Atatürk Üniversitesi
Açıköğretim Fakültesi

Görsel Programlama I

ATATÜRK ÜNİVERSİTESİ
AÇIKÖĞRETİM FAKÜLTESİ

Görsel Programlama I

ISBN: 978-975-442-935-0

ATATÜRK ÜNİVERSİTESİ AÇIKÖĞRETİM FAKÜLTESİ YAYINI

ERZURUM, 2017

İÇİNDEKİLER

Editör

Doç. Dr. ERSİN KARAMAN

Dr. Öğr. Üyesi DENİZ DAL

GÖRSEL PROGRAMLAMA EDİTÖRÜ KURMA VE AYARLARINI YAPMA

Atatürk Üniversitesi
Açıköğretim Fakültesi

İÇİNDEKİLER

- Visual Studio 2012 Kurulumu
- Visual Studio.NET Program Geliştirme Ortamı
- Visual Sdudio.NET Araçları
  - Menü Çubuğu ve Araç Çubuğu
  - Form ve Kod Penceresi
  - Toolbox Penceresi
  - Solution Explorer Penceresi
  - Properties Penceresi

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Visual Studio 2012 Express Edition Programını kurabilecek,
  - Visual Studio .NET program geliştirme ortamını tanıyabilecek,
  - Visual Studio .NET'de bulunan pencerelerin özelliklerini bilecek ve bu pencerelerin nasıl kullanılabileceğini öğreneceksiniz.

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr.
Yusuf Ziya AYIK

ÜNİTE

1

GİRİŞ

Visual Studio.NET platformunun ilk sürümü 2002 yılında kullanıma sunulmuştur.

Görsel Programlama, kodlamasında görsel nesnelerin kullanılabildiği nesneye dayalı üst seviyeli programlama mantığıdır. Görsel programlamada nesnelerin kod yapıları hazır olarak kullanılabilir ve bu da program geliştirmeyi oldukça kolaylaştırır.

Günümüzde yaygın olarak kullanılan görsel programlama ortamlarına Eclipse, NetBeans, Dev-C++ ve Microsoft Visual Studio.NET örnek olarak verilebilir. Visual Studio.NET, özellikle son yıllarda program yazmayı kolaylaştırabilen tümleşik geliştirme ortamı (IDE- Integrated Development Environment) sayesinde programcılar tarafından sıkça tercih edilmektedir. Bu kitapta, görsel programlama konuları Visual Studio 2012 platformu üzerinde Visual Basic programlama dili kullanılarak anlatılacaktır.

Visual Studio.NET kullanılarak Windows, Web veya mobil tabanlı programlar geliştirilebilir. Visual Studio.NET sunmuş olduğu IDE içerisindeki birçok araç ile yazılım geliştirme sürecinin verimli şekilde ilerlemesine katkıda bulunmaktadır. Görsel programlamada Visual Studio.NET platformunun tercih edilmesini sağlayan bazı özellikler şunlardır:

- Esnek, kullanıcı tercihlerine göre özelleştirilebilir editöre sahip olması
- Tümleşik bir derleyici ve hata ayıklayıcıya sahip olması
- Kod dosyalarının hiyerarşik bir şekilde görülebilmesine imkân vermesi
- Değişik programlama dillerini desteklemesi
- Programcılar için çeşitli hazır araçlar sunması

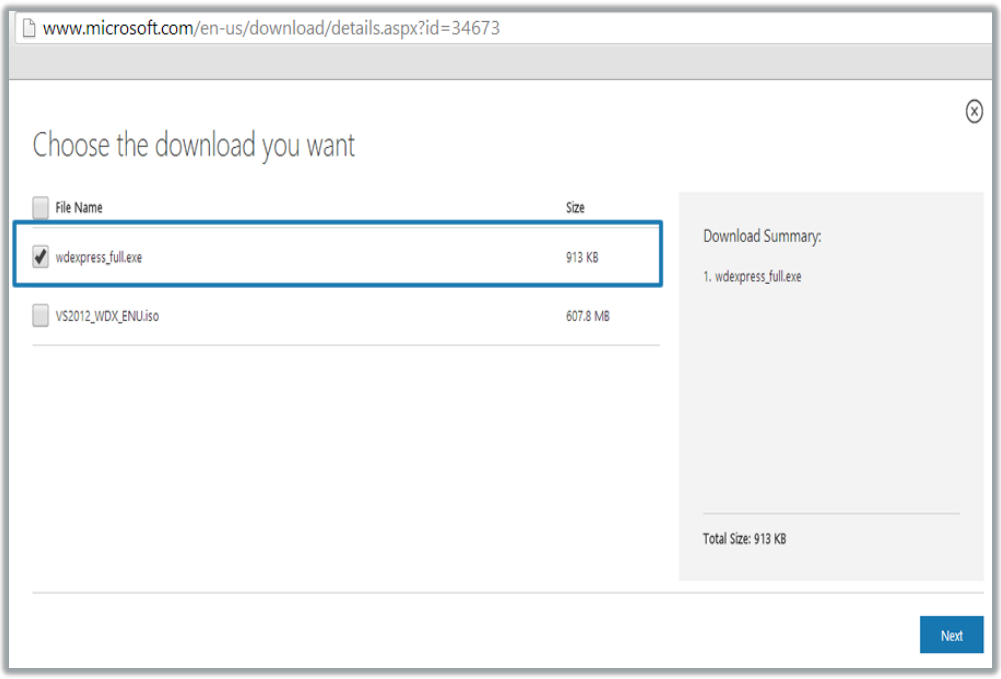
VISUAL STUDIO 2012 KURULUMU

Visual Studio 2012 yazılımının kurulumu için ilk olarak http://www.microsoft.com/en-us/download/details.aspx?id=34673 adresine gidilir. Açılan Web sayfasında Visual Studio 2012 Express Edition yazılımının internet yükleyicisi bulunmaktadır. Bu yükleyiciyi indirmek için ilk olarak yükleyici yazılımının dilini seçmek gerekir. Dil olarak “English” seçilir ve ardından hemen sağ tarafta bulunan “Download” butonuna basılır.

Resim 1.1 Visual Studio 2012 Express internet yükleyicisinin bilgisayara indirilmesi

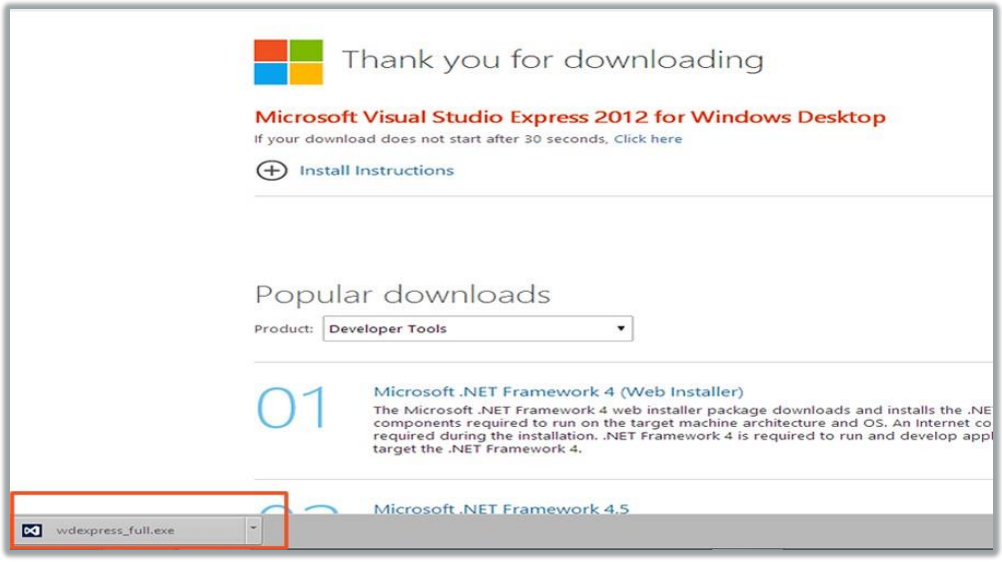
Kurulum yapılırken dil seçimi yapılmalıdır.

"Download" komutunun verilmesinin ardından ekranda yeni bir pencere açılır. Bu pencere Visual Studio 2012 Express yazılımının internet yükleyicisi tarafından veya ISO kalıbı ile indirilmesi alternatifini sunmaktadır. Kullanıcı 913 KB boyutundaki *"wd_express_full.exe"* seçeneğini işaretlemeli ve ardından sayfanın sağ alt tarafında bulunan *"next"* butonuna basmalıdır.



Resim 1.2: Visual Studio 2012 Express yükleme seçenekleri

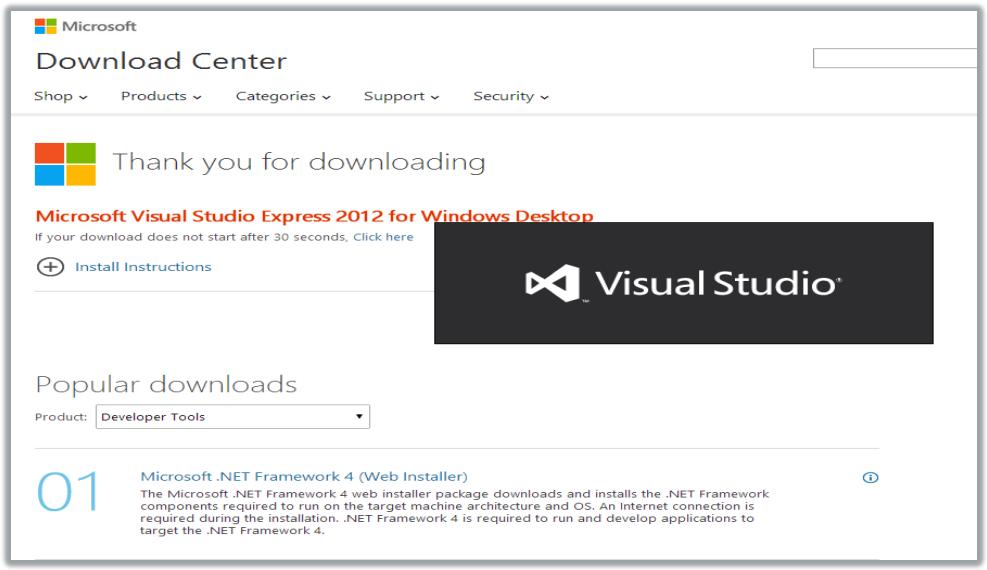
Bu işlemin ardından Visual Studio 2012 Express Edition yazılımına ait internet yükleyicisi kullanıcının bilgisayarına otomatik olarak indirilecektir. İndirme işleminin bitmesiyle birlikte internet yükleyicisi çalıştırılır.



Resim 1.3 Visual Studio 2012 Express internet yükleyici uygulama dosyası

İnternet yükleyicisinin çalıştırılması ile ekranda ilk olarak *"Visual Studio"* logosu belirir.

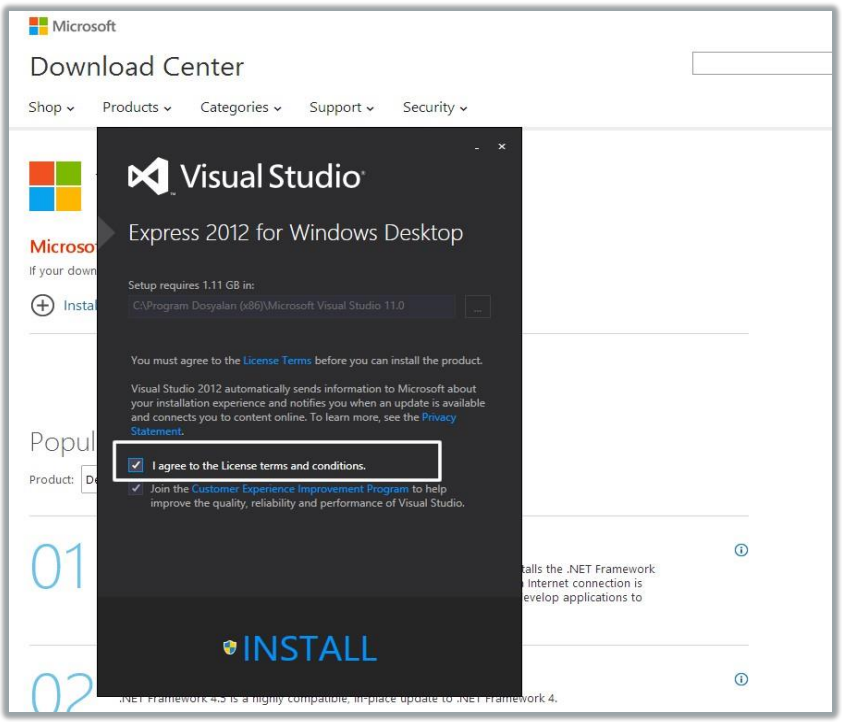
Kurulum yapılırken Lisans sözleşmesi kabul edilmelidir.



Resim 1.4: Visual Studio 2012 Express internet yükleyicisinin başlatılması

Ardından Lisans sözleşmesinin bulunduğu yeni bir pencere açılır. Bu pencerede kullanıcı lisan sözleşmesini kabul etmelidir. Lisan sözleşmesini kabul etmek için *"I agree to the Licanse terms and conditions"* seçeneğinin işaretlenmesi gerekmektedir. Bu seçenek işaretlendiğinde pencerenin hemen altında *"INSTALL"* butonu görüntülenir. Kullanıcı yüklemeye devam etmek için bu butona basmalıdır.

Visual Studio 2012 Express Edition yazılımının otomatik olarak indirilip kurulmaya başlaması için INSTALL butonuna basınız.



Resim 1.5: Visual Studio 2012 Express yazılımı Lisans Sözleşmesi penceresi

"INSTALL" butonuna basılmasının ardından Visual Studio 2012 Express Edition yazılımı, otomatik olarak indirilip kurulmaya başlar.

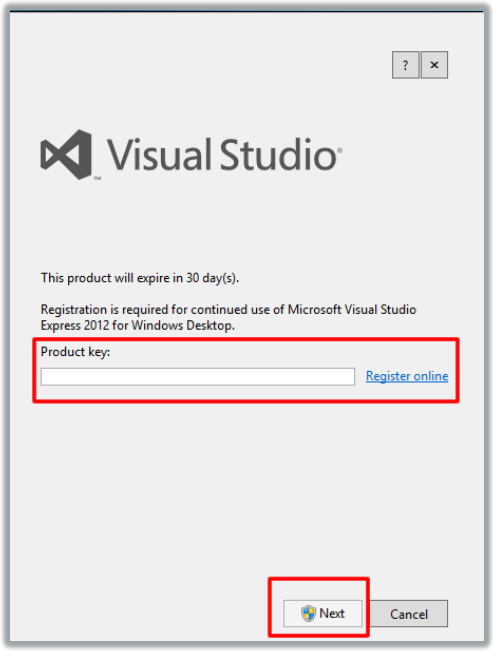
Resim 1.6: Visual Studio 2012 Express yazılımının yüklenmesi

Visual Studio 2012 Express Edition yazılımını çalıştırmak için "LAUNCH" butonuna basınız.

Resim 1.6. ekranında kurulum tamamlanıncaya kadar bekledikten sonra yükleme başarılıysa ekranda yeni bir pencere açılır. Bu pencere; kurulumun başarılı bir şekilde gerçekleştiğini kullanıcıya gösterir. Kullanıcı, Visual Studio 2012 Express Edition yazılımını çalıştırmak için "LAUNCH" butonuna basmalıdır.

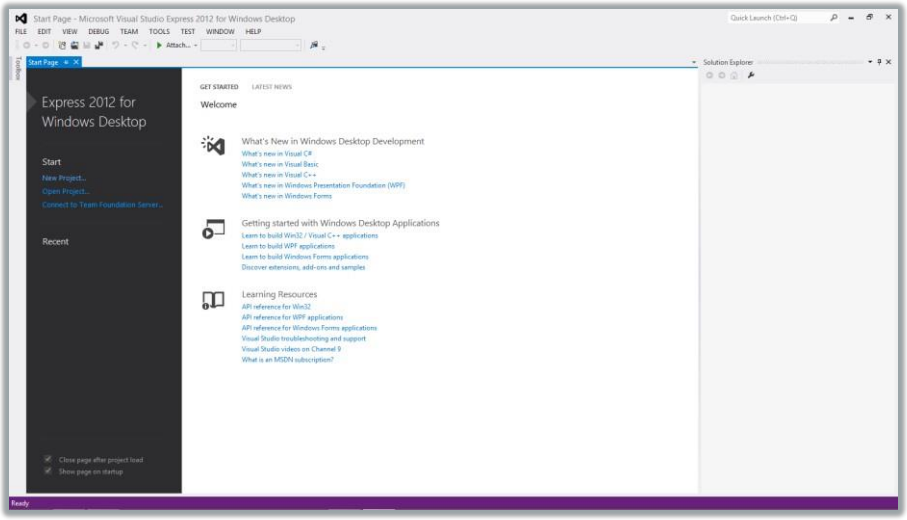
Resim 1.7: Visual Studio 2012 Express yazılımının başarılı bir şekilde yüklendiğini gösteren pencere

Kurulan yazılım 30 günlük deneme sürümüdür. "LAUNCH" butonuna basıldığında ekranda Visual Studio 2012 Express Edition yazılımına ait lisans anahtarının istendiği bir pencere açılır. Eğer kullanıcının elinde bu anahtar varsa ilgili alana yazarak "NEXT" butonuna basar.



Resim 1.8: Visual Studio 2012 Express yazılımının lisans anahtarı girişi

Eğer kullanıcının elinde lisans anahtarı yoksa "Cancel" butonuna basar ve 30 gün süresince kullanabileceği Visual Studio 2012 Express Edition yazılımı otomatik olarak başlatılır.

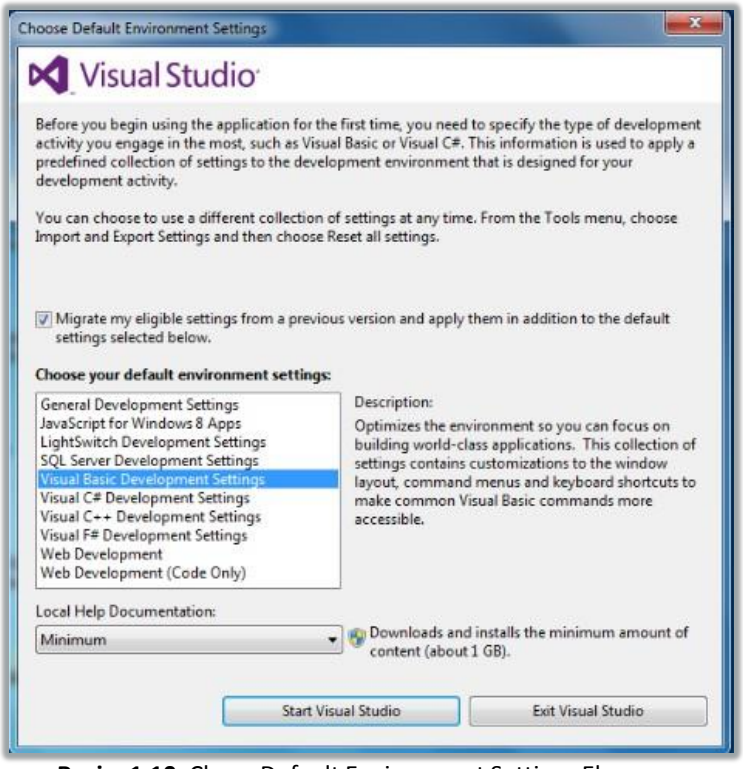


Resim 1.9: Visual Studio 2012 Express yazılımının ekran görüntüsü

VISUAL STUDIO .NET PROGRAM GELİŞTİRME ORTAMI

Visual Studio .NET ile Standart Windows uygulamaları, Windows konsol uygulamaları, Windows servisleri, Windows kontrolleri ve Windows kontrol kütüphaneleri ile Web uygulamaları geliştirebilir.

Visual Studio .NET programının bilgisayara kurulum işlemi tamamlandıktan sonra, program çalıştırıldığında Resim 1.6.'da görüntülenen "Chose Default Environment Settings" başlıklı ekran görüntülenecektir.

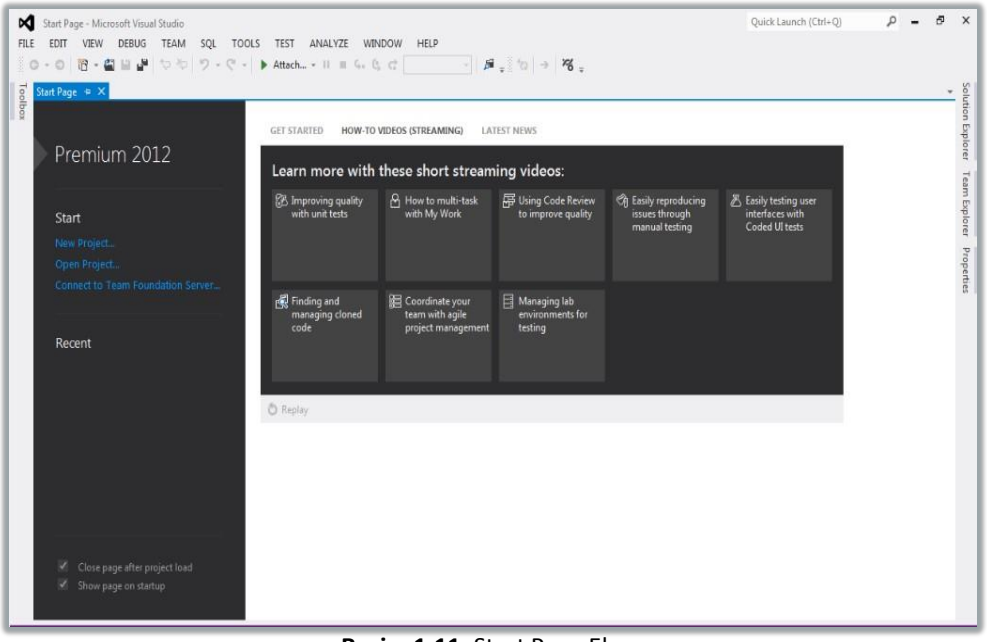


Resim 1.10: Chose Default Environment Settings Ekranı

Bu ekrandan geliştirilecek programın amacına ya da kullanım alanına uygun bir geliştirme ortamı tercih edilir. Örneğin Visuol Studio kullanılarak sadece Web uygulamaları geliştirilecekse *Web Development* ya da SQL Server'a yönelik veritabanı işlemleri ağırlıklı uygulamalar geliştirilecekse *SQL Server Development Settings* seçenekleri kullanılır. Bu kitapta konular ve örnekler Visual Basic programlama dili kullanılarak anlatılacağı için geliştirme ortamı olarak *Visual Basic Development Settings* tercihi seçilmelidir. Bu seçim sonucunda program geliştirme ortamı Visual Basic .NET programlama dilinin varsayılan ayarlarına göre düzenlenmiş olacaktır.

Bu ekranda seçilen tercih *Tools* menüsünden *Import and Export Settings* seçeneği tıklanarak gelen diyalog penceresinden *Reset all settings* komutu kullanılarak değiştirilebilir.

Visual Studio .NET çalıştırıldığında yeni bir proje oluşturabilmek veya önceden hazırlanmış projeleri açabilmek için *Start Page* ekranı görüntülenir.

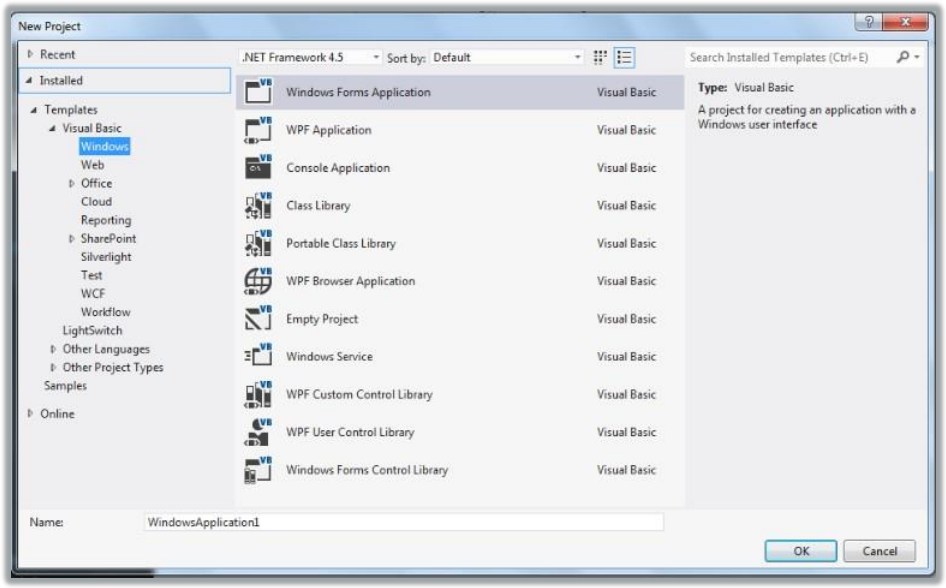


Resim 1.11: Start Page Ekranı

Önceden oluşturulmuş projelere Open Project seçilerek erişilebilir.

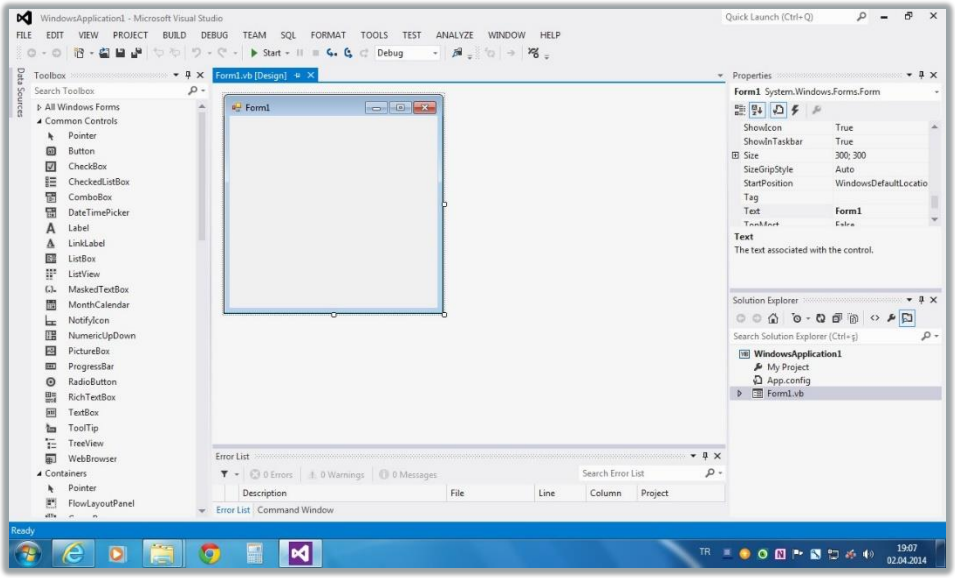
Bu sayfada görüntülenen *Recent* menüsünde varsa önceden oluşturulmuş projeler listelenir. Mevcut projelerden herhangi birisini açmak için *Open Project* seçilmelidir. Open Project seçildiğinde Belgelerim klasöründe bulunan Visual Studio 2012 klasörü altındaki Projects klasöründe önceden hazırlanmış projelere erişmek mümkün olacaktır. Geliştirilen her projenin, Projects klasörü altında projenin tüm dosyalarını içine alacak şekilde hazırlanarak ayrı bir klasör olarak oluşturulduğu görülecektir.

Yeni bir proje oluşturulmak istenildiğinde yine bu sayfada görüntülenmekte olan New Project seçilmelidir. Bu durumda Resim 1.8.'deki New Project ekranı görüntülenecektir.



Resim 1.12: New Project Ekranı

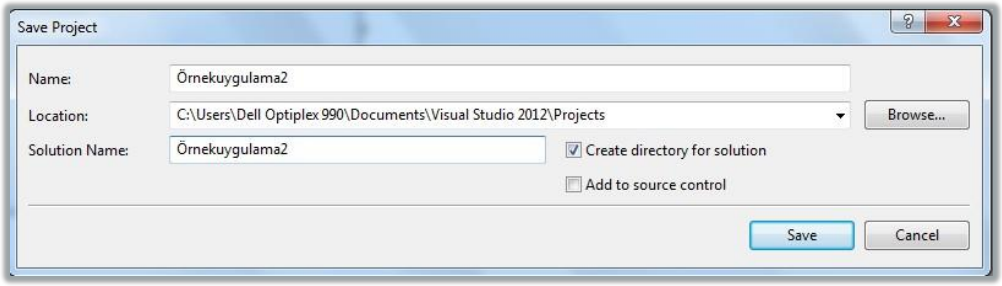
New Project penceresinden programlama dili için "Visual Basic", proje tipi için de "Windows Forms Application" seçilir. Name kısmında ise proje adı belirlenir. Bu işlemler yapıldıktan sonra Visual Studio.NET, Visual Basic programlama diline ait Toolbox, Properties, Solution Explorer ve Form nesnesini içeren bir geliştirme ortamı yapılandırır (Resim 1.13).



Resim 1.13: Proje örneği Ekranı

Yeni açılan projede kendiliğinden adı Form1 olan yeni ve boş bir form oluşturulur.

Oluşturulan projeyi kaydetmek için File menüsünden Save All seçeneği kullanılır. Aynı işlem Ctrl+Shift+S tuş kombinasyonu kullanılarak da yapılabilir.



Resim 1.14: Save Project Ekranı

VISUAL STUDIO.NET ARAÇLARI

Visual Studio.NET'in görünmeyen pencerelerine View menüsü altında Other Windows seçeneğini tıklayarak erişebilirsiniz.

Visual Studio.NET platformu her türlü yazılım geliştirme ihtiyaçlarına yönelik hazır bir altyapı sunarak uygulama geliştiricilere Windows, Web ve Mobil platformlara yönelik uygulamaları, çok daha hızlı ve kolay bir şekilde geliştirebilmelerine olanak tanımaktadır.

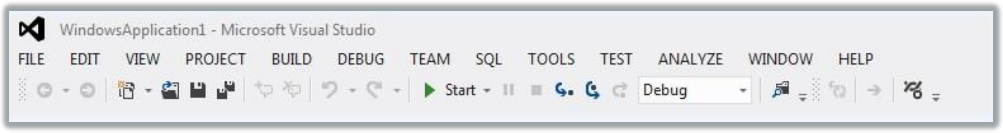
Bu başlık altında Micrososft Visual Studio.Net platformunun arayüzünü tanımak ve etkili bir şekilde kullanmak amaçlanmıştır.

Visual Studio ortamını oluşturan ve kullanımını kolaylaştıran başlıca bileşenler şunlardır:

Menü Çubuğu: Birçok yazılım geliştirme ortamında olduğu gibi Visual Studio da benzer öğeler üzerinde işlevleri olan komutları menüler halinde gruplar. Menü çubuğu sabittir ve özelleştirilemez.

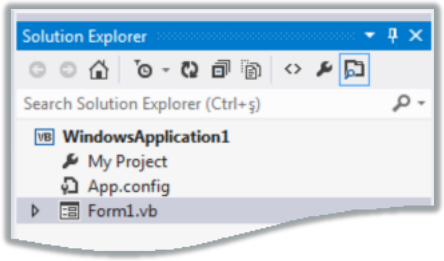
Visual Studio kullanıcıya kendi araç çubuklarını oluşturma imkânı sağlar.

Araç Çubuğu: Visual Studio menü komutları için görsel kısayolları araç çubuğunda yer alan düğmeler ile sunar. Benzer işlevler için kullanılan komutlar bir araç çubuğunda grup halinde bulunur. Araç çubukları varsayılan olarak menülerin altında bulunur. Araç çubukları taşınarak yerleri değiştirilebilir veya kayar duruma getirilebilir. Ayrıca istenen çubuklar sabitleştirilebilir. Araç çubukları listesini görmek için View menüsünden Toolbars alt menüsü seçilir.



Resim 1.15: Menü ve Araç Çubukları

Solution Explorer: Visual Studio çalışma ortamında projeler bir Solution altında açılır. Bir Solution içerisine farklı dilde ve tipte projeler dâhil edilebilir. Visual Studio ile bir proje açıldığında, Solution Explorer paneli ile Solution içinde bulunan tüm projeler ve ilgili dosyalar görüntülenir. Solution Explorer paneli CTRL+ALT+L tuş kombinasyonu kullanılarak görüntülenebilir.

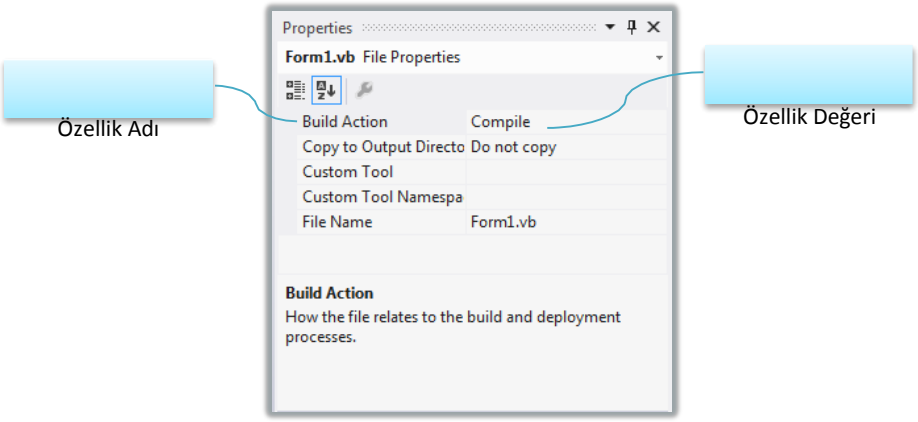


Resim 1.16: Solution Explorer

Properties penceresi seçili olan nesnenin özelliklerini görüntüler

ve özelliklerin yeniden düzenlenmesine fırsat verir.

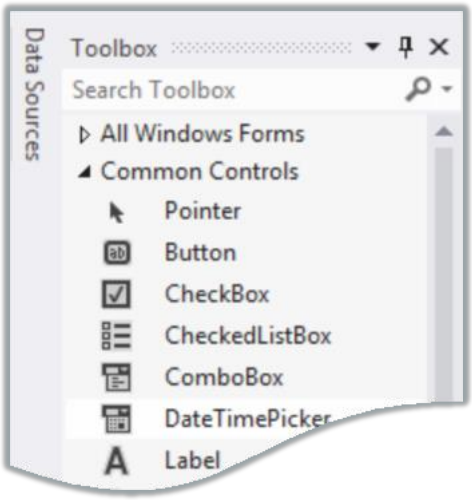
Properties Paneli: Form üzerinde seçilen bir nesneye ait özellikleri görüntülemek için bu panel kullanılır. Panelde görüntülenen özellikler işlevlerine göre gruplandırılmıştır. Ancak istenildiği zaman alfabetik olarak sıralanabilir. Panel, özellik adı ve bu özelliğin değeri görünümündedir.



Resim 1.17: Properties Penceresi

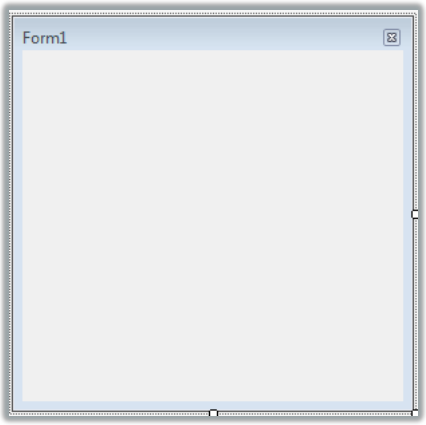
ToolBox'ın içeriği ve görünümü kullanıcı tercihlerine göre özelleştirilebilir.

ToolBox (Araç Kutusu) Paneli: Proje içerisine eklenebilecek tüm hazır nesneler bu panel içerisinde listelenir. Bu paneldeki nesneler işlevlerine göre sekmeler hâlinde gruplandırılmıştır. Form içerisinde kullanılmak istenen nesneler ToolBox panelinden seçilerek form üzerine sürüklenip bırakılır. ToolBox panelini görüntülemek için CTRL+ALT+X tuş kombinasyonu kullanılır(Resim 1.18.)



Resim 1.18: Toolbox Paneli

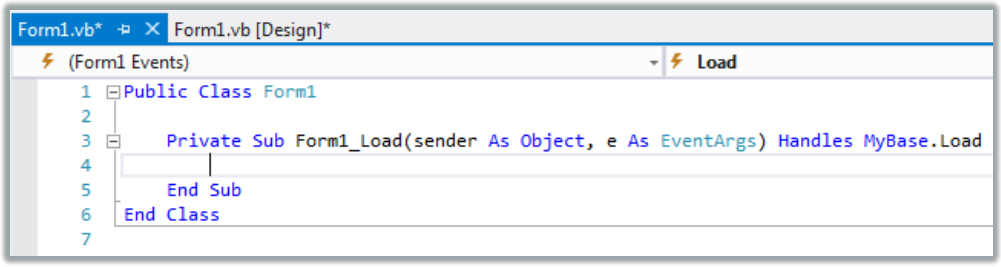
Form Nesnesi: Geliştirilecek uygulamanın ara yüzünü oluşturmak için kullanılır. Projede kullanılan tüm nesneler form üzerinde yer alır.



Resim 1.19: Form Designer

Form üzerindeki herhangi bir noktaya veya bir nesneye çift tıklanarak kod penceresine erişilebilir.

Kod Penceresi: Form üzerine eklenen nesneler ve forma ait kodlar bu pencerede yer alır.



Resim 1.20: Kod Penceresi

Özet

- Visual Studio .NET son derece gelişmiş, kolay program yazılabilmesine imkân veren bir program geliştirme ortamıdır. Visual Studio .NET orijinal anlamda IDE(Integrated Development Environment) özelliğine, yani bütünleşik yazılım geliştirme özelliğine sahiptir. Bu özellik birkaç programlama dilinin aynı ortamda kullanılabilmesine imkân verilmesi anlamına gelmektedir.
- İlk önce Visual Studio 2012 yazılımının bilgisayarınıza kurulumu gerekmektedir. Bunun için öncelikle Visual Studio 2012 yazılımı http://www.microsoft.com/en-us/download/details.aspx?id=34673 adresinden indirilerek kurulmalıdır.
- Visual Studio .NET geliştirme ortamının en temel araçları ve pencerelerinden birkaçı Menü çubuğu, Windows Forms Designer ve Kod penceresi, Toolbox penceresi, Solution Explorer penceresi ve Properties penceresidir.

DEĞERLENDİRME SORULARI

1. Visual Studio .NET'i başlatma yollarından hangisi doğrudur?
   a) File menüsünden New Project seçilir.
   b) Start düğmesinden Visual Studio .NET klasöründen Visual Studio .NET simgesi tıklanır.
   c) Start düğmesinden Visual Studio .NET klasöründen Visual Studio .EXE simgesi tıklanır.
   d) File menüsünden Open Project seçilir.
   e) File menüsünden New File seçilir.

2. Var olan bir proje nasıl açılır?
   a) File menüsünden New Project seçilir.
   b) Start düğmesinden Visual Studio .NET klasöründen Visual Studio .NET simgesi tıklanır.
   c) Start düğmesinden Visual Studio .NET klasöründen Visual Studio .EXE simgesi tıklanır.
   d) File menüsünden Open Project seçilir.
   e) File menüsünden New File seçilir.

3. Visual Studio .NET başlatılmak istenildiğinde aşağıda yöntemlerden hangisi kullanılmalıdır?
   a) File menüsünden New Project seçilir.
   b) File menüsünden New File seçilir.
   c) Başlat menüsünden Visual Studio .NET simgesi tıklanır.
   d) File menüsünden Recent Files seçilir.
   e) File menüsünden Open Project seçilir.

4. Bir nesnenin özelliği Visual Studio'nun hangi aracı kullanılarak değiştirilir?
   a) Properties Penceresi
   b) Araç Çubuğu
   c) Solution Explorer
   d) ToolBox
   e) Form Designer

5. Bir program nasıl çalıştırılır?
   a) F6 tuşuna basılır.
   b) Start düğmesine tıklanır.
   c) Start Page'e basılır.
   d) File menüsünden Open Project seçilir.
   e) File menüsünden New File seçilir.

6. Visual Studio .NET'ten nasıl çıkılır?
   a) File menüsünden Save All komutu tıklanır.
   b) Menü çubuğundan Project menüsü tıklanır.
   c) File menüsünden Exit komutu tıklanır.
   d) Debug menüsünden Step Over komutu tıklanır.
   e) Edit menüsünden Quick Find komutu tıklanır.

7. Visual Studio .NET'te aşağıdaki programlama dillerinden hangisi veya hangileri kullanılabilir?
   a) Visual Basic .NET, Visual C# .NET
   b) Visual Basic .NET, Visual C++ .NET
   c) Visual C++ .NET, Visual C# .NET
   d) Visual Basic .NET, Visual C++ .NET, Visual C# .NET
   e) Visual Basic .NET

8. Visual Basic.NET'de oluşturulan program dosyalarının listelendiği birim hangisidir?
   a) Properties Penceresi
   b) Araç Çubuğu
   c) Solution Explorer
   d) ToolBox
   e) Form Designer

9. Menü çubuğunun altında komut çalıştırmaya ve Visual Studio geliştirme ortamını denetlemeye hizmet eden ve düğmeler koleksiyonundan oluşan araç hangisidir?
   a) Dynamic Help
   b) Menü çubuğu
   c) Görev çubuğu
   d) Properties penceresi,
   e) Standart araç çubuğu

10. Visual Studio.NET editöründe görüntülenmemiş olan pencereler hangi menü kullanılarak görüntülenirler?
   a) View
   b) File
   c) Edit
   d) Project
   e) Build

Cevap Anahtarı

1.b,2.d,3.c,4.a,5.b,6.c,7.d,8.c,9.e

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Ayık Y. Ziya, (2009), Algoritma ve Programlama Metodolojisi, MuratHan, Trabzon

Ayık Y. Ziya, (2011), Atatürk Üniversitesi, Uzaktan Eğitim Merkezi, Görsel Programlama I Ders Notları

Atatürk Üniversitesi Baum Web Sitesi

http://www.microsoft.com Web Sitesi

Yanık Memik(2010), Visual Basic 10, Seçkin Yayınevi, Ankara

FORMLAR VE ÖZELLİKLERİ

İÇİNDEKİLER

- Windows Formlar
  - SDI Formlar
  - MDI Formlar
- Form Özellikleri
  - Text ve Controlbox Özelliği
  - Enabled ve Font Özelliği
  - Visible ve Startposition Özelliği
  - Maximizebox, Minimizebox ve Windowstate Özelliği
  - Opacity ve Showintaskbar Özelliği
  - Acceptbutton ve Cancelbutton Özelliği
  - Backcolor ve Backgroundimage Özelliği
  - Cursor Özelliği

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - SDI form ve MDI form arasındaki farklılıkları ayırt edebilecek,
  - Projenize yeni formlar ekleyebilecek,
  - Formlarınızın özelliklerini istediğiniz ölçüde değiştirebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA

Okt. Daha ORHAN

ÜNİTE

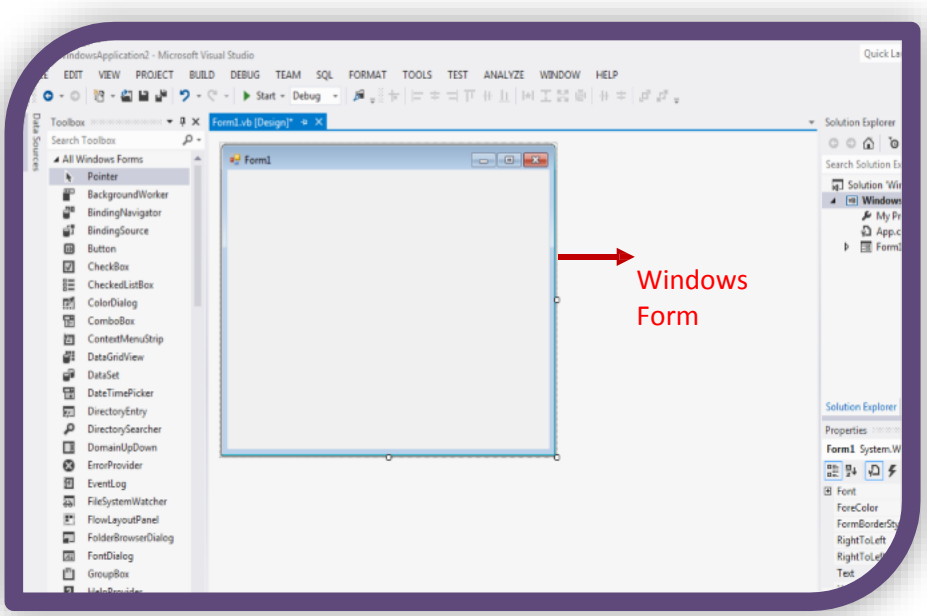
2

GİRİŞ

Formlar üzerine nesneler ekleyebileceğimiz ve kod parçacıkları ile projemizin çalışma esnasında gerçekleştirebileceğimiz olayları belirlememize fayda sağlayan nesnelerdir.

Bu ünitede Visual Studio.Net platformu üzerinde Visual Basic Programlama dilini seçtikten sonra sıklıkla kullanacağımız form yapısı üzerinde durulmaktadır. Formlar üzerine nesneler ekleyebileceğimiz ve kod parçacıkları ile projemizin çalışma esnasında gerçekleştirebileceğimiz olayları belirlememize fayda sağlayan yapılardır.

Visual Studio.NET ile seçtiğimiz programlama diline göre hazırlayacağımız projeyi başlattığımızda karşımıza Visual Studio pencerisinin ortasında görüntülenen Form [Design] gelmektedir.



Şekil 2.1 Proje başlatıldığında Visual Studio penceresine ait görünüm

Visual Basic projeleri genel itibariyle formlar, bu formlar üzerine toolbox'tan dahil edilen kontrol nesneleri ve nesnelere ait olay prosedürlerinden oluşmaktadır. Bu ünitede formların projeye nasıl eklendiği, Windows formların çeşitleri ve formların özelliklerine değinilecektir.

WİNDOWS FORMLAR

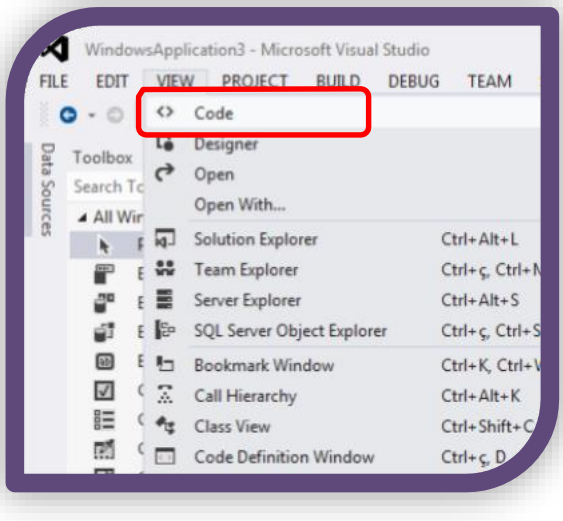
Visual Studio 2012 programını başlattıktan sonra karşımıza gelen ekranda sırasıyla File -> New -> Project menü yolunu veya ctrl + shift + N kısayol tuşunu kullanarak açılan New Project penceresinde templates sekmesinden visual basic programlama dilini ve Windows Forms Application seçeneğini işaretleyip ok butonuna bastıktan sonra projemizi açabiliriz.

Windows formlar iki ana başlık altında incelenmektedir. Bunlar SDI (Single Document Interface) formlar ve MDI (Multiple Document Interface) formlardır. Yani tek formlu arayüzler ve çok formlu arayüzlerdir. Bunlar Ana ve Çocuk Form olarak da adlandırılmaktadırlar. Detayları anlatılırken bu terimlere yer verilecektir ve neden bu şekilde adlandırıldıkları daha iyi anlaşılacaktır.

SDI Formlar

Başka bir forma ihtiyaç duymaksızın tek başına çalışabilen formlara SDI formlar denmektir. Yeni başlattığımız bir projede ilk olarak ekrana gelen formlar SDI formlardır. Şekil-1'de de proje başlatıldığında gelen form SDI formdur.

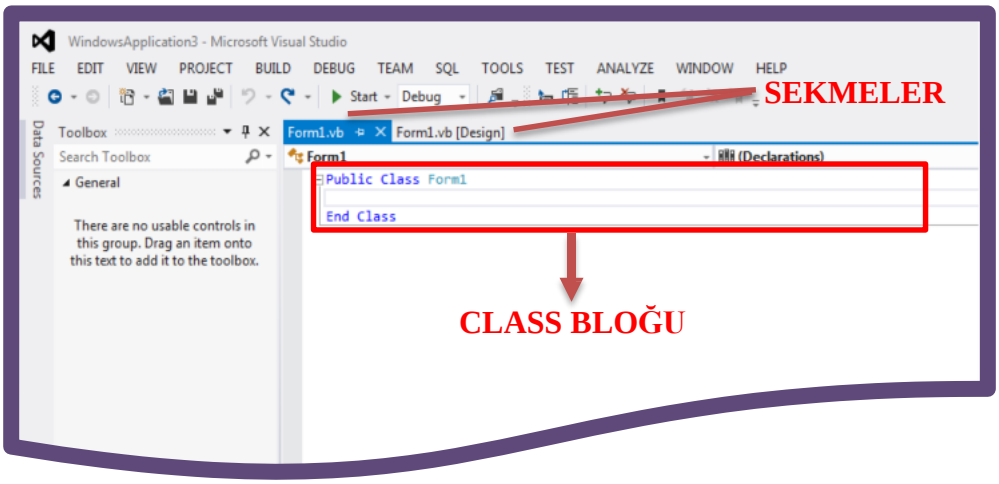
Formlar üzerinde çalışırken sadece dizayn kısmında işlem yapmak, istediğimiz uygulamaları geliştirmek için yetersiz kalabilir. Projemizde açık olan formlardan o anda aktif olan formun kod kısmını görüntülemek için view menüsünden code seçeneğini seçebiliriz. (Şekil-2)



Şekil 2.2 Kod editörüne geçmek için izleyeceğimiz menü yolu

SDI formlar tek arayüze sahip formlardır.

Yukarda bahsettiğimiz menü yolunu takip ettikten sonra visual basic projemizdeki her bir form için ayrı ayrı hazırlanan sınıf bloğunun bulunduğu alanı göreceğiz.



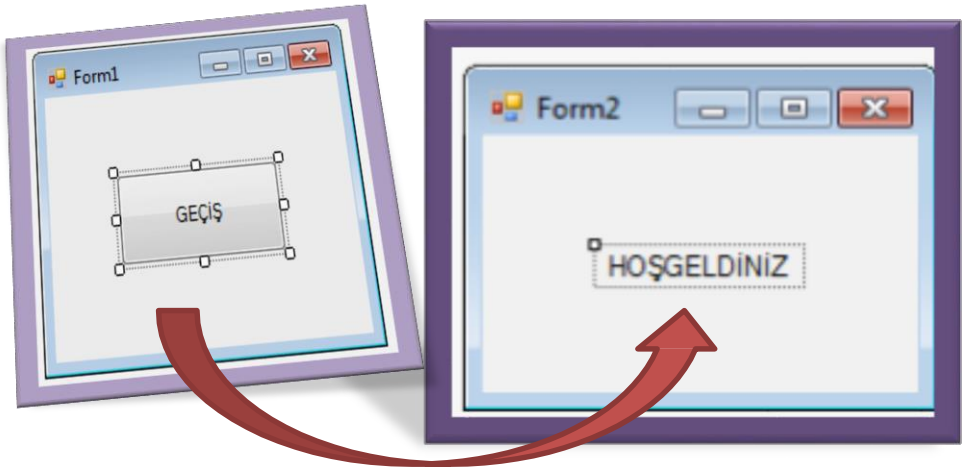
Şekil 2.3 Form1'in kod editörü

Hazırlayacağımız projelerde birden fazla SDI form kullanabiliriz.

Şekil 1.3. de gördüğünüz üzere dizayn ve kod kısımlarının görüntülenmesi sekmeler hâlinde olmaktadır. Bu sekmelere tıklayarak rahat bir şekilde dizayn veya kod alanına geçiş işleminizi gerçekleştirebilirsiniz. SDI formları daha iyi kavrayabilmek için aşağıdaki örneği inceleyelim.

Örnek

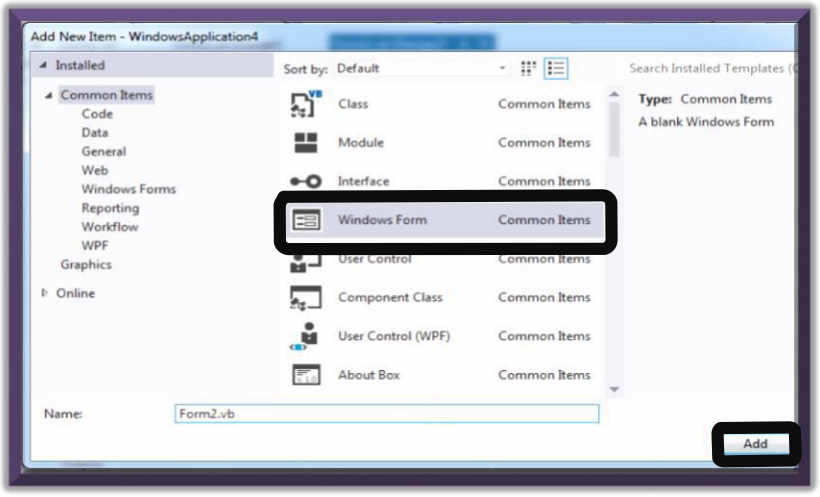
•Projemizde var olan iki form arasında nasıl geçiş yapılacağının, kod ve dizayn göstermenin örneği aşağıdaki gibidir.



Form1 üzerine toolboxtan yerleştirmiş olduğumuz butona çift tıklayarak karşımıza gelen kod alanına aşağıdaki kodu yazarak çalıştırdıktan sonra; geçiş olarak text özelliğini değiştirmiş olduğumuz butona tıklayarak eklediğimiz form2'nin görüntülendiğini göreceğiz.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
        Form2.Show()
    End Sub
End Class
```

Sadece bu kod yazılacak

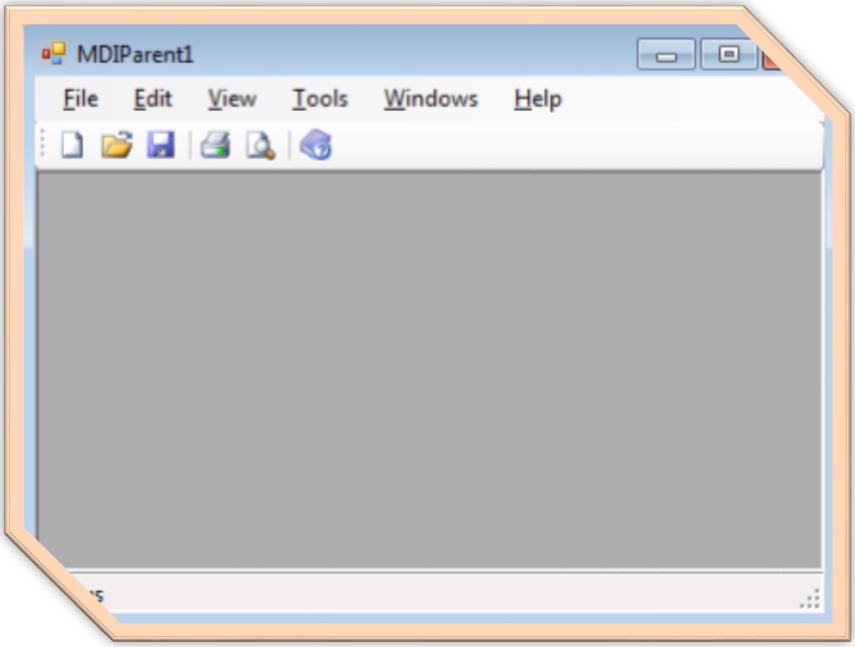


Şekil 2.4 ctrl + shift + A kısayol tuşunu kullanarak karşımıza gelen pencere ile projemize yeni form ekleme

MDI Formlar

MDI formlar bir ana pencere altında bu ana pencereye bağlı olarak bir veya birden fazla çocuk pencere çalıştırabilen yapılardır.

Microsoft firması tarafından İngilizce Multiple Document Interface kelimelerinin baş harflerinden oluşturulmuş bir kavramdır. Türkçe karşılığı çoklu doküman arayüzü anlamına denk gelmektedir. Yani MDI formlar bir ana pencere altında bu ana pencereye bağlı olarak bir veya birden fazla çocuk pencere çalıştırabilen yapılardır.



Şekil 2.5 Bir MDI forma ait görünüm

Şekil 1.5.'te göründüğü gibi projemize bir MDI form eklemek için sırasıyla şu adımları takip edebiliriz:

- Visual Studio ekranının sağ tarafında bulunan Solution Explorer penceresi üstündeyken projenin adına fare ile sağ klik,
- Açılan menüde "Add" seçeneğini işaretleriz,
- Açılan menüde "New Item" seçeneğini işaretleriz,
- Karşımıza gelen pencereden MDI Parent Form'u seçer ve Add butonuna tıklarız.

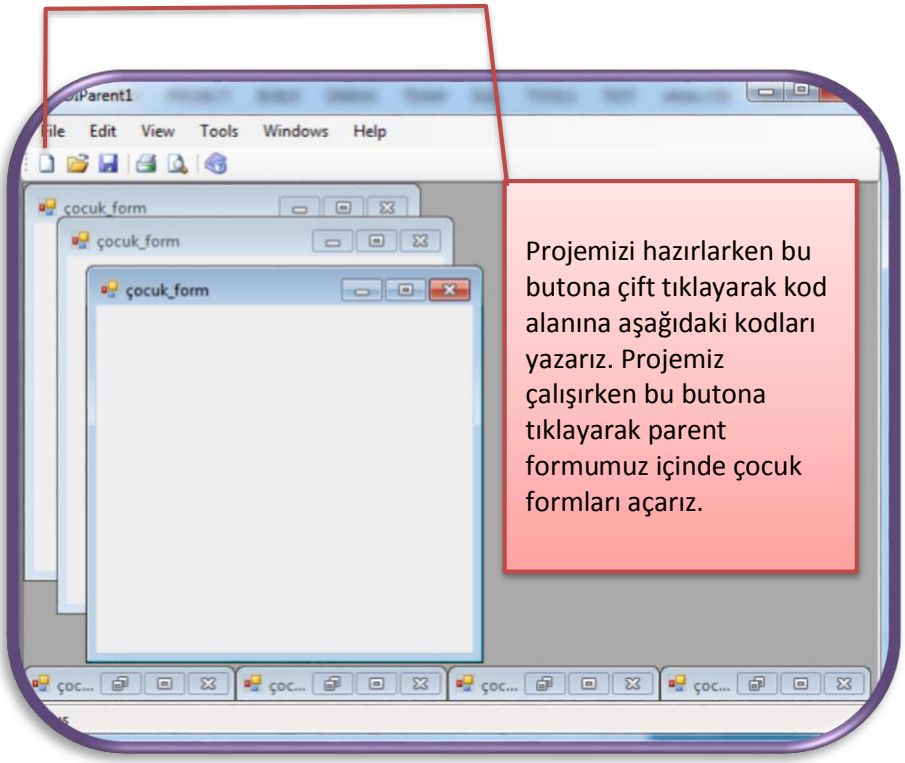
Ana pencere olan MDI forma, parent (ebeveyn) form denir. Bu ana form içinde açılan formlara ise child (çocuk) form denilmektedir. Şayet projemizde parent form mevcut ve bununla beraber child formumuzda var ama birbirinden bağımsız çalışıyorsa kendi içinde olduğunu tanıtmamışızdır. Bu hususa dikkat etmenin faydalı olacağını düşünüyorum. Birazdan inceleyeceğimiz örnekte bunu daha iyi kavrayacaksınız.

Projemizi çalıştırdığımız zaman karşımıza SDI form gelmiş ve bu formu MDI form olarak ayarlamak istiyorsak: Form seçildikten sonra sağ tarafta bulunan properties panelindeki IsMdiContainer özelliğini True olarak değiştirmemiz yeterli olacaktır.

Örnek

•Şekil 1.5'teki MdiParent form içinde bulunan yeni butonuna tıklayarak child formların nasıl görüntülendiğinin örneği aşağıdaki gibidir.

Child formlar simge durumuna getirildiklerinde görev çubuğuna değil Parent form içine inerler.




```
Private Sub ShowNewForm(ByVal sender As Object, ByVal e As
EventArgs) Handles NewToolStripMenuItem.Click,
NewToolStripButton.Click, NewWindowToolStripMenuItem.Click
    Dim Cocuk_Form As New Cocuk_Form ()
```

Daha önce eklemiş olduğumuz formu tanımlıyoruz.

```
    Cocuk_Form.MdiParent = Me
```

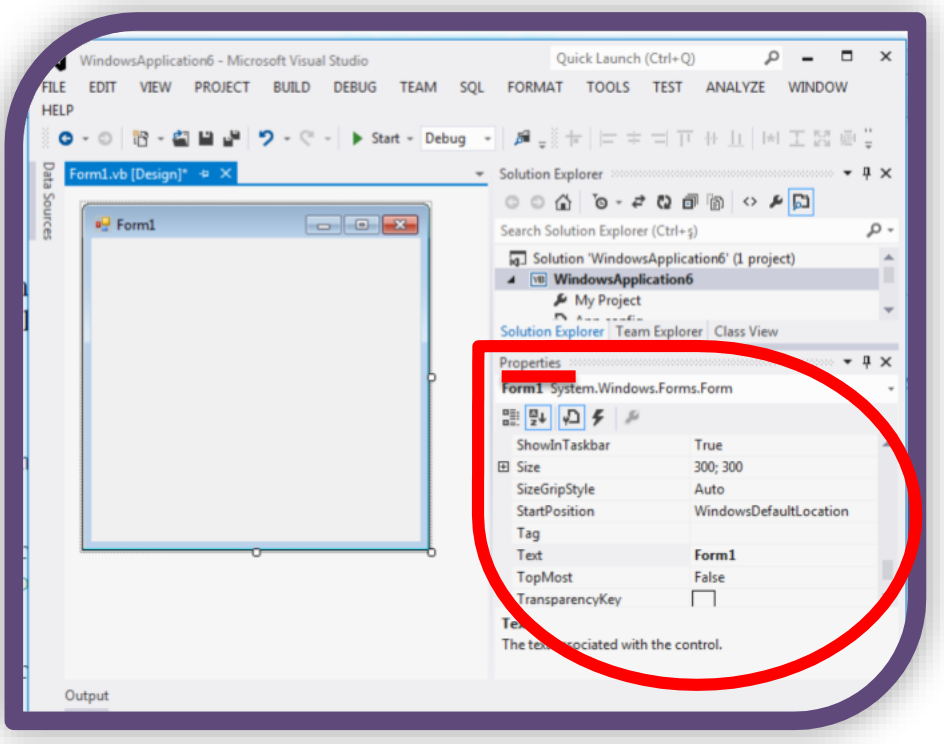
Çocuk formun parent forma bağımlı olarak çalışmasını sağlıyoruz.

```
    Cocuk_Form.Show()
End Sub
```

Örnekteki ekran görüntüsüne dikkat edecek olursak child formlar basamak hâlinde yerleşmişlerdir. Bu yerleşim hâline cascade adı verilmektedir. MDILayout metodu ile bu yerleşim şekli değiştirilebilir.

FORM ÖZELLİKLERİ

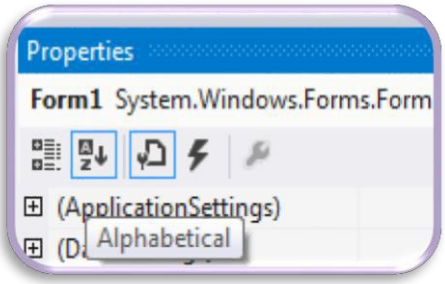
Form özelliklerinden bahsetmeden önce bu özelliklere nereden ulaşabileceğimizi anlatmanın faydalı olacağını düşünüyorum. Visual Studio ortamında ekranın sağ tarafına yerleşen properties paneli üzerinden, çalışma esnasında aktif nesneye ait olan özellikleri ve bu özelliklerin o an itibariyle değerlerinin görüntülendiği bir alan bulunmaktadır. Bu alan üzerinden formumuza ait özellikleri değiştirme imkânına sahibiz.



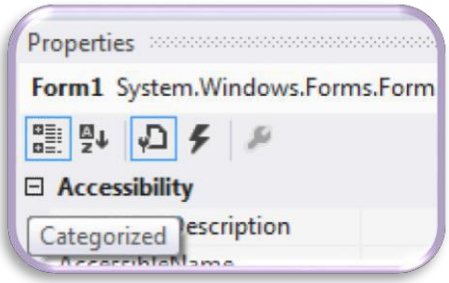
Şekil 2.6 Visual Studio ortamında Properties Paneli

Properties paneli, alfabetik veya gruplar halinde özellikleri listeleme imkanı sunmaktadır.

Properties paneli iki farklı görünümde özellikleri listeleyebilir. Bunlardan biri *alphabetical* seçeneğidir(Şekil 7a). Bu seçenek ile aktif olan kontrolün özellikleri panel içinde alfabetik olarak sıralanır. Diğer bir listeleme şekli ise *categorized* seçeneğidir(Şekil 7b). Bu seçenek ise özellikleri bir takım ortak gruplar hâlinde (Appearance, Behavior, Data vb.) birleştirerek listelemek için kullanılır.



Şekil 2.7a Alfabetik listeleme



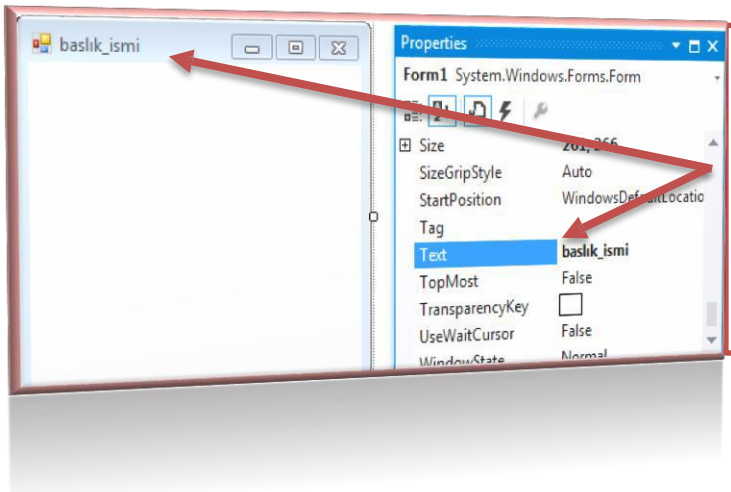
Şekil 2.7b Gruplandırılmış listeleme

Programımızı çalıştırdığımız zaman properties paneli ekranda görünmüyor ise F4 kısayol tuşunu kullanarak ekrana getirebiliriz. Bu ünitede properties panelindeki bütün özellikler yerine *formlarda* sıklıkla kullanılan özelliklere yer verilecektir.

Text Özelliği

Formun başlık çubuğunda görüntülenen isminin ayarlandığı özellik seçeneğidir. Burada dikkat etmemiz gereken nokta bu isim sadece projemizin tasarım kısmındaki isimdir. Text özelliği haricinde bir değişiklik yapılmadıysa kod alanında çalışırken formumuzun ismi programın varsayılan olarak atadığı (Form1, Form2 vb.) isimdir. Şayet kod alanında da sizin yazdığınız ismin olmasını istiyorsanız *"name"* özelliğini değiştirmelisiniz.

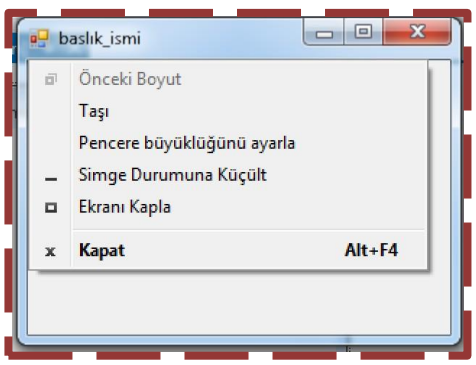
Formun text özelliği ile name özelliği birbirinden farklı değerlerdir.



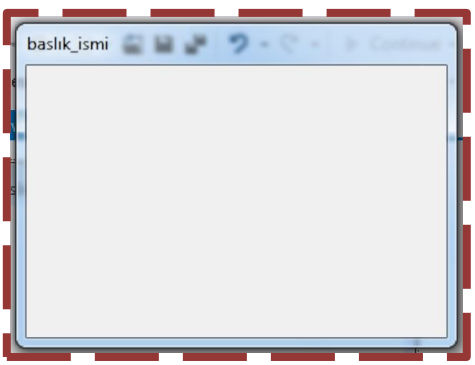
Şekil 1.8: Text özelliği ile formun başlığını değiştirme

ControlBox Özelliği

Bu özellik program çalışırken anlaşılabilecek bir yapıdadır. Mantıksal olarak değer alabilen iki seçenek söz konusudur. Bunlardan biri true (doğru veya 1) diğeri ise false (yanlış veya 0) değerleridir. True değeri geçerli olduğu zaman programı çalıştırıp formun başlık çubuğunda sol üst köşeye tıklayarak karşımıza bir kontrol menüsünün geleceğini görürüz (Şekil 1.9a). Fakat bu özellik False yapılmışsa hem bu kontrol menüsünün hem de sağ üst köşedeki (simge durumuna, tam ekran, kapat) butonların olmadığını fark ederiz (Şekil 1.9b).



Şekil 2.8a ControlBox = True



Şekil 2.8b ControlBox = False

Enabled Özelliği

Forma ait enabled özelliği form üzerindeki diğer kontrolleri de etkiler.

Controlbox özelliğinde olduğu gibi True ve False şeklinde iki farklı değer alabilen mantıksal bir büyüklüktür. Bu özelliğin true olması durumunda form aktif konumdadır ve formun üzerindeki nesneler ile işlem yapılması mümkündür. Formun enabled özelliğini false yapacak olursak bu defa form pasif konuma geçecektir ve forma yerleştirdiğimiz nesneler de aynı şekilde pasif konumda olacaktır.

Örnek

- Formun enabled özelliğini true ve false olarak değiştirdiğimizde; toolboxtan forma yerleştirdiğimiz textbox ve iki adet checkbox'ın programı çalıştırdığımız andaki ekran görüntüsü:

Enabled = True	Enabled = False
Nesneleri kontrol edebiliriz.	Nesneleri kontrol edemeyiz.
Textbox içine yazı yazabiliriz.	Textbox içine veri girişi yapamayız.
Checkbox'ları işaretleyebiliriz.	Onay kutularını işaretleyemeyiz.

Formun bu ve benzeri özelliklerini properties panelinden değiştirebileceğimiz gibi kod editörüne geçiş yaparak oraya yazacağımız kod satırlarıylada ayarlama imkânına sahibiz.

Formu pasif konuma geçirmek için:

```
Form_ismi.Enabled = False
```

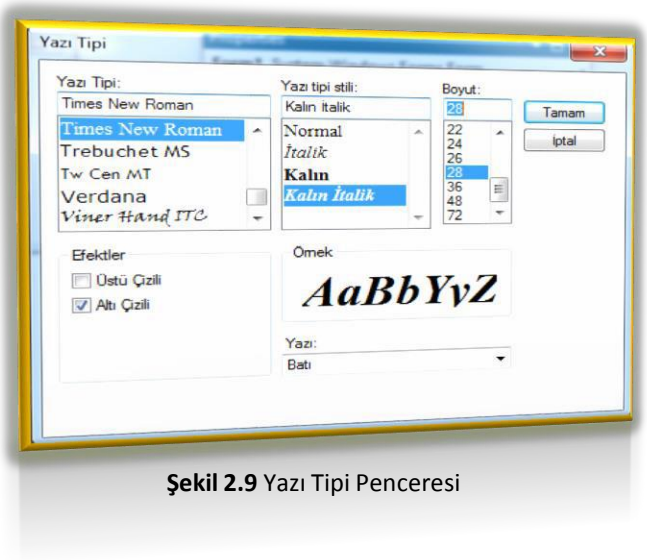
Formu aktif konuma geçirmek içinse:

```
Form_ismi.Enabled = True
```

Kod satırlarını yazmamız yeterli olacaktır.

Font Özelliği

Bu özellik ile form üzerindeki nesnelere ait text özellikleri, form üzerindeki yazı stilleri, fontu ve boyutu istenilen ölçüde ayarlanır ve o parametreler doğrultusunda görüntülenmesi sağlanır. Properties panelindeki font özelliğine gelindikten sonra, özelliğe ait sağ tarafında bulunan … butonu tıklanır ve şekil 1.10'daki gibi karşımıza gelen pencereden istenilen ayarlar yapılarak tamam butonuna tıklanır.



Şekil 2.9 Yazı Tipi Penceresi

Visible Özelliği

Aktif formun visible özelliği değiştirilemez.

Projemizi çalıştırdığımız esnada formun ekran üzerinde görünür olup olmama durumunu belirleyen bir değerdir. Mantıksal olarak iki farklı değer alabilir. Bu değerler daha önce aşina olduğumuz true ve false değeridir.

Bu özellik formlar için Visual Studio 2012'de properties panelinde bulunmamaktadır. Formun görünür olup olmama durumunu kod editöründen aşağıdaki kod satırını yazarak ayarlamamız gerekiyor.

Formu görünür konuma geçirmek için:
Form_ismi.Enabled = False

Formu görünmez konuma geçirmek içinse:
Form_ismi.Enabled = True

Kod satırlarını yazmamız yeterli olacaktır.

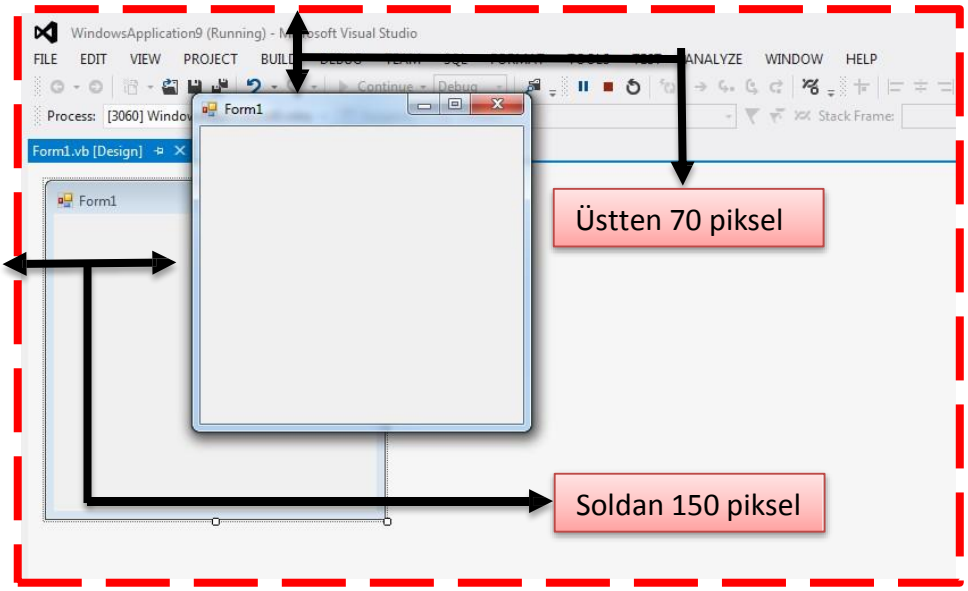
Fakat burada dikkat etmemiz gereken bir husus bulunmaktadır. Bu husus ise üzerinde çalıştığımız formun visible özelliğini değiştirmemektedir. Örneğin form yüklenirken bu değeri false olarak değiştirmemiz çok makul olmayacaktır. Bundan dolayı program build ederken hata verip çalışmayacaktır.

StartPosition Özelliği

Projemizin çalışması esnasında formların nerede ve nasıl konumlanacağını startposition özelliği ile ayarlarız.

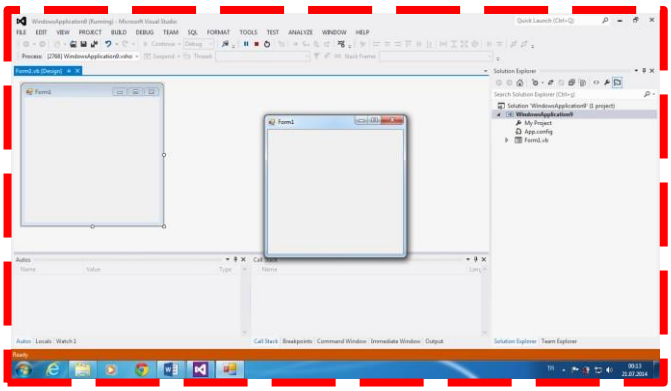
Bu özellik sayesinde hazırladığımız projelerimizi kullanıcılar çalıştırdığı zaman formun ekranın neresinde konumlanacağını ayarlamamızı sağlar. Visual Basic 6'da bu özellik Form Layout isimli grafiksel bir pencere altında gerçekleştiriliyordu. Visual Studio 2012'de ise properties panelinde bulunan startposition özelliği ile 5 farklı şekilde gerçekleştirebileceğiz. Bunlar sırasıyla: Manual, CenterScreen, WindowsDefaultLocation, WindowsDefaultBounds ve CenterParent seçenekleridir. Bu seçenekleri kullanarak projemizi çalıştırdığımız zaman formların nasıl konumlanacağını ve neler yapmamız gerektiğini sırayla inceleyelim.

Manuel: Bu özelliği kullandığımız zaman aynı panel üzerinde bulunan location özelliğini de ayarlamamız gerekiyor. Location kısmının sağ tarafında noktalı virgülle ayrılan iki adet sayı bulunmaktadır. Bunlardan ilki ekranımızda soldan uzaklığı, ikincisi ise üstten uzaklığı piksel cinsinden ifade etmektedir. Bu değerleri elimizle 150;70 girerek projemizi çalıştıralım.



Şekil 2.10 StartPosition özelliğinin manuel olarak ayarlanmış ekran görüntüsü

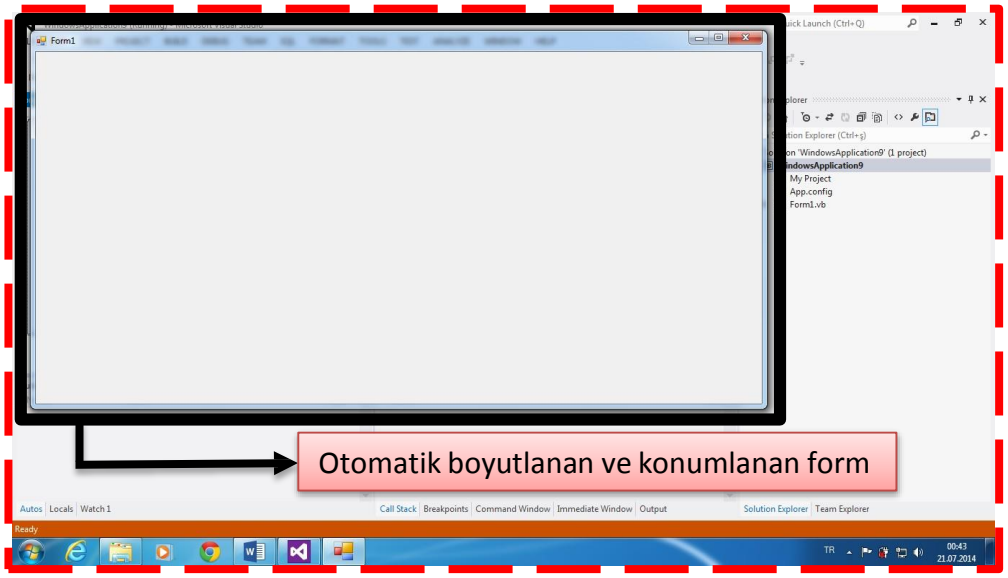
CenterScreen: Bu seçeneği işaretleyerek projemizi başlatacak olursak formumuz ekranın ortasında konumlanacaktır.



Şekil 2.11 StartPosition özelliğinin manuel olarak ayarlanmış ekran görüntüsü

WindowsDefaultLocation: Bu özellik programın sunmuş olduğu varsayılan değerdir. Windows tarafından belirlen bir konumda proje başlatılır. Genellikle bu konum ekranın sol üst köşesidir.

WindowsDefaultBounds: Bu seçenek ile hazırlamış olduğumuz form Windows uygulamaları için standart bir boyutta varsayılan pozisyonda konumlanacaktır. (Şekil 1.13)



Şekil 2.12 StartPosition özelliğinin WindowsDefaultBounds olarak ayarlanmış ekran görüntüsü

CenterParent: Bu seçeneği işaretlediğimiz form kendini çalıştıracak olan parent formun ortasında konumlanır. Birbirleriyle ilişkili olan MDI formlar için kullanışlıdır.

MaximizeBox, MinimizeBox ve WindowState Özellikleri

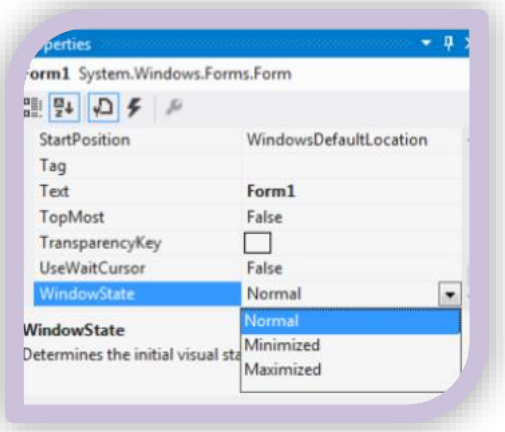
Maximizebox ve minimizebox özelliklerine aynı anda false değeri atanırsa formun üzerinde bulunan bu değere ait butonlar görünmeyecektir.

Projemizde hazırladığımız formun çalışma esnasında tam ekranı kaplaması, simge durumuna küçülmesi veya görev çubuğunda görünmesiyle ilgili özellikleri ayarlayabiliriz.

Projemizi çalıştırdığımız zaman formun sağ üst köşesinde bulunan  ekranı kapla butonunun aktif olmasını istemiyorsak, properties panelinde bulunan maximizebox özelliğini false olarak değiştirmeliyiz.

Projemizi çalıştırdığımız zaman formun sağ üst köşesinde bulunan  simge durumuna butonunun aktif olmasını istemiyorsak, properties panelinde bulunan minimizebox özelliğini false olarak değiştirmeliyiz.

Şayet yukarda bahsettiğimiz bu iki özelliği aynı anda false olarak değiştirirsek projemizi çalıştırdığımız zaman başlık çubuğunda bu butonlar görünmeyecektir. Bunlardan sadece birini false olarak ayarlarsak ekranda görünürler fakat false olarak ayarlanan özellik pasif olur ve üstünde herhangi bir işlem gerçekleştirilemez.

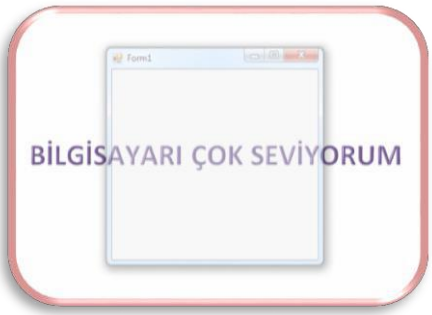


WindowState özelliği için yandaki şekilde de göründüğü gibi üç farklı durum söz konusudur. Bunlardan ilki olan *Normal* seçeneği varsayılan olarak atanan değerdir. Bunu seçtiğimiz zaman formumuz ayarlanan boyutlarda ekran üzerinde görünecektir. *Minimized* seçeneği ise form açıldığı zaman ilk olarak formun görev çubuğuna yerleşerek başlamasını ifade etmektedir. Görev çubuğundaki simgesine tıklayarak ekrana getirebiliriz. Son seçenek olan *Maximized* seçildiğinde ise form, ekranı tam kaplayacak şekilde açılmaktadır.

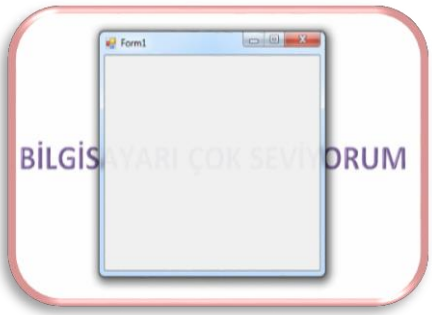
Opacity özelliği en fazla 100 en az 0 değerini alabilir.

Opacity Özelliği

Bu özellik 0-100 arasında değer alabilen bir yapıdadır. Aksi değerler girildiği takdirde hata mesajı vermektedir. İşlerliğinden bahsedecek olursak, formumuzun saydamlığını ayarlamamıza imkân sağlamaktadır. En küçük değer olan sıfır değerini yazdığımızda formun arkasında ne oluğunu tam olarak görebiliriz. Yüz değerini yazdığımızda ise formun arkasını göremeyiz.



Şekil 2.13a Opacity=40%



Şekil 2.13b Opacity=95%

Yukardaki şekillerde de gördüğümüz gibi opacity değeri %40 olarak ayarlanmış (şekil 1.14a) formun arkasında kalan yazı okunabiliyorken, %95 olarak ayarlanmış (şekil 1.14b) formun arkasında kalan yazı okunamıyor.

ShowInTaskbar Özelliği

Bu özellik projemizin simgesinin çalışma esnasında görev çubuğu üzerinde bulunup bulunmayacağını ayarlamamıza imkân sağlar. True veya False olmak üzere iki farklı değer alabilir.

Şekil 2.14 Showintaskbar özelliğinin true değeri için görev çubuğunun görüntüsü

AcceptButton ve CancelButton Özellikleri

Acceptbutton ve cancelbuttonun aktif olması için formun seçilmesi gerektiği unutulmamalıdır.

Projelerimizde bazen gerekli alanları doldurup onay işlemi vermemiz gerekebilir. Durum tersi şeklinde ilerleyerek iptal etmemiz de gerekebilir. Bu gereklilikler karşısında onay ve iptal işlemleri için formumuza butonlar yerleştiririz. Burada kullanacağımız özellik bu butonları fare yardımı kullanmadan işlevsel yapmamızı sağlayacaktır.

Onay işlemleri için genellikle klavye üzerinde bulunan enter tuşu kullanılmaktadır. Acceptbutton özelliği ile formumuz seçili iken form üzerinde bulunan butonlar görüntülenir. Bu butonlardan hangisini onay işlemi için kullanmak istiyorsak onu seçeriz. Bu işlem yapıldıktan sonra seçilen butonumuz enter tuşuna basıldıkça görevini icra edecektir.

Acceptbutton için yaptığımız işlemlerin aynısını Cancelbutton için yaparak iptal işleminde kullanacağımız butonla ESC tuşunu eşleştirmiş oluruz. Aşağıdaki örneği inceleyerek daha iyi kavrayabiliriz.

Örnek

•Formumuz üzerine bir adet textbox ve iki adet buton yerleştirerek tasarım alanında ve kod editöründe yapmamız gerekenleri aşağıdaki adımları izleyerek gerçekleştirelim, enter ve ESC tuşları için ekran görüntüsünü inceleyelim.

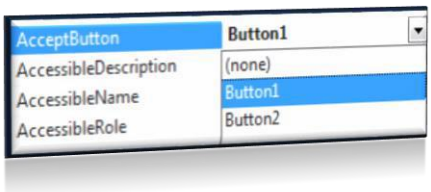
- Formumuz üzerine ilk yerleştirdiğimiz butonun text özelliğini properties panelinden “onay” olarak değiştirelim.
- İkinci butonun text özelliğini “iptal” olarak değiştirelim.
- Onay butonuna çift tıklayarak kod editörüne aşağıdaki kod satırını yazalım.

```
Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
        MsgBox("adınızı enter tuşu ile onayladınız")
    End Sub
```

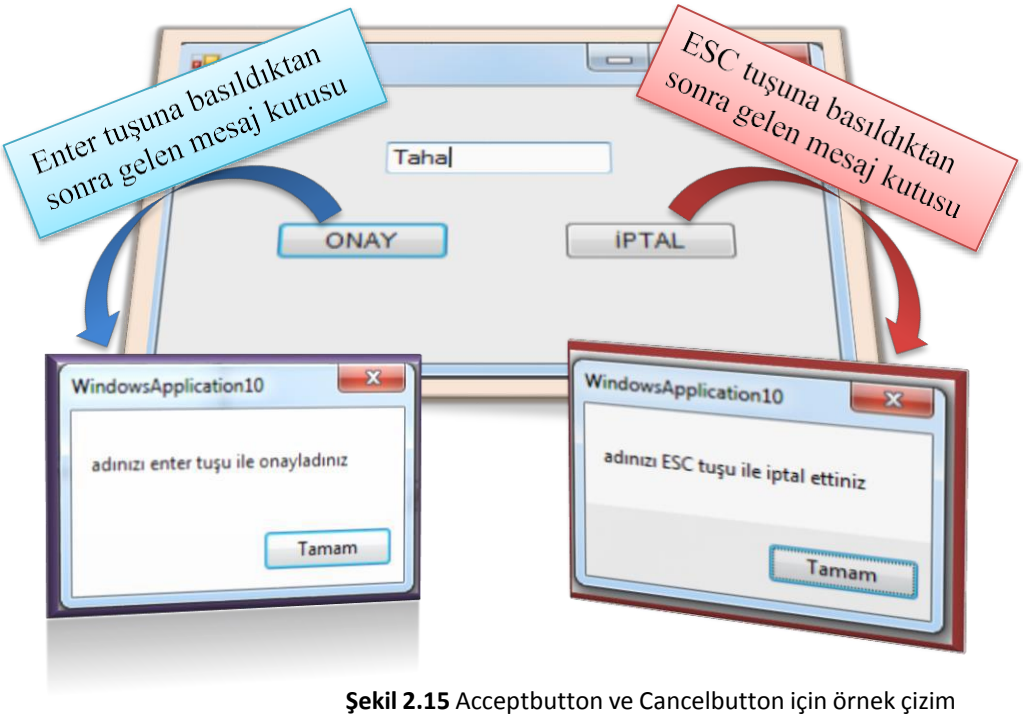
- İptal butonuna çift tıklayarak aşağıdaki kod satırını yazalım.

```
Private Sub Button2_Click(sender As Object, e As EventArgs)
Handles Button2.Click
        MsgBox("adınızı ESC tuşu ile iptal ettiniz")
    End Sub
```

- Dizayn kısmında *formumuzu seçtikten sonra* properties panelinden sırasıyla acceptbutton özelliği için buton1’i cancelbutton özelliği için ise buton2’yi seçiniz.



- Start butonuna basarak projemizi çalıştıralım.



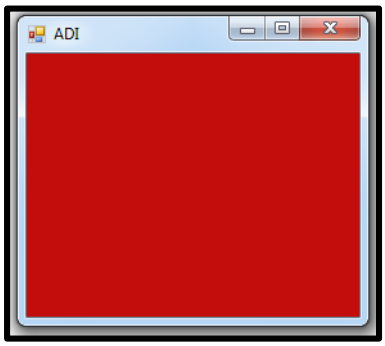
Şekil 2.15 Acceptbutton ve Cancelbutton için örnek çizim

Yukardaki adımları sırayla takip edip programımızı çalıştırdıktan sonra şekil-15'deki ekran görüntüsünü elde ederiz. Tabi enter ve esc tuşlarına bastığımız zaman alacağımız mesaj kutuları ayrı ayrı karşımıza gelecektir.

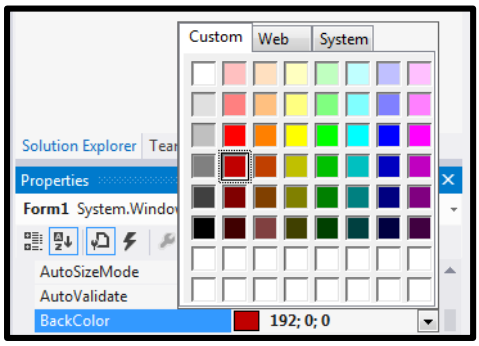
BackColor Özelliği

Forma arka plan rengi vermek için backcolor özelliğini kullanmalıyız.

Formun arka plan rengini değiştirme gereği duyulduğunda kullanılan bir özelliktir. Bu özelliğin yanındaki oka tıkladığımızda Custom, Web ve System olmak üzere üç sekme altında bize renk alternatifi sunmaktadır. Aşağıda Şekil 1.17b'de custom sekmesinden seçilmiş bordo rengindeki formun çalışma esnasında görüntüsü şekil 1.17a'da verilmiştir.



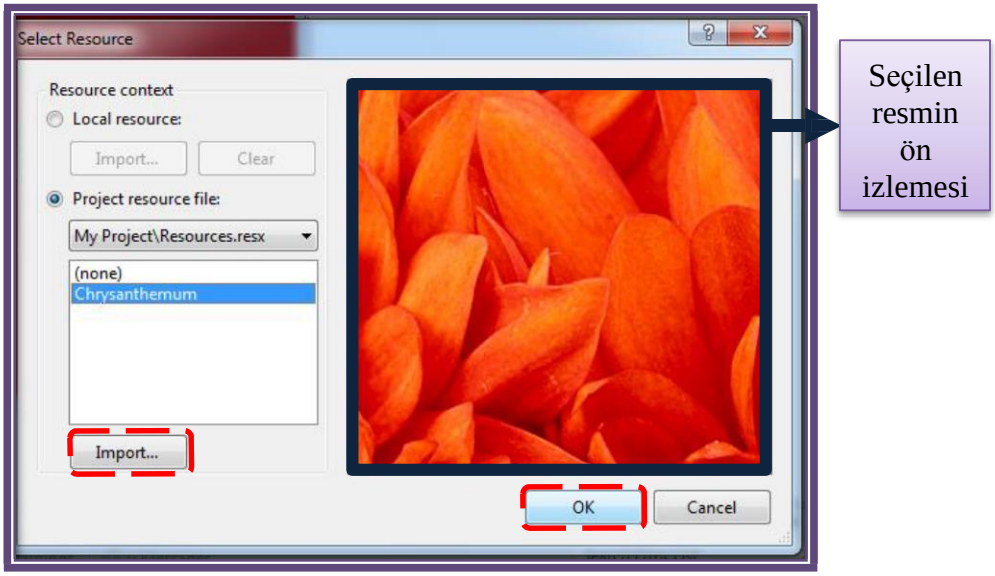
Şekil 2.16a Arka plan rengi değiştirilmiş örnek form



Şekil 2.16b BackColor özelliği Custom sekmesi

BackgroundImage Özelliği

Bu özellik sayesinde hazırladığımız formları istediğimiz ölçüde özelleştirebiliriz. Formun arka planına var olan bir resim dosyası yükleyebiliriz. Properties panelinde bulunan BackgroundImage özelliğinin yanındaki ... butonuna tıkladığımızda karşımıza gelen pencereden (Şekil 1.18) import butonuna tıklayarak eklemek istediğimiz resmin bilgisayarımızdaki yolunu gösterir ve ardından ok butonuna tıklarız.



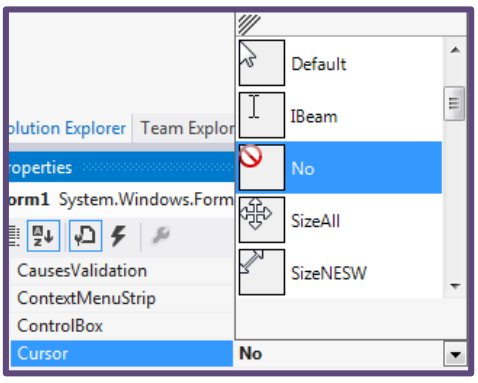
Şekil 2.17 Forma resim dosyası ekleme

Burada dikkat etmemiz gereken bir husus bulunmaktadır. Resim dosyasını formumuza ekledikten sonra bu resmin form üzerinde nasıl konumlanacağını ayarlamamız gerekmektedir. Bu işlemi yapabilmek için *BackgroundImageLayout* özelliğini kullanabiliriz. Bu özellik içinde dört farklı hizalama seçeneği bulunmaktadır. Bunlar: Tile, Center, Stretch ve Zoom seçenekleridir. Projemize en uygun olan seçeneği kullanarak projemizi çalıştırabiliriz.

Form üzerine eklediğimiz resmin konumunu ayarlayabilmek için back ground image layout seçeneğini kullanmalıyız.

Cursor Özelliği

Fare işareti yapacağı göreve veya bulunduğu duruma göre farklı şekillerde olabilir. Projemizin çalışması esnasında fare ile form üzerinde gezinirken farenin görünmesini istediğimiz şekli belirlemeye yarayan özellik cursor'dür.



Şekil 2.18 Fare işaretçisini seçebileceğimiz seçeneklerin bulunduğu cursor özelliği

Özet

- Visual Studio .NET aracılığıyla hazırlayacağımız projelerde visual basic programlama dilini kullanmak bize windows pencere yapısında formlar hazırlamamıza imkân sunmaktadır. Hazırlayacağımız projelerin SDI veya MDI arayüze sahip olması bizim kişisel tercihimiz dahilindedir.
- Visual Studio 2012 programını başlattıktan sonra karşımıza gelen ekranda sırasıyla File -> New -> Project menü yolunu veya ctrl + shift + N kısayol tuşunu kullanarak açılan New Project penceresinde templates sekmesinden visual basic programlama dilini ve Windows Forms Application seçeneğini işaretleyip ok butonuna bastıktan sonra projemizi açabiliriz.
- Projemizi başlatıp form yapısını belirledikten sonra formu istediğimiz ayarlarda kullanabilmek için dizayn alanında bulunan properties panelindeki özellikleri değiştirerek işlemlerimizi gerçekleştirebiliriz. Buna ek ve daha detaylı olarak visible gibi birtakım özellikleri kod editörünü kullanarak da ayarlama imkânına sahibiz.

DEĞERLENDİRME SORULARI

1. Visual Studio 2012 programını başlattıktan sonra hangi kısayol tuşunu kullanarak New Project penceresini açabilirsiniz?

    a) ctrl + shift + M
    b) ctrl + N
    c) shift + N
    d) ctrl + shift + N
    e) ctrl + Alt + M

2. Projemize yeni bir form eklemek için Add New Item penceresini hangi kısayol tuşunu kullanarak açabilirsiniz?

    a) ctrl + shift + A
    b) ctrl + shift + N
    c) ctrl + shift + S
    d) ctrl + shift + M
    e) ctrl + shift + F

3. Projemizde var olan bir SDI formu properties panelindeki hangi özellik sayesinde MDI yapıdaki bir forma dönüştürebiliriz?
    a) Enabled
    b) IsMdiContainer
    c) Visible
    d) Opacity
    e) Background

4. Projemizde var olan formla ilgili kod editöründe işlem yapmak için kullandığımız ismi hangi özellik içinden alırız?
    a) Cursor Özelliği
    b) Opacity Özelliği
    c) Name Özelliği
    d) Visible Özelliği
    e) Font Özelliği

5. Aşağıdaki seçeneklerden hangisi Startposition özelliğinin ayarları arasında bulunmaz?
    a) Manual
    b) Centerscreen
    c) Windowsdefaultlocation
    d) Windowsdefaultbounds
    e) Windowstate

6. Projemizde var olan formların opacity özelliğinin alabileceği en büyük ve en küçük değerler aşağıdakilerden hangisinde doğru verilmiştir?

    a) 100 – 30
    b) 100 – 0
    c) 95 – 40
    d) 150 – 10
    e) 95 – 0

7. Formumuz üzerinde bulunan butonlardan herhangi birine enter tuşu ile işlem yapabilme özelliğini aktarmak istiyorsak aşağıdakilerden hangisini değiştirmeliyiz?

    a) Acceptbutton
    b) Cancelbutton
    c) Maximizebox
    d) Minimizebox
    e) Windowstate

8. Projede var olan bir formun arka plan rengini değiştirmek için kullanacağımız backcolor özelliği hangi sekmelerden oluşmaktadır?

    a) Custom – Net – System
    b) Color – Web
    c) Web – System
    d) Custom – System
    e) Custom – Web – System

9. Back ground Image Layout özelliğinde aşağıdaki seçeneklerden hangisi bulunmaz?

    a) Tile
    b) Stretch
    c) Center
    d) Maximize
    e) Zoom

10. Projemizin çalışması esnasında fare ile form üzerine gelindiğinde farenin ne şekilde görüneceğini hangi özellik ile ayarlarız?

a) Acceptbutton
b) Opacity
c) Cursor
d) Visible
e) Enabled

Cevap Anahtarı

1.d,2.a,3.b,4.c,5.e,6.b,7.a,8.e,9.d.10.c

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Yeniman Yıldırım, E. (2007). Yüksekokul Öğrencileri İçin Uygulamalı Görsel Programlama Visual Basic, 2. Baskı, ANKARA: Nobel Yayın Dağıtım.

Halvorson, M. (2010). Microsoft Visual Basic 2008. (Ü. Tezcan, Çev.). ANKARA: Arkadaş Yayınevi. (2008)

Halvorson, M. (2001). Microsoft Visual Basic 6.0 Professional, 2. Baskı. (S. Göksu, Çev.). ANKARA: Arkadaş Yayınları. (1998)

Yanık, M. (2010). Visual Studio 2010 ile Microsoft Visual Basic 10 for .NET Framework 4.0, 1. Baskı, ANKARA: Seçkin Yayıncılık.

STANDART NESNELER

İÇİNDEKİLER

- Görsel Programlama
- .NET Framework
- Visual Studio
- Formlar ve Kontroller
- Örnek Uygulama

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - .NET Platformunu bilebilecek,
  - Görsel Programlamayı tanımlayabilecek,
  - Sınıf ve nesne kavramlarını öğrenebilecek,
  - Visual Studio Kontrollerini tanıyabilecek,
  - Basit Kontrolleri kullanabileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr. Mustafa KESKİNKILIÇ

ÜNİTE

3

GİRİŞ

Görsel programlama genel olarak resim ve grafiksel öğelerden faydalanılarak yapılan bir programlama türüdür. Özellikle Windows işletim sisteminin yaygın olarak kullanılmasından sonra görsel nesneleri içeren programlara ihtiyaç duyulmuş ve bunların kullanımı oldukça artmıştır.

Bu ünitede günümüzde görsel programlamanın en güçlü temsilcilerinden biri olan Visual Studio.Net platformu tanıtılacak ve bu platformdaki kullanılan hazır kontrollerin bazı özellikleri incelenecektir.

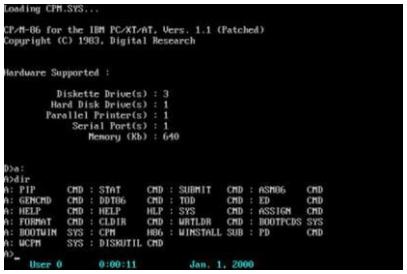
GÖRSEL PROGRAMLAMA NEDİR?

Resim ve grafiksel ögeler ile yapılan bilgisayar programlamasına görsel programlama denir. Görsel programlama dilleri program kodunun tamamının ya da bir kısmının görsel araçlar ile üretildiği dillerdir.

Görsel programlama sayesinde bilgisayarlar daha anlaşılır, daha kolay kullanımlı, daha göze hitap eden bir hale gelmiştir.

Windows tabanlı sistemler için Visual Basic, Visual C++, C#, Java gibi diller görsel programlama dilleri olarak kullanılabilmektedir. Ancak F# fonksiyonel bir programlama dilidir.

Görsel programlama ile bilgisayar işletim sistemleri (Windows, Linux, MacOS), aktif web siteleri, cep telefonu uygulamaları, tablet uygulamaları geliştirilmektedir.



Resim 1.1. MS-DOS Görsel Programlamasız İşletim Sistemi

Resim 1.2. Windows 7 Görsel Programla İşletim Sistemi

.NET FRAMEWORK NEDİR?

.NET Framework, Microsoft tarafından geliştirilen, açık İnternet protokolleri ve standartları üzerine kurulmuş bir "uygulama" geliştirme platformudur. Visual Studio.Net ile geliştirilen uygulamaların çalışması için .Net Framework'e ihtiyaç duyulur. Bu yüzden kullanım amacı olarak Java Virtual Machine uygulamasına benzetilebilir.

.Net Framework ile Visual Studio.Net platformunda geliştirilen uygulamaların farklı bilgisayarlarda veya aygıtlarda çalışabilmesi sağlanmaktadır. Visual Studio ile geliştirilen uygulamaların ihtiyaç duyduğu tüm kütüphaneler *.Net Framework*'te tanımlıdır. Kısacası .Net Framework olmadan Visual Studio.Net ile geliştirilen uygulamalar çalışmaz. Bu sebeplerden ötürü .Net Framework bir

çalıştırma ortamı (runtime environment) olarak adlandırılmaktadır.

VISUAL STUDIO NEDİR?

Microsoft tarafından 2000'li yılların başlarında kullanıma sunulan Microsoft Visual Studio ürünü, programcıların her türlü yazılım geliştirme ihtiyacını karşılamak amacıyla geliştirilen bir yazılım geliştirme platformu olarak tanımlanmaktadır.

Sağlamış olduğu tümleşik geliştirme ortamı(IDE) sayesinde programcılara yazılım geliştirirken büyük kolaylıklar sağlamaktadır. Visual Studio'da geliştirilen yazılımlar Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework ve Microsoft Silverlight tarafından desteklenen tüm platformlarda çalışır. Bir başka ifadeyle birçok farklı teknoloji ve platform tarafından desteklenmektedir.

Visual Studio içinde bulunan Windows Forms uygulamaları, web sayfaları ve web servisleri ile birlikte konsol ve grafiksel kullanıcı ara yüzüne sahip uygulamalar geliştirilebilmektedir. Bunun yanında aynı projede farklı programlama dilleri ile kod yazabilir, birçok türde veri kaynağına erişim sağlayabilir ve burada sayamadığımız birçok işlemi çok daha hızlı, kolay ve güvenilir bir şekilde gerçekleştirebilmektedir. Bütün bu sebeplerden ötürü Visual Studio günümüzde en çok tercih edilen yazılım geliştirme platformlarından birisidir.

KONTROL NESNELERİ

Bütün sekmelerde Pointer (Fare işaretçisi) ortak olarak bulunur çünkü işaretçi görsel programların vazgeçilmez ana öğesidir.

Windows işletim sistemindeki pencereler, Visual Studio'da Form olarak adlandırılmaktadır. Toolbox'ı kullanarak forma eklediğimiz her kontrol (buton, yazı kutusu), kontrol nesnesi olarak adlandırılmaktadır.

Bütün formlar ve onlara eklediğimiz her kontrol, birer nesnedir. Bunlar birer nesne olduğu için bunların özellikleri, olayları ve metotları vardır.

Günümüzün yüksek ihtiyaçlarına karşılık verecek, göze hitap eden, yüksek işlevli bir nesnenin programcı tarafından geliştirilmesi can sıkıcı uzunlukta sürmektedir. Visual Studio'da hem kullanıcıların hem de programcıların ihtiyaçlarına karşılık veren onlarca kontrol nesnesi hazır bir şekilde bulunmaktadır. Ayrıca programcılar istedikleri nesneleri Toolbox'a ekleyebilmektedir. Bu kontrolleri programcılar istediği gibi kullanabilmektedir.

Görsel bir program 2 ana parçadan oluşmaktadır: Arayüz kısmı, Kod kısmı. Arayüze kontrol eklerken Toolbox kullanılır. Eklediğimiz her kontrol birer nesnedir. Bütün nesneler gibi özellikleri ve metotları vardır. Ancak kontrol nesneleri 3.Ünite'de gösterildiği gibi birkaç satır kod ile yazılamamaktadır. Kontol nesnelerinin geliştirilmesi çok uzun ve yoğun bir çalışma gerektirir. Kontrol nesneleri basit nesneler değildir, gelişmiş nesnelerdir.

Kontrol nesneleri ileri düzey ihtiyacınıza tam cevap verir, programa kolaylıkla eklenir, özelliklerine ve metotlarına kolaylıkla erişilir. Kontrol nesnelerin onlarca özelliği ve onlarca metodu hazır haldedir. Nesne hazırlamak için uğraşmazsınız. Yazılımı kolaylıkla geliştirirsiniz, olası hataları en aza indirirsiniz ve ortaya çıkan hataları kolaylıkla giderirsiniz. Bu .NET Framework platformunun avantajlarından biridir.

FORMLAR ve KONTROLLER

Visual Studio'nun içerisinde varsayılan olarak bulunmakta olan kontrollerdir. Neredeyse bütün yazılımlarda kullanılan, temel ve en çok ihtiyaç duyulan kontrolleri barındırır.

- *All Windows Forms:* Bütün sekmelerdeki kontrol nesnelerin topluca bulunduğu sekmedir.
- *Common Controls:* En sık kullanılan nesnelerin bulunduğu sekmedir.
- *Containers:* Konteynırlar birer nesne oldukları gibi içerilerine de nesneler eklenebilir. Formlar da konteynırlara örnektir ancak buradan eklenmezler. Konteynırlar bu sekmededir.
- *Menus & Toolbars:* Arayüze menü ve araç çubuğu nesnelerinin eklendiği sekmedir.
- *Data:* Veri tabanı işlemlerini gerçekleştirmemizi sağlayan nesnelerin bulunduğu sekmedir.
- *Components:* Arka planda çalışan önemli bileşenlerini barındırır. Bu sekmedeki nesneler eklendiğinde görsel arayüzde bulunmaz ancak "Design" sekmesinin alt kısmında görülebilir.
- *Printing:* Yazdırma işlemiyle ilgili nesnelerin bulunduğu sekmedir.
- *Dialogs:* Çeşitli iletişim kutularının bulunduğu sekmedir.
- *WPF Interoperability:* WPF uygulamaları ile birlikte çalışmayı sağlayan nesnelerin bulunduğu sekmedir. İçerisinde fare işaretçisi dışında 1 öge bulunmaktadır.
- *Visual Basic Power Packs:* Visual Basic Güçlendirme Paketindeki nesneleri bulundurur.
- *General:* Özelleştirebileceğimiz bir sekmedir. İstediğimiz nesneyi sürükle-bırak yöntemiyle bu sekmeye ekleyebiliriz.

Tablo 1.1. Sık Kullanılan Kontroller

NO	AD	AÇIKLAMA
COMMON CONTROLS		
1	Button	Tıklama butonu
2	CheckBox	İsteğe göre seçim kutusu
3	CheckedListBox	Liste şeklinde seçim kutusu
4	ComboBox	İçinden seçim yapılan açılır liste kutusu
5	DateTimePicker	Tarih seçimi yapılan kontrol
6	Label	Yazı etiketi
7	LinkLabel	Link etiketi
8	ListBox	Liste kutusu
9	ListView	Liste görünümü
10	MaskedTextBox	Çok fonksiyonlu metin kutusu
11	MonthCalendar	Aylık takvim görünümü
12	NotifyIcon	Uyarı simgesi
13	NumericUpDown	Sayaç
14	PictureBox	Resim kutusu
15	ProgressBar	İlerleme düzeyi çubuğu
16	RadioButton	Çok seçenekten birini seçme butonu
17	RichTextBox	Zengin metin kutusu
18	TextBox	Metin kutusu
19	ToolTip	İpucu kutusu
20	TreeView	Ağaç görünümlü liste
CONTAINERS		
21	FlowLayoutPanel	Kontrolleri sıraya koyan panel
22	GroupBox	Kontrolleri gruplandıran kutu
23	Panel	Panel
24	SplitContainer	Genişletilebilir panel
25	TabControl	Sekme kontrolü
26	TableLayoutPanel	Tablo tarzı panel
MENUS & TOOLBARS		
27	ContextMenuStrip	Sağ tıklama menüsü
28	MenuStrip	Menü çubuğu
29	StatusStrip	Durum çubuğu
30	ToolStrip	Araç çubuğu
31	ToolStripContainer	Araç çubuğu sıralayan nesne
DIALOGS		
32	ColorDialog	Renk seçimi diyaloğu
33	FolderBrowserDialog	Klasör seçimi diyaloğu
34	FontDialog	Yazı tipi seçimi diyaloğu
35	OpenFileDialog	Dosya açma diyaloğu
36	SaveFileDialog	Dosya kaydetme diyaloğu

KONTROLLERİN GENEL ÖZELLİKLERİ

Bütün kontrol nesnelerinde ortak bulunan 14 özelliği tabloda görebilirsiniz.

Tablo 1.2. Kontrol Nesnelerinin Ortak Özellikleri

AD	AÇIKLAMA
Anchor	Kilitler. Formun boyutu değişse de yeri sabit kalır.
BackColor	Arkaplan rengini ayarlar.
Cursor	Kontrolün üzerindeyken fare imlecini ayarlar.
Dock	Formun bir kısmının kontrolle dolmasını ayarlar.
Enabled	Kontrolün aktifliğini ayarlar.
Font	Kontrolün üzerindeyken yazı özelliklerini ayarlar.
ForeColor	Kontrolün üzerindeki yazının rengini ayarlar.
Location	Kontrolün konumunu ayarlar.
Size	Kontrolün boyutunu ayarlar.
TabIndex	TAB tuşu ile kontrol geçiş sırasını ayarlar.
TabStop	TAB tuşu geçişinin kontrolü atlamasını ayarlar.
Text	Yazı özelliğidir.
TextAlign	Kontrollerin üzerindeki yazının yerini ayarlar.
Visible	Kontrolün görünürlüğünü ayarlar.

KONTROLLERİN GENEL OLAYLARI

Bütün kontrol nesnelerinde ortak bulunan olayı tabloda görebilirsiniz.

Tablo 1.3. Kontrol Nesnelerinde Bulunan Ortak Olaylar

AD	AÇIKLAMA
BackColorChanged	Kontrolün BackColor özelliği değişince kontolü ele alır.
BackgrondImageChanged	Kontolün BackgroundImage özelliği değişince kontolü ele alır.
Click	Kontrole tıklanınca devreye girer.
ControlAdded	Kontrolün üzerine yeni bir kontrol eklenince kontolü ele alır.
ControlRemoved	Kontrolün üzerindne bir kontrol silinince kontolü ele alır.
CursorChanged	Kontrolün Cursor özelliği değişince kontolü ele alır.
Disposed	Kontrol Dispose metoduyla yok edilirken çağrılır.
DoubleClick	Kontrole çift tıklanınca devreye girer.

DragDrop	Kontrole sürükle bırak yapılınca kontrolü ele alır.
DragEnter	Kontrole sürüklenen şey sınırdan içe geçince kontrolü ele alır.
DragLeave	Kontrole sürüklenen şey sınırdan çıkınca kontrolü ele alır.
DragOver	Kontrole bir şey sürüklenip daha bırakılmayınca devreye girer.
GotFocus	Kontrole odaklanınca kontrolü ele alır.
KeyDown	Kontrol aktifken bir tuşa basılınca kontrolü ele alır.
KeyPress	Kontrol aktifken bir tuşa basıp bırakınca kontrolü ele alır.
KeyUp	Kontrol aktifken basılı bir tuş bırakılırken kontrolü ele alır.
Layout	Kontrol ekranda gözükünce kontrolü ele alır.
LocationChanged	Kontrolün yeri değişince kontrolü ele alır.
MouseClick	Kontrole tıklanınca kontrolü ele alır.
MouseDoubleClick	Kontrole çift tıklanınca kontrolü ele alır.
MouseDown	Kontrole tıklanırken Mouse butonu inerken kontrolü ele alır.
MouseEnter	Kontrolün sınırına Mouse ile girilince kontrolü ele alır.
MouseLeave	Kontrolün sınırından Mouse ile çıkılırken kontrolü ele alır.
MouseMouse	Kontrolün üzerinde Mouse hareket edince kontrolü ele alır.
MouseUp	Kontrole tıkladıktan sonra buton yukarı çıkarken devredir.
MouseWhell	Kontrol üzerinde Mouse tekerleği hareketlenince devrededir.
Move	Kontrol hareket ederken kontrolü ele alır.
Paint	Kontrol ekranda çizilirken kontrolü ele alır.
Resize	Kontolün boyutu değişince kontrolü ele alır.
SizeChanged	Kontrolün Size özelliği değişirken kontrolü ele alır.
TextChanged	Kontrolün Text özelliği değişirken kontrolü ele alır.
Visible	Kontrolün Visible özelliği değişirken kontrolü ele alır.

Kontrollerle ilgili daha geniş bilgiye MSDN web sitesinden ulaşılabilir.

STANDART KONTROLLER

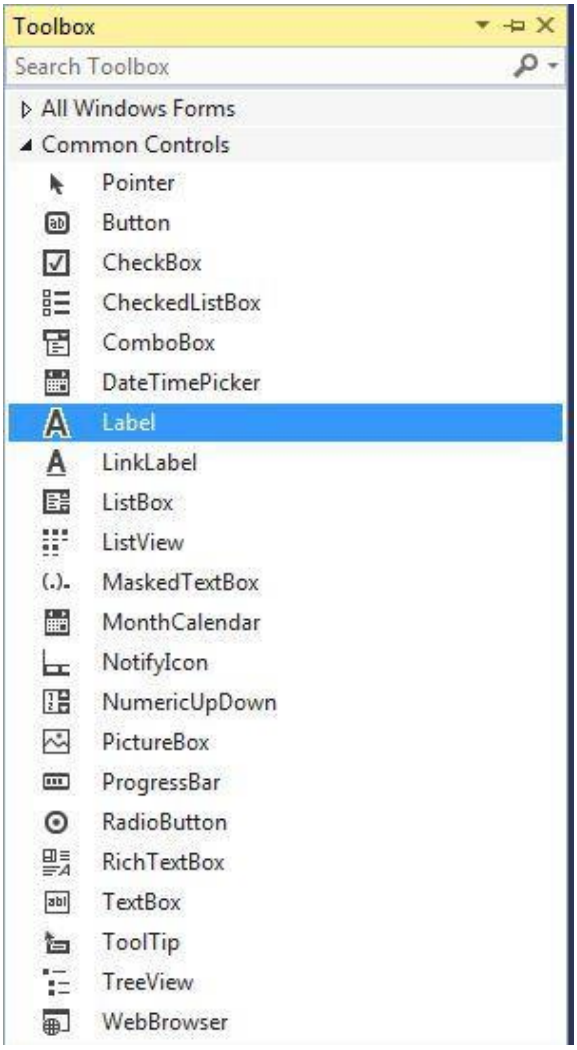
Visual Studio.Net içerisinde Windows uygulamalarında kullanılabilecek birçok kontrol bulunmaktadır. Bu kontrollerin bazıları hemen hemen her uygulamada kullanılır. Bu başlık altında Windows uygulamaları geliştirirken en sık kullanılan *Label, TextBox ve Button* kontrolleri incelenecektir.

Label Kontrolü

Form üzerinde kullanıcıya bilgi vermek amacıyla genelde içeriği sabit bilgilerin görüntülemesinde kullanılır. Aşağıda Form üzerine sürüklenip bırakılan bir Label kontrolünün bazı özelliklerine değer atanmasını gösteren kodlar görülmektedir:

```
Label1.Text = "Bilgiler burada gözükecek"

Label1.ForeColor = Drawing.Color.Red
```

Label kontrolünde görüntülenen bilgiler düzenlenemez veya değiştirilemez. Label kontrolüne Toolbox'ta Common Controls sekmesi altından ulaşılabilir.



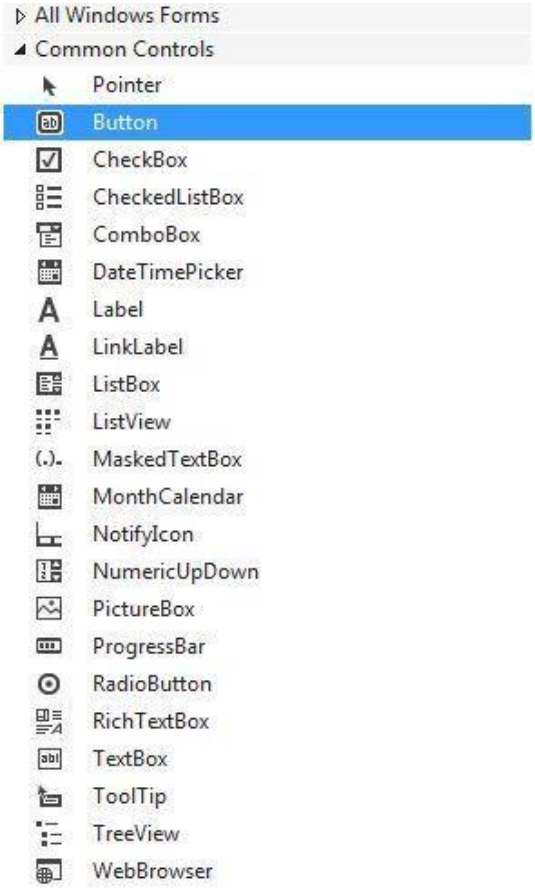
Resim 1.3. Toolbox paneli

Button Kontrolü

Herhangi bir işlemi gerçekleştirmek ya da bir komut vermek amacıyla kullanılan kontroldür. En sık kullanılan olayı Click'tir. Aşağıdaki örnekte Button1 isimli kontrol nesnesinin tıklanma (Click) olayına yazılan ve bir MessageBox gösteren kod örneği görülmektedir.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    MsgBox("Bu bir Test mesajıdır", MsgBoxStyle.Information)
End Sub
```

Koddaki Private Sub Button1_Click kısmı Form Designer'da Button1 nesnesinin üzerine iki kez tıklandığı zaman otomatik olarak oluşur. Bu yüzden Button nesnesinin Click olayına kod yazılmak istendiğinde nesnenin üzerine iki kez tıklanarak çalışması istenen kodların yazılacağı yordam oluşturulabilir.



Resim 1.4. Button kontrolü

Button kontrolün temel kullanım amacı, kullanıcı üzerine tıkladığında bir olay tetiklenmesini sağlamaktır. Button üzerinde görünecek olan metin içeriği Text özelliği ile belirlenir.

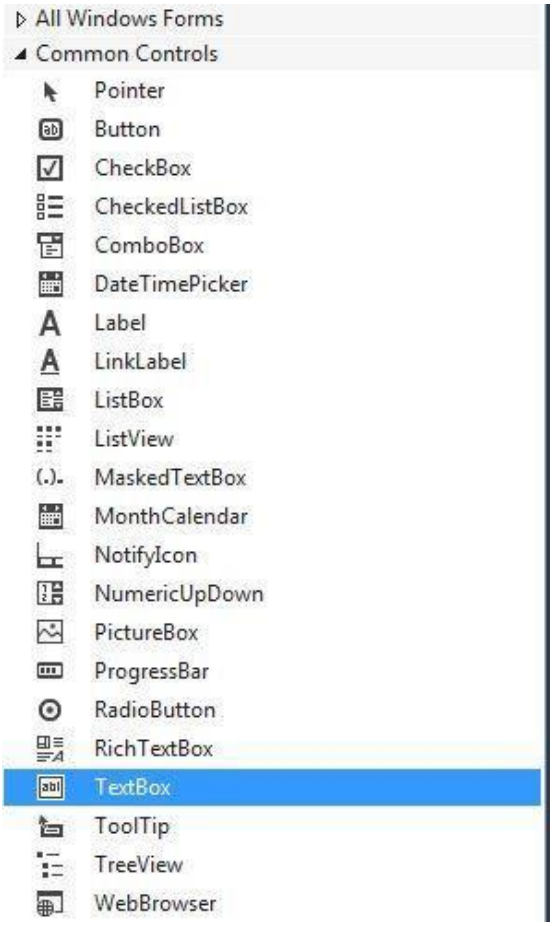
TextBox Kontrolü

İçeriği değiştirilebilecek bilgilerin görüntülenmesi ve kullanıcıdan bilgi almak için kullanılır. En sık kullanılan özelliği olan "Text" ile görüntülenecek veya giriş yapılacak metin belirlenir. Aşağıdaki örnekte TextBox1 nesnesine değer aktarmanın nasıl yapıldığı görülmektedir.

```
TextBox1.Text = "Deneme"
```

TextBox kontrolüne değer atanırken *çift tırnak ("")* kullanılır. TextBox'ın diğer sık kullanılan özellikleri Multiline, Forecolor, Readonly ve Visible özellikleridir. Bu özellikler ile ilgili detaylı bilgiye MSDN web sitesinden ulaşılabilir.

TextBox kontrolü Toolbox'ta Common Controls sekmesi altında bulunmaktadır.

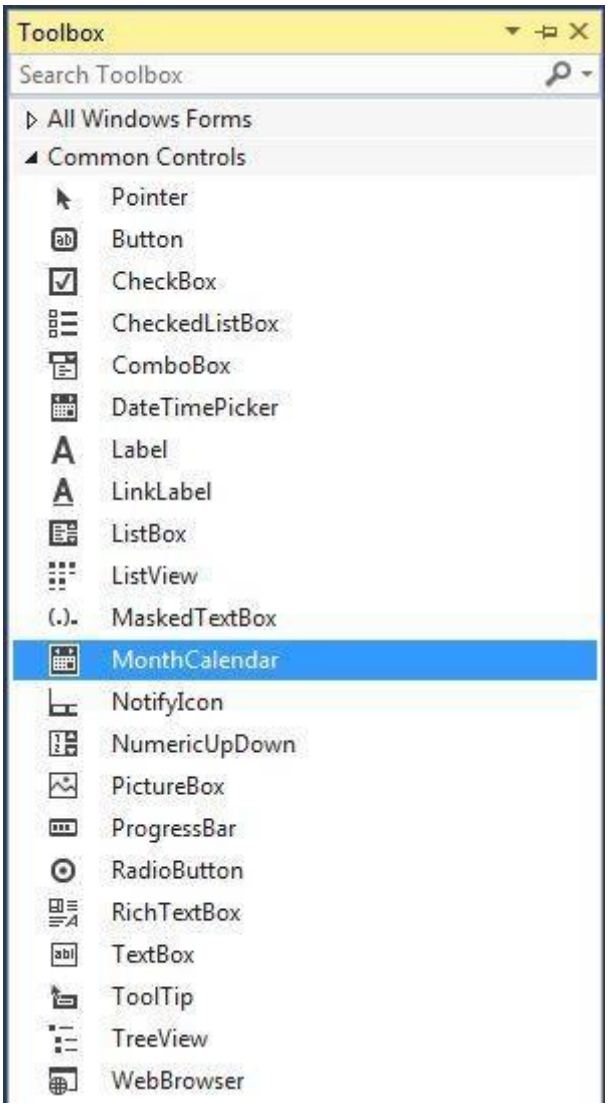


Resim 1.5. Texbox kontrolü

MonthCalendar Kontrolü

Bir tarihi grafiksel olarak göstermeyi ve bu tarihler arasında gezinmeyi sağlayan kontroldür. MonthCalendar ile kullanıcıdan istenilen formatta tarih verisi alınabilir ya da kullanıcıya takvim üzerinde belirli bir tarih gösterilebilir. Ayrıca MonthCalendar kontrolü kullanılarak veritabanından gelen tarih formatındaki bilgiler görüntülenebilir, MonthCalendar'da seçilen tarih parametre olarak veritabanına veya bir fonksiyona gönderilebilir.

MonthCalendar kontrolü ToolBox'ta Common Controls sekmesi altında bulunmaktadır.



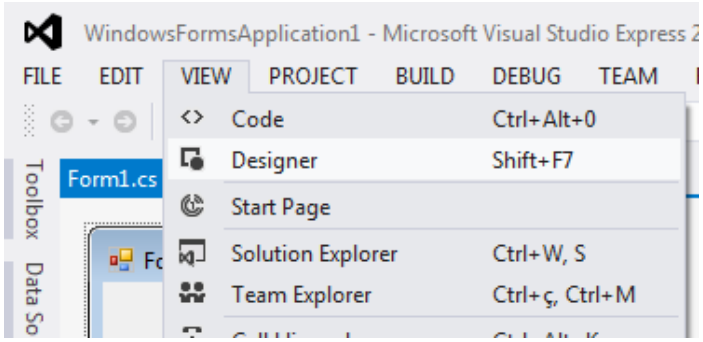
Resim 1.6. MonthCalendar kontrolü

MonthCalendar kontrolünün en sık kullanılan özellikleri şu şekildedir:

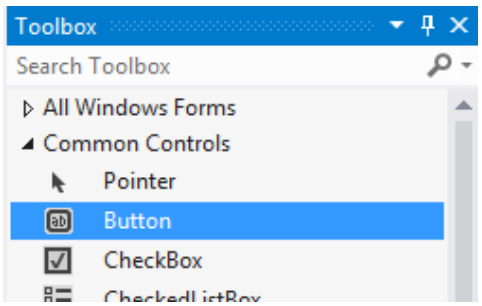
- *SelectedDate:* Bu özellik ile seçilen tarih aralığı alınır.
- *DayNameFormat:* Bu özellik ile gün isimlerinin formatları ayarlanabilir.
- *SelectMonthText:* Bu özellik takvim üzerinde seçilen ay bilgisinin görüntülenmesini sağlar.
- *SelectWeekText:* Bu özellik takvim üzerinde seçilen hafta bilgisinin görüntülenmesini sağlar.

KONTROL NESNELERİYLE BASİT ARAYÜZ TASARLAMAK

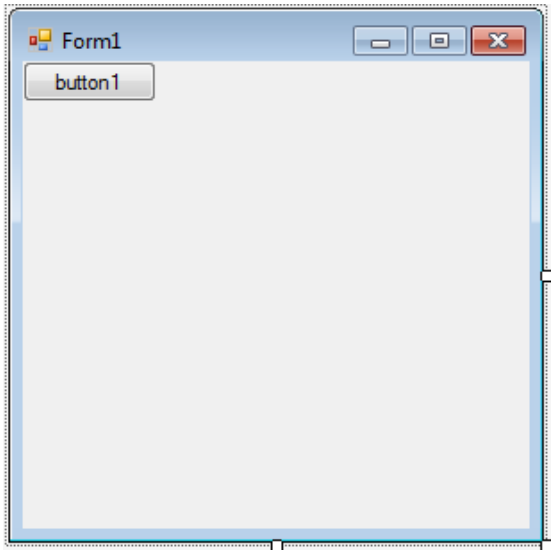
- Kontrol nesnelerini arayüze eklemek için “Design” sekmesine geçilir. “Design” sekmesine geçmek için menüden “View” seçilir. Açılan menüden de “Desinger” seçilir.



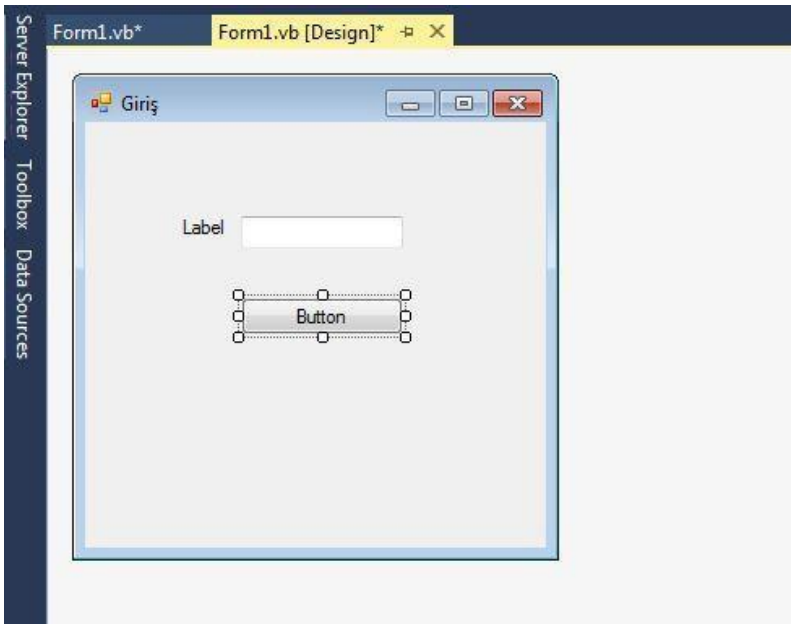
- Toolbox açılır ve istenen nesne seçilir.



- İster çift tıklayarak ister sürükle-bırak ile nesne forma eklenir.



- Properties penceresinden seçili nesnenin özelliklerini değiştirebilir, nesneyi sürükleyerek farklı bir yere taşıyabilir, üzerine çift tıklayarak nesnenin kodlamasını yapabilirsiniz.
- İhtiyacınıza göre nesneler ekleyin ve isteğinize göre tasarlayın.



Bireysel Etkinlik

- Windows Form Designer'ı kullanarak Common Control sekmesinde yer alan kontrolleri inceleyiniz.

Özet

- Bilgisayar, insanların birtakım ihtiyaçlarını (eğlence, bilgi edinme, hesaplama) karşılamak için programlanmaktadır. İnsanların ihtiyaçlarının artması ve bu ihtiyaçların çözümünün zorlaşmasıyla programcıların çok daha fazla çalışması gerekirdi. Programlar geliştikçe boyutları ve bakımının zorluğu arttı. Nesne Yönelimli Programlama ve Nesne Tabanlı Programlama, programlama dünyasındaki karmaşıklığı gidermek için ortaya çıkarılmış bir programlama yaklaşımıdır. Görsel Programlama sayesinde bilgisayarlar göze hitap eden, çok daha kolay kullanılır ve çok daha kolay anlaşılır bir hâl aldı. Bir çok firmanın yaptığı gibi Microsoft da programcılar için kolaylık sağlayan ve bu yeni programlama yaklaşımını destekleyen bir platform (.NET Framework) ve program geliştirme aracı (Visual Studio) hazırladı.
- Form, üzerinde kullanıcıya bilgi vermek amacıyla genelde içeriği sabit bilgilerin görüntülenmesinde kullanılır.
- Button kontrolü herhangi bir işlemi gerçekleştirmek ya da bir komut vermek amacıyla kullanılan kontroldür.
- TexBox kontrolü içeriği değiştirilebilecek bilgilerin görüntülenmesi ve kullanıcıdan bilgi almak için kullanılır

DEĞERLENDİRME SORULARI

1. “Windows işletim sistemindeki pencereler, Visual Studio’da …… olarak adlandırılmaktadır.” Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmektedir?
    a) Form
    b) Container
    c) Component
    d) Dialog
    e) PictureBox

2. Veri tabanı işlemlerinin gerçekleştirilmesini sağlayan nesnelerin bulunduğu ToolBox sekmesi aşağıdakilerden hangisidir?
    a) Containers
    b) Components
    c) Data
    d) Printing
    e) Dialogs

3. İçinden seçim yapılabilen açılır liste kutusu aşağıdakilerden hangisidir?
    a) CheckBox
    b) CheckedListBox
    c) ListBox
    d) ListView
    e) ComboBox

4. Form üzerinde kullanıcıya bilgi vermek amacıyla içeriği sabit bilgilerin görüntülemesinde kullanılan kontrol aşağıdakilerden hangisidir?
    a) TextBox
    b) Label
    c) PictureBox
    d) ProgressBar
    e) OpenFileDialog

5. Kullanıcıdan bilgi almak ve içeriği değiştirilebilecek bilgileri görüntülemek için kullanılan kontrol aşağıdakilerden hangisidir?
    a) OpenFileDialog
    b) Label
    c) PictureBox
    d) TextBox
    e) ProgressBar

6. Kontrol aktifken bir tuşa basıp bırakınca kontrolün ele alınmasını sağlar?
    a) DragOver
    b) GotFocus
    c) KeyDown
    d) KeyPress
    e) KeyUp

7. Kontrol nesnelerinin arka plan rengini ayarlayan özellik aşağıdakilerden hangisidir?
    a) Anchor
    b) BackColor
    c) Cursor
    d) Dock
    e) Enabled

8. Kontrol nesnesinin görünürlük özelliğini ayarlamak için aşağıdakilerden hangisi kullanılır?
    a) Enabled
    b) TabStop
    c) Text
    d) TextAlign
    e) Visible

9. Aşağıdakilerden hangisi Containers sekmesinde yer alan kontrollerden biri değildir ?
    a) Panel
    b) SplitContainer
    c) ContextMenuStrip
    d) TabControl
    e) TableLayoutPanel

10. Kontrol nesnelerinin Enabled özelliği ile aşağıdakilerden hangisi yapılır?
    a) Kontrolün aktifliğini ayarlar.
    b) Formun bir kısmının kontrolle dolmasını ayarlar.
    c) Kontolün üzerindeyken yazı özelliklerini ayarlar.
    d) Kontrolün üzerindeki yazının rengini ayarlar.
    e) Kontrolün görünürlüğünü ayarlar.

Cevap Anahtarı

1.a,2.c,3.e,4.b,5.d,6.d,7.b,8.e,9.c,10.a

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Sharp, John, Visual C# 2012 Step by Step, Microsoft Press, ISBN 978-0-7356-6801-0, Washington, 2012.

Halvorson, Micheal, Visual Basic 2010 Step by Step, Microsoft Press, ISBN 978-0-7356-2669-0, Washington, 2010.

AKTAŞ, Volkan, Visual Studio 2010 ile Her Yönüyle C#, Kodlab, İstanbul, 2011.

MASTAR, Muhammed, VURAL, M. Tahir, ERİŞ, Süha, Visual Studio 2012, Kodlab, İstanbul, 2013.

GÜNGÖREN, Bora, JAVA ile Temel Programlama, Seçkin Yayınevi, İstanbul, 2003.

KESKİNKILIÇ, Mine, JAVA ile Programlama, Seçkin Yayınevi, Ankara, 1997.

DURSUN Hüseyin, Nesneye Yönelik Programlama ve C++, Pusula, İstanbul, 1996.

DEMİRALP Fehmi, C++ ile Nesneye Dayalı Programlama, Beta, İstanbul, 1993.

YENİMAN YILDIRIM, Ebru, Yüksekokul Öğrencileri için Uygulamalı Görsel Programlama Visual Basic, Nobel Yayın Dağıtım, İstanbul, 2007.

http://msdn.microsoft.com/

http://www.ismailgursoy.com.tr/calendar-kontrolu/

GİRİŞ VE MESAJ PENCERELERİ

İÇİNDEKİLER

- Kod Yazımı ve Kuralları
- Bilgi Girişi
- Bilgi Çıkışı ve Mesaj
- Örnek Program

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Form Designer ve Kod penceresinin kullanımını öğrenebicek,
  - Projenin solution oluşturarak kaydedilebilmesini bilebicek,
  - Kod yazım kurallarını ve açıklama satırı oluşturabilmesini görebicek,
  - Programa veri girişi için INPUTBOX, programdan veri çıkışı için MSGBOX komutlarının kullanımlarını gerçekleştirebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr.
Y. Ziya AYIK

ÜNİTE

4

GİRİŞ

Bütün programlama dillerinin ortak amacı bilgisayar ve program arasında haberleşmeyi sağlamaktır. 1 ve 0'lardan oluşan ve en alt programlama dili olarak kabul edilen makine dili ile program yazılması oldukça zor olduğu için zamanla konuşma diline yakın, belirli gramer ve söz dizimi kurallarına sahip yüksek seviye programlama dilleri geliştirilmiştir.

Bu ünitede Visual Studio .NET programa geliştirme ortamı altında çalışan Visual Basic .Net programlama dilinin temel özellikleri ve bu dil kullanılarak kod yazabilmek için bilinmesi gereken önemli noktalar üzerinde durulacaktır. Bir programlama dilini öğrenmek için ilk önce bu dilin çalışma mantığını ve de temel özelliklerini bilmek gerekmektedir. Bu yüzden bu ünitede anlatılan konuların özümsenmesi sonraki ünitelerdeki konuların anlaşılması açısından önem arz etmektedir.

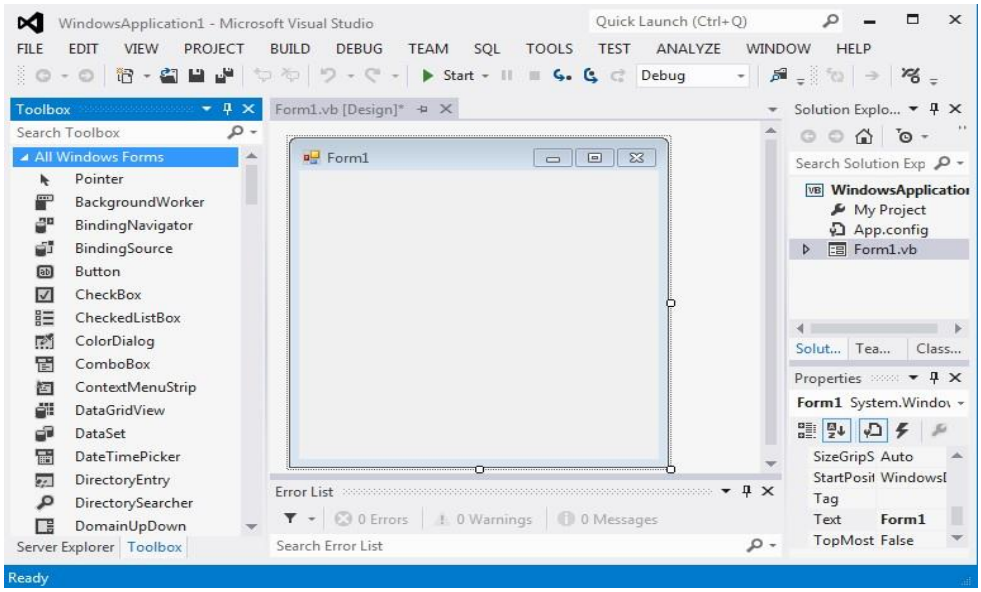
KOD YAZIMI VE KURALLARI

Kod yazım kurallarını genel ve programlama diline özel olmak üzere ikiye ayırmak gerekir. Genel kurallar bütün programlama dillerinde uyulması gereken ortak kurallar olarak tanımlanır. Bu ünitede sadece başlangıç seviyesindeki konulara değinileceği için şimdilik aşağıdaki genel kuralları bilmek yeterlidir:

- Kod yazarken değişken, sınıf ve metod isimlendirmeleri açıklayıcı ve anlaşılabilir olmalıdır.
- Belirli bir notasyon (yazım tarzı) benimsenmelidir.
- Yazılan kod yorum satırları ile desteklenmeli ve açıklayıcı yorumlar yazılmalıdır.
- Projedeki dosya ve sayfa isimlendirmelerinde özel karakterler kullanılmamalıdır.

Programlama diline özel kurallar ise sadece belirli bir programlama diline ait kurallar olarak tanımlanabilir. Bu ünitenin konusu çerçevesinde Visual Basic.Net diline dair bazı önemli kurallarda sırasıyla incelenecektir.

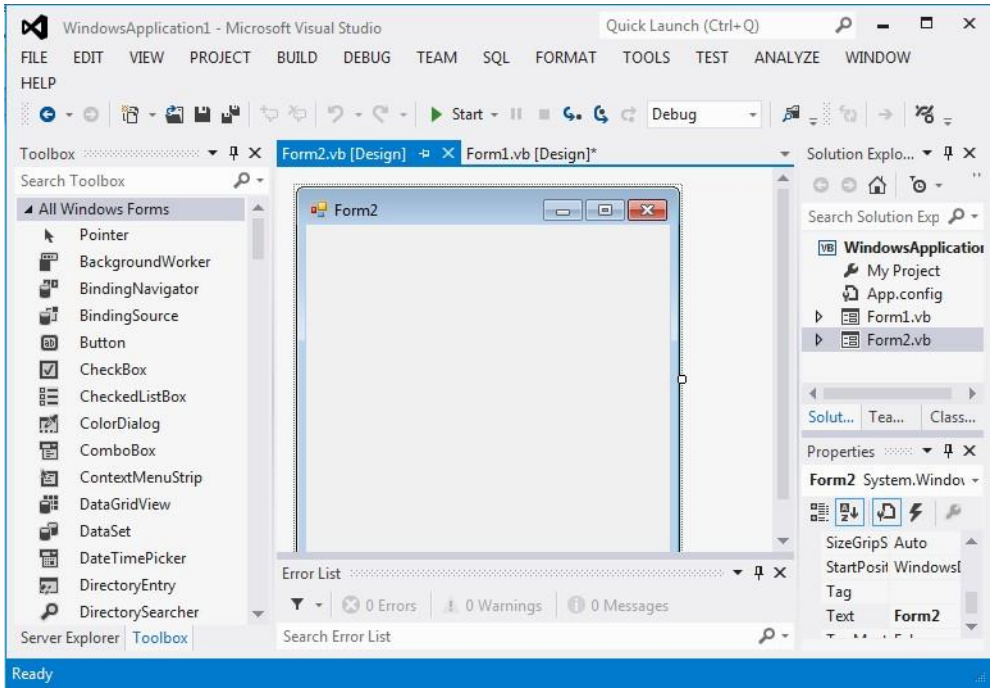
Visual Studio ile Windows tabanlı bir uygulama geliştirmek istendiğinde Form nesnesi kullanılır. Visual Basic .NET ile yeni bir proje oluşturulduğunda, projeye otomatik olarak dahil edilen form, Form Designer adı verilen alanda görüntülenmektedir. Oluşan bu form *Sysytem.Windows.Forms* adlı *namespace'deki* Form sınıfının mirasçısı olup bu *namespace'nin* özelliklerini taşımaktadır.



Şekil 4.1. Yeni oluşturulan bir proje ve form nesnesi

Projenin ikinci formdan başlatılması için Project menüsünün Properties seçeneği aktif hâle getirilip Startup Form menüsünden Form2 seçilir.

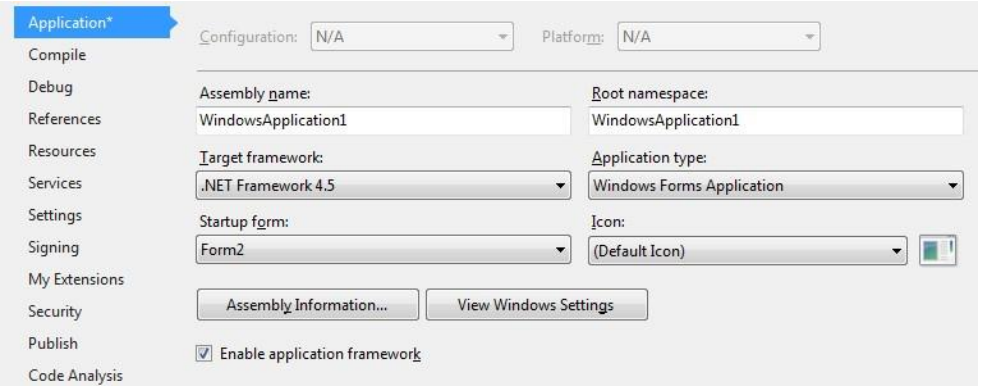
Projelere ikinci bir form ekleme söz konusu olabilir, bu durumda normal olarak ilk oluşturulan form aktif olarak görüntülenecektir. İkinci formu eklemek için Project menüsünden Add Windows Form seçilmelidir.



Şekil 4.2. Projeye 2. form eklenmesi

Ancak istenildiğinde oluşturulan ikinci formun aktif hale getirilmesi ve hatta programın ikinci formdan başlatılması sağlanabilir. Bunun için Project menüsünün Properties seçeneği aktif hâle getirilerek Startup Form menüsünden Form2 seçilmelidir.

Form tasarımı etkin olduğunda kod sayfasına geçmek için View menüsünden Code seçeneğine tıklanabilir.

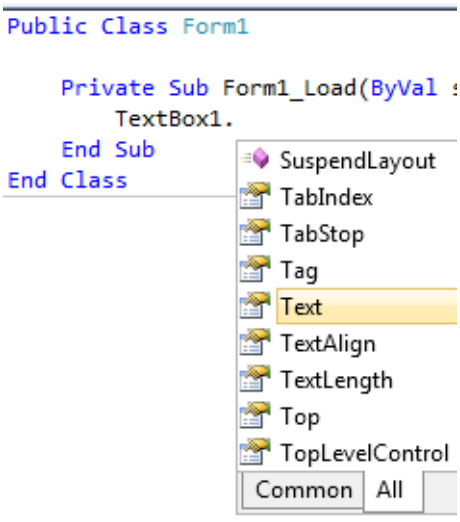


Şekil 4.3. Projenin 2. formdan başlatılması

Program kodları Editörde bulunan kod penceresine yazılabilirler. Formlardan herhangi birine ait kodlar veya projedeki bütün diğer nesneler dahil diğer tüm kodlamalar görüntülenmek isteniyorsa kod penceresine geçilmelidir.

Kod penceresinin aktif hale getirilebilmesi birkaç şekilde yapılabilir. Bunlardan birincisi *View* menüsündeki *Code* seçeneğini tıklamak, ikincisi ise klavyeden F7 tuşuna basmaktır. Üçüncü seçenek Solution Explorer penceresinden View Code düğmesini tıklamak iken dördüncü bir yol ise form nesnesini Mouse ile çift tıklamak olabilir.

Program kodları *Sub Main() ve End Sub* satırları arasında yazılmalıdır. Kod yazımında editör tarafından yönlendirme yapılabilmektedir. Herhangi bir nesne veya komut adı girildikten sonra "." tuşuna basıldığında o nesne veya komutla birlikte kullanılabilecek bütün parametrelerin listesi bir kutu içerinde görüntülenmektedir. Bu özelliğe *intelliSense* denilmektedir. Bu durum programcı için önemli bir destek sağlamaktadır. Şekil 1.5'te TextBox1 nesnesi ile birlikte kullanılabilecek parametreler listesi görüntülenmektedir. Programcı kullanacağı parametrenin ilk karakterini girdiğinde liste yeniden düzenlenecek ve girilen o harfle başlayan parametreler listelenecektir. İstenilen parametre mouse tıklanarak veya *Tab* tuşuyla aktif hâle getirilecektir.



Şekil 4.4. İntelliSense özelliği

Yorum(Açıklama) satırları, kesme sembolü veya Rem komutu kullanılarak oluşturulur

Kod penceresi görüntülenen formun ya da nesnenin görsel yapısını görüntülemek için View menüsünden Designer seçilmeli veya Solution Explorer penceresinden View Designer düğmesi tıklanmalıdır. Ayrıca kod penceresinden tasarım görünümüne klavyeden Shift+F7 tuşu kullanılarak da geçiş yapılabilmektedir.

Yapısal programlamada, program akışı, programın ilk satırından başlayarak aşağıya doğru gittiği için mantık baştan sona doğru gitmektedir. Ancak Class, Namespace ve Obje(nesne) özelliklerine sahip Visaual Basic .NET programlama dilinde program kodları private(özel) kullanılabilmekte ve her komut akışı kendi içinde değerlendirilebilmektedir. Ancak Private oluşturulan program kodlarının ortak kullanımı ve nesnelerin özelliklerinin diğer nesneler aktarılması da söz konusudur ve bu durum .NET programlama mantığının en önemli özelliklerinden biridir.

Sub Main() ve End Sub satırları arasında yazılması gereken program kodlarının yazımında bazı kurallara dikkat edilmelidir.

Kod Yazım Kuralları

- Her komut tek satıra yazılmalıdır.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Button1.Text = "GİRİŞ"
    TextBox1.Text = "Merhaba Visual Studio 2012"
End Sub
```

- Bir satıra birden çok komut yazılmak isteniyorsa iki komut arasına " : " konulmalıdır.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Button1.Text = "GİRİŞ" : TextBox1.Text = "Merhaba Visual Studio 2012"
End Sub
```

- Yazılan satırın açıklama satırı olarak kullanılması düşünülüyorsa satırın başına " ' "(kesme) işareti konulabilir.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    ' TextBox uygulaması
    TextBox1.Text = "Merhaba Visual Studio 2012"
End Sub
```

- Yazılan satırın açıklama satırı olarak kullanılması için ikinci yöntem satır başına REM komutunun yazılmasıdır.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    REM TextBox uygulaması
    TextBox1.Text = "Merhaba Visual Studio 2012"
End Sub
```

- Tek bir satıra sığmayan uzun bir komut ikinci satıra kaydırılırken komutun birinci satırındaki ifadenin sonuna " _" (boşluk tuşu + alt tire) konulmalıdır.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs
    PictureBox1.Image = System.Drawing.Image.FromFile _
     ("c:\Ziya\Resimlerim\Bahar.bmp")
End Sub
```

- Yine tek bir satıra sığmayan uzun bir komut ikinci satıra kaydırılırken komutta bulunan herhangi bir " , " (virgül) karakterinden itibaren kaydırma yapılmalıdır.

```
Private Sub Form1_Load(ByVal sender As System.Object,
                       ByVal e As System.EventArgs) Handles MyBase.Load
    PictureBox1.Image = System.Drawing.Image.FromFile("c:\Ziya\Resimlerim\Bahar.bmp"
End Sub
```

Programa bilgi girişi Inputbox metodu ile yapılır.

- Program yazarken, herhangi bir komutu yanlış yazarsanız komutun altı otomatik olarak çizilecektir. Bu, size komutun doğru yazılmasında yardımcı olacak önemli bir özelliktir.

```
Private Sub Form1_Load(ByVal sender As System.Object,
                       ByVal e As System.EventArgs) Handles MyBase.Load
    Pictureox1.Image = Sytem.Drawing.Image.FromFile("c:\Ziya\Resimlerim\Bahar.bmp"
End Sub
```

Bu örnekte PictureBox1 ve System ifadeleri yanlış yazıldığı için altları program tarafından otomatik olarak çizilmiştir.

BİLGİ GİRİŞİ

Visual Basic .Net Programlama dilinde programa dışarıdan veri girişi yapma işlemi INPUTBOX komutu ile yapılmaktadır. Aynı zamanda bir metot olan INPUTBOX komutu bir diyalog kutusu içerisinde kullanıcı tarafından programa bilgi girişi yapılması işlemini gerçekleştirmektedir. Bu diyalog kutusunda standart olarak *"OK" ve "Cancel"* düğmeleri bulunmaktadır. Veri girişleri bu penceredeki düğmelerden uygun olanı seçilerek yapılmaktadır. INPUTBOX komutunun genel kullanım şekli aşağıda verilmektedir. Bu kullanımda zorunlu olan parametrelerin yanı sıra kullanılması seçimlik olan parametreler de söz konusudur. Bu parametrelerden sadece bir tanesi veya birkaçı bir arada kullanılabilir.

Tek parametreli kullanım şekli:

```
INPUTBOX (Mesaj)
Mesaj: Girilmesi istenilen değer için açıklama veya soru
```

Örnek 1

- personel=InputBox(“Personel Sayısını Giriniz.”)

Çok parametreli kullanım şekli:

INPUTBOX (Mesaj, Pencere başlığı, Varsayılan değer, x, y)

Mesaj: Girilmesi istenilen değer için açıklama veya soru
Pencere başlığı: Açılacak pencerenin başlığı
Varsayılan değer: Giriş kutusunda başlangıçta bulunulması istenilen değer
x, y: Pencerenin sol üst köşesinin koordinatlarını belirtir.

Örnek 2

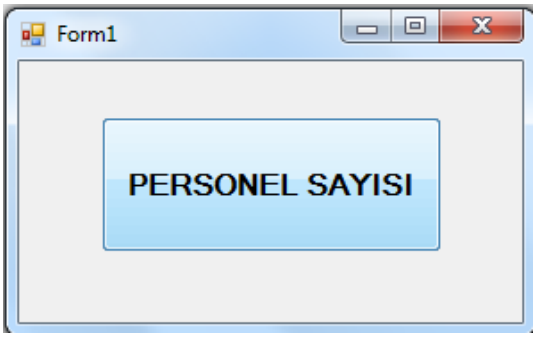
- personel=InputBox(“Personel Sayısını Giriniz.”, "Personel Sayısı", 157)

Örnek 2 için geliştirilen program kodu ve ekran çıktıları aşağıda verilmektedir.

Inputbox metodu ile birlikte, Mesaj, Pencere başlığı ve varsayılan değer parametreleri kullanılabilir.

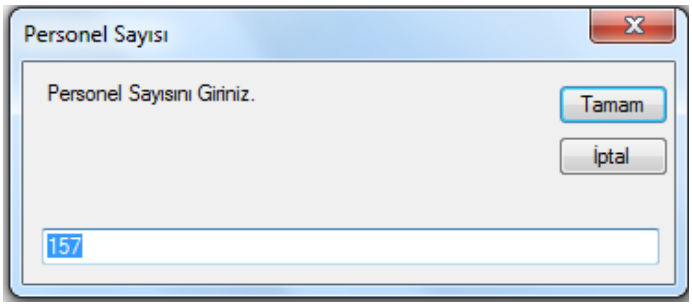
```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.Ev
        Dim personel As String
        personel = InputBox("Personel Sayısını Giriniz.", "Personel Sayısı", 157)
        MsgBox(personel)
    End Sub
End Class
```

Örnek 2 için geliştirilen program çalıştırıldığında Şekil 1.6’daki ekran görüntülenecek ve “Personel Sayısı” butonuna basılması beklenecektir.



Şekil 4.6. Başlangıç ekranı

Buton tıklandığında Şekil 1.7’deki Giriş Ekranı görüntülenecektir. Burada Personel sayısı program kodu ile önceden 157 verildiği için 157 sayısı görüntülenecektir. Bu rakam değiştirilmeden “Tamam” butonuna basıldığında sonuç ekranında 157 sayısı onaylanmış olacaktır.

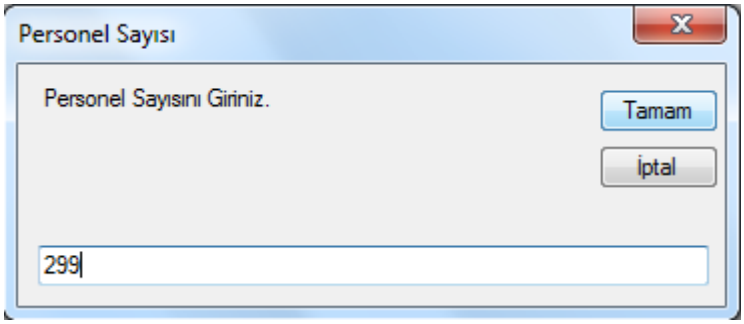


Şekil 4.7. Giriş ekranı

Şekil 4.8. Sonuç ekranı

Varsayılan değer programın çalışma anında farklı bir değerle değiştirilebilir.

Personel sayısı değiştirilerek "Tamam" butonuna basıldığında ise Şekil 1.9'da görüldüğü gibi girilen yeni personel sayısı görüntülenecektir.



Şekil 4.9. Giriş ekranı

Şekil 4.10. Sonuç ekranı

BİLGİ ÇIKIŞI VE MESAJ

Programdan bilgi çıkışı MessageBox nesnesi ile yapılır.

Veri ve bilgilerin bilgisayarda işlem gördükten sonra kullanıcı veya programcılara bir şekilde aktarılmaları gerekmektedir. Bu amaçla MessageBox nesnesi geliştirilmiştir. Mesaj kutusu program akışı sırasında kullanıcı veya programcıya bilgi vermek, işlem gören veri ve bilgilerin sonuçlarını iletmek ve sonuçların onaylatmak amacıyla kullanılmaktadır. MessageBox nesnesi MsgBox metodu çağrılarak görüntülenmektedir.

Kullanım Şekli:

MSGBOX(Mesaj, Tip, Pencere başlığı)

Mesaj: Yazılması istenilen açıklama, soru veya cevap

Tip: Pencerede yer alacak seçenekler, ikonlar, varsayılan değerler ve öncelikler
Not: Tip parametresinde kullanılacak seçenek değerleri 0,1,2,3,4,5; icon değerleri 16,32,48,64; varsayılan değerler 0,256,512 ve öncelik değerleri 4096 ve 0 olabilir.

Pencere başlığı: Açılacak pencerenin başlığı

Bu değerler ve anlamları şu şekildedir:

Seçenek		**İkon**	
0	vbOKOnly	16	vbCritical
1	vbOKCancel	32	vbQuestion
2	vbAbortRetryIgnore	48	vbExclamation
3	vbYesNoCancel	64	vbInformation
4	vbYesNo		
5	VbRetryCancel		
Varsayılan		**Öncelik**	
0	vbDefaultButton1	4096	System Modal
256	vbDefaultButton2	0	Normal
512	vbDefaultButton3		

Tip parametresi için istenilen sembolik ifade veya sembolün sayısal karşılığı kullanılabilir.

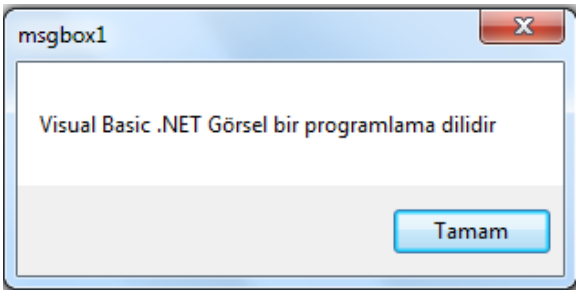
Pencere başlığı: Açılacak pencerenin başlığı

Helpfile, Context: Bir yardım dosyası ismi ve bir konu numarası belirlenerek, kullanıcının yardım istemesi durumunda ilgili help dosyasının açılması ve ilgili konunun görüntülenmesi sağlanır.

Bu parametrelerden sadece Mesaj parametresinin kullanılması zorunludur. Diğer parametrelerin kullanılması isteğe bağlıdır. Bir seferde bunlardan sadece biri veya bir kaçı aynı anda kullanılabilir.

Örnek 3

```
• MsgBox( "Visual Basic .NET Görsel bir programlama dilidir…")
```

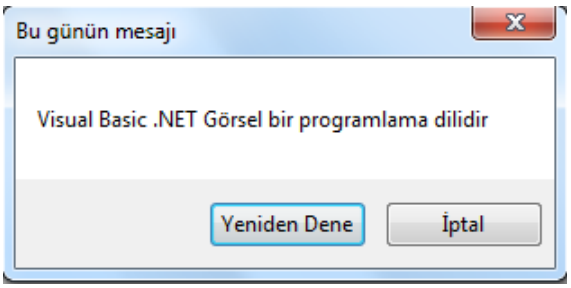


Şekil 4.11. Başlangıç ekranı

Örnek 3'de sadece Mesaj parametresi kullanılmıştır.

Örnek4

```
• Cevap=MsgBox("Visual Basic .NET Görsel bir programlama
  dilidir", vbRetryCancel, "Bu günün mesajı")
```



Şekil 4.12. Başlangıç ekranı

Örnek 4'de Mesaj, Tip(vbRetryCancel) ve Pencere başlığı parametreleri kullanılmıştır.

Bu örnekte MessageBox penceresinde görüntülenen Evet veya Hayır seçeneklerinden birine verilecek onay Cevap değişkenine değer olarak aktarılmaktadır. Bu sayede programdan çıkış yapılmakta veya çıkış iptal edilmektedir.

Örnek 5

```
• Cevap=MsgBox("Visual Basic .NET Görsel bir programlama
  dilidir", vbYesNo + vbQuestion, "Bu günün mesajı")
```

Örnek 5'te Tip seçeneği için (vbYesNo) Tip ikonu için de (vbQuestion) ifadeleri kullanılmıştır.

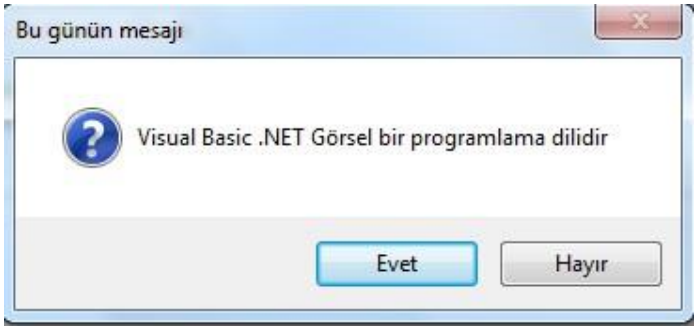
Örnek5 ve Örnek 6'da görüldüğü gibi, Tip seçeneği ve ikonu bir arada kullanıldığında ikisi arasında "+" sembolü kullanılmalıdır.

Örnek 6

```
•Cevap=MsgBox("Visual Basic .NET Görsel bir programlama
  dilidir", 1+ 32, "Bu günün mesajı")
```

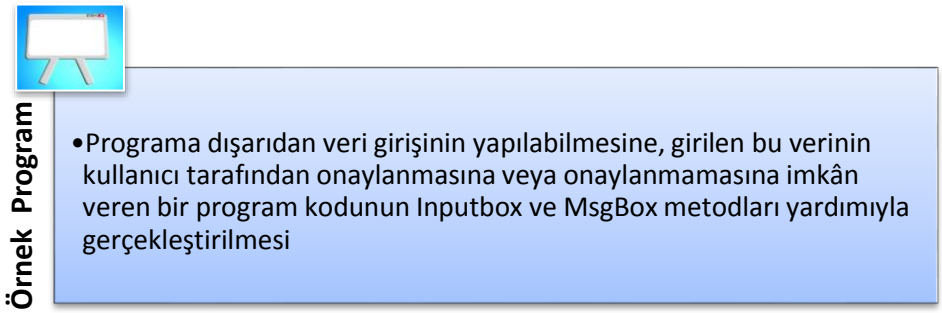
Örnek 6'da, Örnek 5' de kullanılan Tip seçeneği için (vbYesNo) Tip ikonu için de (vbQuestion) ifadeleri yerine bu ifadelere karşılık gelen sayısal değerler kullanılmıştır.

Örnek 5 ve örnek 6 kodlamalarının sonuçları aynıdır ve her ikisinin çalıştırılması sonucunda da aşağıda görüntülenen çıktı ekranı oluşur.

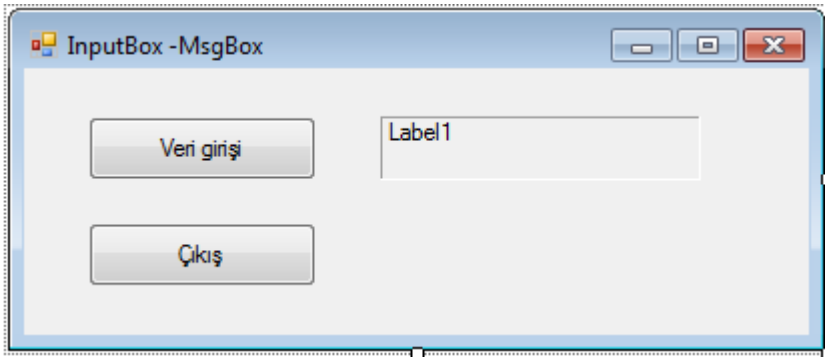


Şekil 4.13. Başlangıç ekranı

ÖRNEK PROGRAM



Form Tasarımı



Şekil 4.14. Başlangıç ekranı

Veri girişi kodları

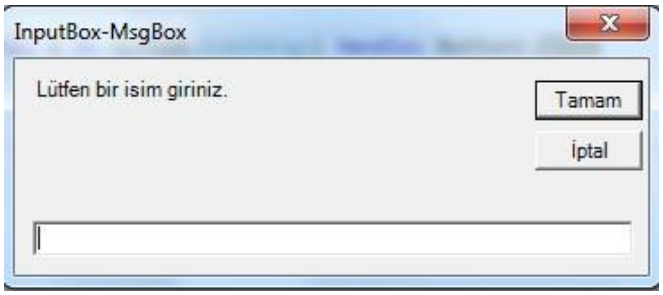
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
    Dim isim, tamad As String
    isim = "Lütfen bir isim giriniz."
    tamad = InputBox(isim, "InputBox-MsgBox")
    MsgBox(tamad, vbOKCancel + vbQuestion, "Girilen veriler....")
End Sub
```

INPUTBOX metodu bir değişkene atanarak kullanılmalıdır.

Çıkış kodları

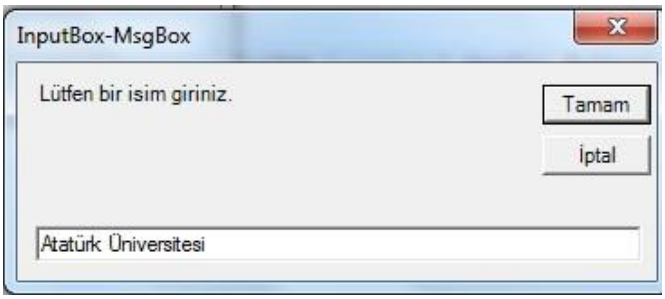
```
    Private Sub Button2_Click(ByVal sender As System.Object,
        End
    End Sub
```

Veri giriş ekranı



Şekil 4.15. Başlangıç Ekranı

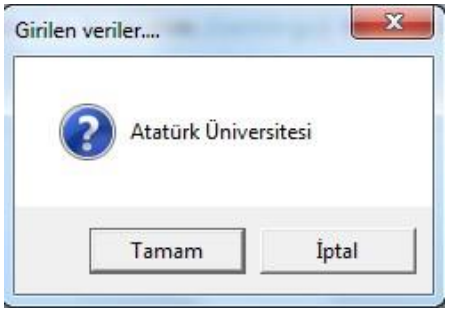
Veri giriş ekranına verinin girilmesi



Şekil 4.16. Başlangıç ekranı

"Atatürk Üniversitesi" ifadesi girilen veridir.

Veri giriş ekranında "Tamam" butonunun tıklanması sonucu oluşan sonuç ekranı şu şekildedir:



Şekil 4.17. Başlangıç ekranı

Veri giriş ekranında “İptal” butonunun tıklanması sonucu oluşan sonuç ekranı şu şekildedir:

Şekil 4.18. Başlangıç ekranı

Özet

•Visual Basi .Net projeleri “vbproj” uzantılıdır. Daha önceden oluşturulmuş bir proje, bu uzantıya sahip dosya seçilmek suretiyle açılabilir. Ayrıca eğer oluşturulmuş ise “sln” uzantılı solution dosyalardan da projelerin açılabilmesi mümkündür.

•Program kodları Sub Main() ve End Sub satırları arasında yazılmalıdır. Kod yazımında bazı kurallara dikkat edilmelidir. Bu kurallar ünite içerisinde anlatılmıştır.

•Programa bilgi girişi için INPUTBOX kullanılabilir. INPUTBOX kullanımında bazı parametreler veri giriş pencerelerinin farklı şekillerde görünümünü sağlar.

•Bilgi çıkışı için MSGBOX komutu kullanılmalıdır. MSGBOX'ın kullanımındaki "Tip" parametreleri de çıkış ekranlarının farklı özelliklerde görünümlerine neden olmaktadır.

DEĞERLENDİRME SORULARI

1. Visual Basic .Net projelerinin uzantısı aşağıdakilerden hangisidir?
   a) İnfo
   b) cs
   c) congig
   d) vbproj
   e) vb

2. Aşağıdaki veri giriş-çıkış kodlamalarından hangisi doğru yazılmıştır?
   a) Adınız = InputBox(Adınızı Girin)
      MsgBox(İsim Giriş Menüsü, , "Adınız")
   b) InputBox(Adınızı Girin)=Adınız
      MsgBox(Adınız, , "İsim Giriş Menüsü")
   c) Adınız = InputBox(Adınızı Girin)
      MsgBox(Adınız, , "İsim Giriş Menüsü")
   d) MsgBox= InputBox(Adınızı Girin, , "İsim Giriş Menüsü")
   e) Adınız = InputBox(İsim Giriş Menüsü)
      MsgBox(Adınız, , "Adınızı Girin")

3. Seçeneklerde belirtilen kodlamalardan hangisi doğrudur?
   a) Dim ‘ isim, tamad As String
   b) Rem MsgBox(tamad, , "Girilen isimler ekranı")
   c) isim = "Lütfen adınızı girin."
   d) “ ‘tamad = InputBox(isim)”
   e) “ Rem isim = "Lütfen adınızı girin."

4. MsgBox(“XXX”, ,”YYY”) işlemi için aşağıdakilerden hangisi doğrudur?
   a) YYY değişkendir.
   b) Bu ifade ekrana bilgi girişi sağlar.
   c) YYY pencere başlığıdır.
   d) XXX pencere başlığıdır.
   e) İfade hatalıdır.

5. Aşağıdaki veri giriş kodlarından hangisi doğrudur?

   a) KullaniciAdi=InputBox(“İsminizi giriniz”)
   b) KullaniciAdi=MsgBox(“İsminizi giriniz”)
   c) KullaniciAdi=Input(“İsminizi giriniz”)
   d) InputBox(“İsminizi giriniz”)=KullaniciAdi
   e) MsgtBox(“İsminizi giriniz”)=KullaniciAdi

6. Aşağıdaki kodlamalardan hangisi doğrudur?

   a) isim, tamad As String
   b) MsgBox, tamad
   c) isim "Lütfen adınızı girin."
   d) tamad = InputBox(isim)
   e) “ Rem isim = "Lütfen adınızı girin."”

7. Aşağıda verilen kodlamalardan hangisi yazım kurallarına uygun yazılmıştır?

   a) TextBox1.Load = "Merhaba Arkadaşlar"
   b) Text.TextBox1. = "Merhaba Arkadaşlar"
   c) TextBox1.Text = Merhaba Arkadaşlar
   d) TextBox1.Text "Merhaba Arkadaşlar"
   e) TextBox1.Text = "Merhaba Arkadaşlar"

8. Bu kodlamada 157 değeri genel tanımlamada hangi ifade yerine kullanılmıştır?
   personel=InputBox(“Personel Sayısını Giriniz.”, "Personel Sayısı", 157)

   a) Pencere başlığı
   b) Mesaj
   c) Varsayılan değer
   d) Seçenek
   e) İkon

9. Aşağıda verilen kodlamada altı çizili ve bold olarak gösterilen 1 değerinin yerine aşağıdakilerden hangisi kullanılabilir?
   MsgBox("Visual Basic .NET Görsel bir programlama dilidir", **1** + 32, "Bu günün mesajı")

   a) vbOkOnly
   b) vbOkCancel
   c) vbAbortRetryIgnore
   d) vbYesNoCancel
   e) vbYesNo

10. Aşağıdaki kodlardan hangisi doğru yazılmıştır?

    a) Sicaklik=INPUTBOX (“Hava sıcaklığını giriniz”)
    b) Sicaklik= MSGBOX(Sıcaklık,20)
    c) Uzunuluk=25 Metre
    d) PRINT “Uzunluk Giriniz….”;20
    e) INPUT “Girişler INPUTBOX ile yapılacaktır….”; INPUTBOX

Cevap Anahtarı

1.d,2.c,3.b,4.c,5.a,6.d,7.e,8.c,9.ł

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Ayık Y. Ziya, (2009), Algoritma ve Programlama Metodolojisi, MuratHan, Trabzon

Ayık Y. Ziya, (2011), Atatürk Üniversitesi, Uzaktan Eğitim Merkezi, Görsel Programlama I Ders Notları

Yanık Memik(2010), Visual Basic 10, Seçkin Yayınevi, Ankara

DEĞİŞKEN TANIMLAMALARI

Atatürk Üniversitesi
Açıköğretim Fakültesi

İÇİNDEKİLER

- Basit Değişkenler
- Değişken Tanımlama
- Değişkenler İlk Değer Atama
- Değişkenler ve Özellikleri
- Dizi Değişkenler
- Sabitler

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Değişken ve Sabit kavramlarını bilecek,
  - Değişken ve Sabitleri tanımlayabilecek,
  - Farklı türlerdeki değişkenlerin özelliklerini bilecek,
  - Basit ve dizin değişken kavramlarını öğrenebileceksiniz.

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr.
Y. Ziya AYIK

ÜNİTE
5

GİRİŞ

Değişkenler veri ve bilgilerin saklanması için kullanılan hafıza birimleridir. Program süresinde değişkenlerin değerleri sürekli değişebilir. Değişkene ilk olarak atanan veri veya bilgi, değişkenin herhangi bir matematiksel veya mantıksal işleme tabii tutulması sonucunda farklı bir veri veya bilgi ile yer değiştirebilir.

Değişken tanımı Dim komutu kullanılarak yapılabilir.

Değişkenler tuttukları veri türüne göre basit ve dizi (indisli) değişkenler olmak üzere iki grupta incelenebilirler. Basit değişkenler aynı anda tek bir veri veya bilgi taşıyabilen değişkenlerdir. Yeni bir değer taşıyabilmeleri için daha önce taşıdıkları veri veya bilgiyi bırakmış olmaları gereklidir. Dizi değişkenler ise indis değerlerine göre birden çok veri veya bilgisi aynı anda taşıyabilirler. Yani, eleman sayıları kadar veri veya bilgi taşırlar. Tek boyutlu veya birden çok boyutlu olabilirler.

Bu ünitede değişkenlerin kullanımı ve değişken türleri detaylı olarak incelenecektir.

BASİT DEĞİŞKENLER

Bir değişken, programınız içinde geçici bir veri depolama yeridir. Kodlama yapılırken sözcük, sayı, tarih ya da diğer özellikleri içeren bir ya da birden çok değişken kullanma ihtiyacı oluşabilir. Değişkenler, çalışmada planlanan her veri parçası için kısa ve kolay hatırlanabilen bir isim verilerek kullanılabileceğinden birçok avantaj sağlar. Veri veya bilginin kendisiyle her seferinde işlem yapabilmek mümkün değildir. Bu nedenle bu veri veya bilgilerin saklandığı hafıza birimleriyle işlem yapmak programlamada önemli bir etkinlik sağlayacaktır. Değişkenler programın çalışması sırasında girilen bilgileri, belirli bir hesaplama sonucunu elde edilen bilgileri veya çıktıları depolayabilirler. Kısaca, değişkenler hemen her türlü bilgiyi depolamak ve izlemek için kullanabileceğiniz yararlı kaplardır.

Bütün bu sebeplerden ötürü değişkenleri kullanmayı bilmek çok önemlidir. Bu başlık altında değişken tanımlama, değer atama ve değişkenlerin temel özellikleri konularına değinilecektir.

Değişken Tanımlama

Bir değişken tanımlanırken değişken adı ve değişken tipi belirtilir.

Birçok programlama dilinde değişkenler kullanılmaya başlamadan önce tanımlanırlar. Visual Studio.Net platformunun desteklemiş olduğu bütün programlama dillerinde de değişkenler kullanılmadan önce tanımlanmalıdır.

Visual Basic .NET'te bir değişkenin tanımlanması için *Dim* (Dim boyut anlamındadır) ifadesinden sonra değişkenin adı yazılmalıdır. Bu tanımlama ile program çalıştığında değişken için bellekte yer ayrılır ve Visual Basic' in daha sonra karşılaşacağı veri türünün ne olduğunun bilinmesi sağlanır. Bu bildirim değişkenin kullanımından önce olmak kaydıyla programın herhangi bir yerinde yapılabilse de genellikle değişken bildirimleri olay yordamlarından ya da kod modüllerinden önce yapılırlar.

Visual Studio .NET ortamında kullanılan programlama dillerinin hepsinde aynı veri tipleri kullanılmasına rağmen, değişkenlerin tanımlanma biçimlerinde farklılıklar vardır. Örneğin Visual Basic .NET'de bit tamsayı değişkenin tanımlanması "Dim sayi as Integer" şeklinde yapılırken Visual C#.NET'de "int sayi;" şeklinde yazılmaktadır.

Visual Basic .NET programlama dilinde değişkenler *Dim* deyimi kullanılarak tanımlanır. Dim deyimi kullanılarak bir değişken tanımının genel yapısı ve uygun örnekler şu şekilde gösterilebilir.

```
Dim değişkenismi As tipi
```

```
Dim K As Integer
Dim Not1,Not2 As Integer
Dim Vize,Final As Integer, Ortalama As double
```

Örneklerde görüldüğü gibi Dim deyimi kullanılarak tek bir değişken tanımlanabileceği gibi değişken isimleri virgüllerle ayrılarak birden çok değişken tanımlanması da mümkündür. Hatta bir Dim deyimi ile farklı tiplerde değişken tanımlamak da mümkündür.

Değişken tanımlamada dikkat edilmesi gereken kurallar özetle şu şekildedir:

- Her değişkenin adı bir harfle başlamalıdır. Değişken adları yalnız harfler, sayılar ve alt çizgiler içerebilir.
- Değişken ismi en fazla 255 karakter olabilir. Bu yüzden değişken adları çok uzun tutulmamalıdır.
- Değişken adları anlamlı olmalıdır. Taşıyacağı veriyi hatırlatıcı bir isim olmasına dikkat edilmelidir.
- Değişken adları olarak Visual Basic anahtar sözcükleri, nesneleri ya da özellikleri kullanılmamalıdır.

Aşağıda doğru ve hatalı değişken isimlendirme örnekleri verilmiştir.

Doğru değişken ismi	Hatalı değişken ismi
XYZ	X Y Z
SINIF3	3SINIF
Sonuç	MDGBOX
Futbol	Futbol?

Değişkenlere İlk Değer Atamak

Değişken tanımı yapılırken değişken isimlendirme kurallarına dikkat edilmelidir.

Bir değişkene ilk değer ataması değişken tanımlanırken yapılabileceği gibi tanımlama işlemi tamamlandıktan sonra da yapılabilir.

Bir değişken tanımlanırken, tanımlanan değişkene ilk değer atama işlemi aşağıda gösterildiği gibi yapılmaktadır.

```
Dim K As Integer=25
Dim X As Integer=500, Unvan As String=”Atatürk Üniversitesi”
```

Örnekte görüldüğü üzere değişkenlere değer atamak için = operatörü kullanılmaktadır. Eşitliği sağ tarafında bulunan değer sol tarafta bulunan değişkene atanır. Bu örnekte değişken tanımlandığı anda ilk değer ataması yapılmıştır.

Bir değişkene ilk değer ataması değişken tanımlandıktan sonra yapılmak istenebilir. Bu durumda değer atamanın nasıl yapılacağı aşağıdaki örnekte gösterilmiştir.

```
Dim K As Integer
K=25

Dim X As Integer, Unvan As String
X=500
Unvan=”Atatürk Üniversitesi”
```

Değişkenler ve Özellikleri

Değişkene ilk değer atama işlemi tanımlama sırasında veya daha sonra yapılabilir.

Bütün programlama dillerinde, değişkenler belirli kurallara uygun olarak tanımlanırlar ve kullanılırlar. Bu değişkenlerin ve taşıdıkları değer olan verilerin tiplerinin belirlenmesi gerekmektedir. Veri tipinin doğru belirlenmesi geliştirilen programın hatasız, hızlı ve daha etkin çalışmasını sağlar. Veri tipinin belirlenmemesi veya uygun olmayan bir tipin tanımlanması programın hızının yavaşlamasına ve verilerin bellekte daha fazla yer kaplamasına neden olmaktadır. Bazen bütün veriler için kullanılabilen ve aktarılan verinin yapısına bakarak uygun özellikler alan örneğin Object tipi değişkenler, diğer tiplere gerek olmadan değişkenlerin normal işlevlerini yerine getirebilmekte ancak bu durum özellikle hızlı ve özlü kodlar hazırlanmak istendiğinde uygun olmamaktadır. Bu nedenle, verinin yapısının hassas bir şekilde dikkate alınarak uygun değişken tiplerinin oluşturulması, geliştirilecek programın etkinliği açısından çok önemlidir.

Visual Studio.Net içerisindeki bütün veri tipleri Ortak Tip Sistemi(Common Type System) adı verilen bir sistem ile tanınmaktadır. Ortak Tip Sistemi sayesinde .N et içerisinde herhangi bir programlama dilinde aynı veri tipleri kullanılır. Bu programlama dilleri farklı sentaksları ve yazım kurallarına sahip olsalar da ort

veri tiplerini kullanırlar. Aşağıda Visual Basic ve C# dillerinde Integer tipinde bir değişkenin tanımı gösterilmiştir.

Visual Basic'de	C#'t a
Dim K As Integer=0	int K=0;

Örnekte görüldüğü üzere K isimli "0" değerine sahip bir değişken tanımlanmıştır. Aralarındaki fark sadece yazım şekli ve sentakstı. Bellekte aynı boyutta yer kaplarlar. Kısacası Ortak Tip Sistemi sayesinde .Net içerisindeki diller arasında veri tipleri noktasında uyum sağlanır ve rahatlıkla bir Visual Basic.Net uygulamasında C# kütüphane dosyaları kullanılabilir.

Common Type System (Ortak Tip Sistemi) özelliği sayesinde Studio .Net ortamı altında kullanılan değişkenlerin diller arasındaki farkını ortaya çıkarabilmek için oluşturulan bir yöntem ise değişkenler için takma adların kullanılmasıdır. Studio .NET uyumlu her programlama dili Common Type System'de tanımlı değişken türlerine kendine göre bir takma ad vermektedir. Aşağıda temel veri tiplerinin Visual Basic .NET'deki adı ve CTS(Common Type System)' deki karşılığı tablo hâlinde verilmiştir.

Visual Basic'de Kullanılan Değişkenlerin T	
CTS Karşılığı	**Visual Basic .NET**
System.Byte	Byte
System.Int16	Short
System.Int32	Integer
System.Int64	Long
System.Single	Single
System.Double	Double
System.Object	Object
System.Char	Char
System.String	String
System.Decimal	Decimal
System.Boolean	Boolean

Char Değişkenler

Char veri tipindeki değişkenler evrensel bir karakter kod olan Unicode standartlarındaki karakterleri temsil etmek için kullanılırlar. Char veri tipinde bir karakter tanımlamak için aşağıda verilen yapı kullanılır:

```
Dim Harf As Char='A'
```

String Değişkenler

String değişkenler karakter veriler için uygundur.

Birçok programlama dilinde olduğu gibi Visual Basic.Net dilinde de metinsel ifadeleri içerecek bir veri tipi mevcuttur. Bu veri tipi *String* olarak adlandırılır. String veri tipinin kullanım amacı matematiksel işlemler uygulanmayacak, karakter özellikli verilerden istenilen uzunlukta saklamaktır. String veri tipi Char veri tipinden farklı olarak sadece bir karakter verinin değil birden çok karakterden oluşabilen bir dizi verinin aynı anda saklanabilmesini sağlar.

```
Dim isim As String
isim=”Bengisu”
```

String tipinde bir değişkene atanacak değer çift tırnak (“ ”) içerisine yazılmalıdır. Ayrıca değişkenler tanımlanırken değişken adının son karakteri olarak “$” sembolü kullanıldığında da tanımlanan o değişkenin tipi String olur.

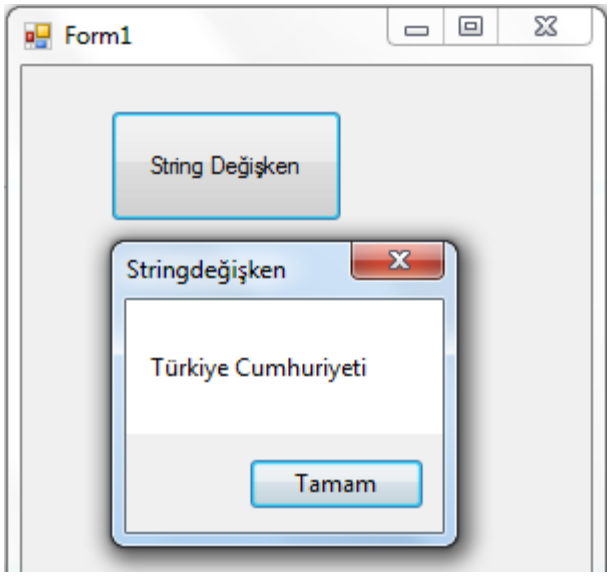
```
Dim isim$
isim=”Bengisu”
```

Örnek 1

- İki ismi String değişken kullanarak birleştirip yazdıran program.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim Ad1$
    Dim Ad2 As String
    Ad1 = "Türkiye"
    Ad2 = "Cumhuriyeti"
    MsgBox(Ad1 + " " + Ad2)
End Sub
```

String değişkenlerin birleştirme işlemi programda da görüldüğü gibi “+” sembolü kullanılarak yapılabilmektedir.



Resim 5.1 Birleştirilmiş değişken değerinin messagebox'ta görüntülenmesi

Boolean değişkenler Evet veya Hayır değerlerini taşırlar.

Boolean Değişkenler

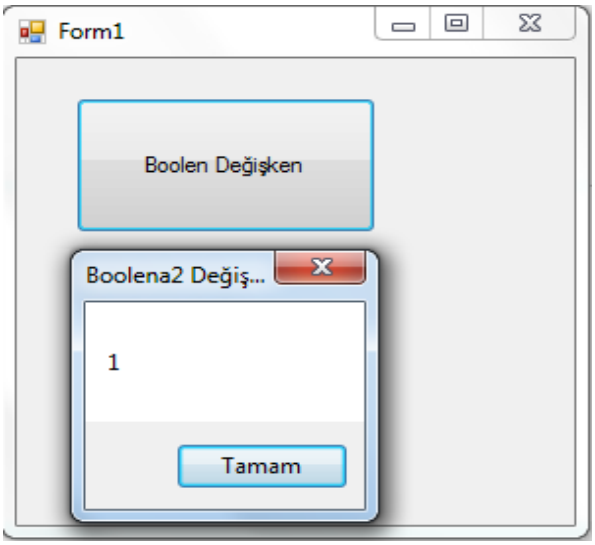
Sadece True (Doğru) ve False (Yanlış) değerlerinin saklanmak istenildiği durumlarda Boolean değişkenler tanımlanmalıdır. Boolean tipinde değişkenler sadece True ya da False değerini alabilirler. Hemen hemen tüm programlama dillerinde Boolean veri tipi büyük önem taşımaktadır. Özellikle program içerisindeki döngülerde ve mantıksal ifadelerde Boolean veri tipi sıkça kullanılır. Boolean tipi değişken tanımlanması, genel değişken tanımlanması kuralına göre yapılmaktadır.

```
Dim karar As Boolean
karar=True
```

Örnek 2

- Boolean değişken tipine aktarılan değerin Integer tipine dönüştürülerek elde edilen karşılığını bulan program.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim durum As Boolean
    Dim sayısal As Integer
    durum = True
    sayısal = Convert.ToUInt16(durum)
    MsgBox(sayısal)
End Sub
```

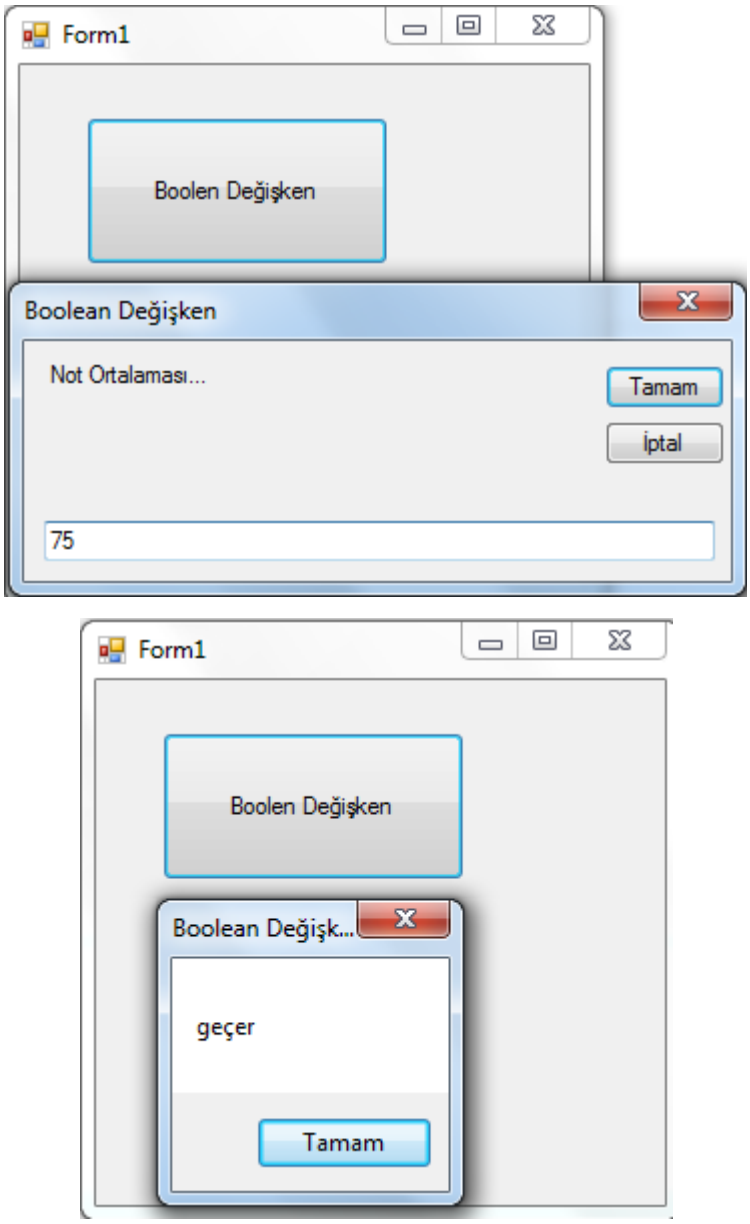


Resim 5.2 Boolen değerin sayıya çevrilip messagebox'ta görüntülenmesi

Örnek 3

- Boolean değişkeninin alacağı değere göre sonuç üretilebilen bir program.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim durum As Boolean
    Dim ortalama As Integer
    ortalama = InputBox("Not Ortalaması...")
    durum = ortalama >= 50
    If durum = True Then
        MsgBox("geçer")
    Else
        MsgBox("geçmez")
    End If
End Sub
```



Resim 5.3. Not ortalamasının hesaplanarak sonucun görüntülenmesi

Integer, Short, Byte ve Long Değişkenler

Tam sayı değerlerin değişkenlerde saklanmak istendiği durumlarda Integer, Short, Byte ve Long değişken tipleri kullanılmalıdır.

Integer değişkenler için bellekte 4 Byte, Short değişkenler için 2 Byte ve Long değişkenler için de 8 Byte yer ayrılır. Byte değişken için de 1 Byte yer gerekmektedir.

Integer değişkenler -2.147.483.648 ile 2.147.483.647 arası, Short değişkenler -32768 ile 32767 arası, Byte değişkenler 0 ile 255 arsı ve Long değişkenler ise -9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arası değişebilen değerleri alabilirler.

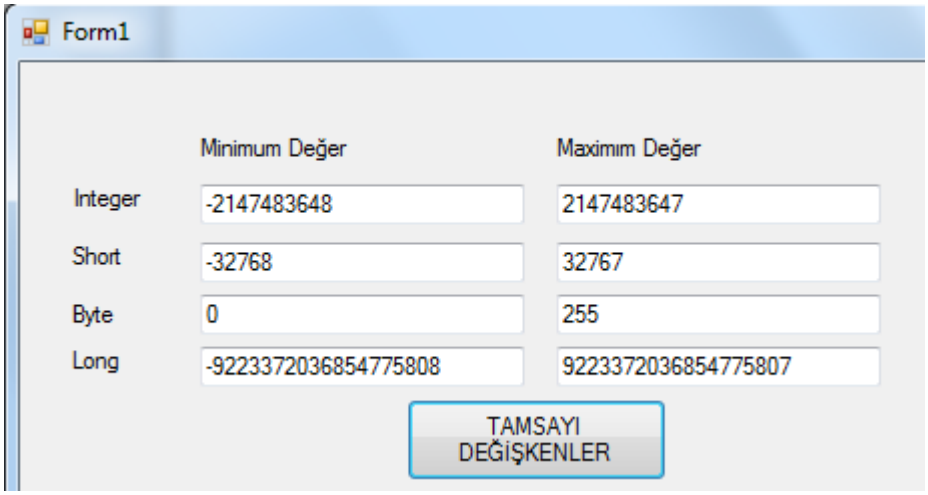
```
Dim sayısal As Integer
Dim sayısal As Short
Dim sayısal As Byte
Dim sayısal As Long
```

Örnek 4

•Tamsayı değişkenlerin alabileceği minumum ve maximum değerlerin görüntülendiği program.

Integer tipindeki değişkenin takma adı Int32'dir.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    TextBox1.Text = System.Int32.MinValue   'Integer
    TextBox2.Text = System.Int32.MaxValue   'Integer
    TextBox3.Text = System.Int16.MinValue   'Short
    TextBox4.Text = System.Int16.MaxValue   'Short
    TextBox5.Text = System.Byte.MinValue    'Byte
    TextBox6.Text = System.Byte.MaxValue    'Byte
    TextBox7.Text = System.Int64.MinValue   'Long
    TextBox8.Text = System.Int64.MaxValue   'Long
End Sub
```



Resim 5.4. Tamsayı değişken tiplerinin olacağı maximum ve minimum değerler

Single ve Double Değişkenler

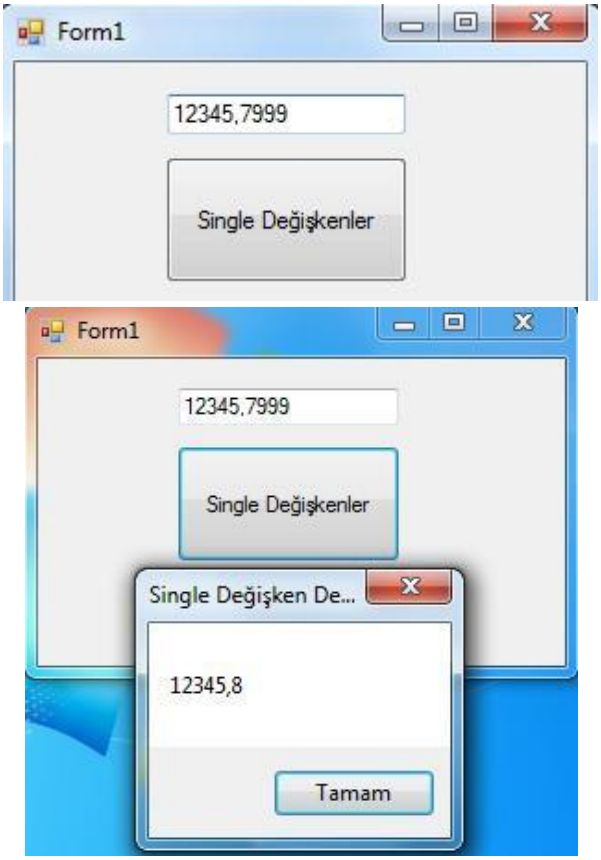
Ondalık değer içeren sayısal verilerin saklanmasın da Single veya Double değişkenler kullanılabilir. Single değişkenler bellekte 4 Byte, Double değişkenler ise 8 Byte yer kaplarlar. Ondalık değer içeren sayısal bilgi en fazla yedi anlamlı basmaktan oluşuyorsa Single, en fazla 15 anlamlı basamaktan oluşuyorsa Decimal tanımlanmalıdır.

```
Dim sayısal_noktalı1 As Single= 77,1234
Dim sayısal_noktalı2 As Double= 77,123456789012
```

Örnek 5

•Single bir değişkene 7 den fazla anlamlı basamaktan oluşan bir verinin aktarılması.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim kesirli As Single
    kesirli = TextBox1.Text
    MsgBox(kesirli, , "Single Değişken Değeri")
End Sub
```



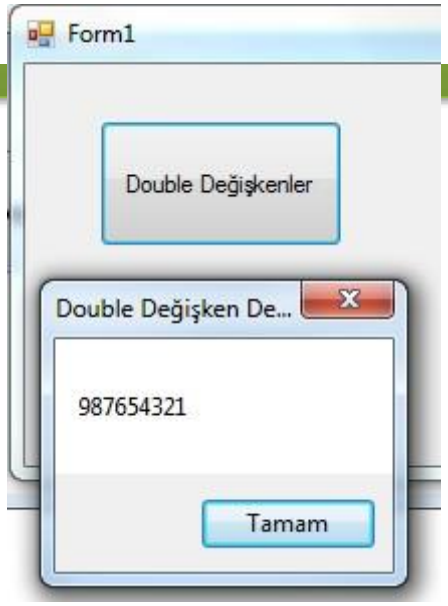
Resim 5.5. Single değişkenin messagebox'ta yuvarlanarak görüntülenmesi

Single tipi değişkenler en fazla 7 anlamlı basamaktan oluşurlar.

Örnek 6

•Bir tamsayı bilginin Double bir değişkene aktarılması. Bu işlem sonucunda hata oluşmaz. Sadece hafızada fazla yer işgal edilmiş olur.

```
Private Sub Button2_Click(ByVal sender As System.Object,
    Dim kesirli2 As Double
    Dim sayısal As Integer = 987654321
    kesirli2 = sayısal
    MsgBox(kesirli2, , "Double Değişken Değeri")

End Sub
```



Resim 5.6. Double değişkeninin görüntülenmesi

Decimal Değişkenler

Decimal değişkenler sayısal özellikteki bütün verileri saklayabilirler.

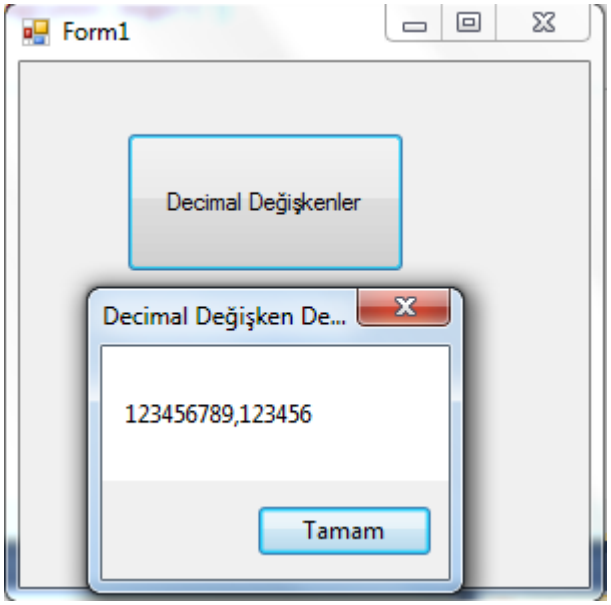
Visual Basic.Net'te bulunan veri tiplerinin her birinin kendine özgü sınırlılıkları bulunmaktadır. Özellikle mali hesaplamalarda kullanımı tercih edilen Decimal tipi değişkenler, yuvarlama hatalarını ortadan kaldırır ve sayıları 28 basamağa kadar içerebilir. Decimal veri tipi bir değişken şu şekilde tanımlanır:

```
Dim Hesap As Decimal
```

Örnek 7

- Decimal özellikli verilerin bu özellikteki değişkenlere aktarılması.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim hesap As Decimal
    hesap = 123456789.123456
    MsgBox(hesap, , "Decimal Değişken Değeri")
End Sub
```



Resim 5.7. Decimal değişkenin messagebox'ta görüntülenmesi

Object Değişkenler

Object veri tipinde bir değişken içerisinde her tip veri barınabilmektedir.

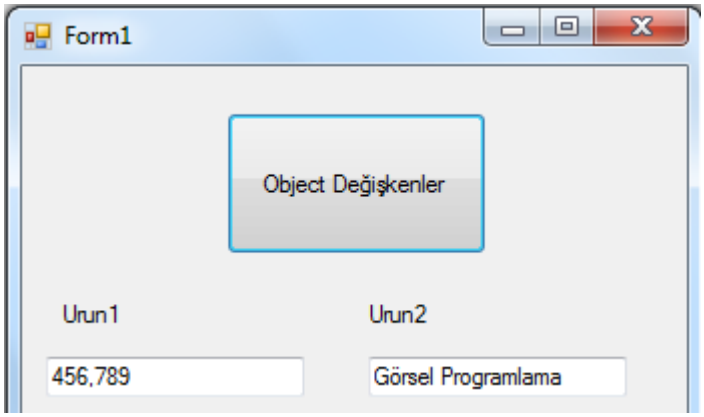
En genel tanımıyla bütün veri tipleri Object sınıfından türer. Bu yüzden *object* veri tipindeki değişkenler bütün tiplerdeki veriyi hatasız şekilde kabul ederler. Aşağıda object tipinde bir değişkenin tanımlanması ve farklı tipte verilerin bu değişkene atanması ile ilgili bir örnek görülmektedir:

```
Dim malzemeler As Object
```

Örnek 8

•Farklı özellikteki verilerin Object tipinde tanımlanması.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim urun1, urun2 As Object
    urun1 = 456.789
    urun2 = "Görsel Programlama"
    TextBox1.Text = urun1
    TextBox2.Text = urun2
End Sub
```



Resim 5.8. Örnek 8'deki kodların çalıştırıldığında oluşan ekran görüntüsü

DİZİ DEĞİŞKENLER

Dizi değişkenler indis değerleri kadar veriyi aynı anda saklayabilen değişkenlerdir.

Aynı anda birden çok değer taşıyabilen değişkenlere Dizi değişken denir. Dizi değişken tanımlanırken değişken adından sonra parantez içerisinde eleman sayısı belirtilerek dizi değişkenin aynı anda kaç veriyi saklayabileceği belirlenebilir.

Dizi değişkenler tek boyutlu veya çok boyutlu olabilirler. Tek boyutlu diziler tanımlanırken değişken isminden sonra parantez içerisinde dizinin eleman sayısını gösteren tek bir değer girilir. Çok boyutlu dizilerde ise dizinin isminden sonra parantez içerisinde dizi boyut sayı kadar farklı eleman değer virgüllerle ayrılarak girilmelidir.

Tek Boyutlu Dizi Tanımlaması

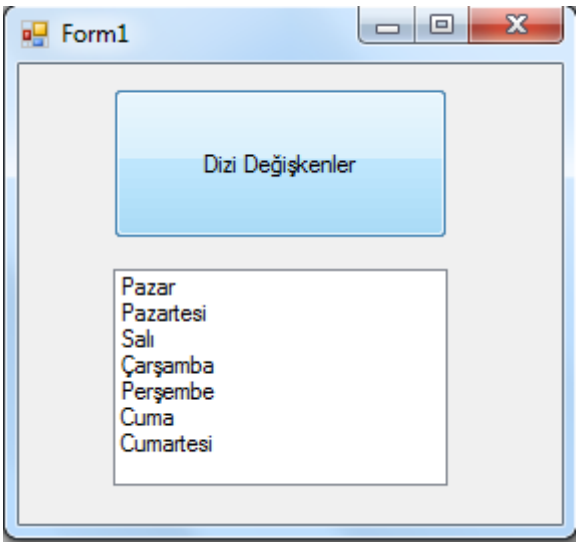
```
Dim Nesne(4) As String
```

Nesne adlı değişken bir boyutlu ve 5 elemanlıdır. Dizi değişkenlerde indis değerleri "0"dan başladığı için bu dizinin eleman sayısı 5'dir.

Çok Boyutlu Dizi Tanımlaması

```
Dim Matris(4,6) As String
```

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim gunler(6) As String
    gunler(0) = "Pazar"
    gunler(1) = "Pazartesi"
    gunler(2) = "Salı"
    gunler(3) = "Çarşamba"
    gunler(4) = "Perşembe"
    gunler(5) = "Cuma"
    gunler(6) = "Cumartesi"
    For i As Integer = 0 To gunler.GetUpperBound(0)
        ListBox1.Items.Add(gunler(i))
    Next
End Sub
```



Resim 5.9. Listbox'a eklenen haftanın günleri

Dizi değişkenler tek boyutlu veya çok boyutlu olabilirler.

SABİTLER

Değişkenler program boyunca aynı değeri koruyabildiği gibi, gereksinmelerinize göre değerlerini birkaç kez değiştirebilirler. Bazı durumlarda değişken değerlerinin program boyunca değiştirilmesi arzu edilmeyebilir. Bu gibi durumlarda sabit verileri tanımlanır. Sabit verileri tanımlamak için *const* anahtar sözcüğü kullanılır. Örneğin Oda sıcaklığı değerini birçok yerde kullanan bir programda, bu değeri sabit olarak tanımlamak mantıklı olacaktır. Böylece programın herhangi bir yerinde oda sıcaklığı derecesi değiştirilirse hata verecektir. Aslında sabitlerin sunmuş olduğu temel avantaj da budur. Böylece sıkça kullanılan bir değer programın herhangi bir yerinde değiştirildiğinde derleyici hata verip bizi uyaracaktır. Aşağıda farklı tiplerde sabitlerin tanımlandığı örnekler görülmektedir.

```
Const ders =”Görsel Programlama”
Const sicaklik As String=18
Const Pi As Double=3.14
Const okul As String=”Bilgisayar Programcılığı”
```

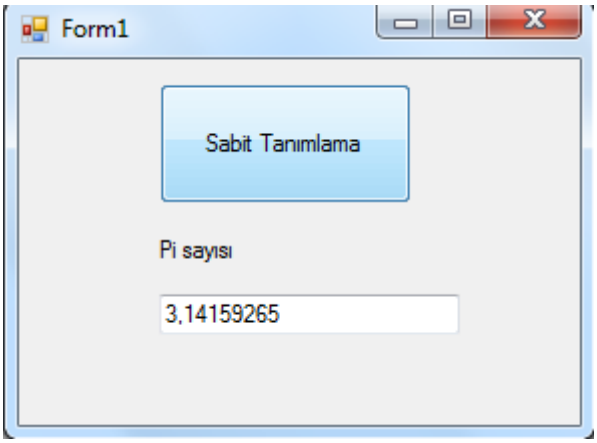
Program içerisinde değeri değişmeyecek veriler için Sabit tanımlanmalıdır.

Özetle tanımlanan sabite aktarılan veri sonradan değiştirilmek istendiğinde hata oluşur, sabitler daha önce de belirtildiği gibi değerlerini program sonuna kadar korurlar.

Örnek 10

- Pi sayısının sabit tanımlanarak kullanılması.

```
Private Sub Button1_Click(ByVal sender As System.Object,
    Const Pi As Double = 3.14159265
    TextBox1.Text = Pi
End Sub
```



Resim 5.10. Örnek 10’daki kodların çalıştırıldığında oluşan ekran görüntüsü

Özet

- Programlama dilleri kullanılarak geliştirilen yazılımlarda veri ve bilgiler tanımlanan değişkenlere aktarılmak suretiyle saklanırlar ve program içerisinde gerektiği yerde çağrılarak işleme tabi tutulurlar.
- Değişkenler basit (aynı anda tek veri tutabilen) veya Dizi şeklinde iki türden oluşurlar.
- Değişken tanımında "DIM" komutu kullanılmalıdır. Tanımlanan değişkenler ünitede anlatılan birçok tipte oluşturulabilirler. Uygun tipte tanımlanmayan değşkenlere aktarılan veriler bazen çalışma hatasına bazen de bellekte gereksiz hafıza kullanımına neden olurlar.
- Ayrıca program içerisinde değeri değişmeyecek veriler için de Sabit tanımlanmalıdır.

DEĞERLENDİRME SORULARI

1. Aşağıdaki değişken tanımlarından hangisi doğrudur?
   a) As ulkeadi As String
   b) As ulkeadi Dim String
   c) Dim ulkeadi String As
   d) Dim As ulkeadi String
   e) Dim ulkeadi As String

2. Aşağıdaki değişken tiplerinden hangisi sayısal özellikte değildir ?
   a) Short
   b) String
   c) Decimal
   d) Single
   e) Integer

3. Aşağıdaki atama işlemlerinden hangisi doğru ifade edilmiştir?
   a) Soyad = "AYIK"
   b) Soyad "AYIK"
   c) “AYIK” = Soyad
   d) Soyad, AYIK
   e) “Soyad”=AYIK

4. Aşağıdaki değişken isimlerinden hangisi doğrudur?
   a) And
   b) Ders Notu
   c) MsgBox
   d) Notlar
   e) 1Ders

5. Aşağıdakilerden hangisinde değişken atama ve ilk değer ataması doğru olarak verilmiştir?
   a) Dim sıcaklık As Integer=”20”
   b) Dim sıcaklık Integer=20
   c) Dim sıcaklık As Integer=20
   d) sıcaklık As Dım String=20
   e) Dim sıcaklık As String=20

6. Long değişken tipinin Common Type System’de kullanılan takma ismi aşağıdakilerden hangisinde doğru gösterilmiştir?
   a) System.Long
   b) System.Int16
   c) System.Int32
   d) System.Int64
   e) Long

7. K verisi bir değişkene aktarılmak istendiğinde değişken hangi türde tanımlanmalıdır?
   a) Integer
   b) Decimal
   c) Single
   d) String
   e) Char

8. Aşağıdakilerden hangisinde bir Sabit tanımlanması ve verinin Sabit'e atanması doğru gösterilmiştir?
   a) Const Mevsim="Kış"
   b) Const okul
   c) Dim Const Pi=3.14
   d) Const Dim Mevsim="Kış"
   e) Mevsim="Kış"

9. Aşağıdaki dizi değişkenle ilgili ifadelerden hangisi doğrudur?
   Dim X(2,4) As String
   a) X dizi değişkeni 2 boyutlu ve 6 elemanlıdır.
   b) X dizi değişkeni 1 boyutlu 2 elemanlıdır.
   c) X dizi değişkeni 2 boyutlu 15 elemanlıdır.
   d) X dizisi sayısal veriler saklamaktadır.
   e) Dizinin adı verilmemiştir.

10. Byte tipi değişkenin alabileceği minimum ve maximum değerler hangi seçenekte doğru gösterilmiştir?
   a) Minimum:-2.147.483.648
      Maximum: 2.147.483.647
   b) Minimum:--32768
      Maximum: 32767
   c) Minimum:-9.223.372.036.854.775.808
      Maximum: 9.223.372.036.854.775.807
   d) Minimum:0
      Maximum: 255
   e) Minimum: A
      Maximum: Z

Cevap Anahtarı

1.e, 2.b, 3.a, 4.d, 5.c, 6.d, 7.e, 8.a, 9.c, 10.d

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Ayık Y. Ziya. (2009). Algoritma ve Programlama Metodolojisi, MuratHan, Trabzon

Ayık Y. Ziya, (2011), Atatürk Üniversitesi, Uzaktan Eğitim Merkezi, Görsel Programlama I Ders Notları

Yanık Memik(2010), Visual Basic 10, Seçkin Yayınevi, Ankara

GELİŞMİŞ NESNELER

İÇİNDEKİLER

- Gelişmiş Kontrol Nesneleri
  - CheckBox Kontrolü
  - CheckedListBox Kontrolü
  - ListBox Kontrolü
  - ComboBox Kontrolü
  - RadioButton Kontrolü
  - PictureBox Kontrolü
  - WebBrowser Kontrolü
  - ProgressBar Kontrolü
  - Veri Erişim Kontrolleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Gelişmiş kontrol nesnelerini tanıyabilecek,
  - Gelişmiş kontrol nesnelerinin özelliklerini bilecek,
  - Gelişmiş kontrol nesnelerini kullanarak arayüz tasarlayabilecek ve basit uygulamalar geliştirebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr. Mustafa KESKİNKILIÇ

ÜNİTE

6

GİRİŞ

Grafiksel Kullanıcı Arayüzü (GUI), bilgisayarda işletilen komutlar ile bunların çıktıları yerine; simgeler, pencereler, düğmeler ve panellerin tümünü ifade etmek için kullanılan genel addır.

Görsel programlama, bilgisayarların kullanım alanlarının genişlemesi, çözülmesi gereken problemlerin sayılarının artması ve problemlerin zorlaşması sonucunda gelişen Nesne Yönelimli Programla yaklaşımıyla beraber ortaya çıkmıştır. Bilgisayarların kullanımının kolaylaştırılması ve hem programlamanın hem de kullanmanın kolaylaşmasını sağlamıştır. Görsel programlama, programlamaya arayüz kavramını getirmiştir. Arayüzler sayesinde aranılan şeylerin kolaylıkla bulunması sağlanmıştır. Kullanıcı dostu ve göze hitap eden programların geliştirilmesi sağlanmıştır. Kısacası arayüzler görsel programlamanın kendisidir.

Arayüzlerin iyi geliştirilmediği yazılımlar kullanıcılar tarafından tercih edilmez. Zaten kullanıcının programla olan ilişkisi arayüz üzerinedir. Kullanıcılar programın çalışma mantığı veya kodlarıyla değil arayüzle uğraşırlar. Büyük yazılım firmalarında iş imkânı başlıklarından biri de "GUI Tasarımı"dır. Örneğin Microsoft firmasının geliştirdiği Windows işletim sisteminin yüksek oranlarda tercih edilmesinin sebebi başarılı arayüzüdür.

Günümüzde kullanıcıların beklentilerinin artmasına paralel olarak program geliştirmelerinde GUI gitgide önem önem kazanmaktadır. Bu ünitede Visual Studio.Net'in ihtiyaçlara cevap veren, kullanım kolaylığı ve görsel zenginliği bir arada sunan bazı gelişmiş kontroller incelenecektir.

GELİŞMİŞ KONTROL NESNELERİ

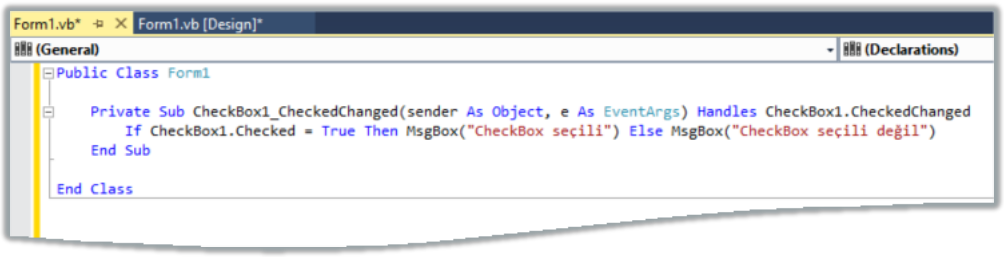
Designer ekranından bir kontrole çift tıklayarak varsayılan olayına erişilebilir.

Önceki ünitelerde TextBox, Button gibi çok sık kullanılan kontroller kullanılmıştı. Ancak Visual Studio.Net'te çeşitli ihtiyaçlara cevap vermek amacıyla geliştirilmiş farklı özelliklere sahip birçok kontrol bulunmaktadır. Bu ünitede Gelişmiş Kontrol Nesnelerinin bazıları incelenecek ve bu kontrollerle ilgili örnekler verilecektir.

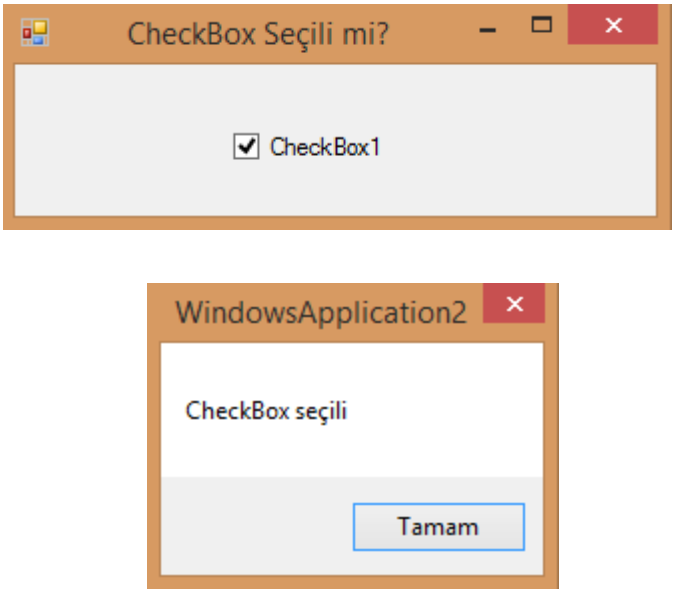
Visual Basic.Net araç kutusunda(Toolbox) programın tasarımı sırasında kullanılmak üzere çok sayıda kontrolü içerir. Bu başlık altında birçok projede sıkça kullanılan bazı kontroller incelenecektir.

CheckBox Kontrolü

Onay veya seçim işlemleri için kullanılan kontrol nesnesidir. Bu kontrolün en çok kullanılan özelliklerinden biri *"checked"* özelliğidir. CheckBox'ın seçili olması durumunda checked özelliği *True*, seçili olmaması durumunda ise bu özellik *False* değerini alır. CheckBox kontrolü genellikle If karar yapısıyla birlikte kullanılır. Örneğin CheckBox'ın seçili olup olmadığını mesaj kutusuyla bildiren programın kodları ve ekran görüntüsü aşağıdaki gibidir:



Şekil 6.1 Örnek Program Kodları

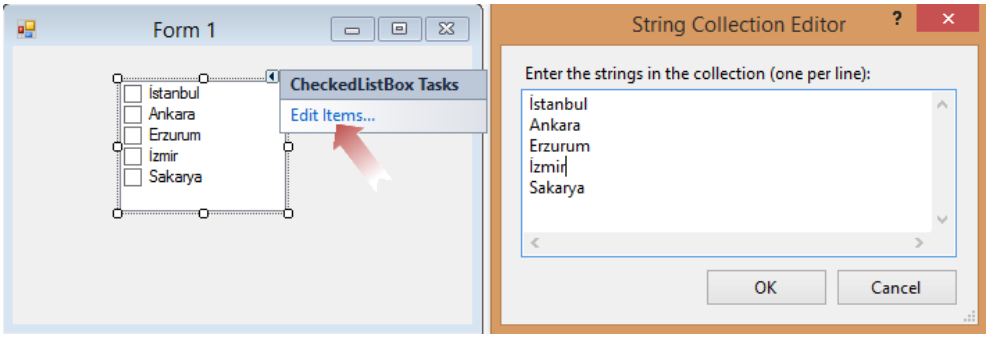


Şekil 6.2 Örnek Program Ekran Görüntüsü

Örnekte CheckBox'ın *CheckedChanged* yani durum değişimi olayı kullanılmıştır. CheckBox'ın *checked* değeri *true* olduğunda "CheckBox seçili", *false* olduğunda ise "CheckBox seçili değil" mesajı ekranda görüntülenecektir.

CheckedListBox Kontrolü

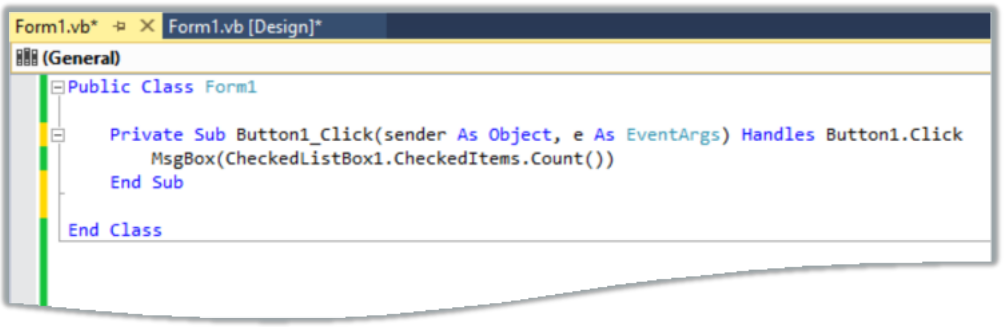
İçerisindeki değerleri, seçilebilir liste şeklinde görüntüleyen kontrol nesnesidir. Bu kontrolün içerisindeki tüm elemanları temsil etmek için *Items* özelliği kullanılırken, seçili elemanları temsil etmek için *CheckedItems* özelliği kullanılır. CheckedListBox içerisine eleman eklemek için Items özelliğinin Add metodu kullanılır. Ancak bu işlem nesne üzerinde bulunan *Edit Items* bağlantısından da yapılabilir.



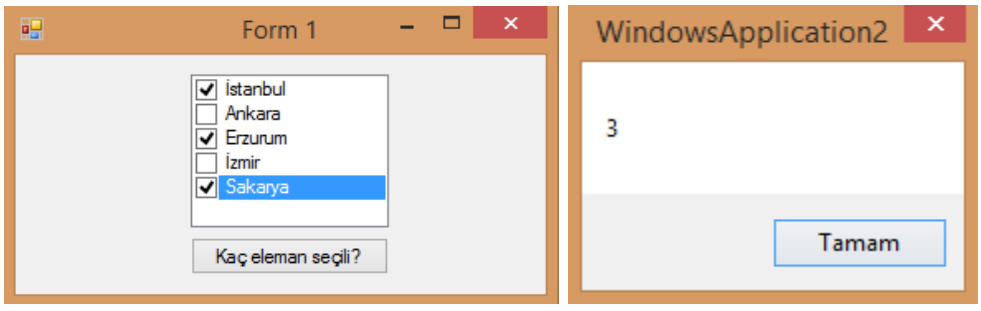
Şekil 6.3 Edit Items

Edit Items bağlantısıyla açılan *String Collection Editor* iletişim kutusuna girilen her eleman CheckedListBox'ta bir eleman olarak görüntülenir.

Örneğin bir CheckedListBox'ta bulunan elamanların kaç tanesinin seçili olduğunu mesaj kutusuyla ekranda görüntüleyen program kodları ve program ekran görüntüsü aşağıdaki gibidir:



Şekil 6.4 Örnek Program Kodları



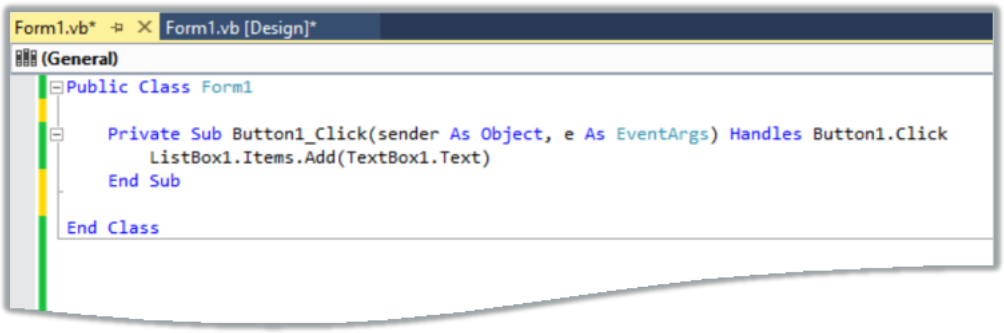
Şekil 6.5 Örnek Program Ekran Görüntüsü

ListBox Kontrolü

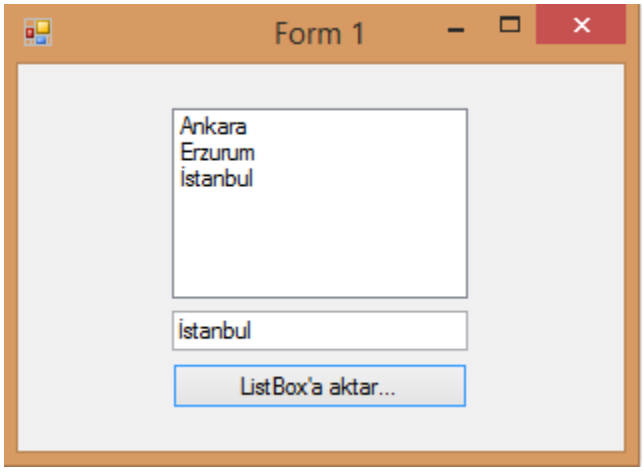
İçerisindeki değerleri bir liste hâlinde görüntüleyen kontrol nesnesidir. ListBox içerisinde bulunan elemanlara index numaraları kullanılarak erişilebilir. ListBox içindeki ilk eleman index numarası olarak 0 değerini alır.

ListBox nesnesinin en çok kullanılan özelliği Items özelliğidir. Bu özellik altında Add, Clear, Count gibi ekleme, silme işlemlerini yapan metotlar bulunur.

Örneğin butona basıldığında TextBox içerisindeki değeri ListBox'a aktaran programın kodu ve ekran görüntüsü aşağıdaki gibidir:



Şekil 6.6 Örnek Program Kodu



Şekil 6.7 Örnek Program Ekran Görüntüsü

ComboBox Kontrolü

İçerisindeki değerleri açılabilir bir liste hâlinde görüntüleyen kontrol nesnesidir. ComboBox nesnesinin özellik ve metotları ListBox nesnesiyle neredeyse aynıdır.

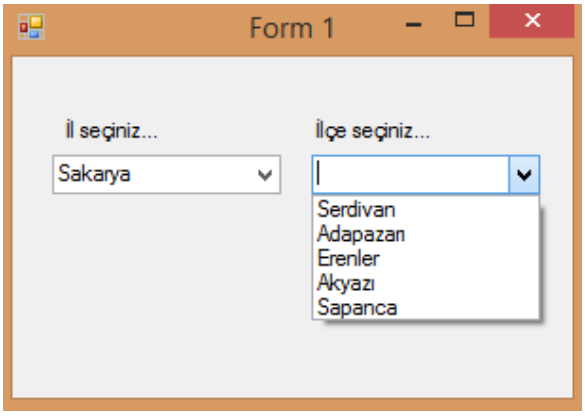
Örneğin form üzerinde iki ComboBox nesnesi bulunsun. Bunlardan birinde il isimleri diğerinde de ilçe isimleri bulunsun. Ancak ilçelerin bulunduğu ComboBox'taki bilgiler ilk ComboBox'tan seçilen il ismine göre gelecektir. Örnek program kodları ve ekran görüntüsü aşağıdaki gibi olacaktır:

Form1.vb* | Form1.vb [Design]*

Form1

```
Public Class Form1
    Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles ComboBox1.SelectedIndexChanged
        ComboBox2.Items.Clear()
        If ComboBox1.Text = "Erzurum" Then
            ComboBox2.Items.Add("Pakandöken")
            ComboBox2.Items.Add("Yakutiye")
            ComboBox2.Items.Add("Aziziye")
            ComboBox2.Items.Add("Oltu")
        ElseIf ComboBox1.Text = "İstanbul" Then
            ComboBox2.Items.Add("Beşiktaş")
            ComboBox2.Items.Add("Şişli")
            ComboBox2.Items.Add("Eminönü")
            ComboBox2.Items.Add("Beykoz")
            ComboBox2.Items.Add("Adalar")
        ElseIf ComboBox1.Text = "Sakarya" Then
            ComboBox2.Items.Add("Serdivan")
            ComboBox2.Items.Add("Adapazarı")
            ComboBox2.Items.Add("Erenler")
            ComboBox2.Items.Add("Akyazı")
            ComboBox2.Items.Add("Sapanca")
        End If
    End Sub
End Class
```

Şekil 6.8Örnek Program Kodları



Şekil 6.9 Örnek Program Ekran Görüntüsü

RadioButton Kontrolü

Form üzerine eklenen birden fazla RadioButton içerisinden sadece biri seçili durumda olabilir.

Birden fazla seçenekten sadece birini seçmenin zorunlu olduğu durumlarda kullanılan kontrol nesnesidir. Gerek işlevi gerek kullanım amacı açısından *CheckBox* kontrolüne benzemektedir. *Checked* özelliği sayesinde RadioButton nesnesinin seçili olup olmadığı kontrol edilir.

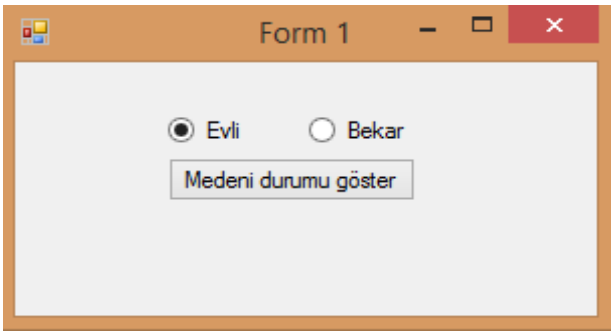
Örneğin kişisel bilgilerin istendiği bir formda medenî durumun seçilmesi gerektiğinde *CheckBox* yerine *RadioButton* kullanmak daha mantıklıdır. Çünkü bir kişinin medenî durumu ya evli ya da bekâr olabilir. Seçilen medenî durumu ekrana mesaj kutusu ile bildiren program kodları ve ekran görüntüsü aşağıdaki gibidir:

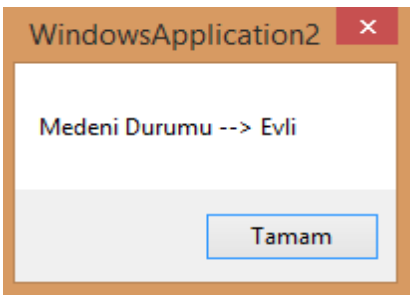
Form1.vb* | Form1.vb [Design]*

(General)

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If RadioButton1.Checked = True Then
            MsgBox("Medeni Durumu --> Evli")
        Else
            MsgBox("Medeni Durumu --> Bekar")
        End If
    End Sub

End Class
```

Şekil 6.10 Örnek Program Kodları

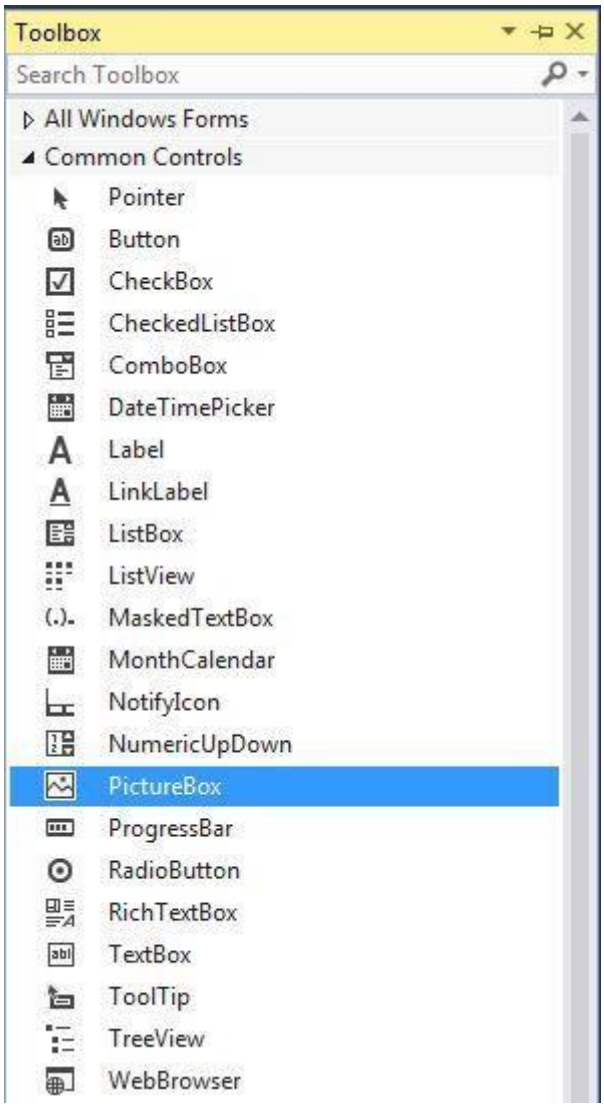




Şekil 6.11 Örnek Program Ekran Görüntüsü

PictureBox Kontrolü

Programa görsellik katmak için bazı yerlere resimler eklenmektedir. Bunun için PictureBox kontrolünü kullanımı sıklıkla tercih edilmektedir. Resim dosyalarını görüntülemek için kullanılan PictureBox, görüntülenecek resmin adresinden resmin boyutlandırılmasına kadar birçok seçenek içermektedir. PictureBox kontrolü ToolBox'ta Common Controls sekmesi altında bulunmaktadır.



Şekil 6.12 PictoreBox

PictureBox kontrolünün bazı özellikleri şunlardır:

- *Image:* PictureBox içerisinde görüntülenecek olan resim dosyası yolunun bilgisini tutar.
- *ImageLocation:* Görüntülenecek resim dosyası internette ise bu dosyasının web adresi bilgisini tutar.
- *SizeMode:* Resim boyutlandırılması ile ilgili seçenekleri içerir.

- *ErrorImage:* Görüntülenecek resim dosyasına erişilemediği zaman gösterilecek alternatif resim dosyasının bilgisini tutar.

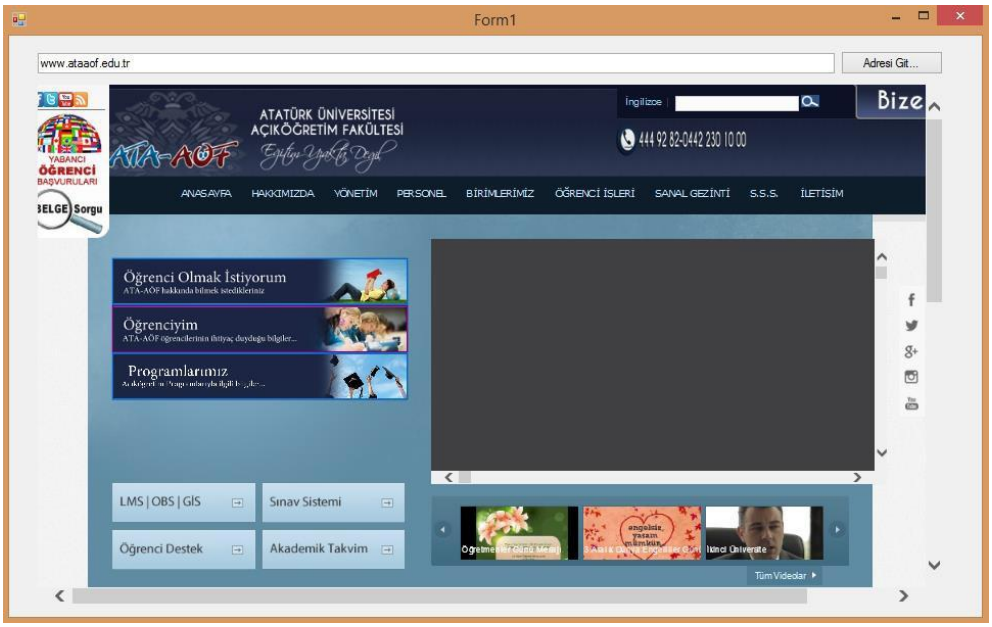
WebBrowser Kontrolü

WebBrowser (tarayıcı), web sayfalarını görüntülemek için kullanılan kontrol nesnesidir. Bu kontrol nesnesinin *navigate* özelliği yardımıyla istenilen adrese gidilerek web sayfası görüntülenir. Ayrıca durdurma, yineleme, ana sayfaya, önceki sayfaya ve sonraki sayfaya gitme işlemleri için *Stop, Refresh, GoHome, GoBack, GoForward* gibi özellikler kullanılır.

Örnek bir tarayıcı uygulaması program kodları ve ekran görüntüsü aşağıdaki gibidir.

```
Form1.vb | Form1.vb [Design]
(General)
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        WebBrowser1.Navigate(TextBox1.Text)

    End Sub

End Class
```

Şekil 6.13 Örnek Program Kodları



Şekil 6.14 Örnek Program Ekran Görüntüsü

Programda form üzerine birer adet Button, TextBox ve WebBrowser kontrolleri eklenmiştir. Butona tıklandığında, WebBrowser kontrolünün *Navigate* özelliği sayesinde TextBox içerisindeki adrese gidilmesi sağlanmıştır.

ProgressBar Kontrolü

Bilgisayarımıza uygulama yüklerken sıklıkla karşılaştığımız bir kontrol olan ProgressBar, uygulamalarımızda gerçekleşen bir işlemin ne kadarının tamamlandığını kullanıcıya göstermek amacıyla kullanılır.

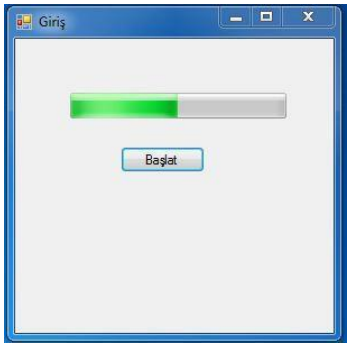
ProgressBar'ı Windows Formlarda kullanmanın birkaç yöntemi vardır. Bu başlık altında en kolay ve temel yöntem olan Value özelliğinin değeri ile kullanımı gösterilecektir.

ProgressBar'ın başlangıç ve bitiş değerlerini tutan iki özelliği vardır. Bu özellikler Minimum (Başlangıç) ve Maximum (Bitiş)'dur. Bir diğer özelliği olan Step ise ProgressBar'ın artış değerini belirlemektedir. Aşağıda verilen örnekte bu üç özellik kullanılarak 0'dan 100'e kadar 1 artış miktarına sahip ilerleme temsil edilmiştir.

```
Form1.vb*  Form1.vb [Design]*
(General)                                   (Declarations)
 1  Public Class Form1
 2
 3      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
 4
 5      End Sub
 6      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
 7          ProgressBar1.Maximum = 100
 8          ProgressBar1.Minimum = 0
 9          ProgressBar1.Step = 1
10          For i As Integer = 0 To 100
11              ProgressBar1.Value = i
12          Next
13      End Sub
14  End Class
15
```

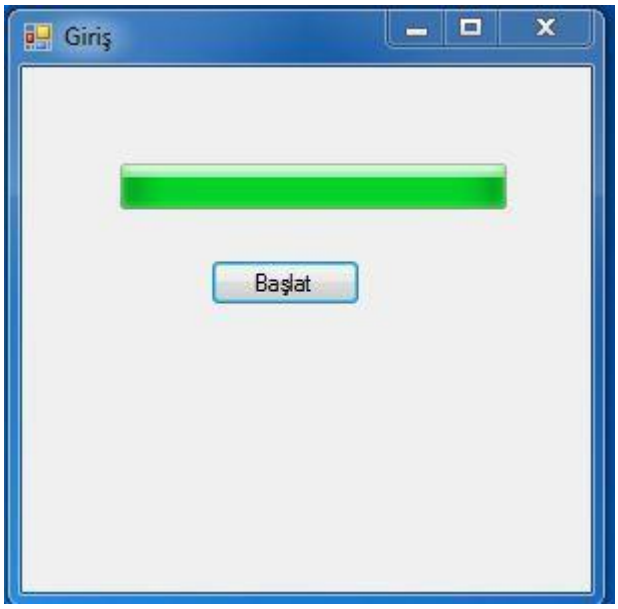
Şekil 6.15 Örnek Program Kodları

Bu örnekte ProgressBar'ın Maximum ve Minumum değerleri 0 ve 100 olarak belirlenmiş ve artış miktarını temsil eden Step özelliğine 1 değeri atanmıştır. Sonrasında ProgressBar'ın o anki değerini temsil eden Value özelliğine for döngüsü kullanılarak 0'dan 100'e kadar değer atanmıştır. Bu kodlar çalıştırıldığında aşağıdaki ekran görüntüsü elde edilmektedir.



Şekil 6.16 Örnek Program Ekran Görüntüsü

Resimde görülen yeşil renkli dolgu i değişkeni 100 olduğunda ProgressBar'In hepsini dolduracaktır ve ProgressBar aşağıdaki gibi görülecektir.

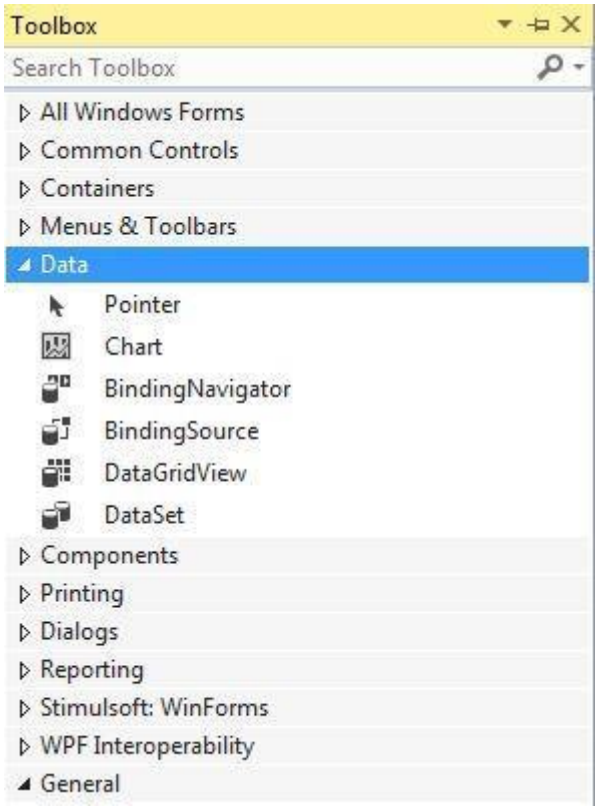


Şekil 6.17 Örnek Program Ekran Görüntüsü

Veri Erişim Kontrolleri

Masaüstü, Mobil ya da Web uygulamalarının birçoğu çalışırken veritabanı kullanmaktadır. Bu yüzden veri yönetimi programcıların hâkim olması gereken bir alan haline gelmiştir. Visual Studio.Net içerisinde veri yönetimi ile ilgili birçok kontrol bulunmaktadır. Bu kontroller kullanılarak veri tabanına kod yazmadan erişilebilir ve veriler üzerinde çeşitli işlemler yapılabilir.

Araç kutusunda (Toolbox) Data sekmesi altında yer alan veri kontrolleri, ADO.Net adı verilen ve .Net'in veri yönetimi ayağını oluşturan aracın en önemli bileşenleri arasındadır. Veri yönetimi ve ADO.Net konularına sonraki ünitelerde daha kapsamlı anlatılacağı için bu ünitede veri erişim kontrollerine değinilmeyecektir.



Şekil 6.18 Veri erişim kontrolleri Data sekmesi altında bulunur

Özet

- Grafiksel Kullanıcı Arayüzü (GUI), bilgisayarda işletilen komutlar ile bunların çıktıları yerine; simgeler, pencereler, düğmeler ve panellerin tümünü ifade etmek için kullanılan genel addır.
- Görsel programlama, bilgisayarların kullanım alanlarının genişlemesi, çözülmesi gereken problemlerin sayılarının artması ve problemlerin zorlaşması sonucunda gelişen Nesne Yönelimli Programla yaklaşımıyla beraber ortaya çıkmıştır.
- Büyük yazılım firmalarında iş imkânı başlıklarından biri de "GUI Tasarımı"dır.
- Önceki ünitelerde TextBox, Button gibi çok sık kullanılan kontroller kullanılmıştır.
- Bu ünitede CheckBox, ListBox, ComboBox gibi gelişmiş kontrol nesnelerinin bazıları incelenmiştir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisinde Grafiksel Kullanıcı Arayüzünün(GUI) tanımı doğru verilmiştir?
    a) Belirli sayıda ve aynı türden verilerden oluşan, tek bir isim ile ulaşılabilen ögeler serisidir.
    b) Bilgisayarda işletilen komutlar ile bunların çıktıları yerine; simgeler, pencereler, düğmeler ve panellerin tümünü ifade etmek için kullanılan genel addır.
    c) Uygulanma programlamak için sunulan arabirimlerin genel adıdır.
    d) Windows tabanlı uygulamaların diğer işletim sistemlerinde çalışması için .Net Framework'un sunmuş olduğu bir arabirimdir.
    e) Veritabanına erişimi kolaylaştıran bileşenlere verilen genel addır.

2. Aşağıdakilerden hangisi kontrol değildir ?
    a) Label
    b) Combobox
    c) Stored Procedure
    d) Rich Textbox
    e) Masked Textbox

3. Birçok elemanı liste biçiminde göstermeye yarayan kontrol aşağıdakilerden hangisidir?
    a) ListBox
    b) TextBox
    c) PictureBox
    d) ComboBox
    e) Label

4. Resimleri göstermek için aşağıdakilerden hangisi kullanılabilir?
    a) ListBox
    b) Button
    c) TextBox
    d) PictureBox
    e) Rich TextBox

5. Kullanıcıya elemanları açılır bir kutu içerisinde sunan ve kullanıcının seçim yapabilmesini sağlayan kontrol aşağıdakilerden hangisidir?
    a) TextBox
    b) ListBox
    c) ComboBox
    d) CheckBox
    e) Radio Button

6. ComboBox'a eleman eklemek için aşağıdaki metodlardan hangisi kullanılır?
    a) Contains
    b) FindByValue
    c) FindByText
    d) Clear
    e) Add

7. CheckBox'ın seçili olup olmadığı hangi özelliğine bakılarak kontrol edilir?
    a) Checked
    b) Text
    c) BackColor
    d) Enabled
    e) Visible

8. CheckBox ve Radio Button'un Checked özelliği hangi tipte değer döndürmektedir?
    a) Integer
    b) String
    c) Float
    d) Boolean
    e) Char

9. Aşağıdaki kontrol çiftlerinden hangisi kullanım amacı ve içerdiği özellikler açısından birbirine en fazla benzemektedir?

   a) TextBox-PictureBox

   b) Label-CheckBox

   c) RadioButton-CheckBox

   d) ComboBox-TextBox

   e) RadioButton-Button

10. Aşağıdakilerden hangisi yanlıştır ?

   a) ComboBox ve ListBox liste tipinde kontrollerdir.

   b) ComboBox'ta items özelliği kullanılarak elemanlar üzerinde işlem yapılabilir.

   c) CheckBox kontrolünün herhangi bir olayı bulunmamaktadır.

   d) ListBox kontrolü listeme işlemlerinde kullanılabilir.

   e) RadioButton kontrolü resim görüntülemek için kullanılabilir.

Cevap Anahtarı

1.b, 2.c, 3.a, 4.d, 5.c, 6.e, 7.a, 8.d, 9.c

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Sharp, John, Visual C# 2012 Step by Step, Microsoft Press, ISBN 978-0-7356-6801-0, Washington, 2012.

Halvorson, Micheal, Visual Basic 2010 Step by Step, Microsoft Press, ISBN 978-0-7356-2669-0, Washington, 2010.

AKTAŞ, Volkan, Visual Studio 2010 ile Her Yönüyle C#, Kodlab, İstanbul, 2011.

MASTAR, Muhammed, VURAL, M. Tahir, ERİŞ, Süha, Visual Studio 2012, Kodlab, İstanbul, 2013.

GÜNGÖREN, Bora, JAVA ile Temel Programlama, Seçkin Yayınevi, İstanbul, 2003.

KESKİNKILIÇ, Mine, JAVA ile Programlama, Seçkin Yayınevi, Ankara, 1997.

DURSUN Hüseyin, Nesneye Yönelik Programlama ve C++, Pusula, İstanbul, 1996.

DEMİRALP Fehmi, C++ ile Nesneye Dayalı Programlama, Beta, İstanbul, 1993.

YENİMAN YILDIRIM, Ebru, Yüksekokul Öğrencileri için Uygulamalı Görsel Programlama Visual Basic, Nobel Yayın Dağıtım, İstanbul, 2007.

http://msdn.microsoft.com

http://www.yazilimevim.com/Makale.aspx?ad=kontroller-ve-ozellikleri

OPERATÖRLER

İÇİNDEKİLER

- Operatör Nedir?
- Aritmetik Operatörler
- Atama Operatörleri
- Mantıksal Operatörler
- Karşılaştırma Operatörleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Operatörleri tanıyabilecek,
  - Operatörleri sınıflandırabilecek,
  - Operatörleri kullanarak çeşitli işlemler yapabilecek,
  - Operatörlerin kullanım mantığını anlayabilecesiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Dr. Öğr. Üyesi
Mehmet Cem BÖLEN

Uzm. Orhan ÇELİKER

ÜNİTE

7

GİRİŞ

Birçok programlama dilinde olduğu gibi Visual Basic.Net programlama dilinde de çalışırken en sık kullanılan yapılardan biri de operatörlerdir. Hesaplamalar, mantıksal işlemler ya da kontrol fonksiyonları gibi birçok yerde operatörler kullanılır. Bu bölümde Visual Basic.Net dilinde çok sık kullanılan temel operatörler incelenecektir.

OPERATÖR NEDİR?

Değişkenler üzerinde birçok işlem operatörler kullanılarak yapılır.

Operatörler, Visual Basic.Net içerisinde kendi başlarına bir anlam ifade etmeyen ancak başta matematiksel ve mantıksal olmak üzere çeşitli işlemleri gerçekleştirmek için kullanılan özel karakter ya da sembollerdir. Programlama ile ilgili kaynaklar incelendiğinde programlama dili, kullanım amacı ve operatörün işlevine göre farklı sınıflara ayrıldığı görülmektedir. Kullanım amacı ve işlevi dikkate alındığında Visual Basic.Net içerisinde operatörlerin 5 sınıfa ayrıldığı söylenebilir. Aşağıda verilen tabloda Visual Basic.Net içerisinde kullanılan bazı operatörler görülmektedir:

Tablo 7.1. Operatör Sınıflandırmaları

Operatörler	
Aritmetik Operatörler	+, - , /, *, Mod
Atama Operatörleri	=, +=, -=, *=,/=, %=
Karşılaştırma Operatörü	>, <, >=, <=, <>
Mantıksal Operatörler	And, Or, Not, AndAlso, OrElse, IsTrue, IsFalse
Özel Amaçlı Operatörler	TypeOf, AddressOf, Await

Yukarıdaki tabloda yer alan Özel Amaçlı Operatörler ileri seviyede bir konu olduğu için bu ünitede anlatılmayacaktır. Ünitede sırasıyla *aritmetik operatörler, atama operatörleri, mantıksal operatörler ve karşılaştırma operatörleri* incelenecektir.

ARİTMETİK OPERATÖRLER

Aritmetik operatörler işlem önceliği mantığına göre çalışır.

Toplama, çıkarma, çarpma, bölme, mod ve üs alma gibi temel matematiksel işlemleri gerçekleştirmek için aritmetik operatörler kullanılır. Aşağıda her bir aritmetik operatörün kullanıma ait birer örnek verilmiştir:

Toplama Operatörü: (+)

```
Dim a As Integer = 10
Dim b As Integer = a + 20
```

Çıkarma Operatörü: (-)

```
Dim a As Integer = 50
Dim b As Integer = a - 10
```

Çarpma Operatörü: (*)

```
Dim a As Integer = 5
Dim b As Integer = a * 4
```

Bölme Operatörü: (/)

```
Dim a As Integer = 25
Dim b As Integer = a / 5
```

Mod Operatörü: (Mod)

```
Dim a As Integer = 25
Dim b As Integer = a Mod 4
```

Üs Alma Operatörü: (^)

```
Dim a As Integer = 5
Dim b As Integer = a ^ 2
```

Aritmetik operatörlerin bazılarının kullanım alanı matematiksel hesaplamalarla sınırlı değildir. Örneğin toplama ve çıkarma işlemleri için kullanılan + ve – operatörleri aynı zamanda işaret operatörü olarak da kullanılır. Bu operatörler bir değişkenin negatif ya da pozitif olmasını sağlar. Yine + operatörü String tipindeki karakterleri birleştirmek için kullanılabilir.

Özetle arimetik operatörler genelde değişkenler veya sabitler üzerinde temel aritmetik işlemleri gerçekleştirmek için kullanılır ve Visual Basic.Net içerisinde en sık kullanılan operatör tiplerinden biridir.

ATAMA OPERATÖRLERİ

Atama operatörleri bir değişekene değer atamak amacıyla kullanılır. Atama operatörlerinin vazgeçilmezi = karakteridir. Değer atama operatörünü kullanırken bilmemiz gereken en önemli şey operatörün sol tarafında değer atanacak değişkenin, sağ tarafında ise değişkene atanacak değerin belirtilmesi gerektiğidir. Aşağıda verilen örnekte bir değişkene basit bir değer ataması yapılmıştır:

```
Dim degisken As Integer
degisken = 1
```

Örnekte görüldüğü üzere = operatörü kullanılarak 1 değeri değişkene atanmıştır. Bazı durumlarda değişkenin varolan değeri üzerinde toplama, çıkarma gibi işlemler yapılabilir. Bu gibi durumlarda eşittir *(=)* operatörünün soluna istenilen aritmetik işlemi temsil eden operatör konur. Bu tip operatörlere işlemli atama operatörü de denebilir. Aşağıda verilen örnekte işlemli atama operatörünün nasıl çalıştığına yönelik bir örnek görülmektedir:

```
Dim sayi1 As Integer = 3
Dim sayi2 As Integer = 5
sayi1 += sayi2
```

Yukarıda sayi1 ve sayi2 değişkenleri tanımlanmış ardından sayi1 isimli değişkene sayi2 isimli değişkenin değeri *+=* operatörü kullanılarak eklenmiştir. Bu işlem sonucunda sayi1 değişkeni 8 değerini alır. İşlemi atama operatörleri sayesinde daha kısa ve sade kodlar yazılabilir.

İşlemli atama operatörlerini bazı durumlarda kodun daha kısa ve sade olmasını sağlar. Aşağıdaki tabloda farklı operatörlerin kullanıldığı aynı çıktıyı üreten kod örnekleri görülmektedir:

Tablo 7.2. Atama Operatörü ve İşlemli Atama Operatörü ile ilgili Örnekler

Atama Operatörü	İşlemli Atama Operatörü
A=A+B	A +=B
A=A-B	A-=B
A=A/B	A/=B
A=A*B	A*=B

MANTIKSAL OPERATÖRLER

Mantıksal operatörler Boolean tipinde değer döndürürler.

Visual Basic.Net'te sıklıkla kullanılan operatör gruplarından biri de mantıksal operatörlerdir. Özellikle program akışını yönlendirirken kurulan döngü ve kontrol yapılarında mantıksal operatörlerden yararlanılır.

Mantıksal operatör karakterleri ya da deyimleri programlama dillerine göre farklılık göstermektedir. Örneğin AND (Mantıksal VE) operatörü C# dilinde && karakteri ile temsil edilirken, Visual Basic.Net'te AND kelimesiyle kullanılır. En sık kullanılan mantıksal operatörler And (Ve),Or (Veya), AndAlso (VeDeğil) ve Not (Değil) operatörleridir. Bu operatörlerle yapılan karşılaştırmaların sonuçları True (Doğru) veya False (Yanlış) olması durumuna göre program akışı belirlenir. Visual Basic.Net dilindeki bu başlıca mantıksal operatörler aşağıdaki gibidir:

And Operatörü

Mantıksal iki ifadenin karşılaştırılması için kullanılır. Sonucun True (Doğru) olması için iki ifadenin de mutlaka doğru olması gereklidir.

Tablo 7.3. And Operatörü

İfade-1	İfade-2	İfade-1 And İfade-2	Örnek
False	False	False	3>5 and "a"="b" → False
False	True	False	3>5 and "a"="a" → False
True	False	False	5>3 and "a"="b" → False
True	True	True	5 >3 and " a"=" a" →

AndAlso Operatörü

AndAlso operatörü And operatöründen daha hızlı çalışmaktadır.

AndAlso operatörünün çalışma mantığı *And* operatörü ile hemen hemen aynıdır. Ancak *AndAlso* operatörü ilk ifade False değeri döndürdüyse ikinci ifadeyi kontrol etmeden False değeri döndürmektedir. Bu nedenle *And* ifadesine göre daha performanslı çalışmaktadır. Özetle döndürdüğü değer olarak *And* operatöründen bir farkı bulunmayıp sadece ilk kontrol ifadesi False değeri döndürünce *And* operatörünün aksine ikinci ifadeyi kontrol etmemektedir.

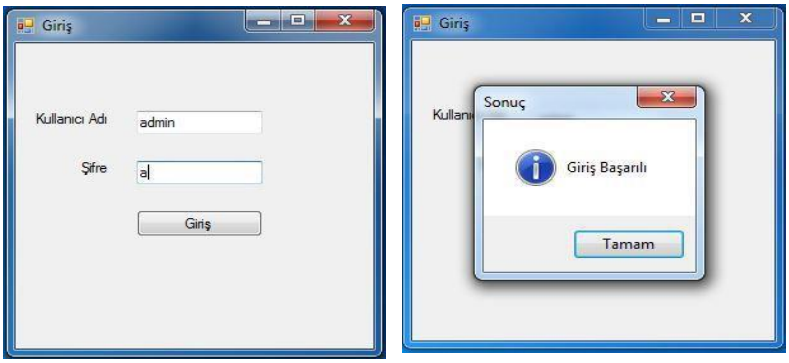
Aşağıdak örnekte kullanıcı adı ve şifreyi kontrol eden bir if döngüsü verilmiştir. Eğer kullanıcı adı TextBox'ına girilen değer *admin*, şifre TextBox'ına girilen değer *a* olursa ekrana *Giriş Başarılı* mesajı gösterilecektir.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub

    Private Sub btnGiris_Click(sender As Object, e As EventArgs) Handles btnGiris.Click

        If txtKullaniciAdi.Text = "admin" AndAlso txtSifre.Text = "a" Then
            MsgBox("Giriş Başarılı", MsgBoxStyle.Information, "Sonuç")
        End If

    End Sub
End Class
```

Resim 7.1. Örnek Program Kodları

Bu döngünün istenilen işlemi yapabilmesi için hem *And* hem de *AndAlso* operatörü kullanılabilir. Ancak *AndAlso* operatörü kullanıldığı takdirde eğer txtKullaniciAdi isimli TextBox kontrolüne girilen değer admin dışında bir şey olursa txtSifre TextBox'ına girilen değer kontrol edilmeden If koşul ifadesi false değeri döndürecek ve MsgBox'ın bulunduğu kod satırı çalıştırılmayacaktır.

Yukarıdaki program çalıştırıldığında ilgili metin kutularına admin ve a değerleri girildiği takdirde Giriş Başarılı mesajıyla karşılaşılacaktır. (Resim 7.2.)



Resim 7.2. Örnek Program Ekran Görüntüsü

Or Operatörü

Mantıksal iki ifadenin karşılaştırılması sonucunda iki ifadeden birinin doğru olması sonucun True (Doğru) dönmesine yeterlidir. İfadelerin her ikisinin de doğru olması sonucu yine True (Doğru) döndürür. Ancak her iki ifade de yanlış olursa sonuç False (Yanlış) olarak döner.

Tablo 7.4. Or Operatörü

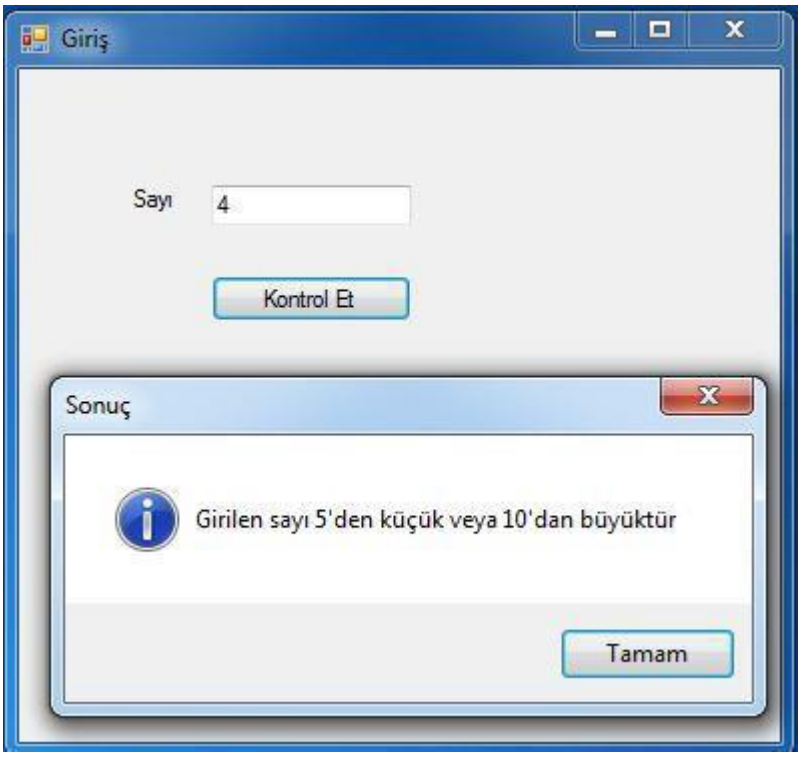
İfade-1	İfade-2	İfade-1 Or İfade-2	Örnek
False	False	False	3>5 or 10>13 → False
False	True	True	3>5 or 10<13 → True
True	False	True	3<5 or 10>13 → True
True	True	True	3<5 or 10<13 → True

Aşağıda verilen örnekte Or operatörünün kullanımı görülmektedir (Resim 7.3).

```
Form1.vb | Form1.vb [Design]
(General)                                   (Declarations)
1  Public Class Form1
2
3      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
4
5      End Sub
6
7      Private Sub btnGiris_Click(sender As Object, e As EventArgs) Handles btnGiris.Click
8
9          If (txtSayi.Text < 5) Or (txtSayi.Text > 10) Then
10             MsgBox("Girilen sayı 5'den küçük veya 10'dan büyüktür", MsgBoxStyle.Information, "Sonuç")
11         End If
12
13     End Sub
14 End Class
15
```

Resim 7.3. Örnek Program Kodları

Örnek incelendiğinde iki koşul ifadesinden ilkinin txtSayi isimli TextBox'ın değerini kontrol ettiği ve 5'ten küçükse True değeri döndüreceği, ikinci ifadenin ise txtSayi isimli TextBox'ın değerini kontrol ettiği ve 10'dan büyükse True değeri döndüreceği görülmektedir. If koşul satırında Or operatörü kullanıldığı için bu iki koşul ifadesinden herhangi biri True değeri döndürürse MessageBox'ın görüntülendiği kod satırı çalışacaktır. Buna göre txtSayi isimli TextBox'a 5'ten küçük veya 10'dan büyük bir sayı girilirse aşağıdaki ekran görüntüsü ile karşılaşılacaktır.(Resim 7.4 ve Resim 7.5)



Resim 7.4. Örnek Program Ekran Görüntüsü

Resim 7.5. Örnek Program Ekran Görüntüsü

Not Operatörü

Mantıksal ifadenin tersini (değilini) almak için kullanılır. Mantıksal ifade doğru ise False (Yanlış), yanlış ise True (Doğru) sonucu geriye döner. Aşağıdaki tabloda Not operatörü kullanılarak bazı değerlerin tersi alınmıştır.

Tablo 7.5. Not Operatörü

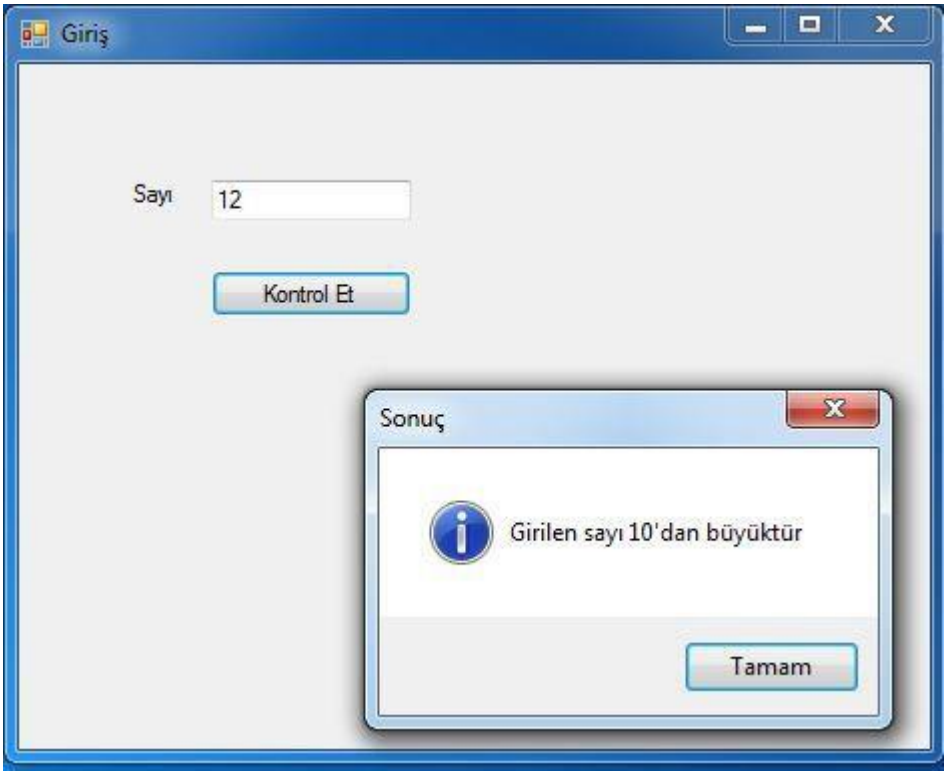
İfade		Sonuç
Not(5>3)	→	False
Not(3>5 or 3>5)	→	True
Not(Not (Not (5<3)))	→	True
Not(True)	→	False

Aşağıdaki kodlar çalıştırıldığında *txtSayi* isimli TextBox'a girilen değerin *10'dan büyük olup olmamasına göre* MessageBox'ta buna uygun bir mesaj gösterilmektedir. (Resim 7.6)

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub

    Private Sub btnGiris_Click(sender As Object, e As EventArgs) Handles btnGiris.Click

        If Not (txtSayi.Text <= 10) Then
            MsgBox("Girilen sayı 10'dan büyüktür", MsgBoxStyle.Information, "Sonuç")
        Else
            MsgBox("Girilen sayı 10'dan küçüktür", MsgBoxStyle.Information, "Sonuç")
        End If

    End Sub
End Class
```

Resim 7.6. Örnek Program Kodları

Örnekte görüldüğü üzere Not operatörü parantez içerisinde verilen koşuldan dönen değeri tersine çevirmektedir. Örneğin txtSayi isimli TextBox'ın değeri 12 olursa bu değer 10'dan büyük olduğu için *False* değeri döndürecek ve Not operatörü bu değeri tekrar *True'ya* çevirecektir. Bundan dolayı kodlar çalıştırıldığında ekran görüntüsü aşağıdaki gibi olacaktır. (Resim 7.7)



Resim 7.7. Örnek Program Ekran Görüntüsü

KARŞILAŞTIRMA OPERATÖRLERİ

Karşılaştırma operatörleri kullanılarak verilen ifadelerin değerleri arasındaki büyüklük, küçüklük veya eşitlik ilişkisi belirlenir. Bu operatörler genellikle if karar yapısı ve döngü ifadelerinde kullanılır. Karşılaştırma operatörleri ve anlamları aşağıdaki tabloda verilmiştir:

Tablo 7.6. Karşılaştırma Operatörleri

Operatör		Anlamı	Örnek (Mantıksal)	Örnek (Matematiksel)
=	→	Eşit	Sonuc=(3=5) Sonuc=False	A=B*10
>	→	Büyük	Sonuc=(5>3) Sonuc=True	(A^2)>3
>=	→	Büyük Eşit	Sonuc=(3>=5) Sonuc=False	A*B>=C
<	→	Küçük	Sonuc=(3<5)	A<B
<=	→	Küçük Eşit	Sonuc=(3<=3) Sonuc=True	35<=A
<>	→	Eşit Değil	Sonuc=(3<>5) Sonuc=True	AB<>(C*56)

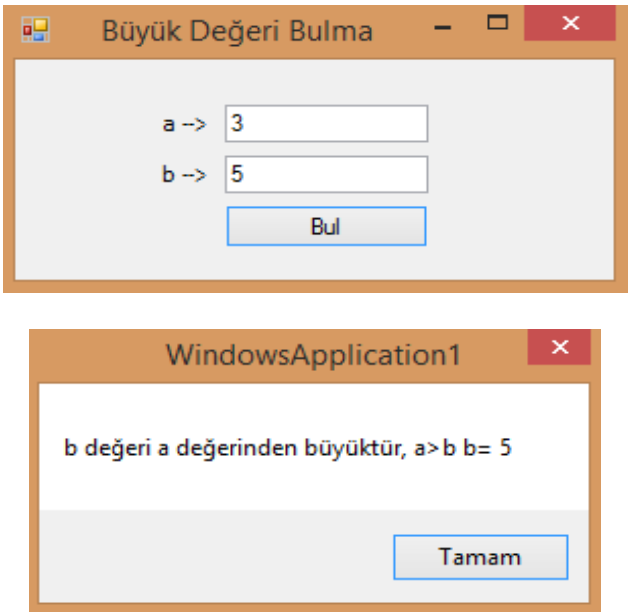
Örnek: Form üzerinde bulunan iki metin kutusundaki (TextBox) değerleri karşılaştıran ve sonucu mesaj kutusuyla (MessageBox) bildiren programın kodları ekran görüntüsüyle birlikte aşağıda verilmiştir:

Form1.vb* | Form1.vb [Design]*

(General)

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim a, b As Integer
        a = TextBox1.Text
        b = TextBox2.Text
        If a > b Then
            MsgBox("a değeri b değerinden büyüktür, a>b a= " & a)
        ElseIf b > a Then
            MsgBox("b değeri a değerinden büyüktür, a>b b= " & b)
        End If
    End Sub
End Class
```

Resim 7.8. Örnek Program Kodları



Resim 7.9. Örnek Program Ekran Görüntüsü

Programda metin kutularına girilen değerler <, > operatörleriyle karşılaştırılır ve büyük olan değer mesaj kutusuyla ekranda görüntülenir (Resim 7.9).

Bireysel Etkinlik

- Bir TextBox ve bir Button kontrolünün olduğu Visual Basic dilinde Windows Forms projesi oluşturunuz.
- Button'a tıklandığında TextBox'a girilen değerin çift mi yoksa tek sayı mı olduğu bilgisini MessageBox'ta görüntüleyecek kodu yazınız.

Örnek: Form üzerinde bulunan şifre adlı metin kutusuna girilen değerin istenilen değere eşit olup olmadığını mesaj kutusuyla bildiren program kodları ve ekran görüntüsü aşağıda verilmiştir:

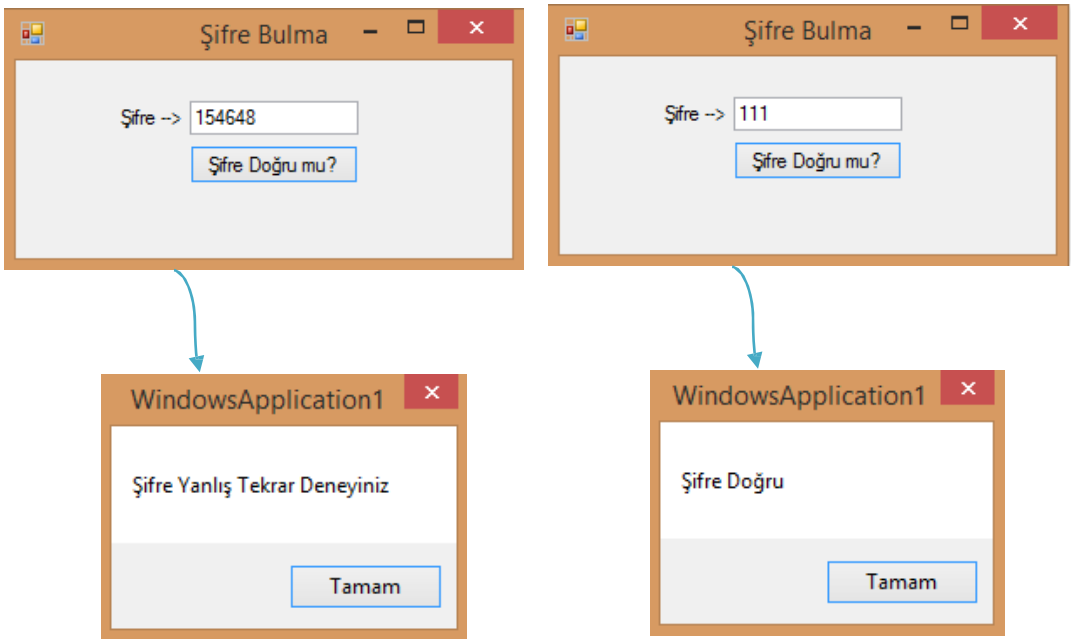
Form1.vb* | Form1.vb [Design]*

(General)

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim a As Integer
        a = TextBox1.Text
        If a = 111 Then
            MsgBox("Şifre Doğru")
        ElseIf a <> 111 Then
            MsgBox("Şifre Yanlış Tekrar Deneyiniz")
        End If
    End Sub
End Class
```

Resim 7.10. Örnek Program Kodları

Programda eşit değil (<>) operatörü kullanıldığından metin kutusuna 111 değeri dışında girilen tüm değerler için "Şifre yanlış tekrar deneyiniz" mesajı verilecektir.



Resim 7.11. Örnek Program Ekran Görüntüsü

Özet

- Operatörler, Visual Basic.Net içerisinde kendi başlarına bir anlam ifade etmeyen ancak başta matematiksel ve mantıksal olmak üzere çeşitli işlemleri gerçekleştirmek için kullanılan özel karakter ya da sembollerdir.
- Kullanım amacı ve işlevi dikkate alındığında Visual Basic.Net içerisinde operatörler 5 sınıfa ayrılır. Bu sınıflar aritmetik operatörler, atama operatörleri, mantıksal operatörler ve karşılaştırma operatörleridir.
- Toplama, çıkarma, çarpma, bölme, mod ve üs alma gibi temel matematiksel işlemleri gerçekleştirmek için aritmetik operatörler kullanılır.
- Atama operatörleri bir değişekene değer atamak amacıyla kullanılır. Atama operatörlerinin vazgeçilmezi = karakteridir.
- Program akışını yönlendirirken kurulan döngü ve kontrol yapılarında mantıksal operatörlerden yararlanılır.
- Karşılaştırma operatörleri kullanılarak verilen ifadelerin değerleri arasındaki büyüklük, küçüklük veya eşitlik ilişkisi belirlenir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi arimetik operatör değildir?
   a) +
   b) Mod
   c) /
   d) Not
   e) –

2. Bir değişkene değer atamak için aşağıdaki operatörlerden hangisi kullanılır?
   a) +
   b) *
   c) And
   d) =
   e) ?

3. Aşağıdakilerden hangisi Visual Basic.Net'te kullanılan operatörlerden biri değildir ?
   a) And
   b) &&
   c) +=
   d) *
   e) Not

4. Aşağıdakilerden hangisi işlemli atama operatörü değildir ?
   a) <>
   b) +=
   c) -=
   d) *=
   e) /=

5. Eşit değildir anlamına gelen operatör aşağıdakilerden hangisidir?
   a) >
   b) ^=
   c) <>
   d) ?=
   e) +=

6. Toplama, çıkarma, çarpma, bölme, mod ve üs alma gibi temel matematiksel işlemleri gerçekleştirmek için ..... operatörleri kullanılır.
   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?
   a) Karşılaştırma
   b) Mantıksal
   c) Atama
   d) Aritmetik
   e) İşlemli Atama

7. String tipindeki karakterleri birleştirmek için aşağıdaki operatörlerden hangisi kullanılabilir?
   a) And
   b) *
   c) -=
   d) Or
   e) +

8. Mantıksal operatörler hangi tipte değer döndürürler?
   a) Boolean
   b) String
   c) Float
   d) Double
   e) Integer

9. Değişkenler veya ifadeler arasındaki büyüklük, küçüklük gibi derecelerin belirlenmesi için hangi operatörler kullanılır?
   a) Mantıksal
   b) Aritmetik
   c) Karşılaştırma
   d) Özel Amaçlı
   e) Atama

10. Aşağıdakilerden hangisi mantıksal operatör değildir?
   a) Mod
   b) Or
   c) And
   d) Not
   e) AndAlso

Cevap Anahtarı:

1.d,2.d,3.b,4.a,5.c,6.d,7.e,8.a,9.c.10.a

YARARLANILAN KAYNAKLAR

ALBAHARI, Joseph & ALBAHARI, Ben, (2012), C# 5.0 in a Nutshell Fifth Edition, O'Reilly Media, California.

ALGAN, Sefer, (2009), Her Yönüyle C#, Pusula Yayıncılık, İstanbul.

AKTAŞ, Volkan, (2013), Her Yönüyle C# 5.0, KODLAB, İstanbul.

GRIFFITHS, Ian, (2013), Programming C# 5.0, O'Reilly Media, California.

HALVORSON, M. (2010). Microsoft Visual Basic 2010 Step by Step, Microsoft Press.

KIZILÖREN, Tevfik, (2013), Her Yönüyle C, KODLAB, İstanbul.

SHARP, John, (2009), Microsoft Visual Studio 2008 Step By Step, çev. Ümit Tezcan, Arkadaş Yayınevi, Ankara.

SKEET, Jon, (2014), C# in Depth 3rd Edition, Manning Publication Co, New York.

KONTROL DEYİMLERİ

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr.
Y. Ziya AYIK

ÜNİTE 8

İÇİNDEKİLER

- IF-Else IF-End IF Deyim Grubu
- Basit IF
- Çok Şartlı IF
- İçi İçe IF
- Select Case - End Select Deyim Grubu

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Kontrol deyimlerini tanıyabilecek,
  - IF-Else IF-End IF ve
  - Select Case - End Select yapısını öğrenebilecek,
  - Program akışını bu yapıları kullanarak değiştirebileceksiniz.

GİRİŞ

Bir programın normal akışı kontrol deyimleri kullanılarak değiştirilebilir.

Programlamanın genel mantığı, işlemlerin kodlamanın ilk satırından son satırına doğru sıralı olarak gerçekleştirilmesidir. Ancak zaman zaman bu sıralı akışın değiştirilmesine veya program akışının istenilen başka bir satıra geçirilmesine ihtiyaç duyulur. Bu gibi durumlarda geçiş işlemini sağlayacak deyimler kullanılabilir. Bu deyimlerin tümüne kontrol deyimleri ya da kontrol elemanları denilmektedir.

Hemen hemen tüm programlama dillerinde program akışını yönlendirmek için çeşitli yapılar sunulmaktadır. Bu ünitede Visual Basic.Net içerisinde çeşitli karar verme mekanizmalarını kurmak için kullanılan *If* ve *Select Case* kontrol deyimlerine değinilecektir. Konular çeşitli örneklerle anlatılıp, ünite sonunda bir akış kontrol mekanizması kuracak bilgiye sahip olunması amaçlanmaktadır.

IF – ELSEIF – END IF DEYİM GRUBU

Kullanıldığı alan ve işlevi incelendiğinde If deyim grubu bir değişken ya da bir başka veri parçasının alacağı True veya False değerlerine göre hareket eden bir kod bloğu olarak tanımlanabilir. IF – ElseIF –End IF deyim grubunun temel kullanım amacı programda verilen koşula bağlı olarak akışının mevcut kod bloğuna geçmesini veya söz konusu bloğun işletilmeyip atlatılmasını sağlamaktır. Bu başlık altında If -Else If -End If deyim yapısı gruplandırılarak incelenmiştir.

Basit IF

Programlamada en fazla ihtiyaç duyulan şeylerden biri de değeri kullanıcıdan alınan veya değeri program esnasında hesaplanan çeşitli değişkenleri test etmektir. Bunu gerçekleştirmek için sıkça tercih edilen yöntemlerden biri de If deyimini kullanmaktır.

If deyimi programın genel akışını kontrol etmek amacıyla kullanılır. En basit kullanımıyla if deyimi aşağıdaki genel şablona göre kullanılır:

```
If koşul ifadesi Then
    İşlem
End If
```

Bu şablona uygun olarak aşağıdaki örnek verilebilir:

```
If 2 > 1 Then
MessageBox.Show("2 sayısı 1'den büyüktür")
End If
```

Visual Basic.Net'te koşul ifadeleri mutlaka Boolean tipinden bir değer döndürmelidir.

Bu örnekte koşul ifadesi True (doğru) değeri döndürür. Çünkü 2 sayısı 1'den büyüktür. Koşul ifadesi doğru değeri döndürdüğü için bir sonraki satırda kod işletilir.

Eğer koşuldan sonra yürütülecek tek deyim varsa aşağıdaki yazım stili de kullanılabilir:

```
If 2 > 1 Then MessageBox.Show("2 sayısı 1'den büyüktür")
```

Programlamada çoğu zaman bir koşul altında birden fazla ifade çalıştırmak istenebilir. Böyle bir durumda *If* kısmındaki ifadeler *End If* kullanılarak bir blok içerisine alınır. Örneğin bir koşulun sadece doğru gerçekleşmesi durumunda *If-End If* grubunun kullanımı şu şekilde olur:

```
If 100 > 1 Then
    MessageBox.Show("100 sayısı 1'den büyüktür")
    Label1.Text = "İşlem Tamamlandı"
End If
```

Yukarıdaki örnekte 100 sayısının 1 den büyük olup olmadığı kontrol edilmiş ve önce MessageBox'ta 100 sayısı 1'den büyüktür mesajı gösterilmiş sonrasında ise Label1 kontrolünün Text özelliğine İşlem Tamamlandı değeri atanmıştır. Yani 2 ifade çalıştırılmıştır ve bu yüzden End If bloğu kullanılmıştır.

If deyiminde parametre olarak verilen "koşul" ifadesi True (doğru) değer döndürüyor ise IF ile END IF arasında yer alan satırlar işletilir. Eğer koşul ifadesi False (yanlış) değer döndürüyor ise IF ile END IF arasında yer alan satırlar atlatılıp programın akışı END IF deyiminden sonra gelen ilk satıra geçer. Aşağıda verilen örnekte koşul şartı yanlış (False) değeri döndürdüğü için MessageBox'ın bulunduğu kod satırı işletilemeyecek ve ekranda herhangi bir mesaj görüntülenmeyecektir.

```
If 1 > 100 Then
        MessageBox.Show("1 sayısı 100'den büyüktür")
End If
```

If deyimi tek başına kullanıldığında sadece incelenen koşul True (doğru) olduğunda çeşitli işlemler yapılabilmektedir. Koşul ifadesinin False (yanlış) olduğu durumlarda çeşitli işlemler yapabilmek için If-Else yapısı kullanılır. If-Else yapısının kullanımı genel olarak şu şekildedir:

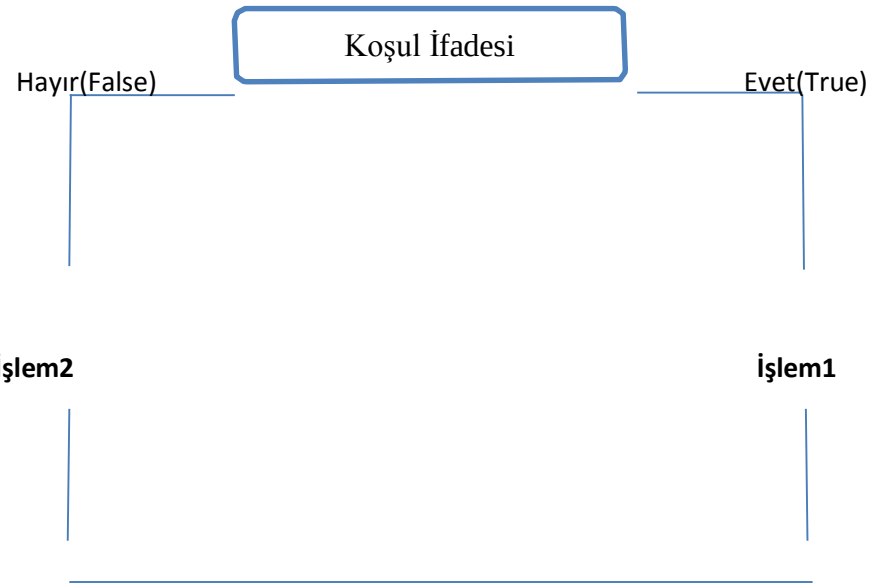
```
If koşul ifadesi Then
                İşlem1
Else
                İşlem2
End If
```

If-Else-End If kullanımında belirtilen koşul ifadesi False değeri döndürür ise ELSE'den sonra belirtilen işlem gerçekleştirilir.

Yukarıdaki şablonda bulunan koşul ifadesi True ya da False değeri döndürmelidir. Aksi takdirde program derlenmez. Koşul ifadesinin döndürdüğü değer eğer True ise İşlem1 yürütülür. Eğer dönen değer False ise İşlem2 yürütülür. Aşağıda verilen örnekte basit bir If-Else kod bloğunun nasıl çalıştığı daha iyi anlaşılabilir:

```
If 2 > 1 Then
MessageBox.Show("2 sayısı 1'den büyüktür")
Else
MessageBox.Show("2 sayısı 1'den küçüktür")
End If
```

2 sayısı 1'den büyük olduğu için 2>1 koşul ifadesi true değerini döndürecektir. Bu yüzden koşuldan sonra 2 sayısı 1'den büyüktür yazısını gösteren kod yürütülecektir. Bu örnekten hareketle if deyiminin akış diyagramı şu şekilde gösterilebilir:



Tıpkı If –End If deyim gruplarında olduğu gibi If-Else-End If deyim grubunun da farklı yazım stilleri bulunmaktadır. Aşağıdaki yazım stilinde Else ifadesine sadece _ simgesi eklenmiştir ve End If kullanımına gerek kalmamıştır. Ancak bu yazım stilinin birden fazla ifadelerde doğru çalışmayacağı unutulmamalıdır.

```
If 2 > 1 Then MessageBox.Show("2 sayısı 1'den büyüktür") Else _
MessageBox.Show("2 sayısı 1'den küçüktür")
```

Farklı yazım stilleri kullanılarak yazılan kod sayısı azaltılabilse de önemli olan kodun istenilen işi hızlı ve doğru yapmasıdır. Hatta kimi zaman If koşulundan sonra gelen ifade tek bir satırdan oluşsa bile bu ifadeyi *If End If* bloğunda yazmak mantıklı olacaktır. Böyle bir yapı özellikle çok sayıda ve karmaşık *If-Else* yapılarının olduğu programlarda yeni bir ifade ekleneceği zaman programcının kodu daha

rahat okumasını ve düzenlemeyi nerede yapacağını daha kolay görebilmesini sağlamaktadır.

Çok şartlı IF

Çok şartın var olduğu durumlarda IF – ElseIF – End IF deyim grubu kullanılır. Bu yapı içerisinde yer alan her bir Else If deyimi yanında olabilecek bir koşul belirtilir ve bu koşulun sağlanması durumunda yürütülecek işlemler sıralanır. Olabilecek bütün koşullar belirtildikten sonra bu şartların dışında kalan durumlar için de Else kullanılır. Bu sayede ortaya çıkması düşünülen veya düşünülmeyen bütün koşullar için gerçekleştirilecek işlemler belirlenmiş olur. If deyiminde parametre olarak verilen "koşul" True (doğru) ise belirtilen işlem gerçekleşecektir. If-Else If-End If deyim grubu aşağıda verilen şablona uygun olarak kullanılır:

```
If koşul1 Then
   İşlem1
ElseIf koşul2 Then
   İşlem2
ElseIf koşul3 Then
   İşlem3
ElseIf koşul4 Then
   İşlem4
Else
   İşlem5
End If
```

If yapısında ikiden fazla koşul söz konusu ise Else If kullanılmalıdır.

Yukarıdaki şablonda, birinci ilk if deyiminden koşul doğru ise İşlem1 çalıştırılır ve program akışı fF bloğunun sonunu belirleyen END IF deyiminin bulunduğu satırdan sonraki satıra geçer. İlk if deyimindeki koşul yanlış ise ikinci if deyimine geçilir ve ikinci if deyimindeki koşul doğru (true) ise işlemgrubu2 çalıştırılır ve program akışı yine End If deyiminin bulunduğu satırdan sonraki satıra geçer. Aynı mantık Else If deyimlerinin bulunduğu bütün bloklar için yürütülür. Else If bloklarından hiç biri doğru (true) değer döndürmez ise akış Else deyiminin bulunduğu bloğa geçer ve işlem5 çalıştırılır, daha sonra da End If deyiminden sonraki satırdan devam edilir.

Aşağıda If-Else If-End If deyim grubunun kullanıldığı bir örnek verilmiştir.

```
Dim K As Integer
K = TextBox1.Text
If K >= 0 And K <= 5 Then
Label1.Text = "Kısa mesafe attınız"
ElseIf (K > 5 And K < 10) Then
Label1.Text = "Orta mesafe attınız"
ElseIf (K >= 10 And K < 20) Then
Label1.Text = "Uzun mesafe attınız"
ElseIf (K >= 20 And K <= 30) Then
Label1.Text = "Çok uzun mesafe attınız"
Else
Label1.Text = "Geçersiz atış yaptınız"
End If
```

Örnekte görülen IF – Else IF – End IF yapısında x değişkeninin aldığı değere göre label nesnesine değer aktarılmaktadır. K değişkeninin aldığı değer 0 ile 5 metre arasında ise label nesnesine “Kısa mesafe attınız”, 6 ile 9 arasında ise “Orta mesafe attınız”, 10 ile 19 arasında ise “Uzun mesafe attınız” ve 20 ile 30 arasında ise “Çok uzun mesafe attınız” değerleri aktarılmaktadır. K değişkeninin yukarıda belirtilenler dışında yani 0’dan küçük veya 30’dan büyük herhangi bir değer alması durumunda ise label nesnesine “Geçersiz atış yaptınız” ifadesi aktarılacaktır.

İç İçe If

İçi içe kullanılan IF yapılarında ilk önce başlatılan IF yapısının END IF deyimi en sonda yazılmalıdır.

Kararların hiyerarşik bir yapı oluşturduğu durumlarda tek bir If-Else If-End If yapısı yeterli olamayabilir. Bu gibi durumlarda If yapıları içe içe kullanılabilir. If-Else If-End If yapısını içe içe kullanmanın bir üst sınırı yoktur. Ancak bu yapının iç içe çok fazla sayıda kullanılmasının kod yazımını ve kodun anlaşılabilirliğini zorlaştıracağı unutulmamalıdır. İki tane If yapısı iç içe kullanılacaksa biri diğerinin içine yerleştirilmelidir. Yani ikinci yapı birinci If deyiminden sonra başlatılmalı ve birinci yapı End If deyiminden önce sonlandırılmalıdır. Aynı kural ikinci If yapısından sonra oluşturulabilecek üçüncü ve dördüncü yapılar için de geçerlidir.

```
Deyim       : IF
Kullanımı   : IF şart THEN
                  işlemgrubu1
                  IF şart THEN
                     işlemgrubu1.1
                  ELSE
                     işlemgrubu2.2
                  END IF
              ELSE
                  işlemgrubu2
              END IF
```

Veya

```
Kullanımı   : IF şart THEN
                  işlemgrubu1
              ELSE
                  işlemgrubu2
                  IF şart THEN
                     işlemgrubu2.1
                  ELSE
                     işlemgrubu2.2
                  END IF
              END IF
```

If yapılarında tek seferde Else If deyimlerinden veya Else deyiminden sadece bir tanesindeki işlem gerçekleştirilebilir.

```
Dim yas As Integer
yas = TextBox1.Text
  If yas => 4 Then
    If yas < 12 Then
       Label1.Text = "çocuk"
    Else
       Label1.Text = "yetişkin"
    End If
  Else
    Label1.Text = "bebek"
  End If
```

Yukarıdaki kod kümesinde ilk If ifadesi yas değişkeninin değerinin 4'ten küçük olup olmadığını kontrol etmektedir. Dolayısıyla yas değişkeni değeri eğer 4'ten küçükse koşul ifadesi yanlış değer döndürecek ve ilk If yapısındaki Else deyiminin altındaki kod yürütülecektir. Yani Label1 nesnesine bebek yazdırılacaktır. Eğer yas değişkeni değeri 4 ya da 4'ten büyükse ikinci bir If yapısı

ile karşılaşılacaktır. Bu sefer Label1 nesnesine yas değişkeni 12'den küçükse çocuk, 12 veya üzeri ise yetişkin yazdırılacaktır.

Bireysel Etkinlik

- İç içe If yapısı ile ilgili örneği bilgisayarınızda çalıştırınız ve yas değişkeni değerini her seferinde farklı girerek Label metnini kontrol ediniz.

SELECT CASE - END SELECT DEYİM GRUBU

Select Case – End Select yapısı, Select Case deyimi ile başlar, End Select deyimi ile sona erer.

Program akışının şarta bağlı olarak değiştirilmesini veya bir işlem grubunun şarta bağlı olarak çalıştırılmasını sağlayan diğer bir yapı Select Case – End Select yapısıdır. Bir değişkenin aldığı değer ya da değer aralığına göre seçim yapılması gerektiği durumlarda kullanılır. Select Case ile yapılan işlemler If-Else yapıları ile de yapılabilseler de Select Case deyimi daha rahat okunduğu için programcılar tarafından bazı durumlarda tercih edilir. Özellikle hata kontrolünü kolaylaştırması ve rahat okunması, Select Case deyiminin en büyük avantajıdır.

Select Case – End Select yapısı, Select Case deyimi ile başlar, End Select deyimi ile sona erer. Select Case deyiminden sonra yapılacak karşılaştırmalarda kullanılacak bir kontrol değişkeni kullanılmaktadır. Daha sonra Case deyimi ile birlikte şartlar veya ifadeler belirtilmektedir. Select Case deyiminden sonra verilen kontrol deyimi Case deyimleri ile birlikte verilen şartlardan hangisi ile uyumlu olursa o Case yapısı içerisinde bulunan işlemler çalıştırılacaktır. Daha sonra değişken ile uyum göstermeyen Case yapıları atlanarak program akışı End Select deyiminden sonraki satıra geçecektir. Case yapılarında verilen şartlardan hiçbiri değişken ile uyumlu olmaz ise Case Else deyimine geçilir. Case Else deyimindeki işlemler çalıştırıldıktan sonra program akışı yine End Select deyiminin bulunduğu satırdan sonraki satıra geçer. Select Case, Case ve Case Else deyimlerinin bir arada kullanıldığı genel şablon aşağıda görülmektedir:

```
Select Case değişken
    Case şart1
        işlemgrubu1
    Case şart2
        işlemgrubu2
    Case şart3
        işlemgrubu3
    Case Else
        işlemgrubu4
End Select
```

Case anahtar kelimesinin yanındaki ifadeler ancak sabit olabilir. Değişken olamaz.

Bazı durumlarda sadece belirli koşullara göre işlemlerin yapılması istenebilir. Böyle bir durumda her Case deyimi için bir koşul oluşturulur. Select Case deyiminde kullanılan değişken hangi Case şartına uygun olursa o Case deyimindeki işlem grubu çalıştırılır. Aşağıda verilen şablonda sadece belirli şartlarda çalışması istenen işlem gruplarının Select Case yapısında nasıl kullanıldığı görülmektedir.

```
Select Case değişken
        Case şart1
    işlemgrubu1
        Case şart2
    işlemgrubu2
        Case şart3
    işlemgrubu3
End Select
```

Select Case ifadesi ile ilgili bir örnek vererek kullanımını pekiştirelim. Aşağıdaki örnekte bir TextBox'a girilen değer bir değişkene atanmış ve bu değişkenin değerine göre MessageBox ile kullanıcıya mesaj gösterilmesi istenmiştir:

```
Dim karakter As String
karakter = TextBox1.Text
Select Case karakter
  Case "A"
    MsgBox("Girdiğiniz harf A")
  Case "B"
    MsgBox("Girdiğiniz harf B")
  Case "C"
    MsgBox("Girdiğiniz harf C")
End Select
```

Case deyimlerinde tek şart veya birden çok şart belirtilebilir.

Karakter değişkenine A atanırsa Case A'daki "Girdiğiniz harf A " , B atanırsa Case B'deki "Girdiğiniz harf B ", C atanırsa Case C'deki "Girdiğiniz harf C " mesajı görüntülenecektir.

Bazı durumlarda bir Case anahtar kelimesinin yanında birden fazla koşul belirtilebilir. Böyle bir durumda her Case deyimi için bir veya birden çok koşul oluşturulur. Select Case deyiminde kullanılan değişken, bir Case'de belirtilen

koşullardan herhangi biri ile uyumlu olursa o Case deyimindeki işlem grubu çalıştırılır. Birden fazla koşulun kontrol edildiği Select Case yapısı aşağıdaki şablona göre kullanılır.

```
Select Case değişken
    Case şart1, şart2, şart3
        işlemgrubu1()
    Case şart4, şart5
        işlemgrubu2()
    Case şart6
        İşlemgrubu2()
End Select
```

Aşağıdaki örnekte Sayi isimli değişkene 30 değeri atanmış ve daha sonra Select Case deyimi kullanılarak bu değişkenin değeri kontrol edilmiştir. 3.Case ifadesinde 30 belirtildiği için Sayımız 30 mesajı kullanıcıya gösterilecektir.

```
Dim sayi As Integer = 30
    Select Case sayi
        Case 10, 20, 50
        MsgBox("Sayımız 10, 20 veya 50")
        Case 2, 3
        MsgBox("Sayımız 2 veya 3")
        Case 30
        MsgBox("Sayımız 30")
End Select
```

CASE lerde değişkenlerin başlangıç ve bitiş değerleri belirtilerek şart belirtilebilir.

Select Case yapısı ile sadece sabit bir değer değil aynı zamanda bir değer aralığı da kontrol edilebilir. Bu kullanımda Case deyimi için koşul başlangıç değeri ve koşul bitiş değeri belirlenir. Select Case deyiminde kullanılan değişken, Case'lerde belirtilen aralıklardan hangisiyle uyumlu olursa o Case deyimindeki işlem grubu çalıştırılır. Select Case yapısında değer aralığı kullanımı aşağıdaki şablona göre yapılır

```
Select Case değişken
   Case şart1 To şart2
      işlemgrubu1
   Case şart3 To Şart4
      işlemgrubu2
   Case şart5 To şart6
      işlemgrubu3
End Select
```

Aşağıda verilen örnekte ortalama değişkenine atanan değer 0 ile 49 aralığında olursa, birinci sırada verilen Case deyimi çalıştırılacak ve "FF aldınız" mesajı görüntülenecektir. Notlar değişkenine atanan değer 50 ile 80 aralığında olursa, ikinci sırada verilen Case deyimi çalıştırılacak ve "CC aldınız " mesajı elde edilecektir. Notlar değişkenine 81 ile 100 aralığında değer girilirse, üçüncü sırada verilen Case deyimi çalıştırılacak ve "AA aldınız" mesajı görüntülenecektir.

```
Dim ortalama As Double
ortalama = TextBox1.Text
Select Case ortalama
   Case 0 To 49
      MsgBox("FF aldınız")
   Case 50 To 50
      MsgBox("CC aldınız")
   Case 81 To 100
      MsgBox("AA aldınız")
        End Select
```

Select Case yapılarında en sık kullanılan anahtar yapılarından biri de *"Is"* kelimesidir. Is anahtar kelimesi karşılaştırma operatörleri birlikte kullanılarak, değişkenin hangi şartlar aralığında olduğu ve hangi Case deyimine uygun olduğu belirlenmektedir. *"Is" operatörü (>,<,>=,<=,=) gibi operatörlerden uygun olanı ile değişkenin hangi değere sahip olduğunun belirlenmesini sağlar.* Bu sayede ilgili Case deyimindeki işlem grubunun çalıştırılması sağlanmış olur. Select Case yapısında Is anahtar kelimesi ile karşılaştırma operatörlerinin kullanımı aşağıdaki şablona göre gerçekleşir:

```
Select Case değişken
        Case Is <= şart1, Is = şart2
           işlemgrubu1
        Case Is = şart3, Is >= şart4
           işlemgrubu2
        Case şart3 > şart5
           işlemgrubu3
End Select
```

Case anahtar kelimeleri yanında karşılaştırma kullanılabilir ve mantıksal işlemler yapılabilir.

Select Case yapısında Is anahtar kelimesinin kullanımını daha iyi anlayabilmek için aşağıdaki örneği inceleyelim. Örnekte hava değişkenine TextBox1'den gelen değer 0'dan küçük ise "Çok soğuk", 10'dan küçük ise "Soğuk", 10 ile 19 aralığında ise "Ilık" mesajı görüntülenecektir. Atanan değer 20 ve daha fazla ise hava değişkenine "Sıcak" verisi atanmış olacaktır.

```
Dim hava As Double
hava = TextBox1.Text
Select Case hava
   Case Is <= 0
      MsgBox("Çok soğuk")
   Case Is < 10
      MsgBox("Soğuk")
   Case Is >= 10, Is < 20
      MsgBox("Ilık")
   Case Is >= 20
      MsgBox("Sıcak")
        End Select
```

Özetle Select Case yapısı uzun ve karmaşık If döngülerini daha anlaşılır, kolay ve zevkli hâle getiren bir alternatif olarak görülebilir. Select Case kullanılarak çözülen tüm problemler If yapısı kullanılarak da çözülebilse de bazı durumlarda Select Case kullanmak hem kod okunabilirliğini arttıracak hem de hata riskini azaltacaktır.

Özet

- Kontrol deyimleri, Visual Basic .NET programlama dilinde program akışına müdahale edebilmek amacıyla yaygın olarak kullanılırlar. En çok bilinen ve kullanılan kontrol deyimleri, IF-ELSE IF–END IF ve SELECT CASE-END SELECT deyim gruplarıdır.
- IF-ELSE IF–END IF deyim grubu şartın tek durumuna göre basit IF veya çok şartlı IF şeklinde kullanılabilirler. Ayrıca karar yapısının bir veya daha çok olması durumuna göre de tek veya iç içe IF yapıları şeklinde oluşturulabilirler.
- Select Case yapısı uzun ve karmaşık If döngülerini daha anlaşılır, kolay ve zevkli hâle getiren bir alternatif olarak görülebilir. Select Case kullanılarak çözülen tüm problemler If yapısı kullanılarak da çözülebilse de bazı durumlarda Select Case kullanmak hem kod okunabilirliğini arttıracak hem de hata riskini azaltacaktır.

DEĞERLENDİRME SORULARI

1. Dim x As Integer
   x = 5
   If x ^ 2 <= 15 Then
   Label1.Text = "Merhaba"
   Else
   Label1.Text = "Son"
   End
   End If

   Yukarıdaki IF – ElseIF – End IF yapısı çalıştırıldığında elde edilen sonuç hangisi olur?

   a) Ekrana “Merhaba” yazar ve çıkar.
   b) Ekrana “15” yazar.
   c) Döngü hatalıdır.
   d) Ekrana “Son” yazar ve programdan çıkar.
   e) Ekrana “Son” yazar.

2. Dim sıcaklık As Double
   sıcaklık = InputBox("Sıcaklık değerini giriniz")
   Select Case sıcaklık
   Case Is < 20
   Label1.Text = "Hava soğuk"
   Case 20 To 30
   Label1.Text = "Hava Sıcak"
   Case Else
   Label1.Text = "Hatalı veri"
   End Select

   Yukarıdaki Select Case – End Select yapısı için seçeneklerdeki ifadelerden hangisi doğrudur?

   a) sıcaklık=20 ise Hava Sıcak
   b) sıcaklık=35 ise Hava Sıcak
   c) sıcaklık=25 ise Hava Soğuk
   d) sıcaklık<20 ise Hatalı veri
   e) sıcaklık=0 ise Hatalı veri

3. Dim X, Y, K, L As Integer
   X = 6 : Y = 2
   If X + Y = 15 Then
   K = X + Y
   L = X - Y
   MsgBox(K & " " & L)
   Else
   MsgBox(X + 3 & " " & Y)
   End If

   Yukarıdaki program kodu çalıştırıldığında elde edilen sonuç hangisi olur?
   a) 8 4
   b) 6 3
   c) 9 2
   d) 9 3
   e) 6 9

4. Dim x As String
   x = TextBox1.Text
   If x= "Ahmet" Then MsgBox("Öğrenci") Else MsgBox("Memur")
   Yukarıdaki program kodu için hangisi doğrudur?
   a) İsim değişkeni Ahmet'e eşit ise ekrana MsgBox yazdırılır.
   b) İsim değişkeni Mehmet'e eşit ise ekrana Öğrenci yazdırılır.
   c) İsim değişkeni Ahmet'e eşit ise ekrana Memur yazdırılır.
   d) İsim değişkeni Ayşe'ye eşit ise ekrana hiçbir şey yazdırılmaz.
   e) İsim değişkeni Mehmet'e eşit ise ekrana Memur yazdırılır.

5. IF – ElseIF – End IF yapısıyla ilgili aşağıdakilerden hangisi söy lenem ez?
   a) IF – ElseIF – End IF yapısı iç içe kullanılabilir.
   b) IF – ElseIF – End IF yapısında End IF kullanılması zorunlu değildir.
   c) IF – ElseIF – End IF yapısında Else IF kullanılması zorunlu değildir.
   d) IF – ElseIF – End IF yapısında Else kullanılması zorunlu değildir.
   e) IF – ElseIF – End IF yapısında IF yapının başında kullanılmalıdır.

6.
```
Dim a1, a2, a3 As Integer
        a1 = Val(TextBox1.Text)
        a2 = Val(TextBox2.Text)
        a3 = Val(TextBox3.Text)
        If (a1 > 0 And a2 > 0) Then
            If (a3 = 90) Then
                MessageBox.Show("Üçgen1")
            ElseIf (a3 >= 91) Then
                MessageBox.Show("Üçgen2")
            Else
                MessageBox.Show("Üçgen3")
            End If
        End If
```

Yukarıdaki kodlamayla ilgili hangisi doğrudur?

a) A1<0 ise ekrana Üçgen1 yazar.
b) A2<0 ise ekrana Üçgen2 yazar.
c) A3<>90 (eşit değil) ise ekrana Üçgen3 yazar.
d) A1<0 ise ekrana hiçbir şey yazmaz.
e) A1<0 ve A3>=91 ise ekrana Üçgen2 yazar.

7.
```
If TextBox1.Text = " " And TextBox1.Text = " " Then
                    Button1.Enabled = False
                    TextBox2.Text = " "
                Else
                    Button1.Enabled = True
            End If
```

Yukarıdaki kodlamayla ilgili hangisi söylenebilir?

a) TextBox1 nesnesi boş TextBox2 nesnesi dolu ise Button1 nesnesi kullanılamaz.
b) TextBox2 nesnesi boş TextBox1 nesnesi dolu ise Button1 nesnesi kullanılamaz.
c) TextBox1 ve TextBox2 nesneleri boş ise Button1 nesnesi kullanılamaz.
d) Button1 nesnesi kullanılarak TextBox1 nesnesine veri girişi yapılır.
e) TextBox1 nesnesi boş ise Button1 nesnesi form üzerinde görüntülenemez.

8.
```
Dim X, Y, Z As Double
        X = 5 : Y = 10
        Select Case X + Y
                Case Is < 10
                                Z = Y - 5
                Case 11 To 30
                        Z = Y / X
                Case Else
                                Z = X ^ 2
        End Select
        MsgBox(Z)
```

Yukarıdaki Select Case – End Select yapısı çalıştırıldığında elde edilen sonuç hangisi olur?

a) 25
b) 0.5
c) 2
d) 15
e) 5

9.
```
Dim sayı1,sayı2 As Double
Sayı1 = TextBox1.Text
    Sayı2 = TextBox2.Text
If RadioButton1.Checked = True Then
TextBox1.Text = sayı1 - sayı2
 End If
If RadioButton2.Checked = True Then
TextBox2.Text = sayı1 * sayı2
End If
```

Yukarıdaki kodlamayla ilgili hangisi doğrudur?

a) RadioButton1 seçiliyse TextBox1'e sayı1 yazdırılır.
b) RadioButton2 seçiliyse sayı1*sayı2 sonucuTextBox2'ye yazdırılır.
c) RadioButton2 seçiliyse sayı1-sayı2 sonucuTextBox2'ye yazdırılır.
d) RadioButton1 seçili değilse sayı1*sayı2 sonucuTextBox2'ye yazdırılır.
e) RadioButton1 ve RadioButton2 seçiliysesayı1-sayı2 sonucuTextBox1'e sayı1*sayı2 sonucuTextBox2'ye yazdırılır.

10. Dim S1 As Double
    S1 = InputBox("Bir sayı giriniz.")
    Select Case S1
        Case 0 To 20
            MsgBox("Merhaba")
        Case 50 To 100
            MsgBox("Nasılsın")
    End Select

    Yukarıdaki kodlamayla ilgili hangisi söylenebilir?

    a) S1 değeri 20 ise ekrana hiçbir şey yazmaz.
    b) S1 değeri 0 ise ekrana Merhaba yazar.
    c) S1 değeri 75 ise ekrana hiçbir şey yazmaz.
    d) S1 değeri 101 ise ekrana Nasılsın yazar.
    e) S1 değeri -5 ise ekrana Merhaba yazar.

Cevap Anahtarı

1.d, 2.a, 3.c, 4.e, 5.b, 6.d, 7.c, 8.c, 9.b, 10.b

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Algan, Sefer, (2009), *Her Yönüyle C#*, Pusula Yayıncılık, İstanbul.

Ayık Y. Ziya. (2009). Algoritma ve Programlama Metodolojisi, MuratHan, Trabzon

Ayık Y. Ziya, (2011), Atatürk Üniversitesi, Uzaktan Eğitim Merkezi, Görsel Programlama I Ders Notları

Kızılören, Tevfik, (2013), Her Yönüyle C*,* KODLAB, İstanbul.

Yanık Memik(2010), Visual Basic 10, Seçkin Yayınevi, Ankara

DÖNGÜLER VE ZAMANLAYICILAR

İÇİNDEKİLER

- Döngüler
- Döngü Oluşturma Kuralları
- For Next Döngüsü
- For Each Next Döngüsü
- Do While Loop Döngüsü
- Döngüden Çıkma (Exit)
- Döngüye Devam (Continue)
- İç İçe Döngüler
- Zamanlayıcılar

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
- Tekrarlı programlama yapılarını, yani döngüleri kavrayabilecek,
- İki farklı For döngüsünün kullanımını öğrenebilecek,
- While döngüsünün kullanımını öğrenebilecek,
- Bir döngüyü ani olarak sonlandırabilmeyi öğrenebilecek,
- Bir döngünün bir sonraki iterasyonuna herhangi bir anda geçebilmeyi kavrayabilecek,
- İç içe döngüleri anlayabilecek,
- Zamanlayıcıları ve Timer isimli kontrol nesnesinin kullanımını kavrayabileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA 1

Yrd. Doç. Dr.
Deniz DAL

ÜNİTE
9

GİRİŞ

Birçok programlama uygulamasında bazı işlemlerin belirli bir sayıda veya bir koşul sağlandığı müddetçe tekrar ettirilmesi gerekmektedir. Bu amaçla kullanılan programlama yapılarına ise döngü adı verilmektedir. Döngülere günlük hayatımızda da rastlamak mümkündür. Örneğin diş fırçalama, diş fırçasının ağız içerisinde farklı dişler üzerinde aşağı yukarı hareketinden oluşan tekrarlı bir işlemdir. Bu işlem esnasında harcanan tekrarlama süresi için iki farklı senaryodan bahsedilebilir: (1) Diş fırçalama için planlanan süre (tekrarlama süresi), örneğin 1 dakika olarak, başlangıçta bellidir. (2) Diş fırçalama için planlanan süre başlangıçta belli değildir, başka bir deyişle önceden kestirilememektedir; dolayısıyla bu işlemin sona ereceği ana (örneğin dişlerin bembeyaz olduğuna kanaat getirildiği ana), ancak tekrar içerisinde karar verilmesi zorunludur. Döngülerin günlük hayatımızdaki kullanımına bir başka örnek ise dönme dolaptır. Dönme dolap insanları bir noktadan alıp, yüksekçe bir noktaya çıkardıktan ve bu esnada panoramik bir şehir turu attırdıktan sonra başladığı noktaya geri getirmektedir. Bu işlem gün içerisinde farklı insanlarla defalarca tekrar edilmektedir.

Bu ünitemizde bir önceki paragrafımızda bahsedilen (1) numaralı senaryoyu programlarımızda kullanabilmemize imkân sağlayan For döngülerinden, yani tekrarın belirli bir sayı adedince gerçekleştirilmesine izin veren programlama yapılarından bahsedilecektir. Ayrıca (2) numaralı senaryoyu programlarımızda gerçekleştirebilmemizi mümkün kılan While döngülerine, yani tekrarın bir koşula bağlı olarak işletilmesini sağlayan yapılara da bu ünite kapsamında değinilecektir. Bu ünitenin son konusu ise zamanlayıcılar olacaktır.

Bu ünite içerisinde rastlanacak tüm değişkenler Deve Notasyonu kullanılarak oluşturulmuştur; ayrıca tüm Visual Basic uygulamaları Microsoft Visual Studio Enterprise 2015 platformu üzerinde geliştirilmiştir ve test edilmiştir.

DÖNGÜLER

Bir döngünün tekrar eden her bir adımı iterasyon olarak adlandırılır.

Tekrarlı programlama yapıları olarak da adlandırılabilecek döngüleri daha iyi anlayabilmek için bir sonraki bölümde anlatılacak döngü oluşturma kurallarını öğrenmekte büyük fayda vardır.

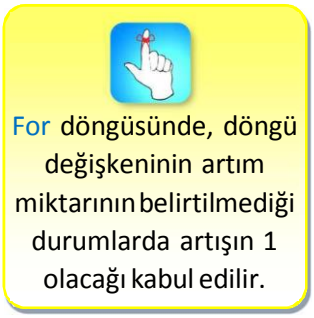
Döngü Oluşturma Kuralları

1 - Döngüyü kontrol edecek bir döngü değişkeni oluşturulur.
2 - Döngü değişkeninin başlangıç değeri belirlenir.
3 - Döngü değişkeninin bitiş değeri belirlenir.
4 - Döngü değişkeninin bitiş değerine ulaşıp ulaşmadığı test edilir.
5 - Tekrarı istenen işlem gerçekleştirilir.
6 - Döngü değişkeni bir sonraki iterasyonda kullanılmak üzere adım miktarı kadar artırılır ya da azaltılır.

For Next Döngüsü

```
For donguDegiskeni=baslangicDegeri To bitisDegeri Step artimMiktari
    Tekrar Deyimi(leri)
Next donguDegiskeni
```

Yukarıdaki şablon For döngüsünün Visual Basic programlama dilindeki söz dizilimidir (syntax). Bu söz diziliminde mavi renk ile koyulaştırılmış kelimeler anahtar kelimelerdir. Her For döngüsünü kontrol edecek bir döngü değişkeni (diğer bir adıyla sayaç), bu değişkenin bir başlangıç değeri, bir bitiş değeri ve bir de kullanımı isteğe bağlı olan bir artım miktarı (negatif de olabilir) vardır. Döngü değişkeninin artım miktarının belirtilmediği durumlarda, yani "Step artimMiktari" ifadesinin şablonun ilk satırında kullanılmadığı durumlarda, her bir döngü iterasyonunda artışın 1 olacağı kabul edilir. Artım miktarı pozitif olan bir For döngüsü döngü değişkeninin başlangıç değerinden başlar ve bu değer bitiş değerinden küçük veya eşit olduğu müddetçe For Next bloğu/gövdesi arasındaki işlemi(leri) tekrarlar ve döngünün her bir iterasyonunda döngü değişkeninin değerini artım miktarı kadar artırır. Öte yandan artım miktarı negatif olan bir For döngüsü döngü değişkeninin başlangıç değerinden başlar ve bu değer bitiş değerinden büyük veya eşit olduğu müddetçe For Next bloğu arasındaki işlemi(leri) tekrarlar ve döngünün her bir iterasyonunda döngü değişkeninin değerini artım miktarı kadar azaltır.

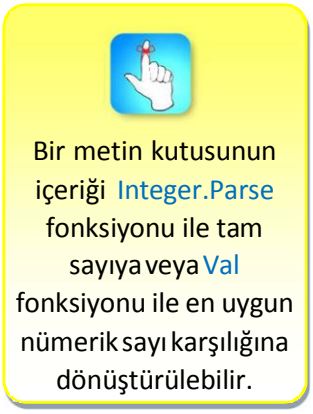


Örnek

- 1 sayısı ile kullanıcıdan bir metin kutusu aracılığıyla alacağı bir tam sayı arasındaki ardışık tam sayıları bir mesaj kutusu içerisinde sırasıyla kullanıcıya sunan bir Visual Basic programı yazınız.

Çözüm:

```
Dim sayac As Integer
Dim ustLimit As Integer = Integer.Parse(TextBox1.Text) For
sayac = 1 To ustLimit 'For sayac = 1 To ustLimit Step 1
    MsgBox(sayac)
Next sayac
```



Yukarıdaki kod parçasında ustLimit isimli değişkenin değerinin 5 olduğunu, yani uygulamanın metin kutusuna 5 değeri girilerek çalıştırıldığını kabul edelim. For döngüsü öncelikle sayac isimli değişkene 1 değerini atar. Sonrasında 1 sayısını ustLimit ile karşılaştırır; 1 ≤ 5 koşulu sağlandığı için döngü gövdesine girilir ve sayac'ın mevcut değeri olan 1 mesaj kutusu üzerinden kullanıcıya sunulur. Döngü gövdesinde gerçekleştirilen işlem sona erdiği için de Next anahtar kelimesi ile karşılaşılır ve program akışı yeniden For satırına yönlendirilir. Döngü değişkeni olan sayac'ın değeri artım miktarı olan 1 kullanılarak 1 artırılır. (For satırında Step 1 kullanılmadığına, kullanılsa da sonucun değişmeyeceğine dikkat ediniz.) 2 ≤ 5 koşulu sağlandığı için döngü gövdesine yeniden girilir ve bu sefer de sayac'ın mevcut değeri olan 2 mesaj kutusu üzerinden kullanıcıya aktarılır. Bu işlem sayac'ın değerinin 6 olması durumuna kadar devam eder ve 6 ≤ 5 koşulu sağlanmadığı için döngü dışına çıkılır, başka bir deyişle program akışı Next ile başlayan satırın bir alt satırına yönlendirilir.

Başlangıç değeri bitiş değerinden büyük ama artım miktarı pozitif bir For döngüsü sıfır kere işletilir, yani işletilmez.
Örnek:

```
For sayac=5 To 1
```

Başlangıç değeri bitiş değerinden küçük ama artım miktarı negatif bir For döngüsü sıfır kere işletilir, yani işletilmez.
Örnek:

```
For sayac=1 To 5 Step -1
```

Örnek

•Kullanıcıdan iki metin kutusu aracılığıyla alacağı bir alt limit ile bir üst limit arasındaki çift sayıları büyükten küçüğe bir mesaj kutusu içerisinde kullanıcıya sunan bir Visual Basic programı yazınız. Alt limitin üst limitten küçük girileceğinin garanti edildiği kabulünü yapabilirsiniz.

Çözüm:

```
Dim altLimit As Integer = Integer.Parse(TextBox1.Text)
Dim ustLimit As Integer = Integer.Parse(TextBox2.Text)
For sayac As Integer = ustLimit To altLimit Step -1
    If (sayac Mod 2) = 0 Then
        MsgBox(sayac)
    End If
Next sayac
```

Bu uygulamanın çözümünde kullanılan algoritma çift sayıları tespit etmek için ilgili aralıktaki tüm sayıları 2'ye tam olarak bölünüp bölünemedikleri noktasında bir teste tabi tutmuştur. Bulunan çift sayıların büyükten küçüğe kullanıcıya sunulması istendiği için de döngü değişkeni üst limitten başlatılmış ve alt limite ulaşıncaya kadar her bir iterasyonda 1 azaltılmıştır.

For Each Next Döngüsü

For Each döngüsü her bir iterasyonda bir dizi/grup/topluluk içerisindeki bir eleman için döngü gövdesindeki deyimleri tekrar etmek amacıyla kullanılır.

```
For Each eleman In grup
        Tekrar Deyimi(leri)
Next eleman
```

For Each döngüsü kullanılarak bir Form uygulaması üzerinde tanımlanmış araç kutusu (toolbox) kontrol nesneleri taranabilir.

Yukarıdaki şablon For Each döngüsünün Visual Basic programlama dilindeki söz dizilimidir. Bu söz diziliminde mavi renk ile koyulaştırılmış kelimeler anahtar kelimelerdir.

Örnek

Doğu Anadolu Bölgesi'nin sesli harfle başlayan şehirlerini içeren bir string dizisi oluşturunuz. Daha sonra bu şehirleri bir For Each döngüsü ile tarayarak bir mesaj kutusu içerisinde sırasıyla kullanıcıya sunacak bir Visual Basic programı yazınız

Çözüm:

```
Dim sehirler() As String = {"Ağrı", "Ardahan", "Elazığ", "Erzincan", "Erzurum",
"Iğdır"}
Dim sehir As String
For Each sehir In sehirler
    MsgBox(sehir)
Next sehir
```

Çözümdeki For Each döngüsü ilk iterasyonda sehirler dizisinin ilk elemanı olan Ağrı isimli stringi ve son iterasyonda da sehirler dizisinin son elemanı olan Iğdır stringini işlemektedir.

Örnek

- Kullanıcıdan bir metin kutusu aracılığıyla temin edeceği bir metnin her bir karakterini sırasıyla bir mesaj kutusu içerisinde kullanıcıya sunan bir Visual Basic programı yazınız.

Çözüm:

```
For Each karakter As Char In TextBox1.Text MsgBox(karakter)
Next karakter
```

Do While Loop Döngüsü

Bir döngünün kaç kez tekrar edeceğinin peşinen bilindiği uygulamalarda For döngüsünden, tekrarın bir koşul sağlandığı müddetçe gerçekleşeceği türden uygulamalarda ise While döngüsünden faydalanabileceğimizden Giriş bölümünde bahsetmiştik. Şimdi While döngülerinin nasıl kullanıldığını öğrenelim.

```
Do While (kosul)
Tekrar Deyimi(leri)
Loop
```

Yukarıdaki şablon Do While döngüsünün Visual Basic programlama dilindeki söz dizilimidir. Bu söz diziliminde mavi renk ile koyulaştırılmış kelimeler anahtar kelimelerdir. While döngüsü, yukarıdaki şablonda kosul ile belirtilen bir Boolean ifadenin durumuna göre işletilir veya işletilmez. Boolean ifade true (doğru) değer üretiyorsa döngü gövdesine girilir, false (yanlış) değer üretiyorsa döngü gövdesine girilmez. kosul oluşturmak için kullanılabilecek Boolean ifadeler büyüklük veya eşitlik gibi ilişkisel operatörler içerebilir. Benzer şekilde bu amaçla ve, veya gibi mantıksal operatörlerden de faydalanılabilir. Döngünün herhangi bir iterasyonunda gövdedeki tekrar deyimleri işletildikten sonra Loop anahtar kelimesi ile karşılaşılır ve program akışı koşulun sağlanıp sağlanmadığının test edilebilmesi için yeniden döngünün başına yönlendirilir. Döngünün sonsuz kere tekrar etmemesi için kosul içerisinde kullanılan döngü değişkeninin yeni bir iterasyona başlamadan hemen önce güncellenmesi gerekir. Aksi takdirde kos sürekli olarak true değer üretir ve döngü sonlanmaz. Özetlemek gerekirse, ko

içerisinde kullanılan döngü değişkenine döngü satırından önce bir ilk değer atanmalıdır ve bu değişken döngü içerisinde yeni bir iterasyon öncesinde güncellenmelidir.

Örnek

•Erzurum ilinin nüfusunun 500000 olduğunu ve her yıl mevcut nüfusun %5'i kadar bir büyüme gerçekleştirdiğini kabul ediniz.
Nüfusun kaç yıl sonra 1500000'u aşacağını bulacak ve bir mesaj kutusu içerisinde kullanıcıya sunacak bir Visual Basic programı yazınız.

Çözüm:

```
Dim nufus As Double = 500000 'Döngü değişkeni Dim sayac
As Integer = 0
Do While (nufus <= 1500000)
nufus = nufus * 1.05 'nufus=nufus+(nufus*5)/100 sayac =
sayac + 1
Loop
MsgBox(sayac & " yıl sonra beklenen nüfusa ulaşılır")
```

Yukarıdaki çözümün bir For döngüsü kullanılarak efektif bir şekilde gerçekleştirilemeyeceğine dikkat ediniz.

Örnek

Aşağıdaki For döngüsünü While döngüsüne dönüştürünüz.

```
For sayac As Integer = 1 To 100 Step 7 MsgBox(sayac)
Next sayac
```

Çözüm:

```
Dim sayac As Integer = 1 Do While
(sayac <= 100)
MsgBox(sayac) sayac = sayac + 7
Loop
```

Döngüden Çıkma (Exit)

Bazı uygulamalarda belirli bir koşul gerçekleştiğinde içerisinde bulunulan döngünün kalan iterasyonlarının işletilmemesi ve döngünün sonlandırılması gerekebilir. Bu amaçla For döngüleri içerisinde Exit For ve While döngülerinde de Exit Do deyimlerinden faydalanılabilir. Böyle bir durumda söz konusu koşulun sağlanıp sağlanmadığı bir If deyimi ile test edilebilir ve ilgili Exit komutu If gövdesinde çalıştırılabilir. Exit komutu işletildiğinde program akışı içerisinde bulunulan dön

sonraki ilk satıra yönlendirilir. (For döngülerinde ilgili Next satırının bir alt satırına, While döngülerinde ilgili Loop satırının bir alt satırına)

Örnek

•1 ile 1000 arasındaki tam sayılardan 3, 5 ve 7 ile tam olarak bölünebilen en küçük tam sayıyı bulacak ve bir mesaj kutusu içerisinde kullanıcıya sunacak bir Visual Basic programı yazınız.

Çözüm:

```
For sayac As Integer = 1 To 1000
If (sayac Mod 3)=0 And (sayac Mod 5)=0 And (sayac Mod 7)=0 Then
MsgBox("Aranan sayı bulundu: " & sayac)
    Exit For 'Aranan en küçük sayı bulundu. Döngüyü ani sonlandır End If
Next sayac
```

Bu uygulamanın çözümünde kullanılan algoritma 1 sayısından başlayan ve 1'er artımlarla 1000'e doğru ilerleyen bir döngü değişkeninin aradığı sayı olup olmadığını bir If deyimi ile test etmektedir. Dolayısıyla bu testi geçen ilk sayı da aranan sayıların en küçüğüdür ve kalan diğer sayıların test edilmesine gerek yoktur. Bu nedenle döngü, If gövdesinde Exit For deyimi kullanılarak ani bir şekilde sonlandırılmıştır.

Bireysel Etkinlik

•Bir önceki uygulamada bizden istenen 1 ile 1000 arasındaki tam sayılardan 3, 5 ve 7 ile tam olarak bölünebilen en büyük tam sayıyı bulmak olsaydı çözüm üzerinde hangi değişiklikleri yapmak gerekirdi?

Döngüye Devam (Continue)

Bazı uygulamalarda belirli bir koşul gerçekleştiğinde içerisinde bulunulan döngünün mevcut iterasyonunun yarıda kesilmesi (işletilen mevcut satırdan sonraki tekrar deyimlerinin atlanması) ve döngünün bir sonraki iterasyonuna geçilmesi gerekebilir. Bu amaçla For döngüleri içerisinde Continue For ve While döngülerinde de Continue Do deyimlerinden faydalanılabilir. Böyle bir durumda söz konusu koşulun sağlanıp sağlanmadığı bir If deyimi ile test edilebilir ve ilgili Continue komutu If gövdesinde çalıştırılabilir. For döngüleri içerisinde işletilen Continue komutu program akışını döngünün For satırındaki artım miktarı alanına yönlendirir ve döngü değişkeni burada güncellenerek yeni bir iterasyona başlanır. Öte yandan While döngüleri içerisinde işletilen Continue komutu program akışını Do While satırındaki koşula yönlendirmektedir.

Örnek

• 1 ile 30 arasındaki tek tam sayıları bir mesaj kutusu içerisinde sırasıyla kullanıcıya sunacak bir Visual Basic programı yazınız.

Çözüm:

```
For sayac As Integer = 1 To 30
    If (sayac Mod 2) = 0 Then
        Continue For 'Çift sayıları atla
    End If
    MsgBox(sayac)
Next sayac
```

Yukarıdaki çözüm dikkatle incelenirse çift sayıların If deyimi ile tespit edildiği ve If gövdesi içerisinde kullanılan Continue deyimi sayesinde de bu sayıların pas (es) geçildiği, yani mesaj kutusu ile kullanıcıya yansıtılmadığı anlaşılır. Continue deyimi mevcut sayacın aranan bir değer olmadığını ve bir sonraki iterasyona geçilmesi gerektiğini vurgulamaktadır.

İç İçe Döngüler

Döngü içinde döngü olarak da bilinen iç içe döngüler, bir döngü içerisinde (dış döngü) gerçekleştirilen işlemlerden birisinin bir başka döngü (iç döngü) olması durumunu karşılamak üzere kullanılan programlama yapıları şeklinde özetlenebilirler. İç içe döngülerde dış döngünün her bir iterasyonunda iç döngü yeni baştan bir daha işletilir. Bu durum dış döngü sona erinceye kadar devam eder. İç içe döngüler, iç içe For döngülerini, iç içe While döngülerini veya bu iki döngü türünün iç içe kombinasyonlarını içerebilir. İç içe döngüler genellikle aralarında iç-dış ilişkisi olan en az 2 döngü ile oluşturulurlar ama iç içe 3 (örneğin matris çarpımı uygulaması) veya daha fazla sayıda döngünün kullanıldığı uygulamalar da mevcuttur. İç içe döngülerde Exit ve Continue deyimlerinin kullanımına oldukça dikkat edilmelidir. Bu deyimlerin işlevlerinin, sadece içerisinde bulundukları döngüyü ilgilendirdiği, diğer döngüleri ilgilendirmediği asla unutulmamalıdır.

İç içe döngülerde döngü kontrol değişkenlerine i, j ve k gibi isimlerin verildiği bir notasyondan sıklıkla faydalanılır.

Örnek

Kullanıcıdan üç metin kutusu aracılığıyla alacağı üç pozitif tam sayının çarpım sonucunu çarpma operatörü kullanmadan (sadece toplama yaparak) bulacak ve bir mesaj kutusu içerisinde kullanıcıya sunacak bir Visual Basic uygulaması yazınız. Sayıların pozitif tam sayılar olarak girileceğinin garanti edildiği kabulünü yapabilirsiniz.

Çözüm:

```
Dim toplam As Integer = 0
Dim sayi1 As Integer = Integer.Parse(TextBox1.Text)
Dim sayi2 As Integer = Integer.Parse(TextBox2.Text)
Dim sayi3 As Integer = Integer.Parse(TextBox3.Text)
For i As Integer = 1 To sayi1       'Dış döngü
    For j As Integer = 1 To sayi2 'İç döngü
        toplam = toplam + sayi3
    Next j
Next i
MsgBox("Hesaplanan Çarpım= " & toplam)
```

sayi1*sayi2*sayi3 çarpımı çarpmanın birleşme özelliği yüzünden sayi1*(sayi2*sayi3) şeklinde de ifade edilebilir. sayi2*sayi3 çarpımı ise sayi2 kere sayi3'ün kendisiyle toplanmasından ibarettir. Dolayısıyla sayi2 kere tekrar eden içteki For döngüsünün gövdesinde başlangıç değeri 0 olan bir toplama sürekli olarak sayi3 eklenmektedir. İçteki döngü sayi1 kere tekrar eden bir dış döngünün içerisine yerleştirildiğinde ise çözüme ulaşılmaktadır. Zira sayi1*(sayi2*sayi3) çarpımı sayi1 kere (sayi2*sayi3)'ün kendisiyle toplanmasıyla elde edilebilmektedir.

Döngü değişkenleri bağımlı iç içe döngüler

Aşağıda, döngü değişkenleri bağımlı (aralarında bir ilişki bulunan) iç içe döngüleri açıklayabilmek amacıyla kullanılan basit bir Visual Basic uygulaması ve bu uygulama çalıştırıldığında mesaj kutusu üzerinden kullanıcıya yansıyan çıktılar alt alta gösterilmiştir.

```
For i As Integer = 1 To 4
    For j As Integer = (i + 1) To 6 MsgBox(i
        & " , " & j)
    Next j
Next i
```

```
1,2   1,3   1,4   1,5   1,6
2,3   2,4   2,5   2,6
3,4   3,5   3,6
4,5   4,6
```

Bu uygulama iç içe 2 For döngüsü barındırmaktadır: i adındaki döngü değişkeni ile kontrol edilen dış döngü ve j adındaki döngü değişkeni ile kontrol edilen iç döngü. i döngü değişkeni 1 değerinden başlamakta, her bir iterasyon sonrasında değerini 1 artırmakta ve 4 değerinde sona ermektedir. Yani dış döngü toplamda 4 kere tekrarlanmaktadır. Öte yandan aynı mantığı iç döngü için bir çırpıda kurabilmek mümkün değildir çünkü j döngü değişkeninin başlangıç değeri her bir iterasyonda mevcut i döngü değişkeninin 1 fazlasına eşit olmaktadır. Dolayısıyla j değişkeninin i değişkenine bir bağımlılığı söz konusudur. Örneğin i=4 iken, yani dış döngü 4. ve son tekrarını yaparken, içteki For döngüsü arka planda

```
For j As Integer = 5 To 6
```

formunu almaktadır ve toplamda 2 tekrar gerçekleştirecek bir döngüye dönüşmektedir. (Bu bilgiler ışığında program çalıştırıldığında mesaj kutusunun ekranda kaç kere görüneceğini hesaplayınız.)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
```

Bireysel Etkinlik

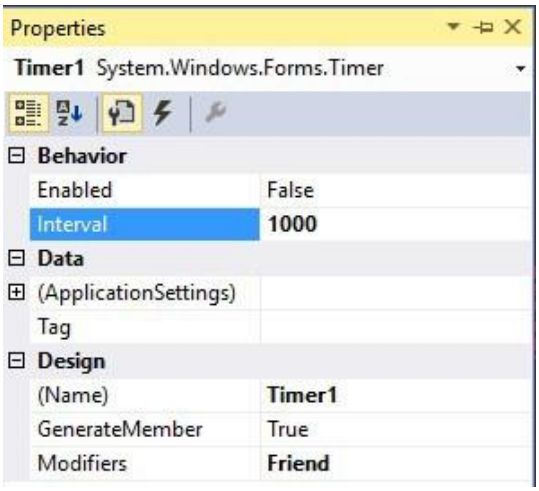
- Çalıştırıldığında yandaki ekran çıktısını Output penceresinde üretecek (Debug.Writeline kullanabilirsiniz) bir Visual Basic programı yazınız. Satırları oluşturacak bir dış döngüye; sütunları oluşturacak ve dış döngüye bağımlı bir iç döngüye ihtiyacınız olacağına dikkat ediniz.

ZAMANLAYICILAR

Bazı interaktif uygulamalarda (örneğin bir oyun uygulamasında veya çoktan seçmeli sorulardan oluşan bir test uygulamasında) kullanıcının tepkisinin belirli bir zaman dilimi içerisinde gerçekleşmesi beklenir. Visual Basic programlama dilinde bu amaçla Timer isimli kontrol nesnesinden faydalanılabilir.

Zamanlayıcı; örneğin bir buton nesnesi gibi, çalışma zamanında görünen ve müdahale edilebilen interaktif bir kontrol nesnesi değildir.

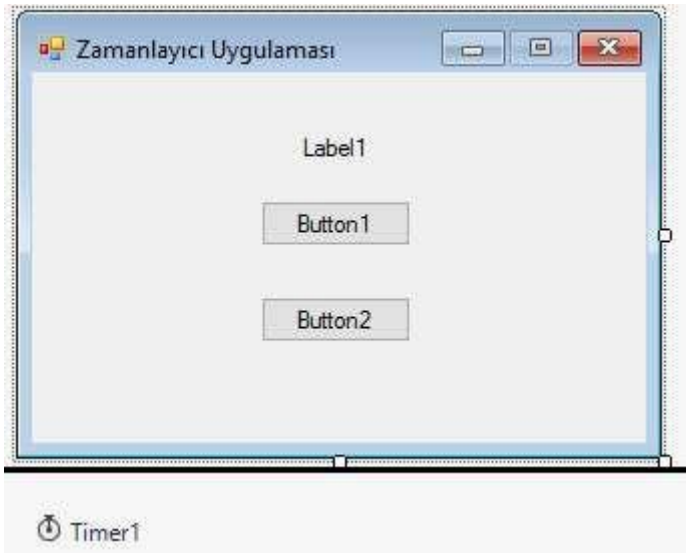
Bir Timer nesnesinin Properties penceresinden ulaşılabilen ve kontrol edilebilen özellikleri yandaki gibidir. Enabled özelliği False değerine sahipken zamanlayıcı devre dışıdır. Benzer şekilde Enabled özelliğinin değeri True iken zamanlayıcı devrededir. Öte yandan Interval özelliğine atanan tam sayı değer zaman ölçümü için kullanılan sürenin mili saniye cinsinden karşılığıdır.



Bir zamanlayıcının kontrolü genellikle 3 aşamalı olarak gerçekleştirilir:
(1) Zamanlayıcının tetiklenme aralığı belirlenir.
(2) Zamanlayıcı aktif hâle getirilir.
(3) Zamanlayıcı pasif hâle getirilir.

Örneğin, Interval değeri 1000 (1000 mili saniyelik ve 1 saniyelik bir zaman ölçümü yapılıyorken) ve zamanlayıcı devrede iken her 1 saniye sonrasında zamanlayıcının _Tick isimli fonksiyonu tetiklenecektir ve istenen işlev bu fonksiyon gövdesinde gerçekleştirilebilecektir.

Bu nesnenin işlevini anlayabilmek için aşağıdaki yerleşim planına sahip bir form uygulaması geliştirelim.



Uygulama ise şöyle çalışsın. Button1 isimli butona tıklandığında Timer1 isimli zamanlayıcı devreye alınsın. Button2 isimli butona tıklandığında ise Timer1 isimli zamanlayıcı devre dışı kalsın. Zamanlayıcı devrede iken Interval değerine ulaşıldığında tetiklenecek fonksiyon (Timer1_Tick) içerisinde de başlangıç değeri sıfır olan bir sayacın değeri her seferinde 1 artırılsın ve Label1 isimli etiketin text

özelliğinin değeri olarak form uygulaması üzerinde gösterilsin. Bütün bu anlatılanları gerçekleyen Visual Basic kodu aşağıdaki gibidir.

```
Public Class Form1
    Dim sayac As Integer = 0
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
    MyBase.Load
    End Sub
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
    Timer1.Tick
        sayac = sayac + 1 Label1.Text
        = Str(sayac)
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
    Button1.Click
        Timer1.Enabled = True
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
    Button2.Click
        Timer1.Enabled = False
    End Sub
End Class
```

Bireysel Etkinlik

- Bir önceki uygulamada Timer1_Tick isimli fonksiyonun içerisinde sadece Label1.Text = TimeOfDay deyimini çalıştırınız ve çıktıyı gözlemleyiniz.

Özet

- Bu ünite kapsamında ilk olarak tekrara izin veren programlama yapıları diye bilinen döngüler ele alınmıştır ve Visual Basic programlama dilinde bu amaçla kullanılabilecek iki farklı tür döngü yapısı incelenmiştir. Bu kapsamda:
  - Döngü oluşturma kurallarından,
  - Tekrar sayısı peşinen bilinen uygulamalarda kullanılabilecek iki farklı For döngüsünden,
  - Tekrar sayısının bir koşula bağlı olarak dinamik bir şekilde tespit edilmesi gereken uygulamalarda kullanılabilecek While döngüsünden,
  - Bir döngünün ani olarak nasıl sonlandırabileceğinden,
  - Bir döngünün yeni bir iterasyonuna henüz işletilmemiş döngü adımları tamamlanmadan nasıl geçilebileceğinden,
  - Döngü içerisinde döngü olarak da bilinen iç içe döngü yapılarından bahsedilmiştir.
- Bu ünite kapsamında ayrıca zamanlayıcılar mercek altına alınmıştır ve Visual Basic araç kutusunda bulunan Timer isimli kontrol nesnesi incelenmiştir. Bir uygulamanın belirli zaman aralıklarında yerine getirmesi gereken işlemler söz konusu olduğunda bu kontrol nesnesinin programlamayı nasıl kolay bir hale getirdiği görülmüştür.

DEĞERLENDİRME SORULARI

1. Her 13 saniyede bir tetiklenmesi istenen bir zamanlayıcının Interval özelliğine atanması gereken tam sayı değer aşağıdakilerden hangisidir?
   a) 1.3
   b) 13
   c) 130
   d) 1300
   e) 13000

2. Bir zamanlayıcının devreye alınması için kullanılacak özellik aşağıdakilerden hangisidir?
   a) Interval
   b) Tag
   c) Name
   d) Enabled
   e) Modifiers

```
For sayac As Integer=37 To 8 Step -4 MsgBox("Merhaba")
Next sayac
```

3. Yukarıdaki For döngüsü kaç kere işletilir?
   a) 4
   b) 7
   c) 8
   d) 25
   e) 29

```
Dim sayac As Integer=16 Do
While (sayac>=0)
    MsgBox("Merhaba")
Loop
```

4. Yukarıdaki While döngüsü kaç kere işletilir?
   a) Sonsuz
   b) 0
   c) 1
   d) 16
   e) -16

```
For i As Integer=1 To 6
    For j As Integer=1 To 9 If
        (j Mod 2)=1 Then
            Continue  For
        End If
        MsgBox("Merhaba")
    Next j
Next i
```

5. Yukarıdaki kod parçası çalıştırıldığında mesaj kutusu ekranda kaç kez görünür?

a) 2
b) 24
c) 42
d) 30
e) 54

```
For i As Integer=1 To 6
    For j As Integer=1 To 9 If
        (j Mod 2)=1 Then
            Exit For
        End If
        MsgBox("Merhaba")
    Next j
Next i
```

6. Yukarıdaki kod parçası çalıştırıldığında mesaj kutusu ekranda kaç kez görünür?

a) 0
b) 6
c) 9
d) 15
e) 54

```
For i As Integer=1 To 6
    For j As Integer=9 To (i+1) Step -2 MsgBox("Merhaba")
    Next j
Next i
```

7. Yukarıdaki kod parçası çalıştırıldığında mesaj kutusu ekranda kaç kez görünür?

a) 4
b) 12
c) 14
d) 12
e) 18

```
Dim sayac As Integer=0
Do While (True)
    MsgBox("Merhaba")
    sayac=sayac+1
Loop
```

8. Yukarıdaki While döngüsü kaç kere işletilir?
   a) 0
   b) Sonsuz
   c) 32
   d) 64
   e) 128

9. Bir For döngüsü ile ilgili olarak aşağıdakilerden hangisi yanlıştır?
   a) Artım miktarı belirtilmezse 1 olduğu kabul edilir.
   b) Döngüyü ani sonlandırmak için Break For komutu kullanılır.
   c) Artım miktarı negatif olamaz.
   d) Döngü değişkeninin bir başlangıç değeri vardır.
   e) Döngü değişkeninin bir bitiş değeri vardır.

```
For i As Integer=1 To 6
    Dim j As Integer=3
    Do While (j<=10)
        MsgBox("Merhaba")
        j=j+9
    Loop
Next i
```

10. Yukarıdaki kod parçası çalıştırıldığında mesaj kutusu ekranda kaç kez görünür?
    a) 0
    b) 1
    c) 3
    d) 6
    e) 10

Cevap Anahtarı

1.e, 2.d, 3.c, 4.a, 5.b, 6.a, 7.e, 8.b, 9.c

YARARLANILAN KAYNAKLAR

Salvage, J. (2002). The Visual Basic Coach, Addison Wesley.

Halvorson, M. (2013). Microsoft Visual Basic 2013 Step by Step Developer, Microsoft Press

Newsome, B. (2015). Beginning Visual Basic 2015, Wrox.

Dal, D. (2016). MATLAB ile Programlama, 5. Baskı, Bursa Ekin Yayınevi.

DİYALOG PENCERELERİ

İÇİNDEKİLER

- Renk Seçimi Penceresi
- Dizin Tarayıcısı Penceresi
- Font Seçimi Penceresi
- Dosya Açma Penceresi
- Dosya Kaydetme Penceresi

HEDEFLER

- Bu üniteyi çalıştıktan sonra aşağıdaki diyalog penceresi türlerini kavrayabilecek ve uygulamalarınızda kullanabileceksiniz:
  - Renk Seçimi Penceresi (ColorDialog)
  - Dizin Tarayıcısı Penceresi (FolderBrowserDialog)
  - Font Seçimi Penceresi (FontDialog)
  - Dosya Açma Penceresi (OpenFileDialog)
  - Dosya Kaydetme Penceresi (SaveFileDialog)

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Yrd. Doç. Dr.
Levent BAYINDIR

ÜNİTE
10

GİRİŞ

Konsol tabanlı uygulamalarla kıyaslandıklarında, görsel uygulamaların temel avantajı kullanıcıya *bilgi vermek* ve kullanıcıdan *bilgi almak* konularında çok daha esnek olmalarıdır.

Görsel uygulamaların temel avantajı bilgi vermek ve almak konusunda daha esnek olmalarıdır.

Görsel uygulamalar *bilgi vermek* için birden fazla pencere ve bu pencereler üzerinde her tür görsel öğeyi (şekil, tablo gibi) kullanabilmeye ek olarak, kullanıcının bu öğeler ile etkileşime girerek öğelerin durumunu değiştirebilmesine olanak sağlar. Buna karşılık, pek çok konsol uygulaması sadece renklendirilebilen metin çıktısı verebilmektedir.

Benzer şekilde, pek çok konsol tabanlı uygulama kullanıcıdan *bilgi almak* için sadece klavyeyi kullanabilirken (metin girişi veya kısayollar ile), görsel uygulamalarda klavyeye ek olarak fare ve dokunmatik ekranların kullanımı mümkündür. Ayrıca kullanıcı, seçimlerini klavyeyi kullanmaksızın pencerelerde gösterilen nesnelerle etkileşime girerek de (listeden bir eleman seçmek gibi) belirleyebilir.

Diyalog pencereleri kullanıcıdan bilgi almak için kullanılan standart pencerelerdir.

Kullanıcıdan bilgi almak için, geliştiricinin kendi tasarımı olan tamamen özelleşmiş pencerelerin kullanımı mümkünken, bazı sık kullanılan işlemleri yapabilmek için ön tanımlı pencereler de kullanılabilmektedir. Bu pencerelere *diyalog pencereleri* denilmektedir.

Diyalog pencereleri genellikle ana pencerenin önüne açılan ve giriş işlemi tamamlanmadan ana pencereye dönülmesine izin vermeyen modal pencerelerdir.

Diyalog pencereleri, özelleşmiş pencerelerle karşılaştırıldıklarında aşağıdaki avantajlara sahiptirler:

- Kullanıma hazır bileşenlerdir. Tasarım ve gerçekleştirim gerektirmezler.
- İlişkili metotları aracılığıyla geliştiriciler tarafından kolayca kullanılabilirler.
- Standart bileşenler olduklarından kullanıcılar tarafından bilinirler ve kolayca kullanılabilirler.

Daha önceki bölümlerde bu pencerelerden Giriş (InputBox) ve Mesaj (MsgBox) diyalog pencereleri tanıtılmıştı. Bu bölümde ise, kullanıcıdan bilgi almak için .NET platformu tarafından uygulama geliştiricilerine sunulmuş, aşağıdaki diyalog pencereleri anlatılacaktır:

- Renk Seçimi Penceresi (ColorDialog)
- Dizin Tarayıcısı Penceresi (FolderBrowserDialog)
- Font Seçimi Penceresi (FontDialog)
- Dosya Açma Penceresi (OpenFileDialog)
- Dosya Kaydetme Penceresi (SaveFileDialog)

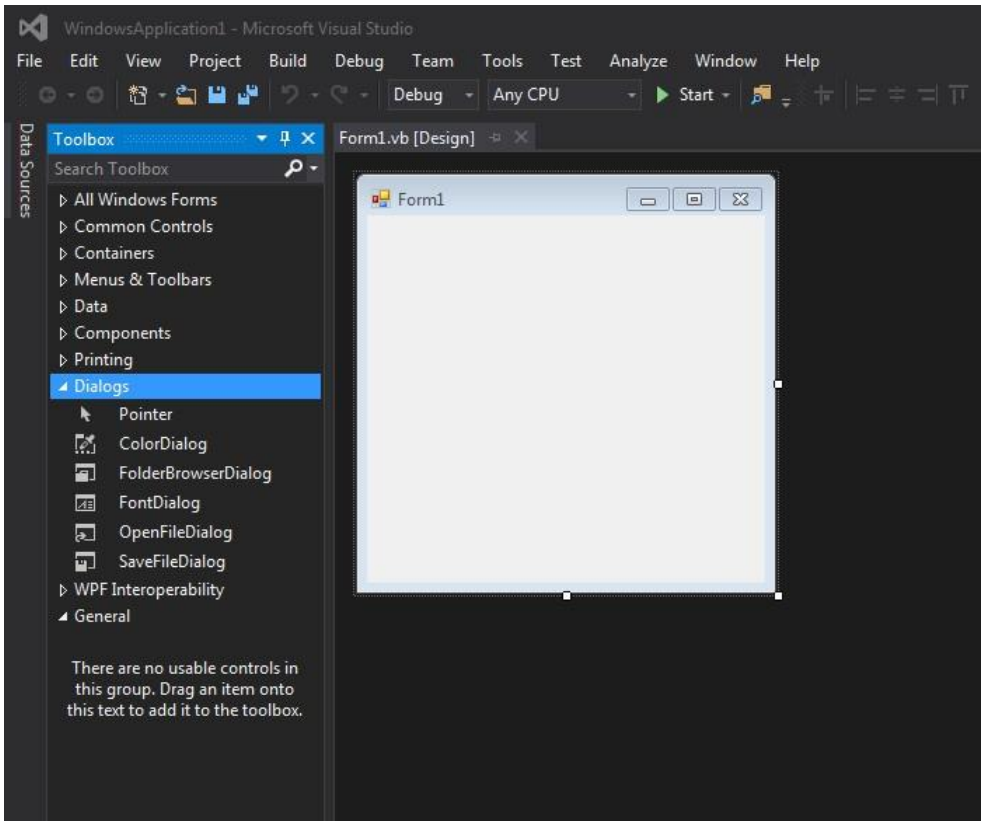
.NET platformundaki bu diyalog pencereleri System.Windows.Forms isim uzayındaki CommonDialog sınıfından türetilmişlerdir ve bu nedenle Tablo 10.1'de listelenen ortak metotlara sahiptirler.

Tablo 10.1. Ünite 10'da anlatılan diyalog pencerelerinin sahip olduğu bazı ortak metotlar ve açıklamaları

Metot	Açıklama
Reset	Tüm pencere seçeneklerini varsayılan değerlerine döndürür. Örneğin son seçilen renk siyah yapılır.
ShowDialog	Renk seçim penceresini kullanıcıya gösterir.

Bu metotlardan ShowDialog metodu diyalog pencerelerini kullanıcıya göstermek için tüm örneklerimizde kullanacağımız önemli bir metottur.

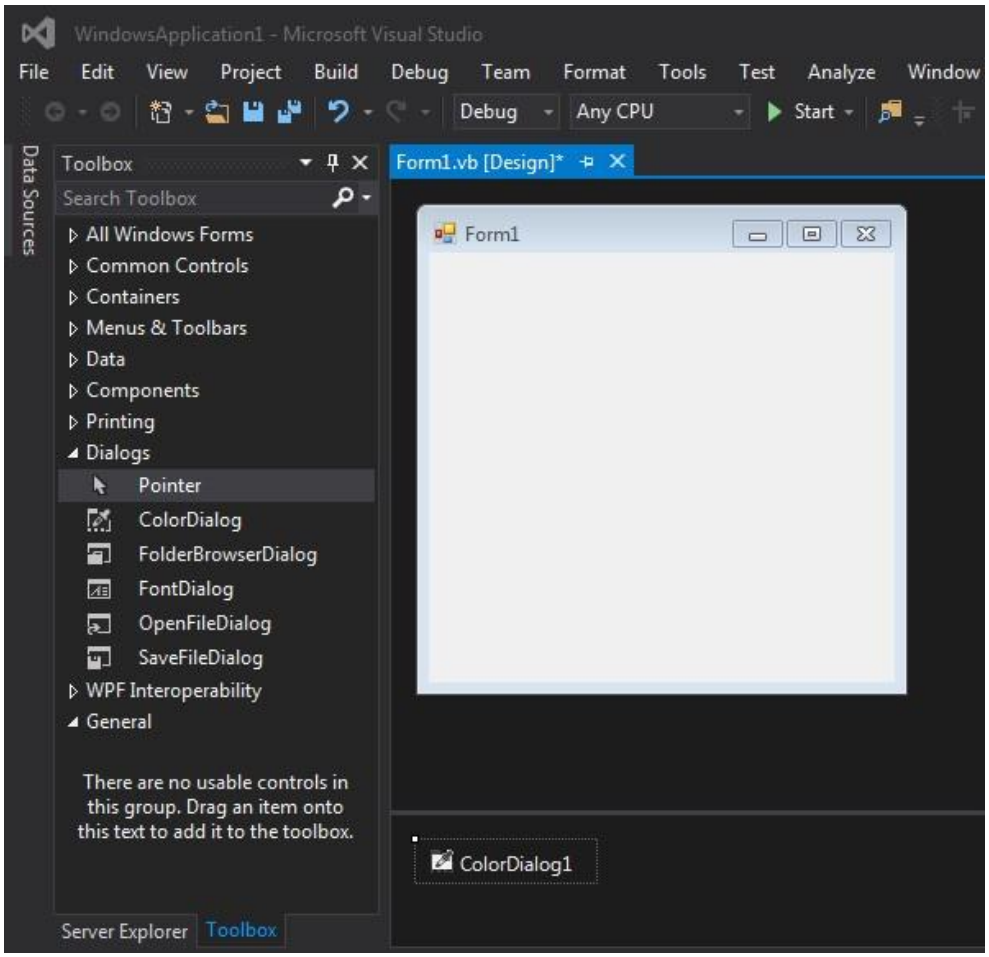
Herhangi bir diyalog penceresini projenizde kullanabilmeniz için öncelikle bu diyalog penceresini projenize eklemelisiniz. Bunun için .NET platformunun Toolbox menüsünde Dialogs alt başlığında listelenen bileşenler kullanılır (Şekil 10.1).



Şekil 10.1. Diyalog pencereleri Toolbox menüsünde Dialogs alt başlığında bulunmaktadır.

Burada bulunan diyalog pencerelerinden kullanılmak istenen üzerine tıklanarak seçilmeli ve fare tuşu bırakılmadan sürüklenerek projenizdeki ana formun üzerine bırakılmalıdır. Bu işlem sonunda eğer formun altında ilgili diyalog penceresinin ismi listelenirse artık bu formu projenizde kullanabilirsiniz.

ColorDialog penceresinin form üzerine sürüklenmesi sonucunda oluşan görüntü Şekil 10.2'de gösterilmektedir. Görüldüğü üzere diyalog penceresinin ismine "1" eklenmiştir. Buradaki "ColorDialog1", ColorDialog sınıfından oluşturulmuş nesnenin referansıdır ve projenizdeki kodlarda ColorDialog türündeki bu nesneye bu ismi kullanarak erişebilirsiniz.



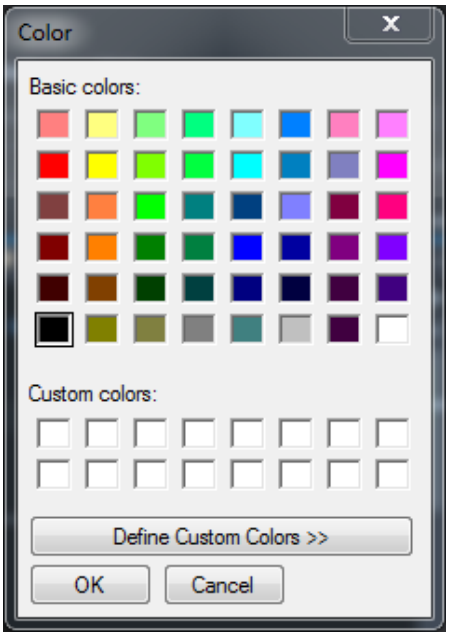
Şekil 10.2. Diyalog pencereleri formlar üzerine sürüklenip bırakılarak projeye dâhil edilebilirler.

Bireysel Etkinlik

- ColorDialog bileşenini formunuzun üzerine bir kere daha sürükleyip bırakınız. Aynı türde ikinci bir nesne oluşturulduğunu ve adının "ColorDialog2" olduğunu göreceksiniz.
- Bu işlemi her tekrar edişinizde, .NET üretilen nesneleri benzer şekilde isimlendirmeye devam edecektir.
- İlgili nesnelerin isimlerini isterseniz "Özellikler" penceresindeki "(Name)" alanını kullanarak değiştirebilirsiniz.

RENK SEÇİMİ PENCERESİ

Kullanıcının renk seçimi yapmasına olanak sağlayan diyalog penceresidir. Bu pencerenin ekran görüntüsü Şekil 10.3'te gösterilmiştir.



Şekil 10.3. Renk seçimi penceresi ekran görüntüsü.

Ekran görüntüsünden anlaşılabileceği gibi, pencere 48 adet ön tanımlı renkten birini seçmenizi veya özelleşmiş renkler tanımlayıp ("Define Custom Colors" butonu ile) onlardan birini seçebilmenizi sağlamaktadır.

Renk seçimi penceresi oluşturmak için kullanılan ColorDialog sınıfına ait özelliklerden bazıları açıklamaları ile birlikte Tablo 10.2'de gösterilmektedir.

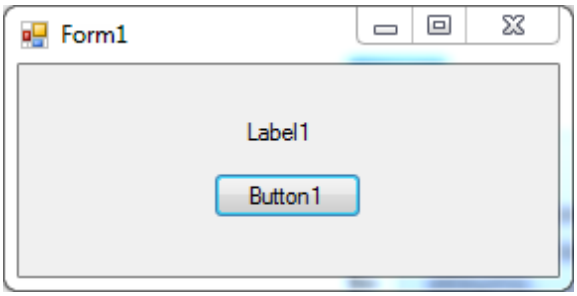
Tablo 10.2. ColorDialog sınıfına ait bazı özellikler ve açıklamaları

Özellik	Açıklama
AllowFullOpen	Seçim penceresinde özelleşmiş renk tanımlanıp tanımlanamayacağını belirten özelliktir.
Color	Kullanıcının seçtiği rengi temsil eder. Bu değer alınabilir ya da değiştirilebilir.
CustomColors	Kullanıcıya gösterilen özelleşmiş renk kümesini temsil eder. Bu küme alınabilir veya değiştirilebilir.
ShowHelp	Seçim penceresinde "Yardım" (Help) butonunun gösterilip gösterilmeyeceğini belirleyen özelliktir.

Örnek

- Bir form üzerine bir etiket ve buton yerleştiriniz.
- Projenize daha önce gösterildiği gibi bir ColorDialog bileşeni ekleyiniz.
- Butona tıklandığında kullanıcıya renk seçim penceresinin gösterilmesini sağlayınız.
- Eğer kullanıcı bir renk seçtiyse, formdaki etiketin ön plan rengini (yazı rengini) seçilen renk yapınız.

Örnekte tasarlanması istenen formun ekran görüntüsü Şekil 10.4'te gösterilmektedir.



Şekil 10.4. Birer etiket ve buton bileşeni içeren form.

Butona tıklandığında istenen işlemleri gerçekleştiren kod parçası aşağıda verilmiştir. Uygulamanın doğru çalışması için (eğer varsayılan değerleri değiştirmediyseniz) formdaki butona çift tıklayarak karşınıza gelen kod parçasını aşağıdaki kod parçası ile değiştirmeniz yeterlidir.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
    Button1.Click
        Dim sonuc = ColorDialog1.ShowDialog()
        If sonuc = DialogResult.OK Then
            Label1.ForeColor = ColorDialog1.Color
        End If
    End Sub
End Class
```

Butona tıklandığında ilk olarak ShowDialog metodu kullanılarak renk seçim penceresi kullanıcıya gösterilmektedir. Kullanıcı bu pencereyi kapatana kadar programın çalışması duraklatılacaktır. Kullanıcı bu pencereyi üç farklı şekilde kapatabilir:

a) Bir renk seçtikten sonra "OK" butonuna tıklayarak
b) "Cancel" butonuna tıklayıp renk seçim işlemini iptal ederek
c) Renk seçim penceresinin sağ üst köşesindeki X butonuna tıklayıp işlemi iptal ederek

Uygulamamızda sadece ilk seçenek gerçekleştiğinde renk değişimi yapabileceğimizden kullanıcının tercihini elde edip ilk tercihi mi yaptığını belirlemeliyiz.

Bu tercih ShowDialog metodunun geri dönüş değerini kullanarak kontrol edilebilir. Eğer kullanıcı birinci tercihi yaptıysa bu metottan "DialogResult.OK" sonucu dönecektir. Bunu örnekte olduğu gibi bir "If" yapısı içerisinde kullanarak kullanıcının birinci tercihi yapması durumunda etiketin ön plan rengini seçilen renk yapabiliriz.

Uygulama çalıştırıldığında, kırmızı renk seçilmiş ise ekran görüntüsü Şekil 10.5'te gösterildiği gibi olacaktır.

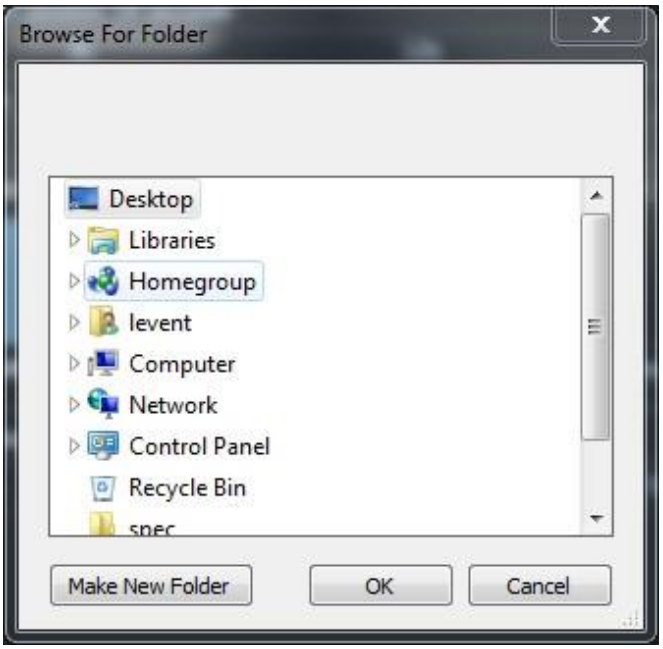
Şekil 10.5. Örnek uygulama çalıştırıldığında ve renk seçim penceresinde kırmızı rengin seçilmesi sonucunda etiket ön plan renginin kırmızıya dönüşmüş hali.

Bireysel Etkinlik

- ColorDialog sınıfının Tablo 10.2'de listelenen özelliklerini kullanan uygulamalar geliştiriniz.

DİZİN TARAYICISI PENCERESİ

Kullanıcının sistem dizinleri arasında gezinerek bir dizin seçmesini sağlayan diyalog penceresidir. Bu pencerenin ekran görüntüsü Şekil 10.6'da gösterilmiştir.



Şekil 10.6. Dizin tarayıcısı penceresi ekran görüntüsü.

Ekran görüntüsünden anlaşılabileceği gibi, pencere mevcut dizin hiyerarşisini bir ağaç halinde göstermekte ve kullanıcının bu ağacın dallarını fare ile açıp kapatabilmesine ve seçebilmesine olanak sağlamaktadır. Buna ek olarak "Make New Folder" butonu aracılığıyla seçili olan dalın altında yeni bir dizin oluşturulmasına olanak sağlamaktadır.

Dizin tarayıcısı penceresi oluşturmak için kullanılan FolderBrowserDialog sınıfına ait özelliklerden bazıları açıklamaları ile birlikte Tablo 10.3'te gösterilmektedir.

Tablo 10.3. FolderBrowserDialog sınıfına ait bazı özellikler ve açıklamaları

Özellik	Açıklama
Description	Diyalog penceresinde gösterilen dizin ağacının üstündeki açıklayıcı bilgiyi tutan özelliktir.
RootFolder	Dizin taramasının başlayacağı kök dizini (root folder) tutan özelliktir.
SelectedPath	Kullanıcı tarafından seçilen patikayı (path) tutan özelliktir.
ShowNewFolderButton	Kullanıcının yeni dizin oluşturmasını sağlayan "Make New Folder" butonunun diyalog penceresinde gösterilip gösterilmeyeceğini belirleyen özelliktir.

Örnek

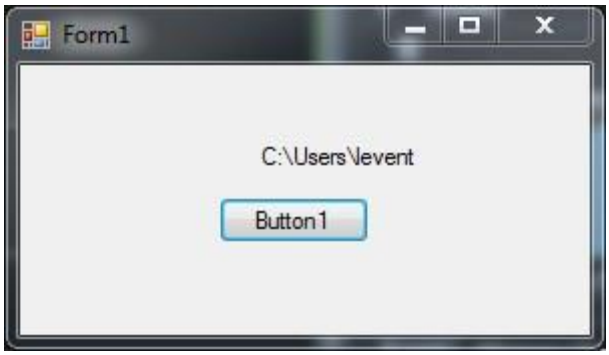
- Şekil 10.4'te gösterilen formu yeniden tasarlayınız.
- Ancak bu sefer, butona tıklandığında kullanıcıya dizin tarayıcısı penceresinin gösterilmesini sağlayınız.
- Eğer kullanıcı bir dizin seçtiyse formdaki etiketin yazısını seçilen dizinin tam yolu yapınız.

Butona tıklandığında istenen işlemleri gerçekleştiren kod parçası aşağıda verilmiştir.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
    Button1.Click
        Dim sonuc = FolderBrowserDialog1.ShowDialog()
        If (sonuc = DialogResult.OK) Then
            Label1.Text = FolderBrowserDialog1.SelectedPath
        End If
    End Sub
End Class
```

Bir önceki örneğe benzer şekilde "ShowDialog" metodu kullanılarak dizin tarayıcısı penceresi kullanıcıya gösterilmiş, kullanıcı eğer bir dizin seçmiş ise bu dizinin tam yolu "SelectedPath" özelliği kullanılarak alınıp etiketin yazısı haline getirilmiştir.

Uygulama çalıştırıldıktan sonra, "levent" adlı kullanıcının ev dizininin seçilmesinin ardından elde edilen uygulama ekran görüntüsü Şekil 10.7'de gösterilmiştir.



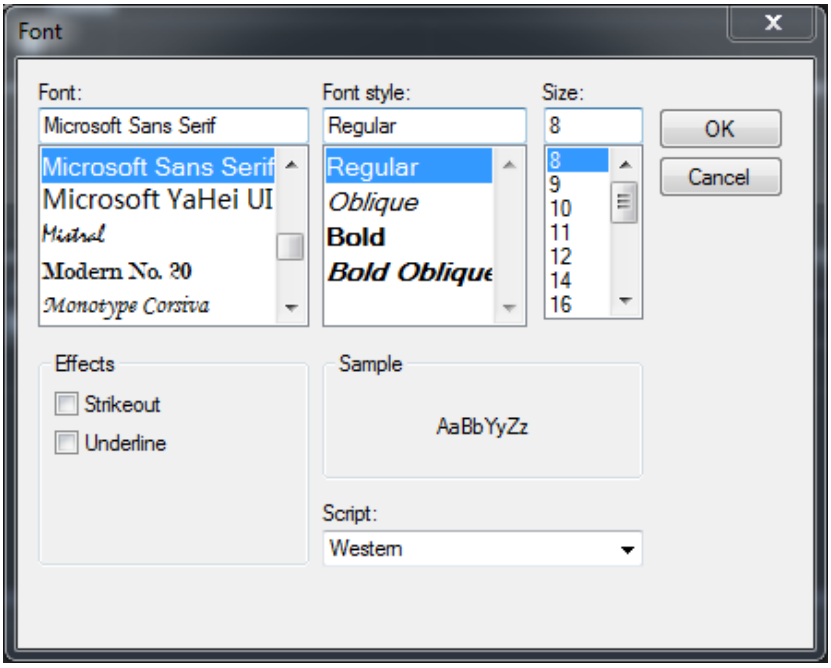
Şekil 10.7. Örnek uygulamada "levent" adlı kullanıcının ev dizininin seçilmesinin ardından elde edilen uygulama ekran görüntüsü

Bireysel Etkinlik

- FolderBrowserDialog sınıfının Tablo 10.3'te listelenen özelliklerini kullanan uygulamalar geliştiriniz.

FONT SEÇİMİ PENCERESİ

Sistemde kurulu olan yazı tipleri ve özelliklerini listeler halinde kullanıcıya sunan ve kullanıcının bu listeler aracılığıyla belirli bir yazı tipini ve bu yazı tipinin özelliklerini belirlemesini sağlayan diyalog penceresidir. Bu pencerenin ekran görüntüsü Şekil 10.8'de gösterilmiştir.



Şekil 10.8. Font seçimi penceresi ekran görüntüsü.

Ekran görüntüsünden anlaşılabileceği gibi, pencere sistemde kurulu olan yazı tiplerini listelemekte ve bu yazı tiplerinin stil, boyut ve efekt gibi özelliklerinin değiştirilebilmesini sağlamaktadır.

Font seçimi penceresi oluşturmak için kullanılan FontDialog sınıfına ait özelliklerden bazıları açıklamaları ile birlikte Tablo 10.4'te gösterilmektedir.

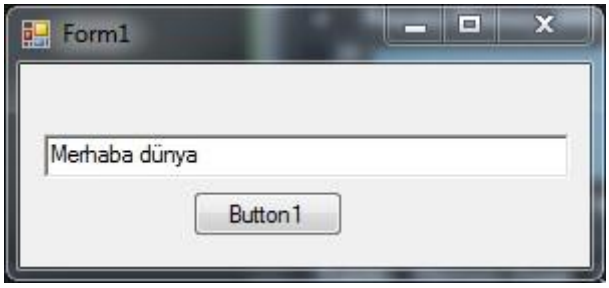
Tablo 10.4. FontDialog sınıfına ait bazı özellikler ve açıklamaları

Özellik	Açıklama
AllowVectorFonts	Vektör fontların seçilmesine izin verilip verilmeyeceğini belirten özelliktir.
AllowVerticalFonts	Yatay yazı tiplerine ek olarak dikey yazı tiplerinin seçilmesine izin verilip verilmeyeceğini belirten özelliktir.
Color	Seçilen yazı tipinin rengini tutan özelliktir.
Font	Seçilen yazı tipini tutan özelliktir.
MaxSize	Kullanıcının seçebileceği en büyük yazı tipi boyutunu belirten özelliktir.
MinSize	Kullanıcının seçebileceği en küçük yazı tipi boyutunu belirten özelliktir.
ShowColor	Diyalog penceresinin seçilen rengi gösterip göstermeyeceğini belirten özelliktir.

Örnek

- Bir form üzerine bir RichTextBox bileşeni ve buton yerleştiriniz.
- Uygulama çalıştırıldığında RichTextBox bileşenine bir yazı giriniz.
- Ardından butona tıklandığında kullanıcıya font seçimi penceresinin gösterilmesini sağlayınız.
- Eğer kullanıcı bir font belirlediyse, RichTextBox bileşenindeki yazının yazı tipini ve rengini seçilen font özelliklerini kullanacak şekilde güncelleyiniz.

Örnek uygulama çalıştırıldıktan ve RichTextBox bileşenine yazı girildikten sonraki ekran görüntüsü Şekil 10.9'da gösterilmiştir.



Şekil 10.9. Bir buton ve RichTextBox bileşeni içeren örnek uygulamanın ekran görüntüsü.

Butona tıklandığında istenen işlemleri gerçekleştiren kod parçası aşağıda verilmiştir.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
    Button1.Click
        FontDialog1.ShowColor = True
        Dim sonuc = FontDialog1.ShowDialog()
        If (sonuc = DialogResult.OK) Then
            RichTextBox1.ForeColor = FontDialog1.Color
            RichTextBox1.Font = FontDialog1.Font
        End If
    End Sub
End Class
```

Bir önceki örneğe benzer şekilde "ShowDialog" metodu kullanılarak font seçimi penceresi kullanıcıya gösterilmiş, kullanıcı eğer bir font seçmiş ise bu fontun rengi ve özellikleri sırasıyla "Color" ve "Font" özellikleri kullanılarak alınıp, RichTextBox bileşeninin ilişkili özelliklerine atanmıştır.

Yukarıdakilere ek olarak font seçim penceresinde kullanıcının renk seçimi de yapabilmesi için "FontDialog1" nesnesinin "ShowColor" özelliği "True" yapılmıştır (varsayılan değer False).

Uygulama çalıştırılıp bir font ve renk seçildikten sonra elde edilen ekran görüntüsü Şekil 10.10'da gösterilmiştir.

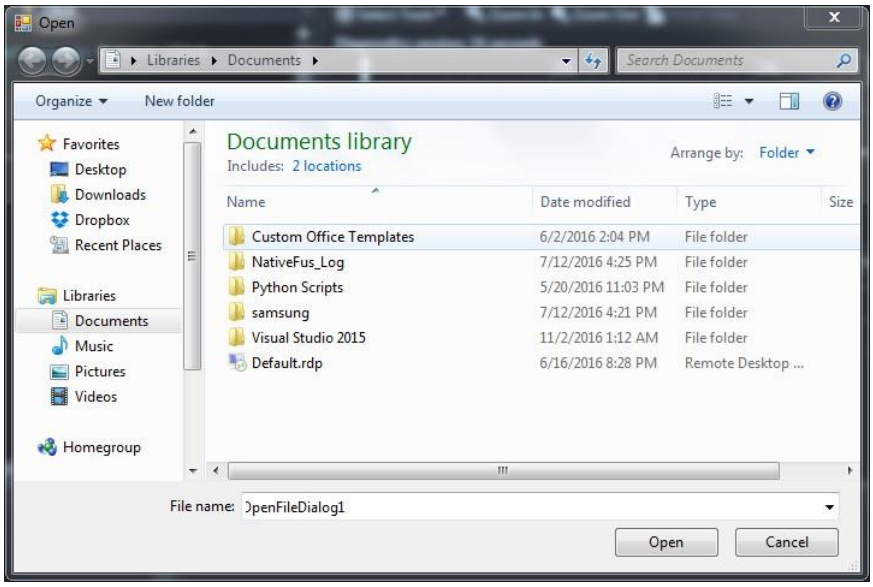
Şekil 10.10. Bir buton ve RichTextBox bileşeni içeren örnek uygulamanın ekran görüntüsü.

Bireysel Etkinlik

- FontDialog sınıfının Tablo 10.4'te listelenen özelliklerini kullanan uygulamalar geliştiriniz.

DOSYA AÇMA PENCERESİ

Kullanıcının açmak amacıyla bir dosya seçmesine olanak sağlayan penceredir. Bu pencerenin ekran görüntüsü Şekil 10.11'de gösterilmiştir.



Şekil 10.11. Dosya açma penceresi ekran görüntüsü.

Ekran görüntüsünden anlaşılabileceği gibi, pencerede mevcut dizin hiyerarşisinde gezinmeye ve herhangi bir dosyayı seçmeye olanak sağlayan bileşenler bulunmaktadır. Ayrıca penceredeki "New folder" butonu ile yeni dizin oluşturulabilmekte, var olmayan bir dosyanın adı verilebilmekte ve penceredeki bileşenlerin gösterimleri değiştirilebilmektedir.

Dosya açma penceresi oluşturmak için kullanılan OpenFileDialog sınıfına ait özelliklerden bazıları açıklamaları ile birlikte Tablo 10.5'te gösterilmektedir.

Tablo 10.5. OpenFileDialog sınıfına ait bazı özellikler ve açıklamaları

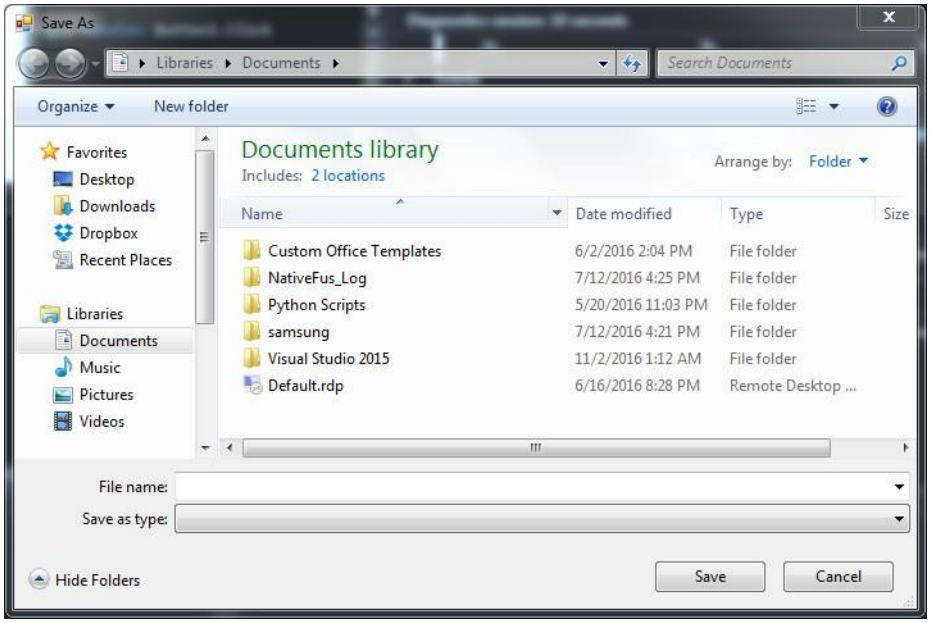
Özellik	Açıklama
AddExtension	Kullanıcı dosya uzantısını belirtmemişse dosya uzantısının otomatik eklenip eklenmeyeceğini belirten özelliktir.
CheckFileExists	Kullanıcı var olmayan bir dosyayı belirtmişse diyalog penceresinin kullanıcıyı uyarıp uyarmayacağını belirten özelliktir.
CheckPathExists	Kullanıcı var olmayan bir dosya yolu belirtmişse diyalog penceresinin kullanıcıyı uyarıp uyarmayacağını belirten özelliktir.
DefaultExt	Varsayılan dosya uzantısını belirten özelliktir.
FileName	Diyalog penceresinde seçilen dosyayı belirten özelliktir.
FileNames	Diyalog penceresinde seçilen dosyaları belirten özelliktir.
InitialDirectory	Diyalog penceresi ilk açıldığında içindeki dosyaların gösterildiği başlangıç dizinini belirten özelliktir.
Multiselect	Diyalog penceresi ile birden fazla dosya seçilip seçilemeyeceğini belirten özelliktir.

Bireysel Etkinlik

- OpenFileDialog sınıfının Tablo 10.5'te listelenen özelliklerini kullanan uygulamalar geliştiriniz.

DOSYA KAYDETME PENCERESİ

Kullanıcının kaydetmek amacıyla bir dosya seçmesine olanak sağlayan penceredir. Bu pencerenin ekran görüntüsü Şekil 10.12'de gösterilmiştir.



Şekil 10.12. Dosya kaydetme penceresi ekran görüntüsü.

Ekran görüntüsünden anlaşılabileceği gibi, pencere ufak değişiklikler haricinde dosya açma penceresinin aynısıdır.

Dosya açma penceresi oluşturmak için kullanılan OpenFileDialog sınıfına ait özellikler ile SaveFileDialog sınıfının özelliklerinin pek çoğu ortaktır. Bu ortak özelliklerden daha önce gösterilmeyen bazı özellikler ve sadece SaveFileDialog sınıfına ait bazı özellikler açıklamaları ile birlikte Tablo 10.6'da gösterilmektedir.

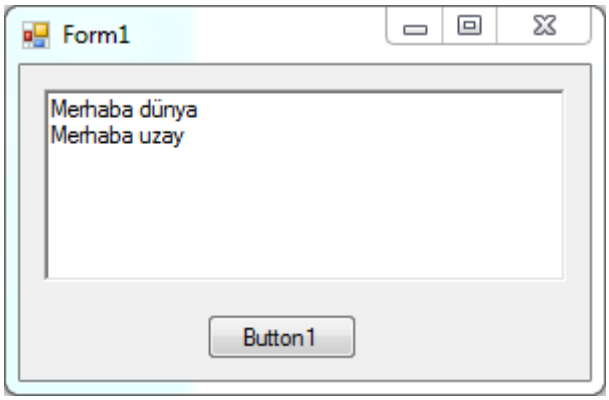
Tablo 10.6. SaveFileDialog sınıfına ait bazı özellikler ve açıklamaları

Özellik	Açıklama
CreatePrompt	Kullanıcı var olmayan bir dosyayı seçmişse bu dosyanın oluşturulması için kullanıcıya sorulup sorulmayacağını belirten özelliktir.
OverwritePrompt	Kullanıcının var olan bir dosyayı seçmesi durumunda kullanıcının uyarılıp uyarılmayacağını belirten özelliktir.
ShowHelp	Diyalog penceresinde "Yardım" (Help) butonunun gösterilip gösterilmeyeceğini belirleyen özelliktir.
Title	Dosya kaydetme penceresinin başlığını belirten özelliktir.

Örnek

- Bir form üzerine bir RichTextBox bileşeni ve buton yerleştiriniz.
- Uygulama çalıştırıldığında RichTextBox bileşenine bir yazı giriniz.
- Ardından butona tıklandığında kullanıcıya dosya kaydetme penceresinin gösterilmesini sağlayınız.
- Eğer kullanıcı kayıt yapılacak bir dosya belirlediyse RichTextBox bileşenindeki yazının ilgili dosyaya kaydedilmesini sağlayınız.

RichTextBox bileşenine yazı girilmiş haliyle örnek uygulamanın ekran görüntüsü Şekil 10.13'te gösterilmiştir.



Şekil 10.13. Dosya kaydetme penceresi örneğinin ekran görüntüsü.

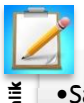
Butona tıklandığında istenen işlemleri gerçekleştiren kod parçası ise aşağıda verilmiştir.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
    Button1.Click
        SaveFileDialog1.Filter = "TXT Files (*.txt*)|*.txt"
        If SaveFileDialog1.ShowDialog = DialogResult.OK Then
            My.Computer.FileSystem.WriteAllText(SaveFileDialog1.FileName,
                                                RichTextBox1.Text, True)
        End If
    End Sub
End Class
```

Bir önceki örneğe benzer şekilde "ShowDialog" metodu kullanılarak dosya kaydetme penceresi kullanıcıya gösterilmiş, kullanıcı eğer bir dosya belirlemişse RichTextBox bileşenindeki yazı bu dosyaya kaydedilmiştir.

Yukarıdakilere ek olarak sadece TXT uzantılı dosyaların kabul edilmesi için "Filter" özelliği kullanılarak SaveFileDialog1 nesnesi için bir filtre tanımlanmıştır.

Seçilen dosya bir editör ile açılacak olursa dosyanın içeriğinin RichTextBox bileşenindeki yazı ile aynı olduğu görülecektir.



Bireysel Etkinlik

- SaveFileDialog sınıfının Tablo 10.6'da listelenen özelliklerini kullanan uygulamalar geliştiriniz.

Özet

- Görsel uygulamalar kullanıcıya bilgi vermek ve kullanıcıdan bilgi almak konularında konsol uygulamalarına kıyasla çok daha esnektirler.
- Diyalog pencereleri görsel uygulamalarda kullanıcıdan bilgi almak için kullanılan.NET platformundaki ön tanımlı pencerelerdir.
- Bu bölümde ColorDialog, FolderBrowserDialog, FontDialog, OpenFileDialog ve SaveFileDialog diyalog pencereleri örneklerle anlatılmıştır.
- ColorDialog kullanıcıdan renk almak için kullanılan standart diyalog penceresidir.
- FolderBrowserDialog kullanıcının sistem dizinleri arasında gezinerek bir dizin seçmesini sağlayan diyalog penceresidir.
- FontDialog kullanıcın bir yazı tipini ve özelliklerini detaylı olarak belirleyebilmesini sağlayan diyalog penceresidir.
- OpenFileDialog ve SaveFileDialog ise sırasıyla dosya açmak ve dosya kayıt etmek amacıyla kullanıcının dosya adını ve patikasını belirtmesini sağlayan diyalog pencereleridir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi diyalog pencerelerinin özelleşmiş pencerelere göre avantajlarından biri değildir?
   a) Kullanıma hazır bileşenler olmaları
   b) Tasarım ve gerçekleştirim gerektirmemeleri
   c) Tamamen özelleştirilebilmeleri
   d) Geliştiriciler tarafından kolayca kullanılabilmeleri
   e) Kullanıcılar tarafından kolayca kullanılabilmeleri

2. Aşağıdakilerden hangisi bu ünitede anlatılan diyalog pencerelerinden biri değildir?
   a) Renk Seçimi Penceresi (ColorDialog)
   b) Dizin Tarayıcısı Penceresi (FolderBrowserDialog)
   c) Font Seçimi Penceresi (FontDialog)
   d) Mesaj Penceresi (MessageDialog)
   e) Dosya Açma Penceresi (OpenFileDialog)

3. Bu ünitede anlatılan diyalog pencereleri aşağıdaki sınıflardan hangisinden türetilmiştir?
   a) CommonDialog
   b) Dialog
   c) BaseDialog
   d) Window
   e) BaseWindow

4. Aşağıdakilerden hangisi Renk Seçimi Penceresi (ColorDialog) özelliklerinden biri değildir?
   a) AllowFullOpen
   b) RootFolder
   c) Color
   d) CustomColors
   e) ShowHelp

5. Aşağıdakilerden hangisi dizin tarayıcısı penceresi (FolderBrowserDialog) özelliklerinden biri değildir?
   a) Description
   b) RootFolder
   c) SelectedPath
   d) FolderName
   e) ShowNewFolderButton

6. Aşağıdakilerden hangisi font seçimi penceresinin (FontDialog) seçilen rengi gösterip göstermeyeceğini belirten özelliğidir?
   a) Color
   b) ShowColor
   c) HideColor
   d) Font
   e) SelectedColor

7. Aşağıdakilerden hangisi dosya açma penceresinde (OpenFileDialog) kullanıcının dosya uzantısını belirtmemesi durumunda dosya uzantısının otomatik eklenip eklenmeyeceğini belirten özelliktir?
   a) DefaultExt
   b) ShowExt
   c) DefaultExt
   d) Multiselect
   e) AutoAdd

8. Aşağıdakilerden hangisi dosya kaydetme penceresinin (OpenFileDialog) başlığını değiştirmekte kullanılabilecek bir özelliktir?
   a) Title
   b) Header
   c) Subject
   d) HeaderName
   e) Window

9. Aşağıdakilerden hangisinde ShowNewFolderButton özelliği bulunmaktadır?
   a) FontDialog
   b) ColorDialog
   c) InputBox
   d) InputDialog
   e) FolderBrowserDialog

10. Aşağıdakilerden hangisinde CheckFileExists özelliği bulunmaktadır?
    a) FolderBrowserDialog
    b) FolderBrowserDialog
    c) InputDialog
    d) OpenFileDialog
    e) FileDialog

Cevap Anahtarı

1.c, 2.d, 3.a, 4.b, 5.d, 6.b, 7.c, 8.a, 9.e

YARARLANILAN KAYNAKLAR

Microsoft Developer Network, Web Site: http://msdn.microsoft.com

Foxall, J. (2015). Teach Yourself Visual Basic 2015 in 24 Hours, Carmel, Sams Publishing.

Türkeli, Y. (2011). Visual Basic. Net Eğitmenim, Ankara, Nirvana Yayınları.

Sole A. D. (2015). Visual Basic 2015 Unleashed, Carmel, Sams Publishing.

DOSYA İŞLEMLERİ

İÇİNDEKİLER

- Dosya İşlemleri
  - Stream
    - Stream Çeşitleri
  - .NET I/O Sınıfları
    - FileStream Sınıfı
    - StreamReader Sınıfı
    - StreamWriter Sınıfı
    - Directory Sınıfı
    - File Sınıfı

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Dosya kavramını açıklayabilecek,
  - Stream nedir ve çeşitleri nelerdir sorusuna cevap verebilecek,
  - .NET I/O Sınıflarını tanımlayabilecek,
  - Bir sürücüdeki dizin ve dosyalar üzerinde işlemler yapabilecek,
  - Bir text dosya üzerinde okuma, yazma ve ekleme işlemlerini gerçekleştirebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Doç. Dr. Ecir Uğur KÜÇÜKSİLLE

ÜNİTE

11

GİRİŞ

Kullanıcılar programlara verileri klavye, fare, mikrofon gibi başka bir sistem tarafından veya kullanıcı tarafından oluşturulmuş dosyalardan da girdi almaktadırlar. Benzer şekilde üretilen çıktı yine doysa olarak saklanabilmektedir. Önceleri çok zorlukla gerçekleştirilebilen dosya işlemleri Visual Basic.NET ile programcıların hizmetine sunulan hazır sınıf kütüphaneleri sayesinde bu özelliklere sahip program yazmak artık oldukça kolay bir hâl almıştır.

Dosya oluşturma, açma, içeriğini okuma gibi Girdi/Çıktı (I/O) işlemleri de bu hazır kütüphaneler arasında yer almaktadır.

Bu ünitede dosya işlemleri; *FileStream, StreamReader, StreamWriter, Directory* ve *File* sınıfları kullanılarak anlatılacaktır.

DOSYA İŞLEMLERİ

Bilgisayarda, içerisinde veri veya bilgi barındıran, bir ismi ve uzantısı olan nesnelere dosya adı verilir. Text dosyalar, program dosyaları vb. gibi, birçok dosya çeşidi bulunmaktadır. Farklı türdeki dosyalar farklı türde bilgi içerirler. Dosyalar bir isme ve uzantıya sahiptirler, uzantıları önemlidir. Çünkü dosyaların uzantısına bakılarak ilgili dosyanın ne tür bir dosya olduğuna karar verilebilir.

Örnek

- docx uzantılı dosyalar Microsoft Word, xlsx uzantılı dosyalar Microsoft Excel, pdf uzantılı dosyalar Adobe Acrobat programlarına aittir.

.NET Framework'de System.IO namespace'i dosya ve dizin işlemlerini gerçekleştirebilmek için gerekli sınıfları içerir. Bu nedenle, dosya ile ilgili işlemleri yaparken bu namespace'in mutlaka programa eklenmesi gerekir.

Stream

Bir dosya okumak veya yazmak için açıldığında stream adını alır.

Bir dosya okumak veya yazmak için açıldığında stream adını alır. Stream, tek boyutlu birbirini takip eden veri kümesi olarak düşünülebilir ve bu veri kümesinin bir başlangıcı ve sonu vardır. Cursor, stream içerisindeki o an bulunulan yeri gösteren bir işaretçidir.

Stream içerisindeki veri bellekten, bir dosyadan veya bir TCP/IP soketten gelebilir. Bir stream üzerinde aşağıdaki 3 işlem gerçekleştirilebilir.

- *Reading:* Bir stream içerisinden veri okunarak, string veya byte dizisi gibi bir veri yapısı içerisine aktarılabilir.
- *Writing:* Bir veri kaynağından okunan veriler bir stream içerisine yazılabilir.
- *Seeking:* Bir stream içerisinde hangi konumda bulunulduğu sorgulanabilir veya konum değiştirilebilir.

Stream çeşitleri

.NET framework'te kullanılan stream çeşitleri ve kullanım yerleri aşağıdaki gibidir.

- *FileStream:* Herhangi bir sürücüdeki dosyayı okumak veya dosyaya yazmak için kullanılır.
- *IsolatedStorageFileStream:* Geliştirilen uygulamaya özel dosyaları oluşturmak, okumak ve yazmak için kullanılır.
- *BufferedStream:* Bir stream üzerinde, okuma veya yazma işlemleri için bir buffer katmanı ekler. Bu sayede okuma ve yazma performansı artar.
- *CryptoStream:* Streamlerin güvenli bir formatta saklanması ve gönderilmesi için kullanılır.
- *MemoryStream:* Veriyi geçici olarak bellekte saklamayı sağlar.
- *NetworkStream:* Ağ üzerinden veri aktarımı için kullanılan temel veri yapısıdır.

Dosya işlemleri için gerekli sınıflar System.IO namespace'inde bulunur.

.NET I/O Sınıfları

System.IO namespace'i dosya oluşturma ve silme, dosyadan okuma ve dosyaya yazma, dosya açma ve kapatma gibi işlemler için birçok sınıf içermektedir. Aşağıdaki tabloda System.IO namespace'i içerisindeki çok kullanılan sınıflar ve kullanım yerleri verilmiştir.

Tablo 11.1. Sınıflar ve Kullanım Yerleri

Sınıf	Kullanım Yeri
BinaryReader	Stream'den temel veri türlerindeki veriyi okumayı sağlar.
BinaryWriter	Stream'e temel veri türlerindeki veriyi yazmayı sağlar.
BufferedStream	Bir stream üzerinde, okuma veya yazma işlemleri için bir buffer katmanı ekler.
Directory	Dizin oluşturma, silme ve taşıma gibi işlemlerin yapılmasını sağlayan statik metodlar sunar.
DirectoryInfo	Spesifik bir dizin üzerinde silme ve taşıma gibi işlemlerin yapılmasını sağlar.
DriveInfo	Sürücüler hakkında bilgi almayı sağlar.
File	Tek bir dosyanın açılması, oluşturulması, taşınması ve silinmesi için statik metotlar sağlar.
FileInfo	Dosyalar üzerinde oluşturma, silme ve taşıma gibi işlemlerin yapılmasını sağlar.
FileStream	Herhangi bir sürücüdeki dosyayı okumak veya dosyaya yazmak için kullanılır.
MemoryStream	Veriyi geçici olarak bellekte saklamayı sağlar.

Path	İstenilen sürücü yolu üzerinde işlemler yapmayı sağlar.
StreamReader	Bir stream'den karakter veya karakterler okumayı sağlar.
StreamWriter	Bir stream'e karakter veya karakterler yazmayı sağlar.
NetworkStream	Ağ üzerinden veri aktarımı için kullanılan temel veri yapısıdır.

Tablo 11.1'de verilen tüm sınıfları ve bu sınıflara ait metodları anlatmak bu bölüm dâhilinde mümkün değildir. Bu nedenle uygulamalarda çok kullanılan sınıflar, bu sınıflara ait çok kullanılan metotlar ve özellikleri hakkında bilgi verilecek, her anlatılan sınıfla ilgili örnekler yapılacaktır.

FileStream Sınıfı

FileStream sınıfı, herhangi bir sürücüdeki dosyayı okumak veya bu dosyaya yazmak için kullanılır. Bu başlık altında, FileStream sınıfının en çok kullanılan kurucu metodu, diğer metot ve özellikleri hakkında bilgi verilecektir. Ayrıca bir uygulama ile de kullanımı pekiştirilecektir.

Kurucu Metot:
FileStream(yol AS String , mod AS FileMode, erişim AS FileAccess, paylaşım AS FileShare)

Yol: Üzerinde işlem yapılmak istenen dosyanın yolunun ve isminin verildiği parametredir.

Mod: Dosyanın hangi modda açılacağının belirlendiği parametredir. Alabildiği değerler ve anlamları aşağıda belirtilmiştir.

- *Append:* Eğer dosya varsa dosyayı bilgi eklemek amacı ile açar ve cursor'ü dosyanın sonuna konumlandırır. Eğer dosya yoksa dosyayı oluşturur.
- *Create:* Yeni bir dosya oluşturmak için kullanılır. Eğer oluşturulmak istenen dosya daha önceden mevcutsa varolan dosyanın üzerine yazar.
- *CreateNew:* Yeni bir dosya oluşturmak için kullanılır. Eğer oluşturulmak istenen dosya mevcutsa IOException ayrıcalıklı durumu oluşur.
- *Open:* Varolan bir dosyayı açmak için kullanılır. Eğer açılmak istenen dosya belirtilen yolda bulunamamışsa FileNotFoundException ayrıcalıklı durumu oluşur
- *OpenOrCreate:* Eğer dosya belirtilen dosya mevcutsa açılır aksi takdirde belirtilen isimde yeni bir dosya oluşturulur.
- *Truncate:* Varolan dosya açılır ve büyüklüğü 0 byte'a düşürülür.

Bir dosya okumak, yazmak veya hem okumak hem de yazmak için açılabilir.

Erişim: Açılan veya oluşturulan dosyaya hangi amaçla ulaşılacağını belirlemek için kullanılan parametredir. Kullanılabilecek değerler Read, ReadWrite ve Write olarak belirlenmiştir. Eğer Read kullanılırsa dosyaya okumak için ulaşılacağı, ReadWrite kullanılırsa hem okumak hem de yazmak için ulaşılacağı, Write kullanılırsa yazmak için ulaşılacağı belirlenmiş olur.

Paylaşım: Üzerinde işlem yapılan dosyaya diğer processler tarafından ulaşılıp ulaşılamayacağını belirlemek için kullanılan parametredir. Bu parametrenin alabileceği değerler ve anlamları aşağıda verilmiştir.

- *None:* Üzerinde çalışılan dosyanın diğer processler tarafından kullanımına izin vermez.
- *Read:* Üzerinde çalışılan dosyaya diğer processlerin de okuma izni ile ulaşmasına izin verir.
- *ReadWrite:* Üzerinde çalışılan dosyaya diğer processlerin de okuma ve yazma izni ile ulaşmasına izin verir.
- *Write:* Üzerinde çalışılan dosyaya diğer processlerin de yazma izni ile ulaşmasına izin verir.

Özellikler:

Tablo 11.2. FileStream Sınıfı Özellikleri

Özellik	İşlev
CanRead	Stream'in okumayı destekleyip desteklemediğini öğrenmeyi sağlar. Boolean değer döndürür.
CanSeek	Stream'in aramayı destekleyip desteklemediğini öğrenmeyi sağlar. Boolean değer döndürür.
CanWrite	Stream'in yazmayı destekleyip desteklemediğini öğrenmeyi sağlar. Boolean değer döndürür.
Length	Stream'in byte olarak uzunluğunu öğrenmeyi sağlar. Long değer döndürür.
Position	Cursor'ün yerini öğrenmeyi veya değiştirmeyi sağlar. Long değer alır ve Long değer döndürür.

Metotlar:

Tablo 11.3. FileStream Sınıfı Metodları

Bir dosyada cursor'ü istenen yere konumlandırmak için seek metodu kullanılır.

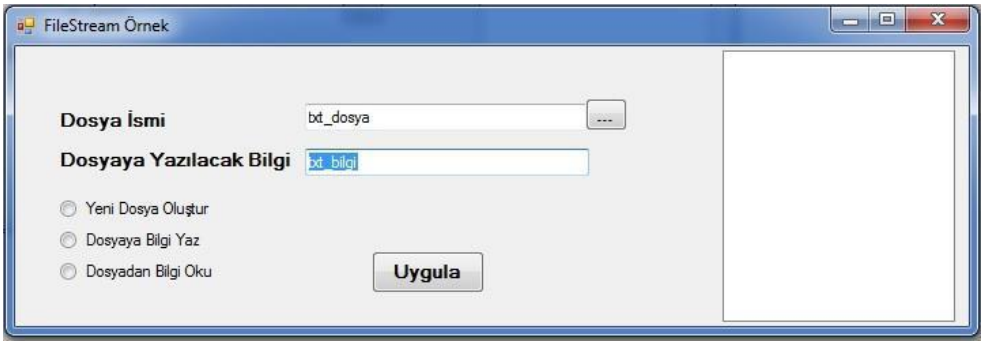
Metod	İşlev
Close()	Stream'i kapatır ve bu stream'e ait olan kaynakları serbest bırakır.
CopyTo(hedef AS Stream)	Kaynak stream nesnesi içeriğini hedef stream nesnesine aktarmayı sağlar.
Dispose()	Stream'in kullandığı tüm kaynakları serbest bırakmayı sağlar.
Flush()	Stream'e ait buffer'ı temizler ve buffer'daki bütün verilerin dosyaya yazılmasını sağlar.
Read(buffer AS Byte(),indis AS Integer,sayi AS Integer) AS Integer	Üçüncü parametrede verilen byte kadar bilgiyi birinci parametrede verilen diziye ikinci parametrede verilen indisten itibaren yerleştirir. Toplam okunan byte sayısını geri döndürür.
ReadByte() AS Integer	Stream'de cursor'un bulunduğu noktadan

	itibaren bir bytelık veri okumayı sağlar. Eğer -1 değeri dönerse, dosya sonuna gelinmiş demektir.
Seek(konum AS Long,başlangıç AS SeekOrigin) AS Long	Stream'deki cursor'ün ikinci parametrede verilen başlangıçtan itibaren birinci parametrede verilen pozisyona konumlanmasını sağlar. Stream içerisindeki yeni pozisyonu geri döndürür.
Write(buffer AS Byte(),indis AS Integer,sayi AS Integer)	Stream'e birinci parametrede verilen veri dizisindeki bilgilerin ikinci parametrede verilen indisten itibaren üçüncü parametrede verilen sayı kadar yazılmasını sağlar.
WriteByte(veri AS Byte)	Stream'e ilk parametrede verilen bir byte'lık veriyi yazmayı sağlar.

Bireysel Etkinlik

- FileStream sınıfına ait diğer metodları bölüm sonunda verilen kaynaklardan inceleyerek uygulayınız.

Bütün bu açıklamalardan sonra hem metotların kullanımlarını daha iyi anlayabilmek hem de uygulama yaparak pekiştirmek amacı ile bir örnek yapılacaktır. Bu örnek için öncelikle bir windows form tasarlanmış ve Şekil 11.1.'de tasarlanan bu form gösterilmiştir. Bu formun amacı FileStream sınıfını kullanarak bir dosya oluşturmak, varolan bir dosyaya bilgi eklemek ve yine var olan bir dosyadan bilgileri çekerek listbox nesnesine eklemektir. Dosya seçmek amacı ile OpenFileDialog nesnesi kullanılmıştır. Şekil 11.1.'in hemen altında da bu formun işlevini yerine getirebilmesi için yazılan kodlar verilmiştir.



Şekil 11.1. FileStream Sınıfı İçin Tasarlanan Form

```
Imports System.IO
Imports System.Text
Public Class Form1

    Private Sub btn_uygula_Click(sender As Object, e As EventArgs) Handles
btn_uygula.Click
        Dim dosya As FileStream
        Dim kodlama As New UnicodeEncoding()
        If rdb_yeni.Checked Then
            If File.Exists(txt_dosya.Text) Then
                MsgBox("Bu dosya zaten mevcut")
            Else
                dosya = New FileStream(txt_dosya.Text, FileMode.CreateNew,
FileAccess.ReadWrite, FileShare.None)
                MsgBox("Dosya Oluşturuldu")
                dosya.Close()
            End If
        End If

        If rdb_yaz.Checked Then
            dosya = New FileStream(txt_dosya.Text, FileMode.Append,
FileAccess.Write, FileShare.None)
            dosya.Write(kodlama.GetBytes(txt_bilgi.Text), 0,
kodlama.GetByteCount(txt_bilgi.Text))
            dosya.Write(kodlama.GetBytes(Environment.NewLine), 0,
kodlama.GetByteCount(Environment.NewLine))
            dosya.Close()
        End If

        If rdb_oku.Checked Then
            dosya = New FileStream(txt_dosya.Text, FileMode.Open,
FileAccess.Read, FileShare.None)
            Dim dizi As Byte() = New Byte(dosya.Length) {}
            dosya.Read(dizi, 0, dizi.Length)
```

```
Dim satirlar As String() = kodlama.GetString(dizi).Split(Environment.NewLine)
            For index = 0 To satirlar.Length - 2
                lst_bilgi.Items.Add(satirlar(index))
            Next
            dosya.Close()
        End If
    End Sub

    Private Sub btn_dsec_Click(sender As Object, e As EventArgs) Handles
btn_dsec.Click
        If fdialog.ShowDialog() = DialogResult.OK Then
            txt_dosya.Text = fdialog.FileName
        End If
    End Sub
End Class
```

StreamReader Sınıfı

StreamReader sınıfı bir dosyadan karakter veya karakterler okumayı sağlar. Bu başlık altında, StreamReader sınıfının en çok kullanılan kurucu metotları, diğer metot ve özellikleri hakkında bilgi verilecektir

Kurucu Metotlar:

StreamReader(yol AS String)

Yol: Okunmak istenen dosyanın yolunun verildiği parametredir.

StreamReader(yol AS String, kodlama AS Encoding)

Yol: Okunmak istenen dosyanın yolunun verildiği parametredir.

Kodlama: Dosyanın istenen karakter kodlama sistemi ile okunmasını sağlar.

Özellikler:

Tablo 11.4. StreamReader Sınıfı Özellikleri

Özellik	İşlev
CurrentEncoding	StreamReader nesnesinin kullandığı karakter kodlama sistemini öğrenmeyi sağlar.
EndOfStream	Cursor'ün stream'in sonunda olup olmadığını öğrenmeyi sağlar. Boolean değer döndürür.

Metotlar:

Tablo 11.5. StreamReader Sınıfı Metodları

Metod	İşlev
Close()	Stream'i kapatır ve bu stream'e ait olan kaynakları serbest bırakır.
Dispose()	Stream'in kullandığı tüm kaynakları serbest bırakmayı sağlar.
Peek() AS Integer	Sırada okunacak karakter olup olmadığını öğrenmeyi sağlar. Geriye Integer değer döndürür. -1 dönerse sırada okunacak karakter yoktur.
Read() AS Integer	Sıradaki karakteri okumayı sağlar. Geriye Integer değer döndürür. -1 dönerse sırada okunacak karakter yoktur.
Read(buffer AS Byte(),indis AS Integer,sayi AS Integer) AS Integer	Üçüncü parametrede verilen byte kadar bilgiyi birinci parametrede verilen diziye ikinci parametrede verilen indisten itibaren yerleştirir. Toplam okunan byte sayısını geri döndürür.
ReadLine() AS String	Sıradaki satırı okumayı sağlar.
ReadToEnd() AS String	Stream'de cursor'un bulunduğu pozisyondan stream'in sonuna kadar tüm karakterleri okumayı sağlar.

Bir stream'in kullanacağı karakter kodlaması için Encoding sınıfı kullanılır.

StreamReader sınıfı ile ilgili örnek uygulama StreamWriter sınıfı anlatıldıktan sonra yapılacaktır.

- StreamReader sınıfına ait diğer metodları bölüm sonunda verilen kaynaklardan inceleyerek uygulayınız.

StreamWriter Sınıfı

StreamWriter sınıfı bir dosyaya karakter veya karakterler yazmayı sağlar. Bu başlık altında, StreamReader sınıfının en çok kullanılan kurucu metodu, diğer metod ve özellikleri hakkında bilgi verilecektir. Ayrıca bir uygulama ile de kullanımı pekiştirilecektir.

Kurucu Metotlar:

StreamWriter(yol AS String)

Yol: Bilgi yazılmak istenen dosyanın yolunun verildiği parametredir.

StreamReader(yol AS String,bilgi AS Boolean ,kodlama AS Encoding)

Yol: Okunmak istenen dosyanın yolunun verildiği parametredir.

Bilgi: Eğer true verilirse yazılan bilgi dosyaya eklenir, eğer false verilirse dosya üzerine yazılır.

Kodlama: Dosyanın istenen karakter kodlama sistemi ile okunmasını sağlar.

Özellikler:

Tablo 11.6. StreamWriter Sınıfı Özellikleri

Özellik	İşlev
AutoFlush	Her yazma işleminden sonra buffer'daki bilginin dosyaya yazılması isteniyorsa true, aksi takdirde false yapılmalıdır.
Encoding	StreamWriter nesnesinin kullandığı karakter kodlama sistemini öğrenmeyi sağlar.
NewLine	Yeni satıra geçme karakterini öğrenmeyi veya değiştirmeyi sağlar.

Metodlar:

Tablo 11.7. StreamWriter Sınıfı Metotları

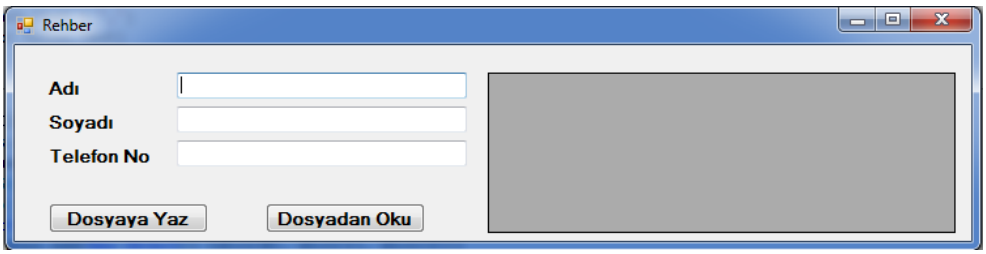
Metod	İşlev
Close()	Stream'i kapatır ve bu stream'e ait olan kaynakları serbest bırakır.
Dispose()	Stream'in kullandığı tüm kaynakları serbest bırakmayı sağlar.
Flush()	Bütün buffer'ı temizler ve buffer'daki tüm veriyi dosyaya yazar.
Write(veri AS veritipi)	Write metodu tüm temel veri tiplerini içeren overload metodlara sahiptir. Stream'e ilgili veri tipindeki veriyi yazmayı sağlar.
WriteLine()	Strem'e yeni satıra geçme karakterinin yazılmasını sağlar.
WriteLine(veri AS veritipi)	WriteLine metodu tüm temel veri tiplerini içeren overload metodlara sahiptir. Stream'e ilgili veri tipindeki veriyi yazdıktan sonra yeni satıra geçme karakterini de yazmayı sağlar.

Bir stream'in kullandığı buffer'ı temzilemek ve verileri dosyaya yazmak için Flush() metodu kullanılır.

Bireysel Etkinlik

- StreamWriter sınıfına ait diğer metodları bölüm sonunda verilen kaynaklardan inceleyerek uygulayınız.

Bütün bu açıklamalardan sonra hem metotların kullanımlarını daha iyi anlayabilmek hem de uygulama yaparak pekiştirmek amacı ile bir örnek yapılacaktır. Bu örnek için öncelikle bir windows form tasarlanmış ve Şekil 11.2.'de tasarlanan bu form gösterilmiştir. Bu formun amacı kişinin adını, soyadını ve telefon numarasını "rehber.txt" dosyasında saklayan ve gerektiğinde listeleyebilen bir programı StreamWriter ve StreamReader sınıflarını kullanarak gerçekleştirebilmektir.



Şekil 11.2. StreamReader ve StreamWriter Sınıfı İçin Tasarlanan Form

```
Imports System.IO
Imports System.Text
Public Class Form1
    Private Sub dosyayaz_Click(sender As Object, e As EventArgs) Handles
dosyayaz.Click
        Dim dosya As New StreamWriter(Directory.GetCurrentDirectory() &
"\rehber.txt", True, Encoding.Unicode)
        dosya.Write(txt_adi.Text + ",")
        dosya.Write(txt_sadi.Text + ",")
        dosya.WriteLine(txt_telno.Text)
        dosya.Close()
    End Sub

    Private Sub dosyaoku_Click(sender As Object, e As EventArgs) Handles
dosyaoku.Click
        DataGridView1.Rows.Clear()
        DataGridView1.ColumnCount = 3
        Dim dosya As New StreamReader(Directory.GetCurrentDirectory() &
"\rehber.txt", Encoding.Unicode)
        While dosya.Peek() >= 0
            Dim dizi As String() = dosya.ReadLine().Split(",")
            DataGridView1.Rows.Add(New Object() {dizi(0), dizi(1), dizi(2)})
        End While
        dosya.Close()

    End Sub
End Class
```

Directory sınıfı

Dizinler üzerinde oluşturma, taşıma, bazı özelliklerini öğrenme ve değiştirme gibi işlemler yapmak için statik metotlar sunan bir sınıftır. Bu başlık altında Directory sınıfının çok kullanılan metotları hakkında bilgi verilecektir.

Metotlar:

Tablo 11.8. Directory Sınıfı Metotları

Metot	İşlev
CreateDirectory(yol AS String) AS DirectoryInfo	Parametre olarak verilen yolda belirtilen klasörü oluşturur.
Delete(yol AS String)	Parametre olarak verilen yoldaki boş klasörü siler.
Delete(yol AS String, ozyineleme AS Boolean)	Eğer ikinci parametre true verilirse, parametre olarak verilen yolda bulunan dizin içerisindeki tüm alt dizinleri ve dosyaları siler.
EnumerateDirectories(yol AS String) AS IEnumerable(Of String)	Parametre olarak gönderilen yolda bulunan dizindeki tüm alt dizin isimlerini (yolları ile beraber) bir koleksiyon olarak geri döndürür.

EnumerateFiles(yol AS String) AS IEnumerable(Of String)	Parametre olarak gönderilen yolda bulunan dizindeki tüm dosya isimlerini (yolları ile beraber) bir koleksiyon olarak geri döndürür.
Exists(yol AS String) AS Boolean	Parametre olarak gönderilen yolda aynı isimli bir klasör olup olmadığını Boolean olarak geri döndürür.
GetCreationTime(yol as String) AS DateTime	Parametre olarak gönderilen yolda bulunan dizinin oluşturulma tarih ve saatini geri döndürür.
GetCurrentDirectory() AS String	Uygulamanın çalıştığı yolu verir.
GetDirectories(yol AS String) AS String()	Parametre olarak gönderilen yolda bulunan dizindeki tüm alt dizin isimlerini (yolları ile beraber) bir String dizisi olarak geri döndürür.
GetFiles(yol AS String) AS String()	Parametre olarak gönderilen yolda bulunan dizindeki tüm dosya isimlerini (yolları ile beraber) bir String dizisi olarak geri döndürür.
GetLastAccessTime(yol AS String) AS DateTime	Parametre olarak gönderilen yolda bulunan dizine son ulaşılma tarih ve saatini geri döndürür.
GetLogicalDrives() AS String()	Sistemdeki tüm mantıksal sürücüleri bir String dizisi olarak geri döndürür.
Move(kaynak AS String, hedef AS String)	Birinci parametre olarak verilen yoldaki kaynak dizini ikinci parametre olarak verilen yoldaki hedef dizine taşır.
SetCreationTime(yol AS String, zaman AS DateTime)	Parametre olarak gönderilen yolda bulunan dizinin oluşturulma zamanını parametre olarak gönderilen zaman yapar.
SetLastAccessTime(yol AS String, zaman AS DateTime)	Parametre olarak gönderilen yolda bulunan dizine son erişim zamanını parametre olarak gönderilen zaman yapar.
SetLastWriteTime(yol AS String, zaman AS DateTime)	Parametre olarak gönderilen yolda bulunan dizine son yazma zamanını parametre olarak gönderilen zaman yapar.

Directory sınıfı statik metodlar sağlar.

Directory ve DirectoryInfo sınıfları arasındaki farkı araştırarak, bir örnek yapınız.

Bu başlığa ait örnek, daha anlamlı olacağı için File sınıfı anlatıldıktan sonra yapılacaktır.

File sınıfı

Bir dosyayı oluşturma, kopyalama, silme, taşıma ve açma gibi işlemleri yapmak için statik metotlar sunan bir sınıftır. . Bu başlık altında Directory sınıfının çok kullanılan metotları hakkında bilgi verilecek ve bir örnek uygulama ile de kullanımı pekiştirilecektir.

Metotlar:

Tablo 11.9. File Sınıfı Metodları

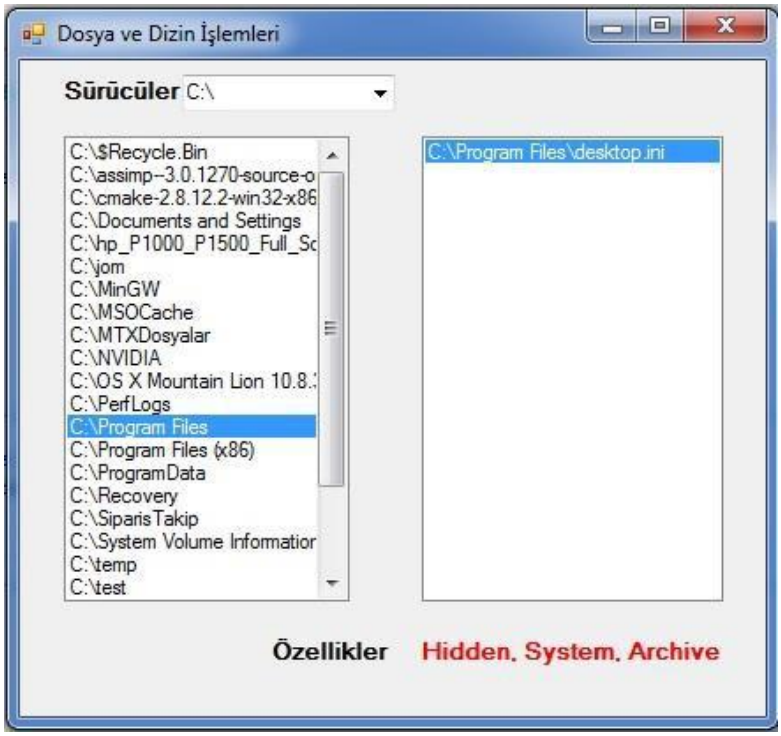
Metod	İşlev
Copy(kaynak AS String, hedef AS String)	Birinci parametre olarak verilen yoldaki kaynak dosyayı, ikinci parametre olarak verilen yola kopyalar.
Create(yol AS String) AS FileStream	Parametre olarak verilen yola ilgili dosyayı oluşturur. Eğer aynı dosya varsa üzerine yazar. Geriye oluşturulan dosyaya ait FileStream sınıfında bir nesne döndürür.
Delete(yol AS String)	Parametre olarak verilen yoldaki dosyayı silmeyi sağlar.
Exists(yol AS String) AS Boolean	Parametre olarak gönderilen yolda ilgili dosyanın bulunup bulunmadığı hakkında bilgi verir. Boolean bilgi döndürür.
GetAttributes(yol AS String) AS FileAttributes	Parametre olarak verilen yoldaki dosyanın özelliklerini öğrenmeyi sağlar. Bilgi FileAttributes enumeration olarak geri döndürülür.
GetCreationTime(yol AS String) AS DateTime	Parametre olarak verilen yoldaki dosyanın oluşturulma zamanını geri döndürür.
GetLastAccessTime(yol AS String) AS DateTime	Parametre olarak verilen yoldaki dosyaya son erişim zamanını geri döndürür.
GetLastWriteTime(yol AS String) AS DateTime	Parametre olarak verilen yoldaki dosyaya son yazma zamanını geri döndürür.
Move(kaynak AS String, hedef AS String)	Birinci parametre olarak verilen yoldaki kaynak dosyayı ikinci parametre olarak verilen yola taşır.
Open(yol AS String,mod FileMode) AS FileStream	Birinci parametrede verilen yolda bulunan dosyayı, ikinci parametrede verilen mod'da açar. Geriye açılan dosyaya ait FileStream sınıfında bir nesne döndürür.
OpenRead(yol AS String) AS FileStream	Birinci parametrede verilen yolda bulunan dosyayı okumak için açar. Geriye açılan dosyaya ait FileStream sınıfında bir nesne döndürür.
OpenWrite(yol AS String) AS FileStream	Birinci parametrede verilen yolda bulunan dosyayı yazmak için açar veya dosya yoksa dosyayı yazmak için oluşturur. Geriye açılan veya oluşturulan dosyaya ait FileStream

	sınıfında bir nesne döndürür.
ReadAllLines(yol AS String) AS String()	Birinci parametrede verilen yolda bulunan dosyayı açar, bütün satırları okuyarak bir String diziye aktararak geri döndürür ve dosyayı kapatır.
SetAttributes(yol AS String, ozellik AS FileAttributes)	Birinci parametrede verilen yolda bulunan dosyaya ikinci parametrede verilen özellikleri atar.
SetCreationTime(yol AS String, zaman AS DateTime)	Parametre olarak gönderilen yolda bulunan dosyanın oluşturulma zamanını parametre olarak gönderilen zaman yapar.
SetLastAccessTime(yol AS String, zaman AS DateTime)	Parametre olarak gönderilen yolda bulunan dosyaya son erişim zamanını parametre olarak gönderilen zaman yapar.
SetLastWriteTime(yol AS String, zaman AS DateTime)	Parametre olarak gönderilen yolda bulunan dosyaya son yazma zamanını parametre olarak gönderilen zaman yapar.
WriteAllLines(yol AS String, icerik AS String())	İlk parametrede verilen dosyayı oluşturur, ikinci parametrede verilen içerik dizisindeki tüm elemanları bu dosyaya yazar ve dosyayı kapatır.

Bireysel Etkinlik

- File ve FileInfo sınıfları arasındaki farkı araştırarak, bir örnek yapınız.

Bütün bu açıklamalardan sonra hem metotların kullanımlarını daha iyi anlayabilmek hem de uygulama yaparak pekiştirmek amacı ile Directory ve File sınıflarını içeren bir örnek yapılacaktır. Bu örnek için öncelikle bir windows form tasarlanmış ve Şekil 11.3.'te tasarlanan bu form gösterilmiştir. Bu formun amacı, öncelikle sistemdeki sürücüleri form açıldığında bir combobox'ın içerisinde listelemektir. Daha sonra, ilgili combobox'tan bir sürücü seçildiğinde bu sürücüdeki dizinler lst_ana isimli listbox nesnesinde listelenecektir. lst_ana isimli listbox'tan bir dizin seçildiğinde ise eğer bu dizin içerisinde dosyalar var ise bu dosyalar lst_detay listbox'ında listelenmektedir. Son olarak da lst_detay listbox'ından bir dosya seçildiğinde bu dosyaya ait özellikler lbl_ozellik isimli label'da gösterilmektedir.



Şekil 11.3. Directory ve File Sınıfları İçin Tasarlanan Form

```
Imports System.IO

Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim suruculer As String() = Directory.GetLogicalDrives()
        For Each isim As String In suruculer
            cmb_suruculer.Items.Add(isim)
        Next
    End Sub

    Private Sub cmb_suruculer_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles cmb_suruculer.SelectedIndexChanged
        lst_ana.Items.Clear()
        Try
            Dim surucu_dizin As List(Of String) = New List(Of
String)(Directory.EnumerateDirectories(cmb_suruculer.Text))
            For Each dizin As String In surucu_dizin
                lst_ana.Items.Add(dizin)
            Next
        Catch hata As Exception
            MsgBox(hata.Message)
        End Try
    End Sub

    Private Sub lst_ana_Click(sender As Object, e As EventArgs) Handles
lst_ana.Click
        lst_detay.Items.Clear()
        Try
            Dim dosya As String() = Directory.GetFiles(lst_ana.SelectedItem)
            If dosya.Length <> 0 Then
                For Each dosya_ismi As String In dosya
                    lst_detay.Items.Add(dosya_ismi)
                Next
            Else
                MsgBox("Dosya İçermiyor")
            End If

        Catch hata As Exception
            MsgBox(hata.Message)
        End Try

    End Sub

    Private Sub lst_detay_Click(sender As Object, e As EventArgs) Handles
lst_detay.Click
        Dim ozellik As FileAttributes = File.GetAttributes(lst_detay.SelectedItem)
        lbl_ozellik.Text = ozellik.ToString()
    End Sub
End Class
```

Özet

- Bilgisayarda, içerisinde veri veya bilgi barındıran, bir ismi ve uzantısı olan nesnelere dosya adı verilir. Her dosya bir isme ve bir uzantıya sahiptir. Dosyaların uzantılarına bakılarak hangi programlar tarafından oluşturulduklarına ve dolayısı ile hangi programlar yardımı ile açılabileceklerine karar verilebilir.
- Visula Basic .NET ile dosyalar üzerinde işlem yapabilmek için .NET çatısı içerisinde System.IO namespace'i altında birçok sınıf tanımlanmıştır.
- Bir dosya okumak veya yazmak için açıldığında stream adını alır.
- FileStream sınıfı, herhangi bir sürücüdeki dosyayı okumak veya bu dosyaya yazmak için kullanılır.
- StreamReader sınıfı yardımı ile bir dosyadan karakter ve karakterler okunabilir.
- StreamWriter sınıfı yardımı ile bir dosyaya karakter ve karakterler yazılabilir.
- System.IO namespace'i altında aynı zamanda dizin ve dosyalar üzerinde işlemler gerçekleştirmek için de Directory ve File isimli statik metodlar sağlayan iki sınıf tanımlanmıştır.
- Directory sınıfı yardımı ile, dizinler üzerinde oluşturma, taşıma, bazı özelliklerini öğrenme ve değiştirme gibi işlemler yapılabilmektedir.
- File sınıfı yardımı ile ise, bir dosyayı oluşturma, kopyalama, silme, taşıma ve açma gibi işlemler gerçekleştirilebilmektedir.

DEĞERLENDİRME SORULARI

1. Bir dosyanın okumak veya yazmak için açıldığında aldığı isme ne ad verilir?
   a) File
   b) Directory
   c) Reader
   d) Stream
   e) Writer

2. Aşağıdakilerden hangisi bir stream çeşidi değildir?
   a) NetworkStream
   b) DirectoryStream
   c) FileStream
   d) MemoryStream
   e) BufferedStream

3. Bir FileStream nesnesinin kaç byte olduğunu öğrenmek için kullanılan özellik aşağıdakilerden hangisidir?
   a) Length
   b) Position
   c) Measure
   d) Long
   e) Meter

4. Aşağıdakilerden hangisi Sytem.IO namepace'i ne ait sınıflardan birisi değildir?
   a) FileStream
   b) PathStream
   c) StreamReader
   d) DirectoryInfo
   e) FileInfo

5. FileStream sınıfında buffer'ı temizleyerek verileri dosyaya yazan metod aşağıdakilerden hangisidir?
   a) Immediate()
   b) ClearWrite()
   c) Flush()
   d) Write()
   e) BufferWrite()

6. StreamReader sınıfında sırada okunacak karakter olup olmadığı hangi metotla öğrenilir?
   a) Exists()
   b) Read()
   c) EOF()
   d) Available()
   e) Peek()

7. Bir stream'in hangi kodlama sistemini kullanacağı hangi sınıfla belirlenir?
   a) Encode
   b) Code
   c) StreamCoding
   d) Encoding
   e) Coding

8. Aşağıdakilerden hangisi FileStream sınıfı kurucu metodunda kullanılan dosya modlarından biri değildir?
   a) Close
   b) Truncate
   c) Create
   d) Append
   e) Open

9. Bir bilgisayardaki sürücüleri öğrenmek için kullanılan metot aşağıdakilerden hangisidir?
   a) GetAllDrivers()
   b) GetLogicalDrivers()
   c) GetDrivers()
   d) GetDirives()
   e) GetLogicalDrives()

10. Bir dosyanın özelliklerini öğrenmek için kullanılan metot aşağıdakilerden hangisidir?
    a) GetProperties()
    b) GetFeatures()
    c) GetAttributes()
    d) GetCharacteristics()
    e) GetQualities()

Cevap Anahtarı

1.d,2.b,3.a,4.b,5.c,6.e,7.d,8.a,9.e,10.c

YARARLANILAN KAYNAKLAR

http://msdn.microsoft.com

http://www.tutorialspoint.com/vb.net/vb.net_file_handling.htm

YAZDIRMA İŞLEMLERİ

İÇİNDEKİLER

HEDEFLER

Bu üniteyi çalıştıktan sonra;

- Rapor kavramını açıklayabilecek,
- PrintForm sınıfını kullanarak hazırladığınız bir formu yazdırabilecek,
- CrystalReport hakkında bilgi sahibi olabilecek,
- Crystal Report kullanarak uygulamalarınızda kullanabileceğiniz raporlar hazırlayabileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Doç. Dr. Ecir Uğur KÜÇÜKSİLLE

ÜNİTE

12

GİRİŞ

Geliştirilen programların amaçlarından biri bilgisayar girdi olarak sunulan verinin işlenerek çıktı elde edilmesidir. Bu çıktılar ekranda gösterildiği gibi basılı olarak da elde edilebilir. Visual Basic.NET sınıf kütüphanesi birçok alanda olduğu gibi yazdırma işlemlerinde de birtakım hazır sınıflar barındırmaktadır. Hazırlanan bir formun çıktısını alabilmek için *PrintForm* adlı sınıf kullanılır.

Bu ünitede form yazdırma ve bir dosyayı kullanarak rapor hazırlama işlemlerine değinilmiştir. Rapor hazırlamada ise *CrystalReports kullanılmıştır.*

YAZDIRMA İŞLEMLERİ

Visual Basic.NET'de yazdırma işlemleri iki başlık altında incelenebilir. Bunlardan birincisi, hazırlanan formun yazıcıya aktarılarak kâğıt üzerine bastırılmasıdır. İkincisi ise, bir dosya veya veritabanı'ndan görülmek istenen bilgilerin çekilerek kâğıt üzerine bastırılmasıdır. Bu bölüm içerisinde sırası ile bu iki işlemin nasıl gerçekleştirilebileceği ele alınacaktır.

Dizayn edilen bir formun yazıcıdan çıktısını alabilmek için PrintForm sınıfı kullanılır.

Form Yazdırma

Hazırlanan bir formun yazıcıya aktarılarak kâğıt üzerine bastırılması işlemidir. Bu işlemi gerçekleştirebilmek için, Visual Basic .NET'te Microsoft.VisualBasic.PowerPacks.Printing namespace'i içerisinde bulunan PrintForm sınıfı kullanılmaktadır. Bu sınıf yardımı ile form bir resim olarak ön izleme penceresine, dosyaya veya yazıcıya aktarılabilmektedir. Standart olarak, formun başlık çubuğu, kaydırma çubukları ve çerçevesi yazdırılmamaktadır. Sadece formun görünür kısmı yazdırılmaktadır.

Bu sınıfı projenize eklemek istediğiniz zaman ilgili namespace'i öncelikle Import etmeniz gerekir. Eğer Import satırını yazarken ilgili namespace'e ulaşamazsanız, bu referansı projenize eklemelisiniz. Bunu yapmak için , öncelikle sağ taraftaki Solution Explorer penceresinde projenizin ismi üzerinde sağ tıklayın. Daha sonra gelen menüden Add Reference seçeneğini seçin. Son olarak da karşınıza gelen Reference Manager penceresinde Assemblies altındaki Extensions seçeneğinden ilgili referansı seçerek OK tuşuna basın.

Şimdi sıra ile bu sınıfın çok kullanılan kurucu metotları, özellikleri, metodları, olayları hakkında bilgi verilecek ve daha sonra da anlatılanları pekiştirmek amacı ile bir örnek uygulama yapılacaktır.

Kurucu Metotlar:

PrintForm()

Hiç parametre almayan kurucu metottur. İlgili özellikler daha sonra akatarılmaktadır.

PrintForm(form AS Form)

Form: Yazdırılacak form nesnesinin verildiği parametredir.

Özellikler:

Tablo 12.1. PrintForm Sınıfı Özellikleri

Özellik	İşlev
Form	Yazdırılacak formu belirlemeyi veya öğrenmeyi sağlar.
PrintAction	Yazdırma işleminin bir ön izleme penceresine mi, yazıcıya mı yoksa dosyaya mı yapılacağını belirlemeyi veya öğrenmeyi sağlar.
PrinterSettings	Yazdırma işleminin yapılacağı yazıcının özelliklerini belirlemeyi veya öğrenmeyi sağlar. Örneğin; kaç kopya yazdırılacağı, yazdırma işleminin yapılacağı yazıcı ismi gibi.

Metotlar:

Tablo 12.2. PrintForm Sınıfı Metodları

Özellik	İşlev
Dispose()	Nesnenin kullandığı tüm kaynakları serbest bırakmasını sağlar.
Print()	PrintAction özelliğinde belirtilen hedefe ilgili formun yazdırılmasını sağlar.
Print(form AS Form, yazımozellik AS PrintForm.PrintOption)	Birinci parametrede verilen formun, ikinci parametrede verilen özellik ile yazdırılmasını sağlar.

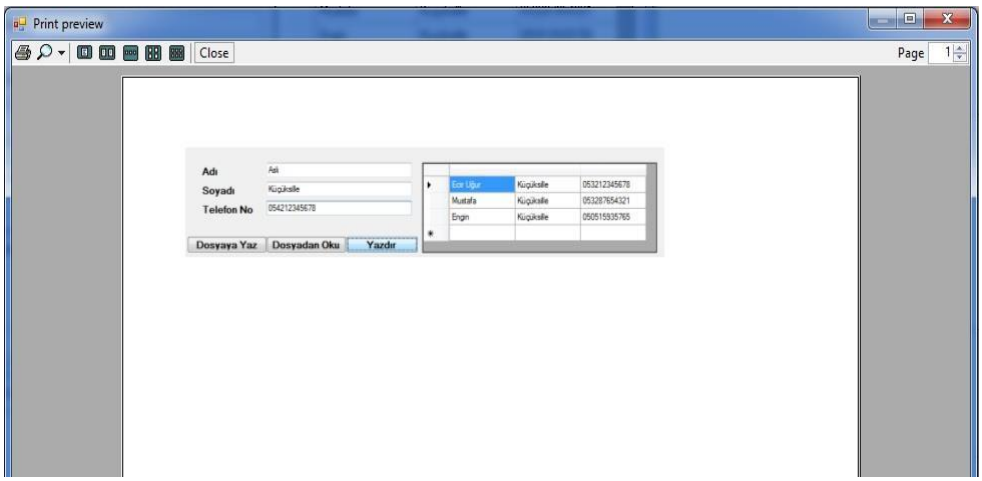
Olaylar:

Tablo 12.3. PrintForm Sınıfı Olayları

Özellik	İşlev
BeginPrint	Dokümanın ilk sayfası yazdırılmadan önce meydana gelir.
EndPrint	Dokümanın son sayfası yazdırıldıktan sonra meydana gelir.
QueryPageSettings	Her sayfa yazdırılmadan önce meydana gelir.

Bu başlık altındaki örnek için bir önceki bölümde yapılmış olan örnek üzerine ekleme yapılacaktır. Yapılan ekleme ile yeni oluşan form ekranı Şekil 12.1.'de verilmiştir. Şekil 12.2.'de ise Yazdır butonuna basıldığında kullanıcı karşısına gelen baskı ön izleme penceresi gösterilmiştir.

Şekil 12.1. FormPrint Sınıfı İçin Tasarlanan Form



Şekil 12.2. Baskı Ön İzleme Penceresi

Bu formdaki dosya yazma ve okuma için yazılması gereken kodlar daha önceki bölümde verildiği için, sadece Yazdır butonu için gerekli kodlar aşağıda verilmiştir.

```
Imports Microsoft.VisualBasic.PowerPacks.Printing
Private Sub btn_yazdir_Click(sender As Object, e As EventArgs) Handles
btn_yazdir.Click
    Dim yazici As PrintForm = New PrintForm(Me)
    yazici.PrinterSettings.DefaultPageSettings.Landscape = True
    yazici.PrintAction = Printing.PrintAction.PrintToPreview
    yazici.Print()
End Sub
```

Yazıcının özelliklerine ulaşmak için PrinterSettings özelliği kullanılır.

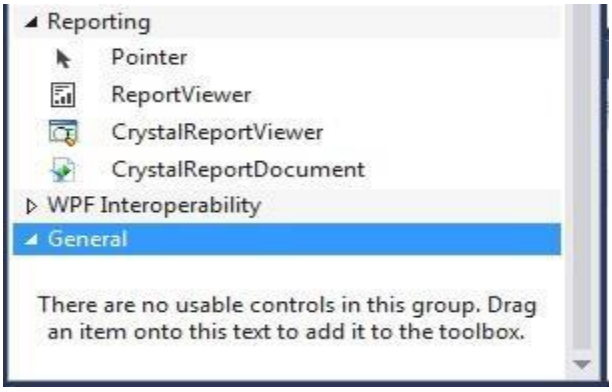
Bir Dosyayı Kullanarak Rapor Hazırlama

Bir dosya denilince ilk bakışta akla sadece dosyalama işlemleri kısmında bahsedilen ve üzerinde çalışmalar yapılan txt uzantılı dosyalar gelebilir. Ama burada dosya kelimesinin içerdiği anlam daha geniştir. Buradaki dosya kelimesi, bir txt dosya olabileceği gibi, bir excel dosyası veya bir veritabanı dosyası da olabilir. Bu tip dosyalar içerisindeki verileri raporlayabilmek için SAP firmasına ait olan CrystalReports ürünü yaygın bir şekilde kullanılmaktadır. Bu ürünü kullanabilmek için Visual Studio .NET'in Professional versiyonunu kullanınız. Yalnız, bu ürün Visual Studio .NET Professional ile birlikte gelmemektedir. Yani, bilgisayarınızda Visual Studio .NET Professional kurulu olması CrystalReport

kullanabileceğiniz anlamına gelmez. Bu ürünün ayrıca indirilerek bilgisayara kurulması gerekir.

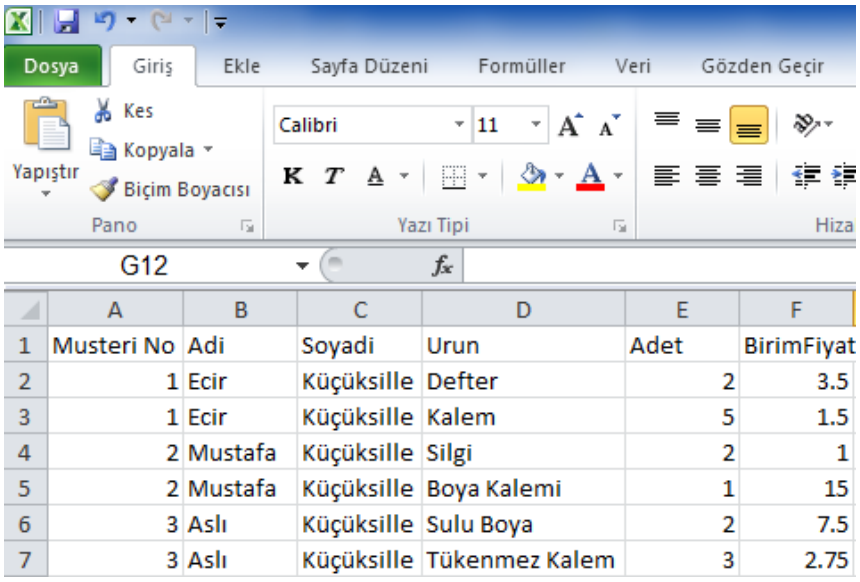
Bir dosya veya veritabanından rapor oluşturmak için SAP firmasının CrystalReport ürünü kullanılabilir.

CrystalReport ürününü indirebilmek için öncelikle http://www.sap.com/solution/sme/software/analytics/crystal-visual-studio/implement/licensing.html sayfasına giderek *free download* linkine tıklanması gerekir. Bu işlemden sonra, sayfadaki ilgili yönlendirmeler takip edilerek SAP Crystal Reports, developer version for Microsoft Visual Studio ürünü bilgisayara indirilebilir. İndirme işlemi gerçekleştirdikten sonra, dosya üzerinde çift tıklanarak ürün bilgisayara kurulmalıdır. Ürün kurulduktan sonra, Visual Studio .N ET ürünü içerisine kendisi entegre olacaktır. Yeni bir VB. NET projesi oluşturulduğunda veya var olan bir proje açıldığında toolbox'da CrystalReport nesneleri Şekil 12.3'te olduğu gibi görülebilir.



Şekil 12.3. ToolBox Penceresi

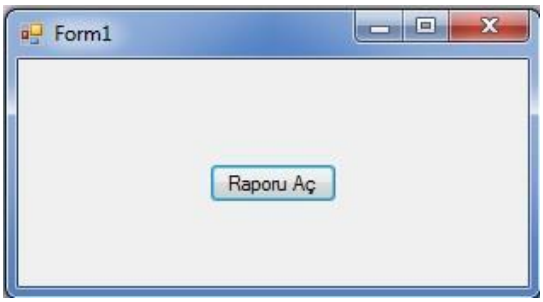
Bu kadar bilgiden sonra yapılacak uygulamalar ile CrystalReport ürününün kullanımı hakkında bilgi verilmeye çalışılacaktır. Yapılacak uygulamalarda bu kitabı kullanan okuyucuların veri tabanı bilgisi olmadığı varsayılmıştır. Bu nedenle veriler bir excel dosyasından okunmaktadır. Kullanılan excel dosyası Şekil 12.4'te verilmiştir. Bu dosya *ornek.xls* olarak kaydedilmiştir.



	A	B	C	D	E	F
1	Musteri No	Adi	Soyadi	Urun	Adet	BirimFiyat
2	1	Ecir	Küçüksille	Defter	2	3.5
3	1	Ecir	Küçüksille	Kalem	5	1.5
4	2	Mustafa	Küçüksille	Silgi	2	1
5	2	Mustafa	Küçüksille	Boya Kalemi	1	15
6	3	Aslı	Küçüksille	Sulu Boya	2	7.5
7	3	Aslı	Küçüksille	Tükenmez Kalem	3	2.75

Şekil 12.4. Örnek Excel Dosyası

Öncelikle VB.Net'de bir Windows Forms Applications Projesi oluşturulur. Bundan sonra, projeye *Project/Add Windows Form* menü seçeneği yardımı ile yeni bir form eklenir. Ardından ilk forma yani Form1'e bir buton nesnesi eklenir. Form1'in bütün bu işlemler yapıldıktan sonraki şekli Şekil 12.5'te verilmiştir.



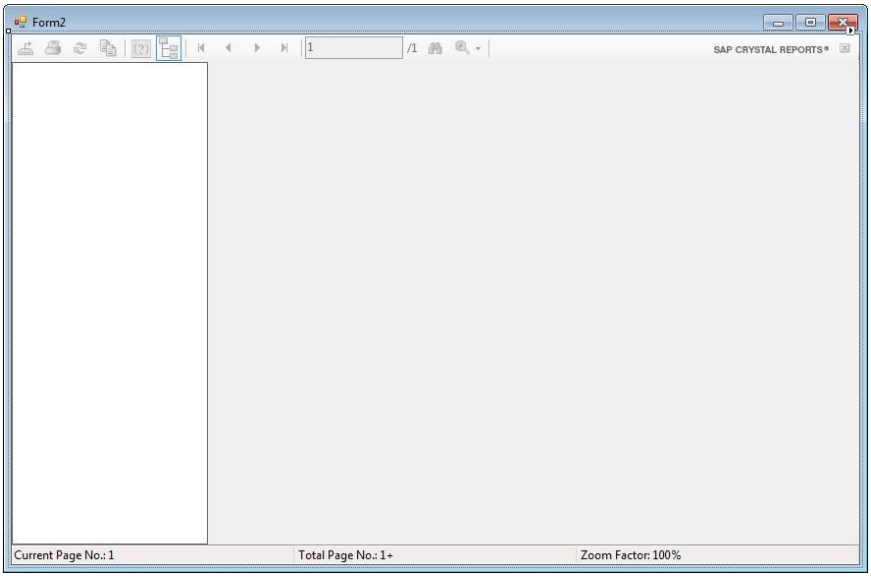
Şekil 12.5. Form1 Ekran Görünümü

Buradaki Raporu Aç butonunun görevi, tasarlanacak olan formun ilişkilendirildiği Form2 formunu açmaktır. Bu sebeple, Raporu aç butonun Click olayına aşağıdaki kod parçası yazılır.

```
Private Sub btn_raporac_Click(sender As Object, e As EventArgs) Handles
btn_raporac.Click
    Dim form2 As New Form2()
    form2.ShowDialog()
End Sub
```

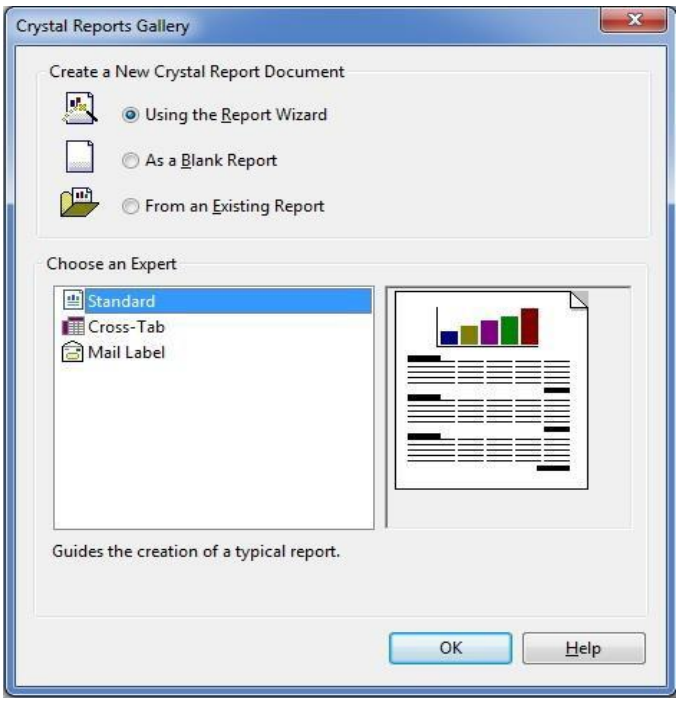
Daha sonra Form2 isimli forma geçilir ve form üzerine CrystalReportViewer nesnesi eklenir. Bu nesne eklendikten sonra Form2 formunun şekli Şekil 12.6'da verilmiştir.

Crystal Report ile hazırlanan raporu görmek için CrystalReportViewer nesnesi kullanılır.



Şekil 12.6. Form2 Ekran Görünümü

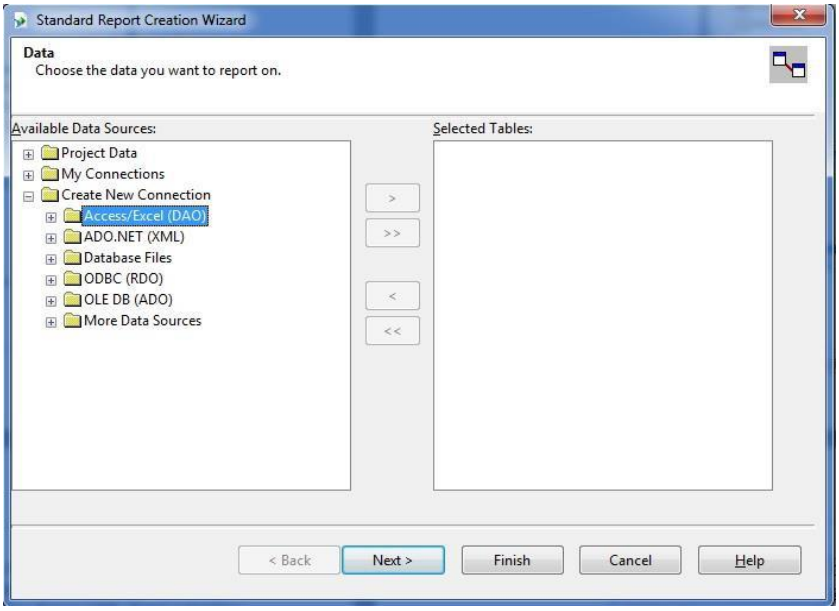
Bundan sonra yapılacak iş, projeye bir CrytsalReports formu eklemektir. Bu işlem *Projects/Add New Item/Reporting/Crystal Reports* menü seçeneği ile yapılır. Bu işlem yapıldığında Şekil 12.7'deki pencere ile karşılaşılır.



Şekil 12.7. Crystal Report Hazırlama Penceresi

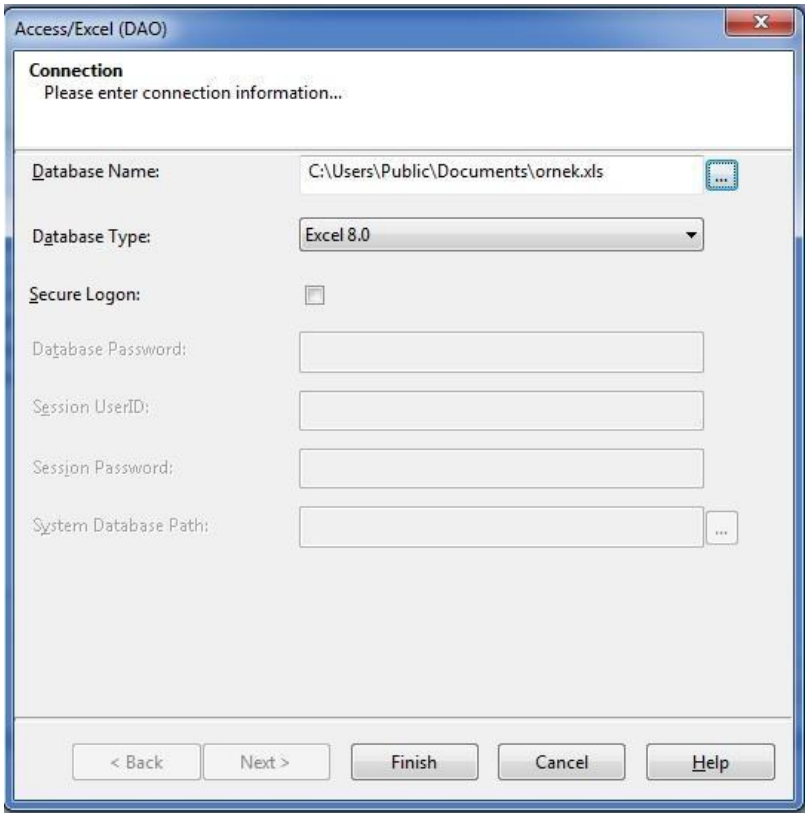
Burada üç seçenek sunulur. Eğer Using the Report Wizard Seçeneği seçilirse bir sihirbaz yardımı ile rapor hazırlanır, As a Blank Report Seçeneği seçilirse rapor dizayn penceresi açılır ve rapor geliştirici tarafından dizayn edilir, From a Existing Report seçeneği seçilirse daha önce hazırlanmış raporlar kullanılarak yeni bir rapor hazırlanabilir. Choose an Expert seçeneğindeki seçeneklerle ise raporun nasıl görüneceği belirlenebilir.

CrystalReport'u tamamı ile anlatmak çok uzun süreceğinden burada sadece ilk seçenekten yani Using the Report Wizard seçeneğinden bahsedilecektir. Using the Report Wizard seçilerek *OK* tuşuna basıldığında Şekil 12.8'de görülen pencere ile karşılaşılır.

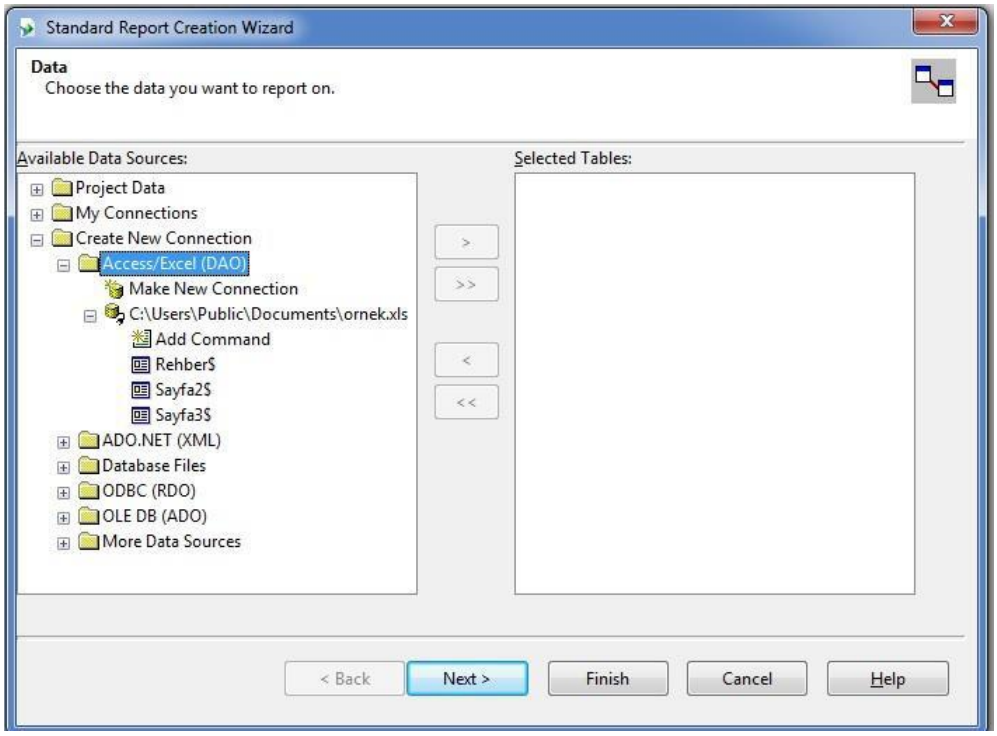


Şekil 12.8. Crystal Report Veri Seti Belirleme Penceresi

Bu pencerede raporda gösterilecek olan verilerin hangi kaynaktan alınacağı belirlenir. Burada daha önce hazırlanmış ve ornek.xls olarak kaydedilmiş olan excel dosyasındaki veriler veri seti olarak kullanılacaktır. Bu nedenle Create New Connection Seçeneği altındaki Access/Excel (DAO) seçeneği yanındaki + işaretine basılır. Basıldığında ekrana gelen penceredeki *Database Type* etiketi yanındaki combobox içerisinden Excel 8.0 seçeneği seçilir. Bundan sonra sırada üzerinde çalışılacak olan excel dosyasının seçilmesi vardır. Bu işlem de Database Name etiketi yanındaki textbox'ın hemen yanında bulunan butona basılarak gerçekleştirilir. Bu butona basıldığında bir dosya seçme penceresi açılır. Bu pencere yardımı ile üzerinde çalışılacak excel dosyası belirlenir. Bütün bu işlemler sonucunda pencerenin son hâli Şekil 12.9'da verilmiştir. Şekil 12.10'da ise Finish butonuna basıldıktan sonra Veri seti belirleme penceresinin son hâli verilmiştir.

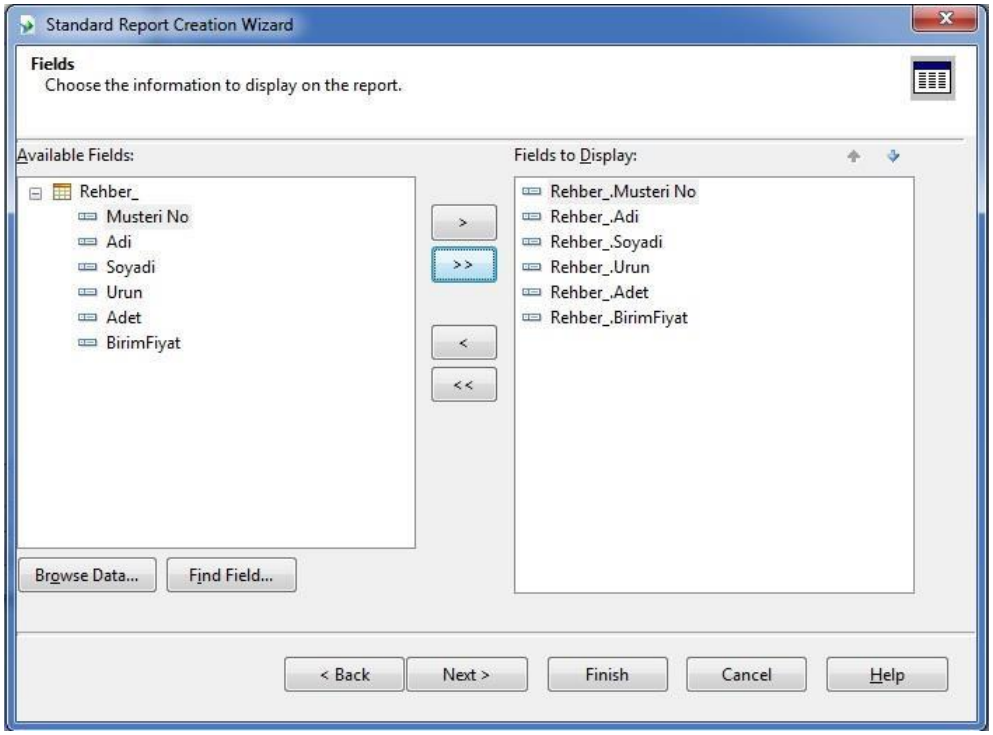


Şekil 12.9. Çalışılacak Dosyayı Belirleme Penceresi



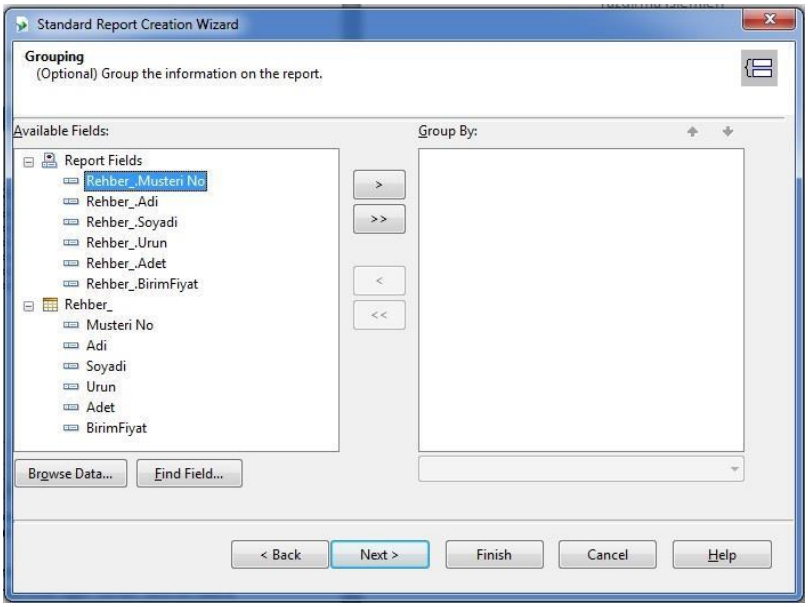
Şekil 12.10. Veri Seti Belirleme Penceresi Son Hâli

Şekil 2.10'daki pencerede de görüldüğü gibi seçilen excel dosyası içerisindeki tüm çalışma sayfaları listelenmiştir. Bu sayfalardan hangisinde çalışılacak veri varsa o sayfa seçilerek Selected Tables listesine aktarılır ve Next butonuna basılır. Bu işlem sonunda Şekil 12.11'de verilen ve Raporda görünmesi istenen alanların seçileceği pencereye geçilmiş olur. Bu örnekte tüm alanlar görülmek istendiğinden tüm alanlar Fields to Display listesine aktarılır.



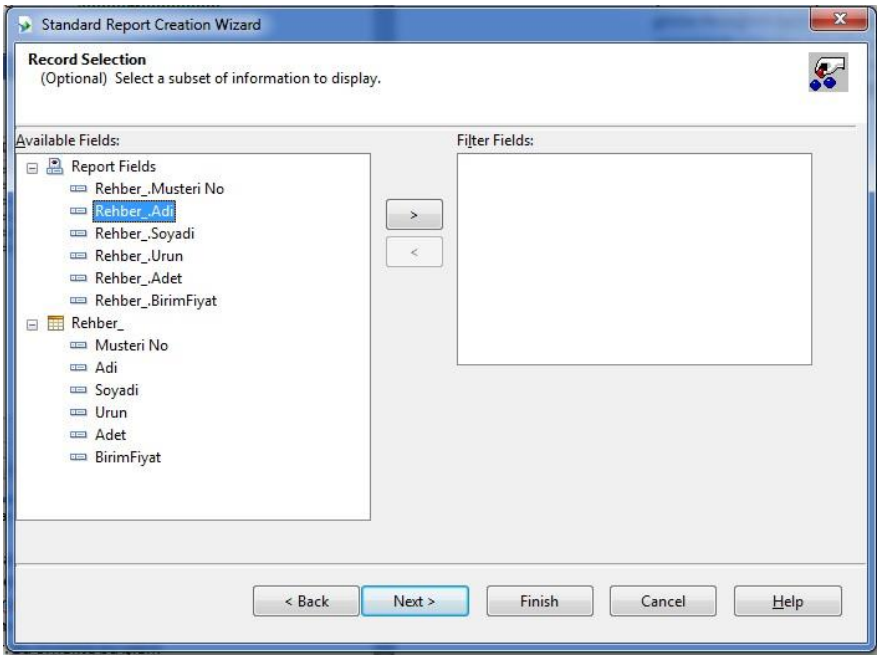
Şekil 12.11. Raporda Görünecek Alanları Belirleme Penceresi

Şekil 12.11'de görülen pencerede Next butonuna basıldığında Şekil 12.12.'de gösterilen gruplama penceresine ulaşılır. Bu pencerede eğer veriler belli bir alana göre gruplanmak isteniyorsa bu alan seçilerek Group By listesine aktarılır. Örnek olarak, her bir müşterinin yaptığı alış veriş toplamı ayrı ayrı gösterilmek isteniyorsa Musteri No alanı Group By listesine eklenmelidir. Bu örnekte bu işlem yapılmayacak ve Next butonuna basılarak geçilecektir. Bir sonraki örnekte ise bu işlem gerçekleştirilerek elde edilecek raporun ekran görüntüsü verilecektir.



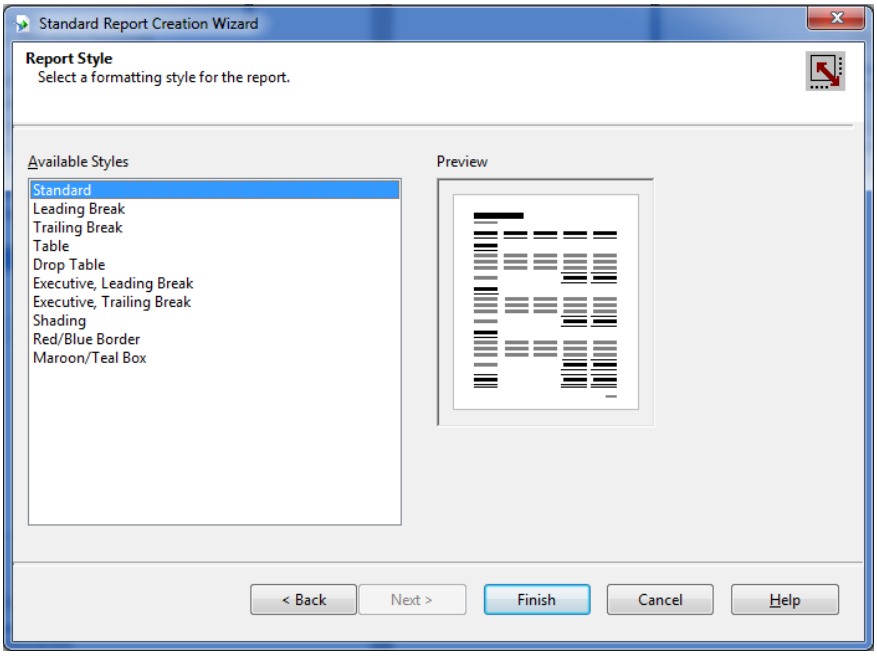
Şekil 12.12. Veri Gruplama Penceresi

Şekil 12.13'te verilen pencerede Next butonuna basıldığında hangi verilerin gösterileceğinin belirlenebildiği Şekil 12.13.'te verilen pencere ile karşılaşılır. Bu pencerede istenilen koşullar yazılarak bazı veriler raporda gösterilmeyebilir. Eğer hiçbir işlem yapılmazsa bu tüm verilerin gösterileceği anlamına gelir.



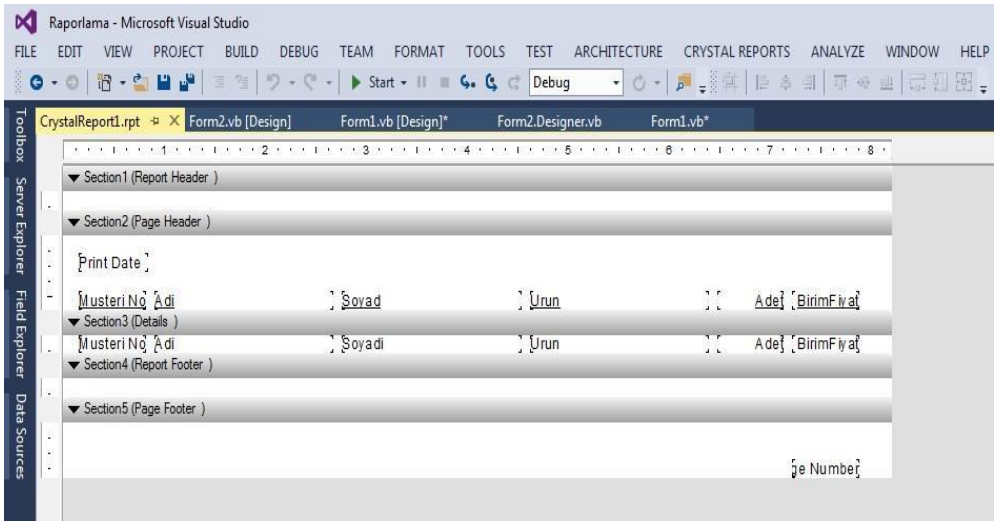
Şekil 12.13. Raporda Görünecek Verileri Belirleme Penceresi

Bu örnekte tüm veriler görülmek istendiğinden Next butonuna basılarak geçilecektir. Bir sonraki pencere ise Şekil 12.14'te gösterilen rapor stilini belirleme penceresidir. Burada birçok rapor stili verilmiştir.



Şekil 12.14. Rapor Stili Belirleme Penceresi

Bu pencerede yapılan örnek için Standard stili seçilerek Finish butonuna basılır. Böylece wizard yardımı ile istenen rapor dizayn edilerek hazırlanır. Hazırlanan tasarım Şekil 12.15'te verilmiştir.



Şekil 12.15. Rapor Dizayn Penceresi

Şekil 12.15.'te verilen dizayn penceresindeki bölümlerin işlevleri hakkında aşağıda kısaca bilgi verilmiştir.

Report Header: Raporun ilk sayfasında, sayfanın üstünde yazılacak olan başlığın belirlendiği bölümdür.

Page Header: Her sayfanın üst kısmında yazılacak olan başlık veya başlıkların belirlendiği bölümdür.

Rapor Dizayn Ekranında veri setindeki kayıtlar Details bölümü yardımı ile görüntülenir.

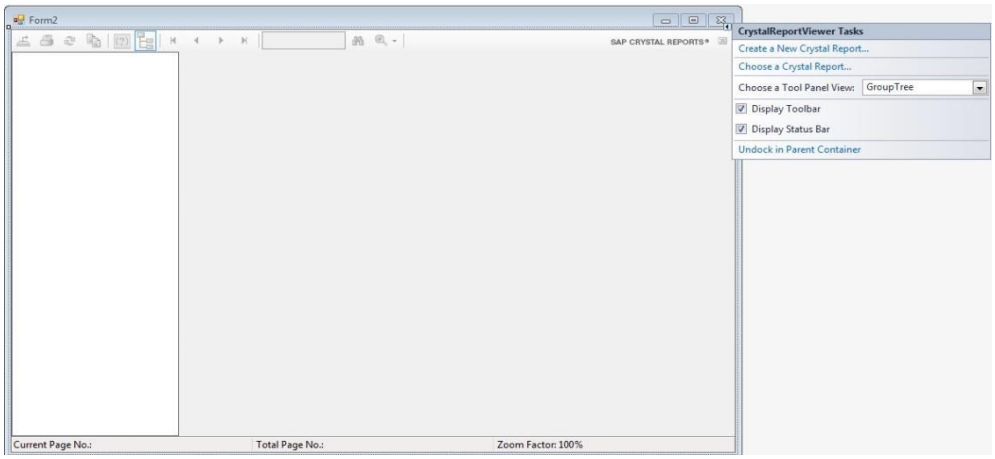
Details: Raporda veri setinden çekilen kayıtlardan görünmesi istenen alanların belirlendiği bölümdür.

Report Footer: Raporun son sayfasında, sayfanın altında yazılacak olan başlığın belirlendiği bölümdür.

Page Footer: Her sayfanın alt kısmında yazılacak olan başlık veya başlıkların belirlendiği bölümdür.

Yukarıda anlatılan bölümlere dizayn penceresinde bulunan ToolBox penceresi yardımı ile yazı, çizgi ve dörtgen eklenebilir. Field Explorer penceresi yardımı ile ise Veri Seti Alanları, Hesaplanmış alan gibi yeni alanlar rapora eklenebilmektedir.

Program çalıştırıldığında raporun Form2 nesnesi üzerinde bulunan CrystalReportViewer1 nesnesi üzerinde görüntülenebilmesi için dizayn edilen CrystalReport1.rpt isimli raporun CrystalReportViewer1 nesnesi ile ilişkilendirilmesi gerekir. Bu işlemin nasıl gerçekleştirileceği Şekil 12.16'da gösterilmiştir.

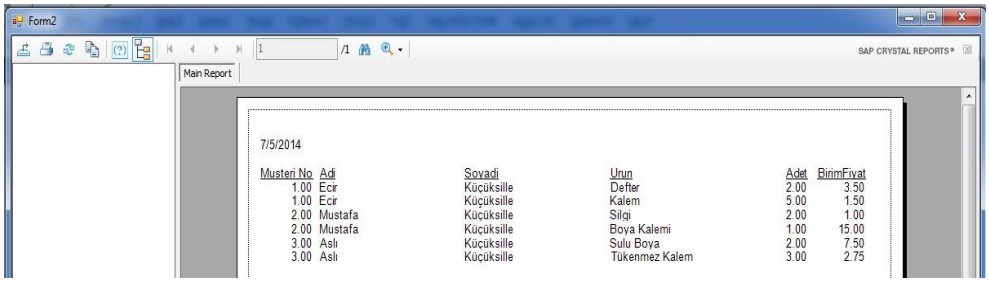


Şekil 12.16. CrystalReport1.rpt Dosyası ile CrystalReportViewer1 Nesnesinin İlişkilendirilmesi

Şekil 12.16'da verilen şeklideki CrystalRpeorViewerTasks penceresinden Choose a Crystal Report seçeneği seçilir. Bu seçenek seçildiğinde Şekil 12.17'deki pencere ekrana gelir. Bu pencereden hazırlanan CrystalReport1.rpt isimli dosya seçilir.

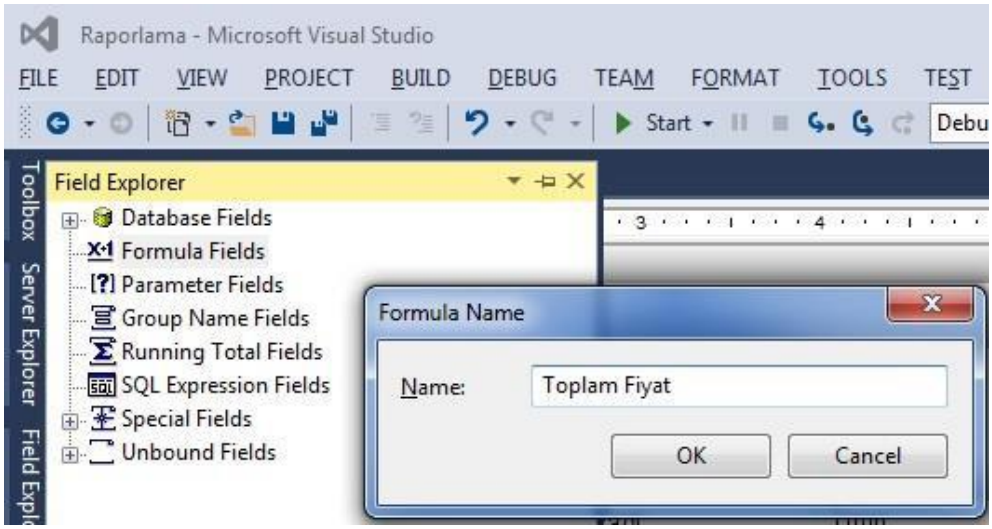
Şekil 12.17. CrystalReport1.rpt Dosyasının Seçimi

Bütün bu işlemlerden sonra program çalıştırılıp Raporu Aç butonuna tıklandığında alınan raporun ekran görüntüsü Şekil 12.18'de verilmiştir.



Şekil 12.18. Hazırlanan Raporun Ekran Görüntüsü

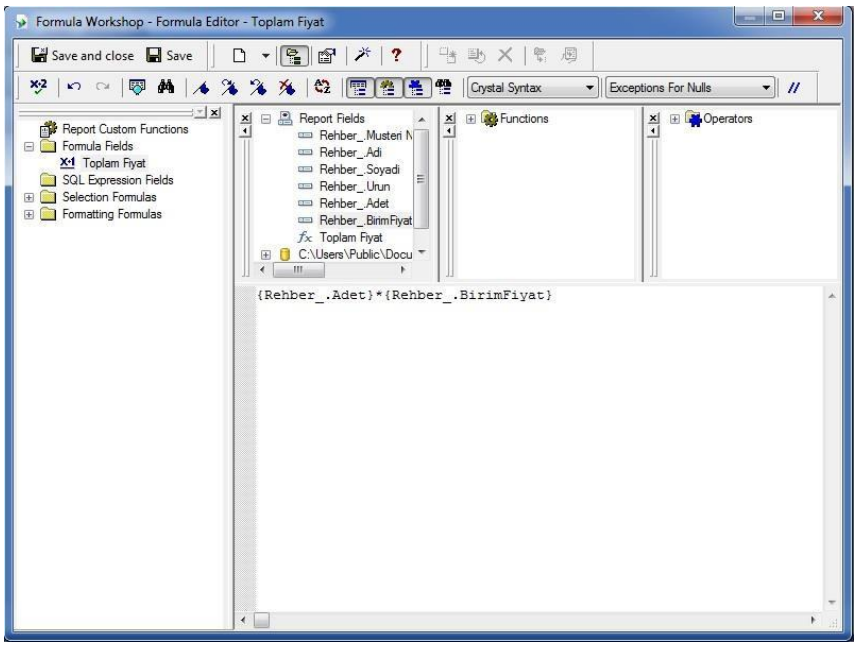
Şimdi hazırlana rapor örneği yeni belirlenecek ihtiyaçlara göre biraz daha geliştirilecektir. Örneğin her kaydın sonuna toplam fiyat isminde bir alan eklensin. Bu alan veri setimizde yoktur. Bu yüzden rapor oluşturulurken hesaplanacak ve kullanıcıya gösterilecektir. Bu şekilde veri setinde olmayan ve daha sonra hesaplanarak elde edilen alanlara Formula Fields adı verilir. Bu nedenle rapora Formula Fields hazırlanarak eklenmelidir. Bu işlem için yandaki Field Explorer Penceresindeki Formula Fields üzerinde sağ tuşa tıklanarak New seçeneği seçilir. Bu durumda ekrana gelen Formula Name penceresi ile hazırlanan alana bir isim verilir. İsim olarak Toplam Fiyat belirlenmiş ve bu pencereye ait ekran görüntüsü Şekil 12.19'da verilmiştir.



Şekil 12.19. Formula Fileds Penceresi

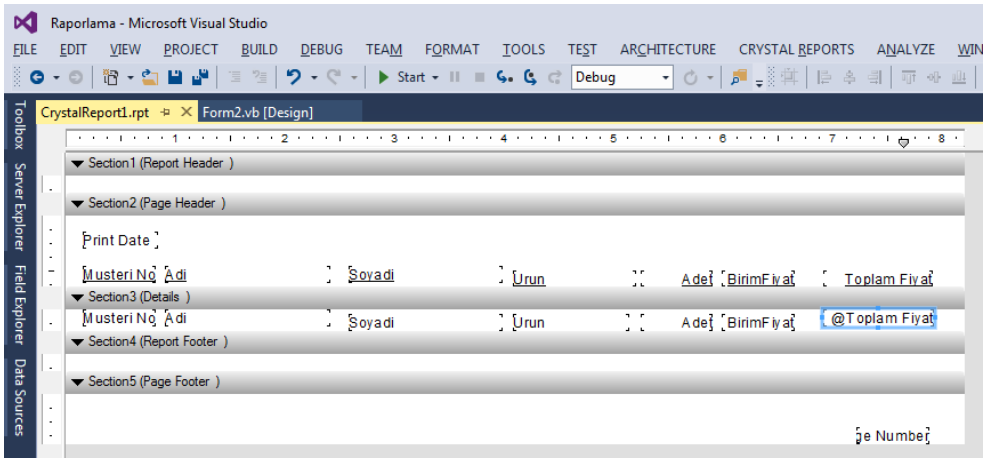
Veri Setinde olmayan hesaplanmış alanlar Formula Fields seçeneği ile belirlenir.

Bu pencerede isim belirlenip OK butonuna basıldığında Formula Editor penceresi ile karşılaşılır. Bu pencere, hesaplamanın hangi alanlar üzerinde nasıl yapılacağının belirlendiği penceredir. Bu pencerede Report Fields Seçeneğindeki alanlar üzerinde çift tıklanarak ilgili alanlar editör üzerine aktarılabilmektedir. Bu örnek için Adet ve BirimFiyat alanları üzerine çift tıklanarak editöre eklenir, aralarına da * işareti konulur ve Save and Close butonuna tıklanır. Bu duruma ait ekran görüntüsü Şekil 12.20.'de verilmiştir.



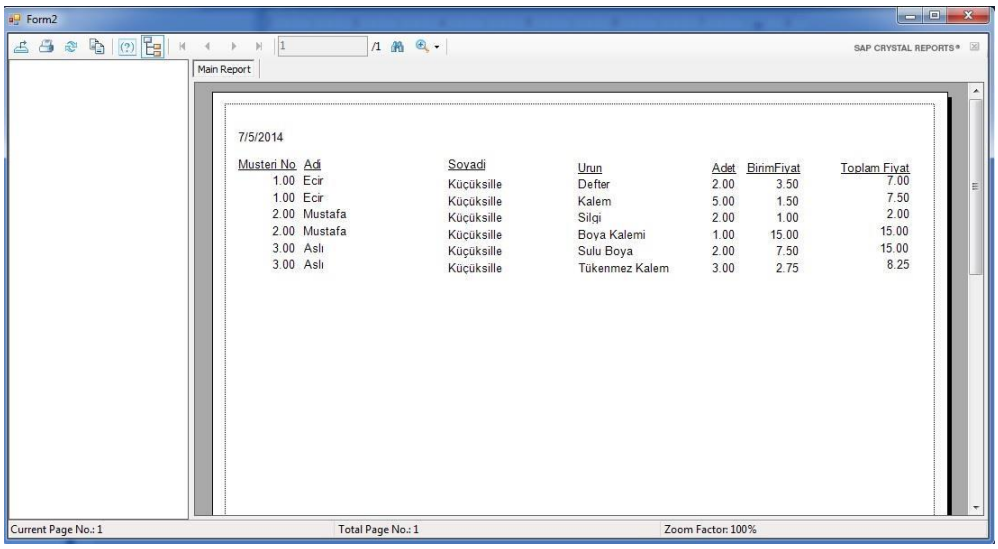
Şekil 12.20. Formula Editor Penceresi

Yapılan işlemler sonucunda Toplam Fiyat isimli Formula Fields oluşturulmuştur. Bu alan Field Explorer penceresinde Formula Fields seçeneği altında listelenir. Şimdi yapılması gereken, bu alanı fare ile tutup taşıyarak Details bölümü altına yerleştirmektir. Bu işlem sonucunda CrystalReport1.rpt isimli dosyanın ekran görüntüsü şekil 12.21'de verilmiştir.



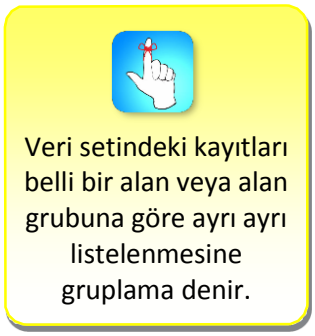
Şekil 12.21. CrystalReport1.rpt Dosyası

Bu işlem sonucunda program çalıştırıldığında alınan rapora ait ekran görüntüsü Şekil 12.22'de verilmiştir.

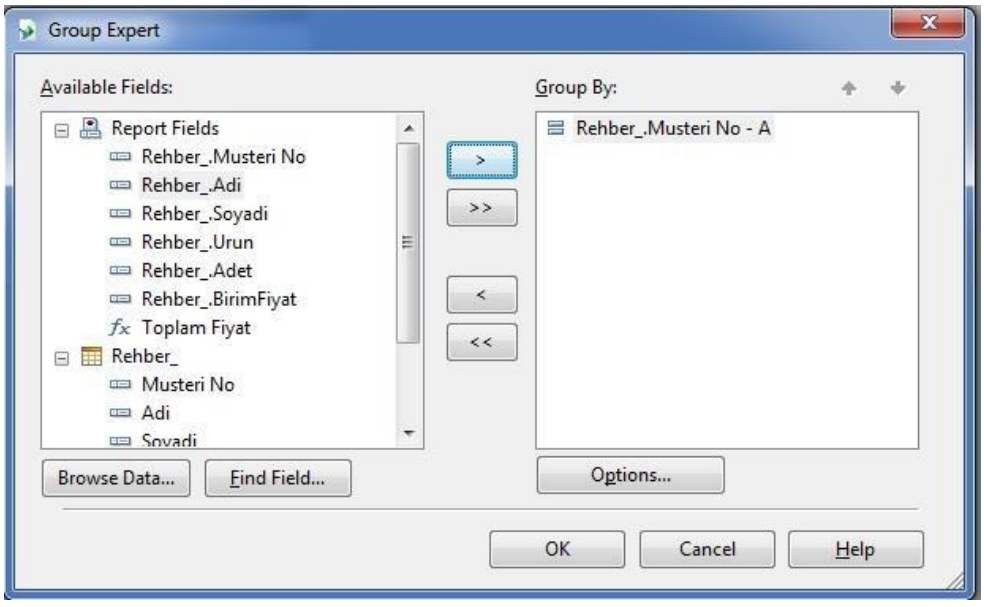


Şekil 12.22. Hazırlanan Raporun Ekran Görüntüsü

Bu bölümde son olarak yapılacak örnek gruplama ile ilgili olacaktır. Yapılacak örnekte amaç, her bir müşterinin ayrı ayrı yaptıkları toplam alışverişleri ve bunların tutarlarını görebilmektir. Bu amaçla öncelikle, gruplamanın hangi alan veya alanlara göre yapılacağının belirlenmesi gerekir. Bu işlem için, Field Explorer penceresi içerisindeki Group Name Fields seçeneği kullanılır. Bu seçenek üstünde sağ tuşa tıklandığında çıkan menüden Gropu Expert seçeneği seçilir. Bu seçeneğin seçilmesi ile Group Expert penceresi ekrana gelir. Bu pencere yardımı ile gruplanacak alan veya alanlar belirlenerek Group By listesine aktarılır. Bu örnek için Musteri No alanı seçilerek Group By listesine aktarılmıştır. Bu işlem sonucunda pencerenin aldığı görünüm Şekil 12.23.'te verilmiştir.

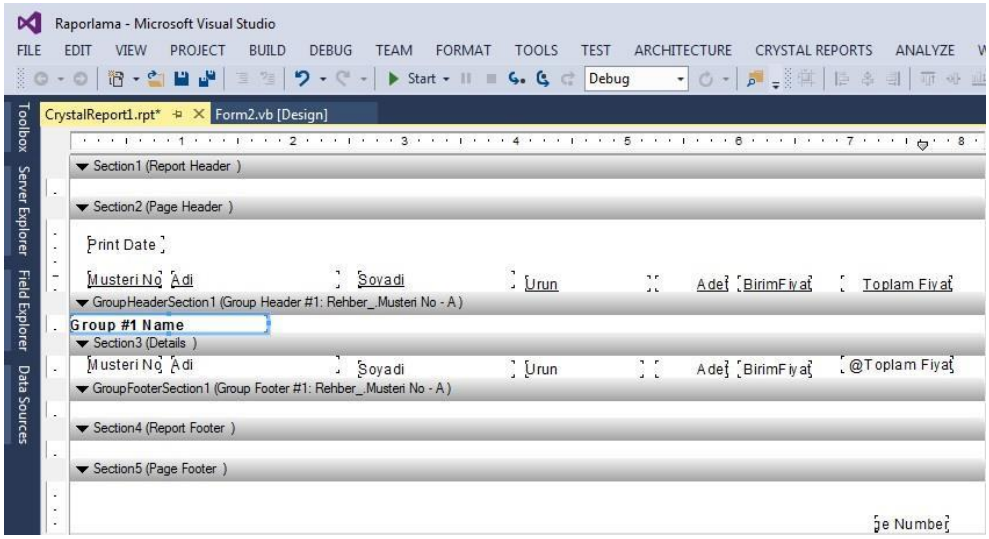


Veri setindeki kayıtları belli bir alan veya alan grubuna göre ayrı ayrı listelenmesine gruplama denir.



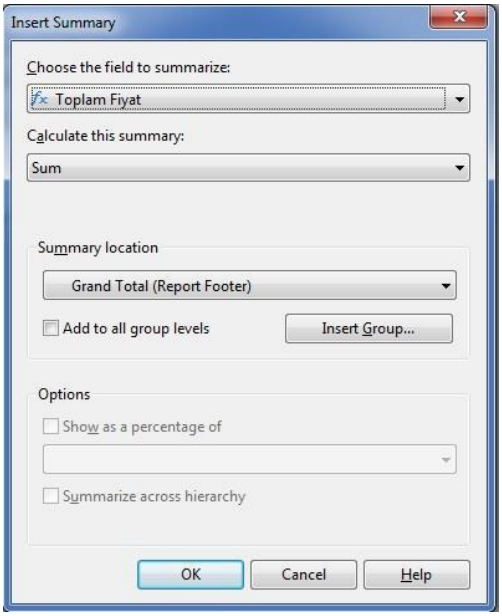
Şekil 12.23. Group Expert Penceresi

Bu pencere OK butonuna basılarak kapatıldığında rapor dizayn penceresinin aldığı görünüm Şekil 12.24.'te verilmiştir.

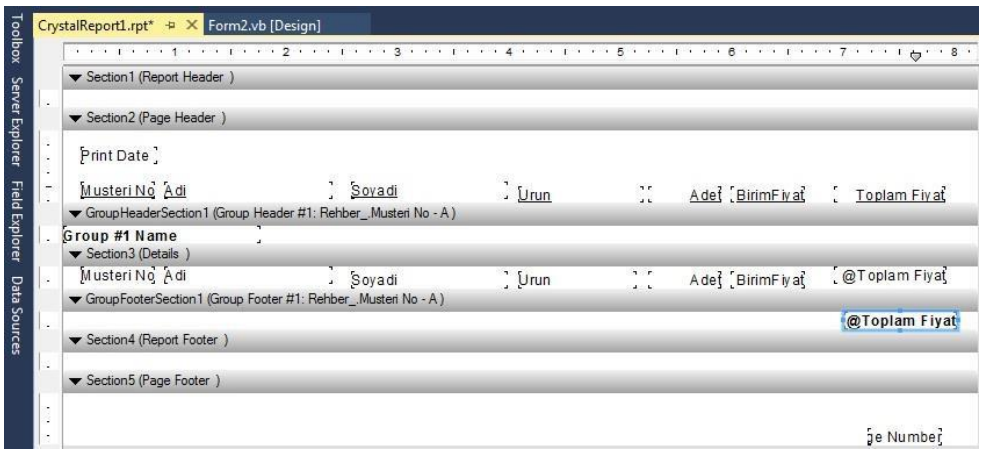


Şekil 12.24. Rapor Dizayn Penceresi

Son yapılan işlem ile aslında gruplama işlemi tamamlanmıştır. Fakat her grupta bulunan müşterilerin toplam kaç liralık alışveriş yaptığı bilgisini görebilmek için son bir işlemin daha yapılması gerekmektedir. Bu işlem için rapor dizayn penceresindeki GroupFooterSection1 bölümünde sağ tuşa tıklanarak Insert/Summary seçeneği seçilir ve ekrana Insert Summary penceresi gelir. Bu pencerede Şekil 12.25.'teki ayarlar yapılarak OK butonuna basılır ve rapor hazır hâle gelir. Rapor dizayn penceresinin son hâli de Şekil 12.26.'da verilmiştir.

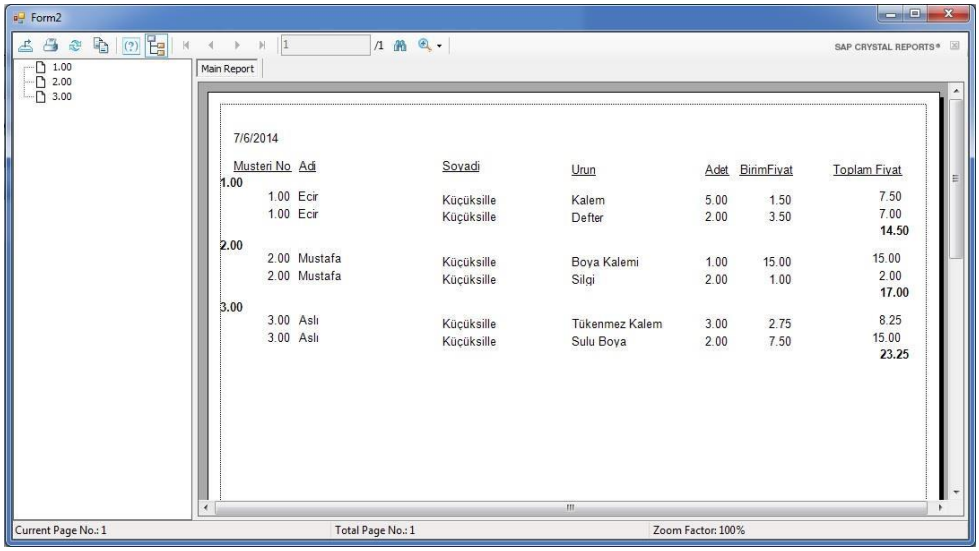


Şekil 12.25. Insert Summary Penceresi



Şekil 12.26. Rapor Dizayn Penceresi

Tüm bu işlemler sonucunda program çalıştırılıp Raporu Aç butonuna tıklandığında hazırlanan raporun görüntüsü Şekil 12.27'de verilmiştir.



Musteri No	Adi	Soyadi	Urun	Adet	BirimFiyat	Toplam Fiyat
1.00						
	1.00 Ecir	Küçüksille	Kalem	5.00	1.50	7.50
	1.00 Ecir	Küçüksille	Defter	2.00	3.50	7.00
						14.50
2.00						
	2.00 Mustafa	Küçüksille	Boya Kalemi	1.00	15.00	15.00
	2.00 Mustafa	Küçüksille	Silgi	2.00	1.00	2.00
						17.00
3.00						
	3.00 Aslı	Küçüksille	Tükenmez Kalem	3.00	2.75	8.25
	3.00 Aslı	Küçüksille	Sulu Boya	2.00	7.50	15.00
						23.25

Şekil 12.26. Hazırlanan Rapor Görüntüsü

Hazırlanan rapor ekranda görüntülendikten sonra yazıcıdan çıktı alınabileceği gibi, pdf, doc, xlsx, csv ve xml gibi dosya formatlarına da dönüştürülebilmektedir.

Ayrıca rapor dizayn penceresindeki Field Explorer penceresindeki Special Fields seçeneği kullanılarak rapora tarih, zaman, grup numarası, Rapor oluşturulma zamanı, toplam sayfa sayısı gibi alanlar da eklenebilmektedir.

Özet

- Bir formun veya bir veri setindeki bilgilerin kâğıt üzerine aktarılması işlemine yazdırma adı verilir.
- Eğer bir tasaralanan bir formun çalışma zamanındaki görüntüsü yazıcıya aktarılmak isteniyorsa VisualBasic.PowerPacks.Printing namespace'i içerisinde bulunan PrintForm sınıfı kullanılır.
- Eğer bir dosya veya bir tabanındaki bilgiler yazıcıya aktarılmak isteniyorsa SAP firmasının CrystalReport ürünü kullanılabilir.
- CrystalReport ürününü kullanabilmek için Visual Studio .NET Professional ürününün bilgisayarda kurulu olması gerekemektedir.
- CrystalReport ile üretilen raporlar başka dosya formatlarına dönüştürülebilmektedir.

DEĞERLENDİRME SORULARI

1. Bir formu yazıcıya aktarabilmek için kullanılan sınıfın ismi aşağıdakilerden hangisidir?
    a) Print
    b) PowerPacks
    c) PrintForm
    d) FormToPrint
    e) PrintToForm

2. Form yazdırılırken yazıcı özelliklerine hangi özellik ile ulaşılır?
    a) Properties
    b) PrinterProperties
    c) PrintAction
    d) Settings
    e) PrinterSettings

3. Formun hangi kısmı yazıcıya aktarılır?
    a) Görünür Kısmı
    b) Hepsi
    c) Mouse ile İşaretlenen Kısmı
    d) Kaydırma Çubukları Hariç Tümü
    e) Başlık Hariç Tümü

4. CrystalReport ürünü hangi firmaya aittir?
    a) Microsoft
    b) SAP
    c) SUN
    d) HP
    e) IBM

5. Crystal Report hazırlama Penceresinde aşağıdaki seçeneklerden hangisi vardır?
    a) Filling
    b) Template
    c) Wizard
    d) Report
    e) As a Blank Report

6. Rapor dizayn penceresinde hangi bölüm yoktur?
    a) Report Header
    b) Page Header
    c) Page Title
    d) Report Footer
    e) Details

7. Veri setindeki kayıtların belli bir alan veya alan grubuna göre ayrı ayrı listelenmesine ne ad verilir?
    a) Gruplama
    b) Seçme
    c) Eleme
    d) Filtreleme
    e) Ayırma

8. Veri setinde olmayan bir alanın diğer alanlar üzerinde işlem yaparak hesaplanması ile oluşan alana verilen isim aşağıdakilerden hangisidir?
    a) Expression Fields
    b) Formula Fields
    c) Calculated Fields
    d) Math Fields
    e) Exp Fileds

9. Sayfa Numarası, Tarih gibi alanları görmek için kullanılan alan aşağıdakilerden hangisidir?
    a) Custom Fields
    b) Specific Fields
    c) Special Fields
    d) Unbound Fields
    e) Private Fields

10. Crystal Report ile hazırlanan raporlar aşağıdaki formatlardan hangisine çevrilemez?
    a) pdf
    b) xlsx
    c) csv
    d) jpg
    e) doc

Cevap Anahtarı

1.c, 2.e, 3.a, 4.b, 5.e, 6.c, 7.a, 8.b, 9.c, 10.d

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

http://msdn.microsoft.com

GRAFİK UYGULAMALARI

İÇİNDEKİLER

- Grafik Uygulamaları
  - GDI+
  - Bitmap Sınıfı
  - Brush Sınıfı
  - Font Sınıfı
  - Graphics Sınıfı
  - Icon Sınıfı
  - Image Sınıfı
  - Pen Sınıfı

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - GDI+ kavramını açıklayabilecek,
  - Bitmap sınıfını uygulamalarınızda kullanabilecek,
  - Graphics sınıfını kullanarak form üzerine çizimler yapabilecek
  - Yapacağınız çizimlerde Pen, Brush,Font sınıflarını kullanabilecek,
  - Icon ve Image sınıflarını uygulamalarınıza entegre edebilecek,
  - Tüm bu sınıfları kullanarak yeni uygulamalar geliştirebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Doç. Dr.
Ecir Uğur KÜÇÜKSİLLE

ÜNİTE
13

GİRİŞ

Grafikler genellikle uygulamaların kullanıcı ara yüzlerini görsel açıdan zenginleştirmek amacıyla kullanılırlar. Günümüzde geliştirilen sistemler, insan-bilgisayar etkileşimi ve kullanılabilirlik gereksinimlerini de karşılamak üzere geliştirilmektedirler. Kullanıcıların daha iyi bilgilendirilmesi, hataların minimize edilmesi ve kullanılan ögelerin gerçek hayatla eşleştirilmesi gibi noktalar günümüzde önem taşımaktadır.

Visual Basic.NET kendi içerisinde bu amaçlardan bazılarını karşılamak üzere hazır grafik sınıflarını barındırmaktadır. Bu ünitede grafik uygulamaları; *Bitmap, Brush, Font, Graphics, Icon, Image* ve *Pen* sınıfları kullanılarak anlatılacaktır.

GRAFİK UYGULAMALARI

Grafik uygulamaları kısaca, form üzerine geometrik şekilleri ve grafikleri çizdirme işlemleri olarak adlandırılabilir.

GDI+

Çizim ile ilgili sınıflar System. Drawing namespace'i içerisinde bulunur.

GDI+ iki boyutlu vektör grafikler, görüntüleme, tipografi gibi işlemleri gerçekleştirmek için geliştirilmiştir ve Windows XP işletim sisteminin bir parçasıdır. GDI+, GDI (Graphic Device Interface) 'nın yeni özellikler eklenerek ve var olan özellikleri optimize edilerek geliştirilmiş yeni şeklidir.

.N ET Framework'de grafikleri oluşturmak için System.Drawing namespace'i kullanılır. Bu bölümde hem bu namespace içerisinde bulunan hem de bu namespace içerisindeki diğer namespace'lerde bulunan ve çok kullanılan sınıflar örnekler ile beraber açıklanmaya çalışılacaktır.

Bu namespace GDI+'ın temel grafik fonksiyonlarını kullanabilmeyi sağlar. Aşağıda bu namespace içerisinde çok kullanılan sınıflar, sınıfların çok kullanılan özellikleri ve metotları anlatılmaya çalışılacaktır.

Bitmap Sınıfı

GDI+ bitmap, bir grafik şekli ve onun özelliklerine ait pixel verilerinin birleşiminden oluşur. Bir bitmap, pixel veriler tarafından tanımlanmış şekiller ile çalışmak için kullanılan bir sınıftır. Bu sınıf bmp, gif, jpeg, exif, png, tiff dosya türlerini destekler.

Kurucu Metotlar:

Bitmap(resim As Image)

Resim: Üzerinde çalışılacak olan resim dosyanın belirlenmesini sağlayan, Image türündeki parametredir.

Bitmap(dosyayolu As String):

Dosyayolu: Üzerinde çalışılacak olan resim dosyanın belirlenmesini sağlayan, String türündeki parametredir.

Özellikler:

Tablo 13.1. Bitmap Sınıfı Özellikleri

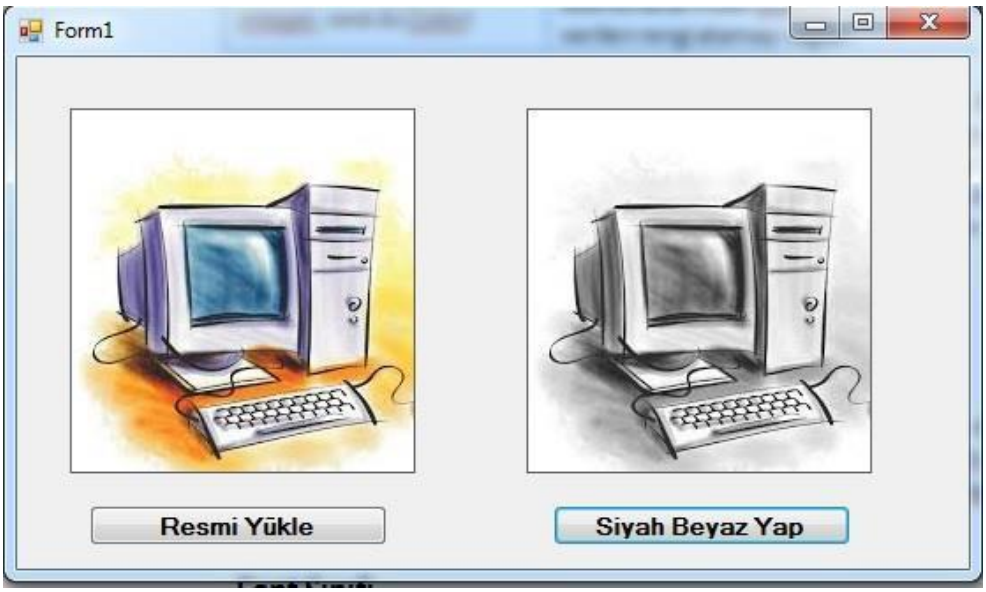
Özellik	İşlev
Height	Üzerinde çalışılan resmin pixel olarak yüksekliğini öğrenmeyi sağlar.
Size	Üzerinde çalışılan resmin pixel olarak yüksekliğini ve genişliğini öğrenmeyi sağlar.
Width	Üzerinde çalışılan resmin pixel olarak genişliğini öğrenmeyi sağlar.

Metotlar:

Tablo 13.2. Bitmap Sınıfı Metotları

Özellik	İşlev
Clone() As Object	Bitmap nesnesinin bir örneğinin oluşturulmasını sağlar.
Dispose()	Bitmap türündeki nesnenin, kullandığı tüm sistem kaynaklarını serbest bırakmasını sağlar.
GetPixel(x As Integer, y As Integer) As Color	Koordinatları birinci ve ikinci parametrede verilen pixel'in rengini öğrenmeyi sağlar.
RotateFlip(dondurme_orani As RotateFlipType)	Bitmap nesnesini, belirli bir açı ve yönde döndürmeyi sağlar.
Save(dosya_ismi As String)	Resmi parametre olarak verilen dosyaya kaydetmeyi sağlar.
SetPixel(x As Integer, y As Integer, renk As Color)	Parametre olarak verilen x ve y koordinatlarındaki pixel'e, yine parametre olarak verilen rengi atamayı sağlar.

Bitmap sınıfı için bir örnek yapılarak bu başlık sonlandırılacaktır. Bu örnekte form üzerine ikişer adet pb1 ve pb2 isimli PictureBox, btnyukle ve btnsiyahbeyaz isimli Button nesnesi yerleştirilmiştir. Ayrıca, pb1 ve pb2 nesnelerinin BackgroundImageLayout özelliği Strech, BorderStyle özelliği de FixedSingle yapılmıştır. Amaç, öncelikle orijinal resmi pb1 nesnesinde göstermek daha sonra da resmi önce siyah beyaza çevirip daha sonra pb2 nesnesinde göstermektir. Bu amaçla tasarlanan formun çalıştıktan sonra oluşan ekran görüntüsü Şekil 13.1.'de verilmiştir.



Şekil 13.1. Bitmap Sınıfı İçin Tasarlanan Form

```
Imports System.Drawing
Public Class Form1
   Private Sub btnyukle_Click(sender As Object, e As EventArgs) Handles
btnyukle.Click
      Dim resim As New Bitmap("c:\Ornek\computer.jpg")
      pb1.Image = resim

   End Sub
```

```
Private Sub btnsiyahbeyaz_Click(sender As Object, e As EventArgs) Handles
btnsiyahbeyaz.Click
      Dim orijinal_resim As New Bitmap("c:\Ornek\computer.jpg")
      Dim yeni_resim As New Bitmap(orijinal_resim.Width,
orijinal_resim.Height)
      Dim orijinal_renk, son_renk As Color
      Dim siyahbeyaz As Integer
      For i = 0 To orijinal_resim.Width - 1
         For j = 0 To orijinal_resim.Height - 1
            orijinal_renk = orijinal_resim.GetPixel(i, j)
            siyahbeyaz = orijinal_renk.R * 0.3 + orijinal_renk.G * 0.59 +
orijinal_renk.B * 0.11
            son_renk = Color.FromArgb(siyahbeyaz, siyahbeyaz, siyahbeyaz)
            yeni_resim.SetPixel(i, j, son_renk)
         Next
      Next
      pb2.Image = yeni_resim
   End Sub
End Class
```

Bir şeklin içini doldurmak için Brush sınıfı kullanılır.

Brush Sınıfı

Dikdörtgen, elips, poligon ve yollar gibi grafiksel şekillerin içini doldurmak için kullanılan nesnedir. Brush sınıfı abstract bir sınıf olduğu için, brush sınıfından bir nesne oluşturmak için SolidBrush, TextureBrush, LinearGradientBrush gibi Brush sınıfından türetilen sınıflar kullanılabilir.

Bu sınıfa ait kullanım, Graphics sınıfı anlatıldıktan sonra yapılacak örnek içerisinde gösterilecektir.

Font Sınıfı

Bir metnin, font tipi, büyüklüğü ve stil özelliklerini tanımlamayı sağlayan bir sınıftır.

Kurucu Metotlar:

Font(font_ismi As String, boyut As Single): Parametre olarak verilen isimde ve büyüklükte bir font nesnesi oluşturmayı sağlar.

Font(font_ismi As String, boyut As Single, stil As FontStyle): Parametre olarak verilen isimde, büyüklükte ve stilde bir font nesnesi oluşturmayı sağlar.

Özellikler:

Tablo 13.3. Font Sınıfı Özellikleri

Özellik	İşlev
Bold	Font nesnesinin, koyu yazma özelliğine sahip olup olmadığını öğrenmeyi sağlar.
Italic	Font nesnesinin, italik yazma özelliğine sahip olup olmadığını öğrenmeyi sağlar.
Name	Font nesnesinin sahip olduğu font ismini öğrenmeyi sağlar.
Size	Font nesnesinin font büyüklüğünü öğrenmeyi sağlar.
Style	Font nesnesinin sahip olduğu stili öğrenmeyi sağlar.
Underline	Font nesnesinin, altı çizili yazma özelliğine sahip olup olmadığını öğrenmeyi sağlar.

Bu sınıfa ait kullanım, Graphics sınıfı anlatıldıktan sonra yapılacak örnek içerisinde gösterilecektir.

Graphics Sınıfı

Bir GDI+ çizim yüzeyini içerir. Çizimlerin birçoğu Graphics sınıfından oluşturulan nesnenin metodları yardımı ile yapılır.

Çizimlerin çoğu Graphics sınıfından oluşturulan nesnelerin yardımı ile yapılır.

Özellikler:

Tablo 13.4. Graphics Sınıfı Özellikleri

Özellik	İşlev
Clip	Graphics nesnesi ile çizim yapılacak bölgeyi belirlemeyi veya öğrenmeyi sağlar.
IsClipEmpty	Graphics nesnesinin bir Clip değerine sahip olup olmadığını öğrenmeyi sağlar.

Metotlar:

Tablo 13.5. Graphics Sınıfı Metodları

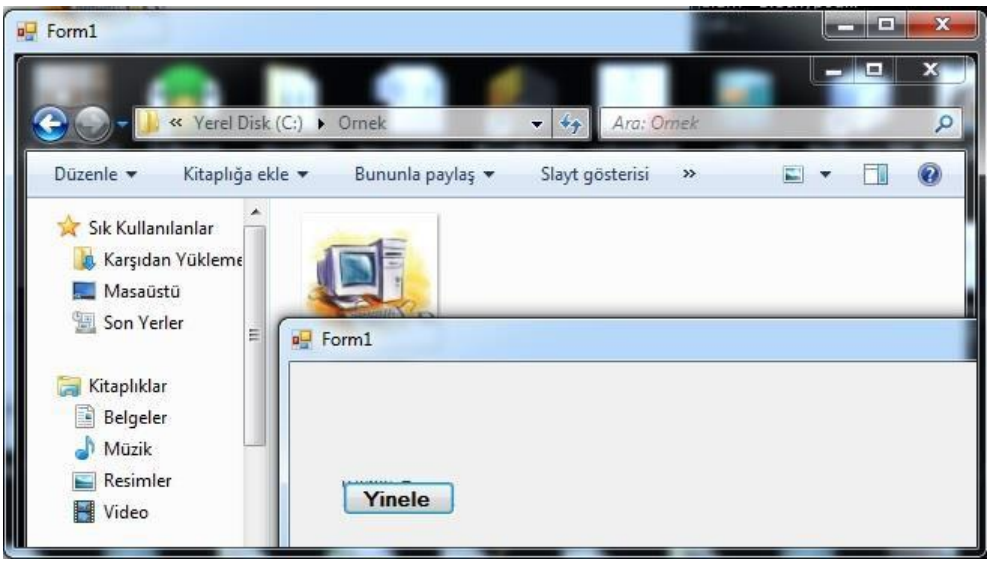
Özellik	İşlev
Clear(renk As Color)	Bütün çizim yüzeyini temizlemeyi ve arka planı parametre olarak verilen renge boyamayı sağlar.
CopyFromScreen(kaynak_koor As Point, hedef_koor As Point ,hedef_alan As Size)	O anki ekran görüntüsünü hedef forma transfer etmeyi sağlar.
DrawArc(kalem As Pen, alan As Rectangle, bas_aci As Single, son_aci As Single)	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, ikinci parametrede belirtilen dikdörtgen alanın sınırlarında, üçüncü parametrede verilen başlangıç açısından itibaren, dördüncü parametrede verilen açıya kadar saat yönünde yay çizmeyi sağlar.
DrawClosedCurve(kalem As Pen, noktalar As Point())	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, ikinci parametrede verilen noktalar dizisindeki tüm noktaların birleşiminden oluşan kapalı bir eğri çizmeyi sağlar.
DrawCurve(kalem As Pen, noktalar As Point())	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, ikinci parametrede verilen noktalar dizisindeki tüm noktaların birleşiminden oluşan bir eğri çizmeyi sağlar.
DrawEllipse(kalem As Pen, sinir As Rectangle)	İkinci parametrede verilen dikdörtgen alanın sınırları içerisinde, ilk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde bir elips çizmeyi sağlar.
DrawIcon(ikon As Icon, alan As Rectangle)	İkinci parametrede verilen dikdörtgen alana, ilk parametrede verilen ikonu çizmeyi sağlar.
DrawImage(resim As Image, nokta As Point)	İkinci parametrede verilen noktadan itibaren, birinci parametrede verilen resmi çizmeyi sağlar.
DrawLine(kalem As Pen, nokta1 As Point, nokta2 As Point)	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, ikinci parametrede verilen

	noktadan, üçüncü parametrede verilen noktaya kadar bir çizgi çizmeyi sağlar.
DrawPie(kalem As Pen, alan As Rectangle,bas_aci As Single, bit_aci As Single)	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, ikinci parametrede belirtilen dikdörtgenin sınırları içerisinde, üçüncü parametrede belirtilen açıdan itibaren, saat yönünde dördüncü parametrede verilen açıya kadar bir pasta dilimi çizmeyi sağlar.
DrawPolygon(kalem As Pen, noktalar As Point())	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, ikinci parametrede verilen noktalar dizisini kullanarak bir poligon çizmeyi sağlar.
DrawRectangle(kalem As Pen, alan As Rectangle)	İlk parametrede belirtilen kalem nesnesinin renk, genişlik ve stilinde, parametrede verilen alana bir dikdörtgen çizmeyi sağlar.
DrawString(yazi As String, font As Font, firca As Brush, nokta As PointF)	İlk parametrede verilen yazıyı, ikinci parametrede verilen fontta, üçüncü parametrede verilen fırça ile dördüncü parametrede verilen noktadan itibaren ekrana çizmeyi sağlar.
FillRegion(firca As Brush, alan As Region)	İkinci parametrede verilen bölgeyi, ilk parametrede verilen fırça ile boyamayı sağlar.
FromImage(resim As Image) As Graphics	İlk parametrede verilen resmi kullanarak bir Graphics nesnesi oluşturmayı sağlar.
GetNearestColor(renk As Color) As Color	Parametre olarak verilen Color nesnesine en yakın Color nesnesini geri döndürmeyi sağlar.

Bireysel Etkinlik

• İsmi Draw ile başlayan metodların Fill ile başlayan hâlleri mevcuttur. Bu metotları inceleyerek farklarını görünüz.

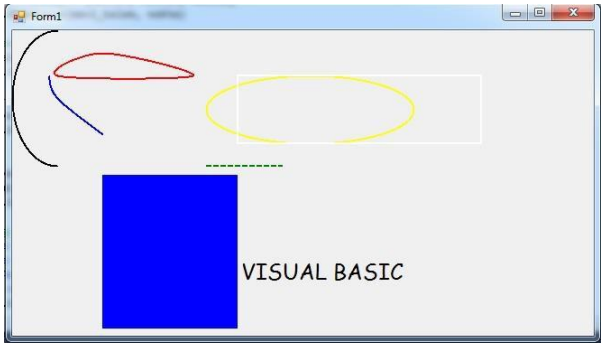
İlk örnek, form üzerine o anki ekran görüntüsünü alarak çizdirmek olacaktır. Bu amaçla bir form tasarlanarak üzerine bir Button nesnesi koymak yeterli olacaktır. Bu amaçla hazırlanan projenin çalıştırıldıktan sonraki ekran görüntüsü Şekil 13.2.'de verilmiştir. Her yinele butonuna basıldığında o anki ekran görüntüsü alınarak form üzerine çizilmektedir.



Şekil 13.2. CopyFromScreen Metodu İçin Tasarlanan Form

```
Public Class Form1
    Dim ekran_resim As Bitmap

    Private Sub Form1_Paint(sender As Object, e As PaintEventArgs) Handles
                                        MyBase.Paint
        Dim grafik As Graphics = Me.CreateGraphics()
        Dim boyut As Size = Me.Size
        ekran_resim = New Bitmap(boyut.Width, boyut.Height, grafik)
        Dim hafiza_resim As Graphics = Graphics.FromImage(ekran_resim)
        hafiza_resim.CopyFromScreen(0, 0, 0, 0, boyut)
        e.Graphics.DrawImage(ekran_resim, 0, 0)
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Invalidate()
    End Sub
```

İkinci örnek olarak ise çeşitli Graphics sınıfı içerisindeki çeşitli metotların kullanımları bir form üzerinde örneklendirilmiştir. Bu formun çalıştıktan sonraki görünümü şekil 13.3.'te verilmiştir.



Şekil 13.3. Grafik Sınıfı Metotları İçin Tasarlanan Form

```
Public Class Form1

    Private Sub Form1_Paint(sender As Object, e As PaintEventArgs) Handles
MyBase.Paint
        'DrawArc
        Dim siyah_kalem As New Pen(Color.Black, 2)
        e.Graphics.DrawArc(siyah_kalem, New Rectangle(0, 0, 100, 150), 90,
180)

        'DrawClosedCurve
        Dim kirmizi_kalem As New Pen(Color.Red, 2)
        Dim nokta1 As New Point(50, 50)
        Dim nokta2 As New Point(100, 25)
        Dim nokta3 As New Point(200, 50)
        Dim noktalar As Point() = {nokta1, nokta2, nokta3}
        e.Graphics.DrawClosedCurve(kirmizi_kalem, noktalar)

        'DrawCurve
        Dim mavi_kalem As New Pen(Color.Blue, 2)
        Dim nokta4 As New Point(40, 50)
        Dim nokta5 As New Point(50, 75)
        Dim nokta6 As New Point(100, 115)
        Dim nokta As Point() = {nokta4, nokta5, nokta6}
        e.Graphics.DrawCurve(mavi_kalem, nokta)

        'DrawEllipse
        Dim sari_kalem As New Pen(Color.Yellow, 2)
        e.Graphics.DrawEllipse(sari_kalem, New Rectangle(215, 50, 230, 75))

        'DrawRectangle
        Dim beyaz_kalem As New Pen(Color.White, 2)
        e.Graphics.DrawRectangle(beyaz_kalem, 250, 50, 270, 75)

        'DrawLine
        Dim yesil_kalem As New Pen(Color.Green, 2)
        yesil_kalem.DashStyle = Drawing2D.DashStyle.Dash
        e.Graphics.DrawLine(yesil_kalem, 215, 150, 300, 150)

        'DrawString
        Dim yazi As String = "VISUAL BASIC"
        Dim font As New Font("Comic Sans MS", 18)
        Dim firca As New SolidBrush(Color.Black)
        Dim ynokta As New PointF(250.0F, 250.0F)
        e.Graphics.DrawString(yazi, font, firca, ynokta)

        'FillRectangle
        Dim yeni_firca As New SolidBrush(Color.Blue)
        e.Graphics.FillRectangle(yeni_firca, 100, 160, 150, 170)

    End Sub
End Class
```

Icon Sınıfı

Bir ikon nesnesi oluşturmayı sağlayan sınıftır.

Kurucu Metotlar:

Icon(dosya As String)

Dosya: Oluşturulacak Icon nesnesi içerisinde kullanılacak olan dosyanın verildiği parametredir.

Icon(dosya As String, boyut As Size):

Dosya: Oluşturulacak Icon nesnesi içerisinde kullanılacak olan dosyanın verildiği parametredir.

Size: Oluşturulacak Icon nesnesinin boyutlarının verildiği parametredir.

Özellikler:

Tablo 13.6. Icon Sınıfı Özellikleri

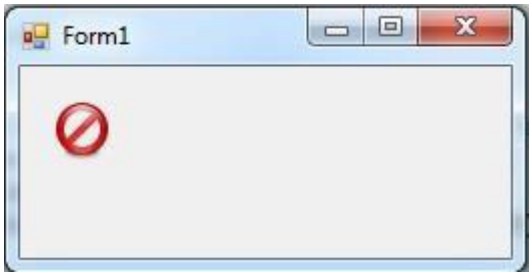
Özellik	İşlev
Height	İkon nesnesinin yüksekliğini öğrenmeyi sağlar.
Size	İkon nesnesinin boyutlarını öğrenmeyi sağlar.
Width	İkon nesnesinin genişliğini öğrenmeyi sağlar.

Metodlar:

Tablo 13.7. Icon Sınıfı Metodları

Özellik	İşlev
Clone() As Object	Icon nesnesinin bir örneğinin oluşturulmasını sağlar.
Dispose()	Icon türündeki nesnenin, kullandığı tüm sistem kaynaklarını serbest bırakmasını sağlar.
ToBitmap() As Bitmap	İkon nesnesini GDI+ Bitmap nesnesine dönüştürmeyi sağlar.

İkon sınıfı için varolan bir ikonu Bitmap nesnesine dönüştürerek form üzerine çizen bir uygulama yapılmış ve uygulamanın ekran görüntüsü Şekil 13.4.'te verilmiştir.



Şekil 13.4. Icon Sınıfı İçin Tasarlanan Form

```
    Private Sub Form1_Paint(sender As Object, e As PaintEventArgs) Handles
MyBase.Paint
        Dim ikon As New Icon("c:\ornek\delete.ico")
        Dim resim As Bitmap = ikon.ToBitmap()
        e.Graphics.DrawImage(resim, New Point(15, 15))
    End Sub
```

Image Sınıfı

Bitmap ve Metafile dosyaları için işlevsellik sağlayan bir abstract sınıftır.

Özellikler:

Tablo 13.8. Image Sınıfı Özellikleri

Özellik	İşlev
Height	Image nesnesinin yüksekliğini öğrenmeyi sağlar.
Palette	Image nesnesinin kullandığı renk paletini öğrenmeyi ve değiştirmeyi sağlar.
PixelFormat	Image nesnesinin kullandığı pixel formatını öğrenmeyi sağlar.
PropertyItems	Image nesnesi içerisinde saklanan tüm özelliklerin (metadata) öğrenilmesini sağlar.
Size	Image nesnesinin boyutlarını öğrenmeyi sağlar.
Tag	Image nesnesi hakkında extra veri saklamayı ve bu veriyi öğrenmeyi sağlar.
Width	Image nesnesinin genişliğini öğrenmeyi sağlar.

Metotlar:

Tablo 13.9. Image Sınıfı Metodları

Özellik	İşlev
Clone() As Object	Image nesnesinin bir örneğinin oluşturulmasını sağlar.
Dispose()	Image türündeki nesnenin, kullandığı tüm sistem kaynaklarını serbest bırakmasını sağlar.
FromFile(dosya As String) As Image	İlk parametrede verilen resmi kullanarak bir Image nesnesi oluşturmayı sağlar.
Save(dosya_ismi As String)	Image nesnesinin içerdiği resmi parametre olarak verilen dosyaya kaydetmeyi sağlar.
Save(dosya_ismi As String, format AS ImageFormat)	Image nesnesinin içerdiği resmi parametre olarak verilen dosyaya, ikinci parametrede verilen formatta kaydetmeyi sağlar.
SetPropertyItem(ozellik As PropertyItem)	Image içerisine parametre olarak verilen metadatayı saklamayı sağlar.

Image sınıfı için ayrı bir form tasarımı yapılmamış bu bölümdeki ilk örneğe ait tasarım kullanılarak yazılan kodlar değiştirilmiştir. Bu örnekte amaç FromFile metodunu kullanmak ve resmi y ekseninde döndürerek yeni halini pb2 nesnesinde göstermektir. İlgili ekran görüntüsü Şekil 13.5.'te gösterilmiştir.

Şekil 13.5. Image Sınıfı İçin Tasarlanan Form

```
Imports System.Drawing
Public Class Form1
   Private Sub btnyukle_Click(sender As Object, e As EventArgs) Handles
btnyukle.Click
         Dim resim As Bitmap = Image.FromFile("c:\Ornek\computer.jpg")
      pb1.Image = resim
   End Sub

   Private Sub btnsiyahbeyaz_Click(sender As Object, e As EventArgs) Handles
btnsiyahbeyaz.Click
      Dim resim As Bitmap = Image.FromFile("c:\Ornek\computer.jpg")
      resim.RotateFlip(RotateFlipType.Rotate180FlipY)
      pb2.Image = resim
   End Sub
End Class
```

Pen Sınıfı

Çizgi ve Eğrilerin kullanacağı çizim nesnesini tanımlamayı sağlar.

Çizimlerin hangi renkte, hangi kalınlıkta kalemle yapılacağı gibi bilgiler Pen sınıfı ile belirlenir.

Kurucu Metodlar:

Pen(firca As Brush)

firca: Pen nesnesinin kullanacağı brush nesnesini belirlemeyi sağlar.

Pen(renk As Color)

renk: Pen nesnesinin kullanacağı renk nesnesini belirlemeyi sağlar.

Pen(firca As Brush, genislik As Single)

firca: Pen nesnesinin kullanacağı brush nesnesini belirlemeyi sağlar.

genislik: Pen nesnesinin kullanacağı çizgi genişliğini belirlemeyi sağlar.

Pen(renk As Color, genislik As Single)

renk: Pen nesnesinin kullanacağı renk nesnesini belirlemeyi sağlar.

genislik: Pen nesnesinin kullanacağı çizgi genişliğini belirlemeyi sağlar.

Özellikler:

Tablo 13.10. Pen Sınıfı Özellikleri

Özellik	İşlev
Brush	Pen nesnesinin kullandığı Brush nesnesi hakkında bilgi almayı ve Brush nesnesini değiştirmeyi sağlar.
Color	Pen nesnesinin kullandığı Color nesnesi hakkında bilgi almayı ve Color nesnesini değiştirmeyi sağlar.
DashStyle	Pen nesnesi yardımı ile çizilecek kesikli çizginin stilini öğrenmeyi veya belirlemeyi sağlar.
EndCap	Bir çizginin sonuna koyulacak olan simgeyi belirlemeyi veya varolan simgeyi öğrenmeyi sağlar.
PenType	Pen nesnesi ile çizilecek çizginin stilini öğrenmeyi sağlar.
StartCap	Bir çizginin başına koyulacak olan simgeyi belirlemeyi veya varolan simgeyi öğrenmeyi sağlar.

Metotlar:

Tablo 13.11. Pen Sınıfı Metodları

Özellik	İşlev
Clone() As Object	Pen nesnesinin bir örneğinin oluşturulmasını sağlar.
Dispose()	Pen türündeki nesnenin, kullandığı tüm sistem kaynaklarını serbest bırakmasını sağlar.

Özet

- GDI+ iki boyutlu vektör grafikler, görüntülem, tipografi gibi işlemleri gerçekleştirmek için geliştirilmiştir ve Windows XP işletim sisteminin bir parçasıdır.
- Bir bitmap, pixel veriler tarafından tanımlanmış şekiller ile çalışmak için kullanılan bir sınıftır.
- Brush, dikdörtgen, elips, poligon ve yollar gibi grafiksel şekillerin içini doldurmak için kullanılan nesnedir.
- Font, bir metnin, font tipi, büyüklüğü ve stil özelliklerini tanımlamayı sağlayan bir sınıftır.
- Çizimlerin çoğu Graphics sınıfından oluşturulan nesnelerin yardımı ile yapılır.
- Icon, bir ikon nesnesi oluşturmayı sağlayan sınıftır.
- Image, Bitmap ve Metafile dosyaları için işlevsellik sağlayan bir abstract sınıftır.
- Çizimlerin hangi renkte, hangi kalınlıkta kalemle yapılacağı gibi bilgiler Pen sınıfı ile belirlenir

DEĞERLENDİRME SORULARI

1. Çizim ile ilgili sınıflar hangi namespace'de bulunur?
    a) System.Graphics
    b) Microsoft.Graphics
    c) System.Drawing
    d) Microsoft.Drawing
    e) System.Bitmap

2. Bir resmin pixel bilgilerine erişmek için hangi sınıf kullanılmalıdır?
    a) Bitmap
    b) Pen
    c) Graphics
    d) Image
    e) Brush

3. GetPixel() metodu geriye hangi tipte bilgi döndürür?
    a) Pen
    b) Size
    c) Brush
    d) Color
    e) Font

4. O anki ekran görüntüsünü alabilmek için Graphics sınıfının hangi metodu kullanılır?
    a) GetScreen
    b) CopyFromSecreen
    c) Screen
    d) PutScreen
    e) PasteScreen

5. Forma bir yay çizdirmek için Graphics sınıfının hangi metodu kullanılır?
    a) DrawArc
    b) DrawBow
    c) DrawArch
    d) DrawCoil
    e) DrawSpring

6. Forma bir dörtgen çizdirmek için Graphics sınıfının hangi metodu kullanılır?
    a) DrawSquare
    b) DrawRectangle
    c) DrawTriangle
    d) DrawQuadrangle
    e) DrawTrapez

7. Icon sınıfından oluşan bir nesneyi Bitmap sınıfına çevirmek için hangi metot kullanılır?
    a) ToGraphics
    b) ToImage
    c) ToBitmap
    d) ConvertImage
    e) ConvertBıtmap

8. Bir resim dosyasından image nesnesi oluşturabilmek için hangi metot kullanılır?
    a) ImageFromFile
    b) ImageFromStream
    c) FromFile
    d) FromStream
    e) GetFile

9. Bir çizginin stili Pen sınıfının hangi özelliği ile belirlenir?
    a) CapStyle
    b) DrawStyle
    c) LineStyle
    d) PenStyle
    e) DashStyle

10. Bitmap sınıfı aşağıdaki dosya türlerinden hangisini desteklemez?
    a) jpeg
    b) gif
    c) exif
    d) psd
    e) tiff

Cevap Anahtarı

1.c, 2.a, 3.d, 4.b, 5.a, 6.b, 7.c, 8.c, 9.e, 10.d

YARARLANILAN KAYNAKLAR

http://msdn.microsoft.com

ÖRNEK UYGULAMALAR

İÇİNDEKİLER

- Örnek Uygulamalar
  - Sayısal Loto Uygulaması
  - Anket Uygulaması
  - Adam Asmaca Uygulaması

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Bu zamana kadar öğrendiğiniz konularla ilgili yeni örnekler yaparak bilgilerinizi pekiştirebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Doç. Dr. Ecir Uğur KÜÇÜKSİLLE

ÜNİTE
14

GİRİŞ

Bu bölümde şimdiye kadar öğrenmiş olduğunuz nesneler, bu nesnelerin özellikleri, olayları ile ilgili üç yeni örnek yapılmıştır. Bu örneklerle bilgilerinizin pekiştirilmesi amaçlanmıştır. Yapılan örneklerde, nesnelerin otomatik olarak kullanılma kabiliyetini artırma, dinamik uygulamalar geliştirme ve uygulamanıza görsellik katacak nesneleri kullanma becerilerinizin artırılması amaçlanmıştır.

ÖRNEK UYGULAMALAR

Ünitede verilen sayısal loto uygulaması ile dinamik kontrollerin kullanımı ve özelliklerine değinilmiştir. Anket uygulamasında bir metin belgesindeki verilerin .NET grafik sınıfları yardımıyla görsel bir şekilde sunulması sağlanmıştır. Adam asmaca uygulamasında ise grafik nesneleri, paneller ve sık kullanılan bazı kontroller kullanılmıştır.

Sayısal Loto Uygulaması

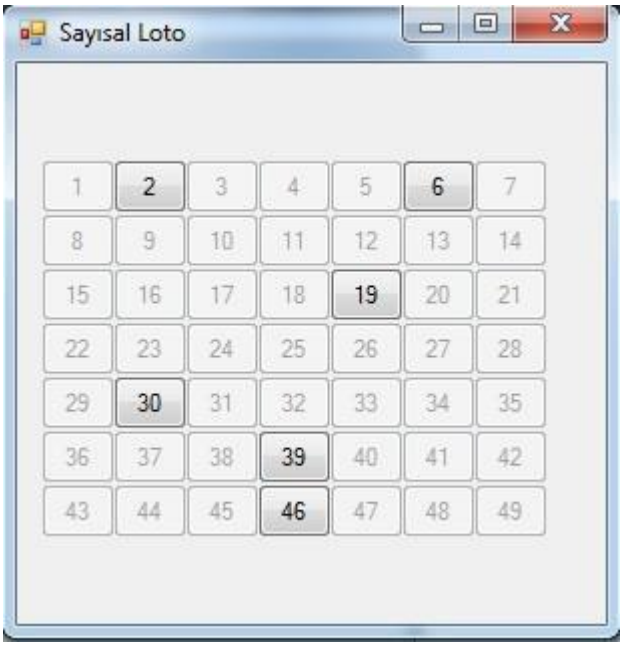
Bu uygulamada, form üzerine dinamik kontrol ekleme ve bu kontrollere daha sonra ulaşarak özelliklerini değiştirebilme işlemlerinin nasıl yapılacağı konusu anlatılmaya çalışılmıştır. Öncelikle form açıldığında form üzerine 49 adet buton yerleştirilmektedir. Uygulamanın amacı, her 'A' veya 'a' tuşuna basıldığında 1 ile 49 arasında rastgele birbirinden farklı altı sayı üreterek ilgili butonları aktif yapmak diğer butonları pasif yapmaktır. Uygulamanın tasarım ekranı görüntüsü Şekil 14.1.'de gösterilmiştir.

Şekil 14.1. Uygulamanın Tasarım Ekranı Görüntüsü

Bu tasarım hazırlandıktan sonra, formun KeyPreview özelliği true yapılmalıdır. Bu özelliğin true yapılmasındaki amaç, tüm nesnelerin key olaylarından önce formun key olaylarının tetiklenmesini sağlamaktır. Tüm bu işlemlerden sonra aşağıda verilen kodlar yazılıp program çalıştırıldığında oluşan ekran görüntüsü Şekil 14.2.'de verilmiştir.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim x As Integer = 12
        Dim y As Integer = 48
        For i = 1 To 49
            Dim buton As New Button()
            buton.Name = "b" & i
            buton.Text = i
            buton.Enabled = False
            buton.Size = New Size(37, 27)
            buton.Location = New Point(x, y)
            If i Mod 7 <> 0 Then
            x += 36
        Else
            x = 12
            y += 27
        End If
    Controls.Add(buton)
    Next
    End Sub

    Private Sub pasif_yap()
        For i = 0 To Controls.Count - 1
            Dim x = DirectCast(Controls(i), Button)
            If x IsNot Nothing Then
                x.Enabled = False
            End If
        Next
    End Sub

    Private Function sayiUret() As Integer()
        Dim sayilar(5) As Integer
        Dim rnd As New Random()
        Dim varmi As Boolean
        Dim sayi, k As Integer
        For index = 0 To 5
            varmi = True
            While varmi
                sayi = rnd.Next(49) + 1
                For k = 0 To 5
                    If sayilar(k) = sayi Then
                        Exit For
                    End If
                Next
                If k = 6 Then
                    sayilar(index) = sayi
                    Exit While
                End If
            End While
        Next
        Return sayilar
    End Function
```

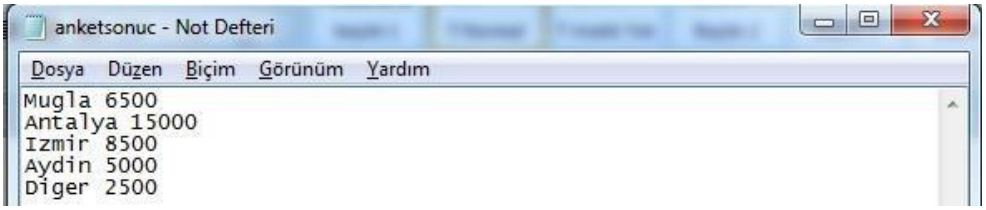
```
Private Sub Form1_KeyPress(sender As Object, e As KeyPressEventArgs)
Handles MyBase.KeyPress
      If e.KeyChar = Microsoft.VisualBasic.ChrW(Keys.A) Or e.KeyChar =
Microsoft.VisualBasic.ChrW(97) Then
         Dim sonuclar(6) As Integer
         sonuclar = sayiUret()
         pasif_yap()
         Dim nesne() As Control
         For i = 1 To 6
            nesne = Controls.Find("b" & sonuclar(i - 1), True)
            nesne(0).Enabled = True
         Next
      End If
   End Sub
End Class
```



Şekil 14.2. Uygulama Ekranı Görüntüsü

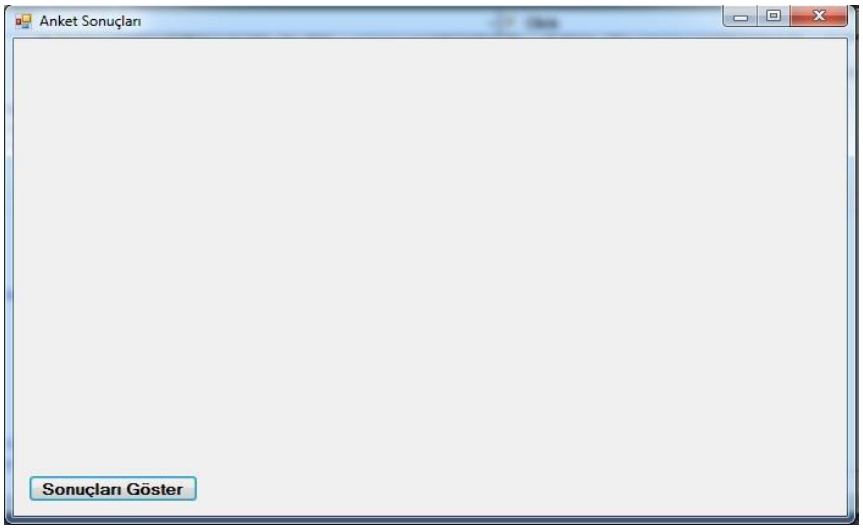
Anket Uygulaması

Bu uygulamada amaç bir text dosyada tutulan anket verilerini okuyarak bu verilere ait bar ve pasta grafikleri ekrana çizebilmektir. Bu amaçla oluşturulmuş anketsonuc.txt isimli dosyanın içeriğine ait ekran görüntüsü Şekil 14.3.'te verilmiştir.



Şekil 14.3. anketsonuc.txt Dosyası

Bu amaçla tasarlanan formun ekran görüntüsü Şekil 14.4.'te verilmiştir.



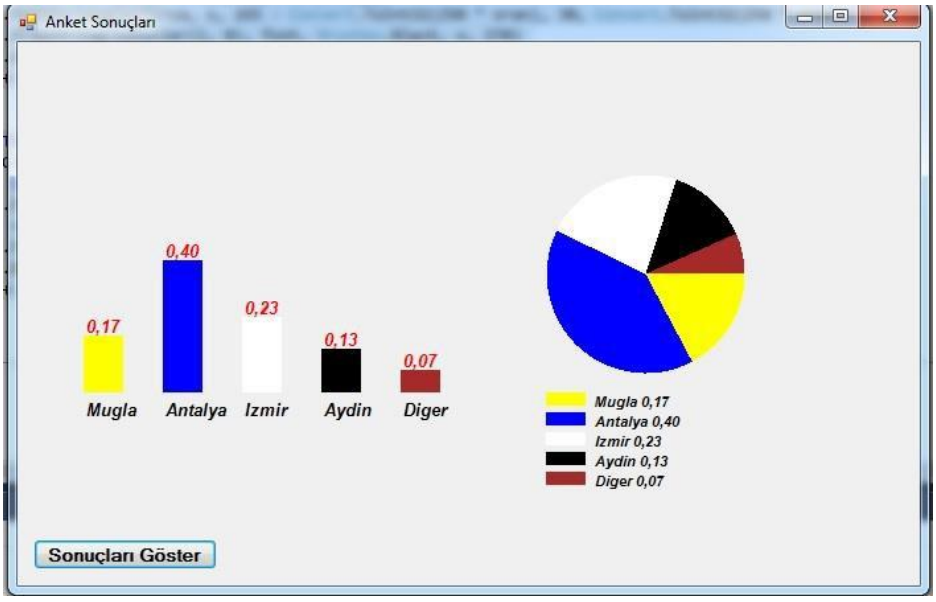
Şekil 14.4. Uygulamanın Tasarım Ekranı Görüntüsü

Bu ekran tasarlandıktan sonra aşağıdaki kodlar yazılmalıdır.

```
Imports System.IO
Imports System.Text
Public Class Form1

   Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
      Dim sonuclar(4, 1) As String
      Dim i, toplam_oy, oran As Double
      Dim x, basaci As Integer
      Dim grafik As Graphics = Me.CreateGraphics()
      Dim renkler() As Color = {Color.Yellow, Color.Blue, Color.White,
Color.Black, Color.Brown}
      Dim firca As New SolidBrush(Color.AliceBlue)
      Dim font As New Font("Arial", 10, FontStyle.Bold Or FontStyle.Italic)
      Dim font1 As New Font("Arial", 8, FontStyle.Bold Or FontStyle.Italic)
      i = 0
      toplam_oy = 0
      x = 50
      basaci = 0
      Dim dosya As New StreamReader(Directory.GetCurrentDirectory &
"\anketsonuc.txt", Encoding.UTF8)
      While dosya.Peek() >= 0
         Dim dizi() As String = dosya.ReadLine().Split(" ")
         sonuclar(i, 0) = dizi(0)
         sonuclar(i, 1) = dizi(1)
         i = i + 1
      End While
```

```
dosya.Close()
        For i = 0 To 4
            toplam_oy = toplam_oy + Integer.Parse(sonuclar(i, 1))
        Next
        For i = 0 To 4
            firca.Color = renkler(i)
            oran = Double.Parse(sonuclar(i, 1)) / toplam_oy
            grafik.FillRectangle(firca, x, 265 - Convert.ToInt32(250 * oran), 30,
Convert.ToInt32(250 * oran))
            grafik.DrawString(sonuclar(i, 0), font, Brushes.Black, x, 270)
            grafik.DrawString(oran.ToString("0.00"), font, Brushes.Red, x, 250 -
Convert.ToInt32(250 * oran))
            x = x + 60
        Next
        x = 265
        For i = 0 To 4
            firca.Color = renkler(i)
            oran = Double.Parse(sonuclar(i, 1)) / toplam_oy
            grafik.FillPie(firca, 400, 100, 150, 150, basaci, Convert.ToInt32(360 *
oran))
            basaci = basaci + Convert.ToInt32(360 * oran)
            grafik.FillRectangle(firca, 400, x, 30, 10)
            grafik.DrawString(sonuclar(i, 0) + " " + oran.ToString("0.00"), font1,
Brushes.Black, 435, x)
            x = x + 15
        Next
    End Sub
End Class
```

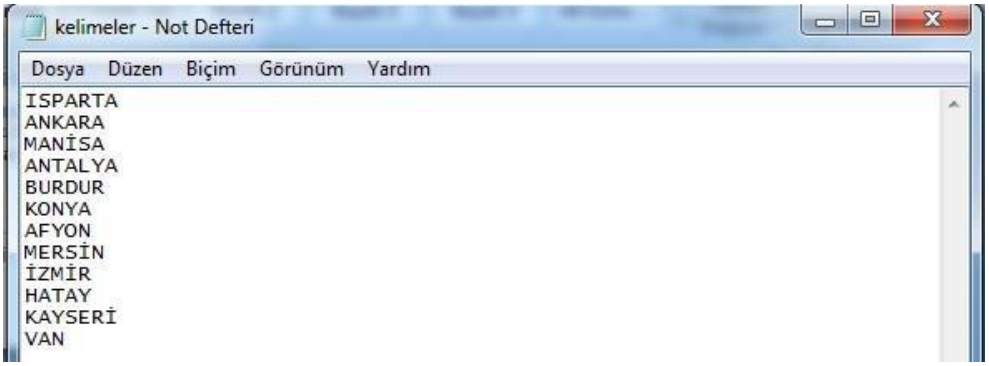
Uygulama çalıştırılıp Sonuçları Göster butonuna basıldığında oluşan ekran görüntüsü Şekil 14.5.'te verilmiştir.



Şekil 14.5. Uygulama Ekran Görüntüsü

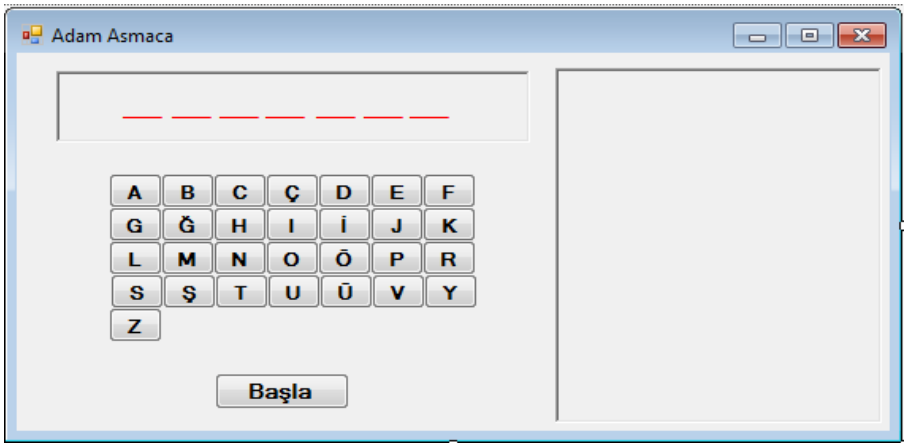
Adam Asmaca Uygulaması

Bu uygulama ile çok bilinen adam asmaca oyunu yapılmaya çalışılmıştır. Bu amaçla kelimeler.txt isimli bir text dosya oluşturularak içerisine rastgele il isimleri yerleştirilmiştir. Bu dosyanın içeriği Şekil 14.6.'da verilmiştir.



Şekil 14.6. kelimeler.txt Dosyası

Bu dosyanın oluşturulmasından sonra uygulama için gerekli form tasarlanmıştır. Tasarlanan forma ait ekran görüntüsü Şekil 14.7.'de verilmiştir.



Şekil 14.7. Uygulamanın Tasarım Ekranı Görüntüsü

Yukarıda görülen formda iki adet panel kullanılmıştır. Bunlardan ilki, tahmin edilecek kelimenin gösterildiği Label'ları içermektedir. Diğeri ise, asılan çöp adamın çizilmesini sağlamaktadır. Kullanıcının üç yanlış hakkı vardır. Bu formda harfleri gösteren butonların hepsine ortak kontrol isimli bir olay eklenmiştir. Bu işlem için, öncelikle harfleri gösteren tüm butonlar seçilir. Daha sonra, Properties penceresinden click olayına gidilir ve karşısına kontrol yazılarak enter tuşuna basılır. Bu uygulama için yazılan kodlar aşağıda verilmiştir.

```
Imports System.IO
Imports System.Text
Public Class Form1
   Dim sec_kelime As String
   Dim hak As Integer
   Dim harf_say As Integer
   Dim p2 As Graphics
   Dim kalem As New Pen(Color.Black, 2)
```

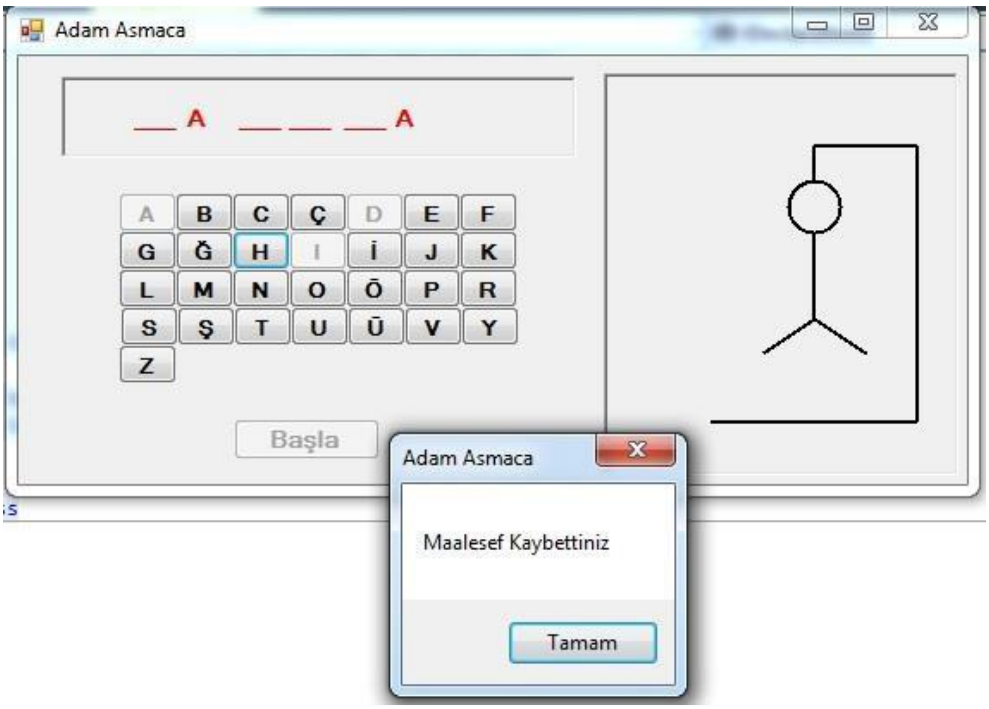
```
Private Sub labelGizle()
    For index = 0 To labelpanel.Controls.Count - 1
        Dim nesne = TryCast(labelpanel.Controls(index), Label)
        If nesne IsNot Nothing Then
            nesne.Visible = False
        End If
    Next
End Sub

Private Sub butonDurum(ByVal durum As Integer)
    For index = 0 To Controls.Count - 1
        Dim nesne = TryCast(Controls(index), Button)
        If nesne IsNot Nothing Then
            If nesne.Name <> "btn_basla" Then
                If durum = 1 Then
                    nesne.Enabled = False
                Else
                    nesne.Enabled = True
                End If
            End If
        End If
    Next
End Sub

Function kelimeGetir(ByVal dosya_ismi As String) As String
    Dim kelime As String = ""
    Dim sayi, rsayi As Integer
    Dim rnd As New Random()
    sayi = 0
    'Dosyadaki kayit sayisi bulunuyor
    Dim dosya As New StreamReader(dosya_ismi, Encoding.Default)
    While dosya.Peek() >= 0
        dosya.ReadLine()
        sayi = sayi + 1
    End While
    'Kayit sayisi bulunması islemi sonu
    'Rastgele sayi secilip ilgili kelime dosyadan alınıyor
    'Pointer tekrar dosya basina getiriliyor
    dosya.BaseStream.Position = 0
    dosya.DiscardBufferedData()
    'Dosya basi islemi sona erdi
    rsayi = rnd.Next(sayi)
    For index = 1 To rsayi - 1
        dosya.ReadLine()
    Next
    kelime = dosya.ReadLine()
    'kelime secme islemi sonu
    Return kelime
End Function
```

```
Function nesneBul(ByVal isim As String) As Label
    Dim sonuc As Label = Nothing
    For index = 0 To labelpanel.Controls.Count - 1
      Dim nesne As Label = TryCast(labelpanel.Controls(index), Label)
      If nesne IsNot Nothing Then
        If nesne.Name = isim Then
          sonuc = nesne
        End If
      End If
    Next
    Return sonuc
End Function

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
    labelGizle()
    butonDurum(1)
    p2 = adampanel.CreateGraphics()
End Sub

Private Sub txt_basla_Click(sender As Object, e As EventArgs) Handles
btn_basla.Click
    sec_kelime = kelimeGetir(Directory.GetCurrentDirectory() +
"\kelimeler.txt")
    labelGizle()
    For index = 1 To sec_kelime.Length
      nesneBul("l" & index).Text = "___"
      nesneBul("l" & index).Visible = True
    Next
    butonDurum(2)
    btn_basla.Enabled = False
    p2.Clear(labelpanel.BackColor)
    hak = 0
    harf_say = 0
End Sub

Private Sub kontrol(sender As Object, e As EventArgs) Handles b9.Click,
b8.Click, b7.Click, b6.Click, b5.Click, b4.Click, b3.Click, b29.Click, b28.Click,
b27.Click, b26.Click, b25.Click, b24.Click, b23.Click, b22.Click, b21.Click,
b20.Click, b2.Click, b19.Click, b18.Click, b17.Click, b16.Click, b15.Click,
b14.Click, b13.Click, b12.Click, b11.Click, b10.Click, b1.Click
    Dim buton As Button = TryCast(sender, Button)
    If (sec_kelime.IndexOf(buton.Text) > -1) Then
      For index = 0 To sec_kelime.Length - 1
        If sec_kelime.Substring(index, 1) = buton.Text Then
          nesneBul("l" & (index + 1)).Text = buton.Text
          harf_say = harf_say + 1
        End If
      Next
```

```
    If harf_say = sec_kelime.Length Then
            MsgBox("Tebrikler Kazandınız")
            btn_basla.Enabled = True
            butonDurum(1)
        End If
      Else
        hak = hak + 1
        If hak = 1 Then
          p2.DrawLine(kalem, 60, 200, 180, 200)
        ElseIf hak = 2 Then
          p2.DrawLine(kalem, 180, 40, 180, 200)
          p2.DrawLine(kalem, 120, 40, 180, 40)
          p2.DrawLine(kalem, 120, 40, 120, 60)
        ElseIf hak = 3 Then
          p2.DrawEllipse(kalem, 105, 60, 30, 30)
          p2.DrawLine(kalem, 120, 90, 120, 140)
          p2.DrawLine(kalem, 120, 140, 90, 160)
          p2.DrawLine(kalem, 120, 140, 150, 160)
          MsgBox("Maalesef Kaybettiniz")
          For index = 0 To sec_kelime.Length - 1
            nesneBul("l" & (index + 1)).Text = sec_kelime.Substring(index, 1)
          Next
          butonDurum(1)
          btn_basla.Enabled = True
        End If
      End If
      buton.Enabled = False
    End Sub
End Class
```

Uygulama çalıştırıldıktan sonraki iki ekran görüntüsü sırası ile Şekil 14.8. ve Şekil 14.9'da verilmiştir.

Şekil 14.5. Uygulama Birinci Ekran Görüntüsü



Şekil 14.6 Uygulama İkinci Ekran Görüntüsü

Özet

- Görsel programlamada nesnelerin özellikleri ve olayları program akışına yön vermektedir.
- Bu bölümde yapılan örneklerde, nesnelerin otomatik olarak kullanılma kabiliyetini artırma, dinamik uygulamalar geliştirme ve uygulamalara görsellik katacak nesneleri kullanma becerilerinin kazandırılması amaçlanmıştır.
- Bu kapsamda;
  - Bir metin belgesindeki verilerin .NET grafik sınıfları yardımıyla görsel bir şekilde sunulması,
  - Grafik nesneleri, paneller ve sık kullanılan bazı kontrollerin kullanılması,
  - Form üzerine dinamik kontrol ekleme ve bu kontrollerin özelliklerinin değiştirilmesi,
  - Stream sınıfını kullanılarak dosya okuma/yazma işlemlerinin yapılması sağlanmıştır.

DEĞERLENDİRME SORULARI

1. Private Sub pasif_yap()
   ……. i = 0 To Controls.Count - 1
   Dim x = DirectCast(Controls(i), Button)
   If x IsNot Nothing Then
   x.Enabled = False
   End If
   Next
   End Sub

   Yukarıdaki kod bloğundaki boşluğa aşağıdakilerden hangisi getirilmelidir?
   a) For
   b) Dim
   c) While
   d) If
   e) Else

2. Dim sonuclar(4, 1) As String şeklindeki kod bloğu ne anlama gelmektedir?
   a) Döngü
   b) Karar yapısı
   c) Değişken tanımlama
   d) Kontrol yapısı
   e) Fonksiyon tanımlama

3. Dim buton As New Button() şeklindeki kod bloğu ne anlama gelmektedir?
   a) Button tipinde bir dizi tanımlanmıştır.
   b) Button tipinde bir koleksiyon tanımlanmıştır.
   c) Button kontrolüne erişim sağlanmıştır.
   d) Yeni bir Button kontrolü oluşturulmuştur.
   e) Button kontrolüne değişken atanmıştır.

4. Form üzerine grafik çizdirmeyi sağlayan komut aşağıdakilerden hangisidir?
   a) Fill
   b) Color
   c) Rectangle
   d) Draw
   e) CreateGraphics

5. Dim font As New Font("Arial", 10) şeklindeki kod bloğu ne anlama gelmektedir?
   a) Font isimli değişkenin boyut ve isim özellikleri belirlenir.
   b) Font isimli değişkenin stili belirlenir.
   c) Font isimli değişkenin rengi belirlenir.
   d) Font isimli değişkenin tipi belirlenir.
   e) Font isimli bir dizi değişkeni tanımlanmıştır.

6. Random() tipinde bir değişken ne için tanımlanır?
   a) Sadece 0 ve 9 arasında yer alacak bir rakam türetmek için
   b) Sadece eksi değerde bir sayı türetmek için
   c) Rastgele bir sayı türetmek için
   d) String tipinde karakterler saklamak için
   e) GUID değeri üretmek için

7. MsgBox() metodunun temel kullanım amacı aşağıdakilerden hangisidir?
   a) Kullanıcının bir değer girmesi için
   b) Mesaj göstermek için
   c) Programda çıkabilecek hataları yakalamak için
   d) Programın arayüzünü zenginleştirmek için
   e) Kodların çalışma performansını arttırmak için

8. Grafik tipindeki bir değişkenin içeriğini yenilemek için aşağıdakilerden hangisi kullanılır?
   a) Create
   b) Fill
   c) Delete
   d) New
   e) Clear

9. Bir kontrolün tıklanabilirliğini değiştirmek için aşağıdaki özelliklerden hangisi kullanılır?
   a) Name
   b) Postback
   c) Text
   d) Enabled
   e) ClientID

10. Random tipinde bir değişkene rastgele negatif olmayan bir tamsayı değeri atamak için aşağıdaki metodlardan hangisi kullanılmalıdır?
    a) Move
    b) Create
    c) Next
    d) GetHashCode
    e) NextDouble

Cevap Anahtarı

1.a, 2.c, 3.d, 4.e, 5.a, 6.c, 7.b, 8.e, 9.d, 10.c