Şekli: değişken=Copy(değişken,baş.değeri,uzunluk);
Değişken=Copy('ifade',baş.değeri,uzunluk);

LowerCase(): Alfabetik bilgiyi küçük harfe çevirir.

Kullanım Şekli: Değişken=LowerCase('ifade');
Değişken=LowerCase(değişken);

NOT: Bu fonksiyon türkçe harfleri tanımaz. Bunun için **Ansilowercase()** fonksiyonu kullanılır.

UpperCase(): Alfabetik bilgiyi büyük harfe çevirir.

Kullanım Şekli: Değişken=UpperCase('ifade');
Değişken=UpperCase(değişken);

NOT: Bu fonksiyon türkçe harfleri tanımaz. Bunun için Ansiuppercase() fonksiyonu kullanılır.

Pos(): Alfasıyal bilgiden yine alfasayısal bir bilgiyi aramak için kullanılır.

Kullanım Şekli: değişken=pos(aranacak dizi,ana dizi);

Comparestr(): İki alfabetik bilgiyi veya bunları içeren değişkenleri karşılaştırır.

Kullanım Şekli: değişken=comparestr(değişken1,değişken2);

Insert(): Alfabetik bilgi içine alfabetik bilgi yerleştirir.

Kullanım Şekli: değişken=(eklenecek,anadizi,başlama noktası);

Delete(): Alfabetik bilginin içinden belirtilen kadarını siler.

Kullanım Şekli: değişken=delete(dizi,başlangıç,silme adeti);

Inttostr(): Tamsayı içeren sayısal bilgiyi alfabetik bilgiye çevirir.

Kullanım Şekli: değişken=inttostr(sayısal değişken);

Strtoint(): Alfabetik bilgiyi tamsayı bilgiye çevirir.

Kullanım Şekli: değişken=strtoint(alfabetik değişken);

Floattostr(): Ondalıksayı içeren sayısal bilgiyi alfabetik bilgiye çevirir.

Kullanım Şekli: değişken=Floattostr(sayısal değişken);

Strtofloat(): Alfabetik bilgiyi ondalık sayı bilgisine çevirir.

Kullanım Şekli: değişken=strtofloat(alfabetik değişken);

NOT: Real ve single bildiri deyimi ile tanımlanan değişkenler 15 basamaktır.

Chr(): 0-255 arası sayının ascii karakterini verir.

Kullanım Şekli: değişken=chr(ascii kodu);

Ord(): ASCII karakter kümesindeki herhangibir karakterin kodunu bulur.

Kullanım Şekli: değişken=ord('tek karakterlik bilgi');;

Val(): Karaktersel bir bilgiyi sayısala çevirir. Ondalık haneyi de gösterir.

Kullanım Şekli: Val(alf.bilgi,sayısal değişken,hata kodu değişkeni);

Abs(): Mutlak değer fonksiyonudur.

Kullanım Şekli: değişken=Abs(sayı)

Int(): Ondalık bir sayının tam kısmını alır.

Kullanım Şekli: değişken=Int(değişken);

Sqrt(): Karekök fonksiyonudur.

Kullanım Şekli: değişken=sqrt(değişken);

Date(): Parametresiz çalışır. Sistem saatini geriye döndürür.

Örnek: tar.text:=Datetostr(date);

Datetostr(): Tarih fonksiyonunun string ifadeye dönüştürülmesinde kullanılır.

Strtodate(): String ifadenin Delphi Tarih fonksiyonuna dönüştürülmesinde kullanılır.

Timetostr(): Zaman fonksiyonunun string ifadeye dönüştürülmesinde kullanılır.

Strtotime(): String ifadenin Delphi Zaman fonksiyonuna dönüştürülmesinde kullanılır.

Örnek: saat.text:=timetostr(time);

Now(): Tarih ve saati birlikte döndürür.

Örnek: hepsi.text:=datetimetostr(now);

Datetimetostr(): Tarih ve saat fonsiyonunu string bilgiye çevirir.

Strtodatetime(): String bilgiyi tarih ve saat bilgisine çevirir.

Dayofweek(): Haftanın gününü belirtir.

Örnek: edit2.text:=inttostr(dayofweek(bugun));

Formatdatetime(): Tarih ve zaman bilgilerinin formatını değiştirmek için kullanılır.

Örnek:

Edit1.text:=formatdatetime('dd/mm/yyy'+'tt',now);

Getdir(): Belirtilen sürücüdeki aktif dizini string değişkene aktarır.

Kullanım Şekli:

```
Var
Dizinadı:string;
Begin
        Getdir(0,dizinadı);
End;
```

Burada; 0, aktif sürücüde aktif dizini,
1, A sürücüsündeki geçerli dizini,
3, C sürücüsündeki aktif dizini simgeler.

Chdir(): Geçerli sürücü ve dizin değiştirirlir.

Örnek: Chdir('C:\');

Mkdir(): Kök altında dizin açar.

Örnek: Mkdır('test');

Rmdir(): Dizin siler.

Örnek: Rmdir('test');

Deletefile(): Dış üniteden dosya veya dosyaları silmek için kullanılır.

Örnek: Chdir('C:\dos');
Deletefile('readme.txt');

Renamefile(): Dış ünitedeki dosya adını değiştirmek için kullanılır.

Örnek: renamefile(edit1.text,edit2.text);

Filegetattr(): Dosya niteliklerini değiştirmek için kullanılır.Öz nitelik normal ise geriye 128 değerini döndürür.

Geriye Döndürülen Değer	Niteliği
128	Normal
1	Readonly
2	Hidden
3	Hidden+readonly
4	System
16	Dizin(klavuz)

13- MENÜ HAZIRLAMAK

Windows uyumlu programlarda pencerenin en başında başlık çubuğu vardır. Açılan menüler bunun hemen altındadır. Menüdeki her nesne uygulamanın özelliğine göre birden fazla seçenek bulundurur. Bu seçeneklerin alt seçenekleri de olabilir.

Seçenekler içinde sağda **"> "** karakteri varsa bu seçeneğin bir alt menüsü olduğunu gösterir.

Seçenekler içinde sağda **"..."** karakteri varsa bu, seçeneğin diyalog kutusu ekrana getireceğini belirtir.

Seçeneklerdeki altılı çizgili karakter ise bunun kısayol için gerekli olduğunu gösterir.

Menü hazırlamak için :

- Bileşenler paletinde, Standart sekmesinde bulunan **Mainmenü** nesnesini temsil eden düğme forma aktarılır. (name varsayım olarak ***mainmenu1*** olur)
- Mainmenu1'in başlangıçta 10 özelliği vardır. Menüler ve menülerdeki komutlar ***items*** özelliği ile belirlenir.
- Items tıklanır ise boş olarak Mainmenu penceresi gelir. Bu boş ekrana ***Menu Designer*** adı verilir.
- İlk ***Menu*** başlığı için özelliklerden, **Caption**'a başlık (Dosya) yazılır. Bu **Name**'de aktarılır.
- Bu şekilde bütün açılan menüler tamamlanır. (Dosya, Düzen, Görünüm, Ekle, Biçim, Araçlar)
- Hangi menüde hangi komutlar bulunacak ise menünün üzerinde tıklama yapılır.
- Burada ilk komut yazılır. Bu **Caption** ve **Name** 'e aktarılır.
- Bu şekilde tüm komutlar girilir.
- Komutlar gruplandırılmak istenirse (araya çizgi) bunada bir isim verilmelidir. Bu ise başlığa (-) (tire) işareti vermekle olur.
- Komut yazma işi bittikten sonra **Mainmenu1** ekranı kapatılır **Form1**'e dönülür.
- Form üzerinde menülerde gezinip, tıklama yapılırsa komutlar görülür.
- Hangi komut için komut yazılacaksa onun üzerinde iken tıklanır veya entere basılır. İlgili yardım gelir.

- **Menuitems** nesnesi üzerinde sağ tıklama ile kısa yol ekranı getirilir. **Menudesigner** getirilip tekrar düzenleme yapılabilir.
- **(>) Altmenu** eklemek için; altmenüsü olacak komut üzerinde iken sağ tuş tıklanırsa menüler ve komutlar ile ilgili yapılacak işleri temsil eden komutlar gelir **Create Submenu** seçilir.

```
Insert
Delete
Createsubmenu
Selectmenu
...............
```

- Artık alt menünün ilk komutu direkt yazılabilir.
- Diğer komutlarda aynen yazılır.
- Delphi menü veya komut nesnelerine verilen başlık bilgisini aynı zamanda menü ve komut adı olarak kullanır.
- Ancak başlıkta Türkçe karakterler var ise bunlar çıkarılır. Menü adı ile başlığı ayırt etmek için menü adının sonuna " **1** " rakamı gelir.

Menülerden Komut Silmek Ve Eklemek

Silmek; menü tasarım penceresinde o komuta ait kısa yol menüsünü açıp **delete** kullanılır.
Eklemek; hangi komut üzerine eklemek istiyorsak komuta ait kısa yol açılıp **insert** seçeneği kullanılır. **Caption** geldiğinde de değişiklik yapılmalıdır.

Menü Ve Komutların Özellikleri

1. Bazı menülerde hangi komut seçili ise o komutun önüne seçme işareti gelebilir. ***(∨)*** Bu işlem **menüitems** nesnesine ait **checked** özelliği ile yapılır. Bunun içinde **checked** seçeneğine **true** değeri aktarılır (o nesne seçili iken). Yalnız bu pratiklik değildir. Bu işareti bulunduracak komut üzerine tıklanınca **unit**'teki yordamı gelir.

```
Begin
        if  pencere11.checked =true then
            pencere11.checked=false
            else pencere11.checked=true
end;
```

Yordamı yazılır (pencere1. seçilecek komut adıdır)

2. Delphi projelerinde bir komutun soluk renkte görünmesi veya seçilememesi için; **Enable** özelliğinden yararlanılır.

Örnek; Satır komutları silik olsun isteniyor. Satırda çift tıklanırsa;

```
begin
        satır1.Enable:=false;
end;
```

Yordamı yazılır.

3. Menüdeki ifadelerden bazılarının altı çizili olsun isteniyorsa **Caption** özelliğinde ilgili karakterin önüne **&** işareti konur. Böylece **Alt+çizili karakter** kısa yol tuşu olarak kullanabilirsiniz.

4. Ayrıca kısa yol tuşu tanımlanabilir. Bunun için **Menu İtems** nesnesine ait **shortcut** özelliği kullanılır ilgili nesnede iken **shortcut** seçilirse **Ctrl+tuşlardan** biri seçilir.

Pop Up Menüler Hazırlamak

Açılan menülere alternatif olarak **pop up** menüleride kullanılmaktadır. Kısa yol menü kullanımıdır. Yani sağ tuşu tıklamadır. Belgenin seçili durumda olan kısmı olan üzerinde hangi işlem yapmak istiyorsak kısa yoldan o komut verilebilir. Bunlara **Pop Up Menu** denir.

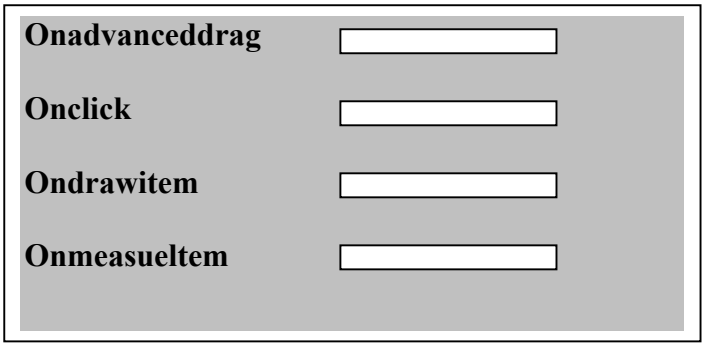
Bunun için;
- **Component Palet**, **Standart**, **Pop Up Menü** nesnesi form üzerine alınır. Bu nesneye varsayım olarak **Popupmenu1** adı verilir. Bu menüdeki komutları iki farklı şekilde belirleyebiliriz.

1. **Popupmenu** seçili iken, özelliklerden **items** tıklanır.
2. **Popupmenu** seçili iken kısa yoldan **Menu Designer** seçilebilir.

Buradan dizayn işlemi için;

1. **Caption** adları yazılarak komutlar tanımlanır.
2. Tasarım yapıldıktan sonra dizayn penceresi kapatılır veya çift tıklama ile kapatılır.
3. Caption isminin sonuna 1 eklenerek name ismi olarak otomatik atanır. Name de değişiklik yapılabilir.

   - Hazırlanan popup menuler form çalışırken sağ tuş tıklanırsa çalışır. Bunu sağlamak için özellikler (form1) den, popmenu'ye adı yazılır. (popup menu1)
   - Bu komutların (açılan menü, popmenu) çalışabilmesi için olay seçilir. Tanımlı dört olay vardır.



On click tıklanırsa ilgili yordam gelir.

Hazır Menüleri Kullanmak

Çok sık kullanılan menü seçenekleri hazır olarak verilmektedir. Bunun için;

1. Forma **Main Menu** eklenir.
2. **Menu Designer** ile tasarım açılır.

3. Hazır menülerden yararlanmak için kısayol (sağ tıklama) açılır. Buradan **InsertFromTemlate** seçilir. Bu seçim sonunda hazır menü şablonları listelenir. Hangisi kullanılacak ise seçim yapılır.

Menüleri Şablon Olarak Kayıt Etmek

Proje menüleri birbirine benzediğinden hazırlanan bir menüyü şablon olarak kayıt edebiliriz. Bunları başka projelerde kullanabiliriz. Bunun için;

1. **Menu Designer**'dan kısayol açılır
2. **Save As Temlete** komutu kullanılır.
3. İsim istenir.İsim verilir.

Bu şablonu daha sonra kulanmak istersek;

1. Kısa yoldan **Insert Form Temlete** seçilir
2. Şablonlar listelenir
3. İstenen şablon seçilir.

Şablonları silmek için;

1. **Menu Designer**'dan kısa yol açılır
2. Şablonlar listelenir
3. **Delete Temletes** seçilir

- Kısa yolda bulunan **Select Menu** seçeneği aynı formda birden fazla MainMenu nesnesi varsa kullanılır.

Komut Önüne Resim Koymak

Komutların önüne işlevi ile ilgili resim konabilir. Bunun için;

1. **Main Menu** seçili iken kısayol açılır,
2. **Menu Designer** çalıştırılır,
3. İlgili komut seçilir,
4. Resim **Bitmap** özelliğinden yararlanılarak belirlenir.Yeri seçili iken bitmap tıklanır,
5. Ekrana **Load Picture** diyalog kutusu gelir,
6. **Load** tıklanarak resimler listeletilir,
7. İstenen resim seçildiğinde yerine yerleşmiş olur.

14- DATABASE DESKTOP VE BORLAND DATABASE ENGINE

Delphi ile birlikte Database Desktop adında bir uygulama daha gelmektedir. Bu uygulama **SQL, PARADOX, DBASE** gibi veritabanı programlarını kullanarak tablolar oluşturmamızı ve üzerinde güncelemeler yapmamızı sağlar.

Borland Database Engine, geliştirdiğimiz uygulamaları son kullanıcılara verirken, son kullanıcı bilgisayarına yüklenmesi gereken bir programdır. **BDE** son kullanıcı bilgisayarına yüklendikten sonra takma adlar yaratılmalı ve sistem konfigurasyonu yapılmalıdır.

BDE programlama elbetteki delphi bileşenlerini kullanarak yapacağımız programlardan daha zordur. Zaman zaman bu fonksiyonları kullanmak zorunda kalabiliriz.

Alias Tanımlarken Gelen Parametreler;

NetDır : Network altında uygulama geliştirilirken kullanılır.

LangDriver : Programlarda türkçe harfler kullanabilmemiz için buranın "paradox 'turk' " olması gerekir.

BlockSize : Kayıtların hafızada tampon bölgede tutulacağı alanın büyüklüğünü verir. Bu büyüklük programın hızınıda belirler. Bu büyüklük 1*24'ün katları olabilir ve en fazla 32 kilobayte'a çıkabilir.

BDE Administrator Programında Gelen Bazı Parametreler:

Local Share : Eğer network altında paradox tabloları kullanarak uygulama geliştiriliyorsa değişik kullanıcıların aynı kayıt üzerinde değişiklik yapmamaları için bu değerin true yapılması gerekir.

Minbufsize ve maxbufsize : Veritabanı bilgilerinin hafızada ne kadarlık bir bölümde tampon olarak tutulacagını gösterir. **Maxbufsize** 128'in katları olamalıdır.

Langdrive : Uygulamalarda tablo içinde türkçe harfler kullanmamızı saglar tabi 'turk' değeri atanmış ise.

Maxfilehandles : Bir defada **BDE** tarafından acılacak dosya sayısını gösterir.

BDE Programı

Delphi'den aldığı emirleri kullanılan veritabanı programının anlayacağı şekle dönüştürür. Örneğin Table1.Next komutu kullanılmış olsun. Bir sonraki kayıta konumlanmayı sağlayacaktır ama dbase veya paradox bunu anlamaz. BDE programı bu dönüşümü sağlar.

Borland Database Engine ile **ODBC (Open Database Connectivity)** temelde aynı işi yapar. **BDE** Borland firmasının **ODBC** ise microsoft firmasının bir ürünüdür.

Ancak BDE'de daha fazla fonksiyon vardır. Bu fonksiyonları kullanmak oldukça zahmetli olmakla beraber performansın önemli olduğu durumlarda bu fonksiyonları kullanmak gerekmektedir.

Elektriklerin Kesilmesi Ve Girilmiş Kayıtların Kaybolması: Girilen bilgiler tampon alana yazılır ve doldukça diske kayıt edilir.Diske kayıt edilmeden elektrikler kesilirse bilgi kaybı oluşur. Bunu önlemek için **BDE** fonksiyonu kullanılabilir. Bu ise tabloda değişiklik yapıldığı anda diske yazılmasını istemek şeklinde olur. Kullanılacak fonksiyon **DBISaveChanges** fonksiyonudur. Bunu kullanmadan önce **uses** bölümüne **BDE** modülü eklenmelidir. Daha sonra kullanılacak kod ise tablolarımızın **AfterPost** olayıdır.

```
Procedure Tform1.Table 1AfterPost(Dataset: TDataset);
Begin
        DBISaveChanges((DataSet as TBDEDataset).Handle)
End;
```

DELPHİ VERİTABANI BİLEŞENLERİ (DATACONTROL SEKMESİ)

TDBNavigator Bileşeni : Tablodaki kayıtlar üzerinde işlem yapmamızı sağlar. Kod kullanmadan kayıtlar üzerinde gezinmemizi, silmemizi, düzeltmemizi, kayıt yapmamızı sağlar.

TactionList Bileşeni: Kullanışlı bileşenlerden biridir. Standard sayfasında bulunmaktadır. Formlar üzerinde birbiri ile ilgili menü seçenekleri ve düğmelerin aynı anda aktif yada pasif moda geçmesi sırasında kontrolümüzü sağlar.Veritabanı operasyonlarında sık kullanılır.

Post Ve Cancel Metotları : Bu metotlar kayıt bazında gerçekleşir. Yani belli bir kayıdın üzerinde iken **post** metodu ile kabul edebilir yada **cancel** metodu ile reddedebiliriz. İki metottan ikiside kullanılmamış ise otomatik olarak post çağrılmış olur.

TDBEdit Bileşeni : Tablo alanı içindeki bilgi görüntülenir. İçerik değiştirilebilir.

TDBText Bileşeni : Tablo alanları içindeki bilginin form üzerinde görüntülenmesini sağlar. İçerik değiştirilemez.

TDBCheckBox Bileşeni : Bu bileşen sayesinde tabloda bulunan **True-False** gibi boolen tipteki herhangibir sütunun görünümünü değiştirmek ve kullanıcı arabirimini daha estetik yapmak için kullanılır.

TDBRadioGroup Bileşeni : TradioGroup bileşenine benzer.

TDBImage Bileşeni : Veritabanında bulunan bir resmin görüntülenmesini sağlar.

TDBListBox Bileşeni : Tlistbox bileşeni gibi kullanılır.

TDBComboBox Bileşeni : Tcombabox bileşeni gibi kullanılır.

TDBLookUpComboBox ve TDBLookUpListBox Bileşeni :

İki bileşenin kullanımı birbirine benzemektedir.

PARADOX DOSYALAR

Paradox formatında veritabanı dosyası hazırlarken değişkenlerin tanımı ad, uzunluk, tip olarak yapılırken bazı özelliklerde kullanılabilir ve tanımlanabilir. Bu tanımları şu şekilde özetleyebiliriz.

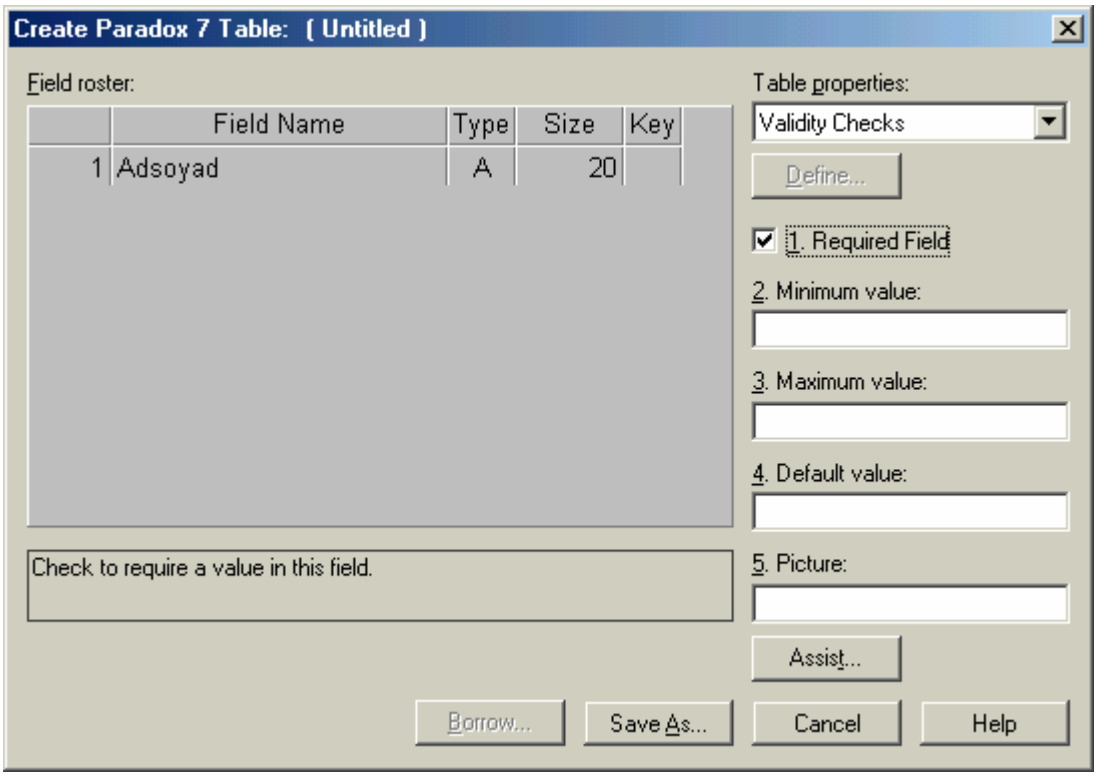


Table Properties: Combobox şeklinde ki bu tablo özellikleri 6 seçenekten oluşur. Bunlar; Validity Checks, Table Lookup, Secondary Indexes, Referential Integrity, Password Securty, Table Language.

Validity Checks: Başlangıçta otomatik olarak seçili gelir.Bu seçili iken Requiered Field onay kutusu, Minimum value, maximum value, Default Value, Picture adında metin kutuları vardır.

Required Field: Table kontrolü aracılığı ile bilgi girişi esnasında ilgili değişkene mutlaka bilgi girilmesi isteniyorsa onay kutusu seçili olmalıdır.

Minimum – Maximum Value: Sayısal alanlar için en küçük ve en büyük değeri bulundurur.

Default value: Buradaki ifade dosyaya ekleme her yeni kayıt için ilgili değişkene yazılır.

Picture: İlgili alana bilgi girirşi esnasında kullanılacak formatı bulundurur. İlgili değişken için kullanılacak format kullanıcı tarafından tanımlanabileceği gibi hazır listeden de seçilebilir. Tanımlanan alanlar başka table'larda da kullanılacak ise hazır listeye eklenebilir. **Maske Picture** alanına direkt yazılabileceği gibi **Assist** düğmesi kullanılarak hazır listeye ulaşılabilir.

Table Language: Veri tabanı dosyasının dil seçimi için kullanılır. Bu seçenekte **Modify** adlı düğme gelir. Bu seçilirse dil listesine ulaşılır.

Secondary Indexed: Ana index özelliği dışında farklı bir indexed daha isteniyorsa kullanılır.Bu seçenek seçili iken **Define** düğmesi görüntülenir. Bu seçilirse define secondary ındex diyalog kutusu gelir. Burada table'da tanımlanan değişkenler listesi gelir. Seçilmek istenen ikinci anahtar için değişken belirlenir. İsim verilerek saklanır. Secondary index ile hazırlanan ilk index için hard diske **XG0** ve **YG0** adlı iki dosya kayıt olur.Aynı özellikte ikinci bir index hazırlanacak olursa **XG1** ve **YG1** dosyaları oluşur ve kayıt olur. Bu özellikle hazırlanan indexler **Object İnspector**'da **IndexName** özelliğinde listelenir. Key sütununa '*' karakteri ile belirlenen index ise **IndexFieldName** özelliğine atılır.

Table Lookup:

Referential Integrity:

Password Security:

ALIAS HAZIRLAMAK

Delphi'de projelerde kullanılan veri tabanı dosyalarını gruplamak için **alias** adında özellik bulunmaktadır.Proje dahilinde çok sayıda veri tabanı hazırlayıp kullanılacaksa öncelikle proje için **alias** hazırlanır.

Veri tabanı dosyaları üzerinde işlem yaparken **Borland Database Engine** programından yararlanılır. Ayrıca bu program çalışma şeklini ayarlamada kullanılan **BDE Administrator** adında yardımcı program vardır.

Alias hazırlamak yada **BDE** programının çalışma şeklini isteklerimize göre ayarlamak istersek **BDE Administrator** çalıştırılır. Bu ise başlat menüsünden gerçekleşir. Bu program ile yapılan ayarlar **.CFG** uzantılı dosyaya yazılır.

BDE Administrator penceresinde;

- Menü çubuğu
- Database sekmesi (bu sekmede alias hazırlanır veya değiştirilir.)
- Configuration sekmesi bulunur. (bunun içinde de Driver ve System seçenekleri bulunur.)

Yeni Bir Alias Hazırlamak İçin;

1. **Databases** sekmesi seçili,

2. Object menüsünden **New,**

3. Ekrana **New Database Alias** ekranı gelir.Organize edilecek veri tabanında kullanılacak sürücü seçilir.

4. **STANDARD** genel olanıdır.

5. Tüm seçenekler kabul edilir**.(ok)**

6. Hazırlanan **alias** soldaki sekmede görülür. Gerekirse adı değiştirilebilir. (örnek olarak adını delphibook olarak alalım.)

7. Sağdaki sekmede de gerekli tanımlar tamamlanır.**(Definition)**

Type	Standart
Default Driver	PARADOX
Enable BCD	False
Path	C:\ORNEKLER

8. Herhangibir dil seçimi, memo, blok size gibi konfigurasyon özellikleri düzenlenecek ise **BDE Administrater Configurasion** sekmesi kullanılır.
Burada;

Version	4.0
Type	file
Langdriver	Dbase
Level	7
MDX Blok size	1024
Memo File blok size	1024

Gibi bilgiler bulunur.

9. Tüm bu tanımlama ve seçimlerden sonra **.CFG** dosyasının saklanması gerekir.

10. **Object** menüsünden **Save As** seçeneği ile saklama işlemi gerçekleştirilir.

Veri Tabanı Dosyası Hazırlamak

Delphi programlama dili bünyesinde veri tabanı araçlarını (database, paradox, interbase v.b.) da bünyesinde bulundurur.

Yeni Bir Veritabanı Dosyası Hazırlamak İçin;

1. **Databasse Desktop** (tools menusu) seçeneği kullanılır. Aynı işlem **başlat** menüsünden, **programlar, Delphi, Database desktop** kullanılarakta yapılabilir.

2. Database desktop penceresinde **file, new** seçilir.

3. New seçeneği üç seçimli ekranı görüntüler

QBE Query	(sorgu oluşturulabilir)
SQL file	(SQL kodu oluşturulabilir)
Table	(tablolar oluşturulabilir)

4. **Table** seçilir.
   Create table ekranı görüntülenir. Burada **tablo type** seçilecektir. Bunun için default değer Paradox 7 dir. Gerekirse değiştirirlebilir.

5. Veri tabanı dosyasının kayıt yapısını bulunduran ekran görüntülenir.
   Burada; Değişkenlerin Alan adı (**field name**), tipi (**type**), uzunluğu (**size**), ondalık hane (**dec**) ve varsa **anahtarlar** tanımlanır.

6. Anahtar tanımı için **Table Properties** kullanılır. Burada index yazılı iken **Define** butonuna basılır.

7. **Define İndex** diyalog kutusu gelir. Burada alan listesi vardır.Değişken seçilir. Daha sonra **Unique** veya **Maintained** alanlarından biri de işaretlenebilir.

   Unique; Anahtarın tekrarlanması isteniyorsa kullanılır.

   Maintained; Dbase IV de iki tür index vardır. Her index için ayrı bir **.NDX** hazırlanmalıdır**.** **.MDX** de ise birden fazla index vardır ve veri tabanının adını alır.
   Eğer **.NDX** kullanılacak ise **Maintained** onayı iptal edilmelidir.

   İndekse veri tabanındaki tüm kayıtları dahil etmek istemiyorsak **Subset Condition (filter) Expression** metin kutusuna filtre şartı yazılabilir.Tüm bu tanımlardan sonra okey seçilirse **Save İndex As** penceresi gelir. Burada ise **.NDX** uzantılı dosya adı yazılır okey tıklanırsa **.NDX** uzantılı dosya oluşturulmuş olur. Benzer şekilde başka **.NDX'**lerde oluşturulabilir. İndex'de değişiklik isteniyorsa **Modify** sekmesi kullanılır. Silmek için ise **Erase** butonu kullanılır.

.MDX uzantılı index kullanılmak isteniyorsa **Maintained** seçili iken index alanı seçilir. Okey onaylanır. **Save İndex As** gelir. **Index Tag Name** metin kutusu gelir. de birden fazla alana göre index bilgileri tutulur. Bu index'lerin her birine **tag** denir. **.MDX** Ayrıca veritabanı dosyasının adı ile index dosyasının adı aynıdır. İndex hazırlandıktan sonra dosya için dil seçilir. Bunun için de **Table Properties** liste kutusundan **Table Language** seçeneği seçilir. Table language seçili iken modify tıklanırsa dil liste kutusu gelir.

8. **Save As** butonu ile veritabanı dosyasına isim verilir. Yol belirlenir. **Alias** seçeneğinden oluşturulan alias (örnek :delphibook) seçilir.

9. **Alias** seçimi yapmak zorunlu değildir.

10. Database dosyası kayıt edildikten sonra **Database Desktop** penceresine dönülür.

11. Burada hazırlanan veritabanı dosyası açılabilir. Bilgi girilebilir, değişiklik yapılabilir.

12. Kayıt girişi veya düzeltme için **table** menüsünden **Edit Data** seçilmelidir. Kayıtlar girilebilir.

13. Yapıda değişiklik varsa **tools** menüsünden **utilities** sonrada **Restructure** seçilir. Burada **select file** ‘dan seç tıklanırsa kayıt yapısı gelir. Değişiklik yapılabilir.

14. Ayrıca delphi’de veritabanı dosyaları hakkında bilgi edinmek veya değişiklik yapmak için **Database Explorer** programı vardır. (Başlat, programlar, delphi, SQL explorer) buradan yararlanılarak alias hazırlanabilir veya silinebilir.

SQL Explorer da

- Soldaki table çift tıklanırsa sağda tablo özellikleri gelir.
- Data sekmesi tıklanırsa kayıtlar güncellenebilir.

VERİTABANI BİLEŞENLERİ

Post ve Cancel Metotları: Olan bir kayıt üzerinde değişiklik yapıldığı zaman yapılanlar direkt diske yazılmazlar. Geçici alanda tutulurlar.Yapılan değişikliklerin fiziksel olarak diske yazılması isteniyorsa **post** metodu kullanılmalıdır.Yapılan değişikliklerden vaz geçilmiş ise **cancel** metodu kullanılmalıdır.

```
Table1.post;
Table1.cancel;
```

Kayıt Ekleme Ve Silme: Yeni kayıt için **append** ve **insert** metotları kullanılır. Append metodu dosya sonuna kayıt ekler. Insert üzerinde bulunulan kayıttan önce boş kayıt ekler. Tablodan herhangi bir kaydı silmek için **delete** metodu kullanılır.

```
Table1.insert;
Table1.delete;
Table1.append;
```

Tablo Alanları İçindeki Değerlere Erişim: Tablo alanlarının içeriklerine erişim için birden fazla yol vardır.

1.metot **Fieldbyname** metodudur.

```
Table1.fieldbyname(‘değişken’).AsString;
```

```
Showmessage(Table1.fieldbyname(‘değişken’).AsString);
```

İçeriği görüntülenecek değişken sayısal ise **AsString** Yerine **AsInteger** veya benzeri durumlarda **AsBoolen, AsFloat, AsDateTime** alır.

```
Showmessage(Inttostr(Tablo1.fieldbyname(‘değişken’)).AsString);
```

2. metot ise **table1[‘değişken’]** seklinde ifade edilir. Bu kullanımda delphi otomatik olarak tip çevirmesi yapar.

```
Showmessage(Table1[‘değişken’];
```

```
Var
i:integer;
Begin
i:=table1[‘sicilno’];
End;
```

Statik Alanlar: Uygulama geliştirmede tablo alanlarına ulaşmada en verimli yöntem onları statik olarak tanımlamaktır.Normalde Delphi alanlara , onları nesne örnekleri içine paketleyerek ulaşır. Uygulama geliştirme aşamasında alanları nesne örnekleri haline getirirsek programın çalışması anında alanları **Fieldbyname** ile aranmasına gerek kalmayacaktır. Kod hızlı çalışacaktır.

Statik alanlar oluşturmak için;

- **Table1** nesnesi form üzerinde iken sağ tıklanır,
- Menüden **Field Editör** seçilir,
- Boş pencere gelir,
- Burada sağ tıklanır,
- Gelen menüden **Add Fields** seçilir,
- Alanlar listesi gelir,
- Tamamı veya istenen alanlar seçilir,
- Pencere kapatılır.

Bu esnada ilgili unit içinde kod başlangıcına alanlar tanımlanmıştır.

```
Table1musno:TintegerField;
Table1musad:TstringField;
```

Alanları statik tanımlama hem hızı artıracaktır hem de kullanıcının kontrolunu artıracaktır.

Tablolar Üzerinde Arama Yapmak: Belli bir kayıt üzerinde konumlanmanın birden fazla yolu vardır. Bunlar FindKey, FindNearest, Locate ve Lookup metotlarıdır. Ancak FindKey ve FindNearest yavaş yavaş terkedilmektedir.

FindKey ve FindNearest: Bu metotların kullanılması için tablo indeksli olmalıdır.Ayrıca bulunacak **FindKey** metodunda kayıt bilgisi tam girilmelidir. **FindNearest** metodunda ise tam yazılmasa konumlanma sağlanır.

```
Begin

Table1.FindNearest([editara.text]);
End;
```

```
Var
Son:boolean;
Begin
 Son:=form3.table2.fındkey([form3.dbedit3.text]);
End;
```

Locate: Delphi 2.0 la birlikte gelen metotlardır.

```
Var
Bulundu:Boolen;
Begin
Bulundu:=Table1.locate('alanadı','bulunacak ifade',[lopartialkey]);
End;
```

3. Parametre iki değer alabilir.

[lopartialkey] : Aranılan tam bulunmasa bile en yakın değere konumlanır.

[locaseinsensitive] : Arama esnasında büyük/küçük harf ayırımı yapılmaz.

İki seçenek birden kullanılacak ise **[lopartialkey,locaseinsensitive]** şeklindedir.
Aranılan bulundu ise true değilse false değeri döndürülür.
Aranılan kayıt bulunduysa ona konumlanılmıştır.

Lookup : Bu metot aranılan kayıt bulunsada oraya konumlanmaz. Kayıt bulunmuş ise son parametrede gönderilen sütunların değerini döndürür.

```
Var
Sonuclar:variant;
Begin
Sonuclar:=table1.lookup('sicil,bolum',
      Vararrayof((2453,'uretım'),'sehır;ulke');
If sonuclar<> NULL
  Then begın
        Showmessage(sonuclar[0]);
        Showmessage(sonuclar[1]);
  End;
 End;
```

Lookup metodu 3 parametre alır. İlk parametre hangi sütun üzerinde arama yapılacaksa onun adı,ikinci parametre aranılacak değerleri ücüncü parametre ise işlem olumlu sonuçlandıysa bu fonksiyon metodunun döndüreceği alan değerleridir.

İlk ve son parametre için birden fazla alan adı kullanılabilir. Yukardaki örnekte olduğu gibi ifade edilir.

Bu metot variant tipinde sonuç döndürür. Variant yerine variant dizilerde kullanılabilir. Her alan variant dizinin bir elemanı olacak şekilde dizi içine atanır.

BookMark'lar: Herhangi bir veri kümesi (dataset) üzerinde çalışırken sık karşılaşılan sorunlardan birisi bir kaydın üzerinde iken başka bir kayda gitmek orada bazı işlemler yapıp tekrar önce durulan kayda dönmektir.Delphide durulan kayda işaret konur tekrar dönüşte bu işaret kullanılır.

```
Var
Oncekikayıt:Tbookmark;
Begin
Öncekikayıt:=table1.getbookmark;
Table1.last;
.......
........
table1.gotobookmark(öncekikayıt);
table1.freebookmark(öncekikayıt);
end;
```

Filtered Öze n kullanılır.

Örneğin 1900 ile 2200 arasındaki fişler listelenmek istensin.

```
Procedure Tform1.FormCreate(Sender:Tobject);
Begin
Table1.Filtered:=true;
Table1.open;
End;
Procedure Tform1.FormClose(sender:Tobject,var action:Tclose action);
Begin
Table1.close;
End;
Prosedure Tform1.table1.filterrecord(dataset:Tdataset;
Var accept:boolean;
Begin
İf (table1fisno.asinteger > 1900)
     And (table1fisno.asinteger<2200)
  Then accept :=true
  Else accept:=false;
End;
```

Filtered özelliği ve Findfirst,FindNext, FindPrior, FindLast metotları: Bu özellikle kullanılan dört metot vardır. Bu metotlar kullanılırken filtered özelliğinin **false** olması gerekir.

```
Procedure Tform1.FormCreate(Sender:Tobject);
Begin
Table1.Filtered:=false;
Table1.open;
End;
```

SetRangeStart, SetRangeEnd ve ApplyRange Metotları : Filtreleme dışında bazı kayıtları seçmek için kullanılan metotlardır.

```
Procedure Tform1.btnFiltreKoy(Sender:Tobject);
Begin
Table1.setRangeStart;
Table1['fisno']:=1700;
Table1.setRangeEnd;
Table1[fisno']:=2200;
Table1.ApplyRange;
End;
```

- Belirlenen aralığı kaldırarak tüm kayıtların aktif hale gelmesi için;

```
Table1.SetRangeStart;
Table1.SetrangeEnd;
Table1.ApplyRange;
```

İfadeleri kullanılır.

- Bu işlem **paradox** ve **dbase** tabloları ile yapılıyorsa ilgili alanların indeksli olması gerekir.

Cache Updates:

Çok kullanışlı özelliklerden biridir. Bu mantıkta belli bir işlem yapılacağında yapılan değişikliklerin tümü geçici bir tampon alanda tutulabilir. Daha sonra kullanıcı onaylarsa değişiklikler tabloya yazılır onaylamaz ise çağrılan tek bir metotla işlemler geri alınabilir.
Örnek olarak form üzerine 3 adet buton ve dbgrid yerleştirelim. Grid üzerinde yaptığımız değişiklikler için işlem öncesi ve sonrası ilgili butonları uygulayalım.

```
Procedure Tform1.FormCreate(Sender:Tobject);
Begin
Table1.open;
End;
Procedure Tform1.FormClose(sender:Tobject,var action:Tclose action);
Begin
Table1.close;
End;
ProcedureTform1.btnCacheclick(sender:Tobject);
Begin
Table1.CachedUpdates:=true;
End;
ProcedureTform1.btnIptalclick(sender:Tobject);
Begin
Table1.CancelUpdates;
End;
ProcedureTform1.btnKaydetclick(sender:Tobject);
Begin
Table1.ApplyUpdates;
Table1.CommitUpdates;
```

OldValue ve NewValue Özellikleri:

Bu özellikler sayesinde , bir tablonun herhangi bir alanı üzerinde değişiklik yaptıktan sonra önceki değerinin ne olduğunu **OldValue** ile ,son değerinin ne olduğunu **NewValue** ile görebiliriz. Her ikiside Variant tipinde değer gönderir. Bunların kullanılabilmesi için **CachaUpdates** özelliğinin **True** olması gerekir.Aksi halde program kırılır.

```
ProcedureTform1.btnOldValuelclick(sender:Tobject);
Begin
Showmessage(table1.adres1.OldValue);
End;
ProcedureTform1.btnNewValueclick(sender:Tobject);
Begin
Showmessage(table1.adres1.NewValue);
End;
```

Master-Detay Taablolar:

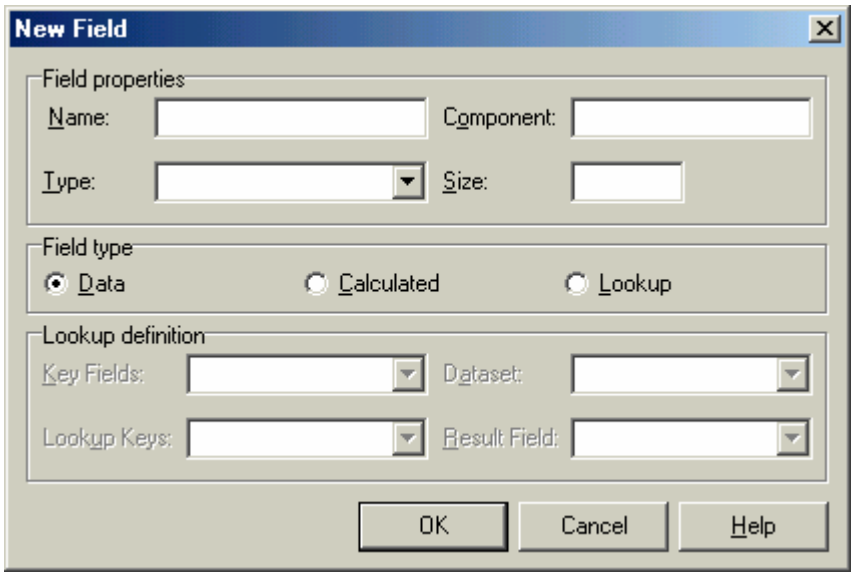
Bu tür tablolarda bir tablodaki bir alanla diğer tablodaki alan arasına ilişki vardır. Bu ilişki **bire-çok veya çoka-bir** olabilir. Bu ilişki sayesinde ilk tablo üzerinde herhangi bir kayıt üzerinde iken ikinci tablo sadece birinci tablodaki aktif kayda bağlı olacak şekilde kendi kayıtlarını filtre eder.

Bu işlem için her iki tablonun ilişkili hali genellikle aynı form üzerinde olur. Form üzerine iki tablo içinde tanımlar yapıldıktan sonra bağlantı için **Object İnspector**'dan **MasterSource ve MasterField** özellikleri kullanılır.

İki tablo arasında **master-detay** ilişki kurulacaksa ,detay tablosunda ilişkiyi sağlayacak olan sütun üzerinde index olması gerekir.

Hesaplanan Alanlar:

Herhangi bir veri kümesi üzerinde çalışırken tabloda olmayan ama programın çalışması sırasında hesaplanabilecek ek bir sütun istenebilir. Böyle bir sütunu eklemek için form üzerinde table nesnesi aktif iken sağ tıklanır. Gelen menüden **Fields Editör** seçilir. Tekrar sağ tıklanır. Gelen menüden **New Field** seçilir.Gelen ekrandan ;



Name: Sütun adı girilir
Type: Listeden tip seçilir.

Tanımlanan sütunun formülü için; form üzerinde table1 nesnesi seçili iken **Object İnspector**'dan events sekmesinden **OnCalcFields** olayına girilir.Buraya formülü ifade eden Delphi kodu yazılır.

TTable Bileşenindeki Önemli olayların listesi:

AfterCancel: Ttable bileşeninin **Cancel** metodu çağrıldıktan sonra oluşur.
AfterClose: Ttable bileşeninin **Close** metodu çağrıldıktan sonra oluşur.
AfterDelete: **Delete** metodu çağrıldıktan sonra oluşur.
AfterEdit:Tabloya bilgi girişi yapıldığı anda oluşur.
AfterİnsertTabloya **Append** ve **İnsert** metodu ile kayıt eklendikten sonra oluşur.
AfterOpen: Tabloyu açtıktan sonra oluşur.
AfterPost: Tablonun **Post** metodu çağrıldıktan sonra oluşur.
BeforeCancel:

BeforeClose:
BeforeDelete:
BeforeEdit:
BeforeInsert:
BeforeOpen:
BeforePost:

Herhangi bir durumda bu olaylar içine yazılacak kodun kontrolü sırasında asıl gerçekleşmesi gereken işlemin olmamasını istiyorsak o zaman bir ayrıcalık **(exception)** yaratmamız gerekir.

```
Procedure Tform1.table1beforeinsert(dataset:Tdataset);
Begin
Raise Exception.Create('yeni bir kayıt eklenemez !');
End;
Procedure Tform1.btnYeniClick(Sender:Tobject);
Begin
Table1.Append;
End;
```

VERİ TABANI DOSYALARI ÜZERİNDE İŞLEM YAPMAK

Veri tabanı dosyaları üzerinde işlem yapmak için project menüsünde **DATAACCESS** sekmesi olmalıdır. Bunun için ise işlem yapmaya yönelik 10 düğme vardır.

Diske kayıt edilen bir veritabanı dosyasının içeriğini görüntülemek veya dosyaya yeni kayıtlar eklemek için ilk gerek duyulan kontrolün adı **table**'dır.

Yapılacak işlemleri sıralarsak ;

1. Table nesnesi forma yerleştirilir.

2. Table nesnesi seçili iken **Object İnspecter**'dan özellikler değiştirilir.

3. Database Name'den **Alias** seçimi yapılır. (delphibook) Eğer alias seçimi yapılmaz ise aynı alana veritabanı dosyasının saklandığı klasörün adının Dos'un yol kurallarına göre yazılması gerekir.

4. Veri tabanı dosyası seçilmesi gerek. **Table Name**'den veri tabanı dosyası seçilir. (ADRES.DBF)

5. Table nesnesine ait **Active** özelliğine **True** değeri aktarılır.

6. Mutlaka yapılması gereken işlemler tamamlanmış oldu.

7. Table nesnesine ait index name geldiğinde **index** seçimi yapılır. (SOYAD)

8. Forma table kontrolünden başka **Datasource** kontrolüde yerleştirilmelidir.

9. Datasource nesnesi eklendikten sonra özelliklerde değişiklik yapılabilir. **Name**'den isim verilir. (Datasource default olarak datasource1 adı verilir. Gerekirse değiştirilebilir.)

10. Dataset özelliğine table nesnelereinden biri aktarılır. (Table1)
    a. Proje çalıştırıldığında veritabanı dosyası açılır,okuma kafası ilk kayıta konumlanır.
    b. Table ve Datasource nesneleri ve özellikleri tanıtıldıktan sonra ek tanımlamalar başlar.

11. Bunun için **Data Kontrol** sekmesi kullanılır. (Buradaki kontroller standart sekmesindeki kontrollere paraleldir. Edit-DBEdit v.b.)

12. **DBEdit** her alan için yerleştirilir. (Dbedit1) Her DBEdit nesnesine hangi Datasource2den bilgi alacağı belirlenir. **Datasource1** seçeneği seçilir.

13. Her alan için **Datafield** 'den alan seçilir.

14. DBEdit'ler için **Label'**lar eklenir.

15. Komut butonları eklenir. Bu aşamaya kadar kod kullanılmadı. Sadece sonraki kayıt butonu için buton seçili iken **Click** yordamına **Next** butonu eklenir.

```
Begin
        Table1.next;
End;
```

16. Önceki kayıt butonunun yordamına **Prior** deyimi eklenir. Bir önceki kaydı aktif kayıt yapar.

```
Begin
        Table1.prior;
End;
```

NOT: Navigator yerine komut butonu yerleştirilerek kütük üzerinde işler sağlanacak ise kullanılacak olan unit yordamları aşağıdaki şekilde olmalıdır.

Table1.append;
Table1.insert;
Table1.delete;

Table1.last;
Table1.next;
Table1.prior;
Table1.first;

NAVIGATOR KONTROLÜ

Kayıtlar arasında dolşmak veya işlem yapmak için buton eklenebileceği gibi navigator kontrolüde kullanılabilir.

Bunun için;

1. **DATACONTROLS** sekmesinden **DBNavigaor** düğmesi forma yerleştirilir.Varsayım olarak DBNavigator1 adını alır.

2. Özelliklerden **DATASOURCE** nesnesinden önceden tanımlı olan Datasource1 seçeneği seçilir. Bu işlemden sonra table, datasource, DBEdit, navigator işlemleri tanımlanmış oldu. Böylece proje çalıştırılabilir.

   Navigator'daki nesnelere tıklandığında işlevleri yazı ilede gösterilsin isteniyorsa;
   - Navigator'ın **Showhint** özelliğine **True** değeri aktarılır.
   - Hints özelliği çift tıklanırsa **StringListEditör** diyalog kutusu gelir. Her satıra bir nesne açıklaması yapılır. (Burada orijinal liste vardır. Gerekirse türkçeleştirilebilir.)

3. Navigator'daki nesnelerden bazıları görünmesi isteniyorsa nesnenin **VisibleButtons** özelliğinden yararlanılır. Çift tıklandığında her düğmenin durumu listelenir.Çıkarılacak olan nesnenin durumu **False** yapılır.

KAYIT ARAMA İŞLEMLERİ

GotoKey, GotoNearest, FindNearest Metotları

SetKey ve GotoKey metotları ile arama işlemi:

SetKey ve GotoKey indeks dosyalarından yararlanılarak kullanılır.

Bu işlem için;

1. Forma indeks için edit alanı ve ara butonu eklenir.
2. Buton prosedürüne ;

```
Begin
        Table1.setkey;
         Table1.IndexFields[0].asstring:=Edit1.text;
        Table1.gotokey;
End;
```

3. Çalıştırılır. Eğer ikinci bir indeks var ise onun üzerinden arama yapılacak ise bu kod satırında belirtilmelidir.

```
Begin
        Table1.IndexName:=’2.anahtar’;
        Table1.SetKey;
        Table1.IndexFields[0].asstring:=Edit1.text;
        Table1.GotoKey;
End;
```

Not: SetKey ve GotoKey metodunda index bilgisi tam girilmelidir.

GotoNearest Metodu ile arama işlemi:

Arama adı tam hatırlanmadığı durumlarda kullanılır. Mevcut bilgilere göre arama yapar.Ancak bu tür aramalardan sonra kullanıcıdan mutlaka onay almak gerekir. Bunun için geliştirilecek kod satırı aşağıdaki şekildedir.

```
Begin
        Table1.IndexName:=’2.anahtar’;
        Table1.SetKey;
        Table1.IndexFields[0].asstring:=Edit1.text;
        Table1.GotoNearest;
End;
```

NOT: 2.anahtar ise yazılır.
Indexfields[] yerine **Fieldbyname** de kullanılabilir.

FindNearest Metodu ile arama yapmak:

Gotokey, GotoNearest metotları ile aynı işleve sahiptir.
Bunun için geliştirilecek kod ortamı aşagıdaki şekildedir.

```
Begin
                Table1.FindNearest([edit1.text]);
End;
```

Eğer arama ikinci indekse göre ise kod satırı aşağıdaki şekildedir.

```
Begin
                Table1.IndexName:=’2.indeks’;
                Table1.FindNearest([edit1.text]);
End;
```

Locate Metodu ile Arama Yapmak:

Bu metotda indekse gerek duyulmaz. Bunun için hazırlanan yordam aşağıdaki şekildedir.

```
Begin
        Table1.IndexName:=’ ’;
        Table1.Locate(‘stokkodu’,edit1.text,[]);
End;
```

Burada indekse boşluk taşınarak devre dışı bırakıldı.Yalnız kullanılan alan bir indeks alanı ise Locate metoduda indeks alanını kullanabilir. Aşağıdaki gibi ifade edilir.

```
Begin
        Table1.IndexName:=’indeks değişkeni ’;
        Table1.Locate(‘stokkodu’,edit1.text,[]);
End;
```

KAYITLARI FİLTRE ETMEK

Belli özelliğe sahip kayıtlar gerekiyorsa kayıtları filtre etmek gerekir. Kayıtlar filtre edilirken indeks dosyalar gerek duyulmaz. Bunun için **Table** nesnesinin **Filter**, **Filtered** özelliğinden yararlanılabilir veya direkt olarak **SQL** kodları kullanılabilir.

Bunun için;

1. **Table** özelliklerinden **Filter** seçeneğine filtre edilecek kayıt girilir.Belirtilen kayıt getirilir.

```
Filter:stokkodu:’ARG110’
```

2. **Table** nesnesinin **Filtered** özelliğine **True** değeri aktarılır. (* karakteride kullanılabilir.)

```
Filter:stokkodu:’ARG*’
```

3. **FilterOption** özelliğinin bir alt seçeneği olan **FoNoParialCompare=False** olmalıdır.

4. Büyük/küçük harf ayırımı istenmiyorsa **FoCaseInsensitive=True** olmalıdır.

Bu işlemler yerine kod yazarakta yapılabilir. Bunun için forma **süz** düğmesi eklenir. Kod satırı aşağıdaki şekilde ifade edilir.

```
Begin
        Table1.Filter:=’stokkodu=”ARG110”’;
        Table1.Filtered:=True;
        Table1.FilterOptions:=[FoCaseInsensitive];
End;
```

Yukarıdaki kod satırında **ARG110** yazılabileceği gibi **ARG*** ifadeside geçerlidir.

QUICK REPORD İLE RAPOR HAZIRLAMAK

Quick report sekmesinde bulunan kontrolleri kullanarak raporlar hazırlanabilir.
Bu işlem için;

1. Proje formuna **TABLE** nesnesi yerleştirilir. (Rapor hazırlarken **Table** yerine **QUERY** kontrolü yerleştirilir.)

2. Table özelliklerinden **database** ve **tablename** özellikleri kullanılarak **Alias** ve **Tablo** seçimi yapılır. (Tablo yerine **Query** kontrolü kullanılıyorsa tablo ve alan seçimi için **SQL** kodu hazırlanmalıydı.)

3. Forma **Quick Report** kontrolü yerleştirilir. (Formun boyutları otomatikman büyür ve kaydırma çubukları eklenir.)

    - Aynı işlem quick report nesnesini yerleştirmek yerine ; **File, New** seçeneği ile **New Items** diyalog kutusundan **New** sekmesinde bulunan **Report** seçilmekle de olur.

4. **Quick Report** nesnesine varsayım olarak **quickreport1** adı verilir.

5. **Dataset** seçmesi yapılır. (table adı aktarılır.)

6. **Object İnspector'dan** diğer özellikler tanımlanır.(örneğin: page, columns, orientation v.b.)

7. Raporun en üst kısmında rapor başlığı olur. İstenirse kullanılır.

8. Sayfa başlığı kısmında her sayfaya yazılacak sayfa başlığı kullanılabilir.

9. Raporda her satırda yazılacak olan olan ayrıntılar tanımlanır.(değişkenler)

10. Her sayfanın alt kısmında yer alacak olan alt bilgi tanımlanır. (eğer istenirse)

11. Ayrıntıların tanımı için; (değişkenlerin)

    - **QRBand** kontrolü kullanılarak alan kullanımı tanımlanır. (QRBand kontrolü forma taşınır.)

    - **QRBand**'ın **BandType** özelliğine **rbdetail** değeri aktarılır.

    - **QRDBText** kontrolü **Qreport** sekmesinden aktif hale getirilir.

    - **Qband** içinde tıklanırsa **QRDBText** nesnesi banda yerleşir.

- **QRDBText** seçili iken **Dataset** seçilir.
- **Datafield** özelliğinden yararlanılarak alan seçimi yapılır.
- Listelenecek her alan için **, işlem tekrarlanır**.

12. Çalışma anında raporu listelemek veya yazıcıya göndermek gerekir. Bunun içinde ;

13. Yeni bir form eklenir. Buna iki buton yerleştirilir. (rapor ve yazdırma butonları olsun.)

14. Raporu ekrandan görmek için **Preview** metodu yazdırmak içinde **Print** metodu kullanılır.

15. Rapor düğmesi yordamı aşağıdaki şekilde hazırlanabilir.

```
Begin
        Form1.table1.open;
        Form1.Quickrep1.Preview;
        Form1.table1.Close;
end
```

16. Print yordamı hazırlanır.

```
Begin
        Form1.table1.open;
        Form1.Quickrep1.Print;
        Form1.table1.Close;
end
```

17. Rapor çalıştırılarak sonucu görülebilir.

- Rapor belli ölçek ile küçültülüp büyültülebilir.
- Satır genişliği tasarım anında **QRBand** nesnesinin yüksekliği ile paraleldir. Burada **Height** ile değiştirilebilir.
- Sayfa başlığı kullanılacak ise ikinci bir **QRBand** nesnesi eklenir. **Band type** özelliğine **rbPageHeader** değeri aktarılır. Başlık yazımı için **QRLabel** kontrolü kullanılır. O aktif iken başlık yazılır. (caption özelliğine) **Font** değiştirilebilir.
- Sayfa başlığı işi sayfa sonu içinde yapılabilir.

QUICK REPORT LIST İLE RAPOR HAZIRLAMAK

Raporları kendimiz hazırlayabileceğimiz gibi **New Items** diyalog kutusundaki **Quick Report List** ile de yapabiliriz. (Forms içinde)

Bunun içinde;

1. **New Items, Forms,Quick Report List** seçili iken **Ok** tıklanır.
2. Projeye yeni bir form eklenmiştir.
3. Form üzerine listelemede kullanılacak **QuickReport** nesnesi yerleşmiştir.
4. En başta **Alias** ve **Tablo** seçimi yapılmış olmalıdır.
5. Raporda **3 QRBand** nesnesi vardır. Bunları kullanım amaçlarına göre ayarlamak gerekir.
6. Kayıtların içeriği için ilgili **QRBand** nesnesi üzerine **QRDBText** nesneleri yerleştirilir.
7. Projeyi çalıştırmadan raporu görmek için kısayoldan **Preview** komutu kullanılır.

8. Raporun özellikleri hakkında bilgi edinmek için kısayoldan **Report Settings** kullanılır. Gerekiyorsa buradan özellikler değiştirilebilir.

QUICK REPORD WIZARD İLE RAPOR HAZIRLAMAK

Rapor sihirbazı kullanılarak, kısa yoldan rapor hazırlamakta mümkündür.

Bu işlem için;

1. **File, New, NewItems** diyalog kutusu getirilir.
2. Business sekmesinden **QuickRepordWizard** seçilir.
3. Ekrana Wizard diyalog kutusu gelir**. Start'a** basılır.
4. Ekrana gelen diyalog kutusundan **Alias, Table** seçilir.
5. **Available Fields** liste kutusundan alanlar seçilir.
6. **Son** (finish) düğmesine basılır.
7. Çalıştırılır.

- Gerekirse üzerinde düzeltmeler yapılabilir.

QUICK REPORT LABELS İLE ETİKET HAZIRLAMAK

Bunun için;

1. **File, NewItems, QuickReport Labels** simgesi seçilir.
2. Çalışılan projeye yeni bir form eklenmiştir. Etiket görünümü için ilgili **bant** gelir.

3. **Table** nesnesi de forma eklenmiştir.

4. Table nesnesi için **Alias** ve **Tablo** seçilir. (özelliklerden)

5. Etiket bilgileri için **QRDBText** nesnesi yerleştirilir.

6. Bilgiler için **Dataset, Datafield** seçimleri yapılır.

7. Proje çalıştırılmadan kısayoldan**, Preview** ile çalışması sağlanabilir. Gerekirse düzeltme yapılır.

8. Proje çalıştırılır.

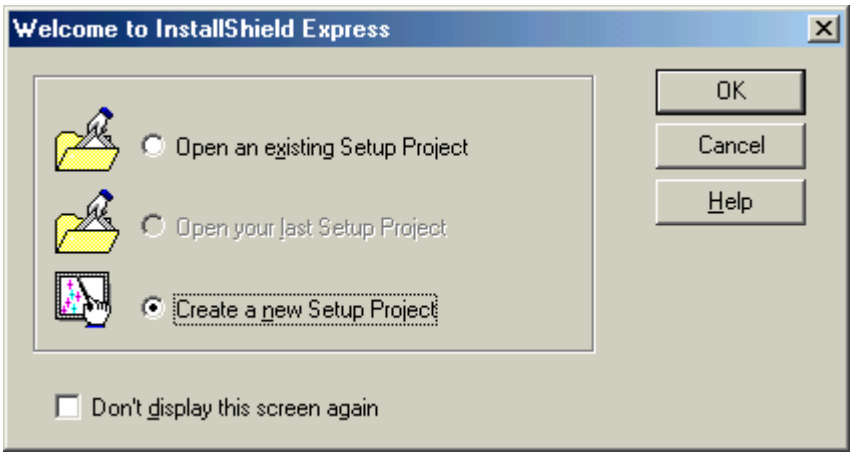
DERLEME VE KURMA DİSKETİ HAZIRLAMA

- Hazırlanan proje **Run** komutu ile çalıştırıldığında tüm bileşenleri derlenir ve **.EXE** dosya hazırlanır. Ayrıca her unit için **.DCU** uzantılı dosyalarda oluşturulur.

- Kurma disketleri hazırlayabilmek için Delphi ile birlikte gelen **InstallShield Express** adındaki programı kullanmak gerekir.Bu programın diske kurulması gerekir.Kurulduktan sonra baslat menüsünde **InstallShield Express**'i temsil eden bir seçenek eklenir.

KURMA DİSKETİ HAZIRLAMAK İÇİN;

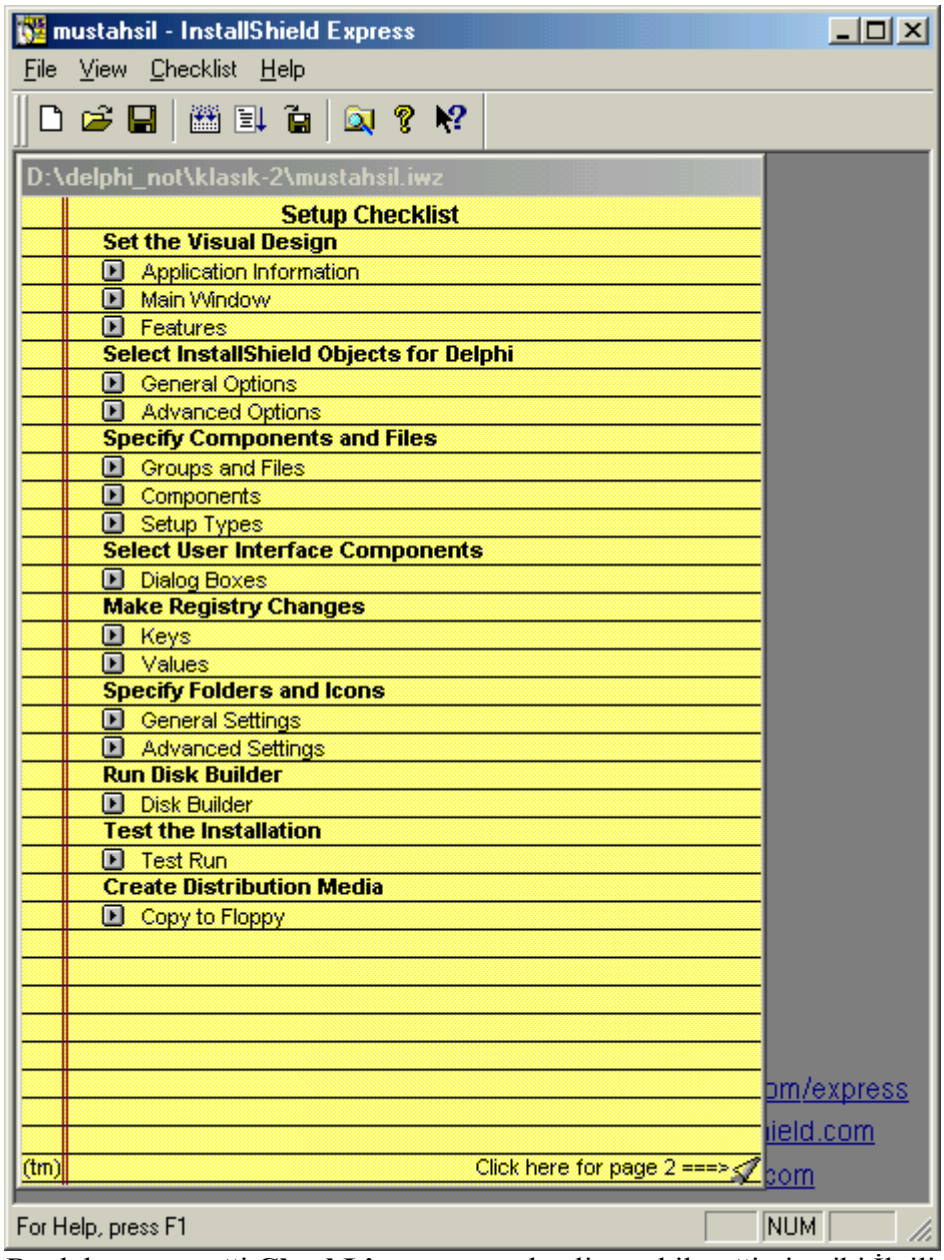
1. **InstallShield Express** çalıştırılır. Ekrana ilgili pencere gelir.



Bu program kullanılarak yapılan ayarlar ve düzenlemeler **.IWZ** uzantılı dosyalara yazılır. Bu ekranda önceden oluşturulan dosya açılıp yeniden düzenlenebilir veya ilk defa oluşturulabilir.

2. İlk defa oluşturma seçilmiş ise hazırlanacak **IWZ** uzantılı dosyanın adının belirtildiği **New Project** ekranı gelir. Adı ve bulunacağı klavuz belirlendikten sonra **Create** butonu ile oluşturulur.

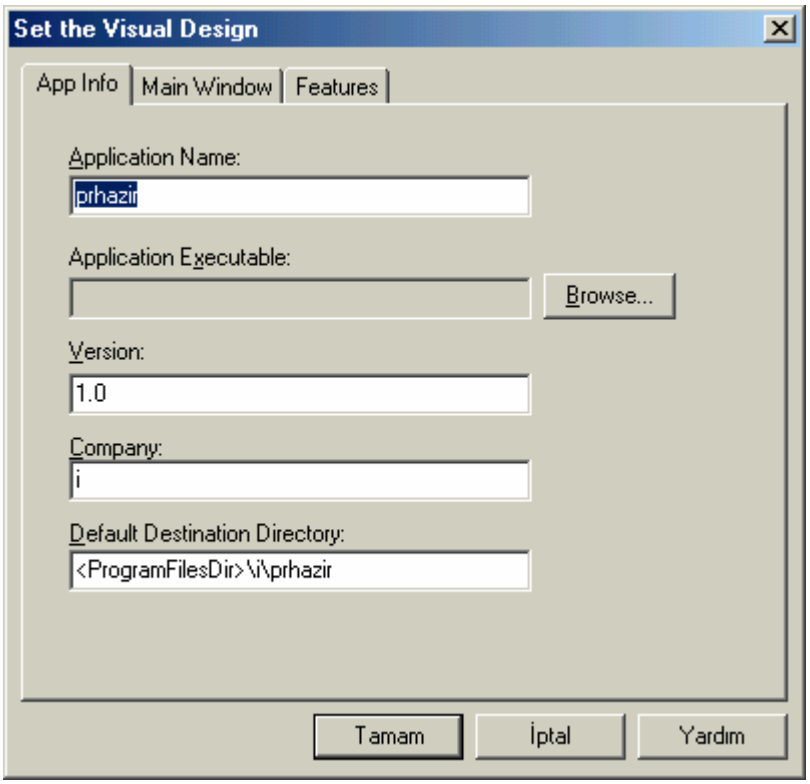
3. **IWZ** uzantılı dosya oluşturulduktan sonra çok seçenekli **InstallShield Express Limited Edition** ekranı gelir.



Bu dokuz seçeneği **CheckList** menüsünden listeyebileceğimiz gibi İlgili satır seçilerekte kullanılabilir. Gerekli tanım ve özellikler için bu seçenekler kullanılır.

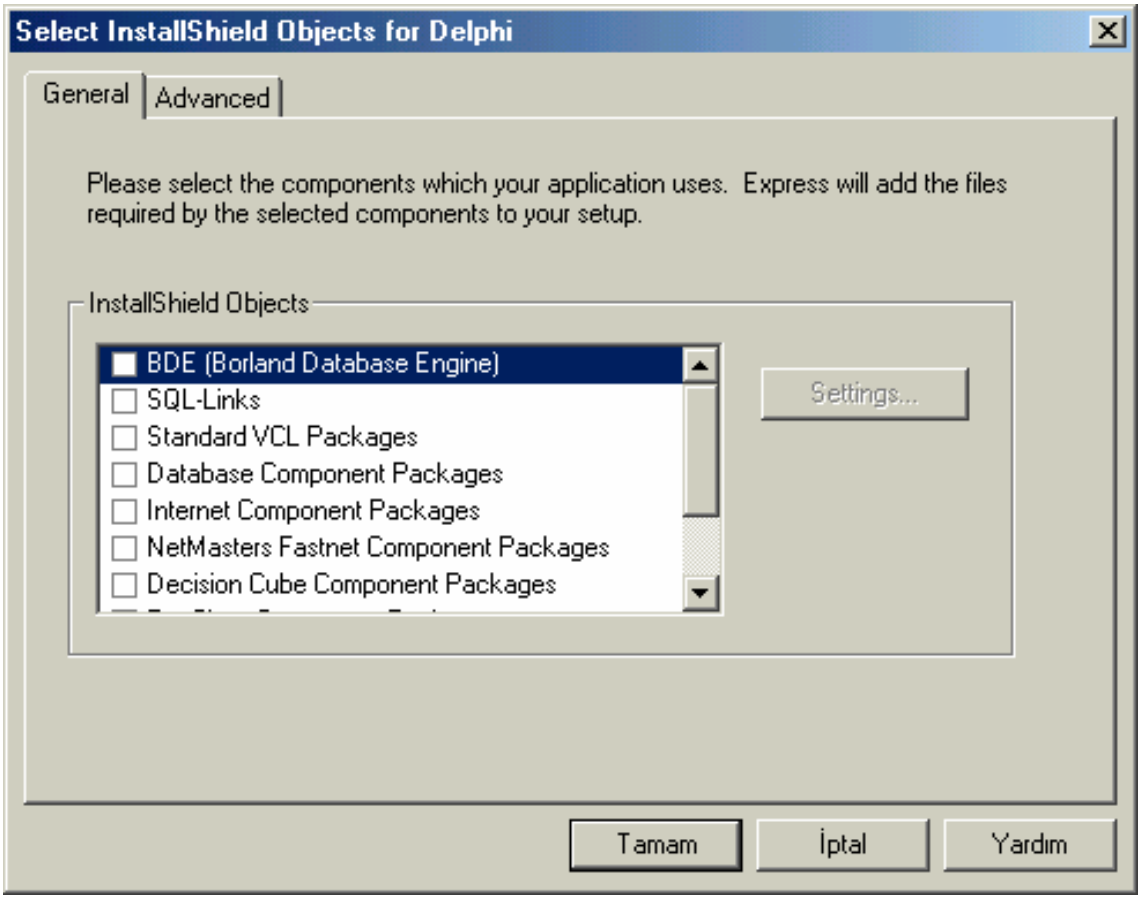
4. Set The Visual Design seçeneği;

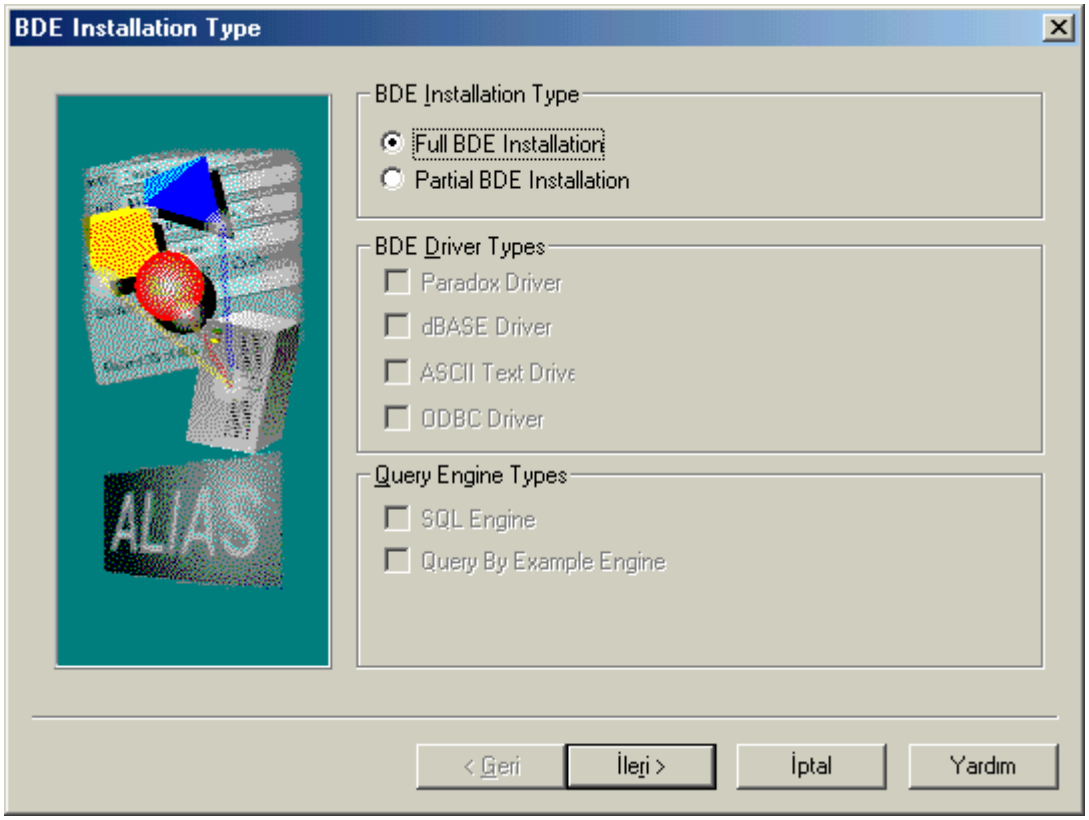
Bu seçenek **App Info, Main Window, Features** sekmelerinden oluşur. **App Info** seçeneğinde projenin adı ve versiyon numarası bulunur. **Main Window** sekmesinde ana pencere başlığı ve logo seçimi yapılır. **Feature** sekmesinde ise onay kutusu vardır. Bu onay kutusu seçili iken **UNİNSTALL** özelliği kurma disketine kazandırılmaktadır**.**

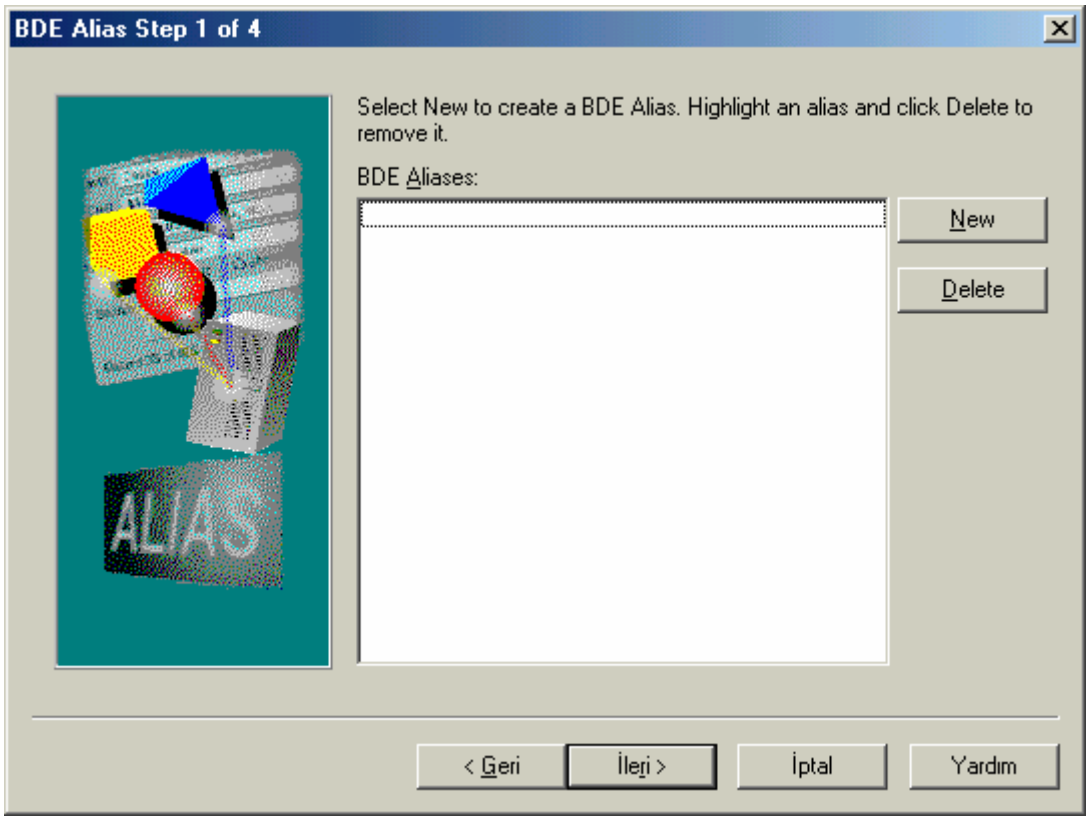


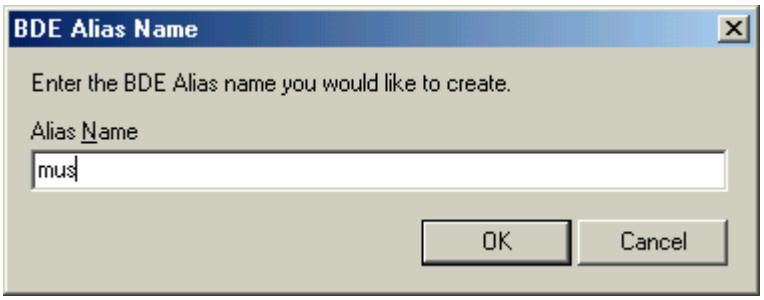
5. Specify InstallShield Objects For Delphi 5 seçeneği;

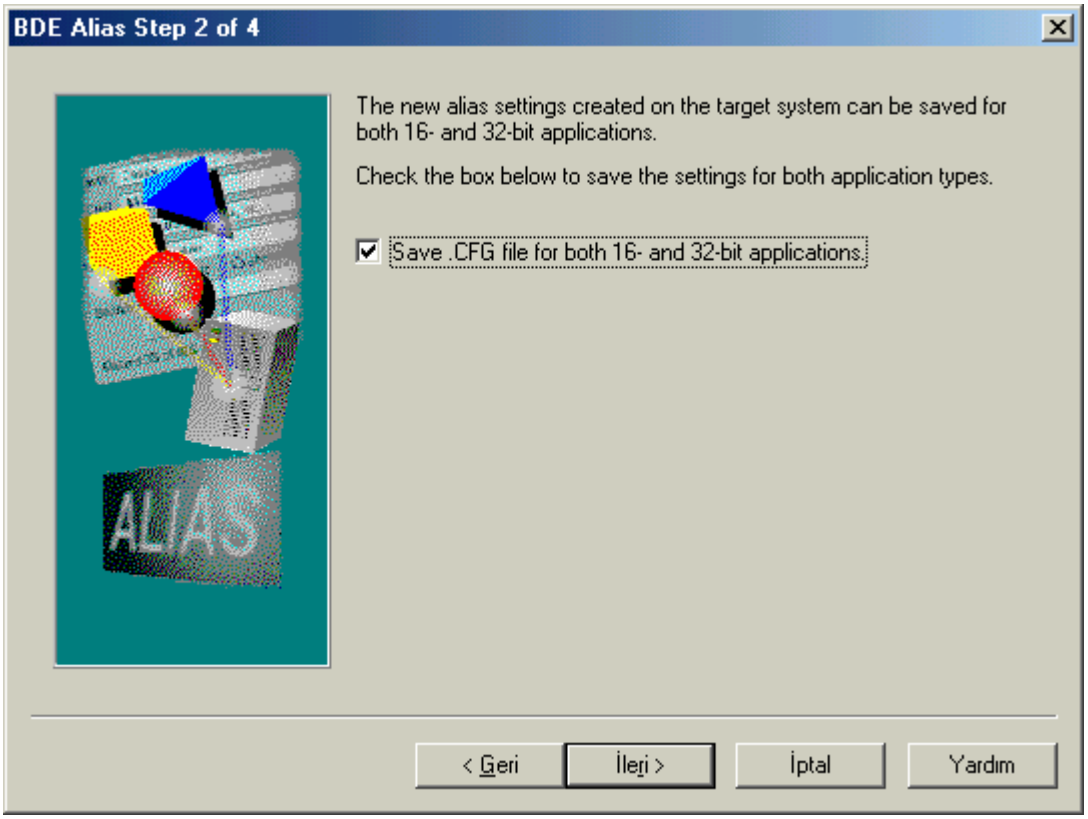
Bu ekranda **General** ve **Advanced** adlı iki sekme bulunmaktadır. **InstallShield** hazırlanan projede database kullanılıyorsa **BDE** seçeneği işaretli olması gerekir. **BDE** onay kutusu işaretlendiğinde ise **BDE Installation Type** başlıklı ekran görüntülenir. Burada **BDE** ile ilgili sorular vardır. Kurma disketine **BDE** ile ilgili tüm detayların kopyalanması isteniyorsa **1. Radio** buton değilse diğerleri kullanılır. **BDE** seçildikten sonra ekrana **Alias** seçimi ile ilgili ekran gelir.Burada alias ayarları yapılır.2. alias ekranındaki onay kutusu seçilirse DBE ile ilgili yapılan ayar ve düzenlemeler **CGF** uzantılı dosyaya da kayıt olur. 3. ve 4. Alias ekranı tanımları da yapıldıktan sonra işlemler sona erdirilirse tekrar **Specify Installshield Objects For Delphi 5** ekranına tekrar dönülür. Buradaki ikinci sekme olan Advanced sekmesinde ise disketlere kopyalanacak dosyalar hakkında seçimler bulunur.

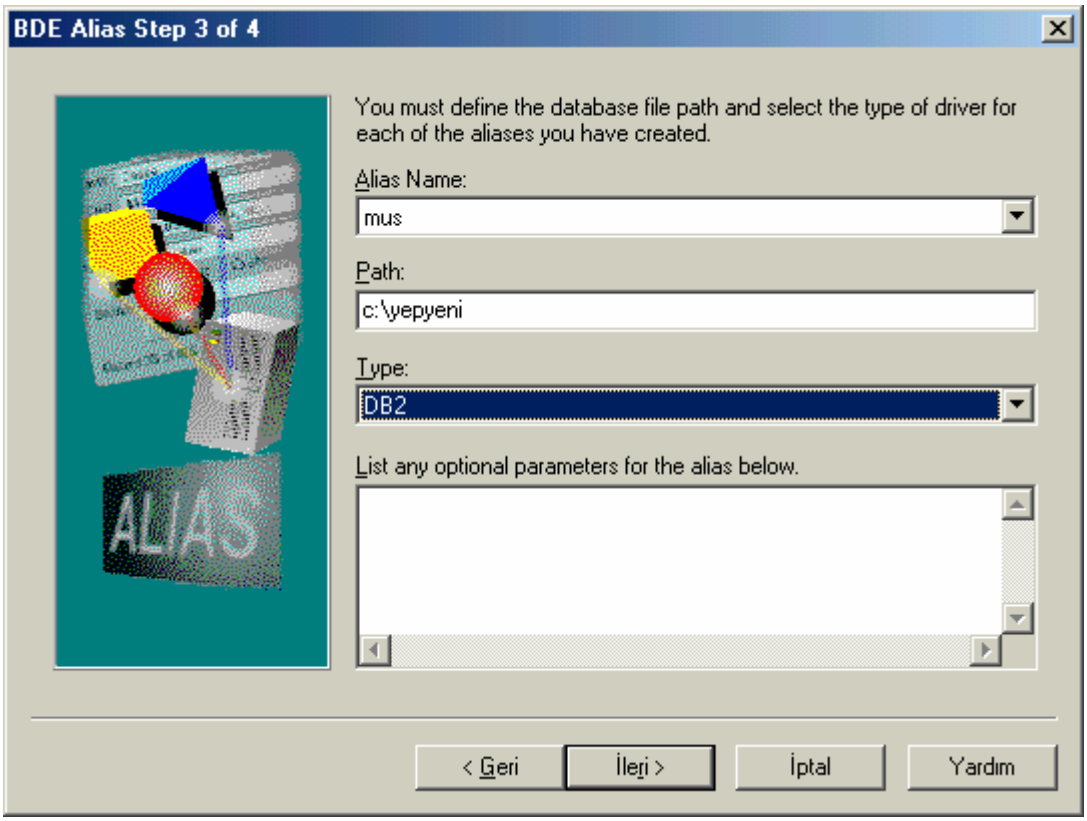

Select InstallShield Objects for Delphi
General
Advanced
Please select the components which your application uses. Express will add the files required by the selected components to your setup.
InstallShield Objects
BDE (Borland Database Engine)
SQL-Links
Standard VCL Packages
Database Component Packages
Internet Component Packages
NetMasters Fastnet Component Packages
Decision Cube Component Packages
Settings...
Tamam
İptal
Yardım

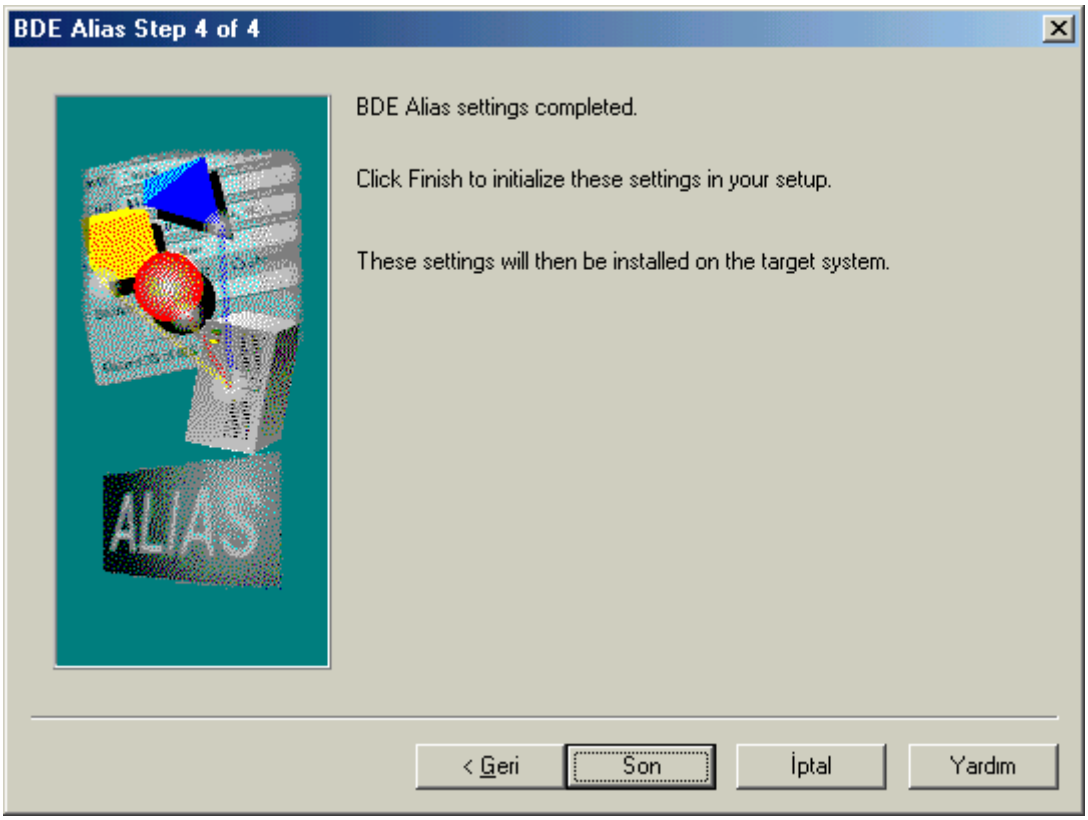

BDE Installation Type
BDE Installation Type
Full BDE Installation
Partial BDE Installation
BDE Driver Types
Paradox Driver
dBASE Driver
ASCII Text Drive
ODBC Driver
Query Engine Types
SQL Engine
Query By Example Engine
< Geri
İleri >
İptal
Yardım

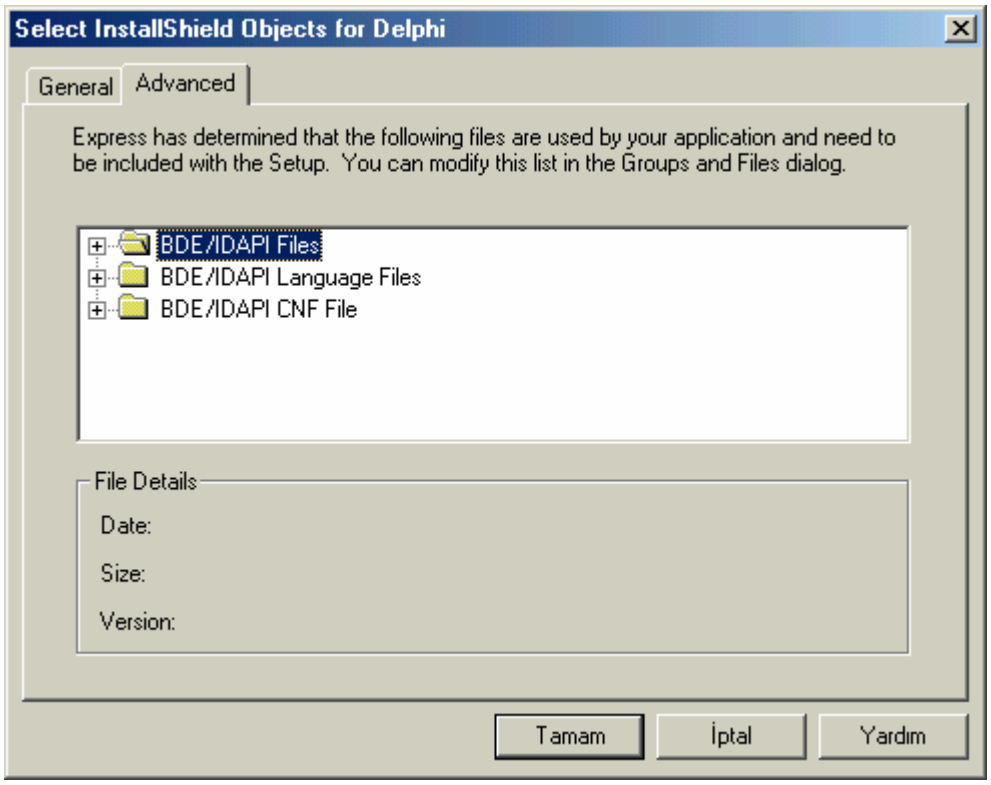

BDE Alias Step 1 of 4
Select New to create a BDE Alias. Highlight an alias and click Delete to remove it.
BDE Aliases:
New
Delete
< Geri
İleri >
İptal
Yardım


BDE Alias Name
Enter the BDE Alias name you would like to create.
Alias Name
mus
OK
Cancel


BDE Alias Step 2 of 4
ALIAS
The new alias settings created on the target system can be saved for both 16- and 32-bit applications.
Check the box below to save the settings for both application types.
Save .CFG file for both 16- and 32-bit applications.
< Geri
İleri >
İptal
Yardım

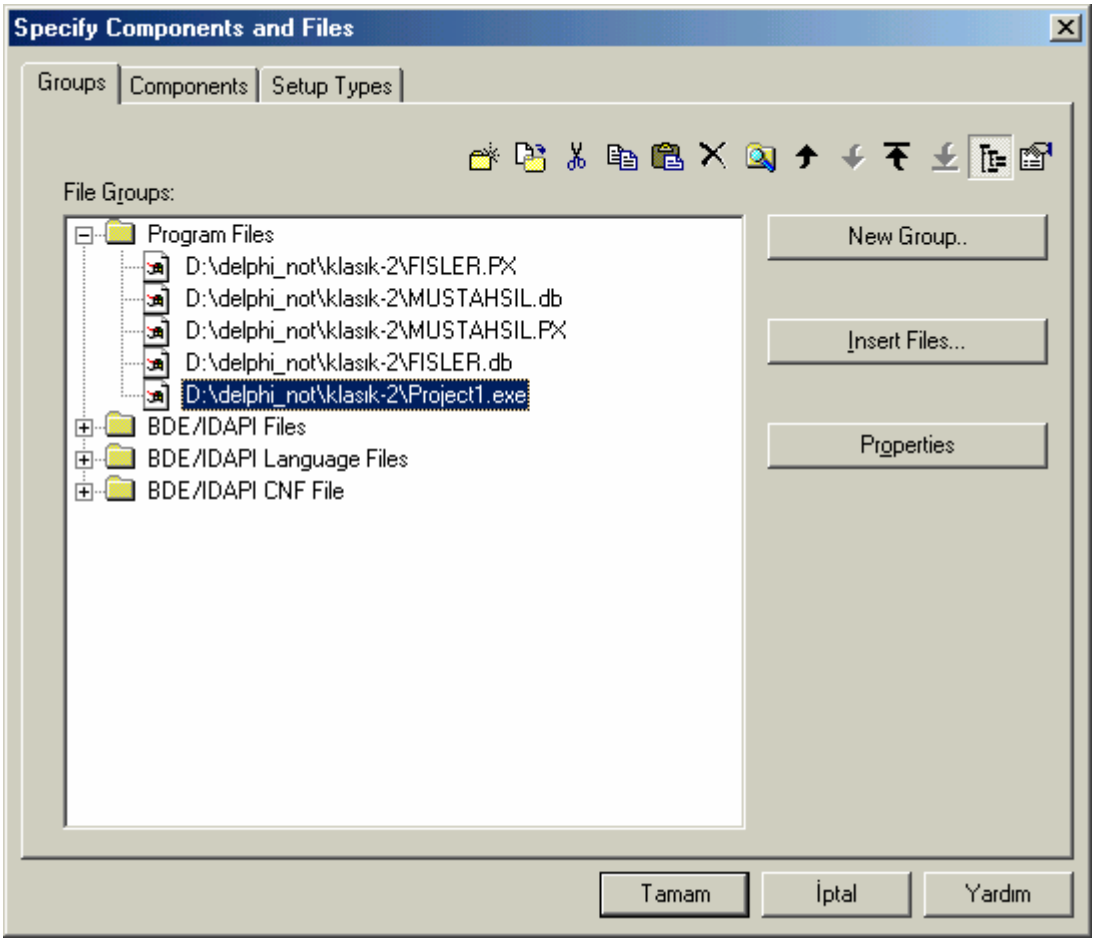

BDE Alias Step 3 of 4
You must define the database file path and select the type of driver for each of the aliases you have created.
Alias Name:
mus
Path:
c:\yepyeni
Type:
DB2
List any optional parameters for the alias below.
< Geri
İleri >
İptal
Yardım

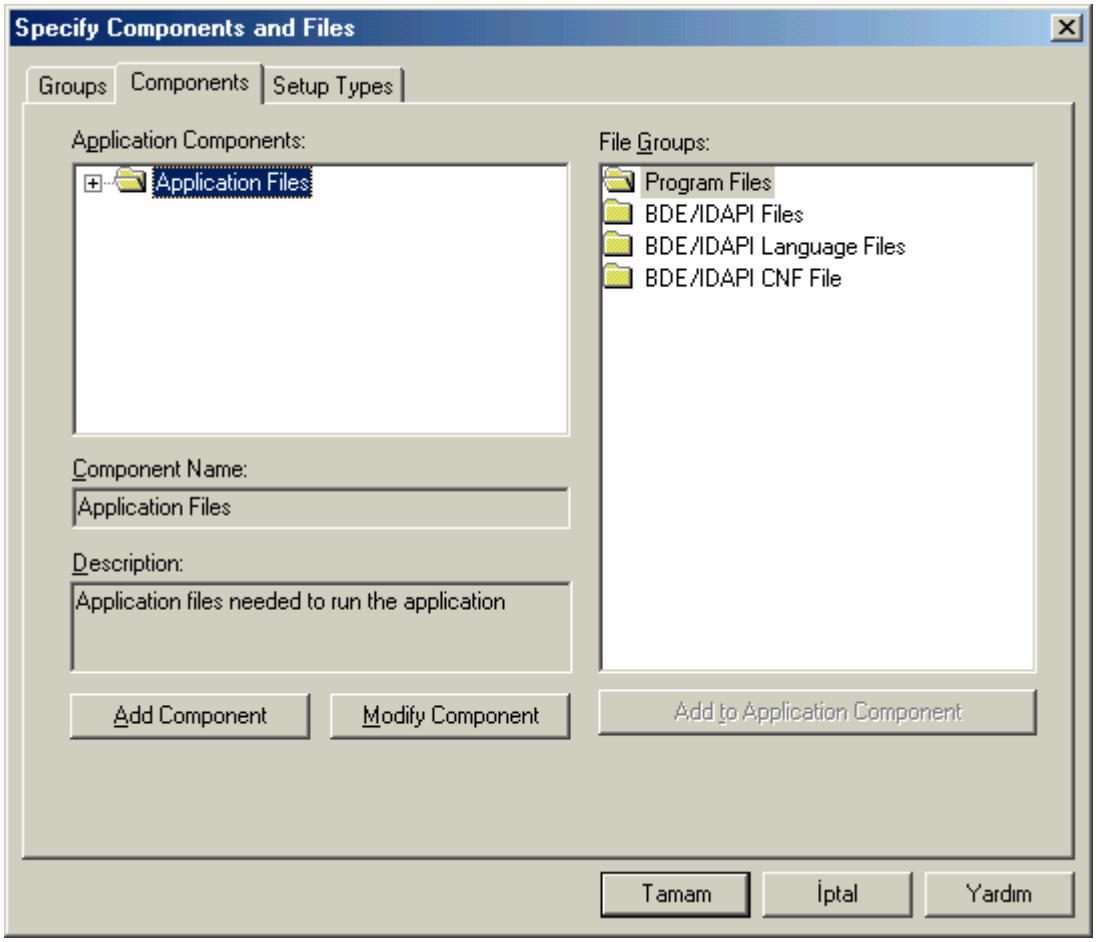

BDE Alias Step 4 of 4
BDE Alias settings completed.
Click Finish to initialize these settings in your setup.
These settings will then be installed on the target system.
< Geri
Son
İptal
Yardım

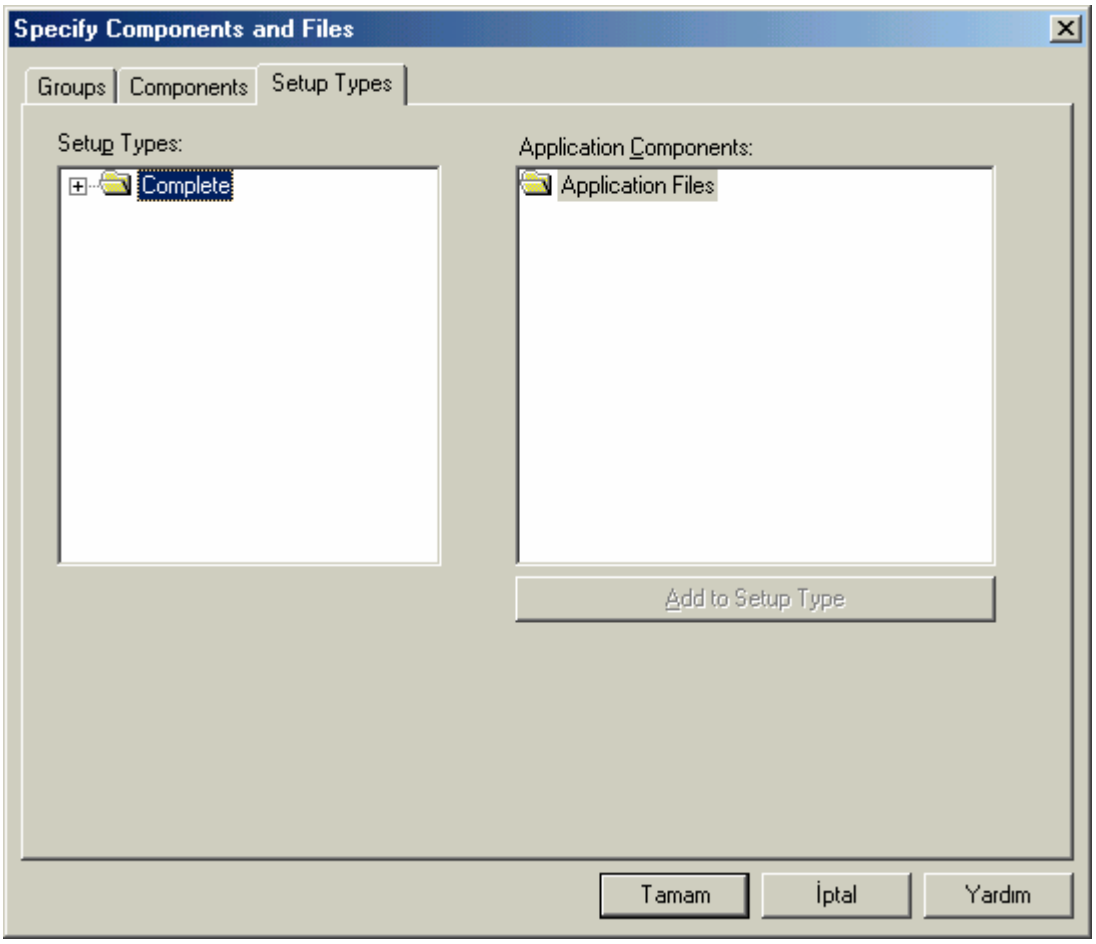


6. Specify Components And Files Seçeneği;

Bu seçenek **Groups, Component, Setup Type** adlı 3 sekmeden oluşur. Burada öncelikle araç çubuğundaki **Launch Explorer** düğmesinden yararlanarak proje dahilinde kullanılacak veri tabanı dosyaları ve **EXE** dosya seçilip projeye dahil edilir**.**

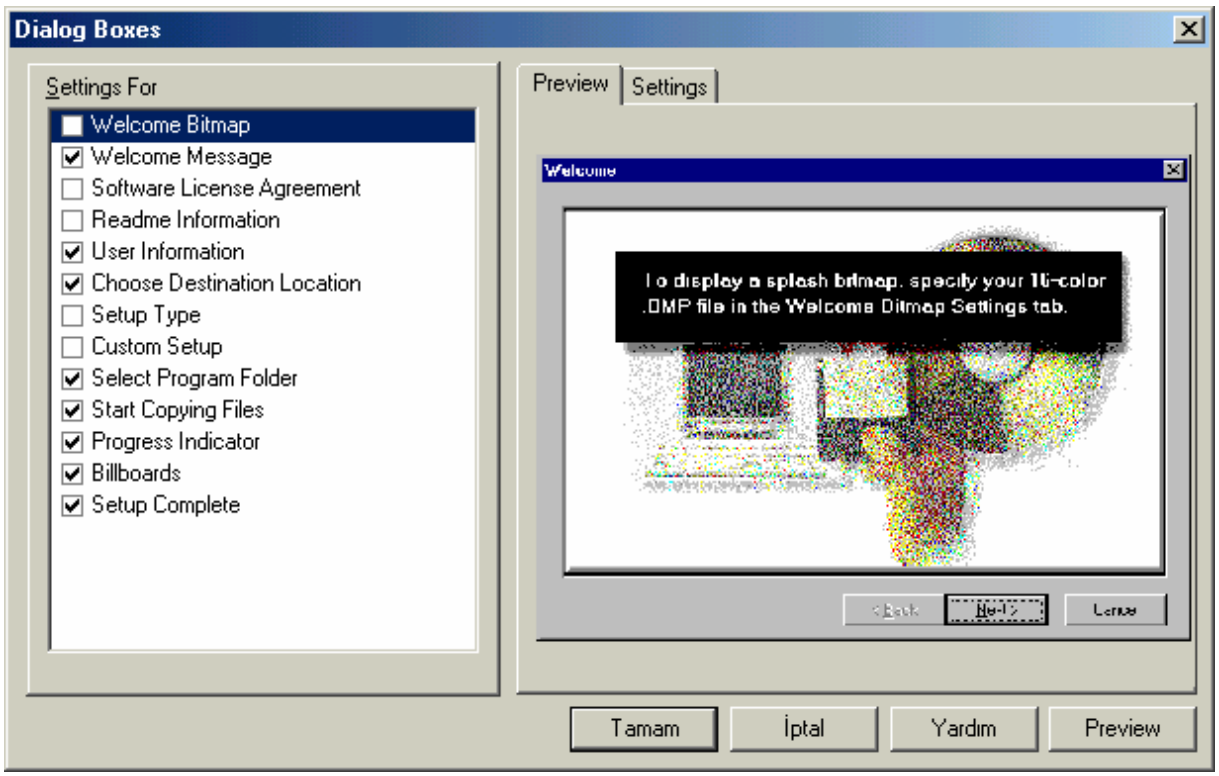

Specify Components and Files
Groups
Components
Setup Types
File Groups:
Program Files
D:\delphi_not\klasık-2\FISLER.PX
D:\delphi_not\klasık-2\MUSTAHSIL.db
D:\delphi_not\klasık-2\MUSTAHSIL.PX
D:\delphi_not\klasık-2\FISLER.db
D:\delphi_not\klasık-2\Project1.exe
BDE/IDAPI Files
BDE/IDAPI Language Files
BDE/IDAPI CNF File
New Group..
Insert Files...
Properties
Tamam
İptal
Yardım


Specify Components and Files
Groups
Components
Setup Types
Application Components:
Application Files
File Groups:
Program Files
BDE/IDAPI Files
BDE/IDAPI Language Files
BDE/IDAPI CNF File
Component Name:
Application Files
Description:
Application files needed to run the application
Add Component
Modify Component
Add to Application Component
Tamam
İptal
Yardım


Specify Components and Files
Groups
Components
Setup Types
Setup Types:
Complete
Application Components:
Application Files
Add to Setup Type
Tamam
İptal
Yardım

7. Select User Interface Components Seçeneği;

Bu seçenek grubundaki seçenekler ile proje bilgisayara kurulurken kullanıcıya yöneltilecek sorular ve varsayılacak değerler bulunur.Buradakı her seçenek bir onay kutusu ile temsil olunur.

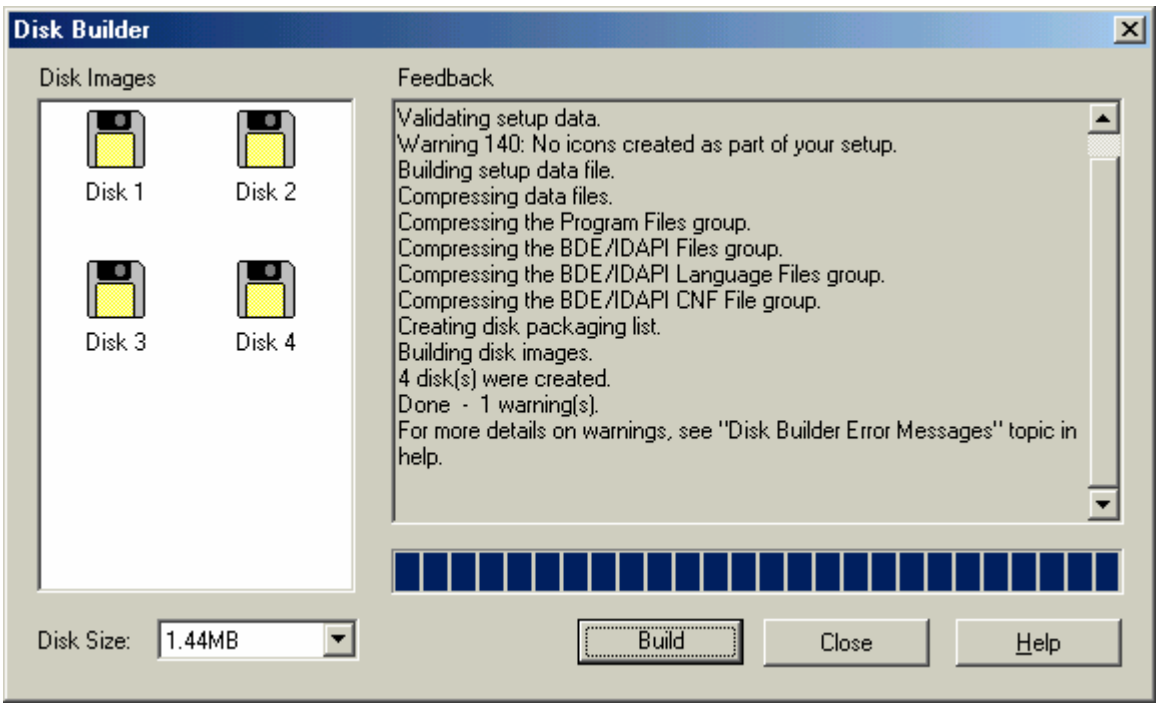


8. Make Registry Changes

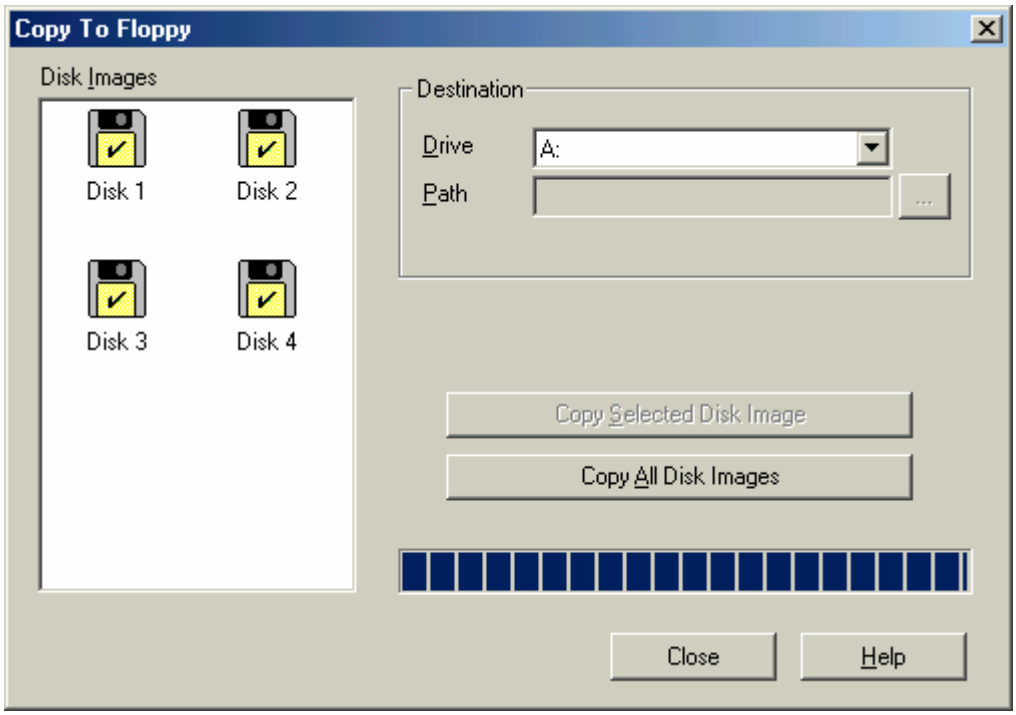
9.Specify Folders and Icons

10.Run Disk Builder

Disk Builder ekranı varken **Build** seçeneği tıklanırsa **Installshield** programı yapılan seçim ve ayarlamalara göre gereken dosyaları derleyip disketlere dağıtılacak şekilde hard diske kayıt eder. Bu ekran kapatılıp **Copy Floppy** seçeneği seçilirse **Copy All Disk Image** seçeneği ile diskte saklanan dosyalar disketlere kayıt olur.



11.Test The Installation
12. Create Distribution Media



WIN32 BİLEŞENLERİ

TpageControl Bileşeni:

Ekranda birbiri ile ilgili bir çok bileşen olduğu zaman ve bu bileşenlerin kendi aralarında gruplanarak kullanıcıya sunulabileceği durumlarda kullanılır.

Hazırlanması:

- Form üzerine **TpageControl** bileşeni yerleştirilir. Pagecontrol1 adını alır.
- Sağ tıklanarak **New Page** seçeneği ile istenilen sayıda **tabsheet1 tabsheet2 v.b.** şekilde sayfalar sıralanır.
- **Object Inspector** kullanılarak özelliklerinde degişiklikler yapılır.

TtreeView Bileşeni:

Bilginin hiyerarşik görüntülenmesi için kullanılır.

Hazırlanması:

- Form üzerine **TtreeView** bileşeni **Win32** sekmesinden aktarılır.
- **Items** özelliği açılır. Editör gelir.
- **New items** seçeneği ile ana seçenekler tanımlanır.
- **New Subitem** seçeneği ile alt seçenekler tanımlanır.
- Oluşturma esnasında text alanına isimler yazılır.
- Çalıştırılır.

Programın çalışması esnasında secenek girilmesi veya silinmeside gerekebilir. Bu durumda ilave edilecek butonlara aşağıdaki kodlar yazılır.

Yeni seçenek eklemek:

```
Begin
  Liste.selected:=liste.items.add(liste.items[0],'yeni kategori');
End;
```

Yeni alt seçenek eklemek:

```
Begin
  Liste.selected=nil then exit;
  if liste.selected.level=0 then
      Liste.items.addchild(liste.selected,'yeni alt secenek');
End;
```

Seçenek iptal etmek:

```
Begin
  Liste.selected=nil then exit;
  if liste.selected.level=0 then
      Liste.selected.deletedchildren;
End;
```

TtreeView bileşeninde sık kullanılan iki olay vardır. Bunlar **Oncollapsing** ve **onExpanding** olaylarıdır. **OnExpanding** olayı liste de + işaretine basarak alt elemanları görmek istediğimiz zaman oluşur. **Oncollapsing** kullanıcı – işaretine basıp alt elemanları toplamak istediği zaman oluşan olaydır.

TmediaPlayer Bileşeni:

Açıklanan bileşenler dışında aşağıda belirtilen **Win32** Bileşenleride mevcuttur.

TheaderControl Bileşeni:

TlistView Bileşeni:

TstatusBar Bileşeni:TtabSet Bileşeni:

TnoteBook Bileşeni:

Ttimer Bileşeni:

TfileListBox Bileşeni:

TdirectoryListBox:

TdriveComboBox Bileşeni:

TfilterComboBox Bileşeni:

ADDITIONAL BİLEŞENLERİ

Genel olarak bu bölümün bileşenleri ile butonlara resim yerleştirme, değişkenleri formatlama, çizim yapma, görüntü kopyalama, resim yerleştirme, yazıları formatlama, kısayol tuşları tanımlama, AVI dosyaları kullanma, grafik çizme gibi işlemler yürütülür.

TbitBtn Bileşeni: Standart sekmesindeki buton nesnesi ile aynı göreve sahiptir. Ancak bu butona resim **(bitmap)** yerleştirilip kullanılabilir.

Bitmap görüntünün atanma işlemi **Glyph** özelliğinden gerçekleştirilir. Bu özellik seçildikten sonra ekrana gelen pictur editöründen load seçimi ile görüntü yüklemesi sağlanır.

Bitmapnesneleri;
C:\ProgramFiles\CommonFiles\BorlandShared\Images\Buttons\Arrowlr.bmp
dosyasındadır.

Bu işlem kod yazarakta programın çalışması anında da gerçekleştirilebilir.

```
BitBTn.Glyph.LoadFromFile( 'c:\ ProgramFiles\CommonFiles\'
+ 'BorlandShared\Images\Buttons\Arrowlr.bmp');
```

Ayrıca **Kind** özelliğinden buttona yerleştirilen resim değiştirilebilir.**Caption** özelliği kullanılarakta buton üzerindeki ifade değiştirilebilir.

Kind özelliğinin değişimi ile birlikte değişen **ModaltResult** özelliğidir. Formlar ekranda iki metot ile görüntülenir. Bunlar **Show** ve **Showmoda**l metotlarıdır. **Form Show** ile görüntülenmiş ve ekranda birkaç tane ise birbiri arasında gezinebiliriz. Ama **Showmoda**l ile görüntülenmiş ise ilgili form kapatılmadan diger form görüntülenmez. Yani **ModaltResult**' a tanacak değerler vardır. **TbitBtn** nesnesinde **Kind** özelliğine bir değer atanırsa (bkclose) **ModaltResult** özelliğide ona paralel değer **(mrClose**) alır. Bu program içinden de kontrol edilebilir. Ancak **Modalt Result** özelliğine **mrNone** değeri atanırsa program calışırken butona basıldığında herhangi bir değer geriye döndürülmeyecektir.

Buton üzerinde görüntünün yerini belirlemek içinde **LayOut** özelliği kullanılır.

TspeedButton Bileşeni: Bu bileşende buton hazırlamak için kullanılır. **Tbutton Windows** standartıdır. Resim görüntüleyebilmek için **TbitBtn** geliştirilmiştir. **TspeedButton** ise aynı amaçla olmakla birlikte sistem kaynaklarını verimli kullanmak için düzenlenmiştir.

Windows sistemi içinde oluşturulan her bileşenin **Handle** adında bir özelliği vardır. Bu özellik, Windows isletim sisteminin o bileşene vermiş olduğu bir numara gibi düşünülebilir. Windows işletim sisteminin **API** (Application Programming Interface) adı verilen bir fonksiyon seti vardır. Windows API fonksiyonlarının bu bileşenleri kullanabilmesi için bu fonksiyonlar parametre olarak , kullanılmasını istediğimiz bileşenin handle özelliğini alırlar. İşte **Speedbutton** için böyle bir handle özelliği yoktur. Böylece daha az sistem kaynağı kullanılmış olur.

Ayrıca SpeedButton'un hoş ,estetik kullanıcı arabirimi oluşturma özellikleride vardır.
Flat özelliği **True** yapılırsa arka fon ile olan keskin hat farkı ortadan kalkar.

TmaskEdit Bileşeni: Bu bileşenin **EditMask** özelliği kullanılarak ülkelere göre önceden hazırlanmış formatlar kullanılarak bilgilerin belli formatta girilmesi sağlanabilir.Ancak bu bileşen Delphi2' nin en hantal ve gelişmemiş bileşenidir. **Edit Mask** için kullanılan bazı karakterlerde bilgilerin girişini formatlar.

TimageBileşeni: Form üzerinde herhangi bir yerde grafiksel bir şeklin görüntülenmesini sağlar. Bu bileşen ile **.BMP** veya **.ICO** dosyası form üzerinde görüntülenebilir.

Tcanvas Nesnesi: **Timage** bileşenlerinin ekranda resim görüntülenmesini sağlayan **Canvas** adında özelliği vardır. Canvasın bir çok metodu sayesinde herhangi bir çizim yaptırılabilir. Canvas nesnesinin kendisi indirekt bir özelliktir. Hemen hemen her bileşenin canvas adında özelliği vardır. Boyama çizme işlemleri bu özellik sayesinde olur. **Brush** canvasın yüzey özelliklerini tasır. Bunlardan biride **Style** özelliğidir. Bu özellik ile çizimlerin içi değişik görünümlerde doldurulabilir.

```
Canvas.brush.style:=bsvertical;
```

Tshape Bileşeni: Bu bileşen sayesinde form üzerindeki belli alana geometrik şekiller çizilebilir. Bunun için **Shape** özelliğini değiştirmek yeterlidir. **Tshape** nesnesinin **Pen** ve **Brush** adında iki özelliği vardır. Pen ile kalemin boyutları değiştirilebilir. Brush özelliği ile de şekillerin fonuna ait özellikler değiştirilebilir.

TstringGrid Bileşeni: Kullanışlı bir bileşendir. Tablo şeklindedir. Yani iki boyutludur. Özellikleri değiştirilebilir. (Satır - sütun sayısı, çalışma anında bilgi girişi, hücre büyüklükleri gibi.)

TdrawGrid Bileşeni: **TStringgrid** şeklinde çalışır.

Tbevel Bileşeni: **Tpanel** bileşeni gibi çalışır. Daha az sistem kaynağı kullanır.

TstaticText Bileşeni: **Tlabel** bileşeni ile aynıdır. Yalnız Tlabel bileşenlerinin **Handle** özelliği yoktur. **TstaticTex**t Bileşeninin **Handle** özelliği vardır.

TcheckListBox Bileşeni: **TlistBox** bileşenine benzer. Ancak aynı anda **Checkbox** bileşeni gibide çalışır.

Tsplitter Bileşeni: Bu bileşen Form üzerine yerleştirilen farklı iki bileşenin birinin kapladığı alan artarken diğerinin azalmasını sağlayan bileşendir. Bunun için sürükle bırak özelliğinden yararlanılır. Kullanımı için en iyi örnek windows gezginidir.

Örnek:

- Form üzerine 2 adet **TlistBox** yerleştirelim
- Birtanesinin **Align** özelliğini **alleft** diğerinin **alclient** yapalım
- **Tsplitter**bileşeni alıp soldaki **listbox** üzerine bırakalım.
- **Tsplitter Minsize** özelliği ile büyüklüğü değişen bileşenin min genişliği için rakam seçilebilir.
- **Tsplitter**'in bir olayı vardır. **OnMoved** olayı Bu olay form üzerinde bileşenlerin büyüklüğü değiştiği anda oluşur. Burayada bir mesaj yazalım.

TrichEdit Bileşeni: **TrishEdit** bileşenleri windows 95 ile gelen bileşenlerdir. **Tmemo** bileşenine benzer ancak yazılan bilgilerin bazılarının görünümleri değiştirilebilir.Yazılar formatı ile saklanıp tekrar çağrılabilir.

```
RichEdit1.Lines.SavetoFile('temp.rtf');
RichEdit1.Lines.LoadFromFile('temp.rtf');
```

Thotkey Bileşeni: Windows'ta kısayol tuş bileşenleri oluşturmak ve bu bileşenlerin kullanıcılar tarafından değiştirilebilmesini sağlamak amacı ile kullanılır.

TcoolBar Bileşeni: Bu bileşen sayesinde kullanıcılar fonksiyonlarla ilgili düğmelerin bulunduğu araç çubuğunun yerini sürükle ve bırak metodunu kullanarak rahatlıkla değiştirebilirler.

ÖRNEK:

- Form üzerine **Tcoolbar** bileşeni yerleştirelim. **(win32)**
- Bunun üzerine **Tpanel** bileşeni alıp **Tcoolbar** üzerine yerleştirelim. Yüksekliğini ayarlayalım.
- İkinci bir **Tpanel** daha **Tcoolbar** üzerine yerleştirelim ve yüksekliğni ayarlayalım.
- **Tpanel** bileşenleri üzerine buton, edit alanı, label gibi nesneler yerleştirelim.
- Çalıştıralım

Tcoolbar aslında **Tcoolband** adı verilen bantlardan oluşur. Bu bantlar üzerine yerleştirilen buton veya nesneler ile kullanıcılara program ile daha etkileşimli bir ortam sağlar. Sürükle bırak yöntemi ile bantlar yer değişir.

Tanimate Bileşeni: Bu bileşen sayesinde ***.AVI** uzanytılı görüntü dosyalarını form üzerinde oynatılabilir. Bunun için yapılması gereken, **Tanimate** bileşenlerinin **FileName** adındaki özelliğine bir ***.AVI** dosyası adını eşitlemek olacaktır. Deneme olarak **Cool.AVI** dosyası kullanılabilir. Bu dosya ,

C:\ProgramFiles\ Borland\delphi5\demos\coolstuf dizinindedir.

Atama işleminden sonra yapılacak tek şey **Active** özelliğini **True** yapmaktır. Bu işlem bir butona bağlanabilir. Oynaması veya durmasını buton ile kontrol edebiliriz.

```
Begin
    Animate1.Active:=not animate1.active;
End;
```

Object Inspector'dan ulaşabileceğimiz CommonAVI özelliğinde hazır windows avi'leri vardır. Bunları seçer,**Active** özelliklerinide **True** yaparsak çalışmalarını görebiliriz.

AVI görüntüleri aslında peşpeşe resim geçirmedir.Bu resimlerin sayısı ise **StopFrame** adlı özellikte bulunur.

Görüntünün kaç kez oynamasını istiyorsak **Repeatitions** özelliği kullanılır. Eğer bu sayı 0 veya –1 ise görüntü durmadan oynar.

Tanimate bileşeni sıkıştıtırmış AVI dosyalarını okumaz. AVI dosyası içinde ses kayıdı varsa okumaz bunlar için **TmediaPlayer** bileşenini kullanmak gerekir.

Tchart Bileşeni: Çok kullanışlı bileşenlerden biridir. Güçlü grafikler üretilebilir. Bu bileşen **additional** sekmesinde,**quickreport** sekmesinde,**data control** sekmesinde bulunmaktadır.

TapplicationEvents Bileşeni: Bu bileşen sayesinde tüm windows mesajları yakalanabilir gerekirse tek merkezde kontrol edilebilir.Kullanışlı bir bileşendir.

DİYALOG (DIALOGS) BİLEŞENLERİ

Bu sekme; Topendialog, Tsavedialog, Topenpicturedialog, Tsavepicturedialog, Tfinddialog, Treplacedialog, Tfontdialog, Tcolordialog, Tprintdialog, TopenPrintSetupdialog düğmelerinden oluşur.

Diyalog kutuları standart ara birimler oluşturmak için kullanılan kutulardır. Bunların tamamı **Execute** metodu ile aktif hale gelir. Bunların kullanımı uluslararası kullanımıda kolaylaştırır. Diyalog kutuları türkçe windows kullanılıyor ise türkçe olarak gelir.

Diyalog kutuları belirlenen işi yapmak için ara birim sağlar, o işin nasıl yapılacağını programcı kod yazarak sağlar.

TopenDialog (dosya aç penceresi):

Herhangi bir dosyayı kullanıma sunar. **Execute** metodu ile aktifleşir. Diyalog penceresinden seçilen dosya opendialog nesnesinin **FileName** özelliği ile öğrenilir.

Özellikleri:

FileName: Dosya adı bulundurur.

Files: Options özelliğinin bir altı olan **Ofallowmultiselect True** ise birden fazla dosya seçilebilir.

Filter: Opendialog penceresinde gösterilecek olan dosya türlerini belirtir. Birden fazla ise "|" karakteri arada gösterilir.

FilterİndexI : Filter'da belirtilen dosyaların hangisinin ilk gözükecegini belirler. Varsayılan 1'dir.

DefaultExt: Dosyaya uzantı verilmez ise buradaki geçerlidir.

FileEditStyle: Open dialog kutusu açıldığı zaman dosya adının girileceği kısmın biçimini bulundurur.

HistoryList:

İnitialDir: Pencere açıldığında gösterilecek dizini belirler.

Options: Opendialog ve save dialog'da etkilidir.

Ofallowmultiselect: Birden fazla dosya seçimine imkan verir.

OfCreatePrompt: Belirtilen dosya diskte yoksa 'dosya yok yaratmak istiyormusun' diye sorar.

OfExtensionDifferent: Girilen dosyanın uzantısının Defaultext'ten farklı olup olmadığı öğrenilir.

OgFileMustExist: Dosya diskte yoksa olmadığına dair mesaj verir.

OfHideReadOnly:

OfReadOnly:

OfNochanceDir: Diyalog kutusu ilk açıldığında içinde bulunulan dizin değişse bile eski halini getirir.

OfNoVAlidate: True ise dosya adına geçersiz karakter kabul etmez.

OfOverWritePrompt: Olan dosya üzerine kayıt yapılsın mı mesajını verir.

OfPathMustExist: Olmayan dizin girildiğinde uyarır.

OfShowHelp:

Title: OpenDialog başlığı için kullanılır.

TsaveDialog: Kaydetme penceresini sunar.

TopenPictureDialog: Kullanılacak resim yüklenmeden önce ön izleme yaptır. (BMP, ICO, EMF, WFM formatları açılmadan izlenir.)

TsavePictureDialog:

TfontDialog: Win standart font penceresini sunar.

TcolorDialog: Renk seçme penceresini sunar.

TprintDialog: Yazdır penceresini görüntüler.

TprinterSetupDialog: Yazıcı ayarları penceresini sunar.

TfindDialog: Metinden herhangibir kelimeyi bulmak için kullanılır.

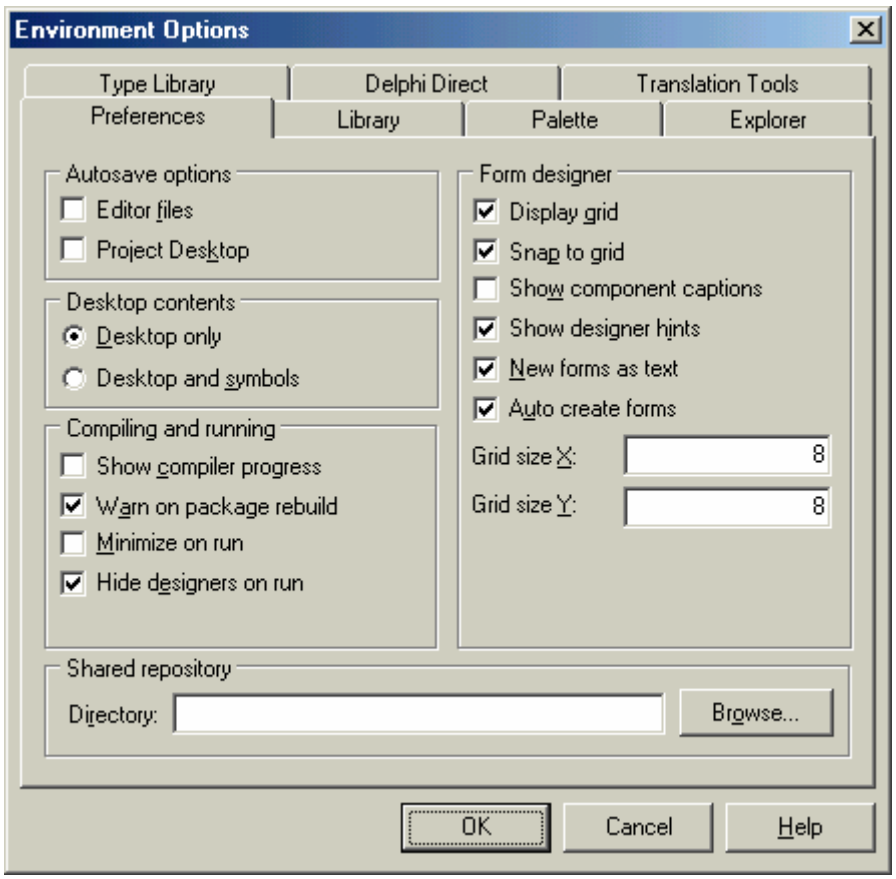
TreplaceDialog: Metinden herhangibir kelimeyi değiştirmek için kullanılır.

DELPHI' DE AYARLAMALAR VE DÜZENLEMELER

Delphi kullanımı esnasında ; Delphi'nin kendi düzenlemelerinde , geliştirilen programda kullanılacak komutlarla ilgili veya program yazılan ortamla ilgili düzenlemeler yapmak gerekebilir. Program geliştirmek ile ilgili düzenlemeler bir bölümü **Tools** menüsündeki **Environment Options** seçeneğinden, program yazım ortamı ile ilgili ayarlar **Tools** menüsünün **Editör Options** seçeneğinden, Yürütülen proje ile ilgili düzenlemeler ise **Project** menüsündeki **Options** seçeneğinden yapılır.

GENEL AYARLAMALAR (ENVIRINMENT OPTIONS)

Program geliştirme ortamı ile ilgili ayarların yapıldığı bu ortam **Tools** menüsünün **Environment Options** seçeneği ile yapılır. Bu seçenekte 7 adet sekme bulunmaktadır. Burada cok kullanılanlara kısaca değinmek istersek.



PREFERENCE SEKMESİ:

Editor Files: Proje çalıştırıldığı anda yazılan kodların ve hazırlanan formların otomatik olarak saklanması istenirse bu onay kutusunun seçilmesi gerekir.

Project Desktop: Proje çalıştırıldığında projenin masaüstü durumununda kaydedilmesi isteniyorsa Onay kutusunun seçilmesi gerekir.

Desktop Only: Bu düğme seçili iken önce yapılan tüm masaüstü ayarlarını hatırlar.

Desktop and Symbol: Proje kayıt edilirken veya kapatılırken masaüstü konumları yanında en son derleme işlemi ile ilgi bilgilerinde kayıt olması isteniyorsa bu düğme seçili olmalıdır.

Display Grid: Program yazımı esnasında formlar hazırlanırken form üzerinde kolay çalışabilmek için klavuz çizgileri (noktalar) bulunmaktadır. Bunlar kaldırılması isteniyorsa onay kutusundaki onayın kaldırılması gerekir.

Snap to Grid: Nesneler forma yerleştirilirken klavuz çizgileri esas alınır. Nesneler yerleşirken ve boyutları belirlenirken klavuz çizgileri gözönüne alınmasın isteniyorsa bu onay kutusunun onayı iptal edilmelidir.

Show Component Captions: Forma yerleştirilen nesnelerin altında adlarıda görünsün isteniyorsa bu onay kutusunun onaylanmış olması gerekir.

Show Designer Hints:Forma yerleştirilen nesnelerin üzerinde mouse biraz tutulduğunda adları hatırlatılsın isteniyorsa bu onay kutusunun seçili olması gerekir.

Auto Create Forms: Hazırlanan projelerde formlara otomatik olarak isim verilmesi istenmiyorsa bu onay kutusunun iptal edilmesi gerekir

PALETTE SEKMESİ:

Bu sekme kullanılarak yeni bileşen grubu hazırlanabilir, düğmelerin veya sekmelerin yerleri değiştirilebilir.

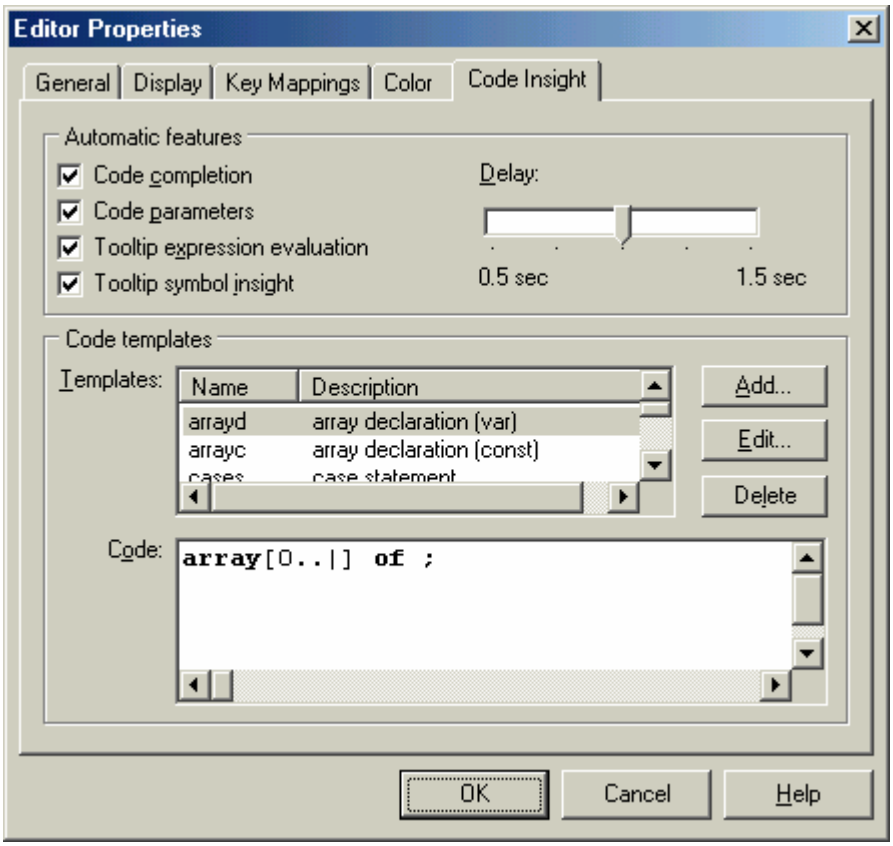
EXPLORER SEKMESİ:

Code Explorer penceresi üzerinde ayarlamalar yapmak için kullanılır**.**

YAZIM ORTAMI AYARLARI (EDITOR OPTIONS)

Bu seçenekte 5 adet sekme bulunmaktadır. Burada cok kullanılanlara kısaca değinmek istersek.

Geliştirilen her programda bir takım eksikliklerin ve hataların olması, bir noktaya kadar kabul edilebilir.Ancak çıkabilecek hataların önceden tahmin edilerek , olumsuz durumların uyarıcı mesajlar ile kullanıcılara bildirilmesi gerekir.



GENERAL SEKMESİ:

Programla ilgili genel ayarların yapıldığı sekmedir. Örneğin **Auto Indent Mode** seçeneği seçili ise program yazım esnasında hizalama düzgün olacaktır.

DISPLAY SEKMESİ:

Program ile ilgili fontlar, büyüklükler ayarlanabilir. Bir diğer kullanımı ise kod yazılan ekranın sonunda **Gutter Bar** adı verilen bar bulunur. Bunun görünüp görünmemesi için kullanılır.

COLOR SEKMESİ:

Renk ayarlamaları için kullanılır.

CODE INSIDHT SEKMESİ:

Code Insight ile birlikte program yazma işi programcılar için çok daha hızlı bir hale gelmektedir. Bununla ilgili parametreleri ve seçenekleri görebilmek için;

- Ana menüden **Tools/Editor Options** seçenekleri girilir,
- **Code Insight** sekmesi seçilir
- **Code Insight** penceresi görüntülenir.

Gelen ekranda dört bölüm vardır.

- **Code Completion** (kod tamamlayıcı)
- **Code Parameters** (kod parametreleri)
- **Tooltip**
- **Code Templates** (kod şablonları)

Kod tamamlayıcı: Bu onay kutusu işaretli olduğu sürece kod yazarken '**.**' işaretine basıldığı anda yazılabilecek metotlar listelenir. Tüm hepsini akılda tutmak mümkün olmadığından hatırlatıcı olarak oldukça kullanışlıdır.

Kod parametreleri: Bu onay kutusu işaretli olduğu sürece kod yazarken kullanılacak olan prosedür,fonksiyon, metot her ne ise bunların parametrelerini hatırlatır. İlk parantez açıldıktan sonra biraz beklenirse o fonksiyon yada metodun parametreleri listelenir.

ToolTip: Hata ayıklama sırasında **Watch** penceresini kullanmaya gerekte kalmayabilir.Program çalışırken her adımda değişkenlerin aldığı değerleri görüntülenmek istenebilir.Eski versiyonlarda bu menüden yürütülmekteydi.Burada onay kutusu işaretli ise mouse değişkenin üzerinde tutulursa değişkenin aldığı değer altında görüntülenir.

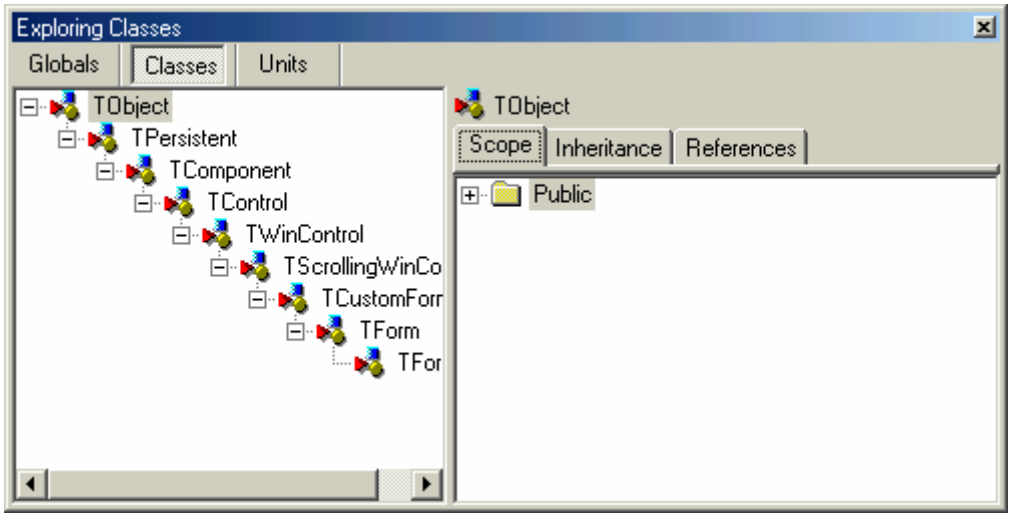
Kod Şablonları: Bu onay kutusu işaretli ise önceden kaydedilen kod şablonları program içinde kullanmak mümkündür. **Kod Insight** penceresindeki **Add** düğmesi kullanılarak ilgili şablon hazırlanır kayıt olur.Kod yazarken verilen isimle kullanılabilir.

Tooltip Symbol Insight, Modül izleyici: Bu özellik sayesinde kod içindeki herhangi bir sınıf,nesne,değişken hakkında kısa ipucu verir.İlgili tanımın üzerinde (örn. ***Tobject***) mause tutulursa ipucu görüntülenir. (örn. ***Type system.Tobject – System.pas (128)***)

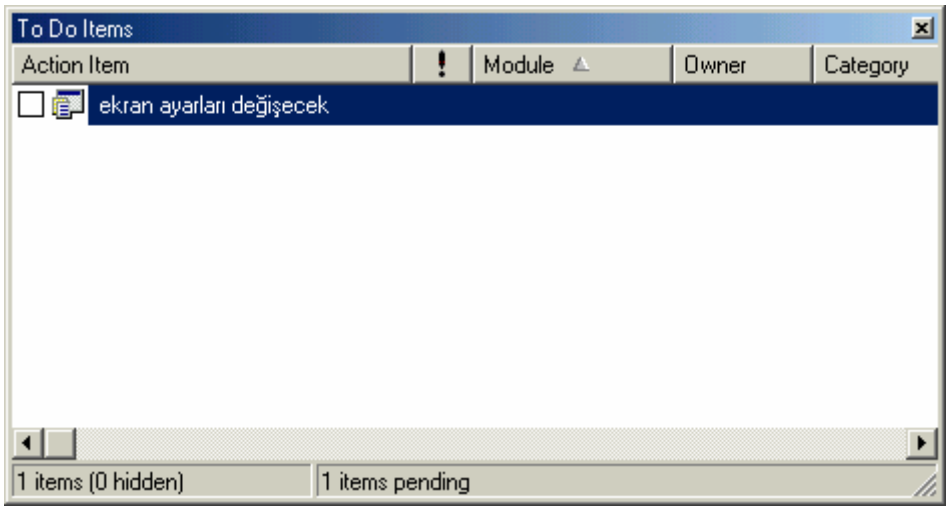
BookMark'lar: Kod listesi içinde bazı satırlara bookmark'lar koymak mümkündür.Bu işaretler sayesinde kod listesi içinde hızlı gezinilir.Ancak bir modül içinde en fazla 10 bookmark konabilir.Bookmark koymak için ilgili satırda **CTRL+SHIFT+sayı** (sayı 0-9 arasında olmalıdır) tuş kombinasyonu kullanılır.Satır solunda işaret belirir. Aynı satırda aynı tuş kombinasyonu tekrar kullanılırsa işlem iptal edilir. Bookmark'lar arasında dolaşmak için **CTRL+sayı** (sayı tanımlanan 0-9 arası değerdir) tuş kombinasyonu kullanılır.

Kod Gezgini: Kod gezgini delphi ile birlikte gelen kullanışlı modüllerinden biridir. Bir modül içindeki sınıfları,fonksiyonları,değişkenleri ve uses bölümündeki modül listesini göstermektedir.Dockable bir pencere olduğu için kod edidörünün dışınada çıkartılabilir.ana **menü/view/code explorer** seçeneğinden de görüntülenebilir.

Proje izleyicisi: Delphi 5 ile gelen bir yeniliktir. Çalışılan projeye ait modülleri,sınıfları,nesneleri izlenebilir. Açmak için ana menüden **View/Browser** seçilir.

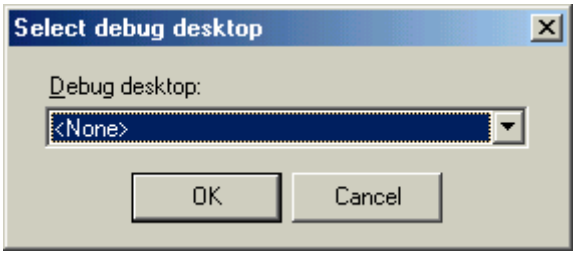


Yapılacaklar listesi **(To Do Lists)** program ile yapılması istenilen işler kullanılır. Hatırlatmalar yazılır. Kullanımı için ana menüden **View/To Do Lists** seçilir. Ekrana gelen form üzerinde sağ tıklanarak gelen ekrana yapılacaklar yazılabilir.



Yapılacakları yazmak için **To Do Items** ekrana geldiğinde sağ tıklanarak gelen ekrandan **Add** seçilir ve gerekli mesajlar yazılır. Yazılan problem veya mesajlar katagorilere ayrılabilir önemine göre sayı ile dercelendirilebilir.En soldaki onay kutusu tıklandığında da işlemin yerine getirildiği anlaşılır. Programcı veya kullanıcılar için not defteri gibi düşünülebilir.

Ayarlanabilir Masaüstü Özellikleri (Customizable Desktop Setting): Delphi 5'de gelen yeni özelliklerden biridir. Çalışılan proje ile ilgi ekran düzeninin ayrlarının saklanmasıdır. Bu işlem için select debug desktop düğmesi kullanılır. Bu düğme menüde en sağda bulunmaktadır. Bu seçilir ise aşağıdaki ekran görüntülenir.



Ayarlara isim verilir. Gerek duydukça kullanılır.

HATA AYIKLAMA

Profesyonel program geliştirmek için program yazma esnasında hataların denetlenmesi ve kullanıcıya bildirilmesi gerekir. Hataların kontrolü , denetlenmesi ve çözümü değişik yöntemlerle gerçekleştirilebilir.

İzlenmek istenilen satırlara duraklama noktası koymak

Bu işlem için;

- Duraklama konacak satıra gidilir,
- Ana menüden **Run/Add BreakPoint/SourceBreakPoint** seçenekleri seçilir,
- Ekrana **Add Source Breakpoint** formu gelir, seçenekler doldurulur,
- Bu durumda ok düğmesi kullanılırsa ilk duraklama konmuş olur.
- Bu durumda kod içindeki satır kırmızı bant şeklinde görüntülenecektir,
- Proğram çalıştırıldığında bir uyarı mesajı ile karşılaşılabilir. Bunun içinde
- Ana menüden **Project/Options** seçilmeli compiler sayfasındaki **Optimization** seçeneği boş bırakılmalıdır.
- Proğram çalıştırıldığında ilgili satıra geldiğinde kaynak kod görüntülenecektir.

Değişkenlerin değerini görmek

Bu işlem için;

- Ana menüden **Run/Add Watch** seçenekleri seçilir,
- Watch Properties ekranı gelir,
- **Expression** seçeneğine değişken adı yazılır,
- **Watch list'de** değişkenin aldığı değer(tabii atama varsa) görüntülenir.
- İşlem devam edilerek birden fazla değişken için görüntülenebilir.

(program çalışması esnasında işlemler yapılmalı ve **F7** ile satır satır devam edilmelidir.Bazen prosedürün satır satır değilde bir defada çalıştırılması istenebilir bunun içinde **F8** tuşu kullanılır.)

Duraklama noktasını kaldırmak

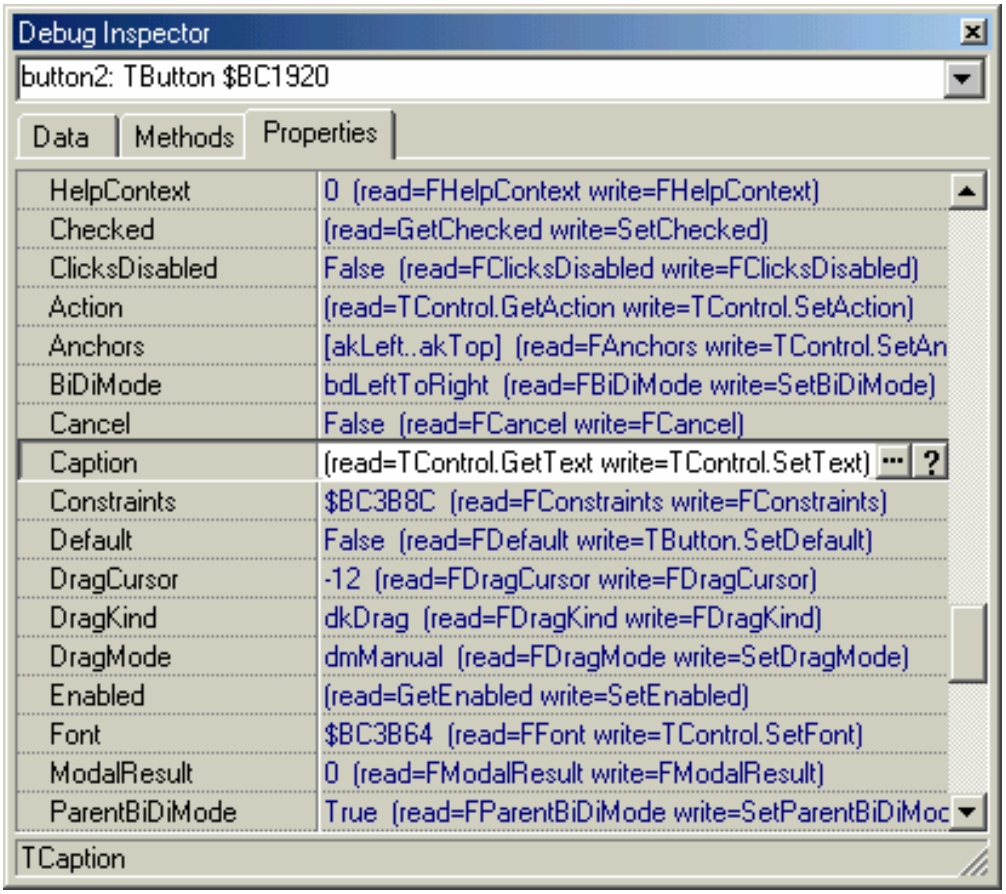
Bu işlem için

- Ana menüden **View /Debugwindows/Breakpoint** seçenekleri seçilir.
- Duraklama noktaları görüntülenir,
- Üzerinde bir kez tıklanır **Delete** tuşuna basılırsa duraklama kaldırılmış olur.

NOT: Duraklama noktası koymak ve kaldırmak için daha kolay yöntem vardır. İlgili satırın en solunda tıklanırsa duraklama konmuş olur. Tekrar tıklanırsa kaldırılmış olur.Yalnız menüden yapılırsa farklı özellikleri kullanmakta mümkündür. Örneğin menüden duraklama konduğunda şartta verilebilir. Eğer bu şart sağlanıyor ise duraklama devreye girer.

Debug Inspector: Delphi ile birlikte gelen **Debug Inspector** sayesinde, hata ayıklama sırasında nesnelere ait özellikleri, metotları ve olayları izlememiz ve gerekirse içlerindeki değerleri değiştirmemiz mümkün olmaktadır.

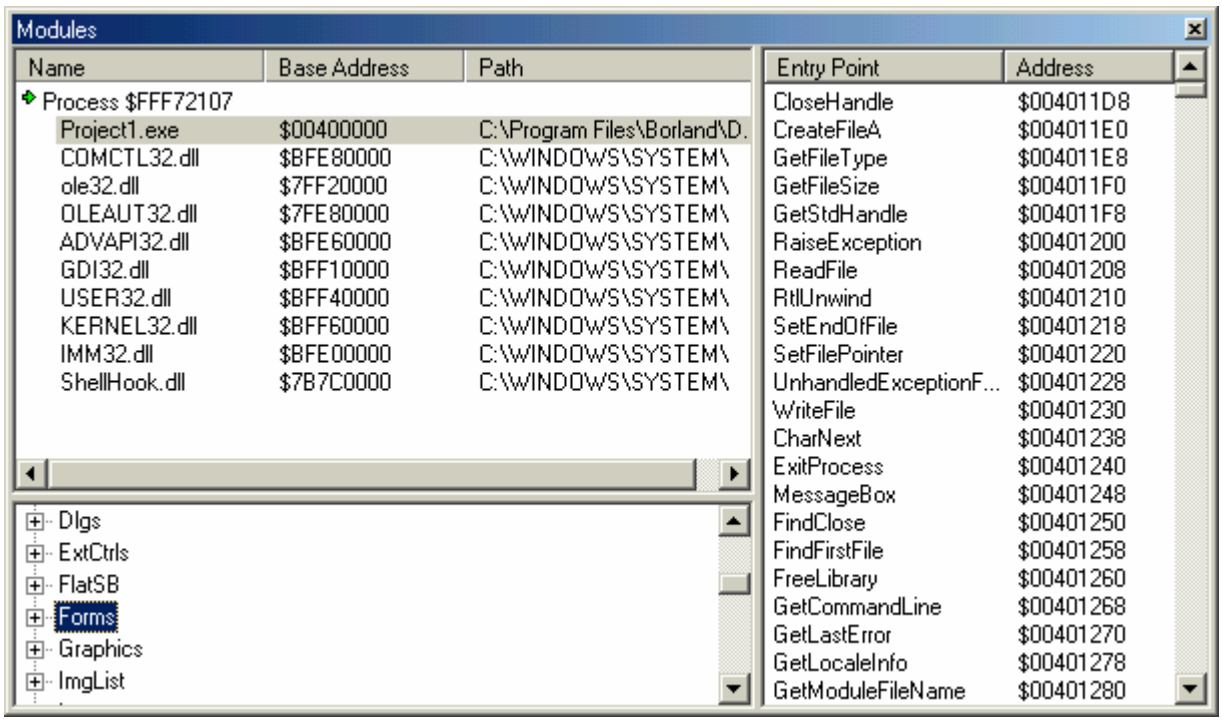
Debug Inspector için ilgili kod satırının üzerinde iken **Alt+F5** kullanılır.Ekrana **Debug Inspector** formu görüntülenir.Bu formda data,methods,properties sekmesivardır.Bunları kullanarak uygulamamızın çalışması esnasında nesnelerle ilgili tüm özellik ve metotları izlememiz mümkün olacaktır.Eğer istenirsede değişiklik yapmamız mümkün olacaktır.



Module Penceresi: Module penceresi uygulamamız çalıştığı anda EXE dosya ile birlikte hafızadaki DLL'leri, EXE içindeki ***.pas** uzantılı modülleri ve modüller içindeki alt yordamları, metotları izlememizi sağlayan penceredir.

Modülü izlemek için;

- ana menüden **View/Debugwindows/Modules** seçenekleri girilir,
- **Modules** formu ekrana gelir,
- Buradan ilgili modül ve yordamlar izlenebilir.

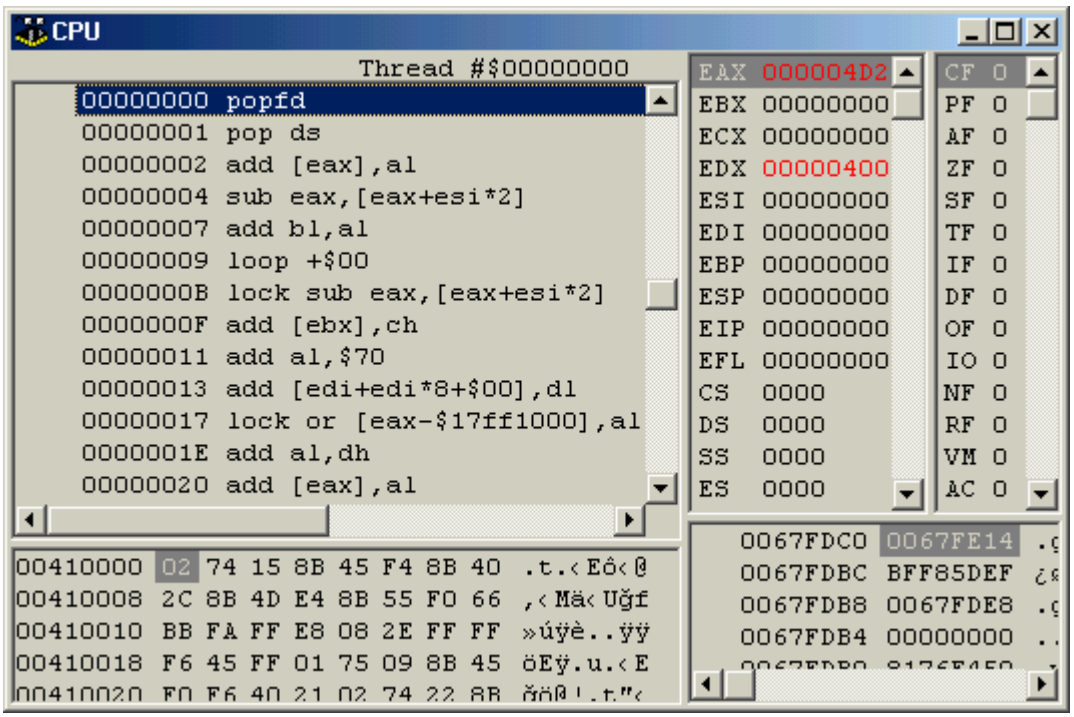


CPU PENCERESİ :

CPU penceresi programın makine dili seviyesinde kontrolünü sağlar.Delphi ile birlikte verilmesine rağmen ciddi bir makine dili bilgisi ister.

İzlemek için;

- program çalışırken,
- Ana menüden **View/Debugwindows/CPU** seçenekleri girilir,
- **CPU** formu ekrana gelir,

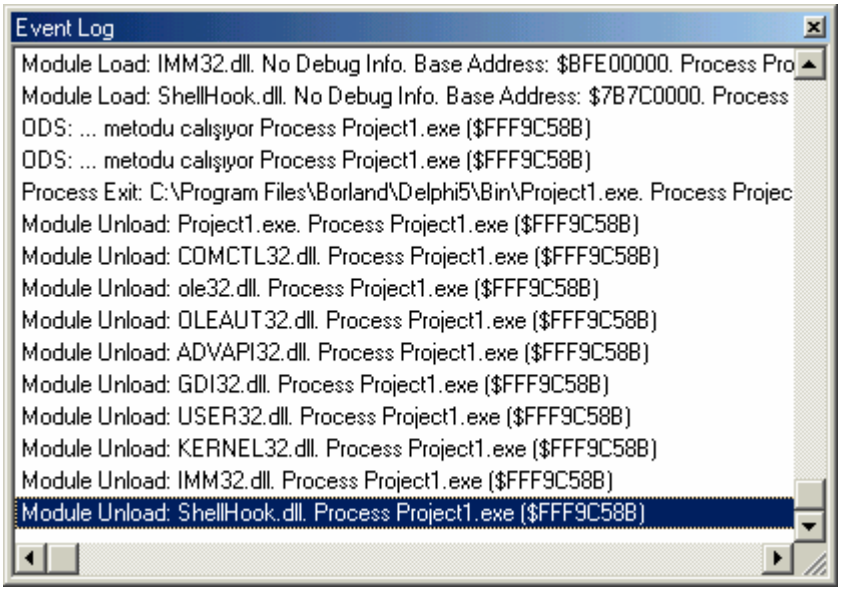


OUTputDebugString:

Programda hata ayıklamaya çalışırken zaman zaman program akışının hangi metotlara sıçradığını bilmek isteyebiliriz.Bu durumda kullanılacak en basit yöntemlerden birisi metotların içine girerek **showmessage** fonksiyonu ile birşeyler yazmak olacaktır.Ancak **OutputDebugString** fonksiyonu ile hangi metotların içine girildiği **Event Log** içine yadırmak mümkündür.Bunun için şu basit uygulama yazılabilir

.

```
Procedure Tform1.ButtonClick(Sender:Tobject);
Begin
OutputDebugString(Pchar('su anda Button1Click metodu çalışıyor'));
end;
```



CLİPBOARD

Clipboard, uygulamaların üzerine bazı bilgileri yazabilecekleri ve daha sonra yazmış oldukları bu bilgileri okuyabilecekleri hafıza alanlarıdır. Windows altında çalışan tüm uygulamaların kullanabileceği bu hafıza alanına değişik uygulamalara ulaşarak değişik bilgiler yazılabilir. Ancak bu durumda Clipboard'a yazılan bilgilerin formatının , bu bilgiyi paylaşacak olan uygulamalar tarafından bilinmesi gerekir. Windows 95 otomatik olarak clipboard içindeki bilgilerden bazılarını tanımaktadır. Bunlar içinde en sık kullanılanlar aşağıda belirtilmiştir:

CF_TEXT.....Düz yazılar
CF_BITMAP..... Bitmap tipindeki grafikler
CF_METAFILE..... Metafile tipindeki grafikler
CF_PICTURET..... Picture tipindeki herhangi bir nesne

Delphi kodu olarak kullanılabilecek deyim grubu aşağıdaki şekildedir.

```
Memo1.selectall;
Memo1.copytoclipboard;
Memo2.pastefromclipboard;
Memo1.selstart:=02;
Memo1.sellength:=2;
Memo1.copytoclipboard;
```

Resim uygulamalarının **Clipboard**'a aktarılmasında dikkat edilecek noktalar vardır.

- Uses paragrafına **ExtCtrls** ve **ClipBrd** modülü eklenmelidir.
- **Stretch** deyimi resmin tam oturmasını sağlar.
- **Hasformat** metodu ile clipboard içindeki bilginin bitmap görüntü olup olmadığını kontrol eder.İçindeki resim ise **True** değeri geri döner.
- Clipboard' sadece yazı yazılmak isteniyorsa Clipboard'ın astext özelliği kullanılır.

Clipboard.astext:='DELPHI';

Clipboard' Windows'un bize verdiği standart tipler (CF_TEXT, CF_BITMAP gibi) kendi verilerimizi de atayabiliriz.

DDE (DYNAMIC DATA EXCHANGE)

DDE, Windows programları arasında bilgi alışverişini sağlayarak daha güçlü uygulamalar geliştirmemizi sağlayan kavramdır. **DDE** yöntemleri bir uygulama içinde değişen bir bilgi diğer bir uygulama tarafından, bilgi değiştiği anda alınabilir. Ya da bir uygula ikinci bir uygulamanın belli bir işi yapması için, ikinci uygulamanın izin verdiği ölçüde emir gönderir.

DDE uygulamalarında kullanılan kavramlar:

Conversation: DDE sırasında iki uygulamanın kendi arasında haberleşirler. Bu bağlantıya conversation adı verilir.

Bağlantı sırasında uygulamalardan birisi bilgi sunmakta diğeri ise bu sunulan bilgiyi almaktadır. Bunlara **Sunucu Uygulama** ve **İstemci Uygulama** adı verilmektedir.

Sunucu uygulamalar **Servis** adı da verilmektedir. Genelde servisler, çalıştırılabilir EXE uzantılı dosyaların EXE olmadan kalan kısmıdır. (örneğin EXCEL.EXE uygulamasını sunucu olarak kullanacaksak EXCEL olacaktır. Bazende yalın adları olmayabilir.)

DDE Topic**:** Iki uygulama arasında paylaşılacak olan bilginin ne olduğunu göstermektedir.(Delphi'de topic olarak form adları kullanılmaktadır.)

DDE Items: uygulamalar arasında paylaşılacak gercek bilginin adıdır. **DDE Items'ler** sunucu uygulamanın davranışına göre değişir. (Hesap tablosunda hücre adresi, veri tabanında alan adı olabilir)

Örnek: Excel tablosunun bir hücresinde bulunan bilgiyi uygulama içine alalım.

Bunun için;

- Excel programında 1. Hücreye 'KONYA' yazalım. ENTER yapalım. Bunu Clipboard' atalım. (Ana menü/düzen/kopyala)
- Delphi uygulasına dönelim.
- Formun üzerine **Tedit** bileşeni yerleştirelim.
- System sekmesinden bir tane **TDDEClientConv** bir tanede **TDDEClientItem** bileşenleri yerleştirelim. **TDDEClientConv** excel sunucu uygulama ile bizim yazdığımız istemci uygulama arasında bağ kuracaktır.
- **DDEClientConv1** bileşeninin **DDEService** özelliği üzerinde üç noktalı yere tıklayalım .
- Gelen formdan **DDEService** ve **DDETopic** adlarını girelim. Yada paste link butonu tıklanırsa bu değerler otomatik yazılır. Burada excel'den clipboard'a kopyalanan bilgi aktif hale gelmiş olur.
- **DDE Info** ekranını **OK** butonuna basarak kapatalım.
- **DDEClientitem1** bileşeni üzerine gelelim ve **Object Inspecter**'dan **DDEConv** özelliğine gelerek değerini **DDEClientConv1** yapalım.
- **DDEitem** özelliğine gelelim listeden **R!C!** hücresini seçelim. (excelde a1 hücresi ile r1c1 hücresi aynıdır. Yani satırlar r1,r2,r3 sütunlar c1,c2,c3 olarak adlandırılabilir.)
- **DDEClientitem1** bileşenini seçelim. **Object Inspecter**'dan **Onchange** olayı üzerinde çift tıklayalım ve aşağıdaki kodu yazalım.

```
procedure TForm1.DdeClientItem1Change(Sender: TObject);
begin
edit1.text:=ddeclientitem1.text;
end;
```

Program çalıştırıldığında excel ile uygulamamız arasında bağlantı otomatik sağlanmış olacaktır. Bu durumda Excel sunucu uygulamamız istemci olmuştur. Excel hücre bilgisi otomatik olarak uygulamanın edit alanına aktarılmıştır. Excel bilgisini değiştirip enter tuşuna basalım uygulamamıza dönüp çalıştırırsak, uygulamaya değişiklik otomatik olarak yansımıştır.

DLL (DİNAMİK BAĞLANTILI KÜTÜPHANELER)

DLL NEDİR? :

DLL'ler içinde kod ve windows kaynakları bulunduran,bir çok windows uygulaması tarafından paylaşılabilen program modülleridir.

DLL kullanmanın en önemli amaçlarından birisi, uygulamaların çalıştırılacak kodları, programın derlenme zamanında değil, çalışma sırasında yüklenebilmesidir. Böylece hafıza daha verimli kullanılmış ve aynı zamanda modül kendisini kullanan her uygulama derlenirken içine eklenmediğinden **EXE** dosyaların daha az yer kaplamasını sağlar.

Bir başka avantaj ise uygulamalrın daha modüler olmasıdır.

Disk üzerinde **DLL**'lerin yapısı, windows altında çalışan herhangi bir EXE uzantılı dosya ile aynıdır. Aradaki fark DLL'ler tek başlarına çalıştırılamazlar. Windows ise bu iki dosya tipini içindeki başlıklardan anlar ve çalışıp çalışmayacağına karar verir.

DLL'ler sadece bir kez belleğe yüklenir. Çok sayıda uygulama bu DLL'e başvurabilir.

Farklı programlama dilleri arasında kullanılabilir.

DLL'in uzantısının mutlaka DLL olması gerekmez. Windows altınta **DRV** veya **FON** uzantılı dosyalarda birer **DLL**'dir.Aynı şekilde Windows'un kendi içinde bulunan **USER.EXE**, **KERNEL.EXE, GDI.EXE** dosyalarıda birer **DLL** dosyadır.

DLL hafızaya yüklendikten sonra diğer uygulamalar arasında kendi kodlarını paylaşabilir. Bunu dinamik bağlar sayesinde gerçekleştirir. Uygulamaların kendi EXE'leri içinde bulunan fonksiyon ve prosedürlerin tümü EXE'ye statik bağla bağlanmıştır. Bu fonksiyonlar uygulamamız derlenirken modüllerimiz içinde bulunduğu için, derleyici, program akışı sırasında bu fonksiyonlar

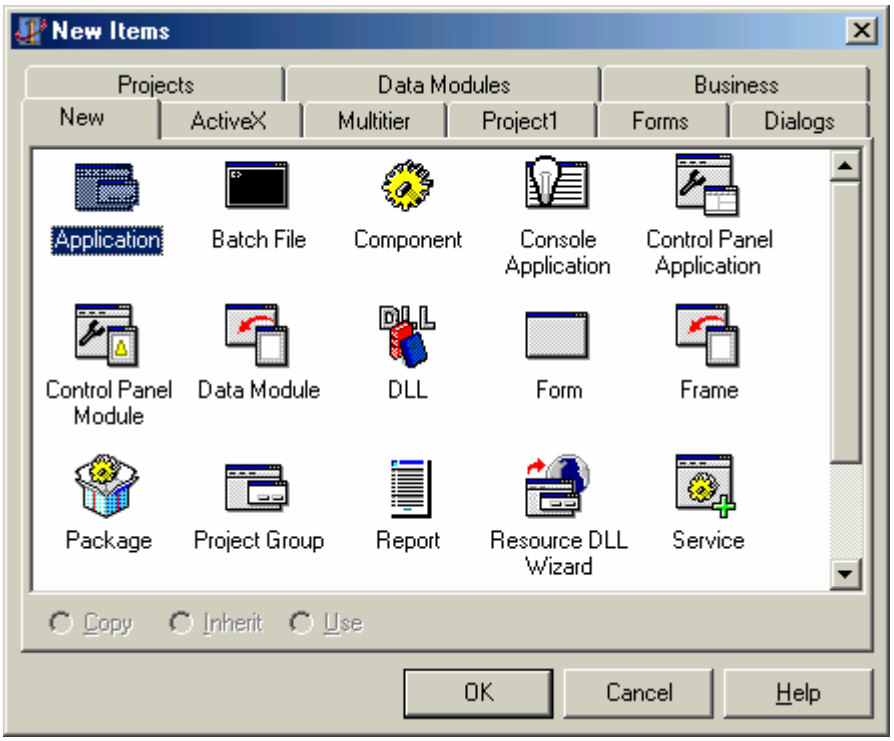
çağrıldığı zaman nereye gitmesi gerektiğini bilecektir. Bu şekilde derlemeler sonucunda kodumuz içinde bulunan fonsiyon ve prosedürler, oluşan çalıştırılabilir EXE dosyanın bir parçası olacaktır. Bu şekilde oluşturulan EXE dosyalar hafızaya yüklendiğinde de bu fonksiyon ve prosedürlerin hafızadaki konumları bilindiği için programın çalışması sırasında hafızada hangi adrese gidilmesi gerektiği belli olacaktır. Bu nedenle, aynı modülleri kullanan iki EXE dosya varsa bu modüller her iki dosyada da olacağından hem disk üzerinde fazla yer kaplamış olacak hemde hafızada fazla yer kaplayacaktır.

Dinamik bağlarda ise programın derlenme aşamasında derleyici, fonksiyonun hangi modül içinde hangi adreste olduğunu tam olarak bilmez. Sadece dinamik olarak bağlanacak herhangi bir fonksiyonun, programın çalışma sırasında hafızaya yüklenecek olan bir DLL içinde olduğunu bilir. Bu nedenle programın derlenme aşamasında sadece bu fonksiyonun hangi DLL içinde bulunduğunun belirtilmesi yeterli olur.

DLL OLUŞTURMA

DLL dosyaları içlerinde fonksiyon bulundurduğu için normal program gibi yazılmazlar. Bunun için;

- Anamenüden **File** seçeneği seçilir. Buradan **New** seçilirse **New Items** ekranı görüntülenir.



- Gelen ekrandan DLL simgesi seçilir. Bu seçimin sonunda DLL kodu yazmak için hazır kod şablonu görüntülenir.

```
library Project2;

{ Important note about DLL memory management: ShareMem must be the
  first unit in your library's USES clause AND your project's (select
  Project-View Source) USES clause if your DLL exports any procedures or
  functions that pass strings as parameters or function results. This
  applies to all strings passed to and from your DLL--even those that
  are nested in records and classes. ShareMem is the interface unit to
  the BORLNDMM.DLL shared memory manager, which must be deployed along
```

- İlk satırdaki **Library** kelimesi bunun bir DLL olacağını söylemektedir.

- Yazılacak fonksiyon, prosedür veya değişken tanımlarının **Begin** satırından önce yazılması ve eğer bu fonksiyonlar dışarıdan çağrılabilecekse fonksiyon tanımının sonunda **Export** deyimi ile bildirilmesi gerekir.

- Yazılan fonksiyonların sonunda, **Begin** satırından önce dışarıdan çağrılabilecek fonksiyonların **Exports** altında bildirilmesi gerekir.

- **DLL** dosyaları **F9** ile çalıştırılmazlar. Derlemek için **Ctrl+F9** tuşunu veya **Project-Compile** seçeneği kullanılır. Bu işlem sonunda hazırlanan modül DLL uzantısı ile derlenir.

- DLL dosyası hazırlanırken bir yandan da test etmek zor olduğundan fonksiyonları bir projede hazırlayıp en son haliyle bir DLL projesine kopyalamak daha uygun olur.

- Herhangi bir DLL dosyasından bir fonksiyon çağırmadan önce onu aşağıdaki şekliyle tanımlanması gerekir.

```
Function AD(parametreleri:Tipi):Tipi;far;external 'DLL adı' index no;
```

DLL ÖRNEĞİ -I-

Herhangi iki sayıyı toplama işini yapan bir DLL hazırlayalım.
Bu işlem için;

1. Ana menüden **File/New** seçeneğinden DLL kod penceresi görüntülenir.
   Kod penceresini aşağıdaki şekilde düzenleyelim. Topla_dll adı ile saklayalım.
   Ctrl+F9 veya **Project/Compile** ile derleyelim.

```
library Project2;

uses
  SysUtils,
  Classes;

 function topla(x,y:integer):integer;export;
 begin
```

2. Bu DLL dosyasını test etmek için küçük bir proje hazırlayalım ve çalıştıralım.

```
function topla(x,y:integer):integer;far;external 'topla_dll.dll';
procedure TForm1.Button1Click(Sender: TObject);
begin
showmessage(inttostr(topla(9,12)));
end;
```

NOT: DLL dosyasındaki **Export** ifadesi bu fonksiyonun dışardan çağrılacağını bildiriyor. Dışardan çağrılabilecek fonksiyonları Exports deyimi ile bildirmek gerekir. Her iki ifade de begin satırından önce olmalıdır.

DLL DOSYASINDA FORM KULLANMA

DLL dosyalarında formda kullanılabilir. DLL dosyasına form eklemek için;

- **File/New/Dll** menülerinden **DLL projesi** başlatın.
- **File/new form** menüleri ile DLL projesine form ekleyin.
- Formu dilediğiniz gibi hazırlayın.
- Form otomatik olarak oluşmayacağı için formu gösterecek kodu aşağıdaki şekilde form içine yazın.

```
procedure formugoster;
var
  Form1: TForm1;

procedure formugoster;
 begin
 form1:=Tform1.create(application);
 form1.ShowModal;
 form1.free;
 form1:=nil;
 end;
```

- DLL projesinden bu prosedürü çağırın.

WIN API (Windows Application Programming Interface)

API NEDİR? :

Dos işletim sistemi kendi fonksiyonlarını interruptlarıyla sunarken Windows API'leri ile sunar. Yani Dos işletim sisteminin yaptığı işleri Windows API'ler ile yapmaktadır. Windows'ta API'leri kullanmanız şart değildir Delphi bunları sizin yerinize kullanır.API'ler sınıflandırılarak DLL ve EXE dosyalara konmuştur.

Win API, windows altında program geliştirilirken kullanılan fonksiyonlar kümesi olarakta tanımlanabilir.

Windows altındaki programlama dilleri aracılığı ile bu fonksiyonlar çağrılabilir ve dönüş değerleri programlara aktarılarak kullanılabilir.

Windows altında program yazma dos altında program yazmadan farklıdır. Windows altında çalışan programlama dilleri windows'un kendi fonksiyonlarınıda kullanılır.

Windows'un sahip olduğu binlerce fonksiyon bir o kadarda mesaj vardır. Bunlarıjn tümü API olarak adlandırırlır.

Dil herhangi bir işlem yapmak istediğinde bu işi kendisi yapmaz. Windows'a haber gönderir. Gelen mesajlar doğrultusunda windows gereken işlemleri gerçekleştirir.

API'leri kullanmak ve bunları kullanarak program yazmak zor ve zahmetli bir iştir. Bu işlemler için Delphi'de **VCL** devreye girer. VCL'deki bileşenler aslında **API fonksiyonlarını** sararak paketler haline getiren sınıflardır.

Delphi ile API fonksiyonları çağrılabilir. Bu tür programlar hızlı çalışır ve az yer kaplar.

Programda API kullanmak için diğer görsel dillerde olduğu gibi API 'yi tanımlamak gerekmez . (ilk versiyonda gerekiyordu) Çünkü bu tanımlar yapılmış windows uniti altına kayıt edilmiştir. Programınızda Unit satırında varsayılan olarak windows unitinin tanımı bulunur. Uses satırımda Windows kelimesinin üzerinde iken sağ tıklanırsa açılan menüden **OPEN File At Cursor** komutu tıklanırsa API'lerin nasıl tanımlandığı kod penceresinden görülebilir. Bazı API tanımları bu dosyada bulunmaz ihtiyaç duyuluyorsa programcı tarafından API tanımlanabilir.

```
c:\program files\borland\delphi5\source\rtl\Win\Windows.pas
Unit1   Windows

{*******************************************************}
{                                                       }
{       Borland Delphi Run-time Library                 }
{       Win32 API Interface Unit                        }
{                                                       }
{       Copyright (c) 1985-1999, Microsoft Corporation  }
{                                                       }
{       Translator: Inprise Corporation                 }
{                                                       }
{*******************************************************}

unit Windows;

{$ALIGN ON}
{$MINENUMSIZE 4}
{$WEAKPACKAGEUNIT}

interface

(*$HPPEMIT '// WIN2K SDK fixes BEGIN ' *)
(*$HPPEMIT '#ifndef WINTRUST_H' *)
(*$HPPEMIT '#include<wintrust.h>' *)
(*$HPPEMIT '#endif' *)
(*$HPPEMIT '// WIN2K SDK fixes END. ' *)
(*$HPPEMIT '//' *)
(*$HPPEMIT '// WINDOWS provides two definitions of BLOB: wtypes.h defines a' *)
(*$HPPEMIT '// BLOB as a tagBLOB struct whereas nspapi.h defines a BLOB as a _BL
1: 1   Insert
```

Delphi'de API'ler iki şekilde tanımlanır. Fonksiyon veya prosedür olarak. Fonksiyon olarak tanımlanan API'ler geriye değer döndürürken prosedür olarak tanımlananlar geriye değer döndürmez.

Function isim(parametreler:tip):tip; **far**;**external** 'kütüphane adı';

Procedure isim(parametreler:tip); **far**;**external** 'kütüphaneadı';

Burada isim, API adı, kütüphane adı ise API'nin bulunduğu dosya adıdır. Bu tanımlardan sonra normal bir fonksiyon veya prosedur gibi çağrılır.

API'ler hakkında bilgi almak için gerekli help dosyaları **Program files** dizini içindeki **Common Files\Borland Shared\MSHelp** alt dizininde mevcuttur. Bu dizindeki **Win32s.hlp** dosyası açılırsa **Standart Windows API'lerinin** bulunduğu yardım dosyasına ulaşılabilir. Buradan listelenen API'ler içinde arama yapılabilir veya belli bir API incelenebilir. Ancak API'ler **C** formatındadır.

API kullanımı için örnekler:

Örnek 1. Sistem ayarlarının değişimi.Bunun için **ExitWindowsEx()** fonksiyonu kullanılır.

Bunun için yeni bir oturum açalım. Form üzerine bir adet kapat butonu koyalım ve aşağıdaki kodları yazalım.

```
procedure TForm1.Button1Click(Sender: TObject);
var
   tus:integer;
begin
   tus :=Application.messagebox
      ('windows tan çıkmak istiyormusun ?',
         'windows oturumunu kapat',
            mb_yesnocancel+mb_defbutton1);
  if tus = 6 then exitwindows(42,1);
end;

end.
```

Örnek 2. Mikroişlemci tipini ve sayısını öğrenmek. Bu işlem için **GetSysteminfo API**'si kullanılır. **Systeminfo** yapısının Delphi karşılığı **Tsysteminfo** olarak tanımlanmıştır.

Bu uygulama için yeni bir oturum açalım .Bir buton oluşturalım ve bunun procedurunu aşağıdaki şekilde düzenleyelim ve çalıştıralım.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  s:tsysteminfo;
begin
   Getsysteminfo(s);
      caption:=inttostr(s.dwnumberofprocessors)+ 'tane'+
               inttostr(s.dwprocessortype)+'tipi işlemci';

end;

end.
```

Örnek 3. Windows'un kurulu olduğu klasörü öğrenmek. Bu işlem için **GetWindowsDirectory()** fonksiyonu kullanılır.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  yol:pchar;
begin
   yol:='......................';
   getwindowsdirectory(yol, length(yol));
```

Örnek 4. Program çalıştırmak. Bu uygulama için **WinExec() API** fonksiyonundan yararlanılır. Bu API dışardan bilgi olarak iki parametre almaktadır. Birinci parametre çalıştırılacak programın adı, ikinci parametre çalıştırılacak dosyanın ilk şeklini belirtmek.şekil belirtmek için **3**(uygulama ekranı kaplar), 6(programa ait pencere simge durumuna küçültülür), 9 (uygulama orijinal boyutları ile ekrana gelir.) sayıları kullanılır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
winexec('c:\win98\notepad.exe',9);
end;
```

Örnek 5. Görev çubuğunu gizleme ve gösterme.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 //gizle
showwindow(findwindow('shell_traywnd',nil),sw_hide);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 //goster
showwindow(findwindow('shell_traywnd',nil),sw_show);
end;
```

Örnek 6. Masaüstü simgelerini gizlemek.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 //gizle
showwindow(findwindow(nil,'program manager),sw_hide);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
```

Örnek 7. CD kapağını açıp kapatmak. Bu işlem **mcisendstring API'**si ile yapılabilir.

Bu api **MMSystem** unitinde tanımlıdır. O nedenle oturumun **Uses** başlığı altındaki satıra **MMSystem** ilavesi yapılmalıdır.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 //aç
mcisendstring('set cdaudio door open', nil,0,handle);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 //kapat
mcisendstring('set cdaudio door closed', nil,0,handle);

end;
```

BİLEŞEN YAZMA TEKNİKLERİ

Delphi'nin getirdiği bileşen teknolojisi sayesinde, yazılım pazarında Delphi kullanarak uygulama geliştiren programcılar iki grupta toplanmaya başladı. Delphi bileşenleri kullanarak uygulama geliştiren programcılar ve Delphi programcılarına bileşen yazan programcılar.

YENİ BİLEŞENLER YAZMAK

Delphi, varolan bileşenleri kullanarak uygulama geliştirebileceğimiz bir ortam dışında, yeni bileşenleri yaratabilme olanağı sağlar.

Bileşenler, Delphi uygulamalarının yapıtaşlarıdır. Kullanıcılar için bileşenler palet üzerinden alarak form üzerine yerleştirilen nesnelerin her biridir. Bileşen üreten kullanıcılar için ise bileşenler nesneye yönelik kaynak kodları ifade etmektedir.

Bileşenlerin tümü **Tcomponent** adlı bir sınıftan üretilmiştir. **Tcomponent** bileşeni, bileşenlerin kendilerini palet üzerinde görüntüleyebilmeleri ve uygulama geliştirme aşamasında form üzerine yerleştirilebilmeleri için gerekli temel özelliklere sahiptir.

Bileşen oluşturma görsel olmayan bir ortamda gerçekleşir. Object Pascal programlama dili kullanılır. Ancak yeni Delphi bileşenleri yaratmak detaylı bir nesne yönelik programlama mantığı ve windows API fonksiyonlarının kullanılmasını gerektirir.

VCL (GÖRSEL BİLEŞEN KÜTÜPHANESİ)

Delphi bileşenleri **VCL** olarak adlandırılan bir hiyerarşi içinde bulunurlar. Bu hiyerarşi içinde en tepede **TObject** adlı sınıf bulunur.

Ancak Delphi bileşenleri bu hiyerarşide **Tcomponent** sınıfından türetilecektir. Zira bu sınıf bileşenlerin Delphi ortamını kullanılabilmeleri için gerekli minimum özellik , metot ve olayları kapsamaktadır. Bileşen yazanlar ; varolan bir bileşeni değiştirebilirler, yeni bir bileşen oluşturabilirler veya görsel olmayan bileşenler oluşturabilirler.

Var olan bir bileşeni değiştirmek: Bileşen yaratmanın en kolay yoludur. Bileşenin bazı özelliklerinin teni bileşende olması istenmiyorsa eski bileşenin bir üst atasına gitmeli ve bileşenimizi bu sınıftan üretmeliyiz.Yeni bileşende olmasını istemediğimiz bazı özellikler varsa bir önceki atası baz alınmalı ve bileşen bu sınıftan üretilmelidir.

Delphi bileşenlerin temel özelliklerini **TCUSTOM** ile başlayan sınıflara yerleştirmiştir.

Yeni Kontrol Yaratmak: Standart kontroller, program çalışırken ekranda gördüğümüz ve etkileşim içinde bulunduğumuz nesnelerdir. Bu nesneler **TcustomControl** sınıfından türetilmiştir. Bu nedenle yeni bir kontrol yaratılacağında **TcustomControl** sınıfı baz alınmalıdır. Standart kontroller, top, left, Hight ve Width özellikleri olan pencerelerdir. Bu pencerelerin her birinin **Handle** özelliği vardır. Bu özellik word değişken tipindedir.Windows altında yaratılan her pencere için Handle özelliği bir değere sahip olur. Bu handle sayesinde windows sistemi, pencereyi tanır ve pencereye mesaj gönderirken handle özelliğinden yararlanır.Yine handle özelliği sayesinde aktif kontrol (input focus) nesneler üzerine geçebilir ve windoes API fonksiyonlarına yaratılan bir nesneyi parametre olarak geçirebiliriz.

Görsel olmayan bileşenleri Yaratmak: Tcomponent bileşeni tüm bileşenleri türetildiği ana bileşendir. Buradan direkt üretilen bileşenler görsel olmayan bileşenlerdir. Ancak bunların sayıları görsellere göre azdır.

BİLEŞEN HİYERARŞİSİ

Delphi bileşenlerinin tümü aslında en tepede bulunan **TObject** adındaki bir sınıftan üretilir. Oluşturulacak bileşenler genellikle işimize çok yarayan bir başka bileşenin kalıtımı yoluyla elde edilebilir. Bu nedenle programcılar delphi bileşenleri özellik ve metotlarını, bileşen hiyerarşisini iyi bilmelidir. Bileşen hiyerarşisi aşağıdaki şekildedir.

1. Tobject
2. Tpersistent
3. Tcomponent
4. Tcontrol
5. TwinControl
6. TgraphicControl

Tobject sınıfı tüm bileşenlerin atasıdır. Bu sınıfın iki önemli metodu vardır. **Create** ve **Destroy**. **Create** metodu ile sınıfa ait nesne örneklerini yaratmak mümkün olmaktadır. **Destroy** ilede bu Nesne örneklerinin hafızada kapladığı alan serbest bırakılmaktadır. Tüm bileşenlerin bu metotları mevcuttur.

Tpersistent sınıfı **Tobject** sınıfından direkt olarak türetilmiştir. Bu sınıfın en önemli özelliği, **Stream** adındaki bilgi kümelerini okuyabiliyor ve yazabiliyor olmasıdır. Bu sınıfın metotları sayesinde, bileşenlerin özelliklerini değiştirdiğimizde bu özelliklere ait değerlerin tutulduğu **DFM** uzantılı dosyaların diske yazılması ve diskten okunması mümkün olmaktadır.

Tcomponent sınıfı, **Tpersistent** sınıfından direkt türetilmiştir. Bileşenlerin kendilerini bileşen paleti üzerinde görüntüleyebilmesi ve uygulama geliştirme aşamasında kendilerini form üzerine yerleştirebilmesi özelliklerini değiştirebilmesi gibi işlemler mümkün olmaktadır. Bu nedenle bileşen yazmayı düşünen programcılar bileşen yazmaya en azından **Tcomponent** sınıfından türetmelidir.

Bu sınıftan türetilen bileşenler programın çalışması sırasında ekranda görünmezler . (TTable gibi) Tcontrol bileşenler sayesinde bileşenlere kendilerini programı çalışma esnasında form üzerinde görüntüleyebilmeleri için bazı özellik ve metotlar eklenmiştir. Bu özellikler genel olarak , bileşenin programın çalışma esnasında form üzerinde hangi koordinatlarda bulunacağını gösteren Top , Left, Width, Hight özellikleri , color ve visible gibi özelliklerdir. Ayrıca programın çalışması esnasında etkileşim sonucu oluşan **OnClick, OnDblClick, ONMauseUp** gibi olaylar da bu bileşenin olayları arasındadır.

Tcontrol sınıfından iki önemli sınıf üretilmiştir. **TwinControl ve TgraphicControl**. **TwinControl**, windows sistemi tarafından tanınabilen bileşenlerin türetildiği sınıftır. Windows sistem içinde türetilen her standart bileşen için bir numara verir. Bu sayede işletim sistemi o bileşeni tanır ve windows API fonksiyonları o bileşen için kullanılabilir. Bu numara bileşenlerin handle özelliğinde tutulmaktadır. (Buttun1.Handle, Edit1.Handle gibi)İşte bu sınıftan türetilen bileşenler için handle özelliği tanımlanmıştır. Bu sayede delphi bileşenleri ile windows arasında ilişki kurulabilmektedir.

Oluşturulan her bileşen için Windows API fonksiyonlarını kullanmak gerekmeyebilir.Handle özelliğine ihtiyaç olmayabilir ve böylece windows kaynakları daha az kullanılabilir. Bunun için Tcontrol sınıfından birde **TgraphicControl** sınıfı türetilmiştir.Bu sınıfa ait bileşenlerin handle özelliği yoktur. Bu nedenle bu bileşenler için Windows API fonksiyonlarını çalıştıramayız.

Delphi'deki **VCL** hiyerarşisi yukarıdaki sayılanlardan daha fazladır. Ancak yukarıdaki sınıflar temeldir. Bu bileşenlerin genel özelliklerinin iyi bilinmesi çok önemlidir.

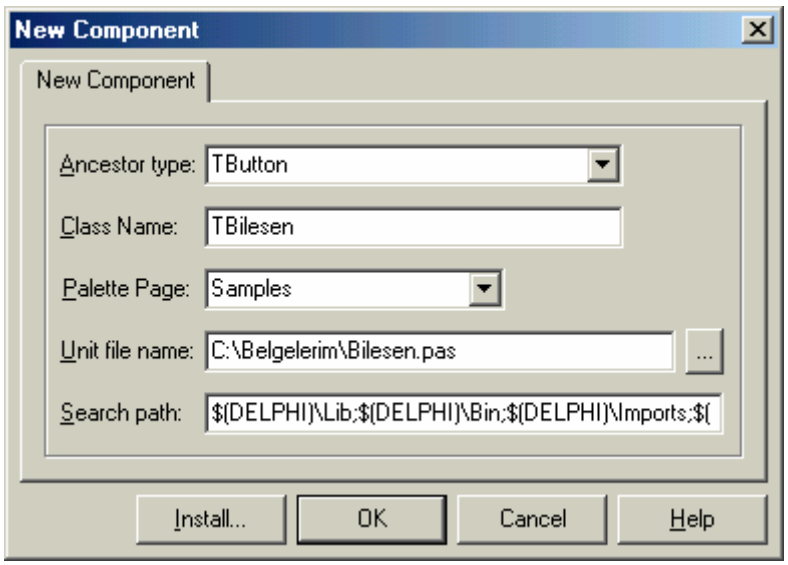
YENİ BİR BİLEŞEN YARATMAK

Örnek: TButton bileşeni ile aynı özelliklere sahip bir bileşen yaratalım.

1. Ana menüden **File/Close All** ile açık olan proje varsa kapatılır.

2. Ana menüden **Component/New Component** seçeneği seçilir.

3. **New Component** penceresi görüntülenir.



Bu pencerede; **Ancestor Type** listesinden Tbutton sınıfı seçilir.Class Name karşısına Tbutton1 yazılmıştır. Bu ismi Tbileşen olarak değiştirelim.Bu oluşturulacak bileşenin sınıfıdır. Palette Page seçeneğinde ise hazırlanan bileşen hangi sekmede görülecek ise o sekme adı seçilir.

4. Gerekli seçimlerden sonra **OK** düğmesi tıklanırsa kod penceresi açılır. Bu kodlar **File/Save File As** seçeneği ile ilkbilesen **.pas** olarak saklayalım. İlk Bileşenin kodu aşağıdaki şekildedir.

```
unit Bilesen;

interface

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

type
  TBilesen = class(TButton)
  private
    { Private declarations }
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
```

Kodu inceleyecek olursak:
Tbileşen nesnesinin Tbutton tipinde olduğu belirtilmektedir.

Private: Bu bölüme yazılan özellikler ve metotlar ancak bu sınıfın (Tbilesen) tanımlandığı modül (ilkbilesen.pas) içinde geçerlidir. Diğer modüllerdeki herhangi bir sınıf yada Tbileşen nesnesine ait bir nesne örneği bu bölümde tanımlanmış olan özellikleri ve metotları çağıramaz.

Protected: Bu bölümdeki özellik ve metotlar da private bölümündekine benzer. Ancak aralarındaki fark protected bölümünde tanımlanan özellik ve metotlar bu sınıftan üretilecek yeni sınflar tarafından (Tbilesen sınıfının torunları tarafından) çağrılabilirler.

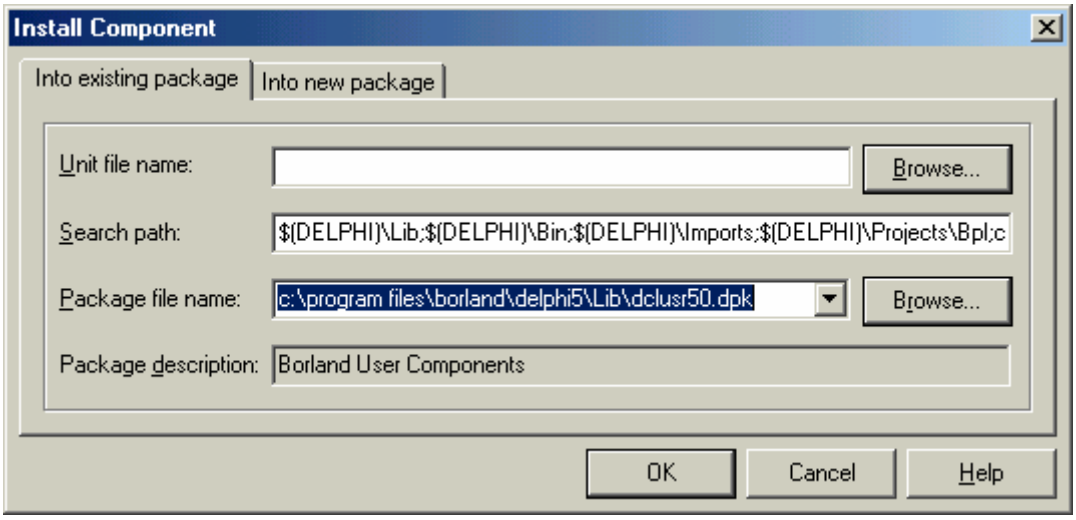
Public: Bu bölümde tanımlanan özellik ve metotlar tanımlanan modülü kullanan tüm modüller içinde kullanabilirler.

Published: Bu bölümde tanımlanan özellik ve metotlar modülü kullanan tüm modüller içinde kullanılabilir.Ayrıca bu özellik ve metotlara uygulamaları geliştirme aşamasında da ulaşılabilir. Object inspectorda bulunan tüm özellikler bu bölümde yaratılmışlardır.

Yukardaki örnekte Tbilesen bileşeni Tbutton adındaki daha önce var olan bir bileşenden yaratıldığı için bahsedilen başlıklara herhangi bir özellik ve metot eklenmemiştir. Tbilesen Tbutton ile aynı özellik ve metotlara sahiptir.

Bu aşamada yapılması gereken bileşenin windows ve Delphi'ye tanıtılmasıdır. Bunun için kullanılan register prosedürü yine delphi tarafından otomatik olarak üretilmiştir. Bu prosedürde Tbileşen adındaki yeni bileşenin Samples adlı sekmede görünmesi **Registercomponent** komutuyla sağlanmaktadır.

Bundan sonra yapılması gereken ise hazırlanan bileşenin **VCL** hiyerarşisinde yerini almasını sağlamaktır. Bunun için anamenüden / **Install Component** seçeneği seçilir. Install component ekranı görüntülenir.



Bu ekranda biraz önce oluşturulan ilkbileşen **.Pas** dosyasını browse seçeneğine girerek seçtikten sonra **OK** düğmesine basıldığında Delphi VCL hiyerarşisini tekrar oluşturacak ve Tbilesen bileşenini **Samples** sayfasına yerleştirecektir.

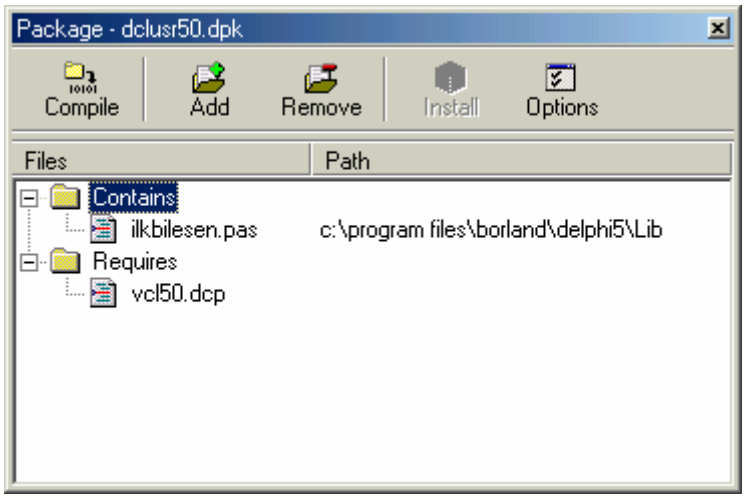
Oluşturulan bileşeni kontrol etmek için ise;

1. **Samples** sekmesine yerleştiği için nesne olarak forma aktarılır kullanılır. Yalnız oluşturma esnasında hata yapılmış ise Delphi kitlenebilir ve makinayı açıp kapamak zorunda kalabilirsiniz.
2. Kaynak kodlar oluşturularak test edilebilir.
   Bunun için ana menüden **File/New Application** ile yeni bir proje başlatalım.
   Bileşenin modül adını unit'in uses bölümüne ekleyelim.
   Form üzerine Samples sekmesindeki Tbilesen'lerden birini ekleyelim.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ilkbilesen;

type
  TForm1 = class(TForm)
    Bilesen1: TBilesen;
    procedure Bilesen1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

Kodu yukardaki sekilde düzenleyelim. Ve çalıştıralım.

Normalde Delphi bileşenleri **Packages** adındaki DLL'e benzeyen modüller içinde tutar.
Bileşenlerimiz **DCLUSR50.DPK** adlı paket modülü içine yerleşmektedir.
Bunu görmek için **Anamenü/File/Open** girilerek
C:\programfiles\borland\delphi5\lib\dclusr50.dpk dosyası açılabilir.



Bileşen yazma ile ilgilenmek isteniyorsa Delphi ile birlikte gelen kodların incelenmesi gerekir. Bu yararlı bir iştir. Çünkü bu kodlar tüm bileşen yazma tekniklerini kapsamaktadır. Delphi ile gelen Tedit , Tbutton gibi bileşenlerin bile kaynak kodu incelenebilir. Bu tür kaynak kodlar Delphi'nin kurulduğu dizin altında **Source/VCL** altında bulunur.Ayrıca bu konu ile ilgi yabancı kaynaklardaki kodlarda incelenebilir.

DELPHİ'DE FLASH DOSYALARININ GÖSTERİLMESİ

İlk önce **SWFLASH.OCX** dosyasını bulun. Genellikle ***C:\Windows\System\Macromed\Flash*** klasöründe bulunur. Daha sonra Delphi'de iken **Component --> Import ActiveX Control** komutunu tıklayın. **SWFlash.OCX** dosyasını seçin ve import edin. Dosyayı birkez yükledikten sonra bir adet **TShockWaveFlash** bileşenine sahip olursunuz ve diğer Delphi bileşenleri gibi onu da form üzerine sürükleyerek bırakabilirsiniz.

PROBLEM: Eğer uygulamanızı dağıtmak isterseniz önemli bir konu, adı geçen **OCX** dosyasının sistemlerde kurulmuş olması gerekmektedir. Aksi halde programınız çalışmayacaktır.

ÇÖZÜM: Programınızı basit olmasını sağlamak ve tek bir dosya halinde dağıtmak için **OCX** dosyasını **Resource** haline getirmek ve sonra EXE'ye dahil etmek. Gerektiğinde dosyayı tekrar açmak ve register etmek. Bu işlemi yapmak için gereken kodlar aşağıda verilmiştir.

RESOURCE DOSYASI OLUŞTURMA

NotePad'de yeni bir text dosya oluşturun ve içine şu satırı yazın:

Flash RCDATA "SWFLASH.OCX"

Dosyayı kaydederken uzantısını **.RC**, örneğin, **FlashOCX.RC** olarak kaydedin. Çalışırken **SWFLASH.OCX** dosyasıyla bu dosyanın aynı klasörde olması gerekmektedir. Komut satırına (DOS

penceresi) geçin ve **brcc32.exe** dosyasını Delphi'yi yüklediğiniz klasör altından bulun. Aşağıdaki komutu yazın ve çalıştırın:

C:\> brcc32 ***c:\windows\system\flashocx.rc***

Şimdi **FLASHOCX.RES** adında yeni bir dosya yaratılmış olacak. Artık, bu resource dosyasını programımıza şu şekilde dahil edebiliriz:

```
{$R *.RES}
{$R FLASHOCX.RES}
```

NOT:
R direktifi, Delphi'ye Resource dosyasını/dosyalarını programa dahil etmesini bildirmektedir.

DOSYANIN ÇIKARTILMASI ve REGISTER EDİLMESİ

İlk yapacağımız, bir Flash dosyasını oynatabiliyor muyuz, oynatamıyor muyuz? test etmeliyiz. Şimdi, **Project --> View Source** (Delphi 5'te) komutunu verelim ve projenin kaynak koduna eklemeler yapalım. Şimdi, form ilk oluşturulurken ***EOleSysError*** hata mesajı alıp almadığımızı kontrol edeceğiz. Eğer böyle bir hata alırsak anlamalıyız ki adı geçen OCX sisteme register edilmemiş ve biz register etmeliyiz.

```
uses comobj;
begin
 Application.Initialize;
 try
   Application.CreateForm(TForm1, Form1);
 except
   On EOleSysError Do
   begin
      // OCX register edilmeli, çünkü dosya bulunamadı.
   end;
 end;
 Application.Run;
end.
```

Aşağıdaki kodlarda ise **OCX** dosyası programdan çıkartılmakta ve Windows Sistem Klasörüne kopyalanmaktadır.

```
uses windows, classes, sysutils;

var
 aSystemDirZ : array[0..2047] of Char;
 fSystemDir  : String;

...

GetSystemDirectory ( aSystemDirZ, 2047 );
fSystemDir := aSystemDirZ;

ResStream := TResourceStream.Create(0, 'Flash', RT_RCDATA);
try
```

```
  FileStream := TFileStream.Create(fSystemDir+'SWFLASH.OCX', fmCreate);
  try
    FileStream.CopyFrom(ResStream, 0);
  finally
    FileStream.Free;
  end;
finally
  ResStream.Free;
end;
```

KODUN TAMAMI

Hâlâ OCX'i sisteme register etmedik. Aşağıda hem kontrol işlemi yapılmakta, hem dosya çıkartılmakta ve hem de sisteme register edilmektedir. Bir OCX'i sisteme register etmek için **regsvr32.exe** programı kullanılır.

Lütfen kodları iyice anlamaya çalışın. Çünkü, başka zamanlarda da işinize yarayabilir.

```
program FlashPlayer;

uses
 Forms, Dialogs, comobj, windows, classes, sysutils,
 uMain in 'uMain.pas' {Form1};

{$R *.RES}
{$R FLASHOCX.RES}

type
 TRegFunc = function : HResult; stdcall;

function WinExecAndWait32( FileName: String; Visibility  : Integer ) : Cardinal;
var
  zAppName     : array[0..512] of char;
  zCurDir      : array[0..255] of char;
  WorkDir      : String;
  StartupInfo  : TStartupInfo;
  ProcessInfo  : TProcessInformation;
begin
  StrPCopy( zAppName, FileName );
  GetDir  ( 0,        WorkDir  );
  StrPCopy( zCurDir,  WorkDir  );

  FillChar( StartupInfo, Sizeof( StartupInfo ), #0 );

  StartupInfo.cb          := Sizeof( StartupInfo );
  StartupInfo.dwFlags     := STARTF_USESHOWWINDOW;
  StartupInfo.wShowWindow := Visibility;

  if ( not CreateProcess( nil,
                          zAppName, { pointer to command line string }
                          nil, { pointer to process security attributes }
                          nil, { pointer to thread security attributes }
```

```
                    false, { handle inheritance flag }
                    CREATE_NEW_CONSOLE or { creation flags }
                    NORMAL_PRIORITY_CLASS,
                    nil, { pointer to new environment block }
                    zCurDir, { pointer to current directory name }
                    StartupInfo, { pointer to STARTUPINFO }
                    ProcessInfo ) ) then
  begin
     Result := $FFFFFFFF; { pointer to PROCESS_INF }
     MessageBox( Application.Handle,
              PChar( SysErrorMessage( GetLastError ) ), 'Oopps!', 0 );
  end
  else
  begin
     WaitforSingleObject( ProcessInfo.hProcess, INFINITE );
     GetExitCodeProcess ( ProcessInfo.hProcess, Result   );
     CloseHandle       ( ProcessInfo.hProcess          );
     CloseHandle       ( ProcessInfo.hThread           );
  end;
end;

var
 aSystemDirZ : array[0..2047] of Char;
 aShortPath  : array[0..2047] of Char;
 fSystemDir  : String;
 aCommand    : String;
 aHandle     : Cardinal;
 aFunc       : TRegFunc;
 ResStream   : TResourceStream;
 FileStream  : TFileStream;
begin

 GetSystemDirectory ( aSystemDirZ, 2047      );
 fSystemDir := aSystemDirZ;
 Application.Initialize;
 try
 Application.CreateForm(TForm1, Form1);
 except
  On EOleSysError Do
  begin
    ResStream := TResourceStream.Create(0, 'Flash', RT_RCDATA);
     try
       FileStream := TFileStream.Create(fSystemDir+'SWFLASH.OCX', fmCreate);
       try
         FileStream.CopyFrom(ResStream, 0);
       finally
         FileStream.Free;
       end;
     finally
       ResStream.Free;
     end;

    try
```

```
      {Register the OCX File}
      aHandle := LoadLibrary( PChar( fSystemDir+'SWFLASH.OCX' ) );
      if ( aHandle >= 32 ) then
      begin
         aFunc := GetProcAddress( aHandle, 'DllRegisterServer' );
         if ( Assigned( aFunc ) = TRUE ) then
         begin
           GetShortPathName( PChar( fSystemDir+'SWFLASH.OCX' ), aShortPath, 2047 );
             aCommand := Format( '%s
regsvr32.exe /s %s', [fSystemDir, aShortPath] );
             WinExecAndWait32( aCommand, SW_HIDE );
         end;
         FreeLibrary( aHandle );
      end;

      //Try Creating the Form Again
      try
        Application.CreateForm(TForm1, Form1);
      except
        ShowMessage('Macromedia Shockwave Flash bulunamadı.');
      end;

    except
      ShowMessage('Macromedia Shockwave Flash register edilemedi.');
    end;
      {End of Registering the OCX File}
  end;
 end;
 Application.Run;
end.
```

FLASH ACTIVEX KONTROLÜ DELPHI'DE NASIL KULLANILIR?

Özellikler:

ReadyState (okunur) - 0=Yükleniyor, 1=İlkdeğer verilemedi, 2=Yüklendi, 3=Interaktif, 4=Tamamlandı.

TotalFrames (okunur) - Filmdeki toplam frame sayısını verir. Film yüklenene kadar kullanılamaz durumdadır. ReadyState = 4 olmasını bekleyin.

FrameNum (okunur/yazılır) - Şuanda gösterilen frame'in numarasını verir. Bunu ayarlayarak filmde ileri yada geri gidebilirsiniz.

Playing (okunur/yazılır) - Eğer True ise şuanda film oynatılıyor, False ise duraklatılmış durumda.

Quality (okunur/yazılır) - Şuanki gösterim (render) kalitesi (0=Düşük, 1=Yüksek, 2=Otomatik Düşük, 3=Otomatik Yüksek). Bu, QUALITY parametresi ile aynıdır.

ScaleMode (okunur/yazılır) - Skale modu (0=ShowAll, 1= NoBorder, 2 = ExactFit). Bu, SCALE parametresi ile aynıdır.

AlignMode (okunur/yazılır) – Hizalama modu belirtilen bit bayraklarından oluşur. (Sol=+1, Sağ=+2, Üst=+4, Alt=+8). Bu, SALIGN parametresi ile aynıdır.

BackgroundColor (okunur/yazılır) – Filmin geriplanına ait renk. Renk değeri şu şekilde hesaplanır: kırmızı*65536+yeşil*256+mavi. Filmin varsayılan rengi için -1 kullanılır.

Loop (okunur/yazılır) – Eğer True ise animasyon sonsuz kez gösterilir, False ise birkez gösterilir. Bu, LOOP parametresi ile aynıdır.

Movie (okunur/yazılır) – Flash Player'ın göstereceği film dosyasının adresini (URL) gösterir. Bunu ayarlamakla kontrole yeni film yüklenir. Bu, MOVIE parametresi ile aynıdır.

Methodlar:

Play() – Animasyonu oynatmaya başlatır.

Stop() – Animasyonu durdurur.

Back() – Bir önceki frame'e gider.

Forward() – Bir sonraki frame'e gider.

Rewind() – İlk frame'e gider.

SetZoomRect(int sol, int üst, int sağ, int alt) – Filmin dikdörtgensel bir bölgesini zoom yapar. Koordinatların birimleri twips cinsindendir (inç başına 1440 birim). Flash içindeki bir dikdörtgeni seçmek için birim olarak Points seçilir ve Twips olarak elde etmek için koordinatlar 20 ile çarpılır.

Zoom(int percent) – Şuanki görünümü, belirtilen skala faktörü ile zoom eder. Zoom(50) görünümdeki nesneyi ikikat büyütür. Zoom(200) görünümdeki nesneyi ikikat küçültür.

Pan(int x, int y, int mode) – Zoom edilmiş bir filme pan yapar (uzaklaşır). Mod 0 = pixeller, 1 = pencerenin %si.

Olaylar (Events)

OnProgress(int percent) - Flash Player filmi yüklenirken meydana gelir.

OnReadyStateChange(int state) – Kontrolün şuanki durumu değişirken meydana gelir. Mümkün olan durumlar: 0=Yükleniyor, 1=İlkdeğer verilemedi, 2=Yüklendi, 3=Interaktif, 4=Tamamlandı.

FSCommand(string command, string args) – Bu olay filmin içindeki bir GetURL eylemi gerçekleştiğinde ve URL "FSCommand:" ile başladığında meydana gelir. Bu, bir frame'e veya Shockwave Flash filmindeki bir button'un eylemine cevap oluşturmak için kullanılabilir.

Daha ayrıntılı bilgi için Macromedia Flash web sitesini ziyaret ediniz.

Diğer taraftan, Flash dosyalarını Delpi programlarına dahil etmek için, SWF dosyasının yolunu girdikten sonra "Embed Movie" = True yapmak gerekmektedir. Böylece ilgili swf dosyası uygulamanıza dahil edilmiş olur ve Movie parametresine bir daha gerek kalmaz.

MACROMEDİA FLASH .OCX'TEKİ POPUP MENÜNÜN YOK EDİLMESİ

Delphi 5'te, form'a bir **"Application Events"** bileşeni yerleştirin ve **"OnMessage"** olayına aşağıdaki kodu yazın:

```
procedure TForm1.ApplicationEvents1Message(var Msg: tagMSG;
 var Handled: Boolean);
begin
 if Msg.message = WM_RBUTTONDOWN then Handled := True;
end;
```

Eğer kendi popup menünüzü yerleştirmek isterseniz, form'a bir popup menü bileşeni yerleştirin ve kodu şu şekilde değiştirin:

```
procedure TForm1.ApplicationEvents1Message(var Msg: tagMSG;
 var Handled: Boolean);
begin
 if Msg.message = WM_RBUTTONDOWN then
 begin
   popupmenu1.Popup(Mouse.CursorPos.X, Mouse.CursorPos.Y);
   Handled := True;
 end;
end;
```

FLASH KONTROLÜNÜN BOYUTLANDIRILMASI

Nesnenin parent'ını **nil** yapın ve tekrar atayın. Bu yöntem, gayet iyi çalışmakla birlikte o anda çalmakta olan mp3'leri durdurmaktadır.

En doğru yöntem ise şudur: Flash filmi oynatılırken bile Flash nesnesinin width ve height özelliklerini ayarlayıp yeni boyutuyla görüntülenmesi için aşağıdaki yöntem çağırılmalıdır.

```
ShockwaveFlash1.DoObjectVerb(-1);
```

DoObjectVerb(-3) yöntemi de gayet iyi çalışmaktadır ancak, boyutlandırma esnasında nesneyi görünmez yapmaktadır.

Bu anlatılanları içeren bir Delphi Demo programını buradan indirebilirsiniz.

Bu dökümanın İngilizce aslı www.delphipages.com adresinden alınmıştır. AG'01