Atatürk Üniversitesi
Açıköğretim Fakültesi

Görsel Programlama II

ATATÜRK ÜNİVERSİTESİ
AÇIKÖĞRETİM FAKÜLTESİ

Görsel Programlama II

ISBN: 978-975-442-713-4

ATATÜRK ÜNİVERSİTESİ AÇIKÖĞRETİM FAKÜLTESİ YAYINI

ERZURUM, 2016

İÇİNDEKİLER

Editör

Doç. Dr. YUSUF ZİYA AYIK

Prof. Dr. UĞUR YAVUZ

KARAKTER ÖZELLİKLİ HAZIR METOTLAR

İÇİNDEKİLER

- Giriş
- Karakter Özellikli Metotlar
  - Left, Right,Mid
  - ToLower, ToUpper
  - Insert, Replace, Remove
  - IndexOf
  - Split, Concat
  - Space

HEDEFLER

- Bu üniteyi çalıştıktan sonra; String özelliğe sahip Visual Baisc .NET kütüphanesinde bulunan hazır metotlardan ünitede anlatılanların işlevlerini bilecek ve bu metotları program geliştirirken kullanabileceksiniz.

GÖRSEL PROGRAMLAMA II

Yrd. Doç. Dr. Y. Ziya AYIK

ÜNİTE

1

GİRİŞ

Visual Basic .NET, kendi kütüphanesinde oldukça gelişmiş karakter veya sayısal özelliğe sahip hazır fonksiyon içeren çok sayıda metot bulundurmaktadır. Metotların kullanılması programlamada birçok işlemin hazır olarak, yeniden kod yazılmasına gerek duyulmaksızın gerçekleştirilmesini sağlamaktadır. Bu ünitede Visual Basic .NET ile program geliştirilirken ihtiyaç duyulacak olan bazı metotların genel yapıları verilmekte ve kullanım şekilleri örnekler üzerinde gösterilerek açıklanmaktadır. Tanıtılacak metotlar string özelliğe sahip olup string veriler üzerinde uygulanmaktadır. String sınıfının bu metotları Shared özelliklidirler, yani bütün sınıflardan erişilebilir durumdadırlar.

Metotlarda kullanılacak değişkenler, string özelliğe veya string metotların uygulanabileceği bir tipe sahip olmalı ve kullanılmadan önce tanımlanmalıdırlar.

KARAKTER ÖZELLİKLİ METOTLAR

String özelliğe sahip metotlar string veriler üzerinde uygulanmaktadır.

Karakter (String) özelliğe sahip veri ve değişkenler üzerinde uygulanabilen metotlardır.

Left Metodu

Left() metodu (fonksiyonu) kendisine parametre olarak verilen karakter veya farklı bir tipteki bilginin soldan itibaren istenilen kadarını elde etmek amacıyla kullanılır.

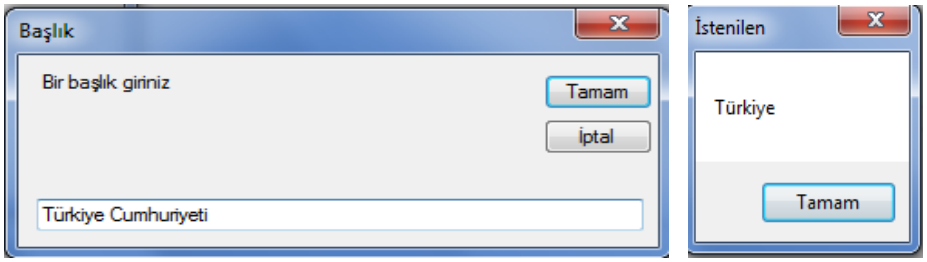
Genel Yazılışı; Left(Değişken veya veri, istenilen karakter sayısı)

Örnek 1

- Klavyeden girilen bir başlığın sol baştan itibaren istenilen kadarını kesip yazdıran program.

```
Private Sub Button1_Click(sender As Object, e As EventArgs)
    Dim baslık, istenilen As String
    baslık = InputBox("Bir başlık giriniz")
    istenilen = Microsoft.VisualBasic.Left(baslık, 7)
    MsgBox(istenilen)
End Sub
```

LEFT METODU



Right Metodu

Right() metodu (fonksiyonu) kendisine parametre olarak verilen karakter veya farklı bir tipteki bilginin sağdan itibaren istenilen kadarını elde etmek amacıyla kullanılır.

Genel Yazılışı; Right (Değişken veya veri, istenilen karakter sayısı)

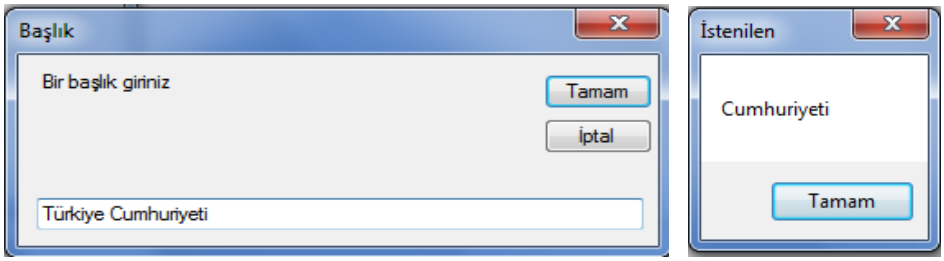
Left metodu bilginin soldan itibaren istenilen kadarını, Right metodu ise sağdan itibaren istenilen kadarını elde etmek amacıyla kullanılır.

Örnek 2

- Klavyeden girilen bir başlığın sağ baştan itibaren istenilen kadarını kesip yazdıran program.

```
Private Sub Button2_Click(sender As Object, e As EventArgs)
    Dim baslık, istenilen As String
    baslık = InputBox("Bir başlık giriniz", "Başlık")
    istenilen = Microsoft.VisualBasic.Right(baslık, 11)
    MsgBox(istenilen, , "İstenilen")
End Sub
```

RIGHT METODU



Mid Metodu

Mid() metodu (fonksiyonu) kendisine parametre olarak verilen karakter veya farklı bir tipteki bilginin istenilen kadarını elde etmek amacıyla kullanılır. Bu metotta 3 parametre kullanılmaktadır. Değişken veya veriden istenilen yerden istenildiği kadar bilgi elde edilebilmektedir.

Genel Yazılışı; Left (Değişken veya veri, başlangıç yeri, istenilen karakter sayısı)

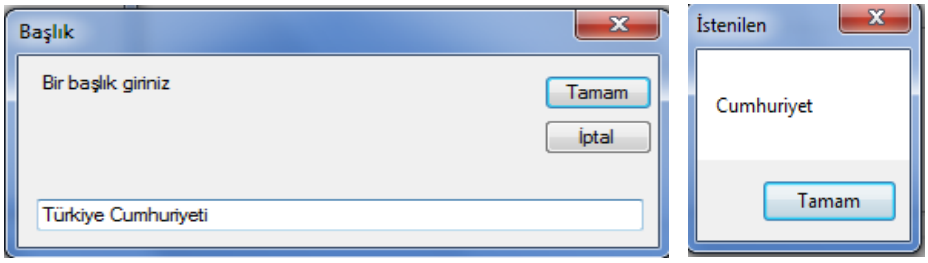
Örnek 3

- Klavyeden girilen bir başlığın istenilen kısmını kesip yazdıran program.

```
Private Sub Button3_Click_1(sender As Object, e As EventArgs)
    Dim baslık, istenilen As String
    baslık = InputBox("Bir başlık giriniz", "Başlık")
    istenilen = Microsoft.VisualBasic.Mid(baslık, 9, 10)
    MsgBox(istenilen, , "İstenilen")
End Sub
```

Mid metodu bilginin belirtilen konumundan istenilen kadarını elde etmek amacıyla kullanılır.

MID METODU



Tolower Metodu

ToLower() metodu (fonksiyonu) parametre olarak verilen karakter tipteki bilginin büyük modda olanlarını küçük moda çevirmektedir.

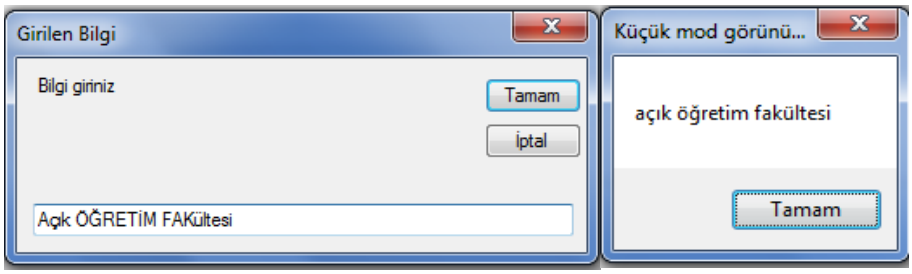
Genel Yazılışı; ToLower(Değişken veya veri)

Örnek 4

- Klavyeden girilen bir başlığı küçük mod görünümünde yazdıran program.

```
Private Sub Button4_Click(sender As Object, e As EventArgs)
    Dim veri, kucukmod As String
    veri = InputBox("Bilgi giriniz", "Girilen Bilgi")
    kucukmod = veri.ToLower
    MsgBox(kucukmod, , "Küçük mod görünümü")
End Sub
```

TOLOWER METODU



Toupper Metodu

ToUpper() metodu (fonksiyonu) parametre olarak verilen karakter tipteki bilginin küçük modda olanlarını büyük moda çevirmektedir.

Genel Yazılışı; ToUpper (Değişken veya veri)

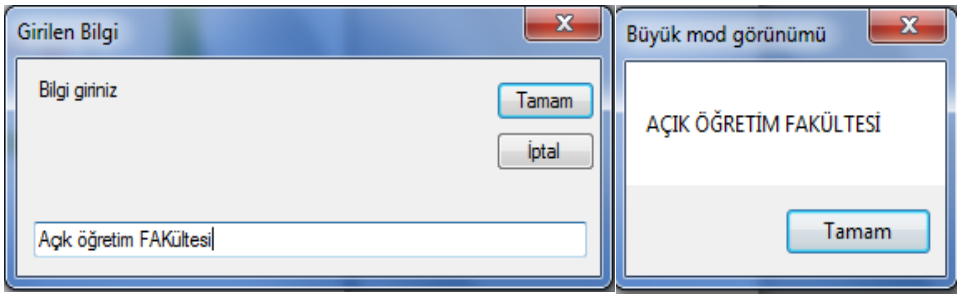
Örnek 5

- Klavyeden girilen bir başlığı büyük mod görünümünde yazdıran program.

```
Private Sub Button5_Click(sender As Object, e As EventArgs)
    Dim veri, buyukmod As String
    veri = InputBox("Bilgi giriniz", "Girilen Bilgi")
    buyukmod = veri.ToUpper
    MsgBox(buyukmod, , "Büyük mod görünümü")
End Sub
```

ToLower metodu bilgiyi büyük moddan küçük moda, ToUpper metodu ise bilgiyi küçük moddan büyük moda çevirmektedir.

TOUPPER METODU



Insert Metodu

String sınıfının Insert metodu, karakter özelliğe sahip bir verinin istenilen yerine başka bir verinin eklenmesi amacıyla kullanılır.

Genel Yazılışı; Değişken.Insert (Başlama sütunu, eklenmek istenen değişken veya veri)

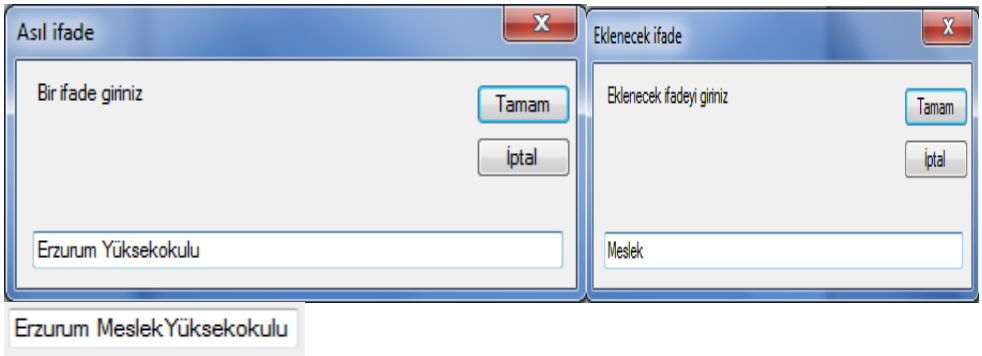
Örnek 6

- Klavyeden girilen bir verinin istenilen bir yerinden başlayarak yeni bir verinin eklenmesi işlemini gerçekleştiren program.

```
Private Sub Button6_Click(sender As Object, e As EventArgs) Handles But
    Dim asılveri, eklenen As String
    asılveri = InputBox("Bir ifade giriniz", "Asıl ifade")
    eklenen = InputBox("Eklenecek ifadeyi giriniz", "Eklenecek ifade")
    TextBox1.Text = asılveri.Insert(8, eklenen)
End Sub
```

Insert metodu ile karakter özelliğe sahip bir verinin istenilen yerine başka bir veri eklenebilir.

INSERT METODU



Replace Metodu

String sınıfının Replace metodu, karakter özelliğe sahip bir verinin içeriğinin tümünü veya istenilen kısmını başka bir veri ile değiştirilmek istendiğinde kullanılır.

Genel Yazılışı; Değişken. Replace (Asıl veri, değiştirilmek istenilen değişken veya veri)

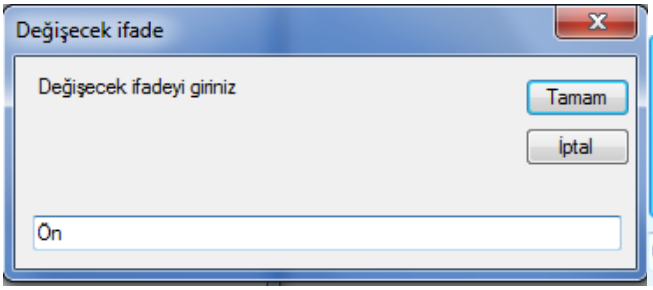
Örnek 7

- Klavyeden girilen bir verinin istenilen bir bölümünü başka bir veriyle değiştirme işlemini gerçekleştiren program.

```
Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7
    Dim asılveri, degisenveri As String
    asılveri = "Yüksek Lisans Dersler"
    TextBox2.Text = asılveri
    degisenveri = InputBox("Değişecek ifadeyi giriniz", "Değişecek ifade")
    TextBox2.Text = asılveri.Replace("Yüksek", degisenveri)
End Sub
```

REPLACE METODU

TextBox'da yazılı olan "Yüksek Lisans Dersler" ifadesindeki "Yüksek" kelimesi "Ön" kelimesiyle değiştirilmiştir. TextBox'da yazılı olan yeni ifade "Ön Lisans Dersler" şeklinde olmuştur.



REPLACE METODU

Ön Lisans Dersler

Replace metodu, verinin içeriğinin tümünü veya istenilen kısmını başka bir veri ile değiştirmek, Remove metodu ise, silmek için kullanılır.

Remove Metodu

String sınıfının Remove metodu, karakter özelliğe sahip bir verinin içeriğinin tümünü veya istenilen kısmını silmek istendiğinde kullanılır.

Genel Yazılışı; Değişken. Remove (Silme başlangıç sütunu, silinecek karakter sayısı)

Örnek 8

•Klavyeden girilen bir verinin istenilen bir bölümünü silme işlemini gerçekleştiren program.

```
Private Sub Button8_Click(sender As Object, e As EventArgs
    Dim asılveri As String
    asılveri = "Açık Öğretim Fakültesi"
    TextBox3.Text = asılveri
    TextBox4.Text = asılveri.Remove(13, 9)
End Sub
```

Indexof Metodu

IndexOf metodu string bilgilerde arama yaparken kullanılmaktadır. Bu metot ile string bilgi içerisinde aranılan bir harfin veya harf grubunun ilk karakterinin konumu belirlenmektedir. Bu metot iki parametreyle birlikte kullanılmaktadır. Birinci parametre aranan bilgiyi, ikincisi ise string bilginin arama yapılacak konumunu belirlemektedir.

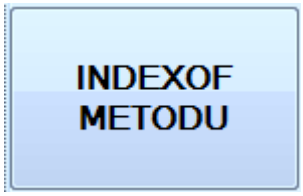
Genel Yazılışı; Değişken.IndexOf (Aranan veri, arama başlangıç konumu)

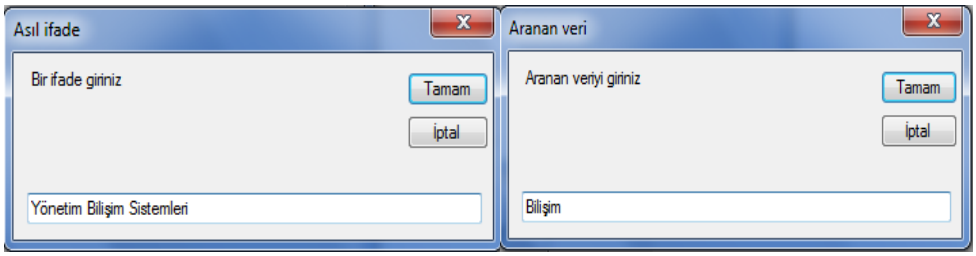
IndexOf metodu string bilgi içerisinde bir veri paçasının aranması amacıyla kullanılabilir.

Örnek 9

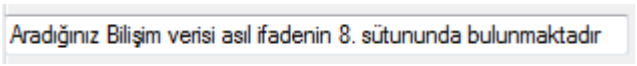
•Klavyeden girilen bir ifade içerisinde bir veri parçasının konumunu bulma işlemini gerçekleştiren program.

```
Private Sub Button9_Click(sender As Object, e As EventArgs) Handles Buttor
    Dim asılveri, aranan As String
    Dim konum As Integer
    asılveri = InputBox("Bir ifade giriniz", "Asıl ifade")
    aranan = InputBox("Aranan veriyi giriniz", "Aranan veri")
    konum = asılveri.IndexOf(aranan, 3)
    TextBox5.Text = ("Aradığınız " & aranan & " verisi asıl ifadenin " _
                     & konum & ". sütununda bulunmaktadır")
End Sub
```





"Yönetim Bilişim Sistemleri" ifadesi içerisinde "Bilişim" verisi ifadenin 3. indis değerli (yani 4. basamak) sütunundan itibaren aranmış ve verinin ilk karakteri, ifadenin 8. indis değerli (yani 9. Basmak) sütununda bulunmuştur.



Split Metodu

String sınıfının Split metodu kullanılarak karakter özelliğe sahip bir ifade istenilen sayıda parçaya ayrılabilmektedir. Bu işlem, ad ve soyad olarak yazılan isimlerin iki farklı alana ayrılması, adres bilgilerinin cadde, sokak, şehir gibi alanlara ayrılması, tarih değerlerinin gün, ay, yıl gibi parçalara ayrılması işlemlerde oldukça işe yarayabilmektedir.

Split metoduyla bir ifade istenilen sayıda bölüme ayrılır.

Split metodunun uygulanabilmesi için string tipinde dizi değişken tanımlanmalıdır. Dizi değişkenin eleman sayısı önceden belirtilmezse, parçalanması düşünülen ifade istenilen sayıda parçaya ayrılabilir.

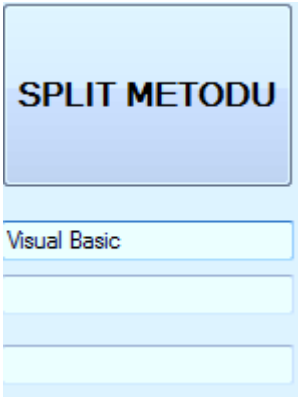
Genel Yazılışı; Split (Karakter özellikli veri, ayırıcı karakter, ayrılacak parça sayısı)

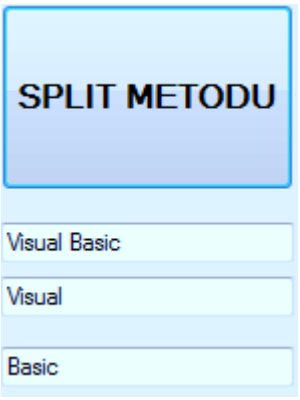
Ayırıcı karakter şeklinde tanımlanan parametre, verinin bölünmesinde dikkate alınacak karakteri ifade etmektedir. Üçüncü parametre olarak kullanılan *ayrılacak parça sayısı* asıl verinin kaç parçaya ayrılacağını belirtmektedir.

Örnek 10

- Bir TextBox nesnesine girilen ifadeyi içerisinde bulunan boşluk karakterini dikkate alarak parçalara ayırma işlemini gerçekleştiren program.

```
Private Sub Button10_Click(sender As Object, e As EventArgs)
    Dim asılveri() As String
    asılveri = Split(TextBox6.Text, " ", 2)
    TextBox7.Text = asılveri(0)
    TextBox8.Text = asılveri(1)
End Sub
```





Bu örnekte üçüncü parametre olarak 2 yerine 3 veya daha farklı bir değer kullanıldığında hata oluşması muhtemeldir. Ayrıca üçüncü parametre olarak 2 kullanılmasına rağmen TextBox nesnesinde birden fazla boşluk karakteri kullanılması durumunda da birinci boşluktan sonraki boşluk karakterleri bölünmede dikkate alınmayacaklardır. Ayrıca TextBox nesnesinde hiç boşluk olmaması durumunda da hata oluşacaktır.

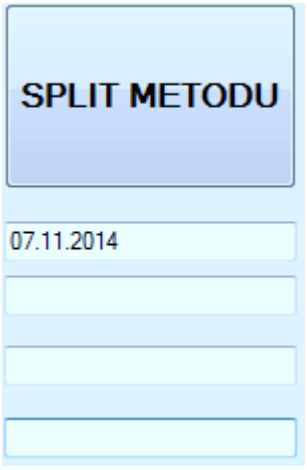
Concat metoduyla farklı alanlarda bulunan veriler birleştirilebilir.

Split metodunun kullanılmasında ayırıcı karakter olarak herhangi bir karakterin kullanılması mümkündür. Aşağıdaki örnekte ayırıcı karakter olarak “.” kullanılmaktadır.

Örnek 11

- Bir TextBox nesnesine girilen tarih değerlerini "." karakteri dikkate alınarak gün, ay, yıl şeklinde parçalara ayırma işlemini gerçekleştiren program.

```
Private Sub Button11_Click(sender As Object, e As EventArgs)
    Dim zaman() As String
    TextBox6.Text = Today()
    zaman = Split(TextBox6.Text, ".", 3)
    TextBox7.Text = zaman(0)
    TextBox8.Text = zaman(1)
    TextBox9.Text = zaman(2)
End Sub
```



Concat Metodu

String sınıfının Concat metodu kullanılarak farklı alanlarda bulunan string veriler birleştirilerek tek bir değişkene aktarılabilir veya tek bir nesnede görüntülenebilir. Birleştirilecek string bilgiler metodun parametreleri olarak kullanılmaktadır.

Genel Yazılışı; String.Concat (Veri1,veri2,veri3, …)

Space metodu ile Concat metodu birbirine zıt işlev gerçekleştirirler.

Örnek 12

- Klavyeden girilen bilgileri ayrı ayrı ve birleşik olarak TextBox nesnesinde görüntüleme işlemini gerçekleştiren program.

```
Private Sub Button12_Click(sender As Object, e As EventArgs) Handle
    Dim unvan, isim, isyeri As String
    unvan = InputBox("Ünvanınızı girin")
    isim = InputBox("İsminizi girin")
    isyeri = InputBox("Çalıştığınız yeri girin")
    TextBox10.Text = unvan
    TextBox11.Text = isim
    TextBox12.Text = isyeri
    TextBox13.Text = String.Concat(unvan, " ", isim, " ", isyeri)
End Sub
```

Space Metodu

Space metodu string veri grubu içerisinde istenilen kadar boşluk karakteri oluşturmak amacıyla kullanılır. Oluşturulmak istenilen boşluk sayısı Space metodunun parametresi olarak verilir.

Genel Yazılışı; Space (Boşluk karakteri sayısı)

Örnek 13

•Klavyeden girilen bilgilerin arasına 10 karakter boşluk koyulmasını sağlayan program.

```
Private Sub Button13_Click(sender As Object, e As EventArgs) Handles
    Dim ad, soyad, meslek As String
    ad = InputBox("Adınızı girin")
    soyad = InputBox("Soyadınızı girin")
    meslek = InputBox("Mesleğinizi girin")
    TextBox11.Text = ad
    TextBox12.Text = soyad
    TextBox13.Text = meslek
    TextBox14.Text = (ad & Space(10) & soyad & Space(10) & meslek)
End Sub
```

Space metodu istenilen kadar boşluk karakteri olusturur.

Özet

- Metotların kullanımı programlamada kolaylık, esneklik, zamandan tasarruf gibi avantajlar sağlar.
- String özelliğe sahip metotlar String veriler üzerinde uygulanırlar.
- Left, Right, Mid, ToLower, ToUpper, Insert, Replace, Remove, IndexOf, Split, Concat ve Space metotları Visual Basic .NET kütüphanesinde hazır olarak bulunan ve string karakter veriler üzerinde uygulanabilen önemli fonksiyonlardandır.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. "Sistem Analizi" ifadesinin sadece "Ana" kısmını dikkate alıp bir değişkene atamak için aşağıdaki fonksiyonlardan hangisi kullanılmalıdır?

   a) Mid
   b) Left
   c) Insert
   d) Space
   e) Right

2.
```
Dim baslık As String
baslık = "print fax copy"
TextBox1.Text = Microsoft.VisualBasic.Left(baslık, 7)
```

   Yukarıda verilen kod çalıştırıldığında aşağıdaki değerlerden hangisi elde edilir?

   a) print
   b) print fax
   c) print fax copy
   d) print f
   e) fax copy

3.
```
Dim veri As String
veri = "Print FAX COpy"
TextBox1.Text = veri.ToLower
```

   Yukarıda verilen kod çalıştırıldığında aşağıdaki değerlerden hangisi elde edilir?

   a) PRINT FAX COPY
   b) Print Fax Copy
   c) print fax copy
   d) print fAX COpy
   e) PRINT fax copy

4.
```
Dim asılveri, eklenecek As String
asılveri = "Kara Bölgesi"
eklenecek = "deniz"
```

Yukarıdaki kodlamayı dikkate alarak, “Karadeniz Bölgesi” ifadesinin elde edilebilmesi için kodlamanın son satırına yazılması gereken fonksiyon kullanımını gösteren kod hangisidir?

a) TextBox1.Text=asılveri.IndexOf(eklenecek,2)
b) TextBox1.Text=asılveri.Insert(3, eklenecek)
c) TextBox1.Text=asılveri.Insert(eklenecek,4)
d) TextBox1.Text=asılveri.InputBox(4, eklenecek)
e) TextBox1.Text=asılveri.Insert(4, eklenecek)

5.
```
Dim asılveri, degisenveri As String
asılveri = "Birlikte on yıl"
TextBox2.Text = asılveri
degisenveri = "mutlu"
TextBox2.Text = asılveri.Replace("on", degisenveri)
```

Yukarıda verilen kod çalıştırıldığında TextBox2 nesnesine ne yazdırılır?

a) on mutlu yıl
b) mutlu on yıl
c) on on yıl
d) Birlikte mutlu yıl
e) Birlikte Birlikte yıl

6.
```
Dim asılveri, aranan As String
Dim konum As Integer
asılveri = "İngilizce kursu"
aranan = "kurs"
konum = asılveri.IndexOf(aranan, 0)
TextBox5.Text = ("Aradığınız veri" & konum & ". sütundadır")
```

Yukarıda verilen kod çalıştırıldığında TextBox2 nesnesine ne yazdırılır?

a) Aradığınız veri konum sütundadır
b) Aradığınız veri 10. sütundadır
c) Aradığınız veri 8. sütundadır
d) Aradığınız veri 14. sütundadır
e) 11. Sütun

7.
```
Dim asılveri() As String
asılveri = Split(TextBox6.Text, " ", 2)
TextBox7.Text = asılveri(0)
TextBox8.Text = asılveri(1)
```

Yukarıda verilen kodu dikkate alarak ve TextBox6'ya "AR-GE Çalışmaları" ifadesinin atandığını düşünerek TextBox7'ye atanan aşağıdaki ifadelerden hangisi doğrudur?

a) AR-GE Çalışmaları
b) AR
c) GE
d) AR-GE
e) Çalışmaları

8.
```
Dim x, y, z As String
x = "Ahmet"
y = "Ayşe"
z = "Okul"
TextBox13.Text = String.Concat(y, x)
```

Yukarıda verilen kodu dikkate alarak ve TextBox13'e atanan aşağıdaki ifadelerden hangisi doğrudur?

a) AyşeAhmet
b) AhmetAyşeOkul
c) AhmetAyşe
d) OkulAyşeAhmet
e) AyşeAhmetOkul

9.
```
Dim x, y, z As String
x = "İnternet"
y = "Yönetim"
z = "Hız"
TextBox14.Text = (x & Space(5) & y & Space(10) & z)
```

Yukarıda verilen kod dikkate alındığında aşağıdaki ifadelerden hangisi doğru olur?

a) Yönetim ile Hız arasında hiç boşluk yoktur.
b) İnternet ile Hız arasında 10 boşluk vardır.
c) Yönetim ile Hız arasında 15 boşluk vardır.
d) İnternet ile Yönetim arasında hiç boşluk yoktur.
e) İnternet ile Yönetim arasında 5 boşluk vardır.

10.
```
Dim asılveri As String
asılveri = "TRT Haber Merkezi"
TextBox3.Text = asılveri
TextBox4.Text = asılveri.Remove(4, 6)
```

Yukarıda verilen kod dikkate alındığında ve TextBox4'e atanan aşağıdaki ifadelerden hangisi doğru olur?

a) Haber Merkezi
b) TRT Haber
c) TRT Merkezi
d) TRT Haber Merkezi
e) Haber Merkezi TRT

Cevap Anahtarı

1.A, 2.D, 3.C, 4.E, 5.D, 6.B, 7.D, 8.A, 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

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

SAYISAL ÖZELLİKLİ HAZIR METOTLAR VE ZAMAN

İÇİNDEKİLER

- Giriş
- Sayısal Özellikli Metotlar
  - Sign,Int, Round
  - Sqrt
  - Random
- Zaman Fonksiyonları
  - Today, Now
  - Day, Month, Year, DateSerial
  - AddDays, AddMonths, AddYears
  - Hour, Minute, Second, TimeSerial
  - AddHours, AddMinutes, AddSeconds
  - DateDiff

HEDEFLER

- Bu üniteyi çalıştıktan sonra; Sayısal ve zaman özelliklli, Visual Baisc .NET kütüphanesinde bulunan hazır metotlardan ünitede anlatılanların işlevlerini bilecek ve bu metotları program geliştirirken kullanabileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Yrd. Doç. Dr. Y. Ziya AYIK

ÜNİTE

2

GİRİŞ

Birinci ünitede de bahsedildiği gibi Visual Basic.NET, kendi kütüphanesinde oldukça gelişmiş karakter veya sayısal özelliğe sahip hazır fonksiyon içeren çok sayıda metot bulundurmaktadır. Visual Basic.NET kütüphanesinde ayrıca tarih ve zaman bilgileri üzerinde işlemler gerçekleştirebilen yine çok sayıda zaman fonksiyonu da bulunmaktadır

Bu ünitede Visual Basic.NET ile program geliştirilirken ihtiyaç duyulacak olan bazı sayısal özellikli metotların ve zaman fonksiyonlarının genel yapıları verilmekte ve kullanım şekilleri örnekler üzerinde gösterilerek açıklanmaktadır. Sayısal metotlar sayısal özelliğe sahip veriler üzerinde, tarih ve zaman fonksiyonları ise zaman özelliğine sahip veriler üzerinde uygulanmaktadır.

Metotlarda kullanılacak değişkenler, sayısal veya zaman tipi özelliğe sahip olmalı ve kullanılmadan önce tanımlanmalıdırlar.

SAYISAL ÖZELLİKLİ METOTLAR

Sayısal özelliğe sahip veri ve değişkenler üzerinde uygulanabilen metotlardır.

Sign

Sayı pozitif ise Sign metodu 1, sayı negatifse -1 döndürür.

Sign metodu, sayısal özelliğe sahip bir veri veya değişkenin pozitif veya negatif olduğunu öğrenmek amacıyla kullanılır.

Genel Yazılışı; Math.Sign (Sayısal değişken veya veri)

Parametre olarak kullanılan veri veya değişken pozitif değere sahipse Sign metodu geriye 1 değerini, negatif değere sahipse -1 değerini döndürür. Parametre 0 değerine sahipse metot geriye 0 değerini döndürür.

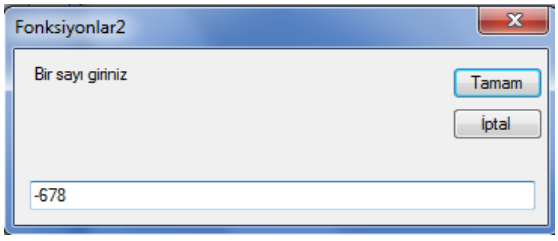
Örnek 1

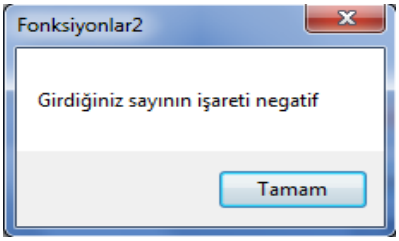
- Klavyeden girilen bir sayının negatig, pozitif veya sıfır olduğunu tespit eden program.

```
Private Sub Button1_Click(sender As Object, e As EventA
    Dim sayı, sonuc As Integer
    sayı = InputBox("Bir sayı giriniz")
    sonuc = Math.Sign(sayı)
    If sonuc = 1 Then
        MsgBox("Girdiğiniz sayının işareti pozitif")
    ElseIf sonuc = -1 Then
        MsgBox("Girdiğiniz sayının işareti negatif")
    Else
        MsgBox("Girdiğiniz sayının sıfır")
    End If
```

SIGN METODU

Klavyeden girilen sayı -678 olduğu için sayının negatif olduğu Sign fonksiyonu kullanılarak tespit edilmiştir. Diğer değer girişlerini deneyiniz.





Int

Kesirli bir sayı INT metodu kullanılarak tam sayıya dönüstürülebilir.

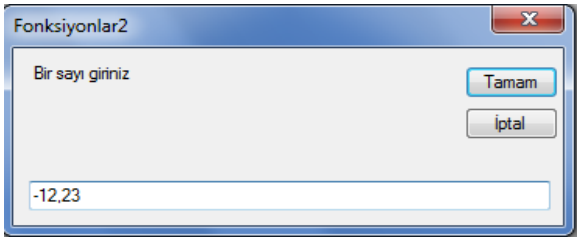
Int metodu ondalık kesir içeren bir sayıyı tam sayıya çevirir. Pozitif kesirli sayılarda kesir kısmı 0,5 den küçük ise küçük sayıya, 0,5 den büyük ise büyük sayıya yuvarlar. Negatif kesirli sayılarda ise bu işlemin tersini gerçekleştirir.

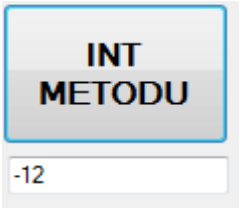
Genel Yazılışı; Int (Kesirli sayısal değişken veya veri)

Örnek 2

- Klavyeden girilen kesirli bir sayının Int ve Fix fonksiyonları uygulanarak Tam sayı şeklinde elde edilmesi işlemini gerçekleştiren program.

```
Private Sub Button2_Click(sender As Object,
    Dim kesirli, tam As Integer
    kesirli = InputBox("Bir sayı giriniz")
    tam = Int(kesirli)
    TextBox1.Text = tam
End Sub
```





Girilin sayı -12,23 sayısında kesir kısmı 0,5 den küçük olduğu için ve sayı negatif olduğu için büyük sayı olan -12 elde edilmiştir. (-12, -13'den büyüktür).

Round

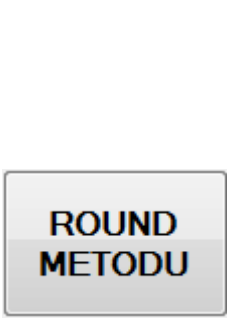
Round metodu kesirli sayıların kesir kısmının gereğinden fazla olduğu durumlarda, kesir kısmının yuvarlatılarak sayının daha anlamlı ve kullanılabilir hâle getirilmesinde kullanılır.

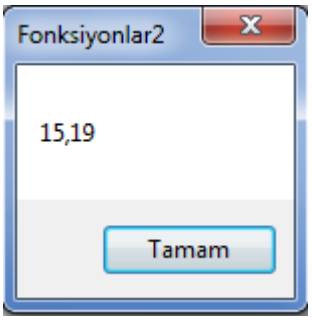
Genel Yazılışı; Math.Round (Kesirli sayısal değişken veya veri, kesirli basmak sayısı)

Örnek 3

- Klavyeden girilen kesirli bir sayının kesir kısmını 2 basamağa yuvarlama işlemini gerçekleştiren program.

```
Private Sub Button3_Click(sender As Object,
    Dim x As Double
    x = Math.Round(15.189, 2)
    MsgBox(x)
End Sub
```





Kesirli sayısal "15.189" sayısı, Round metodu kullanılırken kesirli basamak sayısı 2 olarak verildiğinden 15.19 olarak hesaplanmıştır.

Sqrt

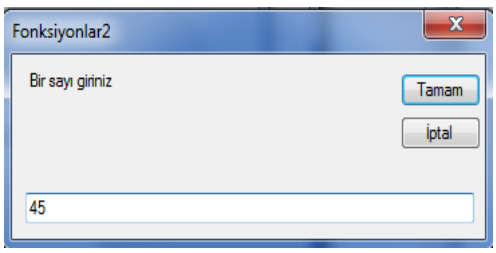
Sqrt metodu pozitif bir sayısal verinin karekökünü hesaplamak amacıyla kullanılır.

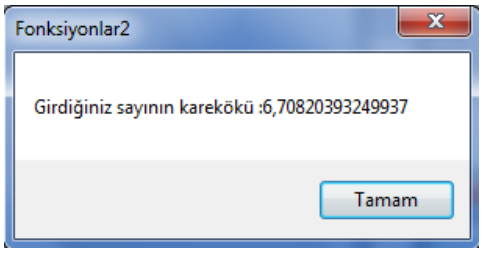
Genel Yazılışı; Math.Sqrt (Sayısal değişken veya veri)

Örnek 4

- Klavyeden girilen bir sayının karekökünü hesaplama işlemini gerçekleştiren program.

```
Private Sub Button4_Click(sender As Object, e As EventArg
    Dim x As Double
    x = InputBox("Bir sayı giriniz")
    If x < 0 Then
        MsgBox("Negatif sayı girilemez")
    Else
        x = Math.Sqrt(x)
        MsgBox("Girdiğiniz sayının karekökü :" & x)
    End If
End Sub
```





Girilen sayının negatif olması durumunda karekök hesaplama işlemi yapılmayacaktır.

Random metodu ile sisteme istenilen aralıkta rastgele sayılar ürettirilebilir.

Random

Random metodu rastgele sayı üretmek amacıyla kullanılır. Üretilmek istenilen sayının istenilen aralıkta olması için Random metodunun Next özelliğine parametre olarak istenilen değer girilmelidir.

Random tipinde bir değişken tanımlanmalı ve Next özelliği bu değişkenle birlikte kullanılmalıdır.

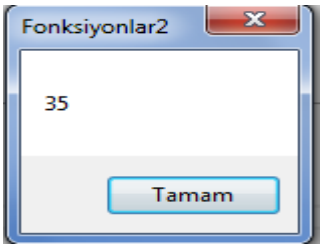
Genel Yazılışı; Dim Değişkenadı As New Random

Değişkenadı.Next(istenilen sayı)

Örnek 5

• 0 ile 50 sayıları arasında rastgele bir sayının bilgisayar tarafından üretilerek değişkene aktarılması işlemini gerçekleştiren program.

```
Private Sub Button5_Click(sender As Object, e
    Dim x As Integer
    Dim rasggelesayı As New Random
    x = rasggelesayı.Next(50)
    MsgBox(x)
End Sub
```



Program çalıştırıldığında bilgisayar tarafından 0 ile 50 arasında rastgele bir sayı üretilmektedir. Bu örnekte 35 sayısı üretilmiştir. Denemelerinizde her seferinde farklı bir sayının tutulabileceğini göreceksiniz.

Today, sistemin tarihini, Now sistemin zamanını verir.

ZAMAN FONKSİYONLARI

Tarih ve zaman bilgileri üzerinde gerçekleştirilen fonksiyonlardır.

Today, Now

Tarih ve zaman bilgileri DateTime tipindeki değişkenlerde saklanmaktadır. DateTime yapısının Today özelliği günün tarihini, Now özelliği ise anlık zamanı saat olarak vermektedir.

Genel Yazılışı; Dim Değişken As DateTime

Değişken.Today ve Değişken.Now

Örnek 6

•Günün tarihini ve zamanı kısa ve uzun modlarda TextBoxlarda görüntüleme işlemini gerçekleştiren program.

```
Private Sub Button6_Click(sender As Object, e As EventArgs)
    Dim tt As DateTime
    Dim zz As DateTime
    tt = DateTime.Today
    zz = DateTime.Now
    TextBox1.Text = tt.ToShortDateString
    TextBox2.Text = tt.ToLongDateString
    TextBox3.Text = zz.ToShortTimeString
    TextBox4.Text = zz.ToLongTimeString
    TextBox5.Text = Now
End Sub
```

Day sistemin gün bilgisini içerir.

Day, Month, Year

Sistem tarihinin gün bilgisini öğrenmek için Day, ay bilgisini öğrenmek için Month ve yıl bilgisini öğrenmek için Year metotları kullanılmaktadır.

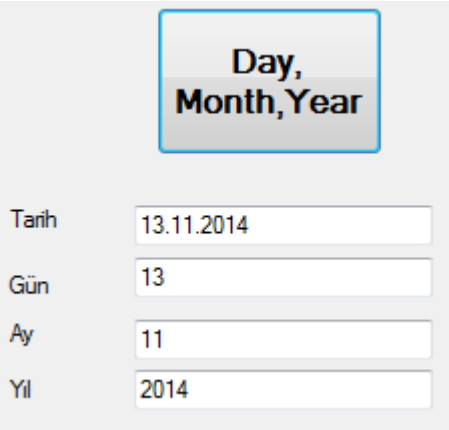
Genel Yazılışı; Dim Değişken As DateTime

Değişken.Day, Değişken.Month, Değişken.Year

Örnek 7

•Sistemin gün, ay ve yıl bilgilerini görüntüleme işlemini gerçekleştiren program.

```
Private Sub Button7_Click(sender As Object, e
    Dim tt As DateTime = Today
    TextBox1.Text = tt
    TextBox2.Text = tt.Day
    TextBox3.Text = tt.Month
    TextBox4.Text = tt.Year
End Sub
```



DateSerial

DateSerial metodu, yıl, ay ve gün bilgilerinden faydalanılarak tarih bilgisi elde etmek amacıyla kullanılır. Bu metot yıl, ay ve gün bilgilerini parametre olarak alarak geriye tarih bilgisi döndürmektedir.

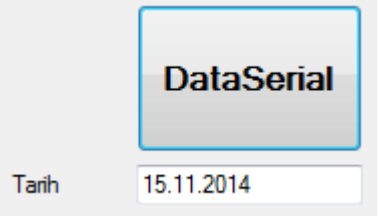
Genel Yazılışı; DateSerial (Yıl,Ay,Gün)

DateSerial yıl, ay ve gün bilgilerini kullanarak tarih bilgisi oluşturur.

Örnek 8

•Sistemin gün, ay ve yıl bilgilerini kullanarak tarih bilgisi elde etme işlemini gerçekleştiren program.

```
Private Sub Button8_Click_1(sender As Object, e
    Dim yıl, ay, gun As Short
    yıl = 2014
    ay = 11
    gun = 15
    TextBox1.Text = DateSerial(yıl, ay, gun)
End Sub
```



AddDays, AddMonths, AdYears

Geçerli tarih bilgisine gün, ay veya yıl eklemek veya çıkarmak için AddDays, AddMonths ve AddYears metotları kullanılmaktadır. Bu metot ile eklenilmek veya çıkarılmak istenilen gün, ay veya yıl metodun sağında parantez içerisinde belirtilmelidir.

Genel Yazılışı; Dim Değişken As DateTime

Değişken.AddDays (günsayısı)

Değişken.AddMonths (ay sayısı)

Değişken.AddYears (yıl sayısı)

Örnek 9

•Sistem tarihine gün, ay veya yıl ekleme veya çıkarma işlemini gerçekleştiren program.

```
Private Sub Button9_Click(sender As Object, e A
    Dim tt As DateTime = Today
    TextBox1.Text = tt
    TextBox2.Text = tt.AddDays(5)
    TextBox3.Text = tt.AddMonths(-3)
    TextBox4.Text = tt.AddYears(1)
End Sub
```

Tarih bilgilerine gün, ay ve yıl eklemek veya çıkarmak için AddDays, AddMonths ve AddYears metotları kullanılır.

Hour, Minute, Second

Sistem zamanının saat bilgisini öğrenmek için Hour, dakika bilgisini öğrenmek için Minute ve saniye bilgisini öğrenmek için Second metotları kullanılmaktadır.

Genel Yazılışı; Dim Değişken As DateTime=Now

TimeValue (Değişken)

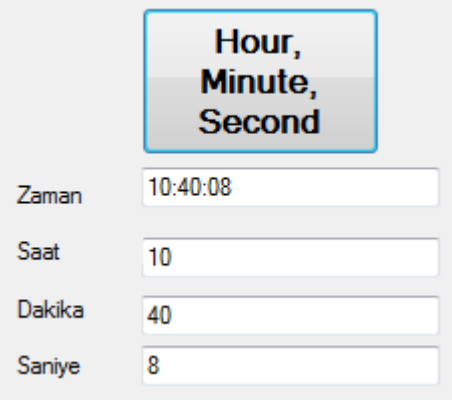
Hour (Değişken),Minute (Değişken), Second (Değişken)

Hour, sistemin saat bilgisini içerir.

Örnek 10

•Sistem saat, dakika ve saniye bilgilerini TextBoxlarda görüntüleme işlemini gerçekleştiren program.

```
Private Sub Button10_Click(sender As Object, e
    Dim zz As DateTime = Now
    TextBox1.Text = TimeValue(zz)
    TextBox2.Text = Hour(zz)
    TextBox3.Text = Minute(zz)
    TextBox4.Text = Second(zz)
End Sub
```



TimeSerial

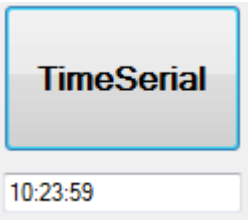
TimeSerial metodu, saat, dakika ve saniye bilgilerinden faydalanılarak zaman bilgisi elde etmek amacıyla kullanılır. Bu metot saat, dakika ve saniye bilgilerini parametre olarak alarak geriye zaman bilgisi göndermektedir.

Genel Yazılışı; TimeSerial (“saat”, “dakika”, ”saniye”)

Örnek 11

•Saat, dakika ve saniye bilgilerini kullanarak zaman bilgisi elde etme işlemini gerçekleştiren program.

```
Private Sub Button11_Click(sender As Object, e As E
    TextBox1.Text = TimeSerial("10", "23", "59")
End Sub
```



AddHours, AddMinutes, AddSeconds

Geçerli zaman bilgisine saat, dakika veya saniye eklemek veya çıkarmak için AddHours, AddMinutes ve AddSeconds metotları kullanılmaktadır. Bu metot ile

eklenmek veya çıkarılmak istenilen saat, dakika veya saniye metodun sağında parantez içerisinde belirtilmelidir.

Genel Yazılışı; Dim Değişken As DateTime=TimeofDay

TimeValue (Değişken)

Değişken.AddHours (saat sayısı)

Değişken.AddMinutes (dakika sayısı)

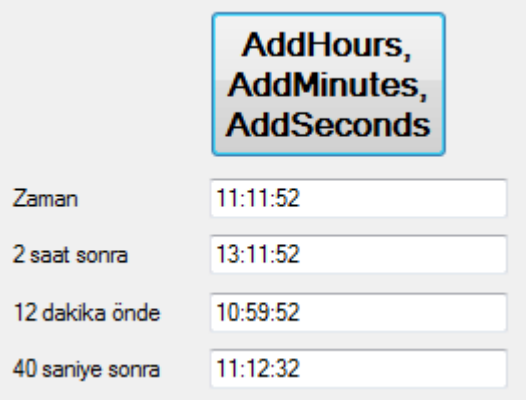
Değişken.AddSeconds (sayiye sayısı)

Zaman bilgilerine saat, dakika ve saniye eklemek veya çıkarmak için AddHours, AddMinutes ve AddSeconds metotları kullanılır.

Örnek 12

- Sistem zamanına saat, dakika ya da saniye ekleme veya çıkarma işlemini gerçekleştiren program.

```
Private Sub Button12_Click(sender As Object, e
    Dim zz As DateTime = TimeOfDay
    TextBox1.Text = TimeValue(zz)
    TextBox2.Text = zz.AddHours(2)
    TextBox3.Text = zz.AddMinutes(-12)
    TextBox4.Text = zz.AddSeconds(40)
End Sub
```



DateDiff

DateDiff metodu iki tarih arasındaki gün, ay, yıl veya iki zaman arasındaki saat, dakika ve saniye farklarının hesaplanması amacıyla kullanılır. Bu metodun kullanımında farkı hesaplanacak zaman türü belirtilmelidir.

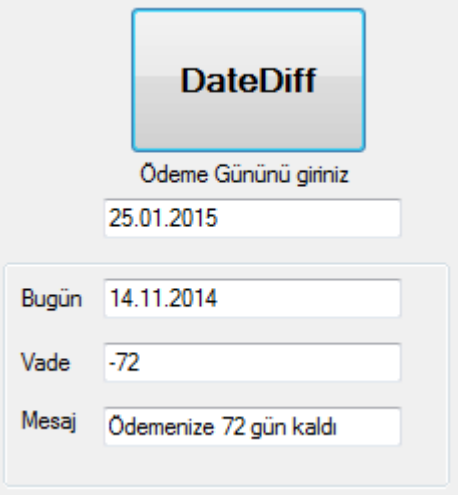
Genel Yazılışı; DateDiff (Zaman türü, son zaman, başlangıç zamanı)

Örnek 13

•İki tarih arasında geçen gün farkını hesaplama işlemini gerçekleştiren program.

DateDiff, iki tarih veya zaman arasında geçen süreyi hesaplar.

```
Private Sub Button13_Click(sender As Object, e As EventArgs) Handles
    Dim tt As DateTime = Today
    Dim odemegunu As DateTime
    Dim durum, durum1 As Double
    odemegunu = TextBox1.Text
    TextBox2.Text = tt
    TextBox3.Text = DateDiff(DateInterval.Day, odemegunu, tt, )
    durum = TextBox3.Text
    durum1 = Math.Abs(durum)
    If durum < 0 Then
        TextBox4.Text = ("Ödemenize " & durum1 & " gün kaldı")
    ElseIf durum > 0 Then

        TextBox4.Text = ("Ödemeniz " & durum1 & " gün geçti")
    Else
        TextBox4.Text = ("Ödemeniz bu gün")
    End If
End Sub
```



Özet

- Metotlar veya fonksiyonlar, programlamada kod tekrarının önlenmesinde ve program yazımında kolaylık sağlar.
- Sayısal özelliğe sahip metotlar sayısal veriler üzerinde, tarih ve zaman özelliğine sahip fonksiyonlar ise zaman verileri üzerinde uygulanırlar.
- Sign, Int, Round, Sqrt, Random Visual Basic .NET kütüphanesinde hazır olarak bulunan, sayısal özellikli metotlar iken; Today, Now, Day, Month, Year,DateSerial, AddDays, AddMonths, AddYears, Hour,Minute,Second,TimeSerial, AddHours ,AddMinutes, AddSeconds ve DateDiff ise tarih ve zaman özellikli fonksiyonlardır.

DEĞERLENDİRME SORULARI

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

1.
```
Dim sonuc, x As Integer
x = -2
sonuc = Math.Sign(x)
MsgBox(sonuc)
```

Verilen program kodu çalıştırıldığında aşağıdaki değerlerden hangisi elde edilir?

a) -2
b) 2
c) 0
d) -1
e) 1

2.
```
Dim x As Double
x = Math.Round(21.3429, 3)
MsgBox(x)
```

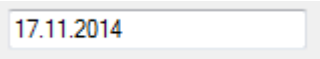
Verilen program kodu çalıştırıldığında aşağıdaki değerlerden hangisi elde edilir?

a) 21,34
b) -21,3429
c) 21,3429
d) -21,343
e) 21,343

3.
```
Dim x As Integer
Dim sayı As New Random
x = sayı.Next(10)
MsgBox(x)
```

Yukarıda verilen program kodu çalıştırıldığında sonuçla ilgili olarak aşağıdaki ifadelerden hangisi doğrudur?

a) 10 tane tamsayı üretilir.
b) 0'dan 10'a kadar olan tamsayılardan bir tanesi üretilir.
c) 1'den 10'a kadar olan tamsayıların hepsi yazdırılır.
d) X değişkenine 10 atanır.
e) sayı değişkenine 10 atanır.

4. 

Yukarıda verildiği gibi günün tarihini kısa modda görüntüleyen kodlama aşağıdakilerden hangisinde doğru gösterilmiştir?

a) TextBox1.Text = DateTime.Today.ToShortDateString
b) TextBox1.Text = DateTime.Today.ToLongDateString
c) TextBox1.Text = DateTime.Today.ToShortTimeString
d) TextBox1.Text = DateTime.Today.ToLongTimeString
e) TextBox1.Text = DateTime.Today.Now

5.
```
Dim yıl, ay, gun As Short
yıl = 2014
ay = 9
gun = 24
TextBox1.Text = DateSerial(yıl, ay, gun)
```

Yukarıda verilen program kodu çalıştırıldığında aşağıdaki değerlerden hangisi elde edilir?

a) 24.2014.09
b) 2014.09.24
c) 24.09.2014
d) 2014
e) 24.09

6.
```
Dim tt As DateTime = "26.11.2014"
TextBox1.Text = tt
TextBox2.Text = tt.AddDays(3)
```

Yukarıda verilen program kodu çalıştırıldığında aşağıdaki değerlerden hangisi elde edilir?

a) 26.11.2014
b) 2014.29.11
c) 26.12.2014
d) 29.11.2014
e) 26.11.2017

7.
```
Dim zz As DateTime = Now
TextBox1.Text = Minute(zz)
```

Yukarıda verilen program kodu çalıştırıldığında TextBox1 nesnesinde görüntülenecek doğru sonuç hangisidir?

a) Şu andaki saati dakikayı ve saniyeyi göstermektedir.
b) Bu günün tarihini göstermektedir.
c) Şu andaki saati göstermektedir.
d) Şu andaki dakikayı göstermektedir.
e) Şu andaki saniyeyi göstermektedir.

8.
```
Dim zz As DateTime = "08:15:12"
TextBox1.Text = TimeValue(zz)
TextBox2.Text = zz.AddSeconds(10)
```
Yukarıda verilen program kodu çalıştırıldığında TextBox2 nesnesinde görüntülenecek doğru sonuç hangisidir?

a) 18:15:12
b) 08:25:12
c) 08:15:22
d) 08:25:22
e) 18:25:22

9. Bir zaman ifadesinde, dakika değerini belirlemek için kullanılan fonksiyon hangisinde doğru olarak yazılmıştır?

a) Hour
b) Minute
c) Second
d) Hours
e) Minutes

10.
```
Dim tt As DateTime = "26.12.2014"
Dim son As DateTime = "26.11.2014"
TextBox1.Text = DateDiff(DateInterval.Day, son, tt, )
```
Yukarıda verilen program kodu çalıştırıldığında TextBox1 nesnesinde görüntülenecek doğru sonuç hangisidir?

a) 1
b) 11
c) 2014
d) 12
e) 30

Cevap Anahtarı

1.D, 2.E, 3.B, 4.A, 5.C, 6.D, 7.D, 8.C, 9.B, 10.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

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

METOT HAZIRLAMAK (PROSEDÜR VE FONKSİYONLAR)

İÇİNDEKİLER

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
- Module oluşturmayı ve kaldırmayı öğrenebilecek,
- Procedure ve fonksiyon kullanmanın avantajlarını tanıyabilecek,
- Procedure (Sub) ve function tiplerinde metot hazırlamayı yapabilecek,
- Metodlardan çıkış yöntemlerini öğrene bileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AOF

GÖRSEL PROGRAMLAMA II

Yrd. Doç. Dr. Y. Ziya AYIK

ÜNİTE

3

GİRİŞ

Birçok işlemin gerçekleştirildiği uzun programlar hazırlandığında aynı değişken veya fonksiyonları kullanan farklı formların veya diğer nesnelerin olması mümkündür. Ancak değişkenler normalde yerel tanımlıdırlar. Yani tanımlandıkları sınıf veya nesne içerisinde kullanılabilirler. Değişkenleri ve fonksiyonları bir proje içinde tüm formlarda veya nesnelerde paylaşmak için, tanımlamaların projede oluşturulacak bir ya da birden çok standart modülde yapılması doğru olacaktır. Bir standart modül dosya uzantısı.vb olan ve programın her tarafında kullanılabilen değişken ve fonksiyonlar içeren özel bir dosyadır. Modüller, Solution Explorer penceresinde ayrı ayrı listelenerek görüntülenirler.

Nesneye yönelik programcılıkta Prosedür ve Fonksiyonların ikisine birden Metot denilmektedir.

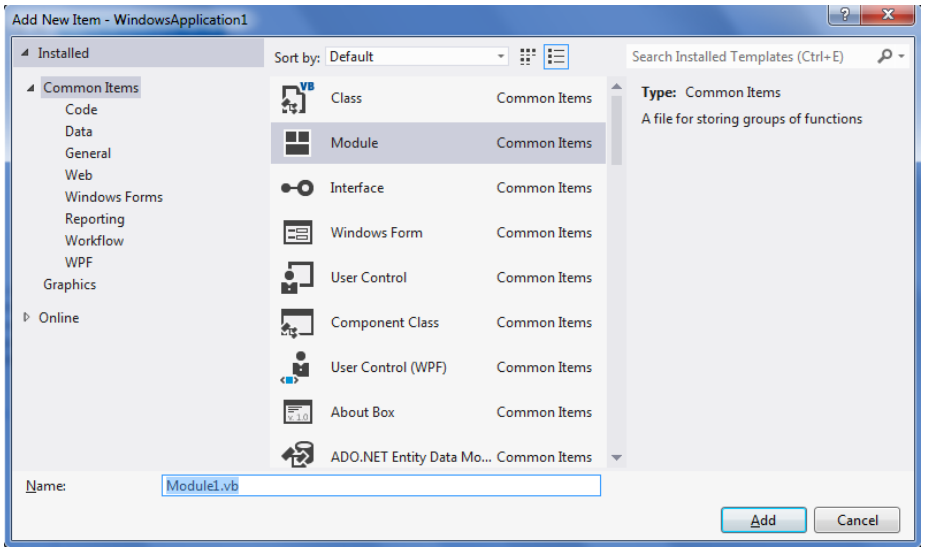
Programlamada tekrar edilmesi gerekebilecek veya daha sonra başka bir programda kullanılabileceği düşünülen kod parçalarının Metot halinde geliştirilmeleri modüller programlama mantığının gereğidir. Bu nedenle bir kısım kodlar Prosedür veya Fonksiyon diye iki grup şeklinde ayrıştırılan Metotlar olarak hazırlanabilirler. Nesneye yönelik programcılıkta Prosedür ve Fonksiyonların ikisine birden Metot denilmektedir. Visual Basic.NET programlama dilinde değer göndermeyen metotlar Procedure(Sub), geriye değer gönderen metotlar ise Function deyimi ile tanımlanmaktadırlar.

Function ve Sub metotlar, modüller içerisinde kullanılmak suretiyle daha etkin ve esnek bir programlama mantığının oluşumunu sağlarlar.

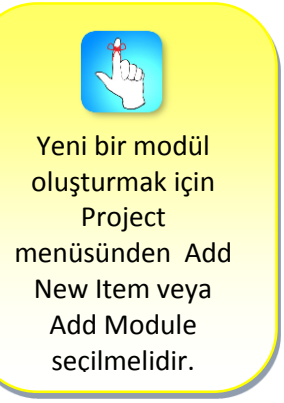
MODÜL OLUŞTURMAK

.NET uyumlu Visual Basic'te kodlar Class ve Structure'lerden ayrı olarak program yazımında birçok avantajı sağlamak amacıyla Modüllere yazılabilmektedir. Hatta Visual Studio ile konsol uygulaması hazırlandığında Main() metodu Class yerine bir Modüle yerleştirilmektedir. Modüllerin en önemli özellikleri Shared olmalarıdır. Yani Private tanımlanmadıkları sürece bütün sınıflar tarafından başka hiçbir tanımlama yapmadan kullanılabilmeleridir.

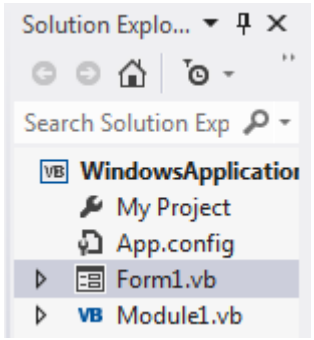
Bir program içerinde modül oluşturmak için PROJECT menüsünden Add Module seçilmeli veya yine PROJECT menüsünden Add New Item seçildikten sonra Module tıklanmalıdır. Modül şablonunun seçilmesini sağlayan iletişim kutusu açıldıktan sonra Modül adının onaylanması veya sizden yeni bir modül adı girilmesi istenecektir. Bu işlemden sonra yeni bir module Code Editörü içerisinde görüntülenecektir.



Modül adı aynı zamanda Solution Explorer penceresinde de listelenecektir. Bir programdaki ilk standart modül varsayılan olarak Module1.vb şeklinde isimlendirilir. Ancak bu ismin programcı tarafından değiştirilmesi mümkündür.



Yeni bir modül oluşturmak için Project menüsünden Add New Item veya Add Module seçilmelidir.



Modüller Module deyimiyle başlar, End Module deyimiyle sona erer.

```
Module Module1

End Module
```

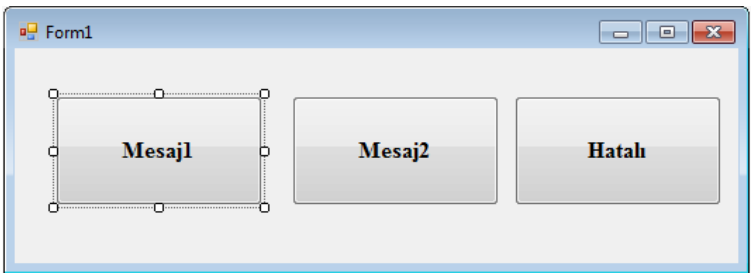
Bir modül ister Form sınıfına ait kod dosyasında olsun isterse diğer nesnelere ait kod dosyalarından birinde olsun modüldeki metodun kullanılma şeklinde bir değişiklik olmayacaktır. Çünkü derleyici açısından Modüllerdeki metotların hepsi share özelliktedir. Yani bütün sınıflardan erişilebilmeleri mümkündür.

Modüllerdeki metotların ayrıca shared olarak tanımlanmalarına gerek yoktur. Aynı kod dosyası içerisinde istenilen sayıda modül hazırlanması mümkündür. Ancak Visual Basic derleyicisi içi içe modül hazırlanmasına imkân vermemektedir.

Bir projede birden fazla modül varsa ve aynı ada sahip birden fazla metot varsa metot işletilirken metot adından önce modül adı yazılmalıdır.

Örnek 1

- İki farklı modülde aynı isimli metod'un(Sub) kullanılması.



Mesaj adlı sub metodu hem Module1 hem de Module2 modüllerinde kullanılmaktadır. Bu nedenle Mesaj adlı modül çağrılırken hangi modülde olduğu belirtilmelidir. Aksi takdirde program hata verecektir.

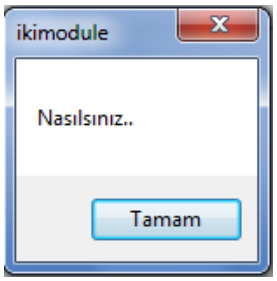
Modüller *Module* deyimiyle başlar, *End Module* deyimiyle sona

Mesaj1 adlı Button nesnesinde mesaj sub metodu kullanılırken module1 modül adı belirtilmiş ve "Merhaba" mesajı elde edilmiştir.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object,
        Module1.mesaj()
    End Sub
End Class
```

Mesaj2 adlı Button nesnesinde mesaj sub metodu kullanılırken module2 module adı belirtilmiş ve "Nasılsınız.." mesajı elde edilmiştir.

```
Public Class Form1
    Private Sub Button2_Click(sender As Object,
        Module2.mesaj()
    End Sub
End Class
```



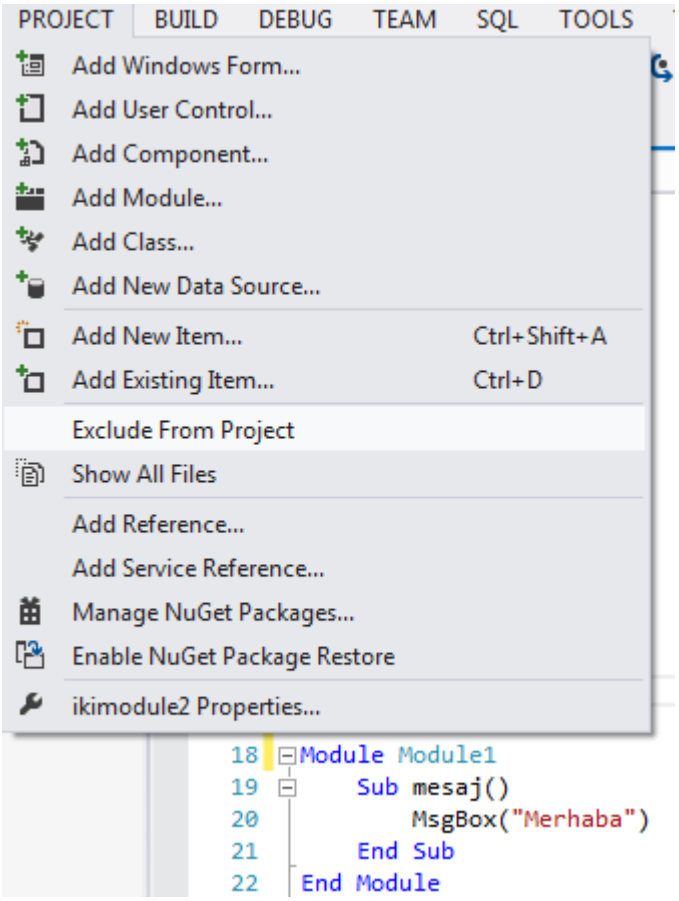
Hatalı adlı *Button* nesnesinde mesaj sub metodu kullanılırken modül adı belirtilmediği için program çalışmamış ve hata vermiştir.

Modüller Private tanımlanmamışlarsa diğer sınıflardan erişilebilirler.

```
Public Class Form1
    Private Sub Button3_Click(sender As Object,
        mesaj()
    End Sub
End Class
```

Modül Kaldırmak

Modüller farklı projelerde kullanılabilirler.



Proje içerisinde önceden oluşturulmuş olan bir modülün projeden kaldırılması *Project* menüsünden *Exlude From Project* satırının seçilmesiyle veya kaldırılmak istenilen modül Mouse sağ tuş ile tıklandıktan sonra açılan pencereden yine *Exlude From Project* satırının aktif hâle getirilmesiyle sağlanabilir.

PROSEDÜR VE FONKSİYON KULLANIMININ AVANTAJLARI

- Kod tekrarının önlenmesi
- Programların daha okunaklı olmasının sağlanması
- Program geliştirmenin kolaylaştırılması
- Diğer projelerde yeniden kullanılabilmeleri

Bu ve benzeri birçok nedenle fonksiyon ve prosedürlerin kullanılması yukarıdaki avantajlara ilaveten program geliştirmede esneklik ve modülerlik sağlamaktadır.

Prosedure(Sub) Tipi Metot Hazırlamak

Nesneye yönelik programcılıkta Procedure(Sub) ve Function yapılarına Metot denilmektedir. Sub tipi metot hazırlamak için bir sınıf içerisine Sub

komutuyla birlikte bir prosedür ismi verilmelidir. Sub metodu, *Sub* anahtar kelimesiyle başlar ve *End Sub* anahtar kelimesiyle sona erer.

```
Public Class Form1
    Sub duyuru()

    End Sub
```

Sub anahtar kelimesi Class içerisinde yazıldığında editör otomatik olarak Sub…..End Sub yapısını oluşturur.

Sub metodu, *Sub* anahtar kelimesiyle başlar ve *End Sub* anahtar kelimesiyle sona erer.

Örnek 2

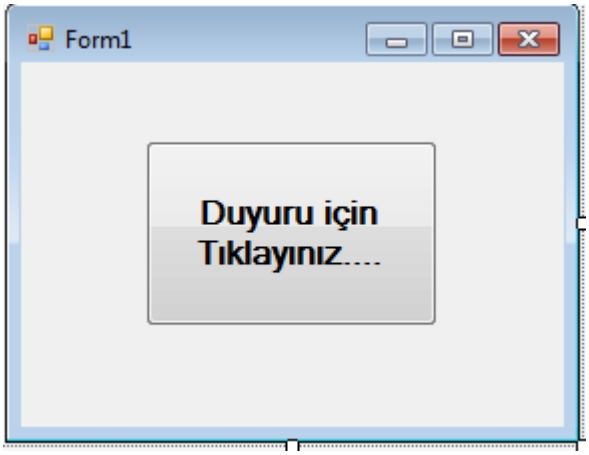
- Duyuru adlı bir Sub metodu oluşturularak ana programdan sub metodunun kullanılması.

Örnek 2 için geliştirilen Sub metodu

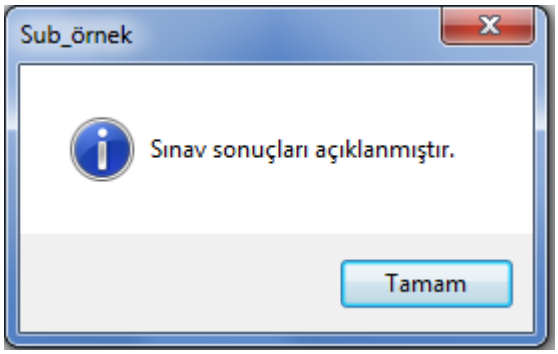
```
Public Class Form1
    Sub duyuru()
        MsgBox("Sınav sonuçları açıklanmıştır.", MsgBoxStyle.Information)
    End Sub
```

Örnek 2 için geliştirilen Ana Program

```
Private Sub Button1_Click(sender As Object, e As EventArgs)
    duyuru()
End Sub
```



Bir Sub metodundan Exit Sub deyimi kullanılarak çıkılır.



Exit Sub Deyimi İle Metot'dan Çıkmak

Programlama mantığı, program satırlarının yukarıdan aşağıya doğru sıralı bir şekilde işletilmesi şeklindedir. Metot içerisinde de aynı mantık geçerlidir. Bütün satırlar işletildikten sonra projenin işletimi söz konusu metodun çağrıldığı satırı izleyen bir sonraki satıra geçer. Çağrılıp işletilen metottan sonra işletilen başka bir metot veya satır yoksa projenin işletimi aktif formda kalır.

Exit Sub deyiminden yararlanılarak geriye değer döndürmeyen metottaki satırlardan duruma göre bir kısmı veya tamamı işletilmeden metottan çıkılabilir.

Örnek 3

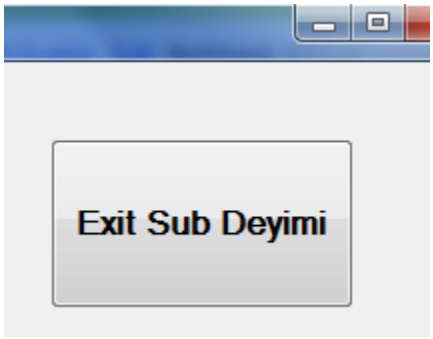
- Girilen iki sayının çarpımını gerçekleştiren,ancak sayılardan herhangi birinin 0 olması hâlinde metottan çıkışın yapıldığı Sub metodu

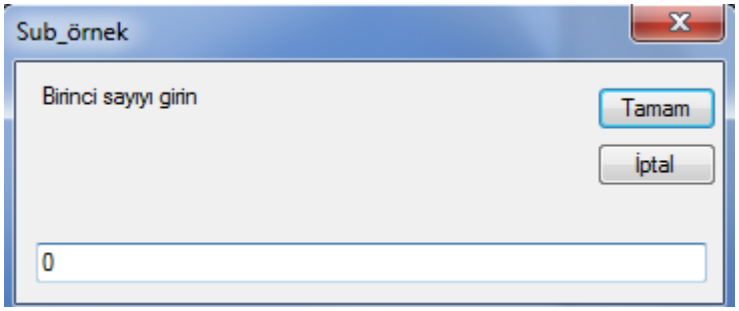
Örnek 3 için geliştirilen Sub metodu

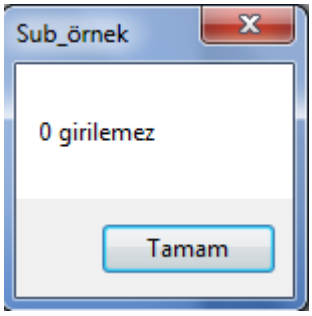
```
Sub carpma()
    Dim s1, S2 As Integer
    s1 = Convert.ToUInt16(InputBox("Birinci sayıyı girin"))
    If s1 = 0 Then
        MsgBox("0 girilemez")
        Exit Sub
    End If
    S2 = Convert.ToUInt16(InputBox("İkinci sayıyı girin"))
    If S2 = 0 Then
        MsgBox("0 girilemez")
        Exit Sub
    End If
    MsgBox("Girilen sayıların çarpımı :" + Str(s1 * S2))
End Sub
```

Örnek 3 için geliştirilen Ana Program

```
Private Sub Button2_Click(sender As Object, e As EventAr
    carpma()
End Sub
```







Fonksiyonlar, geriye değer döndüren metotlardır.

FUNCTION TİPİ METOT HAZIRLAMAK

Önceden de belirtildiği gibi Function tipi metotlar yani fonksiyonlar ile Sub tipi metotlar arasındaki en önemli fark fonksiyonların geriye değer döndürmeleridir. Yani geriye değer döndüren bir metot hazırlanmak istendiğinde Function değimi kullanılmalıdır.

Fonksiyonlar, yani geriye değer döndüren metotlar *Function* anahtar kelimesiyle başlar ve *End Function* anahtar kelimesiyle sona erer.

```
Public Class Form1
    Function sayılar()

    End Function
```

Function anahtar kelimesi Class içerisinde yazıldığında editör otomatik olarak Function….. End Function yapısını oluşturur.

Örnek 4

- Klavyeden girilen sayıların kare kökünü hesaplayan bir programın Fonksiyon kullanılarak kodlanması.

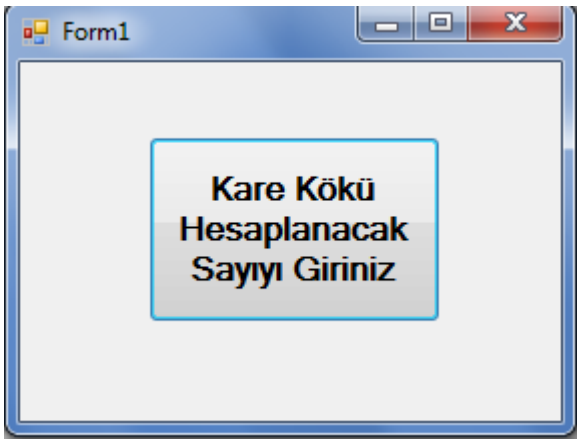
Fonksiyonlar, *Function* anahtar kelimesiyle başlar ve *End Function* anahtar kelimesiyle sona erer.

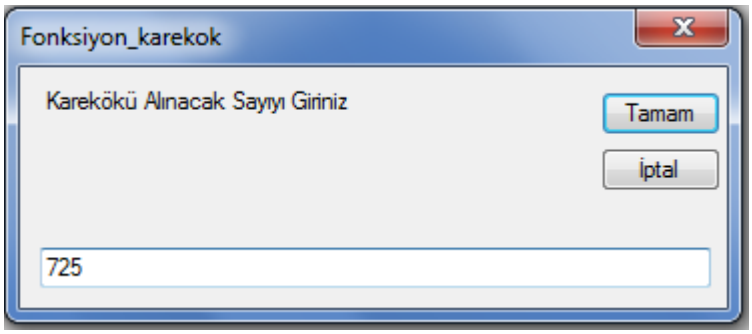
Örnek 4 için geliştirilen Fonksiyon

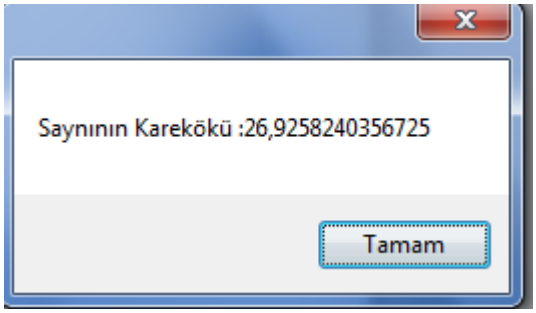
```
Public Class Form1
    Function karekokhesapla(ByVal X As Double) As Double
        Select Case Math.Sign(X)
            Case 1
                karekokhesapla = Math.Sqrt(X)
                Exit Function
            Case 0
                karekokhesapla = 0
                Exit Function
            Case -1
                karekokhesapla = -1
        End Select
    End Function
```

Örnek 4 için geliştirilen Ana Program

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim kok1 As Double
        Dim say1 As Double
        say1 = Convert.ToDouble(InputBox("Karekökü Alınacak Sayıyı Giriniz"))
        kok1 = karekokhesapla(say1)
        Select Case kok1
            Case 0
                MessageBox.Show("Girilen sayı 0")
            Case -1
                MessageBox.Show("Girilen sayı negatif")
            Case Else
                MessageBox.Show("Saynının Karekökü :" & kok1.ToString())
        End Select

    End Sub
End Class
```







Hazırlanan karekokhesapla() metodu ana programdan girilecek sayıların karekökünü hesaplamak için kullanılmaktadır. Bu metoda dışarıda Double tipinde bir veri girilmesi düşünüldüğü için metot tanımlanırken parametre tipi olarak Double yazıldı. Bu fonksiyonda kullanılan Sign() ve Sqrt() metotları.NET Framework ile gelen Math sınıfının metotlarıdır. Math sınıfı System adlı namespace'de yer aldığından herhangi bir bildirim yapılmadan kullanılması mümkündür.

Geriye döndürülecek bilginin tipi fonksiyon adından sonra belirtilir. Karekokhesapla() metodu geriye Double bilgi döndürür.

Function'dan geriye döndürülecek bilginin tipi function adından sonra belirtilmelidir.

Özet

- Bir standart modül dosya uzantısı .vb olan ve programın her tarafında kullanılabilen değişken, prosedür ve fonksiyon içeren özel bir dosyadır. Modüller, Solution Explorer penceresinde ayrı ayrı listelenerek görüntülenirler. Bir program içerinde modül oluşturmak için PROJECT menüsünden Add Module seçilmeli veya yine PROJECT menüsünden Add New Item seçildikten sonra Module tıklanmalıdır.
- Nesneye yönelik programcılıkta Prosedür ve Fonksiyonların ikisine birden Metot denilmektedir. Visual Basic .NET programlama dilinde değer göndermeyen metotlar Procedure(Sub), geriye değer gönderen metotlar ise Function deyimi ile tanımlanmaktadırlar.
- Sub metodu, Sub anahtar kelimesiyle başlar ve End Sub anaht**ar** kelimesiyle sona erer. Bir Fonksiyon ise Function anahtar kelimesiyle başlar ve End Function anahtar kelimesiyle sona erer.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. Visual Basic.Net'de proje içerisinde oluşturulan yeni bir modülün ismi aşağıda belirtilen pencerelerden hangisinde görüntülenmektedir?
   a) ToolBox
   b) Form
   c) Properties
   d) Solution Explorer
   e) Server Explorer

2.
```
Sub dogumyeri(ByVal Person As String)
    '.........
End Sub
```
Yukarıdaki procedure için hangisi doğrudur?
   a) Procedure'nin adı Person 'dur.
   b) Sub ile End sub arasına yordam ifadeleri yazılmalıdır.
   c) Byval değişken adıdır.
   d) Metot Fonksiyon olarak tanımlanmıştır.
   e) Procedure hatalı kullanılmıştır.

3. Bir modül programdan nasıl kaldırılır?
   a) Exlude From Project seçilir.
   b) Configuration Manager seçilir.
   c) Exceptions seçilir.
   d) Delete seçilir.
   e) Rename seçilir.

4. Aşağıda oluşturulmuş metotlardan hangisi doğru kodlanmıştır?
   a) Sub aaa
      End Function
   b) Sub aaa
      End Sub
   c) Function
      End Function
   d) Function Sub
      End Function
   e) Sun Function
      End Sub

5. Bir prosedür metodundan aşağıda belirtilen deyimlerden hangisi kullanılarak çıkılır?
   a) Exlude From
   b) Exit
   c) Exit Function
   d) Delete Function
   e) Exit Sub

6.
```
Public Class Form1
    Private Sub Button1_Click(sender As Object,
        Module1.mesaj()
    End Sub
End Class
```

   Yukarıda verilen kodlama ile hangi işlem gerçekleştirilmektedir?
   a) Module1 adlı modül oluşturulmaktadır.
   b) mesaj() adlı modül çağrılmaktadır.
   c) mesaj() adlı fonksiyon çağrılmaktadır.
   d) mesaj() adlı prosedür çağrılmaktadır.
   e) Module1 adlı modülün adı mesaj() yapılmaktadır.

7. Ana programa değer gönderen metot hangisinde doğru verilmiştir?
   a) Sub
   b) Procedure
   c) Function
   d) Metot
   e) Module

8. Bir function metodunun oluşturulma işlemi hangisinde doğru verilmiştir?
   a) Function
      End Function
   b) İsim Function
      Function
   c) End Function
      Function
   d) Function İsim
      End Function
   e) Function İsim
      End

9.
```
Public Class Form1
    Function xxx()
    End Function
    Private Sub Form1
        xxx()
    End Sub
End Class
```

Yukarıda verilen kodlama ile hangi işlem gerçekleştirilmektedir?

a) xxx() module'i oluşturulmuş ve çağrılmıştır.
b) xxx() procerure'si oluşturulmuş ve çağrılmıştır.
c) xxx() Function'u oluşturulmuş ve çağrılmıştır.
d) Form1 içerisinde xxx() procedure'i oluşturulmuştur.
e) Form1 prosedure'i çağrılmıştır.

10. Aşağıdakilerden hangisi bir prosedür veya fonksiyon oluşturmanın avantajlarından biri değildir?

a) Kod tekrarının önlenmesi
b) Projenin maliyetinin düşmesi
c) Programların daha okunaklı olmasının sağlanması
d) Program geliştirmenin kolaylaşması
e) Diğer projelerde yeniden kullanılabilmeleri

Cevap Anahtarı

1.D, 2.B, 3.A, 4.B, 5.E, 6.D, 7.C, 8.D, 9.C, 10.B

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 II Ders Notları

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

SINIF YAPISI VE NESNE YÖNELİMLİ PROGRAMLAMA

İÇİNDEKİLER

- Nesne Yönelimli Programlama
- Nesne Yapısı
- Sınıf Yapısı
- Sınıf Tanımlama
- Özellikler
- Erişim Belirleyiciler
- Nesne Tanımlama
- Nesne Yönelimli Programlamanın Temel Özellikleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Nesne ve Sınıf kavramlarını tanıyabilecek,
  - Yeni bir sınıf oluşturabilecek,
  - Bir sınıftan nesne yeni bir nesne türetebilecek,
  - Nesne yönelimli programlama mantığını anlayabilece,k
  - Nesne yönelimli programlamanın temel özelliklerini öğrenebileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Arş. Gör. Mehmet Cem BÖLEN

ÜNİTE

4

GİRİŞ

Yazılım teknolojileri son 40 yılda artan bir ivmeyle gelişmeye devam etmektedir. Buna paralel olarak zamanla artan ve çeşitlenen kullanıcı istekleri doğrultusunda geliştirilen yazılımların ölçekleri ve boyutları büyümektedir. 1970'li yılların sonlarına doğru birçok yazılım projesinin boyutu klasik fonksiyonel programlama tekniklerinin başa çıkabileceği seviyeyi aşmaya başlamıştı. Bunun üzerine bilgisayar bilimciler yazılım geliştirme aşamasını kısaltmak ve sistematik hâle getirmek amacıyla yeni yollar aramaya başlamış ve ortaya atılan çözümlerden biri olan nesne yönelimli programlama tekniği yazılım dünyası tarafından kısa sürede geniş bir kitle tarafından kabul görmüştür. Bu ünitede nesne yönelimli programlama tekniği ve bu tekniğin en önemli yapıları üzerinde durulacaktır. Özellikle.Net programlama konusunda uzman olmak isteyen her programcının nesne yönelimli programlamaya hâkim olması gerekmektedir. Bu nedenle ünite boyunca teorik bilginin yanında kod örnekleri verilerek konunun daha kolay anlaşılması amaçlanmıştır.

NESNE YÖNELİMLİ PROGRAMLAMA

Nesne yönelimli programlama, program yazmayı kolaylaştıran, kod anlaşılabilirliğini arttıran, yazılım geliştirme sürecini kısaltan ve sistematikleştiren bir teknik ya da paradigma olarak tanımlanabilir. Klasik programlama tekniklerine göre daha esnek ve sağlam bir yapı sunan nesne yönelimli programlama, ilk olarak C++ programlama dilinde desteklenmiştir. C++, klasik C dilinde olduğu gibi yapısal programlamaya imkân tanımakla birlikte dönemin yeni tekniği olan nesne yönelimli programlamaya destek vermekteydi.

C++'ın nesne yönelimli programlama dillerinden biridir.

C++'ın kullanımının yaygınlaşmasıyla birlikte nesne yönelimli programla tekniği tüm dünyada kabul görmeye başlamış ve bu da ilerleyen dönemde Java, C#, Vb.Net gibi farklı nesne yönelimli dillerin ortaya çıkmasına neden olmuştur. Özellikle yapısal ve fonksiyonel programlama teknikleri ile karşılaştırıldığında sunmuş olduğu sınıf ve nesne yapılarıyla gerçek hayattaki kavramların kod hâlinde ifade edilmesini çok kolay hâle getirmiştir. Nesne yönelimli programlamanın temel mantığı programlama ortamındaki her şeyi bir nesne olarak kabul etmesidir.

Nesne yönelimli programlamayı anlamak için iki temel kavramı bilmek gerekmektedir. Bu kavramlar nesne (Object) ve sınıf (Class) tır. Bu kavramlar birbirleriyle yakın ilişkili oldukları için karıştırılabilmekte olup, nesne yönelimli programlamanın anlaşılması açısından tam anlamıyla öğrenilmesi önem taşımaktadır. Bu yüzden nesne yönelimli programlamanın özelliklerine geçmeden önce nesne ve sınıf kavramlarını tanıyacaksınız.

Nesne Yapısı

Günlük hayatta işlerimizi nesneleri kullanarak hâllederiz. Örneğin arabalar kullanılacağı amaca göre tasarlanmış ve çeşitli özellikleri olan nesnedir. Kısaca nesneler gerçek hayatta görebildiğimiz, somut cisimlerdir. Bu örnekten hareketle nesneyi belirli bilgileri taşımak, bazı işlevleri gerçekleştirmek için kullanılan programlama birimi olarak tanımlayabiliriz. Aslında nesne kavramını klasik programlama (nesne yönelimli olmayan) bakış açısından değerlendirdiğimizde bir değişken gibi düşünebiliriz. Nesne yönelimli programlama mantığında nesneler sınıflardan türerler ve bu sınıflarda tanımlanan özellik ve fonksiyonlara sahip olurlar.

Programcılık terminolojisinde nesne kavramı bellekte belirli bir yer işgal eden ve belirli bir türe sahip herşeydir. Nesne yönelimli programlama tekniği bakış açısından değerlendirildiğinde ise programlamadaki en küçük birimdir. Bir başka ifadeyle nasıl insan hücreden, madde atomdan oluşuyor ise nesne yönelimli programlamada da herşey nesneden oluşur.

Nesne yapısında veri bulundurduğu gibi bu verileri yönetmek adına çeşitli fonksiyonlara da sahip olabilir. *Bir nesneyi diğer bir nesneden ayıran onun özellikleri ve işlevleridir. Bu özellik ve işlevler ise nesnenin ait olduğu sınıfta belirlenir.*

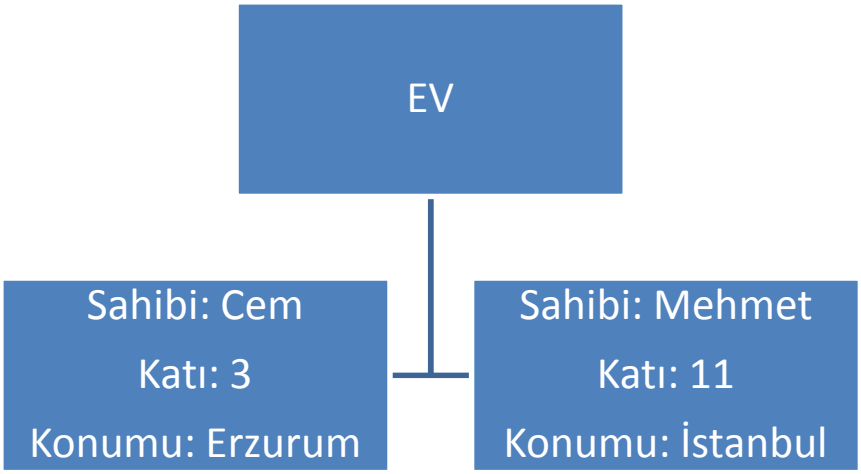
Sınıf Yapısı

Nesnelerin özellikleri ve işlevleri sınıflar tarafından belirlenir.

Sınıf (Class) yapısı programlamaya yeni başlayanların anlamakta zorlandıkları bir konu olmakla birlikte nesne yönelimli programlama mantığını öğrenmek için kesinlikle bilinmesi gerekmektedir. *Visual Basic.Net'te nesne yönelimli programlamanın temelini sınıf yapısı oluşturur. Sınıflar kullanılarak nesneler tanımlanır ve nesneleri birbirinden ayıran şey üyesi olduğu sınıftır.*

Sınıf yapısını gerçek hayattan bir örnekle anlatmak gerekirse buna mimarların çizmiş olduğu bina tasarımları örnek verilebilir. Mimarları programcı olarak kabul edersek, yapmış oldukları tasarımı Visual Basic.Net dilinde bir sınıfa benzetebiliriz. Gerçek hayatta mimar inşa edilecek bina için bir tane tasarım yapar ve inşaatı yapan şirket bu tasarıma uyarak aynı tipte istediği kadar bina inşa edebilir.

Sınıf yapısını daha iyi anlamak için aşağıdaki şekli inceleyelim. Ev isimli bir sınıfımız olsun. Bu ev sınıfından birkaç tane ev türetelim.



Şekil 1. Sınıf Örneği

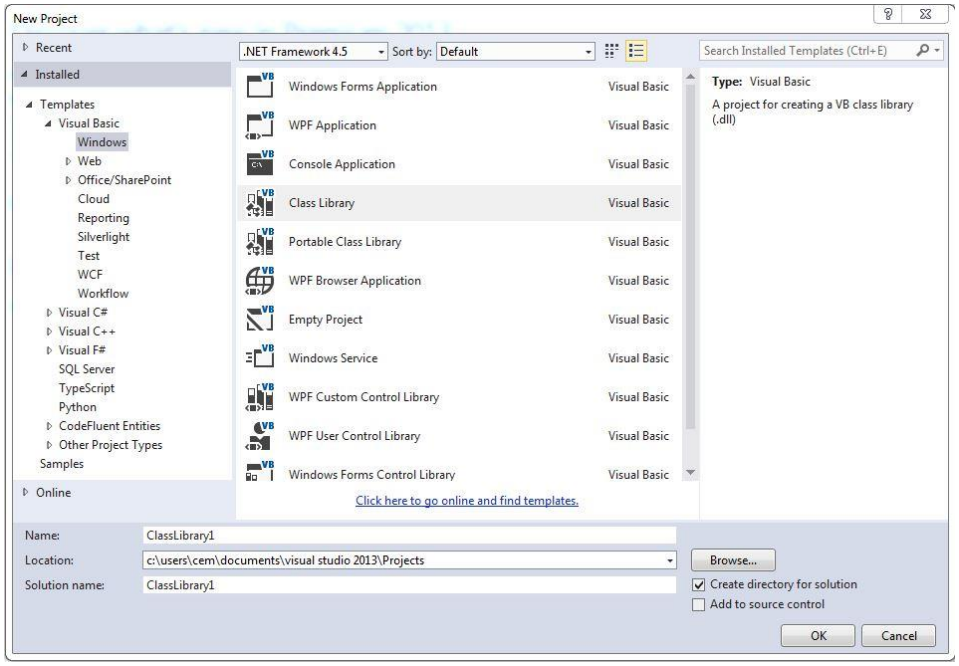
Fonksiyonlar sınıf yapısı kullanılarak bir çatı altında toplanır. Bu da yazılan kodun yeniden kullanılabilir ve genişletilebilir olmasını sağlar.

Şekil 1'deki örnekte Cem ve Mehmet'in evlerinin kat ve konum özelliklerinin birbirlerinden farklı olduğu ancak ikisinin de ortak noktasının ev olduğu görülmektedir. Bir başka ifadeyle sınıf bir tür bilgisi olup, bu türden türetilen nesneler ise farklı işlev ve özelliklere sahip olabilirler.

Gerçek hayatta nesneleri tanımlamak istediğimizde ona ait belirli özellikleri ve fonksiyonları tarif ederiz. Bu özellik ve fonskiyonlar sınıf yapısı içerisinde tanımlanır. Yine programlarımız içerisinde sıklıkla kullanılan değişken ve fonksiyonlar sınıf yapısı kullanılarak bir çatı altında toplanır. Bu da yazılan kodun yeniden *kullanılabilir (Reusability)* ve *genişletilebilir (Extensibility)* olmasını sağlar.

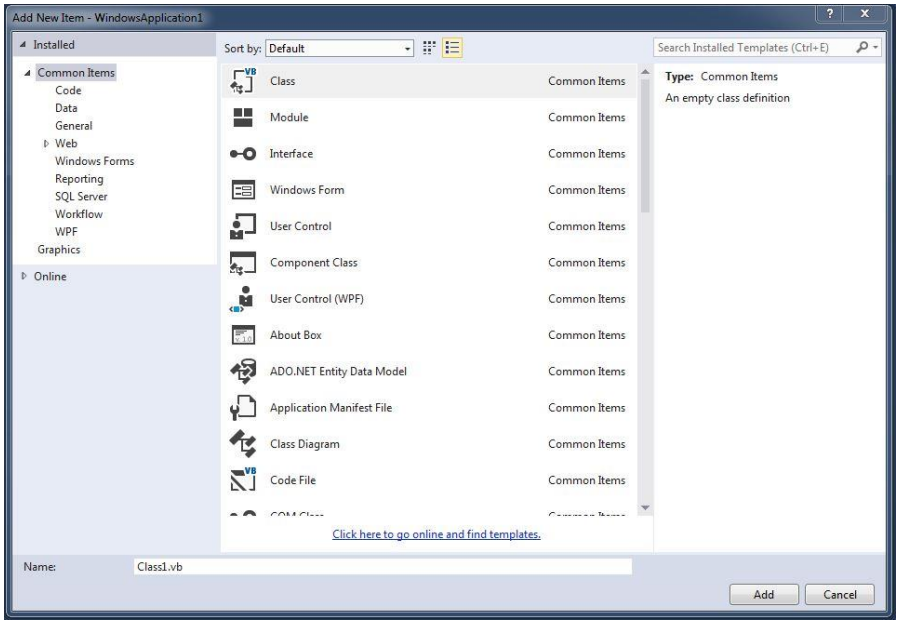
Sınıf tanımlama

Visual Basic.Net'te sınıf oluşturmak çok kolaydır. Yeni bir sınıf oluşturmanın 2 yolu vardır. Eğer oluşturulacak Class bir dll olarak derlemek istiyorsak bunun için File/New Project menüsüne tıklayıp karşımıza gelen proje şablonları içerisinden class Library şablonunu seçmemiz gerekmektedir. (Resim 1)



Resim 1. Yeni Proje Oluşturma Penceresi

Eğer Class dosyası ayrı bir proje oluşturulup derlenmeyecekse yani varolan bir proje içerisine yeni bir Class eklenecekse, Project menüsünden Add Class seçeneği ile ekleme işlemi yapılabilir. (Resim 2)



Resim 2. Mevcut Projeye Bileşen Ekleme Penceresi

Alan adı verilen yapılarda nesnelere ait bilgiler saklanır.

Bu işlemleri yaptıktan sonra sıra sınıf bildiriminin yapılmasına geldi. Sınıf bildirimi ile ilgili bilinmesi gereken ilk şey sınıf tanımının ve sınıf ile ilgili yazılacak tüm kodların (metodlar, özellikler *vb.*) "class" bloğu içerisinde yapılacağıdır. Class

anahtar sözcüğünden sonra sınıfın adı yazılır. Sonrasında sınıfın özelliklerinin ve işlevlerinin kodlanması class bloğu içerisinde yapılır.

Aşağıda verilen örnekte Araba isimli bir sınıf tanımlanmış ve bu sınıfa ait 3 *alan (field)* belirlenmiştir. Örnekte ilk dikkat çeken Private anahtar kelimesi ile tanımlanan ve sadece araba sınıfı içerisinden erişilebilen _marka,_model ve _maxhiz isimli alanlardır. *Alan (Field)* adı verilen bu yapılar kullanılarak nesnelerin bilgileri saklanır.

```
Public Class Araba

    Private _marka As String
    Private _model As String
    Private _maxhiz As Integer

    Property Marka() As String
        Get
            Return _marka
        End Get
        Set(ByVal Value As String)
            _marka = Value
        End Set
    End Property
    Property Model() As String
        Get
            Return _model
        End Get
        Set(ByVal Value As String)
            _model = Value
        End Set
    End Property
    Property MaksimumHiz() As Integer
        Get
            Return _maxhiz
        End Get
        Set(ByVal Value As Integer)
            _maxhiz = Value
        End Set
    End Property
    End Class
```

Alanların tanımlanması ve kullanımı değişken tanımlamayla hemen hemen aynıdır. Alan tanımında kullanılan Private anahtar sözcüğü, erişimin sadece sınıf içerisinde yapılmasını sağlamaktadır. *Eğer Private yerine Public anahtar sözcüğü*

kullanılsaydı bu alanlara sınıf dışından da erişim mümkün olacaktı. Örnekte dikkat çeken bir başka anahtar kelime de Property'dir. Araba sınıfının özellikleri olan marka, model ve fiyat Property anahtar kelimesi ile tanımlanır. Bir başka ifadeyle Araba sınıfından türetilen bir nesnenin özellikleri Property anahtar kelimesi ile belirlenir.

Özellikler

Bir özelliğin değeri değiştiğinde çalışmasını istediğimiz kodları Set bloğu içerisine yazarız.

Alanlar (Fields) ve özellikler (Properties) kavramları hemen hemen aynı niteliklere sahip olmakla birlikte nesne yönelimli programlama mantığında kullanım amaçları farklıdır. Önceki örneğimizde alan olarak tanımlanan _marka, _model ve _maxhiz değişkenleri eğer *Private* olarak değil *Public* olarak tanımlansalardı Araba sınıf türünden tanımlanan nesneler üzerinden erişilebilirdi. Böyle bir durumda da özellik (Property) kullanımına gerek kalmazdı. Fakat nesne yönelimli programlama yaklaşımında sınıf alanlarının private yapılması ve bu alanlara erişimin Özellik yapısı üzerinden sağlanması önerilmektedir. Bunun sebebi alanlara dışarıdan verilecek değerlerin kontrol edilmesi ve özellik yapısında nesne dışında kodlar tarafından değiştirilebilen bir esnekliğin sağlanmasıdır. Bu yüzden nesne yönelimli programlama bakış açısında nesnelerin özelliklerine özellik (Property), bu özelliklere sınıf içerisinde veri alışverişi ise alanlar(Fields) tarafından sağlanır. Bu da nesne yönelimli programlamanın sarmalama(encapsulation) ilkesine uyulmasını sağlar.

Visual Basic.Net dilinde özellik tanımı Property anahtar kelimesi ile yapılmaktadır. Özellik değerleri Get bloğu içerisinde Return anahtar kelimesi kullanılarak döndürülür. Bir özelliğe değer ataması yapılmak istendiğinde set bloğunda yer alan kodlar çalışır. Atanmak istenen değer Set metoduna gönderilen bir parametre (value) ile özelliğe atanır.

Bir önceki verilen örnekte yer alan özelliklerden birini inceleyelim. Aşağıdaki Marka isimli özellikte, veri aslında _marka isimli değişkende tutulur. Ancak Marka() isimli özelliği, sınıf dışında bu değişkene erişmek için kullanırız.

```
Property Marka() As String
Get
        Return _marka
End Get
Set(ByVal Value As String)
        _marka = Value
End Set
End Property
```

Bir sınıf içerisinde tanımlanan özelliğin sadece okunması yani değer ataması yapılmaması istenebilir. Bu durumda Property anahtar kelimesinin önüne Readonly kelimesi eklenir. Ayrıca Set bloğu kaldırılır. Aşağıda verilen örnekte salt okunur bir özellik tanımlamasına ait kod örneği görülmektedir:

Bir sınıf içerisinde tanımlanan özelliğin sadece okunması yani değer ataması istendiğinde Property anahtar kelimesinin önüne *Readonly* kelimesi eklenir.

```
Readonly Property Marka() As String
Get
        Return _marka
End Get
End Property
```

Özellikler Get ve Set blokları içerisinde kod çalıştırabilmesi ile metodlara değer atama ve okuma yönünde ise alanlara benzemektedirler. Sınıfa ait nitelikler özellikler kullanılarak belirlenir.

Erişim belirleyiciler

Şu ana kadar ismini bilmediğimiz ancak kullandığımız bir yapı olan erişim belirleyiciler (Access Specifiers), hazırlanan sınıf yapıları içerisindeki değişken, metod ve özelliklere başka sınıflardan ya da formlardan erişimini yönetmek için kullanılır. Tablo 1'de Visual Basic.Net dilinde kullanılan erişim belirliyeci adları ve kullanıldıkları sınıf üyesine sağladıkları erişim durumu görülmektedir:

Tablo 1. Erişim Belirleyiciler Kullanıldıkları Metod ve Özellikler

İfade		Erişilme Durumu
Public	→	Üyeye tanımlandığı sınıf içerisinden veya dışarıdan erişim serbesttir.
Protected	→	Üyeye sadece tanımlandığı sınıf ve bu sınıftan türemiş alt sınıflardan erişilebilir.
Friend	→	Sadece üyenin bulunduğu proje içerisinden erişilebilir.
Protected Friend	→	Sadece üyenin bulunduğu proje içerisinde üyenin sınıfından türemiş alt sınıflardan erişilebilir.
Private	→	Sadece üyenin tanımlandığı sınıf içerisinden erişilebilir.

Ünite boyunca yapılacak örneklerde Public ve Private anahtar kelimeleri kullanılacaktır. Diğer erişim belirleyiciler genelde orta ve ileri seviye nesne yönelimli programlama kodlarında yer aldığı için bu ünitedeki kod örneklerinde yer almayacaktır.

Metot

Belirli bir işlemi yerine getirmek için tanımlanan alt programlara metot denir. Metotların temel kullanım amacı defalarca çalışacak kodları metot adı verilen yapıların içerisine yazarak hem kodun okunabilirliğini arttırmak hem de bu kodların tekrar kullanılabilirliğini sağlamaktır.

Sınıfların işlevselliği sahip oldukları metotlara bağlıdır denebilir. Visual Basic.Net'te tanımlanan sub ve fonksiyonlar aynı zamanda metot olarak da adlandırılırlar.

Nesne Tanımlama

Vb.Net ve C# programlama dillerinde nesne yönelimli programlamanın hemen hemen tüm özellikleri ortak olup, sadece yazım kuralı farklılıkları vardır.

Visual Basic.Net'te bir sınıftan nesne (instance) türetmek için New anahtar kelimesi kullanılır. Örneğin Araba sınıfı türünden yeni bir nesne bildirimi aşağıdaki gibi yapılır:

```
Dim ornekaraba As New Araba
```

Bu işlem yapıldıktan sonra ornekaraba nesnesi için bellekte bir yer ayrılır. Artık ornekaraba isimli nesne kullanılarak Araba sınıfında tanımlanan özellik ve metotlara erişilebilir. Araba sınıfında tanımlanan özellik ve metodları görüntülemek için ornekaraba yazdıktan sonra nokta (.) konulmalıdır.

```
Dim ornekaraba As New Araba
        ornekaraba.Marka = "Ferrari"
        ornekaraba.Model = "F458"
        ornekaraba.MaksimumHiz = 340
```

Yukarıda verilen örnekte daha önceden oluşturmuş olduğumuz Araba sınıfının özelliklerine değer atanmıştır.

Bireysel Etkinlik

- Visual Basic.Net ortamında Ev isimli bir sınıf oluşturunuz ve aklınıza gelen özellikleri bu sınıfa tanımlayınız. Ardından bir Windows Form'da bu sınıftan nesne türeterek bu özelliklere uygun değerler atayınız.

NESNE YÖNELİMLİ PROGRAMLAMANIN TEMEL ÖZELLİKLERİ

Nesne yönelimli programlama üç temel kavram üzerine kurulmuştur. Bu kavramlar Tablo 2'de görülmektedir.

Tablo 2. NYP'nin Temel Özellikleri

Nesne Yönelimli Programlamanın Temel Özellikleri
Sarmalama (Encapsulation)
Miraslık (Inheritance)
Çok biçimlilik (Polymorphism)

Sarmalama

Nesne yönelimli programlamanın temel özelliklerinden olan sarmalama (Encapsulation), oluşturulan nesnenin işlev ve özelliklerinin isteğe göre diğer nesnelerin erişimine kapatılmasıdır. *Sarmalama özelliğinin temel kullanım amacı nesne içeriğine gereksiz erişimi engellemek ve bilgi güvenliğini sağlamaktır.* Sınıf içerisinde tanımlanan metotlar, özellikler ve alanlara erişimin, erişim belirleyiciler ile sınırlandırılması sarmalamaya en iyi örnektir.

Nesne Yönelimli Programlamanın Temel Özellikleri; Sarmalama, Miraslık ve Çok biçimliliktir.

Miraslık

Yazdığımız sınıflar bazen istediklerimizi yapmaya yeterli olmayabilir. Bu durumda ilk akla gelen mevcut sınıfın işlevleri ve özellikleri genişletilmesidir. Ancak bunu yapmak, sınıfın gereksiz büyümesine ve kodların okunabilirliğinin azalmasına yol açabilir. Bunu engellemek adına nesne yönelimli programlamanın sunmuş olduğu *Miraslık özelliğinden yararlanarak mevcut sınıfa yeni özellik ya da işlev eklemek yerine bu sınıfın bazı özelliklerini miras alıp, gereken yeni özellik ve işlevlerin eklenerek yeni bir sınıf oluşturulabilir*. Kısacası Miraslık, miras alınan sınıfın belirli özellikleri ve işlevlerini içermekle birlikte bu sınıfta yer almayan yeni özellik ve işlevlere sahip yeni bir sınıfa sahip olmaya yaramaktadır.

Nesne yönelimli programlama terminolojisinde yeni sınıfın türetilmesi için özellik ve işlevleri miras alınan eski sınıfa temel sınıf (base class), yeni sınıfa ise türemiş sınıf (derived class) denir. Visual Basic.Net'te miraslık özelliği kapsamında kullanılan bazı anahtar kelimeler ve açıklamaları şu şekildedir:

MustInherit: Eğer bir sınıf, MustInherit anahtar kelimesi kullanılarak tanımlanmışsa bu sınıf kendi başına kullanılamaz ancak türetilerek kullanlabilir. MustInherit anahtar kelimesiyle bir sınıf oluşturulmasının amacı nesne yönelimli programlama bakış açısıyla sınıf mimarisinde genel bir çerçeve belirlemektir.

NotInheritable: Kendisinden bir türetme yapılmayacağı düşünülerek tasarlanmış sınıflar NotInheritable anahtar kelimesi kullanılarak tanımlanırlar.

Inherits: Türemiş bir sınıf tanımlanmak istendiğinde Inherits anahtar kelimesi kullanılarak temel alınacak sınıf adı yazılır ve böylece bu sınıfın metot ve özelliklerine erişilir.

Çok Biçimlilik

Temel sınıfta yazdığımız özellik ve metotları, bu sınıftan türetilen bir sınıfta da kullanabiliriz. Ancak bazı durumlarda temel sınıfta kullanılan metodu veya özelliği türemiş sınıfta farklı bir biçimde kullanmak isteyebiliriz. Bu durumda nesne yönelimli programlama tekniğinin bize sunmuş olduğu Çok Biçimlilik (Polymorphism) özelliğini kullanmamız gerekecektir.

Temel sınıfta tanımlanmış özellik ve metotların, taban sınıftan türeyen bir sınıfta farklı biçimlerde tanımlanarak kullanabilmesine Çok Biçimlilik denir. Çok Biçimliliğin kullanılmasının temel sebebi kodlardaki karmaşıklığı azaltması ve temel sınıf türünde tanımlanan bir nesnenin, bu sınıf türünden türetilmiş ya da türetilecek sınıflardaki üyeleri (fonksiyon, metot *vb.*) kullanabilmesini sağlamasıdır. Bunu gerçekleştirmek için Visual Basic.Net'te Overridable ve Overrides anahtar kelimeleri kullanılır. Örneğin taban sınıf da bir metot overridable olarak tanımlanırsa, bu sınıftan türetilen bir başka sınıfta bu metot Overrides anahtar kelimesi kullanılarak üzerinden geçilebilir yani yeniden tanımlanabilir.

Özet

- Nesne yönelimli programlama, program yazmayı kolaylaştıran, kod anlaşılabilirliğini arttıran, yazılım geliştirme sürecini kısaltan ve sistematikleştiren bir teknik ya da paradigma olarak tanımlanabilir.
- Programcılık terminolojisinde nesne kavramı bellekte belirli bir yer işgal eden ve belirli bir türe sahip herşeydir.
- Visual Basic.Net'te nesne yönelimli programlamanın temelini sınıf yapısı oluşturur. Sınıflar kullanılarak nesneler tanımlanır ve nesneleri birbirinden ayıran şey ,üyesi olduğu sınıftır.
- Visual Basic.Net dilinde özellik tanımı Property anahtar kelimesi ile yapılmaktadır. Özellik değerleri Get bloğu içerisinde Return anahtar kelimesi kullanılarak döndürülür. Bir özelliğe değer ataması yapılmak istendiğinde set bloğunda yer alan kodlar çalışır. Atanmak istenen değer Set metoduna gönderilen bir parametre (value) ile özelliğe atanır.
- Nesne yönelimli programlama Sarmalama, Miraslık Ve Çok Biçimlilik olmak üzere üç temel kavram üzerine kurulmuştur.
- Nesne yönelimli programlamanın temel özelliklerinden olan sarmalama (Encapsulation), oluşturulan nesnenin işlev ve özelliklerinin isteğe göre diğer nesnelerin erişimine kapatılmasıdır.
- Miraslık özelliğinden yararlanarak mevcut sınıfa yeni özellik ya da işlev eklemek yerine bu sınıfın bazı özelliklerini miras alıp, gereken yeni özellik ve işlevlerin eklenerek yeni bir sınıf oluşturulabilir.
- Temel sınıfta tanımlanmış özellik ve metotların, taban sınıftan türeyen bir sınıfta farklı biçimlerde tanımlanarak kullanabilmesine Çok Biçimlilik denir.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi nesne yönelimli bir programlama dili değildir?
   a) C++
   b) C#
   c) Visual Basic.Net
   d) C
   e) Java

2. Nesne yönelimli programlama bakış açısına göre programlamanın en küçük birimi nedir?
   a) Sınıf
   b) Metot
   c) Fonksiyon
   d) Nesne
   e) Alan

3. Nesnelerin özellikleri ve işlevleri ……. tarafından belirlenir.
   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) özellikler
   b) sınıflar
   c) alanlar
   d) metotlar
   e) yapılar

4. Aşağıdakilerden hangisi Visual Basic.Net içerisinde kullandığımız bir erişim belirleyici anahtar kelimesi değildir?
   a) Sealed
   b) Public
   c) Private
   d) Friend
   e) Protected

5. Visual Basic.Net'te bir sınıftan yeni bir nesne (instance) türetmek için aşağıdaki anahtar kelimelerden hangisi kullanılır?
   a) Inherits
   b) Public
   c) Equals
   d) New
   e) Private

6. …… özelliğinin temel kullanım amacı nesne içeriğine gereksiz erişimi engellemek ve bilgi güvenliğini sağlamaktır.

   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

   a) Miraslık
   b) Çok biçimlilik
   c) Yeniden kullanabilme
   d) Sarmalama
   e) Fonksiyonellik

7. Bir özelliğin değeri değiştiğinde çalışmasını istediğimiz kodları …… bloğu içerisine yazarız.

   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

   a) Sub
   b) Get
   c) If
   d) Function
   e) Set

8. Bir sınıfa ait üyeye tanımlandığı sınıf içerisinden veya dışarıdan erişimin serbest olması isteniyorsa hangi erişim belirleyici kullanılmalıdır?

   a) Public
   b) Friend
   c) Protected
   d) Protected Friend
   e) Private

9. Sınıf tanımı için aşağıdaki anahtar kelimelerden hangisi mutlaka kullanılmalıdır?

   a) Inherits
   b) MustInherit
   c) Class
   d) Public
   e) Property

10. Kendisinden bir türetme yapılmayacağı düşünülerek tasarlanmış sınıflar …….. anahtar kelimesi kullanılarak tanımlanırlar.

    Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

    a) NotInheritable
    b) Inherits
    c) Overrides
    d) MustInherit
    e) Public

Cevap Anahtarı

1.D, 2.D, 3.B, 4.A, 5.C, 6.D, 7.E, 8.A, 9.C, 10.A

YARARLANILAN KAYNAKLAR VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

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

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.

Clark, D., & Sanders, J, (2011). Beginning C# object-oriented programming, Apress.

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

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

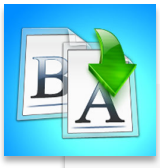
Sharp, John, (2009), Microsoft Visual Studio 2008 Step By Step, çev. Ümit Tezcan, Arkadaş Yayınevi, Ankara.

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

Taşdelen,Aykut,(2010),C# ile Veritabanı Programlama ve Ado.Net, Pusula Yayıncılık, İstanbul.

Türkoğlu, Tansu, (2009), Profesyonel Programlama Teknikleri .Net, Kalitte Bilgi Teknolojileri Basım ve Yayıncılık, Ankara.

KONSOL UYGULAMALARI



İÇİNDEKİLER

- Giriş
- Konsol Ekranını Açma
- Konsol Ekranında Değer Yazma
- Konsol Ekranından Değer Okuma
- Konsol Ekranında Matematiksel İşlemler
- Konsol Ekranında Yeni Module ve Class Ekleme
- Konsol Ekranına Windows Formları Ekleme

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Konsol ekranını açabilecek,
  - Konsol ekranında mesaj verdirebilecek,
  - Konsol ekranında girilen bir değeri değişken içinde tutabilecek,
  - Matematiksel işlemleri gerçekleştirebilecek,
  - Yeni module ve classlar ekleyebilecek,
  - Konsol ekranında windows formlarla çalışabileceksiniz.

ATATÜRK ÜNİVERSİTESİ
ATA-AÖF

GÖRSEL PROGRAMLAMA II

Okt. Daha ORHAN

ÜNİTE

5

GİRİŞ

Önceki konularda Visual Studio.Net platformu üzerinde Visual Basic Programlama dilini seçtikten sonra sıklıkla kullandığımız Windows form uygulamaları yardımı ile kod yazabileceğimiz görsel özellikleri olan yapılardan bahsetmiştik. Bu ünitede bunları yapabileceğimiz konsol ekranı üzerinde duracağız. DOS penceresinde görsel öğelere yer vermeden çalışmaya hazır .NET uyumlu uygulamalara konsol uygulamaları denmektedir.

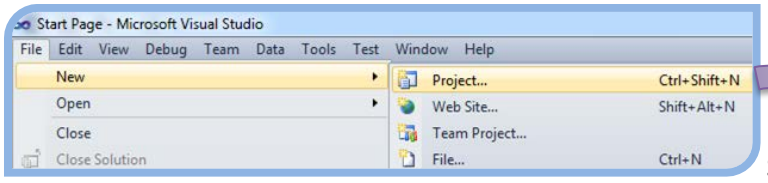
.Net uyumlu DOS penceresinde çalıştırılabilen uygulamalara konsol uygulaması denilmektedir.

Konsol uygulamalarını kullanmak, görsel ortamda hazırlanan projelere istinaden pek avantajlı görünmeyebilir. Fakat konsol uygulamaları sayesinde program geliştirmenin arka planında gerçekleşen olaylar biraz daha rahat bir şekilde anlaşılacaktır.

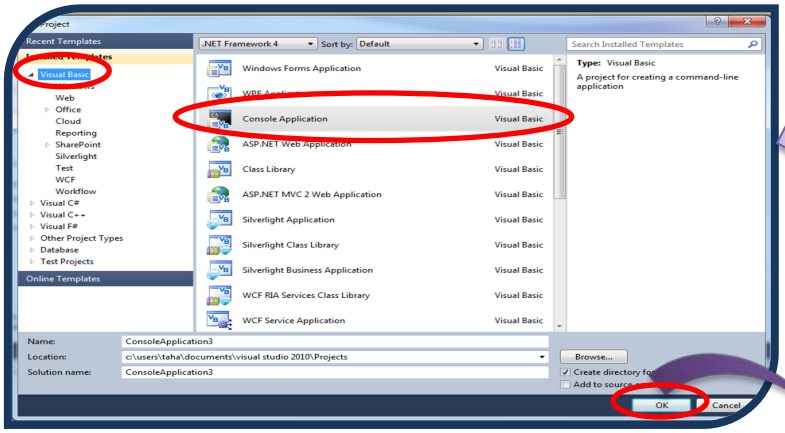
Hazırlayacağımız uygulamaları not defteri yardımıyla derledikten sonra hayata geçirmek de mümkündür. Ancak bu ünitede Visual Studio içinde barınan Console Application şablonundan yararlanarak yapacağız.

KONSOL EKRANINI AÇMA

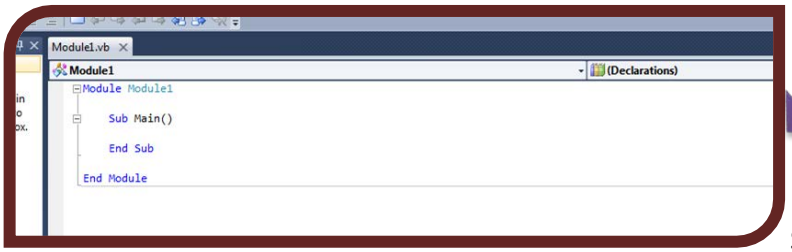
Visual Studio 2012 programını başlattıktan sonra karşımıza gelen ekranda sırasıyla File -> New -> Project menü yolunu veya ctrl + shift + N kısayol (Şekil-1a) tuşunu kullanarak açılan New Project penceresinde Installed Templates sekmesinden visual basic programlama dilini ve Console Application(Şekil-1b) seçeneğini işaretleyip ok butonuna bastıktan sonra projemizi açabiliriz.



Şekil-1a



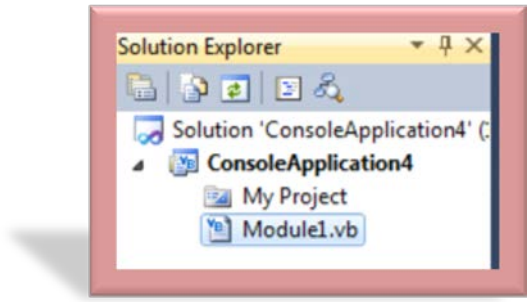
Şekil-1b



Şekil-1c

Yukarıdaki adımları takip ettikten sonra karşımız şekil1-c'deki ekran gelecektir. Bu alan içinde yapmak istediğimiz işlemleri main() fonksiyonu içine yazarak çalıştırabiliriz.

Açtığımız uygulamayla ilgili seçenekleri Solution Explorer penceresinde görebiliriz (Şekil-2). İlk etapta bu kayan panel üzerinde My Project ve Module1.vb seçenekleri bulunmaktadır. Eğer tüm seçenekleri görmek istiyorsanız Show All Files butonuna tıklayabilirsiniz.

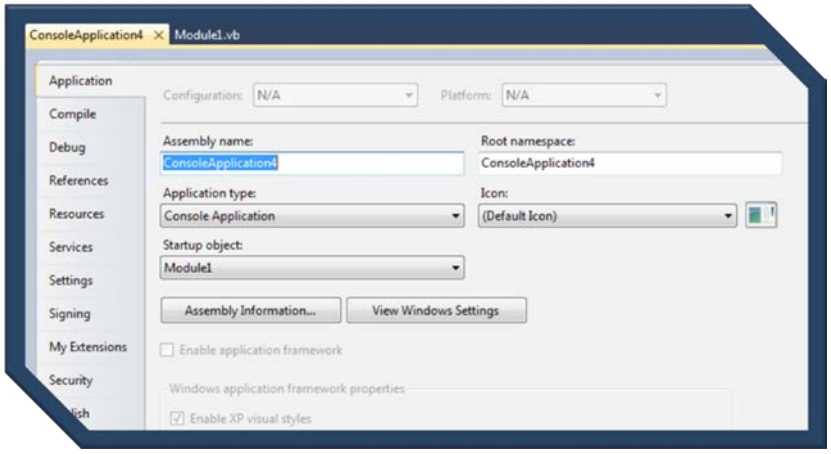


Şekil – 2

Hazırlayacağımız uygulama ile alakalı bilgileri içeren çok sayıdaki dosya My Project seçeneğinde bulunmaktadır.

Hazırlayacağımız uygulama ile alakalı bilgileri içeren çok sayıdaki dosya My Project seçeneğinde bulunmaktadır. Module1.vb ise Şekil-1c'deki kod dosyasına işaret etmektedir.

Solution Explorer penceresinde My Project seçeneğine çift tıklayarak veya menü yolu olarak Project → Console Application Properties… seçeneğine tıklayarak uygulama ile ilgili alternatif özelliklere ulaşabileceğimiz bir sekmeye ulaşabiliriz (Şekil-3).



Şekil – 3

Şekil-3'teki alan üzerinden hazırlanan projenin ikonu değiştirilebilir veya birden fazla uygulama içeriyorsa bunlardan hangisi ile projenin başlatılacağı belirlenebilir veya benzeri özellikler kullanılabilir.

KONSOL EKRANINDA DEĞER YAZMA

Bu bölümde Visual Studio programının konsol ekranında görsel ögeleri kullanmadan mesaj verme konusundaki bilgilere değineceğiz. Bununla alakalı olarak iki metot bulunmaktadır. Bunlardan biri Write metodu diğeri ise Writeline metodudur. Write metodunu kullanarak mesaj verdiğimiz zaman bu metottan

sonra gelecek işlemler çıktı ekranında aynı satır üzerinden devam edecektir. Writeline metodunda ise sıradaki işlemler bir alt satırdan devam etmektedir. Kullanım yöntemleri:

```
Console.Write("MESAJ")

Console.WriteLine("MESAJ")
```

şeklindedir. Konunun daha iyi anlaşılması için aşağıdaki iki farklı uygulamayı inceleyelim.

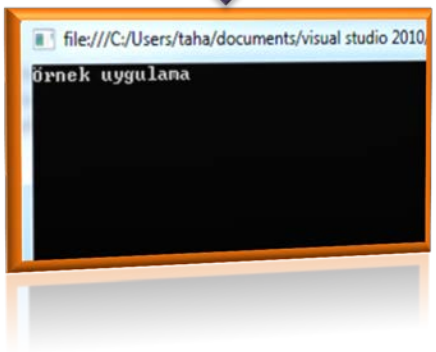
Write() metodu ile mesajlar yanyana yazdırılırken, WriteLine() metodu ile mesajlar alt alta yazdırılır.

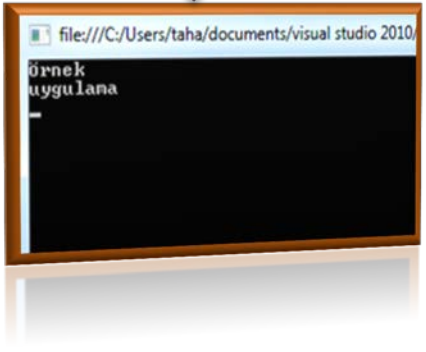
Örnek

•Visual Studio programını kullanarak consol application şablonu ile DOS ekranında Örnek uygulama çıktısını yan yana ve alt alta veren kod gösterimi ve ekran görüntüsü aşağıdaki gibidir.

Yan yana çıktı veren kod satırları

```
Module Module1
    Sub Main()
Console.Write("Örnek ")
Console.Write("uygulama")
Console.ReadLine()
    End Sub
End Module
```





Write metodu veya WriteLine metodu ile bir veya birden fazla değişkenin içeriği yazdırılabilir. Metot kullanılırken tırnak kapatıldıktan sonra virgül konularak değişken isimleri sırası ile aralarına virgül konularak yazılır. Tırnak içinde değişkenleri işaret eden {0} {1} … deyimleri kullanılır. Aynı kod satırı içinde birden fazla veri tipine ait değişken içeriğini yazdırmak mümkündür.

Kullanım yöntemi:

Console.Write(“ilk içerik {0} diğer içerik {1}”,birinci_degisken,ikinci_degisken)

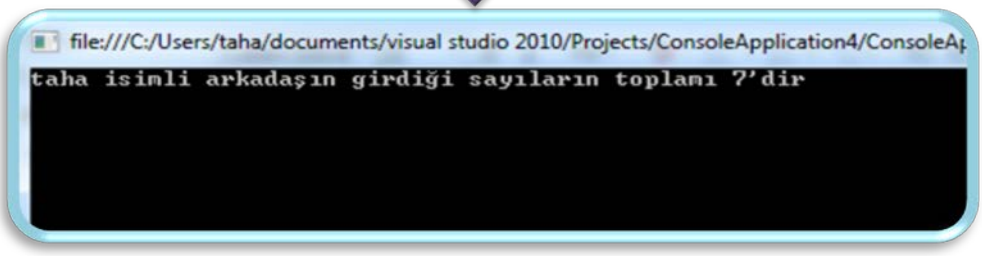
şeklindedir.

Örnek

•Konsol ekranında değişken içeriğini ekrana yazdıran örnek kod gösterimi ve ekran görüntüsü aşağıdaki gibidir.

```
        a, b ve isim değişkenlerine kullanıcı sırasıyla 3, 4 ve taha değerlerini
    girdiğini varsayalım
Module Module1
    Sub Main()
        Dim a, b, t As Integer
        Dim isim As String
        Console.Write("ilk sayı: ")
        a = Console.ReadLine()
        Console.Write("ikinci sayı: ")
        b = Console.ReadLine()
        Console.WriteLine("isminiz nedir:")
        isim = Console.ReadLine()
        t = a + b
        Console.Clear()
        Console.Write("{0} isimli arkadaşın girdiği sayıların
toplamı {1}'dir", isim, t)
        Console.ReadLine()
    End Sub
End Module
```

Ekrandakileri temizleyen kod



Konsol ekranında zemin rengi veya yazı tipiyle ilgili farklı seçenekleri gerçekleştirebiliriz.

Konsol ekranında çalışmalarımızı gerçekleştirirken aklımızda sadece siyah beyaz bir ortamın olduğu gelmemelidir. Arka plan rengi, yazı rengi gibi değişiklikleri de uygulayabilmemiz mümkündür. Örneğin konsol arka planının mavi yazıların ise beyaz olduğu bir uygulama yapalım.

```
Module Module1
    Sub Main()
Console.BackgroundColor = ConsoleColor.Blue
Console.ForegroundColor = ConsoleColor.White
Console.Clear()
Console.WriteLine("Takımınız")
Console.ReadLine()
    End Sub
End Module
```

Yukarıdaki uygulamada Console.Clear() komutunu kullanmamızın sebebi konsol ekranının tamamında renk geçerliliğini sağlayabilmektir. Aksi takdirde vermiş olduğumuz arka plan rengi sadece yazının altında kalan kısım için geçerli olacak ve diğer alanlar yine siyah olarak kalacaktır.

KONSOL EKRANINDAN DEĞER OKUMA

Değer okuma işlemi gerçekleştirilirken kullanılan metot ve üzerine yazılacak değişkenin veri tipi arasındaki bağlantının önemli olduğunu unutmayınız.

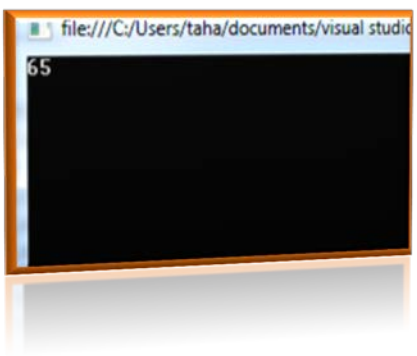
Programı hazırlarken çalışma esnasında Write metodu ile konsol ekranında yazdığımız yazıların nasıl görüntüleneceğini öğrendik. Şimdi ise konsol ekranında yani programın çalışması esnasında klavyeden gireceğimiz verileri nasıl tutacağımızı inceleyeceğiz.

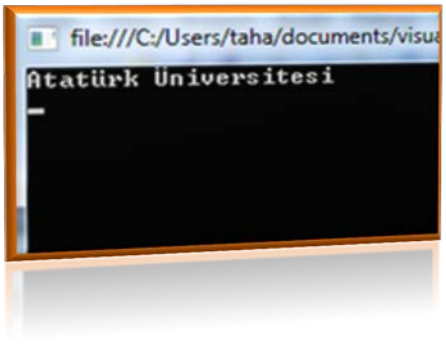
Değer okumak için genelde Readline metodunu kullanmak avantajlı olacaktır. Tabi almak istediğimiz değerin veri tipine bağlı olarak veya hazırlayacağımız projenin özelliklerine binayen farklılıklar gösterebilir. Eğer okutmak istediğimiz değer string bir ifadeyse bu durum karşısında Read metodunu kullanmak hatalı sonuç doğuracaktır. Öyleyse almak istediğimiz ifadeyi ReadLine metodunu kullanarak çekmek daha doğrudur. Konunun daha iyi anlaşılması için aşağıdaki iki farklı uygulamayı inceleyelim.

Örnek

•Consol ekranında kullanıcıdan okuduğu üniversitenin istenildiği programın Read ve ReadLine metodu kullanılarak alındığı kod gösterimi ve ekran görüntüsü aşağıdaki gibidir.

```
Module Module1
    Sub Main()
    Dim egitim As String
Console.WriteLine
("Okuduğunuz üniversite?")
egitim = Console.Read()
Console.Clear()
Console.WriteLine(egitim)
Console.ReadKey()
    End Sub
End Module
```





Yukarıdaki örneklerde de görüldüğü gibi “egitim” değişkeni içindeki değeri read metodu ile aldığımız zaman; değişken içine girilen değerin ilk karakteri okunur ve mesajda da görüldüğü gibi bu değerin ASCII karşılığı olan 65 sonucunu karşımızda gösterir. Readline metodu ile değeri aldığımız durumda ise mesajı olduğu hâliyle karşımızda göstermektedir.

Console.ReadLine() metodunu kullanarak okutmak istediğiniz karakter sayısını artırmak istiyorsanız, Buffer değeri üzerinde değişiklik yapmanız gerekmektedir.

Not: Farklı bir işlem yapılmadığı takdirde Console.ReadLine metodu kullanılarak okutulabilecek maksimum karakter sayısı 256’dır.

Aynı zamanda yukardaki örneği çalıştırdığımızda mesaj verdirildikten sonra karşımızdaki ekranın beklediğini görmekteyiz. Bunu sağlayan kod satırı “Console.ReadKey” dir. Klavyeden basılan herhangi bir tuşun okunmasını sağlamaktadır. Bizim örneğimizdeki kullanım amacı ise herhangi bir tuşa basılana kadar mesaj ekranını bekletmektir.

KONSOL EKRANINDA MATEMATİKSEL İŞLEMLER

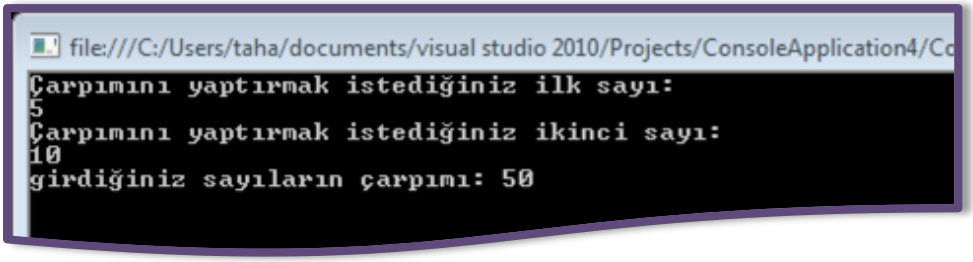
Değer yazma ve değer okuma işlemleri inceledikten sonra uygulamalarımıza matematiksel işlemleri de dâhil etmek faydalı olacaktır. Burada dikkat etmemiz gereken unsur tanımladığımız değişkenin veri tipidir. Bu durum karşısında dönüştürme işlemlerini kullanmamız gerekebilir.

Konunun daha iyi anlaşılması için aşağıdaki uygulamayı inceleyelim.

Örnek

•Konsol ekranında kullanıcı tarafından girilen iki sayının çarpımını gösteren örnek kod ve ekran görüntüsü aşağıdaki gibidir.

```
Module Module1
  Sub Main()
    Dim a, b, sonuc As New Integer
Console.WriteLine("Çarpımını yaptırmak istediğiniz ilk sayı:")
a = Console.ReadLine()
Console.WriteLine("Çarpımını yaptırmak diğer sayı:")
b = Console.ReadLine()
sonuc = a * b
Console.WriteLine("girdiğiniz sayıların çarpımı: " & sonuc)
Console.ReadKey()
  End Sub
End Module
```

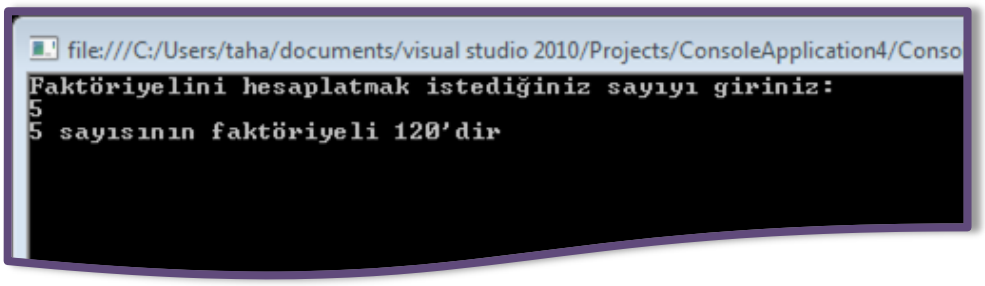


Şimdi program oluşturarak klavyeden girilen bir sayının faktöriyelini hesaplayan bir proje yapacağız. Burada tür dönüşümlerinden "Convert.ToInt32" kullanılacaktır. Çünkü aktaracağımız değişken tam sayı tipindedir.

Örnek

•Konsol ekranında kullanıcı tarafından girilen bir sayının faktöriyelini hesaplayan programın örnek kod gösterimi ve ekran görüntüsü aşağıdaki gibidir.

```
Module Module1
    Private Function Faktoriyel(ByRef sayi As Integer) As
Integer
        Dim sonuc = 1, i As Integer
        For i = 1 To sayi
           sonuc = sonuc * i
        Next
        Return sonuc
    End Function
    Sub Main()
        Dim a, git As New Integer
        Console.WriteLine("Faktöriyelini hesaplatmak
istediğiniz sayıyı giriniz:")
        a = Convert.ToInt32(Console.ReadLine())
        git = Faktoriyel(a)
        Console.WriteLine("{0} sayısının faktöriyeli {1}'dir
", a, git)
        Console.ReadKey()
    End Sub
End Module
```



Konsol ekranında matematiksel işlemleri gerçekleştirirken tür dönüşümlerini kullanmanız gerekebilir.

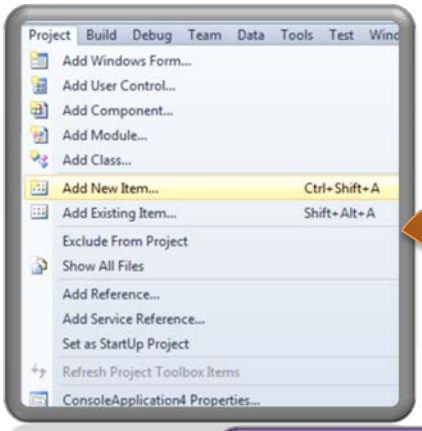
Örnekler incelendiği zaman konsol uygulamalarında da birçok matematiksel işlemin gerçekleştirilebileceği görülmektedir. Hazırlanacak olan projenin amacı doğrultusunda bu örnekleri çoğaltmak mümkündür. Burada önemli olan sayısal değerleri işimize yarayacak ölçüde kullanabilmektir. Reel bir değişken projemizde mevcutsa Convert.ToDouble fonksiyonundan, metin olarak algılanan bir değişken varsa Val fonksiyonundan faydabiliriz. Bu vb. durumları çoğaltmak, tekrar edileceği üzere bizim hazırlayacağımız proje doğrultusunda mümkündür.

KONSOL EKRANINDA YENİ MODULE VE CLASS EKLEME

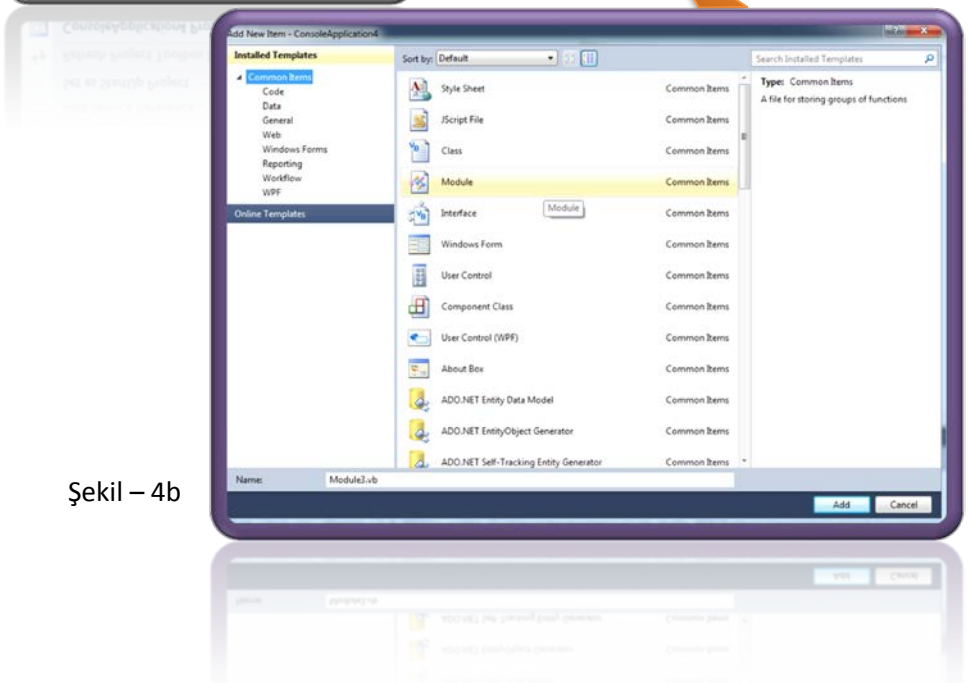
Konsol uygulamalarında birden fazla modue veya class ile çalışmanız mümkündür.

Konsol uygulamalarında çalışırken birden fazla module veya class'a sahip olabiliriz. Bunların arasında işlemler gerçekleştirmemiz mümkündür. Öncelikle projemize yeni bir module ekleyelim.

Bu işlemi gerçekleştirmek için Şekil – 4a'da gösterildiği üzere Project → Add New Item menü yolunu veya Ctrl+Shift+A kısayol tuşunu kullanmamız gereklidir.

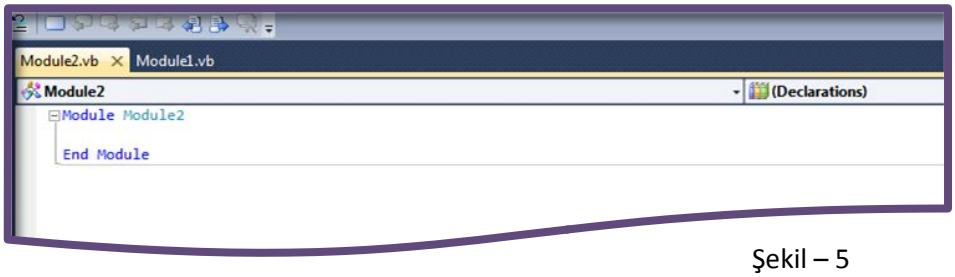


Şekil – 4a



Şekil – 4b

Şekil – 4a'da gösterilen yolu izledikten sonra karşımıza gelen Şekil – 4b'deki pencerede, ilk olarak 1 numaralı adımdaki "module" seçeneğini işaretleyip 2 numaralı adımdaki "Add" butonuna tıklarız. Böylelikle çalışmamıza Şekil – 5'Te gösterildiği gibi bir module daha eklemiş oluruz.



Şekil – 5

Konuyu daha iyi kavrayabilmek için aşağıdaki örnek uygulamayı inceleyelim.

Örnek

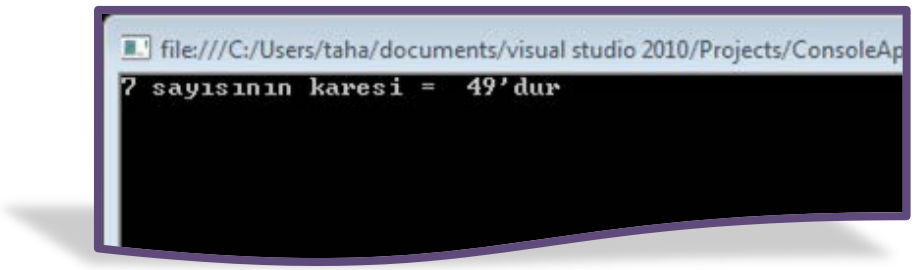
•Konsol ekranında 7 sayısının karesini, eklenen modul2 üzerinde hesaplayan programın örnek kod gösterimi ve ekran görüntüsü aşağıdaki gibidir.

```
Module Module1
    Sub Main()
        Dim a, git As New Integer
        a = Module2.kare(7)
        System.Console.WriteLine("7 sayısının
karesi =  " & a & "'dur")
        System.Console.Read()
    End Sub
End Module

Module Module2
    Function kare(ByRef x As Integer) As Integer
        Return x * x
    End Function
End Module
```

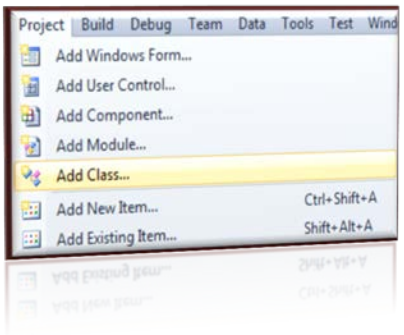
Programı çalıştırdığımız ilk modül

Eklenen ikinci modül

Class eklerken Project menüsündeki Add class komutunu vermek işlemimize hız kazandırır.



Şimdi ise projemize yeni bir class nasıl eklenir ondan bahsedeceğiz. Şekil – 6'da gösterildiği gibi Project → Add Class… menü yolunu kullanırız. Ardından karşımıza gelen pencerede Class seçeneği otomatik olarak seçili olduğundan "Add" butonuna tıklamak projemize yeni bir class eklemek için yeterli olacaktır.



Şekil – 6

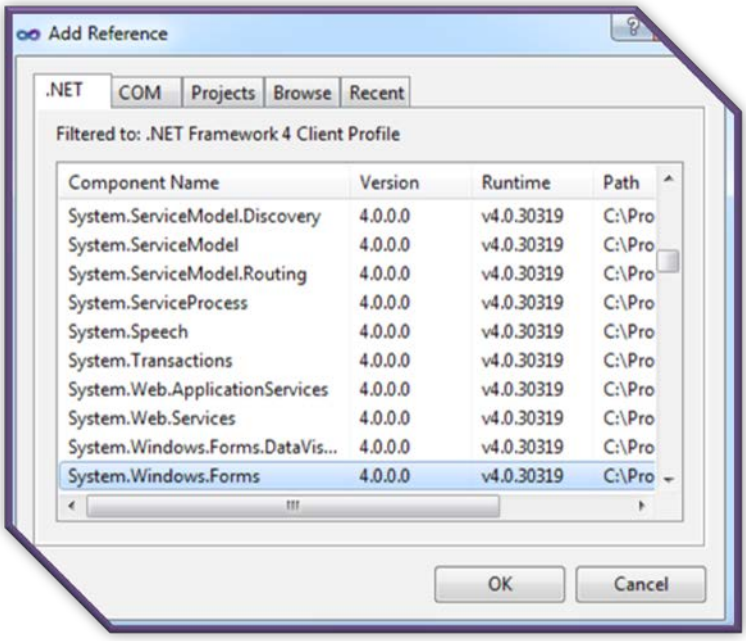
Class'larla alakadar bir uygulama, konsol ekranında Windows formları kullanma başlığı altında gösterilecektir.

KONSOL EKRANINA WİNDOWS FORMLARI EKLEME

Ünitenin başında bahsettiğimiz üzere konsol üzerinde hazırladığımız projeleri DOS ekranında görüntülemekteyiz. Bu durum görsel öğelerden hiç yararlanamama yanılgısını oluşturmamalı. Bu yanılgıyı önlemek için bu başlık altında bir uygulama hazırlamak yerinde olacaktır.

Konsol ekranında Windows formaları kullanabilmek için gerekli olan .dll uzantılı dosyayı referans olarak eklemeliyiz.

Bu uygulamamızı gerçekleştirmek için ilk önce yapmamız gereken bir takım işlemler bulunmaktadır. Konsol ekranını açma başlığı altındayken uygulamayla ilgili seçeneklerin Şekil – 2'de gösterilen Solution Explorer penceresinde bulunduğundan bahsedilmişti. Buradaki bütün bileşenleri görmek için Show All Files butonuna tıklamamız yeterli olacaktır. Bu konuya tekrar değinmemizin sebebi ise Windows formları kullanabilmemiz için gerekli olan referansı eklemektir. Bu işlem için Solution Explorer kayan panelinde uygulama isminin üzerine fare imleci getirildikten sonra sağ tıklanır ve açılan menüden "Add Reference" komutu verilir. Bu adımı takip ettikten sonra karşımıza gelen Şekil – 7'deki pencerede .Net sekmesi altında bulunan System.Windows.Forms seçeneği işaretlenip "OK" butonuna tıklarız.



Şekil – 7

Böylelikle uygulamamızda Windows formları kullanabilmemize imkân tanıyan referansı eklemiş olduk.

Bu adımlar gerçekleştirildikten sonra projemize yeni bir class ekleyelim. Bu class içinde inherits deyiminden yararlanarak Solution Explorer panelini üzerinden eklediğimiz "System.Windows.Forms" adlı namespace kullanılır hâle getirilecektir.

Uygulamamızda bize kolaylık sağlaması için eklediğimizi referansı imports deyimini kullanarak module başında işaret ediyoruz.

Bunları yaptıktan sonra aşağıdaki örnek uygulamayı gerçekleştirmek konunun daha iyi anlaşılmasına yardımcı olacaktır.

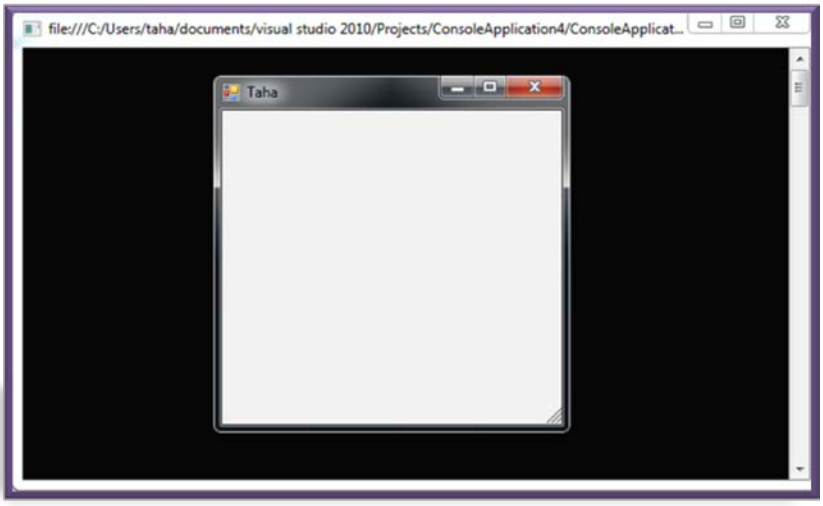
Örnek

•Konsol ekranında windows formlarını kullanan programın örnek kod gösterimi ve ekran görüntüsü aşağıdaki gibidir.

```
Imports System.Windows.Forms
Module Module1
    Sub main()
        Dim t As Class1
        t = New Class1
        t.ShowDialog()
    End Sub
End Module

Public Class Class1
    Inherits System.Windows.Forms.Form
    Public Sub New()
        Me.Text = "Taha"
    End Sub

End Class
```

Programı çalıştırdığımız ilk modül içine yazılan kod satırları

Eklenen Class içine yazılan kod satırları



Form nesnelerini kullanabilmek için class hazırlama mecburiyetimiz yoktur.

Örneğimiz incelendiği takdirde yapılması gereken adımlar tamamlanarak ve yukarıdaki kod satırları module1 ve class1 içine yazıldıktan sonra programımızı çalıştırıp konsol üzerinde form ekranını görüntüleme imkânına sahibiz. Fakat burada dikkat edilmesi gereken nokta şudur: Uygulamamıza bir form nesnesi dâhil etmek için ayrı bir class hazırlamak zorunda değiliz. Konsol ekranında class ekleme başlığı altında gerçekleştirecek uygulamanın burada yapılacağı belirtiği için bu

örnek böyle düzenlenmiştir. Aksi takdirde eklediğimiz Class'ı silip module1 içindeki kod satırlarını aşağıdaki gibi düzenleyerek örneğimiz için aynı ekran görüntüsünü elde etme hakkına sahibiz.

```
Imports System.Windows.Forms
Module Module1
    Sub main()
        Dim t As System.Windows.Forms.Form
        t = New System.Windows.Forms.Form()
        t.Text = "taha"
        t.ShowDialog()
    End Sub
End Module
```

Yukardaki örnek uygulama için module1 üzerinde düzenlenen kod satırları

Bunlarla beraber uygulamamıza class ve module ekleme yollarına benzer şekilde Project menüsü üzerinden Add new item komutu ile Windows form seçeneğini işaretleyip add butonuna tıklayarak yeni bir form dâhil etmek mümkündür. Bu formu görüntülemek için module1 içinde tanımlar ve Show veya ShowDialog metotlarını kullanırız.

Özet

- Visual Studio .NET, hazırlayacağımız projelerde visual basic programlama dilini kullanarak bize DOS ekranında işlemler gerçekleştirebileceğimiz bir ortam sunmaktadır. Bu sayede program geliştirmenin arka planında gerçekleşen olayları daha rahat bir şekilde kavrayabiliriz.
- Konsol ekranını açmak için: Programı başlattıktan sonra File -> New - > Project menü yolunu kullanıp ardından gelen pencerede Console Application seçeneğini işaretlememiz gereklidir. Açılan konsol Ekranında Write veya WritLine gibi metotlar kullanılarak değer yazdırılabilir.
- Kullanıcıya verilen mesajlar yoluyla istenilen değerleri okumamıza imkân sağlayan metotlar da mevcuttur. Bu yolla alınan değerler üzerinde gerekli veri tipleri kullanılarak matematiksel işlemler de gerçekleştirebiliriz.
- Konsol uygulamalarımızı module içinde yazdığımız kod satırları sayesinde F5 fonksiyon tuşu aracılığıyla çalıştırıp gözlemleme imkânına sahibiz. Burada dikkat edilmesi gereken iki unsur bulunmaktadır. Bunlardan birincisi: Uygulamalarımızı tek bir module içine sıkıştırmak zorunda değiliz. Gerekli durumlarda yeni module veya classlar ekleyebiliriz. Bir diğer unsurumuz ise görsel ögelerden tamamen arındırılmış uygulamalar olarak düşünülmemelidir. İstenildiği takdirde uygulamalarımıza gerekli referansları ekleyerek görsel ögelerden de faydalanabiliriz.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. Konsol ekranında hazırlanan bir uygulamada verilen mesajları yan yana görüntülemek için hangi metodu kullanmamız gerekir?
   a) Console.WriteLine()
   b) Console.Clear()
   c) Console.Read()
   d) Console.Write()
   e) Console.ReadLine()

2. Uygulamayla alakalı tüm seçenekleri görmek için Solution Explorer panelindeki hangi butonu kullanmalıyız?
   a) Show All Files
   b) Properties
   c) Refresh
   d) View Code
   e) View Class Diagram

3. Aşağıdaki kod satırlarından hangisi konsol ekranının arka plan rengini kırmızı olarak ayarlamamızı sağlar?
   a) Console.ForegroundColor = ConsoleColor.Red
   b) Console.BackgroundColor = ConsoleColor.Red
   c) t.ShowDialog()
   d) Console.ReadLine()
   e) Imports System.Windows.Forms

4. Farklı bir işlem yapılmadığı takdirde Console.ReadLine metodu kullanılarak okutulabilecek maksimum karakter sayısı kaçtır?
   a) 1024
   b) 512
   c) 256
   d) 100
   e) 2048

5. Tam sayı tipinde matematiksel işlem için kullanılacak bir değişken için hangi sayısal dönüştürme fonksiyonunu kullanmamız gerekir?
   a) Convert.ToByte()
   b) Convert.ToChar()
   c) Convert.ToBoolean()
   d) Convert.ToBase64String()
   e) Convert.ToInt32()

6. Konsol ekranında hazırlayacağımız bir uygulamaya yeni bir module veya class eklemek için hangi menü yolunu kullanabiliriz?
   a) Project → Show All Files
   b) Project → Add New Item
   c) Project → ConsoleApplication1 Properties…
   d) File → Source Kontrol
   e) File → Page Setup

7. Konsol ekranında Windows formları kullanabilmek için aşağıda verilen referanslardan hangisini uygulamamıza dâhil etmemiz gerekir?
   a) System.Windows.Forms
   b) System.Web.Services
   c) System.Transactions
   d) System.Speech
   e) System.ServiceProcess

8. Projemizde birden fazla module'nin bulunduğu bir durumda bunlardan hangisi ile başlayacağımızı belirleyeceğimiz alanı açmak için hangi menü yolunu kullanmamız gereklidir?
   a) Debug → Start Debugging
   b) File → New
   c) Project → Add Class
   d) View → Team Explorer
   e) Project → ConsoleApplication1 Properties…

9. Console.Read() metodunu kullanarak bir değişkenin içine string bir ifadeyi aktarıp programımızı çalıştırdığımızda sonuç ne olur?
   a) Program çalışmaz.
   b) İfade olduğu gibi aktarılır.
   c) İfadenin son karakterinin ASCII karşılığı verilir.
   d) İfadenin ilk karakterinin ASCII karşılığı verilir.
   e) İfade üç defa ekrana yazdırılır.

10. Ekrana değer yazdırmak için kullanılan metot gösterimleri için aşağıdakilerden hangisi doğrudur?
    a) Write(“ilk içerik {0} diğer içerik {1}” ,birinci_degisken ,ikinci_degisken)
    b) ConsoleWrite(“ilk içerik {0} diğer içerik {1}” ,birinci_degisken ,ikinci_degisken)
    c) Console.Write(“ilk içerik {0} diğer içerik {1}” ,birinci_degisken ,ikinci_degisken)
    d) Console.Read(“ilk içerik {0} diğer içerik {1}” ,birinci_degisken ,ikinci_degisken)
    e) Console.Clear(“ilk içerik {0} diğer içerik {1}” ,birinci_degisken ,ikinci_degisken)

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

Aktaş, V. (2010). Visual Basic .Net 10, 1. Baskı, İSTANBUL: Kodlab Yayıncılık.

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

Demirli, N. ve İnan Y. (2010). Visual Basic .Net 2008 Ado .Net 3.0 &Sql Server 2008, ANKARA: Palme Yayıncılık.

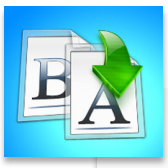
Yazıcı, Ü. (2011). Visual Basic 2008 ile Windows Uygulamaları Geliştirmek, 1. Baskı, İSTANBUL: Pusula Yayıncılık.

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

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

Yanık, M. (2009). Visual Studio 2008 ile Microsoft Visual Basic 9.0 for .NET Frame 3.5, 1. Baskı, ANKARA: Seçkin Yayıncılık.

HATA YAKALAMAK VE AYIKLAMAK



İÇİNDEKİLER

- Hata Yakalamak ve Ayıklamak
  - Hata Ayıklamak
    - Breakpoint
    - Exceptions
  - Hata Yakalama
    - Throw
    - Try, Catch ve Finally Blokları
    - İç İçe Catch Bloğu
    - Genel Hata Yakalama İşlemleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Hata yakalama ve ayıklama ile ilgili bilgi sahibi olabilecek,
  - Try, Catch ve Finally bölümlerinin işlevleri hakkında temel bilgi düzeyine ulaşabilecek,
  - Debug menüsünün işlevini ve önemli bileşenlerini öğrenebileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Uzm. Orhan ÇELİKER

ÜNİTE

6

GİRİŞ

Görsel programlamada gerçekleştirdiğimiz projelerde üzerinde önemle durmamız gereken konulardan biri de hata yakalama ve ayıklama üzerine alacağımız tedbirlerdir. Programlama yaparken en büyük amacımızın hatasız ve doğru çalışan programlar üretmek olduğu düşünüldüğünde; gerçekleştirdiğimiz programların hatalardan arındırılmış olması ve olabildiğince doğru çalışması önemli görülmektedir.

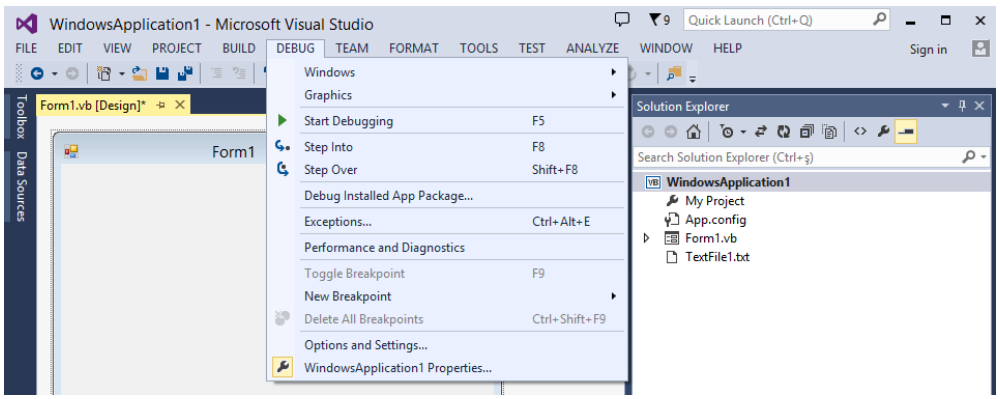
Günlük hayatta birçok ortamda hata yakalama ve ayıklamayla ilgili örnek durumla karşılaşmaktayız. Örneğin bir site için üyelik bilgilerini doldurmamız istendiğinden eksik veya hatalı işlem yaptığımızda hemen uyarı mesajı ile karşılaşmaktayız. Bu bölümde programımızın doğru kullanılması için kullanıcılardan gerçekleştirmesi istediğimiz eylemlerin nasıl belirlenmesi gerektiği ve hataları nasıl tespit edip önlemler alınması gerektiğini göreceğiz.

HATA AYIKLAMA

Debug = Hata Ayıklama

Günümüzde programlama yazılımlarının gelişmesi ve çeşitlenmesi ile programcıların üzerindeki birçok yükü yazılımlar yüklenmektedir. Bunlardan biri de hata yakalama üzerine yazılımların sunmuş olduğu destektir.

Hata ayıklama; geliştiriciler için program hazırlarken önceden belirleyemedikleri veya program çalışırken ortaya çıkabilecek sorunların tespitinde kullanılmaktadır. Bu sayede riskler önceden satın alınarak kullanım esnasında yaşanabilecek aksaklıkların önüne geçilmektedir. Hata ayıklama büyük problemlerin önüne geçmeye imkân tanıdığından kullanılan programlarda bu isimde bir menü bulunmaktadır. Öncelikle programın doğru ve eksiksiz çalışması için bu menünün derinlemesine incelenmesi gerekmektedir (Resim 1).



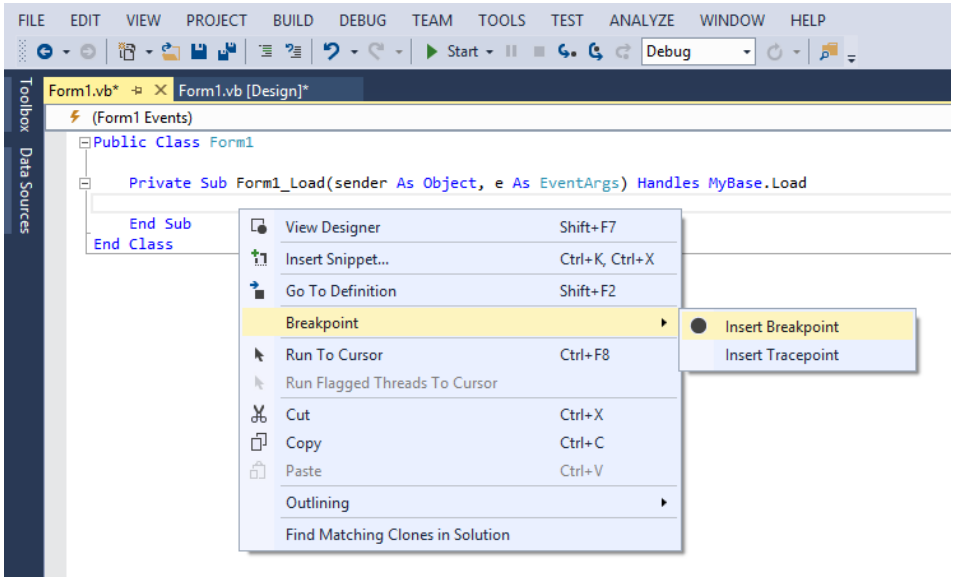
Resim 1. Debug (Hata Ayıklma) Menüsü

Breakpoint

Breakpoint uzun kod bloklarını parçalara ayırarak hata olan kısımların daha rahat görülmesine imkân tanır.

Debug (Hata Ayıklama) menüsünde öncelikle *Breakpoint (kırılma noktası, mola noktası)* yapma kısmı açıklanacaktır. Bu özellik ile çok fazla kod bloklarının parçalara ayrılarak daha rahat hata tespitinin yapılmasına imkân tanınmaktadır. Yaptığımız çalışmaya göre kod blokları bazen içinden çıkılamaz hâle gelebilmektedir. Bu gibi durumlarda *Breakpoint* aracılığıyla bir kırılma noktası oluşturulur ve hata olan satır çok rahat bir şekilde tespit edilebilir.

Bu özelliği kullanmak için kod bloğunun içinde *breakpoint* oluşturacağımız kısma sağ tıklayarak yeni bir kırılma noktası oluşturabiliriz (Resim 2). Ancak hazırlanan breakpoint'lerin hatalar düzeltildikten sonra kaldırılması gerekmektedir.

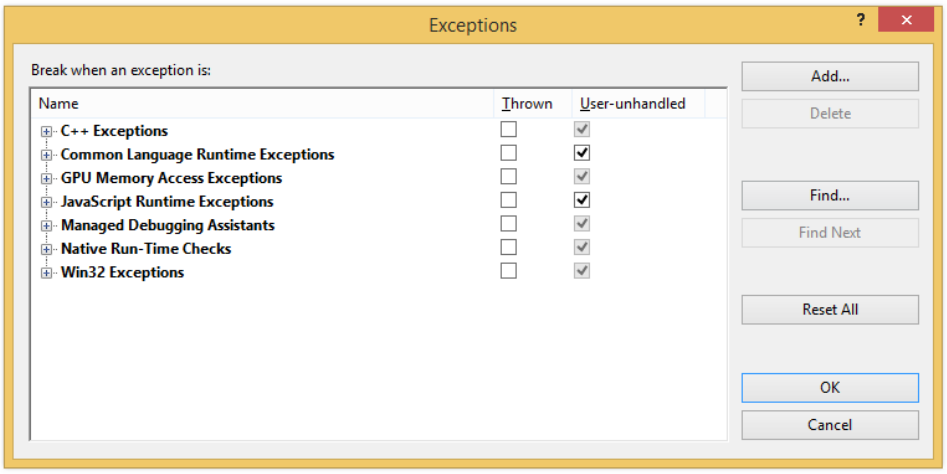


Resim 2. Breakpoint oluşturma

Uzun kod bloklarında birden fazla breakpoint noktası oluşturularak programcının hangi aşamada hata yaptığının tespiti sağlanır. Her kırılma noktası kendi içinde çalışır ve hata yok ise bir sonraki noktaya kadar gidilir. Bu sayede var olan hata programcıya en kısa yoldan ve parça parça gösterilir.

Exceptions

Bu iletişim kutusu ile belirlediğimiz tüm istisnai durumlar listelenmektedir. Bu iletişim kutusu aracılığıyla istisnai durumları aratabilir *(Find)* veya yeni durumlar tanımlanabilir *(Add)*. Ayrıca var olan istisnai durumların tamamını yine kaldırabiliriz *(Reset All)*. Ayrıca *Thrown* altındaki kutucukları işaretleyerek belirlediğimiz istisnai durumların *Try – Catch* bloğu içerisinde çalışmasının da önüne geçebiliriz.

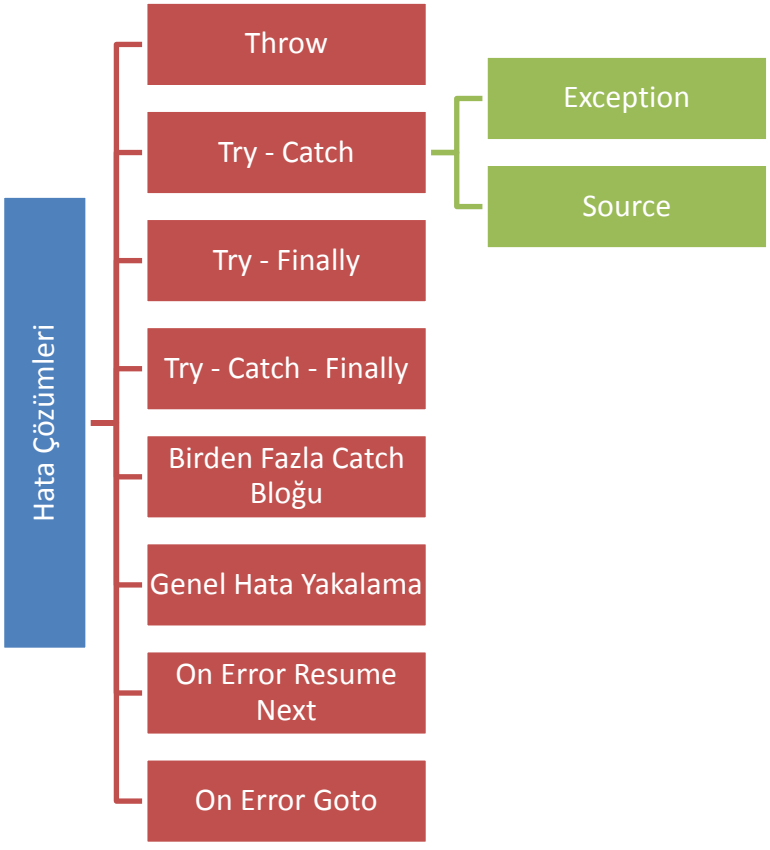


Resim 3. Exceptions Menüsü

HATA YAKALAMA

Bu kısımda geliştirdiğimiz programlarda karşımıza çıkabilecek hatalara yönelik alabileceğimiz tedbirler anlatılmaktadır. Yazılan programlar istenmedik bir şekilde hata vererek kullanılamaz veya gerekli işlemi yapamaz hâle gelmesini önlemek için hata yakalama kodları kullanılmaktadır. Bu kodlar sayesinde kullanıcıya hata ile ilgili bir mesaj kutusu görüntülenir. Bu bağlamda aşağıdaki komutlar anlatılacaktır.

Şekil 1: Hata çözümlerine yönelik gerçekleştirilecek işlemler

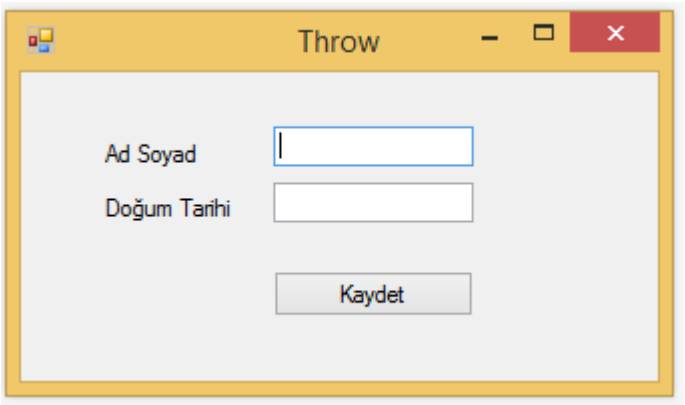


Throw

Throw komutu programın kullanılmasına yönelik gerçekleştirilmiş özel hataların yakalanmasında kullanılan bir komuttur.

Throw komutu programın kullanılmasına yönelik gerçekleştirilmiş özel hataların yakalanmasında kullanılan bir komuttur. Yani yapmakta olduğumuz bir programda herhangi bir alanın boş bırakılmasını önlemek istediğimizi düşünelim. Bu durumda kullanmamız gereken komut *Throw* komutudur. Bu durumu aşağıda belirtilen örnekte daha iyi anlayabilirsiniz.

ÖRNEK – 1



Resim 4. Throw form ekranı

Yukarıdaki programda kullanıcıların "Ad Soyad" ve "Doğum Tarihi" bilgilerine ulaşmak istediğimizi düşünelim. Ancak istediğimiz alanların boş bırakılmamasını istiyoruz. Bu durumda "Kaydet" butonuna tıkladığımızda aşağıdaki belirtildiği gibi kod bloğunu şekillendirmemiz gerekecek.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim Adsoyad As String
        Dim Dogumtarihi As String
        Adsoyad = TextBox1.Text
        Dogumtarihi = TextBox2.Text
        If ((Adsoyad = "") Or (Dogumtarihi = "")) Then
            Throw New Exception("Lütfen boş alanları doldurun")
        End If
    End Sub
End Class
```

Resim 5. Throw kod ekranı

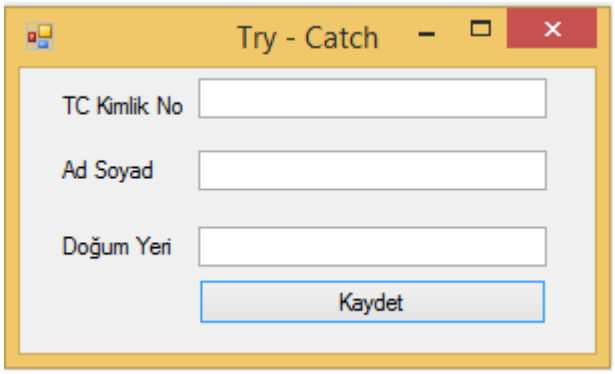
Throw satırından sonraki kod satırı işletilmeyeceğinden buraya farklı bir kodun eklenmesi gereksiz olacaktır. Ancak farklı bir hata yakalama kodu kullanacaksak Throw komutundan sonraki kısma eklenebilir.

Try – Catch

Try-Catch bloğunda hata oluşabilecek kısım Try kısmından sonra hata oluştuğunda işleyecek kodlar ise Catch kısmına yazılır.

En sık kullanılan hata yakalama kod bloklarından biridir. Hazırladığımız programlarda kullanıcıların yanlış veya eksik veri girmelerinden kaynaklanan hataların ayıklanmasında *"Try – Catch"* kod bloğu kullanılmaktadır. Bu blokta hata oluşabilecek kısım *Try* kısmından sonra hata oluştuğunda işleyecek kodlar ise *Catch* kısmına yazılır. Bu durumu aşağıda belirtilen örnekte daha iyi anlayabilirsiniz.

ÖRNEK – 2

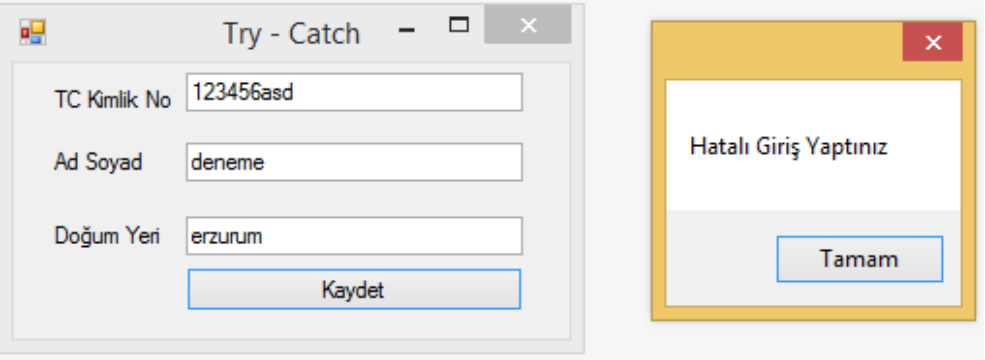


Resim 6. Try - Catch form ekranı

Geliştirdiğimiz programda kullanıcıların kayıt işlemleri için "TC Kimlik No", "Ad Soyad" ve "Doğum Yeri" bilgilerini almak istediğimizi düşünelim. Ancak "TC Kimlik No" kısmında kullanıcıların sadece sayı değeri girmelerini ve bu durum gerçekleşmediğinde onlara bir uyarı göndermemiz gerektiğini düşünelim. Benzer şekilde "Ad Soyad" ve "Doğum Yeri" kısımlarında sadece harf değeri girmelerini istemekteyiz. Bu durumda aşağıdaki şekilde programımızı hazırlamamız gerekmektedir.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim AdSoyad, Dogumyeri As String
        Dim TCKimlikNo As Integer
        Try
            TCKimlikNo = Convert.ToInt32(TextBox1)
            AdSoyad = Convert.ToString(TextBox2)
            Dogumyeri = Convert.ToString(TextBox3)
        Catch
            MessageBox.Show("Hatalı Giriş Yaptınız")

        End Try

    End Sub
End Class
```

Resim 7. Try - Catch kod ekranı



Resim 8. Try - Catch hata uyarısı

Exception

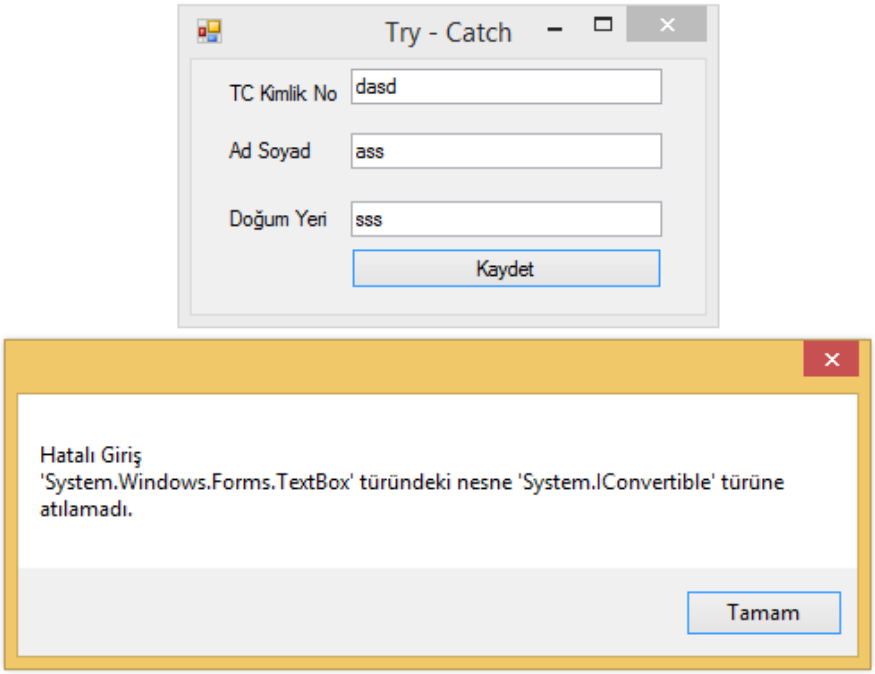
Exception parametresi ile Try – Catch komutunda meydana gelebilecek bir hataya yönelik mevcut sistem mesajı gösterilmektedir.

Bu parametre ile *Try – Catch* komutunda meydana gelebilecek bir hataya yönelik mevcut sistem mesajı gösterilmektedir. Ancak istenildiği takdirde sistem hata mesajı ile birlikte kişinin belirlemiş olduğu mesaj da gösterilebilir.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim AdSoyad, Dogumyeri As String
        Dim TCKimlikNo As Integer
        Try
            TCKimlikNo = Convert.ToInt32(TextBox1)
            AdSoyad = Convert.ToString(TextBox2)
            Dogumyeri = Convert.ToString(TextBox3)
        Catch HataMesaji As Exception
            MessageBox.Show(HataMesaji.Message)
        End Try
    End Sub
End Class
```

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim AdSoyad, Dogumyeri As String
        Dim TCKimlikNo As Integer
        Try
            TCKimlikNo = Convert.ToInt32(TextBox1)
            AdSoyad = Convert.ToString(TextBox2)
            Dogumyeri = Convert.ToString(TextBox3)
        Catch HataMesaji As Exception
            MessageBox.Show("Hatalı Giriş" + vbNewLine + HataMesaji.Message)
        End Try
    End Sub
End Class
```

Resim 9. Exception Kullanımı



Resim 10. Exception hata uyarısı

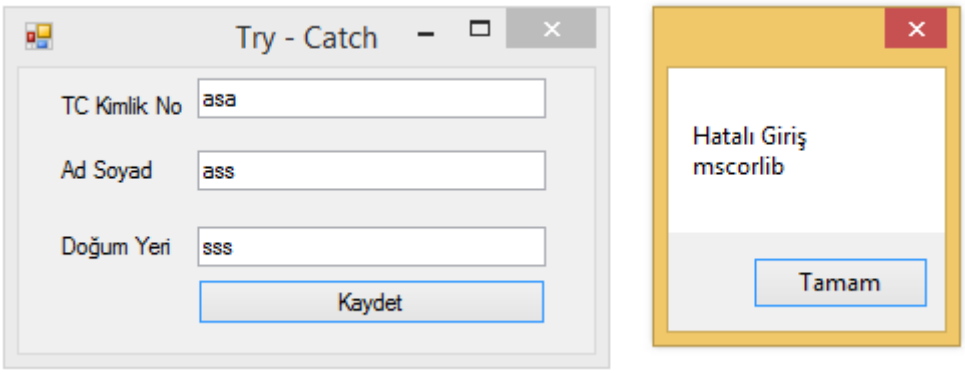
Source

Source parametresi hatanın oluştuğu sınıf veya uygulamayı ifade eder.

Bu parametre hatanın oluştuğu sınıf veya uygulamayı ifade eder. Bazı durumlarda hatanın oluştuğu kaynak dosyanın belirlenmesi önemli olduğundan dolayı bu dosyayı görüntülemek için Source parametresi kullanılmaktadır.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim AdSoyad, Dogumyeri As String
        Dim TCKimlikNo As Integer
        Try
            TCKimlikNo = Convert.ToInt32(TextBox1)
            AdSoyad = Convert.ToString(TextBox2)
            Dogumyeri = Convert.ToString(TextBox3)
        Catch HataMesaji As Exception
            MessageBox.Show("Hatalı Giriş" + vbNewLine + HataMesaji.Source)
        End Try
    End Sub
End Class
```

Resim 11. Source Kullanımı



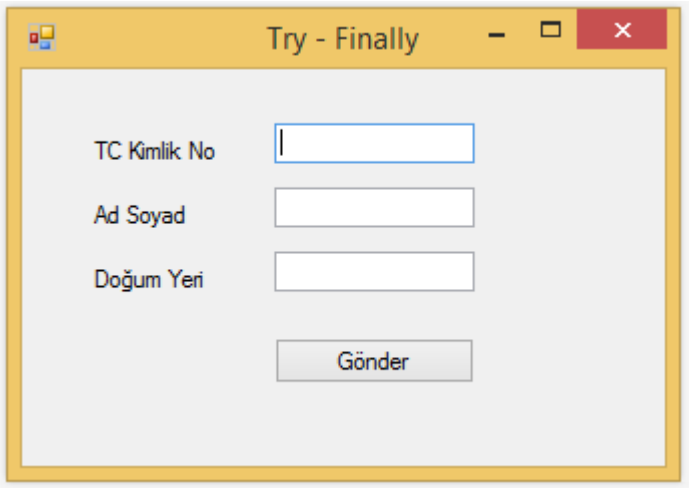
Resim 12. Source hata uyarısı

Try – Finally

Finally kod bloğunda hata olsun ya da olmasın belirttiğimiz durum gerçekleşir.

Bu kod bloğunda hata olsa da olmasa da yapmak istediğimiz herhangi bir işlem var ise onu gerçekleştiririz. Yani *Finally* bloğunu hatadan bağımsız işleyen bir blok olarak düşünebiliriz. Bu durumu aşağıda belirtilen örnekte daha iyi anlayabilirsiniz.

ÖRNEK – 3



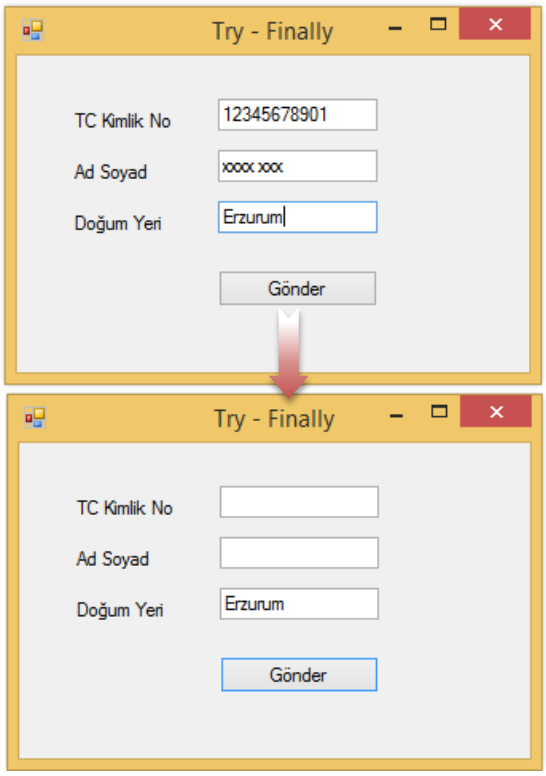
Resim 13. Try - Finally form ekranı

Örnek 2'de belirtilen durum için *Try* kısmından sonra *Finally* bloğunun kullanımı üzerine odaklanalım. Geliştirdiğimiz programda kullanıcıların kayıt işlemleri için "TC Kimlik No", "Ad Soyad" ve "Doğum Yeri" bilgilerini almak istediğimizi düşünelim. Bu bilgiler girildikten sonra Gönder butonuna tıklandığında değer girilen "TC Kimlik No" ve "Ad Soyad" Textbox'ların içindeki verilerin silinmesini istediğimizi düşünelim. Bu durumda aşağıdaki şekilde programımızı hazırlamamız gerekmektedir.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim TCKimlikNo As Integer
        Dim
        Dim
        Try
            Adsoyad = Convert.ToString(TextBox2.Text)
            Dogumyeri = Convert.ToString(TextBox3.Text)
        Finally
            TextBox1.Text = Nothing
            TextBox2.Text = Nothing
        End Try
    End Sub
End Class
```

Unused local variable: 'TCKimlikNo'.

Resim 14. Try - Finally kod ekranı



Resim 15. Try - Finally program çıktısı

Try – Catch – Finally

Aslında kullanım açısından en mantıklı olan kod bloğu *Try – Catch – Finally* kod bloğudur. Çünkü *Try – Finally* bloğunda hata olsun veya olmasın bizim *Finally* bloğunda belirttiğimiz işlemler yapılmaktaydı. Ancak üçünün birlikte kullanıldığı bu yapıda zaten herhangi bir hata meydana gelmediyse *try* bloğundan sonra *finally* bloğu işletilir. Herhangi bir hata meydana geldiyse *try* bloğundan sonra *catch* bloğu işletilir daha sonra *finally* bloğuna geçilir. Bu durumu aşağıda belirtilen örnekte daha iyi anlayabilirsiniz.

Try

Hatanın oluşabileceği kod bloğu

Catch

Hata durumunda çalıştırılacak kod bloğu

Finally

Kodların hatalı veya hatasız olmasına bakılmaksızın çalıştırılacak kod bloğu

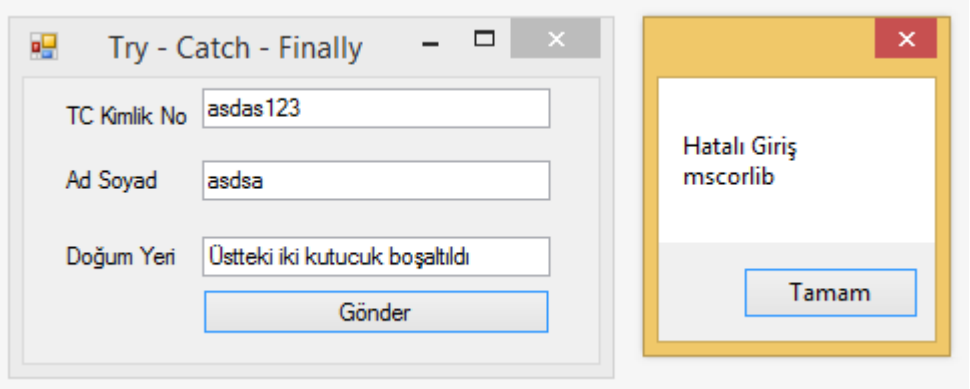
ÖRNEK – 4

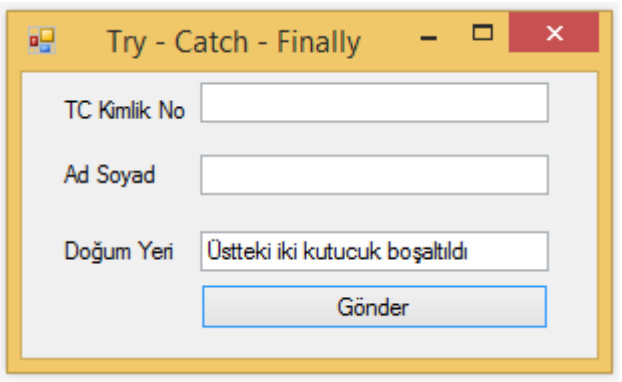
Resim 16. Try - Catch - Finally form ekranı

Kendi sitemiz için bir üyelik formu hazırladığımızı düşünelim. Bu formda kişilerin TC Kimlik No, Ad Soyad ve Doğum Yeri bilgilerine ihtiyacımız olacak. Bu durumda Ad Soyad kısmının harflerden ve TC Kimlik No kısmının rakamlardan oluşması bizim için önemli görülmektedir. Diğer alanlarda kullanıcıların harf veya rakam kullanımı fark etmemektedir. Kullanıcı bilgilerini forma girdikten sonra Gönder butonuna tıkladığından TC Kimlik No ve Ad Soyad kısımlarının boşaltılmasını istiyoruz. Bu durumda aşağıdaki şekilde programımızı hazırlamamız gerekmektedir.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim AdSoyad, Dogumyeri As String
        Dim TCKimlikNo As Integer
        Try
            TCKimlikNo = Convert.ToInt32(TextBox1)
            AdSoyad = Convert.ToString(TextBox2)
            Dogumyeri = Convert.ToString(TextBox3)
        Catch HataMesaji As Exception
            MessageBox.Show("Hatalı Giriş" + vbNewLine + HataMesaji.Source)
        Finally
            TextBox1.Text = Nothing
            TextBox2.Text = Nothing
        End Try
    End Sub
End Class
```

Resim 17. Try - Catch - Finally kod ekranı





Resim 18. Try - Catch - Finally hata uyarısı ve program çıktısı

Birden fazla hatanın muhtemel olduğu durumlarda, hataları tespit etmek için birden fazla catch bloğu yazılır.

Birden Fazla Catch Bloğunun Kullanılması

Bazı uygulamalarda kullanıcıların birden fazla hata yapmaları muhtemeldir. Bu gibi durumlarda hataları tespit etmek için birden fazla *catch* bloğu yazılır. Ancak burada tek tek bütün *catch* satırları işletilmez. Herhangi bir hata var ise o hatayla ilgili *catch* bloğu çalışır ve sonrasında *finally* bloğu kullanıldıysa oradaki işlemler yapılır. Bu durumda aşağıdaki şekilde programımızı hazırlamamız gerekmektedir.

ÖRNEK – 5

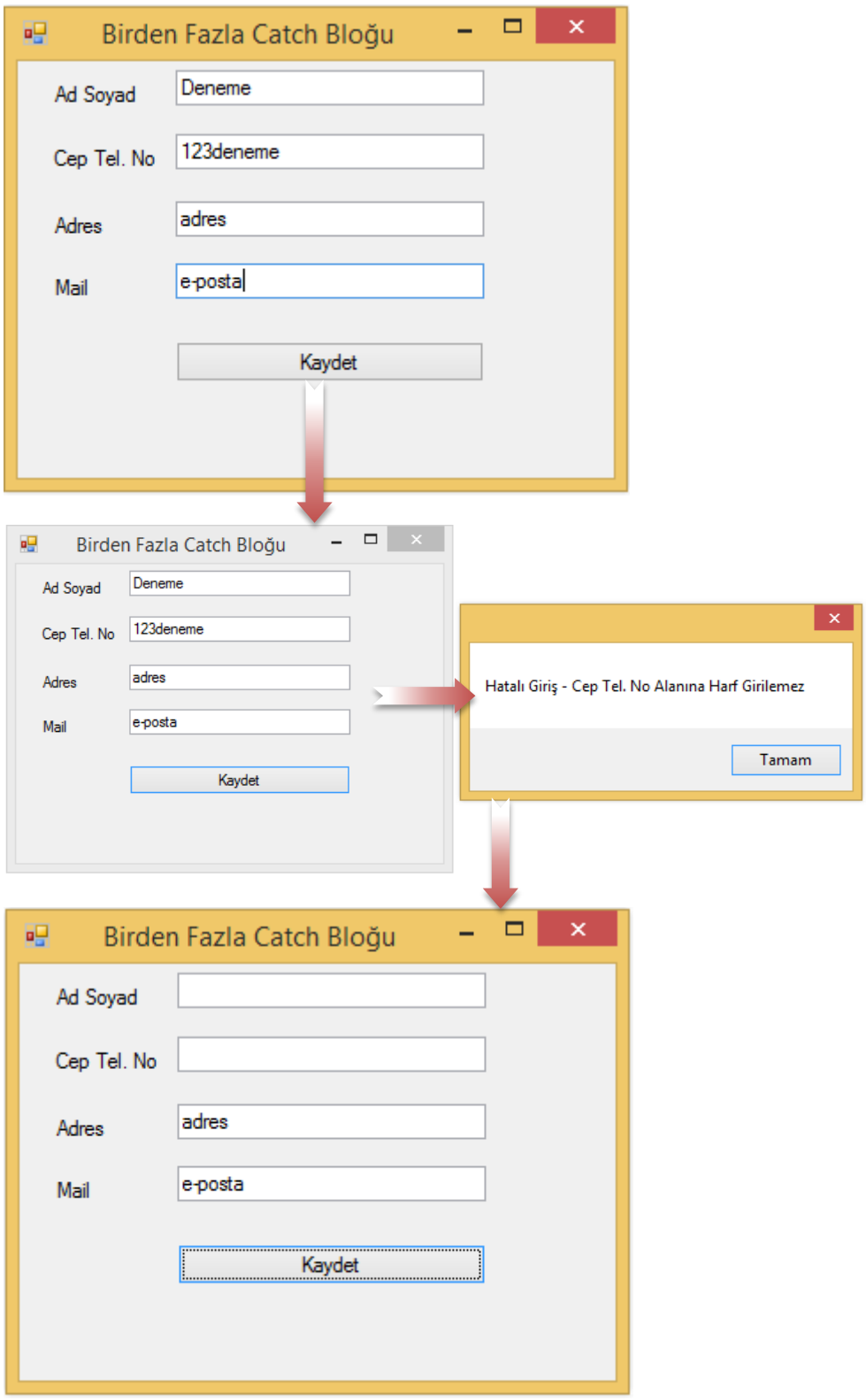
Resim 19. Birden Fazla Catch bloğu form ekranı

Kullanıcıların "Ad Soyad", "Cep Telefonu No", "Adres" ve "Mail" bilgilerini almamız gerekiyor. "Ad Soyad" bölümünün tamamen harflerden oluştuğunu ve "Cep Telefonu No" kısmının rakamlardan oluştuğunu tespit edeceğiz. Son olarak bütün alanlar doğru olarak doldurulduysa *Kaydet* butonuna tıklandığında ilk iki alanın boşaltılmasını sağlayacağız.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim AdSoyad As String
        Dim CepTel As Integer
        Dim Adres As String
        Dim Mail As String
        Try
            AdSoyad = Convert.ToString(TextBox1.Text)
            CepTel = Convert.ToInt32(TextBox2.Text)
            Adres = Convert.ToString(TextBox3.Text)
            Mail = Convert.ToString(TextBox4.Text)
        Catch HataMesaji As FormatException
            MessageBox.Show("Hatalı Giriş - Cep Tel. No Alanına Harf Girilemez")
        Catch HataMesaji As Exception
            MessageBox.Show("Hatalı Giriş - Ad Soyad Alanına Rakam Girilemez")
        Finally
            TextBox1.Text = Nothing
            TextBox2.Text = Nothing
        End Try
    End Sub
End Class
```

Resim 20. Birden Fazla Catch bloğu kod ekranı

Programımızın çalışan kısmında öncelikle bütün alanları harflerden oluşan bir veri girişi yapıp Cep Tel No alanındaki hatanın nasıl çalıştığını görelim. Daha sonra bu hata belirtildikten sonra üstteki iki satır boşaltılacaktır.



Resim 21. Birden Fazla Catch bloğu hata uyarısı ve program çıktısı

Genel Hata Yakalama

Yukarıda bahsedilen yöntemler aracılığıyla uygulama içindeki yerel hatalar kolaylıkla çözülebilir. Ancak bazen bu komutların yetersiz kaldığı durumlar yaşanabilir. Bu gibi durumlarda özellikle programcıların bilmesi gereken ve uygulamanın daha güvenli olmasını sağlayan bir kod bloğunun geliştirilmesi gerekmektedir.

On Error Resume Next komutu ile program içindeki hata satırı atlanır.

On Error Resume Next ve On Error Goto Komutları

On Error Resume Next komutu ile program içinde mevcut olan hata atlanır. Bu komut satırı kullanıldığında hata oluşan satır üzerinde herhangi bir işlem yapılmamaktadır. Bu kodun kullanılması genellikle önerilmemektedir. Çünkü bütün hatalar geçileceğinden dolayı nerede hata yapıldığı tam olarak anlaşılamaz. Bunun yerine *On Error Goto* komutunun kullanılması daha etkili olacaktır. *On Error Goto* komutu ile hata olan satırdan sonra programcının belirlemiş olduğu bir kod bloğu işe koşulur.

Özet

- Bu ünitede hata yakalama ve ayıklama konuları üzerinde durulmuştur. Aslında günlük hayatta bu durumla sürekli karşı karşıya kalmaktayız. Herhangi bir siteye üye olduğumuzda eksik veya hatalı bilgi girdiğimiz takdirde sistem tarafından bize uyarı mesajları verilmektedir. Ünite kapsamında anlattığımız durum ise tam olarak bu konu ile ilgili görülmektedir. Bu bağlamda Throw, Try, Catch ve Finally gibi komutlardan sıklıkla bahsedilmiştir.
- Try; h*atanın oluşabileceği kod bloğunu ifade etmektedir. Buna ek olarak* Catch bloğu ile *hata durumunda çalıştırılacak işlemleri belirleriz. Son olarak* Finally bloğunda ise k*odların hatalı veya hatasız olmasına bakılmaksızın çalıştırılacak işlemler belirtilir.*
- Bu yapı kodlama aşamasında farklı şekillerde sürekli kullanılmakta ve geliştiricilere büyük kolaylıklar sağlamaktadır. Ancak bazen bu durumların yetersiz olduğu ve gözümüzden kaçan durumlar yaşanabilir. Bu gibi durumlarda Exceptions'ların istisnai durumların belirlenmesinde çok önemli bir yeri vardır.
- Ayrıca Uzun kod bloklarında çalışmak geliştiriciler için gerçekten zor bir işlemdir. Oluşabilecek muhtemel hataların kodun hangi kısmında gerçekleştiğinin bulunması zordur. Bu durumda kod bloğunda breakpoint (kırılma noktası) oluşturulur. Kod bloğu parça parça taranarak yaşanabilecek sorunlar ortaya çıkarılır.

DEĞERLENDİRME SORULARI

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

1. **Bir program içindeki hataları yakalamak için aşağıdaki menülerden hangisi kullanılır?**
   a) File
   b) Window
   c) Debug
   d) Test
   e) Tools

2. **Uzun kod bloklarını parçalara bölerek her bir parçada oluşabilecek hataları daha kolay tespit etmeye yarayan özellik aşağıdakilerden hangisidir?**
   a) Exceptions
   b) Breakpoint
   c) Diagnostics
   d) Windows
   e) Graphics

3. **Hata ayıklama işleminde hata oluşabilecek kısmın yazıldığı kod bloğu hangi alandır?**
   a) Try
   b) Catch
   c) Finally
   d) Thrown
   e) Insert

4. **Aşağıdakilerin hangisi hata ayıklama bloğunda hata olsun ya da olmasın yazılan kodun işletildiği kod bloğudur?**
   a) Try
   b) Catch
   c) Finally
   d) Thrown
   e) Delete

5. **Hata ayıklama işleminde hata oluştuğunda işleyecek kodlar aşağıdaki bloklardan hangisine yazılır?**
   a) Try
   b) Catch
   c) Finally
   d) Thrown
   e) Delete

6. Bazı durumlarda hatanın oluştuğu kaynak dosyanın belirlenmesi önemli olduğundan dolayı bu dosyayı görüntülemek için …….. parametresi kullanılmaktadır.

   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

   a) Exception
   b) Thrown
   c) Finally
   d) Catch
   e) Source

7. I. Throw
   II. Try – Catch
   III. Try – Finally
   IV. Try – Catch – Finally

   Yukarıdakilerin hangisi veya hangilerinde programda hata olsun veya olmasın kod bloğunun sonunda belirtilen durum işletilir?

   a) Yalnız I
   b) I ve III
   c) III ve IV
   d) I, III ve IV
   e) I, II, III ve IV

8. **Birden çok Catch bloğu hangi amaçla kullanılır?**
   a) Hataları belirlemek için
   b) Mevcut olan bir hatayı ayrıntılı çözmek için
   c) Programın hatalardan etkilenmesini önlemek için
   d) Birden çok hatanın belirlenmesi ve muhtemel çözümlerin üretilmesi için
   e) Birden çok hata olduğunda programın verileri silmesi için

9. …… komutu ile program içinde mevcut olan hata atlanır.

   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

   a) On Error Goto
   b) Thrown
   c) Finally
   d) Catch Hata As Exception
   e) On Error Resume Next

10. **On Error Goto komutu hangi amaçla kullanılır?**

a) Hata olan satırdan sonra belirlenen kod bloğununun işletilmesi için
b) Hata olmadan önce belirlenen kod bloğununun işletilmesi için
c) Hataları göstermeden direkt atlamak için.
d) Hata oluşmasını engellemek için.
e) Geliştiricinin göremediği muhtemel hataların tespit edilmesi için.

Cevap Anahtarı

1.C, 2.B, 3.A, 4.C, 5.B, 6.E, 7.C, 8.D, 9.E, 10.A

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Aktaş, V. (2013). Her Yönüyle C# 5.0. Kodlab Yayın Dağıtım Yazılım ve Eğitim Hizmetleri. İstanbul.

Demirli N. & İnan Y. (2010). Visual Basic .Net 2008 ADO . Net 3.0 & SQL Server 2008. Palme Yayıncılık. Ankara.

http://msdn.microsoft.com/en-us/library/ms123401.aspx

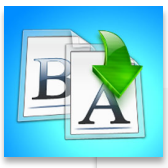
http://msdn.microsoft.com/tr-tr/library/system.exception.innerexception.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1

TEMEL STRING İŞLEMLERİ

ATATÜRK ÜNİVERSİTESİ
ATA-AÖF

GÖRSEL PROGRAMLAMA II

Uzm. Orhan ÇELİKER



İÇİNDEKİLER

- String İşlemleri
  - Lenght
  - Chars
- String Metotları
  - Copy, Clone, Len, Trim, LTrim, RTrim, StrConv, StrReverse, Join, InStr, InStrRev, StrComp, StrDup, Asc, AscW, Chr, ChrW, Val

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - String ifadelerin genel özellikleri hakkında bilgi sahibi olabilecek,
  - String ifadelerde uygulanan metotlar ve bu metotların temel özelliklerini kavrayabileceksiniz.

ÜNİTE

7

GİRİŞ

.Net kodu yazarken String işlemleri büyük önem taşımaktadır. String'ler kelimeler üzerinde yapılan işlemlerde kullanılmaktadır. Gerçekleştireceğimiz çalışmalarda da bu veri tipini sürekli kullanacağımızdan dolayı String ifadeleri ve fonksiyonları çalışmaların doğru sonuçlar üretmesi için önemli görülmektedir.

Bu bölümde temel string işlemleri (string türü değişkenlerle yapılabilecek işlemler) ve string fonksiyonları üzerinde durulacaktır. Ayrıca bu bölümden farklı olarak bölüm 1'de de bazı temel string metotlarını da görebilirsiniz.

STRING İŞLEMLERİ

String'ler genellikle harflerden oluşan verilerdir.

Gerçekleştirdiğimiz çalışmalarda genellikle değişkenler tanımlar ve bu değişkenler üzerinden işlemlerimizi gerçekleştiririz. Tanımladığımız değişkenin türü ve özellikleri daha sonra gerçekleştireceğimiz işlemlerin doğru sonuçlar üretmesinde oldukça önemlidir. String'ler genellikle herhangi bir rakam değeri almayan ve harflerden oluşan verilerdir. Bunu aşağıdaki örnekte görebilirsiniz.

```
Dim Universite As String = "Atatürk Üniversitesi"

                VEYA

Dim Ders As String
Ders = "Görsel Programla II"
```

Örnekte de görüldüğü gibi kelimelerden oluşan bir veri belirlenen değişkene atandı. Bunun tanımlama yerine string sınıfının yapıcı yöntemleri de kullanılabilir.

String'lerin genel olarak iki temel özelliği vardır. Bunun haricinde gerçekleştirilen diğer işlemler stringler üzerinde yapılabilecek işlemlere yöneliktir. Bu özelliklere genel olarak bakacak olursak;

1. **Lenght:** Herhangi bir string ifadenin uzunluğunu vermeye yarayan özelliktir.
2. **Chars:** String ifadede ulaşmak istenilen karekteri getiren özelliktir.

String Metotları

String ifadelerin özelliklerinin belirlenmesinin ardından bu değişkende kullanılabilecek yöntemlerin bilinmesi gerekmektedir. Bu bağlamda tabloda bazı string metotları ve fonksiyonları açıklanmıştır.

Tablo 1. String metotları ve açıklamaları

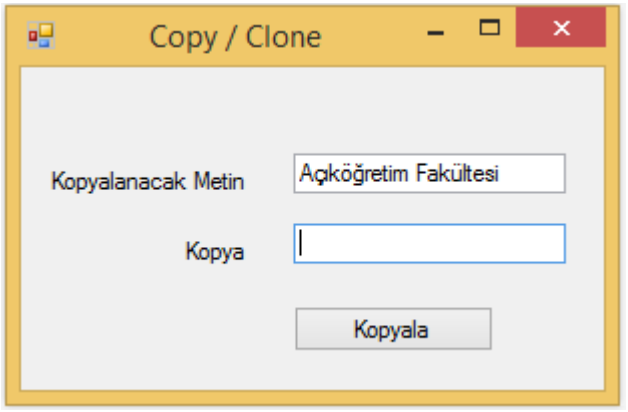
String Metotları	Açıklama
Copy	String bir ifadenin kopyasını oluşturur.
Clone	String değişken ve bu değişkenin bilgilerinin kopyasını oluşturur.
Len	Bir string ifadede boşlukları da sayarak karakter sayısını verir.
Left	Bir string ifadenin solundan başlayarak belirtilen sayıda karakter içeren bir metin oluşturur.
Right	Bir string ifadenin sağından başlayarak belirtilen sayıda karakter içeren bir metin oluşturur.
Trim	Belirtilen string ifadenin başındaki ve sonundaki boşlukları kaldırır.
LTrim	Belirtilen string ifadenin başındaki boşlukları kaldırır.
RTrim	Belirtilen string ifadenin sonundaki boşlukları kaldırır.
ToLower	Belirlenen string ifadenin tamamını küçük harflere dönüştürür.
ToUpper	Belirlenen string ifadenin tamamını büyük harflere dönüştürür.
StrConv	ToLower ve ToUpper metotlarından farklı olarak belirtilen ifadenin ilk harflerini büyük hale getirir.
StrReverse	Belirtilen string ifadenin tersini alır.
Mid	Metinsel ifadede başlangıç noktasından belirtilen noktaya kadar olan karakteri getirir.
Insert	String ifadenin istenilen yerine farklı bir ifade eklemeyi sağlar.
Replace	String ifadenin herhangi bir yerinin farklı bir ifade ile değiştirilmesini sağlar.
Remove	String ifadenin herhangi bir yerinin silinmesini sağlar.
IndexOf	String ifadeler içinde arama yapmayı sağlar. Aranan bilginin başlangıç noktasını gösterir.
Split	String bir ifadenin istenilen sayıda parçalara ayrılmasını sağlar.
Join	Parçalara ayrılmış string bir ifadenin birleştirilmesini sağlar.
InStr	Bir metinsel ifade içindeki farklı bir ifadeyi baştan itibaren arar.
InStrRev	Bir metinsel ifade içindeki farklı bir ifadeyi sondan itibaren arar.
StrComp	İki string ifadenin birbiri ile karşılaştırılmasında kullanılır.
StrDup	Belirtilen karakterden istenilen sayıda türetmeyi sağlar.
Space	Belirtilen sayı kadar boşluk oluşturur.
Asc , AscW	Verilen karakterin ASCII kodunu gösterir.
Chr , ChrW	ASCII'de verilen kodun karakter karşılığını gösterir.
Val(String)	Verilen String ifadeyi sayıya dönüştürür.

String ifadelerin yöntemlerinin çok çeşitli olduğu söylenebilir. Ancak bunların içinden projelerde bize kolaylık sağlayacak ve daha iyi programlar hazırlamaya imkân tanıyacaklardan bazıları aşağıda gösterilmiştir.

Copy ve Clone Metotları

Copy metodu ile string bir ifadenin kopyası oluşturulur.

Bu metotlar ile string bir ifadenin kopyası oluşturulur. Örnek kod bloğunu aşağıda görebilirsiniz.



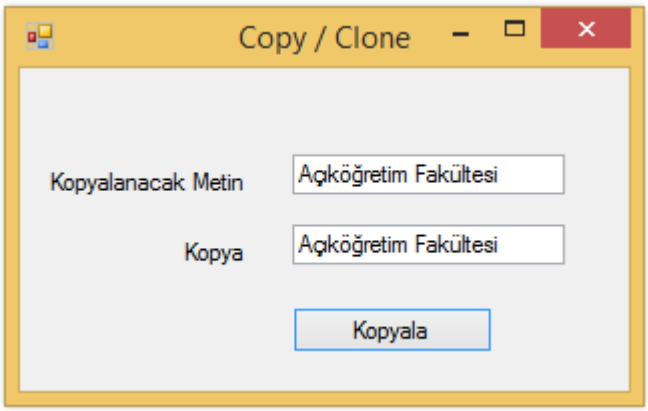

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = Fakulte
        TextBox2.Text = String.Copy(Fakulte)
    End Sub
End Class
```

Resim 1. Copy metodunun kullanımı

veya

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = Convert.ToString(Fakulte.Clone())
    End Sub
End Class
```

Resim 2. Clone metodunun kullanımı

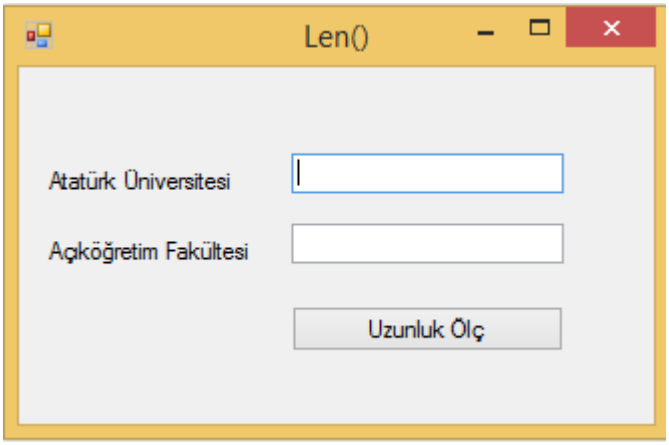


Ancak *Clone* yönteminin yazıldığı kod bloğunda programın çıktısı farklı olacaktır. Program çıktısı olarak sadece *Copy* yönteminin işleyişi gösterilmiştir.

Len metodu string ifadenin karakter sayısı açısından uzunluğunu verir.

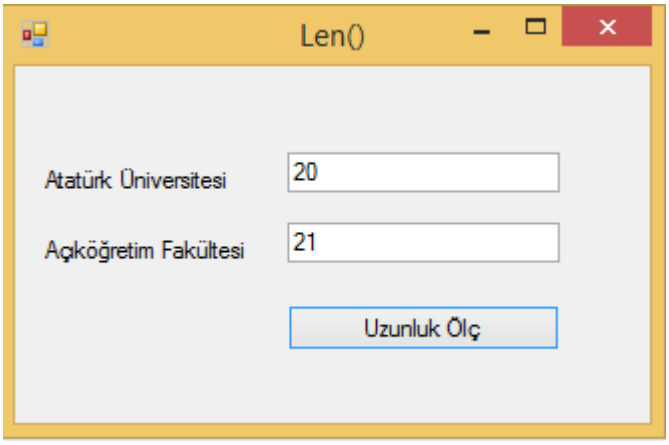
Len Metodu

Bu yöntem ile verilen bir ifadenin karakter sayısı olarak uzunluğu bulunur. Öncelikle belirlenen "Atatürk Üniversitesi" ve "Açıköğretim Fakültesi" ifadelerinin uzunluklarını ölçelim. Bu sayede *Len* metodunun boşlukları da karakter olarak saydığını görebilirsiniz. Örnek kod bloğunu aşağıda görebilirsiniz.




```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "Atatürk Üniversitesi"
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = Len(universite)
        TextBox2.Text = Len(Fakulte)
    End Sub
End Class
```

Resim 3. Len metodunun kullanımı



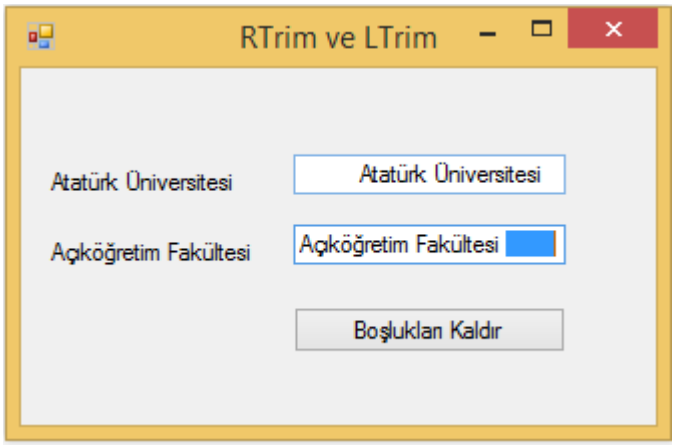
LTrim, RTrim ve Trim metotları string ifadedeki istenmeyen boşlukları alır.

LTrim, RTrim ve Trim Metotları

Genel olarak bu üç metotta da amaç aslında aynıdır. Belirtilen string ifadenin sağındaki, solundaki veya her iki tarafındaki boşlukların alınmasını sağlar. *LTrim* metodu herhangi bir string ifadenin solundaki (başındaki) boşlukların silinmesini sağlarken; *RTrim* metodu sağındaki (sonundaki) boşlukların alınma

sağlar. *Trim* metodu ise ifadenin her iki tarafında bulunan boşlukların alınmasını sağlar. Bu metotlarla ilgili örnek kod bloğunu aşağıda görebilirsiniz.

Öncelikle *LTrim* ve *RTrim* metotları aynı kod bloğu içinde kullanılacaktır. Öncelikle başında ve sonunda boşluk bırakılan ifadeler aşağıda gösterilmektedir. Sonra bu boşlukları kaldırmaya yarayan kod bloğu belirtilmiştir.



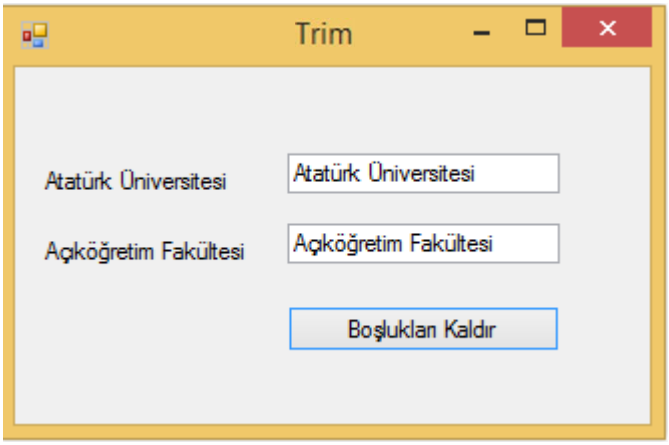

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "          Atatürk Üniversitesi"
        Dim Fakulte As String = "Açıköğretim Fakültesi          "
        TextBox1.Text = LTrim(universite)
        TextBox2.Text = RTrim(Fakulte)
    End Sub
End Class
```

Resim 4. LTrim ve RTrim metotlarının kullanımı

Trim metodunda ise ifadenin her iki tarafında bulunan boşlukların kaldırıldığı belirtmiştir. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "          Atatürk Üniversitesi                "
        Dim Fakulte As String = "         Açıköğretim Fakültesi          "
        TextBox1.Text = Trim(universite)
        TextBox2.Text = Trim(Fakulte)
    End Sub
End Class
```

Resim 5. Trim metodunun kullanımı



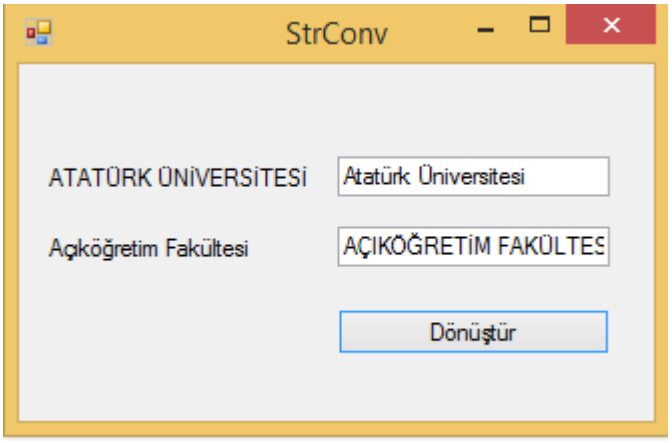
String ifadenin sadece baş harflerini büyütmek için StrConv metodu kullanılır.

StrConv Metodu

ToLower ve *ToUpper* yöntemlerinde olduğu gibi string ifadelerinin büyük veya küçük harflere dönüştürülmesinde kullanılabilir. Bunun yanı sıra belirtilen iki yöntemden farklı olarak bir ifadenin sadece baş harflerinin büyük olması gerektiği durumlarda da kullanılmaktadır. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "ATATÜRK ÜNİVERSİTESİ"
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = StrConv(universite, VbStrConv.ProperCase)
        TextBox2.Text = StrConv(Fakulte, VbStrConv.Uppercase)
    End Sub
End Class
```

Resim 6. StrConv metodunun kullanımı

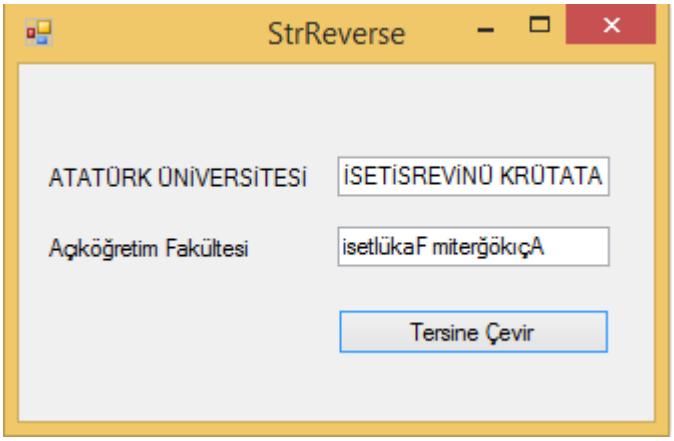


StrReverse Metodu

Bu metot ile belirtilen string ifadenin tersten yazılışı elde edilir. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.


```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "ATATÜRK ÜNİVERSİTESİ"
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = StrReverse(universite)
        TextBox2.Text = StrReverse(Fakulte)
    End Sub
End Class
```

Resim 7. StrReverse metodunun kullanımı



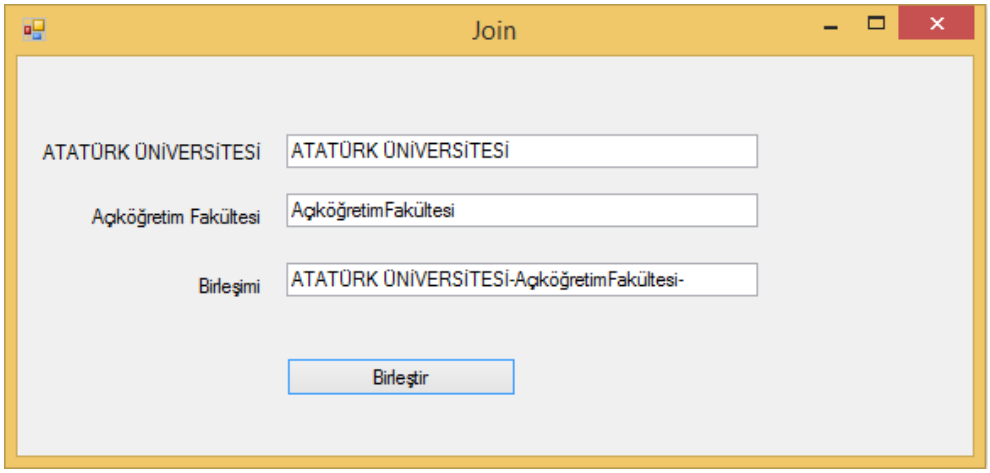
Join Metodu

String bir ifadeyi parçalara ayırmada Split, birleştirmede Join metotları kullanılır.

Join metodu ile split metodunun tersi bir işlem yapılmaktadır. Yani ayrı string ifadeleri birleştirilmektedir. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "ATATÜRK ÜNİVERSİTESİ"
        Dim Fakulte As String = "AçıköğretimFakültesi"
        Dim dizi(2) As String
        TextBox1.Text = universite
        TextBox2.Text = Fakulte
        dizi(0) = TextBox1.Text
        dizi(1) = TextBox2.Text
        TextBox3.Text = String.Join("-", dizi)
    End Sub
End Class
```

Resim 8. Join metodunun kullanımı



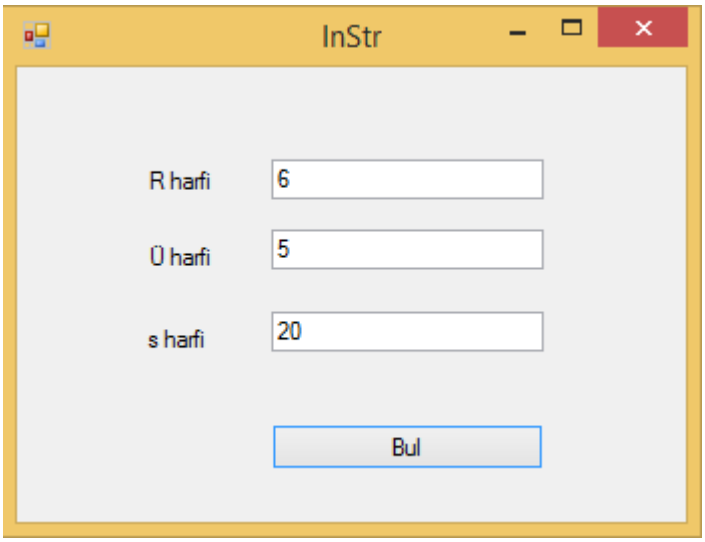
InStr ve InStrRev Metotları

String bir ifadedeki karaktersel bir bilgiyi aramada InStr veya InStrRev Metotları kullanılır.

InStr metodu ile bir metinsel ifade içindeki farklı bir ifade baştan itibaren aranır. Yani herhangi bir string ifadede bulmak istenilen bir harfin yeri bu yöntem kullanılarak soldan sağa arama yapılarak bulunabilir. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "ATATÜRK ÜNİVERSİTESİ"
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = InStr(1, universite, "R")
        TextBox2.Text = InStr(1, universite, "Ü")
        TextBox3.Text = InStr(1, Fakulte, "s")
    End Sub
End Class
```

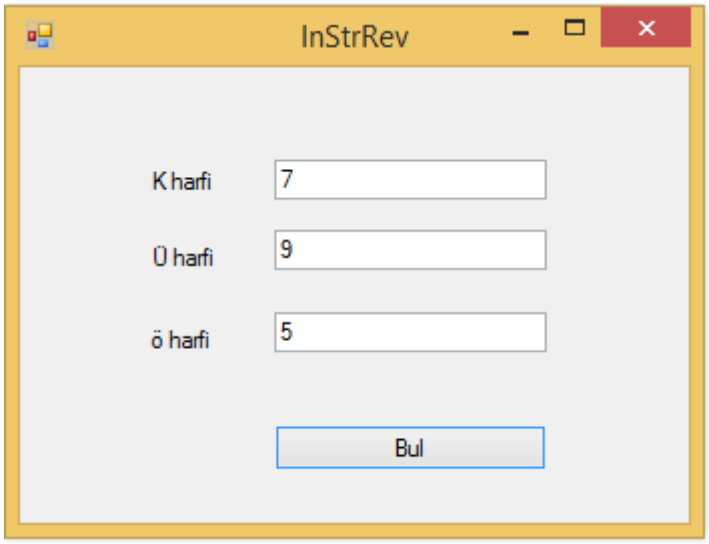
Resim 9. InStr metodunun kullanımı



InStrRev komutu ile de InStr komutunun tersi bir işlem yürütülür. Yani arama soldan sağa değil, sağdan sola doğru yapılır. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim universite As String = "ATATÜRK ÜNİVERSİTESİ"
        Dim Fakulte As String = "Açıköğretim Fakültesi"
        TextBox1.Text = InStrRev(universite, "K", 20)
        TextBox2.Text = InStrRev(universite, "Ü", 20)
        TextBox3.Text = InStrRev(Fakulte, "ö", 21)
    End Sub
End Class
```

Resim 10. InStrRev metodunun kullanımı



StrComp Metodu

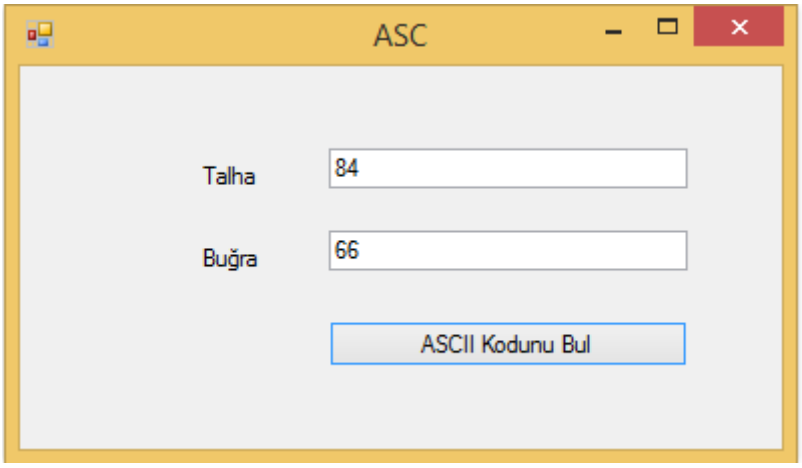
İki string ifadenin birbiri ile karşılaştırılmasında kullanılan bir yöntemdir. Sisteme girilen iki ifadenin aynı veya farklı olma durumları üzerine odaklanmaktadır. Ancak büyük – küçük harf ayrımı yapılmak istenirse CompareMethod.Binary sabitinin verilmesi gerekmektedir.

Asc Metodu

Bu metot herhangi bir string ifadenin ASCII kodunu bulmak için kullanılmaktadır. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim isim1 As String = "Talha"
        Dim isim2 As String = "Buğra"
        TextBox1.Text = Asc(isim1)
        TextBox2.Text = Asc(isim2)
    End Sub
End Class
```

Resim 11. Asc metodunun kullanımı



Chr Metodu

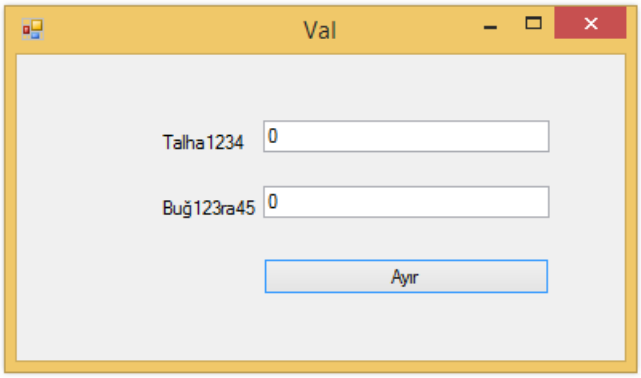
Bu metot ile Asc metodundan farklı olarak herhangi bir sayısal ifadenin ASCII kodu bulunur. Dikkat edilmesi gereken durum 0 – 255 arasındaki sayısal ifadelerin kullanılması gerekliliğidir. 255'ten büyük olan verilerde sistem otomatik olarak hata mesajı gönderecektir.

Val metodu string ifade içindeki rakamların elde edilmesinde kullanılmaktadır.

Val Metodu

Bu metot ile string ifadelerin sayısal ifadeye dönüştürülmesinde kullanılır. Yani string ifadelerden sayısal ifadeleri ayırır. Ancak sayısal ifadeler string olan kısma kadar ilerler. Başlangıçta bir harf varsa sonuç sıfır olur. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

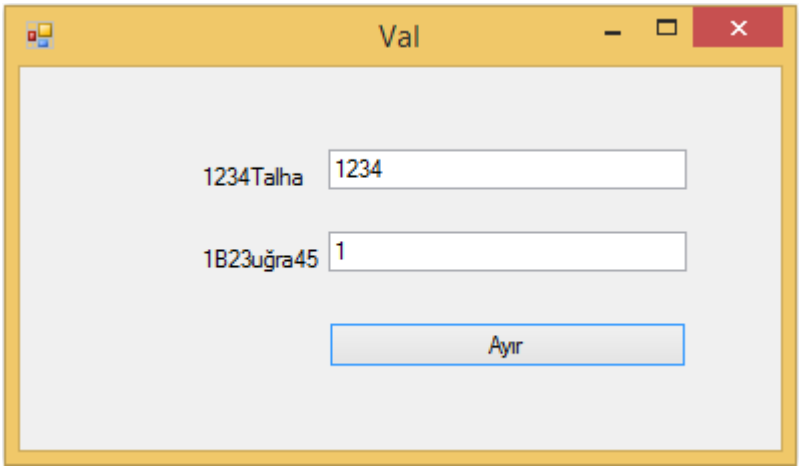
```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim isim1 As String = "Talha1234"
        Dim isim2 As String = "Buğ123ra45"
        TextBox1.Text = Val(isim1)
        TextBox2.Text = Val(isim2)
    End Sub
End Class
```



Resim 12. Val metodunun kullanımı

Ancak başlangıçta bir harf yoksa harf olan kısma kadar içerik alınır. Bununla ilgili örnek kod bloğunu ve program çıktısını aşağıda görebilirsiniz.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim isim1 As String = "1234Talha"
        Dim isim2 As String = "1B23uğra45"
        TextBox1.Text = Val(isim1)
        TextBox2.Text = Val(isim2)
    End Sub
End Class
```



Özet

- Bu ünitede .Net konusunda önemli bir yere sahip olan String işlemleri üzerinde durulmuştur. String'ler genellikle herhangi bir rakam değeri almayan ve harflerden oluşan verilerdir. String'lerin genel olarak iki temel özelliği vardır. Bunun haricinde gerçekleştirilen diğer işlemler stringler üzerinde yapılabilecek işlemlere yöneliktir. Bu özelliklerden Lenght; herhangi bir string ifadenin uzunluğunu vermeye yarayan özelliktir. Bunun yanı sıra Chars ise string ifadede ulaşmak istenilen karekteri getiren özelliktir.
- String'ler üzerinde gerçekleştirilebilecek işlemler genel olarak çok çeşitlidir. Bu işlemler metinsel ifadelerin doğru yönetilmesi ve hazırlanan programların düzgün çalışması için önem arz etmektedir. Genel olarak string ifadeler üzerinde gerçekleştirilen işlemler aşağıda sunulmuştur.
- Copy, Clone, Len, Left, Right, Trim, LTrim, RTrim, ToLower, ToUpper, StrConv, StrReverse, Mid, Insert, Replace, Remove, IndexOf, Split, Join, InStr, InStrRev, StrComp, StrDup, Space, Asc, AscW, Chr, ChrW, Val(String)

DEĞERLENDİRME SORULARI

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

1. **Aşağıdakilerin hangisi Clone metodunun işlevidir?**
   a) Belirtilen string ifadenin başındaki boşlukları kaldırır.
   b) String değişken ve bu değişkenin bilgilerinin kopyasını oluşturur.
   c) Parçalara ayrılmış string bir ifadenin birleştirilmesini sağlar.
   d) İki string ifadenin birbiri ile karşılaştırılmasında kullanılır.
   e) Verilen String ifadeyi sayıya dönüştürür.

2. I. Trim
   II. LTrim
   III. InStr
   IV. IndexOf

   Yukarıdaki string metotlarından hangisi veya hangilerinde temel amaç string ifadede bulunan boşlukları kaldırmaktır?
   a) Yalnız I
   b) Yalnız II
   c) I ve II
   d) I, III ve IV
   e) I, II, III ve IV

3. **String ifadenin herhangi bir yerinin silinmesini sağlayan metot aşağıdakilerden hangisidir?**
   a) Insert
   b) Replace
   c) Join
   d) Remove
   e) RTrim

4. ……. metodu verilen bir string ifadenin solundan başlayarak belirtilen karakter sayısı kadar ifadeden yeni bir metin oluşturmak için kullanılmaktadır

   Cümlede boş bırakılan yere aşağıdaki metotlardan hangisi getirilmelidir?
   a) Right
   b) Mid
   c) Finally
   d) Left
   e) Delete

5. **Bir string ifadenin bütün karakterlerini büyütmek için aşağıdaki metotlardan hangisi kullanılır?**
   a) ToLower
   b) StrConv
   c) Space
   d) StrReverse
   e) ToUpper

6. **Bir string ifadenin tersten yazılmasını sağlayan yöntem aşağıdakilerden hangisidir?**
   a) StrReverse
   b) StrConv
   c) Mid
   d) Replace
   e) Insert

7. **IndexOf metodu hangi amaçla kullanılmaktadır?**
   a) String ifade içindeki karakterler arasına boşluk atar.
   b) String ifadenin bütün harflerinin küçük yazılmasını sağlar.
   c) String ifadenin belirtilen kısmını çıkararak bunun yerine yeni bir ifade ekler.
   d) String ifadenin belirtilen kısmının silinmesini sağlar.
   e) String ifade içindeki bulunması istenilen ifadenin başlangıç noktasını oluşturur.

8. Dim isim1 As String = “Esma”
   Dim isim2 As String = “Betül”
   TextBox1.Text = isim1 + ……. + isim2

   Yukarıdaki kod bloğunda belirtilen iki string değeri arasına istenildiği kadar boşluk bırakmak için aşağıdakilerden hangisi getirilmelidir?
   a) Insert
   b) Add
   c) Remove
   d) Space
   e) Replace

9. **İki string ifadenin birbiri ile karşılaştırılmasını sağlayan metot aşağıdakilerden hangisidir?**
   a) StrComp
   b) StrConv
   c) StrReverse
   d) Asc
   e) Chr

10. **InStrRev metodu hangi amaçla kullanılır**?

a) Bir metinsel ifade içindeki farklı bir ifadeyi baştan itibaren arar.
b) Bir metinsel ifade içindeki farklı bir ifadeyi sondan itibaren arar.
c) String ifadenin ASCII kodunu gösterir.
d) String ifade içindeki sayısal karakterleri ayırır.
e) Ayrı string ifadeleri birleştirir.

Cevap Anahtarı

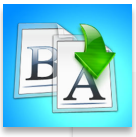
1.B, 2.C, 3.D, 4.D, 5.E, 6.A, 7.E, 8.D, 9.A, 10.B

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

Yanık, M. (2010). Visual Studio 2010 Eşliğinde Microsoft Visual Basic 10 for .Net Framework 4.0. Seçkin Yayıncılık. Ankara.

http://msdn.microsoft.com/en-us/library/ms123401.aspx

VERİTABANI BAĞLANTISI
VERİTABANI TASARIMI



İÇİNDEKİLER

- Temel Veritabanı Kavramları
  - Veritabanı-Veritabanı Yönetim Sistemleri
  - İlişkisel Veritabanı Yönetim Sistemleri
  - Veritabanı Oluşturmak
  - Tablolar
    - Anahtarlar
- ADO.NET'e Giriş
  - Visual Studio ile Veritabanına Bağlanmak
    - SqlConnection
  - Visual Studio.Net ile Veritabanı Tasarlamak

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Temel veritabanı kavramlarını öğrenebilecek,
  - Yeni bir veritabanı ve tablo oluşturabilecek,
  - ADO.NET sınıf kütüphanesini bilecek,
  - Visual Studio ile veritabanına bağlanabilecek ve yeni veritabanı oluşturabileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AOF

GÖRSEL PROGRAMLAMA II

Uzm. Orhan ÇELİKER

ÜNİTE

8

GİRİŞ

Günümüzde bankacılık, ulaşım, telekomünikasyon sistemleri, büyük çaplı otomasyon sistemleri gibi birçok alanda sürekli bir veri akışı olur ve bu verilerin saklanması, özetlenmesi veya gerektiğinde kullanılması istenir. Örneğin; bir bankada müşteri bilgilerinin kayıt altına alınması ve bu kayıtların düzenli bir şekilde saklanarak kolaylıkla takip edilmesi istenir. Bu kayıtların içerisinde müşterilerin kişisel bilgileri, hesap hareketleri gibi bilgiler bulunmaktadır. Veri güvenliği ön planda tutularak bu kayıtlara istenildiği zaman ulaşılabilir ve kayıtlar üzerinde ekleme, değiştirme ve silme gibi birçok işlem yapılabilir. Günümüzde küçük ya da büyük ölçekli tüm kurumlarda veri yönetimi oldukça önemlidir.

Verilere erişim süresinin kısa olması, verilerin tam, tutarlı ve güvenilir olması gerekir. Veriler bilgisayar ortamında saklansa bile kolayca verilere ulaşma, veri tekrarını önleme ve güvenlik açısından basit olan programlar (Ör. Excel) yeterli değildir. Örneğin stok bilgilerini Excel programındaki tablolarda tutan bir işletmeci bilgi girişi esnasında veri kaybı yaşayabilir. Birçok iş el ile yapıldığından basit harf hataları ve buna bağlı tutarsızlıklar olabilir. Verilere erişim ya da farklı biçimler sunma konusunda ciddi sıkıntılar yaşanır. Kullanım ölçeği büyüdükçe ortak veri kullanımı ve anlık ekleme silme işlemleri için daha profesyonel çözümlere ihtiyaç duyulur.

TEMEL VERİTABANI KAVRAMLARI

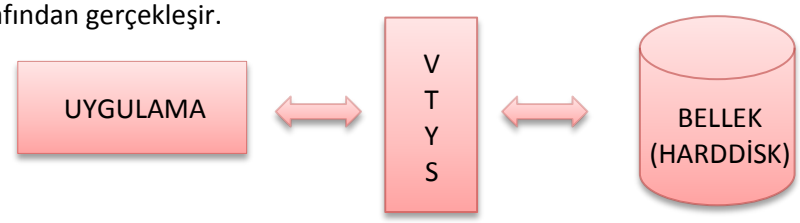
Bir programcı kullandığı veritabanı yönetim sistemi ve veriye erişim teknolojisi fark etmeksizin öncelikle veritabanı ile ilgili temel kavramları belirli bir düzeyde bilmek durumundadır. Bu nedenle aşağıda veritabanı ile ilgili temel kavramlara değinilmiştir.

Veritabanı – Veritabanı Yönetim Sistemi

Veritabanının en önemli özelliklerinden biri verileri birbirleri ile ilişkilendirmesi, tekrara yer vermeden saklanması ve verileri farklı biçimlerde sunabilmesidir.

Bilgilerin bilgisayar ortamı üzerinde düzenli bir şekilde saklandığı bilgi deposuna *veritabanı* denir. Bu bilgi depoları yönetilebilir, taşınabilir, güncelleştirilebilir, eklenebilir veya silinebilir veriler içerir.

Veritabanı, bilgisayardaki düzenli bilgileri ifade ederken bu bilgileri bellek üzerinde organize eden, işleyen, isteklere cevap veren uygulamalara *Veritabanı Yönetim Sistemi - VTYS (DataBase Management System - DBMS)* adı verilir. Verilerin etkin kullanılması, saklanması ve sunulması VTYS sayesinde olur. VTYS veriler ile veriyi kullanan uygulamalar arasında köprü görevi görür. Başka bir ifade ile bir veri doğrudan belleğe yazılıp okunmaz. Taleplere göre bu işlemler VTYS tarafından gerçekleşir.



Şekil 1. Veritabanı Yönetim Sistemi (VTYS)

VTYS; veri tekrarını önler, veri güvenliğini sağlar ve otomatik güncellendiği için tutarlılık sağlar.

VTYS veri alış verişi ve erişimi kontrol etmenin yanı sıra uygulamada birçok avantaj sunar. Bu avantajları aşağıdaki gibi özetlemek mümkündür:

- Veri tekrarını önler.
- Otomatik güncellendiği için tutarlılık sağlar.
- Verilerin paylaşımını kolaylaştırır.
- Aynı zamanda birden fazla kullanıcı ya da uygulama tarafından kullanımına imkân verir.
- Verilerin üzerinde işlem yapma yetkileri sayesinde veri güvenliğini sağlar.
- VTYS'ler veritabanını yöneten uygulamalar ya da veritabanı uygulaması olarak adlandırılır.

Veritabanları başka programlar tarafından veri deposu şeklinde kullanıldığı gibi kendi içindeki araçlar sayesinde tek başına aynı zamanda bir uygulama programı gibi kullanılabilir. İşte bu zaman, veri, finansal kayıpları ortadan kaldırmak ya da en aza indirme ihtiyacı, veritabanın çıkmasına neden olmuştur.

Bilgisayar sistemleri üzerinden hizmet alınan yerlerin nerdeyse tamamında işlemler veritabanı aracılığıyla gerçekleşmektedir. Bu işlemlerde kullanıcılar pek de farkında olmadan veritabanına erişmektedirler. Örneğin bir uçuş rezervasyonu esnasında kullanıcı web üzerinden gideceği yeri ve diğer bilgileri girerek kurumun veritabanındaki uçuş bilgileri görüntülenir. Kredi kartı hesap özeti aslında banka veritabanındaki kayıtların bir raporudur.

İlişkisel Veritabanı Yönetim Sistemleri

Bilgi teknolojileri geliştikçe veritabanı sistemlerinde de birçok yenilik olmuştur. İlk veritabanı sistemlerinde veriler üzerinde değişiklik yapmak oldukça zordu. Veriler arasında bütünlüğü sağlamak amacıyla verilerin eşleştirilmesi ve ilişki kurulması gerektiğinde mevcut sistemler yetersiz kalmakta ve güvenlik sağlanamamaktaydı. Bu gibi durumları ortadan kaldırmak için ilişkisel veritabanlarına ihtiyaç duyulmuştur. Günümüzde en çok kullanılan bu ilişkisel veritabanlarında veriler; tablolar, satırlar ve alanlar şeklinde tutulmaktadır. Bu sayede veriler arasında ilişkilendirmeler yapılabilmekte, verilerin kullanımı ve yönetimi daha verimli bir hâle getirilmektedir. Günümüzde en sık kullanılan ilişkisel veritabanı yönetim sistemleri arasında Microsoft SQL Server, Oracle, Microsoft Access, PostgreSQL ve MySQL gösterilebilir. Bu kitapta ilişkisel veritabanı sistemi olarak Microsoft SQL Server kullanılacaktır. Bununla birlikte bir veritabanı yönetim, sorgulama ve programlama ortamı olan MS SQL Server Management Studio da kullanılacaktır.

Bu bölümde veritabanının temel öğelerini oluşturma, güncelleme ve silme işlemlerine değinilecektir. Veritabanlarıyla çalışırken tüm işlemler *SQL (**S**tructured **Q**uery **L**anguage – Yapısal Sorgulama Dili)* kullanılarak gerçekleştirilir. SQL, veritabanı ile kullanıcı arasındaki iletişimi sağlar. SQL ile yeni bir veritabanı oluşturulabilir, veritabanının adı değiştirilebilir veya veritabanı silinebilir. Aynı zamanda SQL kullanılarak bir tablo oluşturulabilir, tabloya yeni alanlar eklenebilir.

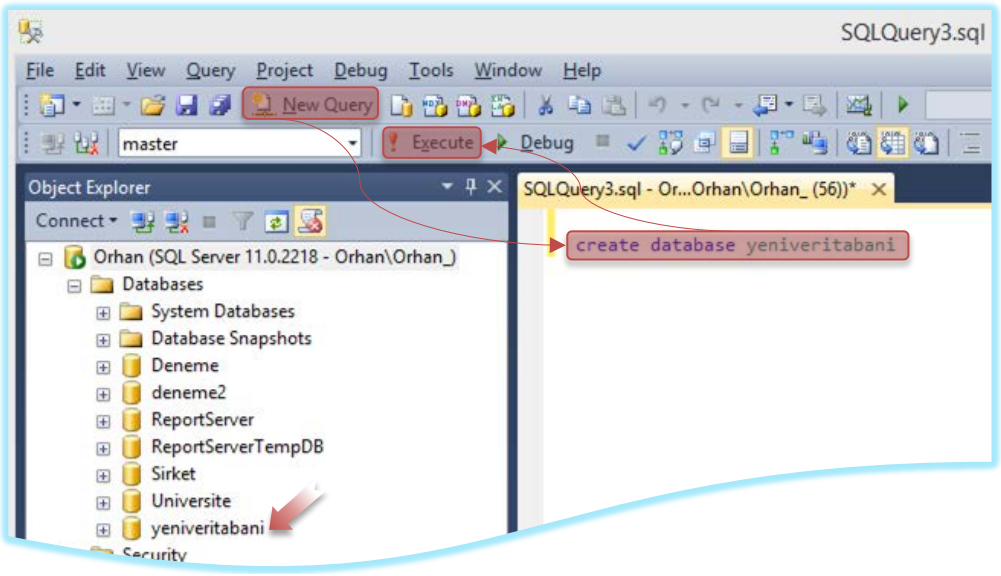
tablo içerisine bilgiler kaydedilebilir. SQL ile hazırlanan sorgular kullanılarak veritabanında depolanan veriler üzerinde tüm işlemler yapılabilir.

SQL dilinin Microsoft SQL Server üzerinde kullanılan sürümüne Transact SQL (T-SQL) adı verilir.

SQL dilinin Microsoft SQL Server üzerinde kullanılan sürümüne *Transact SQL (T-SQL)* adı verilir. T-SQL, SQL üzerine bazı özelliklerin eklenmesiyle oluşturulmuştur. T-SQL sorgularının sonuçları, ilişkisel veritabanı yönetim sistemi tarafından oluşturulur ve kullanıcıya gönderilir. Bu sayede kullanıcı, veritabanıyla uğraşmadan sadece sorgular yazarak veritabanı üzerinde tüm işlemleri yapabilir.

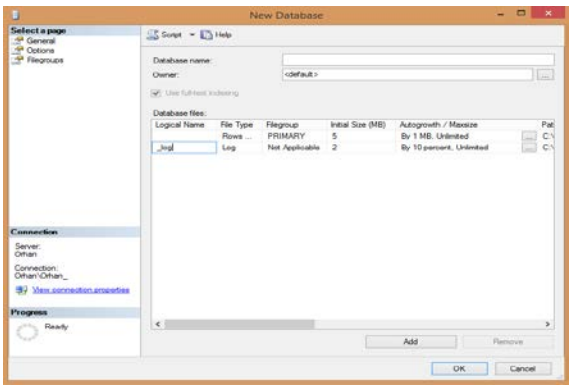
Veritabanı Oluşturmak

MS SQL Server Managemenet Studio kullanılarak iki şekilde yeni bir veritabanı oluşturulabilir. Veritabanı bağlantısı yapıldıktan sonra Resim 1'de gösterilen *New Query* seçeneği kullanılarak bir sorgu sayfası açılır ve burada *"create database veritabani_adi"* ifadesi yazılarak sorgu, *F5* tuşu veya *Execute* komutu ile çalıştırılarak yeni bir veritabanı oluşturulur.



Resim 1. Management Studio - New Query

Management Studio içerisinde bulunan *Object Explorer* bölümünde yer alan *Databases* klasörü üzerine sağ tıklanarak açılan menüden *New Database* seçeneği kullanılarak yeni veritabanı oluşturma penceresi görüntülenir. Bu pencereden veritabanı adı, boyutu gibi özellikler ayarlanarak *OK (Tamam)* butonu kullanılır.



Resim 2. New Database Penceresi

Tablolar

Veritabanlarında tüm veriler tablolar içerisinde saklanır. Tablolardaki her bir satır kayıt (Row) olarak isimlendirilir.

Veritabanlarında tüm veriler tablolar içerisinde saklanır. Tablolar Word ya da Excel tablolarına benzer bir şekilde satır ve sütunların yer aldığı iki boyutlu bir yapıya sahiptir. Veritabanlarında tablolarda her bir sütun *alan (Column)* olarak adlandırılır. Sütunların isimleri ve bu sütunda (alanda) saklanacak verilerin özellikleri tablolarda belirlenir. Bu tablolardaki her bir satır ise *kayıt (Row)* olarak isimlendirilir. Örneğin; ad, soyad, telefon, fax, şehir ve adres bilgilerinin yer aldığı telefon rehberi isimli bir tablo aşağıda gösterilmiştir.

Şekil 2. Alanlar - Kayıtlar

Telefon Rehberi isimli tablo; ad, soyad, telefon, fax, şehir ve adres alanlarından oluşmaktadır. Bu alanların hangi veri tipinde veri saklayabileceği de tablo tasarlanırken belirlenebilmektedir. Örneğin Şekil 2'deki tabloda ad, soyad alanları metinsel ifadeleri, telefon, fax alanları da rakamsal ifadeleri içerebilir.

Tablo oluştururken tablo içerisindeki alanların hangi tipte (sayı, metin, tarih, vb.) veri saklayabileceğini belirleyen temel veri tipleri bulunmaktadır. T-SQL'de bulunan temel veri tiplerinden bazıları Tablo 1'de gösterilmiştir.

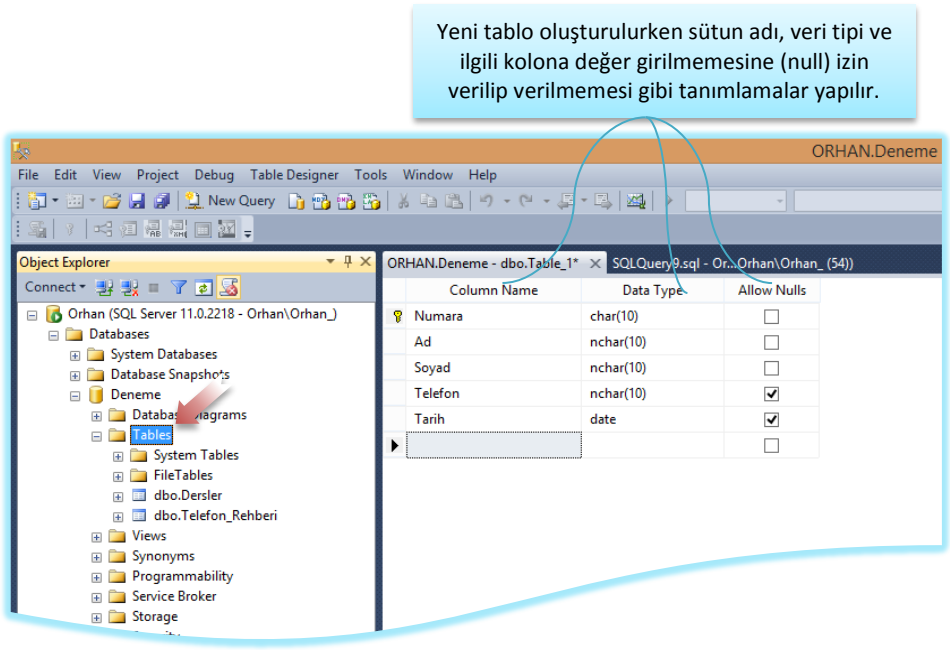
Tablo 1. Veri Tipleri

	Tip	Değer Aralığı
Metinsel Veri Tipleri	char(n)	ASCII türünden ve sabit boyutta veri saklar. En fazla 8000 karakter tutulabilir. (n) alabileceği en fazla karakteri belirler.
	nchar(n)	Unicode türünden ve sabit boyutta veri saklar. En fazla 4000 karakter tutulabilir.
	varchar(n)	ASCII türünden ve değişken uzunlukta veri saklar. En fazla 8000 karakter tutulabilir.
	nvarchar(n)	Unicode türünden ve değişken uzunlukta veri saklar. En fazla 4000 karakter tutulabilir.
	varchar(MAX)	varchar veri tipi ile aynı özelliklere sahiptir ve 2 GB'a kadar veri tutabilmektedir.
	nvarchar(MAX)	nvarchar veri tipi ile aynı özelliklere sahiptir ve 2 GB'a kadar veri tutabilmektedir.
	text	ASCII türünden metin saklamak için kullanılır. 2 GB'a kadar sınırı vardır.
	ntext	Unicode türünden metin saklamak için kullanılır. 2 GB'a kadar sınırı vardır.

Alanlar içine eklenebilecek veri türünü ve boyutunu belirleyen veri tipleri, ilgili alana yanlış türde değer girilmesini engeller.

	Tip	Değer Aralığı
Sayısal Veri Tipleri	int	Yaklaşık -2 milyar ile +2 milyar arasındaki tamsayı değerlerini tutar.
	bigint	Yaklaşık -2^{63} ile $+2^{63}$ arasındaki tamsayı değerleri tutar.
	smallint	Yaklaşık -32 bin ile +32 bin arasındaki tamsayı değerlerini tutar.
	tinyint	0-255 arasındaki tamsayı değerlerini tutar.
	float(n)	Kayan noktalı sayı değerlerini tutar. -1.79e+308 ile 1.79E+308 arasında değer tutabilir. n, 1 ile 53 arasında değer alabilir. 1 ile 24 arasında olduğunda 7 haneye kadar hassasiyet ve 4 byte yer ayrılması söz konusudur. 25 ile 53 aralığı için ise 15 haneye kadar hassasiyet ve 8 byte yer ayrılması söz konusudur. Varsayılan olarak n değeri 53'tür.
	real	-3.40e+38 ile 3.40e+38 arasında değerler alabilir. 7 haneye kadar hassasiyet sunar ve 4 byte yer kaplar. Bu veri tipi float(24)'ün karşılığıdır. Eğer 7 haneye kadar hassasiyet gerekiyorsa real tipi varsayılan float tipi yerine tercih edilebilir.
	money	Yaklaşık -922 milyar ile +922 milyar arasındaki değerleri tutar. Bu tip genelde parasal değerlerin tutulacağı alanlarda kullanılır.
Tarihsel Veri Tipleri	datetime	01.01.1753 ile 31.12.9999 arasındaki tarih ve zaman bilgisini tutar.
	smalldatetime	01.01.1900 ile 06.06.2079 arasındaki tarih ve zaman bilgisini tutar.
	date	01.01.0001 ile 31.12.9999 arasındaki tarih bilgisini tutar.
	time	00:00:00.0000000 ile 23:59:59.9999999 arasındaki zaman bilgisini tutar.
Diğer Veri Tipleri	bit	Boolean değerler tutmak için kullanılan veri tipidir. Sadece 1 veya 0 değerlerini alabilir. 1 True, 0 False değerlerini temsil eder.
	image	Resim dosyalarının veritabanında tutulması için kullanılan veri tipidir. 2 GB'a kadar resim dosyası tutabilmektedir.
	xml	XML dosyalarını ve XML kodlarını saklayabilen veri tipidir. 2 GB'a kadar veri taşıyabilir.
	binary(n)	Sabit uzunluktaki binary veriyi tutmak için kullanılır. Maksimum uzunluğu 8000 byte' tır. Varsayılan uzunluğu ise 1 byte' tır.
	varbinary(n)	Değişken uzunlukta binary veriyi tutmak için kullanılır. Maksimum uzunluğu 8000 byte' tır. Varsayılan uzunluğu ise 1 byte' tır.
	varbinary(MAX)	Maksimum 2 GB binary veriyi tutabilen veri tipidir.

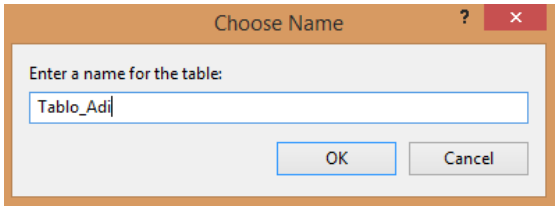
Management Studio ara yüzünden tablo oluşturmak için *Object Explorer* bölümünde bulunan *Databases* içerisindeki *Tables* klasörü sağ tuş menüsünden *New Table* seçeneği kullanılır.



Resim 3. Yeni Tablo Oluşturmak

Bir veritabanında bulunan iki tabloya aynı isim verilemez. Benzer şekilde tablo alan isimleri de birbirinden farklı olmalıdır.

Tablo tasarımında *Allow Nulls* alanındaki kutucuk seçilirse kayıt sırasında ilgili alan herhangi bir değer girilmeden boş geçilebilir. Örneğin Resim 3'teki tabloda numara, ad, soyad alanları boş geçilemezken telefon ve tarih alanlarına bir değer girilmesi zorunluluğu yoktur. Tablo tasarımı bittikten sonra tablo kaydedilir ve tablo ismi belirlenir (Resim 4).



Resim 4. Tablo İsmi Belirleme

Anahtarlar

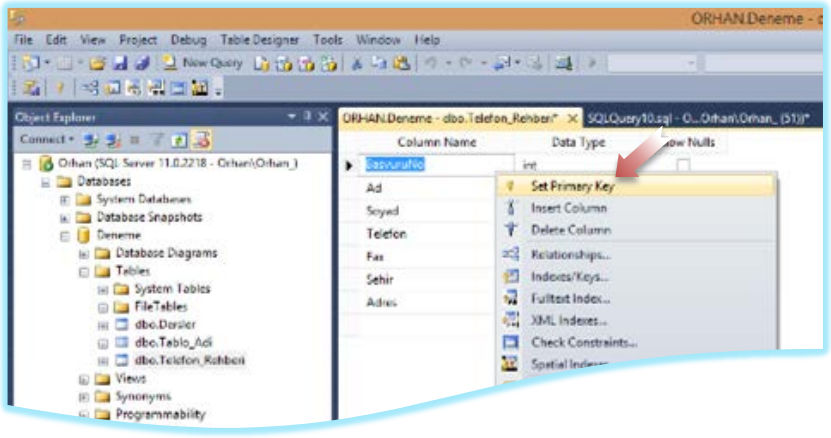
Bir kayıt içerisinde farklılıkları ve nitelikleri gösteren belirleyicilere *anahtar (key)* denir. Tablodaki kayıtları birbirinden ayırt edebilmek için tablo içindeki alanlara belirli anahtarlar atayarak birçok işlem kolaylaştırılabilir. Bir tablo içerisinde bulunabilecek anahtarlar, birincil anahtar (primary key), tekil anahtar (unique key), referans anahtar (foreign key) ve birleşik anahtardır (composite key).

Birincil anahtar olan bir veri aynı tablo içerisinde tekrarlanamaz.

Birincil anahtar (Primary key): Bir tablo içerisindeki kayıtları birbirinden ayırt etmek için kullanılır. Birincil anahtar alanındaki veri boş bırakılamaz, yani NULL değeri alamaz. Tek bir alan birincil anahtar olabileceği gibi bazı tablolarda birden fazla alanın birleşmesiyle birincil anahtar oluşabilir.

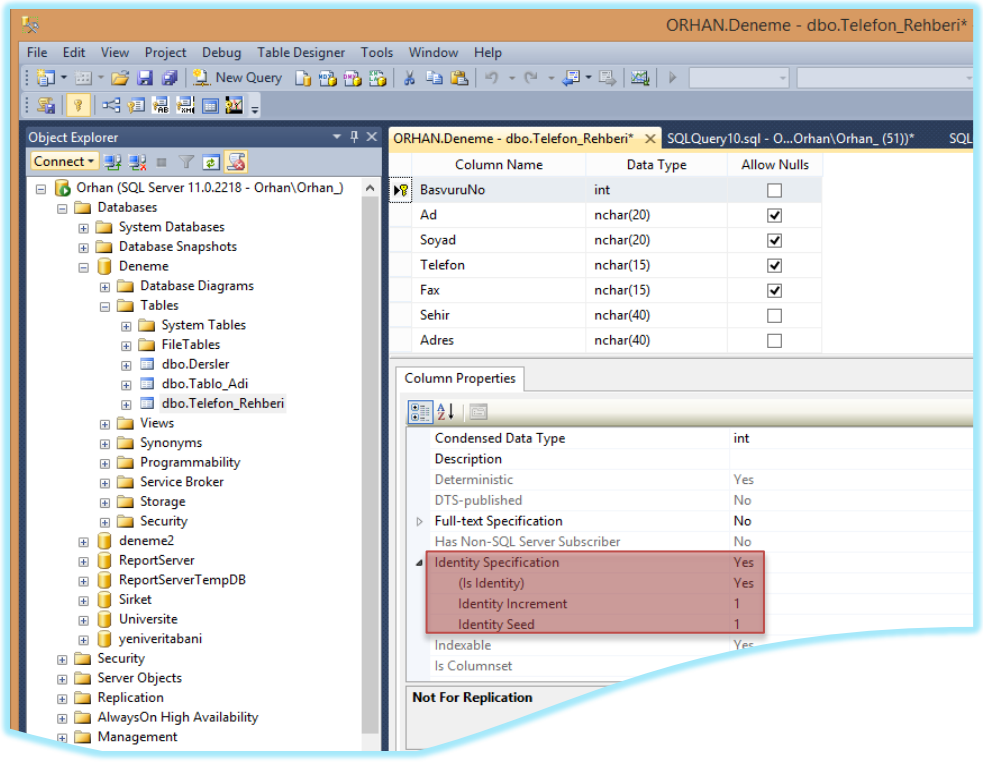
Birincil anahtar, Türkiye Cumhuriyeti'ndeki her bireye verilen TC Kimlik No gibi düşünebilir. Çünkü aynı isim ve soyada ait birden fazla kişi olabilir. Veritabanında da arama, güncelleme, değiştirme, silme işlemleri yaparken birincil anahtar kullanıldığında sorunsuz şekilde işlemler yapılabilir. Örneğin; veritabanında personel bilgilerini tutan bir tablo olsun. "Ali" isimli personel işten çıkarıldığında veritabanında kaydın silinmesi gerekir. Eğer ismi "Ali" olan kaydı sil komutu verilirse aynı isimde başka personel olduğunda o kayıt da silinecektir. Bu da veritabanında sorunlara neden olacaktır. Ama her personele ait bir personel numarası olduğunu düşünürsek, işten çıkarılan "Ali" isimli personelin numarasına göre silme işlemi yapıldığında sadece o kayıt silinecektir.

Birincil anahtar kullanmak veri tekrarını önler, veriye hızlı ulaşmayı sağlar, boşluk girmeyi engeller ve tablolar arasında ilişki kurmayı sağlar. Bir alanı birincil anahtar olarak kullanmak için tablo tasarım görünümünde iken birincil anahtar olarak kullanılmak istenilen alan üzerine sağ tuş yapılır ve açılan menüden *Set Primary Key* seçeneği kullanılır (Resim 5). Bu işlemin ardından birincil anahtar olarak belirtilen alanın sol tarafında anahtar simgesi görüntülenir.



Resim 5. Set Primary Key

Birincil anahtar alanları için genellikle otomatik artan ya da azalan değerli alanlar kullanılır. Tabloya yeni bir kayıt eklendiğinde bu alanın değeri otomatik olarak veritabanı yönetim sistemi tarafından atanır. Management Studio kullanılarak bu işlem çok kolay bir şekilde yapılmaktadır. Tablo tasarlanırken otomatik değer alması istenilen alan seçilir ve *Column Properties* penceresinden *Identity Specification* kısmındaki *IS Identity* özelliği *Yes* yapılır. Ayrıca bu kısımda alanın *Identity Seed* özelliği ile başlangıç değeri, *Identity Increment* ile de artış miktarı belirlenir. Resim 6'da *BasvuruNo* alanı için sayaç 1'den başlayıp 1'er artarak değer üretir. Burada Identity Increment -5, Identity Seed ise 10 olsaydı, alan değeri 10, 5, 0, -5, -10 … şeklinde devam ederdi.



Resim 6. Bir Alana Otomatik Değer Atama

Tablonun tekil anahtar olarak tanımlanmış bir alanına aynı değer sadece bir kez girilebilir.

Tekil anahtar (Unique key): Tablonun tekil anahtar olarak tanımlanmış bir alanına aynı değer sadece bir kez girilebilir. Birincil anahtardan farklı olarak, tabloda bu alana ait sadece bir kayıt NULL değeri alabilir. Birincil anahtar aynı zamanda tek anahtar olarak sayılabilir fakat tek anahtarlar birincil anahtar değildirler.

Referans anahtar (Foreign key): Tablodaki bir veriyi başka tablodaki bir veri ile ilişkilendirmeyi sağlar. İki tablo arasında yapılan bu ilişkilendirme ile referans anahtar olarak tanımlanmış alana sadece ilişkilendirdiği tablonun alanındaki veriler eklenebilir.

Birleşik anahtar (Composite Key): Birden fazla alanın birleştirilmesiyle birincil anahtar görevini üstlenecek tanımlamalar yapılabilir. Bunlar birleşik anahtar olarak adlandırılır.

ADO.NET'E GİRİŞ

ADO.NET, .NET Framework içerisinde bulunan ve çeşitli veri kaynakları ile iletişim kurma amacı ile tasarlanmış bir sınıf kütüphanesidir. ADO.NET, veri işlemlerini kolaylaştıran ve nesneye yönelik programlama (Object Oriented Programming) modeline uygun yapısıyla da oldukça kullanışlı bir platformdur.

.NET ile geliştirilen tüm uygulamalar, veri kaynaklarına erişim için ADO.NET'ten yararlanmaktadır.

ADO.NET, .NET Framework'ün merkezinde yer alır ve birçok platformda kullanılır. Bu nedenle .NET ile geliştirilen tüm uygulamalar, veri kaynaklarına erişim için ADO.NET'ten yararlanmaktadır. ADO.NET ile farklı veri kaynaklarına bağlanma, verileri alıp uygulamalarda kullanma, yeni veriler ekleme veya güncelleme işlemleri oldukça kolay bir hâl almaktadır.

ADO.NET ile çok farklı veritabanı ve veritabanı yönetim sistemleri kullanılabilir. Bu sistemlerin kendilerine ait farklı standartları mevcuttur. Bu durumdan dolayı ADO.NET kütüphanesinde farklı standartları destekleyen isim alanları (Namespace) bulunmaktadır. Bu isim alanlarının tamamı *System.Data* isim alanı içerisinde bulunmaktadır. System.Data, ADO.NET ile uygulama geliştirirken kullanılan ve tüm veri sağlayıcılar için ortak olan bileşenlerin bulunduğu isim alanıdır. System.Data altında yer alan ve farklı sistemlere ait isim alanlarından bazıları aşağıdaki gibidir:

System.Data.SqlClient: SQL Server standartlarını destekleyen ve bu sisteme bağlantı kurmayı sağlayan tipleri içerir.

System.Data.OleDB: Microsoft Access, Oracle, Text dosyaları, Excel dosyaları vb. gibi OleDb arayüzü sağlayan tüm sistemlere bağlantı kurmak için gerekli olan tipleri barındırmaktadır.

System.Data.Odbc: ODBC (Open DataBase Connectivity) standartlarını destekleyen ve bu sistemlere bağlantı kurmayı sağlayan tipler bu isim alanı altında yer alır.

Visual Studio ile Veritabanına Bağlanmak

Visual Studio'da geliştirilen bir uygulama ile bir SQL veritabanı farklı iki platformda bulunduğu için uygulama ile veri kaynağı arasında bir bağlantı oluşturarak bunların iletişiminin sağlanması gerekmektedir. Bu işlem için veritabanına ait isim, konum bilgisi veya güvenlik ayarları gibi parametreleri içeren *bağlantı (Connection)* nesnesine ihtiyaç duyulmaktadır. Sql veritabanıyla bağlantı yapılırken *SqlConnection*, Access veritabanıyla bağlantı yapılırken de *OleDBConnection* sınıfı kullanılır.

SqlConnection

SqlConnection sınıfı, SQL Server veritabanlarına bağlantı kurma, bu bağlantıya ait özellikleri belirleme veya kurulan bağlantıyı kapatma gibi çeşitli parametreleri içerir.

SqlConnection sınıfında yer alan en önemli özelliklerden biri bağlantı ile ilgili kullanıcı rolleri, yetkileri, güvenlik ayarları gibi bilgilerin yer aldığı bağlantı cümlesidir (ConnectionString).

Bağlantı ile ilgili kullanıcı rolleri, yetkileri, güvenlik ayarları gibi bilgilerin yer aldığı *bağlantı cümlesi (ConnectionString),* SqlConnection sınıfında yer alan en önemli özelliklerden biridir. Bağlantı cümlesindeki her bilgi ";" karakteri ile ayrılır ve bunlar *segment* olarak adlandırılır. Bağlantı cümlesinde *Data Source, Initial Catalog* ve *Integrated Security* segmentleri yer almak zorundadır. Bu segmentlerden bazıları şunlardır:

Data Source: Bağlanılmak istenilen SQL Server ismi bu segmentte belirlenir. Aynı makine üzerindeki veritabanına bağlantı yapılması durumunda Data Source kısmına "." karakteri yazılır (Data Source=.). Farklı bir server üzerindeki veritabanına bağlantı yapılacaksa Data Source kısmına veritabanının kaynağı yazılır (Data Source=192.168.1.1\\VeriTabani).

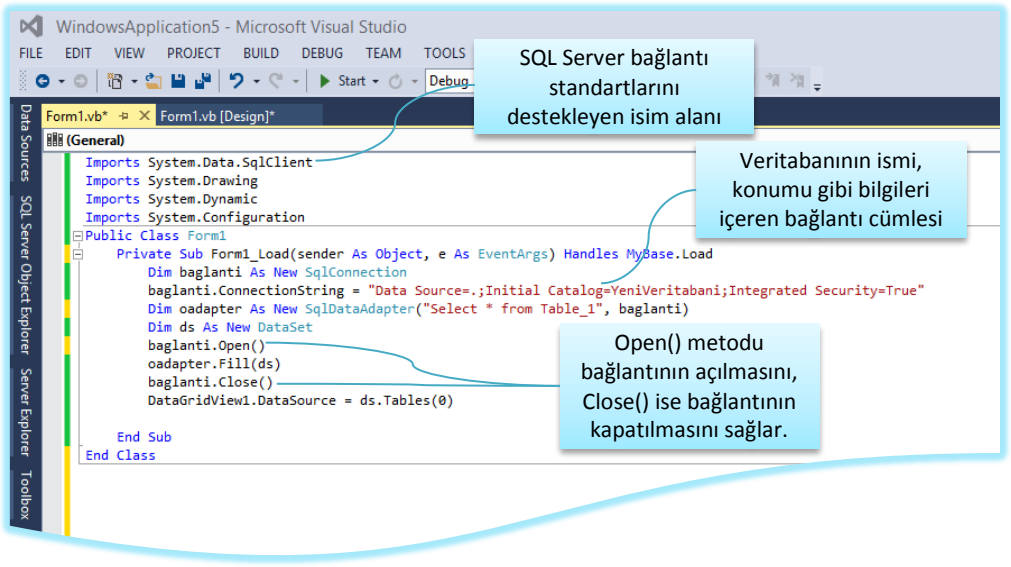
Initial Catalog: Bağlanılmak istenilen veritabanı ismi bu segmentte belirlenir (Initial Catalog=Deneme).

Integrated Security: SQL Server'da iki şekilde oturum açmak mümkündür. Bunlardan biri Windows oturumu açıldıktan sonra tekrar yeni bir şifre girmeyi gerektirmeyen *Windows Authentication* modu, diğeri ise veritabanı kullanıcı adı (User ID) ve şifresinin (Password) girilmesini gerektiren *SQL Server Authentication* modudur. Integrated Security segmenti True ve False şeklinde iki parametre alır. True parametresi Windows Authencation modunda, False parametresi ise SQL Server Authencation modunda oturum açılmasını sağlar. Ancak SQL Server Authencation modunda oturum açılması durumunda bağlantı cümlesinde User ID ve Password segmentleri belirtilmelidir. Aşağıda iki farklı örnek bağlantı cümlesi (ConnectionString) gösterilmiştir:

```
"Data Source=.;Initial Catalog=Deneme;Integrated
Security=True"

"Data Source=.;Initial Catalog=Deneme;Integrated
Security=False; User ID=sa; Password=1234"
```

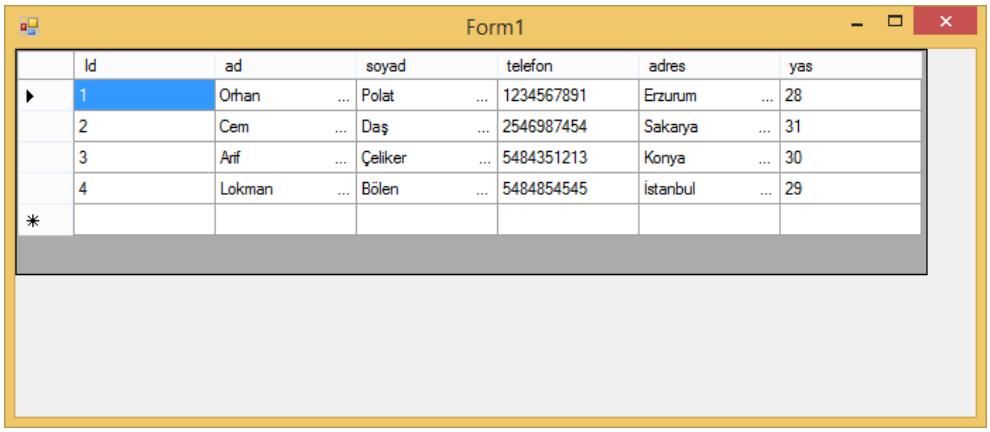
Bağlantı kurulduktan sonra bağlantının hazır olması ve açılmasını sağlamak için *Open()* metodu kullanılır. Bağlantının açık, kapalı gibi durumları da *ConnectionState* metodu ile öğrenilebilir. *Close()* metodu ise bağlantının kapatılmasını sağlar.

Bu bilgiler göz önünde bulundurularak bir uygulama üzerinden örnek bir veritabanı bağlantısı aşağıdaki gibi yapılabilir:



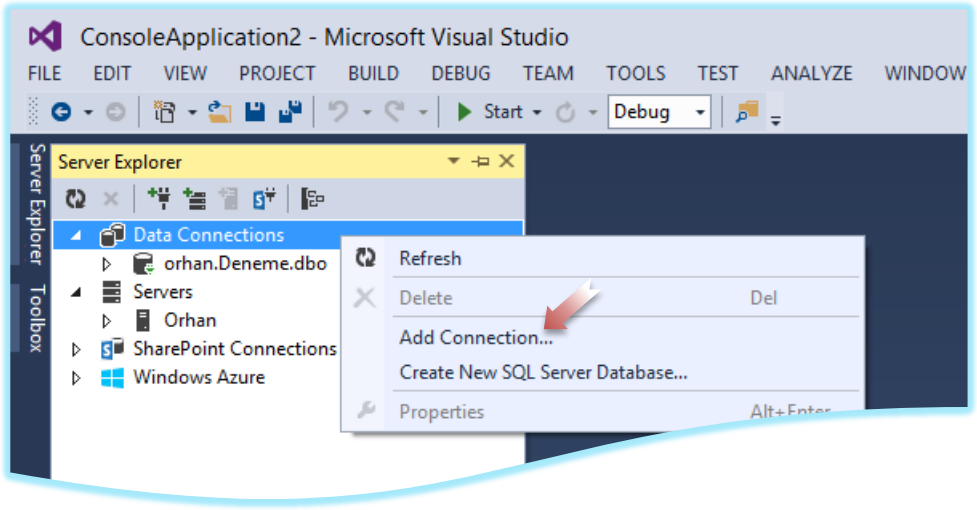
Resim 7. Visual Studio Üzerinden Veritabanına Bağlanma

Bu uygulamada SQL Server'a bağlantı kurularak veritabanı kullanıma hazır hâle getirilmiş ve ardından Close() metodu ile de bağlantı kapatılmıştır. Sistem kaynaklarının gereksiz yere kullanılmaması için veritabanı bağlantısı gerektiğinde açılmalı ve ihtiyaç olmadığı durumlarda kapatılmalıdır. Resim 7'deki uygulama çalıştırıldığında aşağıdaki gibi bir ekran görüntüsü oluşur. Burada forma eklenen DataGridView nesnesi içerisine veritabanındaki Table_1 tablosundan verilerin aktarıldığı görülmektedir.



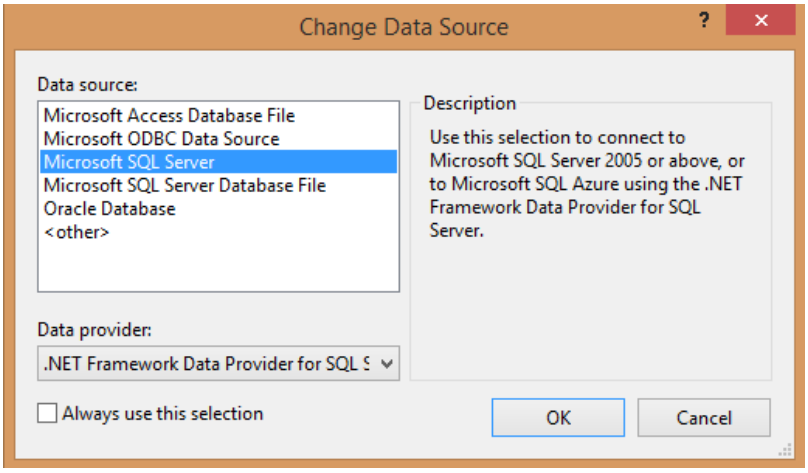
Resim 8. Uygulama Ekran Görüntüsü

Visual Studio üzerinden veritabanı bağlantısı kod yazılarak yapılabildiği gibi *Server Explorer* penceresi kullanılarak da yapılabilir. Bu işlem için Server Explorer penceresinde yer alan *Data Connections* seçeneği sağ tuş menüsünden *Add Connection* komutu kullanılır.



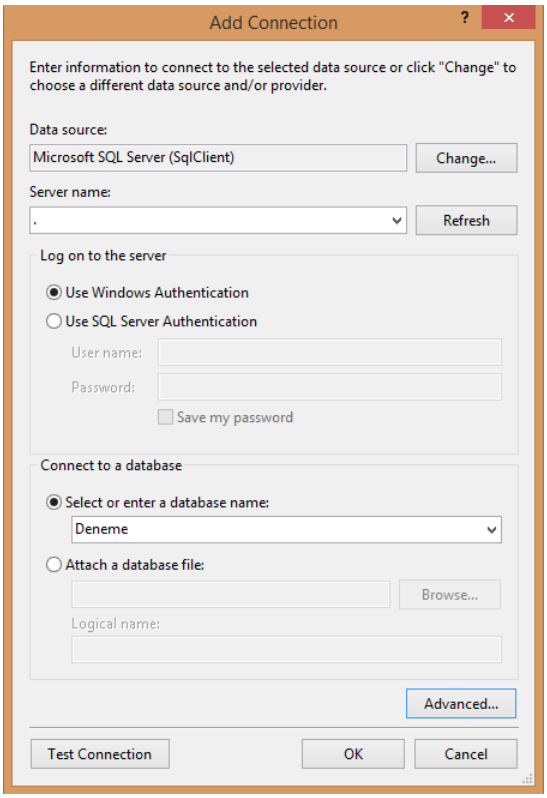
Resim 9. Add Connection

Add Connection komutu kullanıldığında Change Data Source diyalog penceresi görüntülenir (Resim 10). Bu pencereden hangi veritabanı yönetim sistemine bağlanılacaksa ona ait veri kaynağı seçilir. SQL veritabanları için *Microsoft SQL Server*, Access veritabanları için de *Microsoft Access Database File* seçeneği kullanılır. Veritabanı bağlantısı SQL veritabanı ile yapılacağından Microsoft SQL Server seçeneği kullanılmıştır.



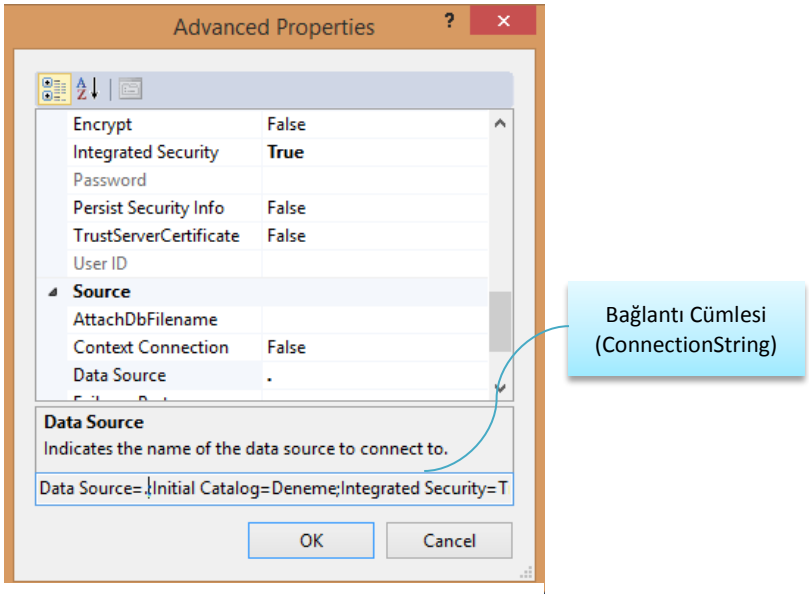
Resim 10. Change Data Source

Veri kaynağı seçildikten sonra *Add Connection* diyalog penceresi görüntülenir (Resim 11). Bu pencereden Server adı, Server'a giriş yöntemi ve bağlantı yapılacak veritabanı belirlenerek veritabanı bağlantısı sağlanır.



Resim 11. Add Connection

Ayrıca Add Connection penceresinde bulunan Advanced düğmesi kullanılarak bağlantı ile ilgili tüm özelliklerin bulunduğu *Advanced Properties* diyalog penceresi görüntülenebilir. Veritabanı bağlantısı ile ilgili bağlantı cümlesi de bu pencerede görüntülenir (Resim 12).



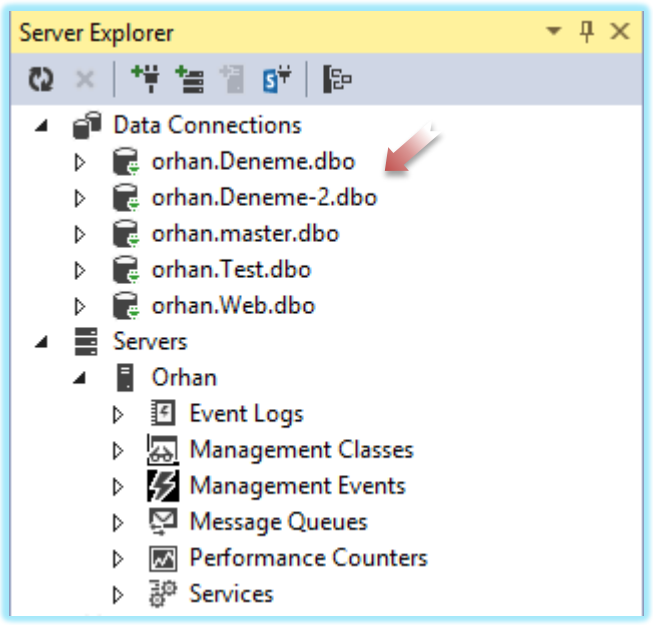
Resim 12. Advanced Properties

Visual Studio.Net ile Veritabanı Tasarlamak

Bu kısımda Visual Studio.NET IDE'si (tümleşik geliştirme ortamı) kullanılarak bir veritabanının nasıl tasarlanacağı gösterilecektir. Visual Studio.NET IDE'si içerisinde barındırdığı araçlarla uygulama geliştiricilerine kolay tasarım ortamları sunmaktadır. Bu araçlardan biri de *Server Explorer* penceresidir. Bu pencere yardımıyla uygulama geliştiriciler sunucu üzerinden yapabildikleri birçok işlemi yapabilirler. Örneğin veritabanı bağlantısı yapmak için Server Explorer penceresi kullanılabilir.

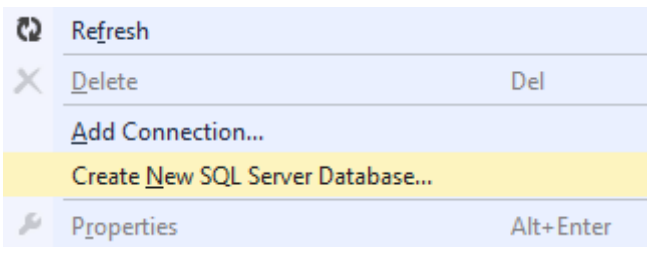
Server Explorer penceresi yardımıyla SQL Server veritabanlarına erişilebilir, yeni veritabanları oluşturulabilir veya mevcut veritabanları üzerinde çeşitli değişiklikler yapılabilir.

Server Explorer penceresi veritabanına erişim işlemi için sıkça kullanılmaktadır. Uygulama geliştiriciler MS SQL Server Management Studio gibi bir yazılımı kullanmadan SQL Server veritabanlarına erişebilir, yeni veritabanları oluşturabilir veya mevcut veritabanları üzerinde çeşitli değişiklikler yapabilir. Resim 13'te Server Explorer penceresi üzerinde bilgisayarda kurulu olan SQL Server içerisindeki veritabanları görüntülenmektedir.



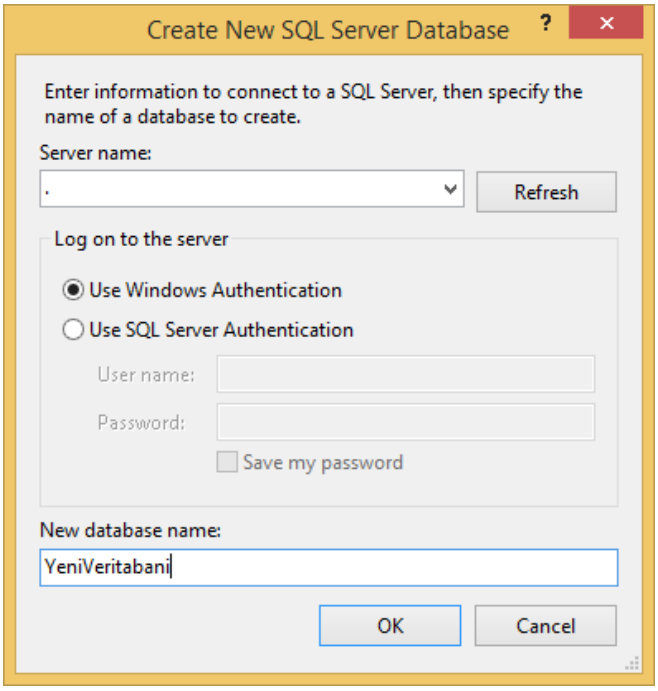
Resim 13. Server Explorer Penceresi - Veritabanları

Yeni bir veritabanı oluşturmak için *Server Explorer* penceresindeki *Data Connections* sağ tuş menüsünde bulunan *Create New SQL Server Database* seçeneği kullanılır (Resim 14).



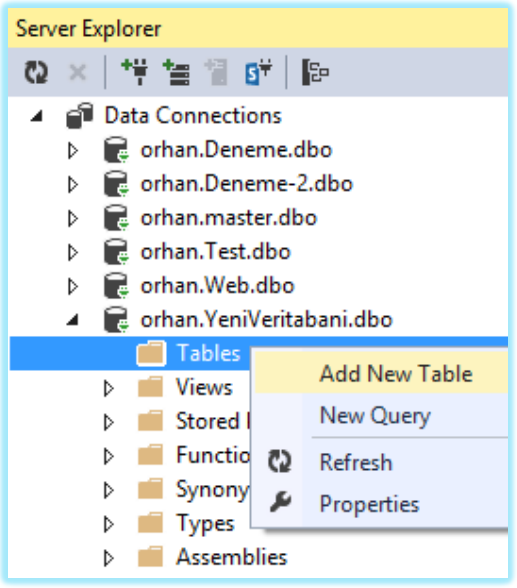
Resim 14. Create New SQL Server Database

Bu seçenek kullanıldıktan sonra açılan pencereden server adı ile veritabanı adı girilir ve güvenlik kipi olarak varsayılan olan *Use Windows Authentication* seçeneği işaretli bırakılıp *Ok* düğmesi tıklanarak yeni bir veritabanı oluşturulur (Resim 15).



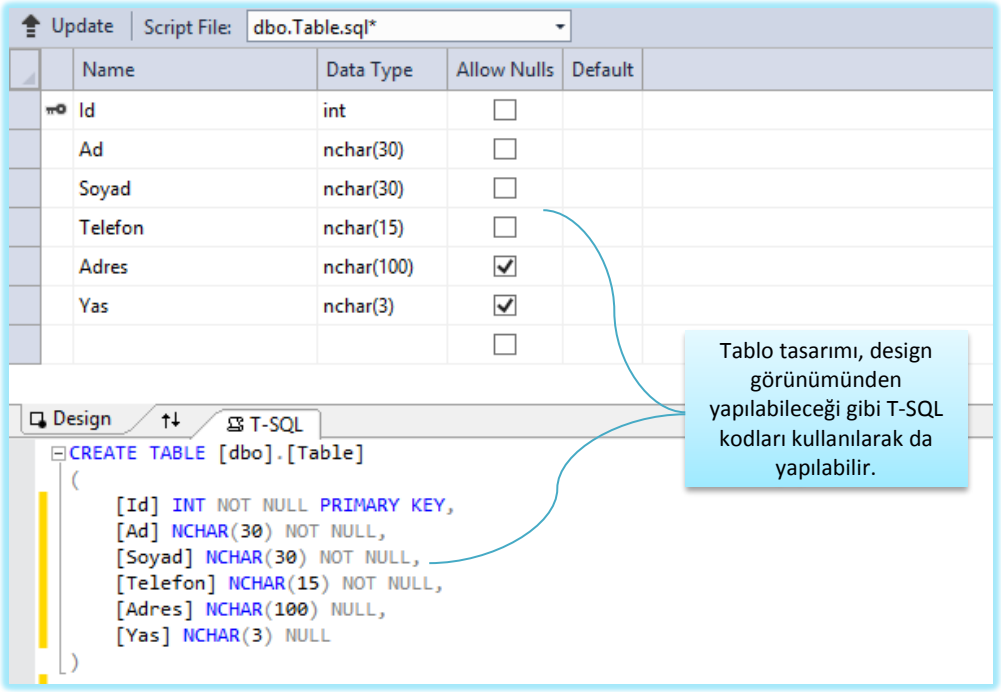
Resim 15. Create New SQL Server Database

Oluşturulan veritabanının anlamlı bir şekilde tasarlanması gerekir. Bu işlem için veritabanına tablolar eklenmeli ve tablolar içerisinde alanlar oluşturulmalıdır. Veritabanına tablo eklemek için *Server Explorer* penceresinde listelenen veritabanlarından ilgili olanı içerisindeki *Tables* bağlantısı sağ tuş menüsünden *Add New Table* seçeneği kullanılır (Resim 16).



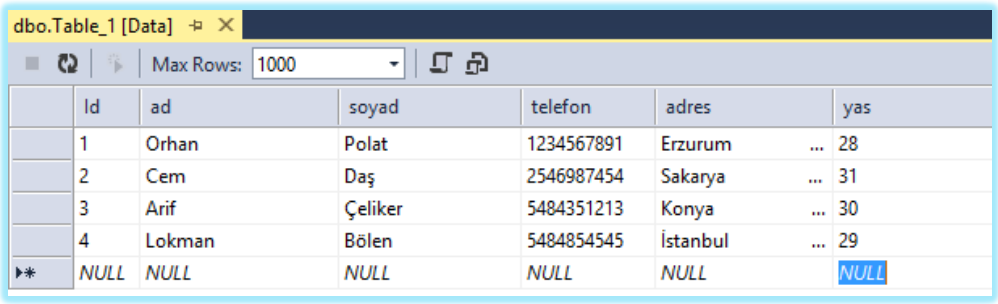
Resim 16. Add New Table

Veritabanına tablo eklendikten sonra tablo tasarımının yapılacağı ekran görüntülenir (Resim 17).



Resim 17. Visual Studio Tablo Tasarım Ekranı

Tasarım ekranında tabloya eklenecek alanlara ait isim, tür, null değer taşıyabilme, varsayılan değer atayabilme gibi özellikler kolaylıkla belirlenebilir. Bu işlemden sonra tabloya kayıt eklemek için Server Explorer penceresinden tabloya ait simge tıklanır ve veri girişi yapılır (Resim 18).



Resim 18. Tablo Veri Girişi Ekranı

Özet

- Veritabanı, bilgisayardaki düzenli bilgileri ifade ederken bu bilgileri bellek üzerinde organize eden, işleyen, isteklere cevap veren uygulamalara Veritabanı Yönetim Sistemi (VTYS) adı verilir.
- Verilerin birbirleri ile ilişkilendirilmesi, tekrara yer vermeden saklanılması ve farklı biçimlerde sunulabilmesi işlemleri oldukça zaman gerektiren ve zor işlemlerdir. Veritabanı yönetim sistemleri bu işlemleri oldukça kolay ve güvenilir bir şekilde gerçekleştirmektedir.
- Bilgi teknolojileri geliştikçe veritabanı sistemlerinde de birçok yenilik olmuştur.
- Veriler arasında bütünlüğü sağlamak amacıyla verilerin eşleştirilmesi ve ilişki kurulması gerektiğinde mevcut sistemler yetersiz kalmakta ve güvenlik sağlanamamaktaydı.
- Bu gibi durumları ortadan kaldırmak için ilişkisel veritabanlarına ihtiyaç duyulmuştur.
- Günümüzde en sık kullanılan ilişkisel veritabanı yönetim sistemleri arasında Microsoft SQL Server, Oracle, Microsoft Access, PostgreSQL ve MySQL gösterilebilir.
- Veritabanlarında tüm veriler tablolar içerisinde saklanır.
- Tablolar Word ya da Excel tablolarına benzer bir şekilde satır ve sütunların yer aldığı iki boyutlu bir yapıya sahiptir.
- Tablo oluştururken tablo içerisindeki alanların hangi tipte (sayı, metin, tarih, vb.) veri saklayabileceğini belirleyen temel veri tipleri bulunmaktadır.
- Tablodaki kayıtları birbirinden ayırt edebilmek için tablo içindeki alanlara belirli anahtarlar atayarak birçok işlem kolaylaştırılabilir.
- ADO.NET, .NET Framework içerisinde bulunan ve çeşitli veri kaynakları ile iletişim kurma amacı ile tasarlanmış bir sınıf kütüphanesidir.
- ADO.NET, .NET Framework'ün merkezinde yer alır ve birçok platformda kullanılır.
- ADO.NET ile çok farklı veritabanı ve veritabanı yönetim sistemleri kullanılabilir.
- ADO.NET kullanılarak Visual Studio ve SQL Server veritabanı arasında bağlantı oluşturulabilir.
- Bağlantı oluşturulurken SqlConnection sınıfından yaralanılır.
- Visual Studio.NET IDE'si içerisinde barındırdığı araçlarla uygulama geliştiricilerine kolay tasarım ortamları sunmaktadır.
- Bu araçlardan biri de Server Explorer penceresidir.
- Uygulama geliştiriciler MS SQL Server Management Studio gibi bir yazılımı kullanmadan SQL Server veritabanlarına erişebilir, yeni veritabanları oluşturabilir veya mevcut veritabanları üzerinde çeşitli değişiklikler yapabilir.

DEĞERLENDİRME SORULARI

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

1. **MS-SQL Server'da kullanılan SQL dili sürümü aşağıdakilerden hangisidir?**
   a) T-SQL
   b) PL/SQL
   c) SQL/PSM
   d) Registered Servers
   e) PL/PSM

2. **MS-SQL Server'da veritabanları hangi bölüm altında bulunur?**
   a) Solution Explorer
   b) Templare Explorer
   c) Object Explorer
   d) Utility Explorer
   e) Toolbox

3. **İlişkisel bir veritabanında veriler aşağıdaki yapılardan hangisinde saklanır?**
   a) Anahtar
   b) Tablo
   c) İndeks
   d) Pencere
   e) Prosedür

4. …….. .NET Framework içerisinde bulunan ve çeşitli veri kaynakları ile iletişim kurma amacı ile tasarlanmış bir sınıf kütüphanesidir.

   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?
   a) SqlConnection
   b) MS-SQL Server
   c) SqlCommand
   d) T-Sql
   e) ADO.Net

5. **ADO.Net'te MS-SQL Server'a bağlanmak için aşağıdaki sınıflardan hangisi kullanılır ?**
   a) SqlCommand
   b) ConnectionString
   c) SqlDataReader
   d) SqlConnection
   e) DataSource

6. **Kullanıcı rolleri, yetkileri, güvenlik ayarları gibi bilgilerin yer aldığı SqlConnection sınıfı özelliği aşağıdakilerden hangisidir?**
   a) ConnectionString
   b) ConnectionTimeout
   c) State
   d) Container
   e) Site

7. **ConnectionString'de bağlanılmak istenen veritabanı hangi segmente yazılır?**
   a) Data Source
   b) Integrated Security
   c) Initial Catalog
   d) Trusted Connection
   e) User ID

8. **Visual Studio'dan veritabanına bağlanmak için kullanılan DataConnections hangi pencerenin altında bulunur?**
   a) Toolbox
   b) Object Explorer
   c) Solution Explorer
   d) Server Explorer
   e) Properties

9. **ConnectionString'de veritabanı kullanıcı adı ve şifresi hangi segmentlere yazılır?**
   a) Integrated Security/Password
   b) Data Source/Password
   c) Password/Data Source
   d) Initial Cataolog/User ID
   e) User ID/Password

10. **ConnectionString'deki segmentler aşağıdakilerden hangisi ile birbirinden ayrılır?**
    a) :
    b) ;
    c) !
    d) ,
    e) .

Cevap Anahtarı

1.A, 2.C, 3.B, 4.E, 5.D, 6.A, 7.C, 8.D, 9.E, 10.B

YARARLANILAN VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

ALGAN, Sefer, (2009), *Her Yönüyle C#*, Pusula Yayıncılık, İstanbul.
AKTAŞ, Volkan, (2013), *Her Yönüyle C# 5.0,* KODLAB, İstanbul.

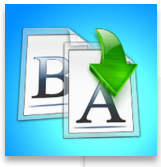
AKKAYA, M. E. (2013). *MS SQL SERVER KOMUTLARI VE MS SQL KOMUTLARI İLE İLGİLİ UYGULAMALAR: Microsoft SQL Server Veritabanına Giriş (Birinci Seviye-İlk Adım)*. Mehmet Emin AKKAYA.

NIELSEN, P., & PARUI, U. (2011). *Microsoft SQL server 2008 bible* (Vol. 607). John Wiley & Sons.

THOMAS, S. P. (2013). *Installing SQL Server 2012 Step By Step: Installing SQL Server 2012 Step By Step*. TekkieBooks.

http://msdn.microsoft.com/en-us/library/System.Data.SqlClient.SqlConnection(v=vs.110).aspx

ADO.NET ve VERİTABANI İŞLEMLERİ



İÇİNDEKİLER

- ADO.NET Mimarisi ve Özellikleri
- ADO.NET Veritabanı Erişim Yöntemleri
- ADO.NET Sınıfları
- ADO.NET ile Veritabanı İşlemleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - ADO.NET teknolojisi ve bu teknolojiye ait temel yapıları öğrenebilecek,
  - ADO.NET ile veri kaynağına erişebilecek,
  - ADO.NET ile temel veritabanı işlemlerini yapabileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Arş. Gör. Mehmet Cem BÖLEN

ÜNİTE

9

GİRİŞ

Bu ünitede Visual Studio.NET ortamında veritabanları üzerinde çeşitli işlemler (Ekleme,Silme, Güncelleme vb.) gerçekleştirmek amacıyla kullanılan ADO.NET teknolojisi, giriş seviyesinde incelenecektir. ADO.NET'in en çok kullanılan nesneleri tanıtılacak ve veritabanı işlemlerinde kullanılan .NET sınıf kütüphanesindeki Systam.Data isim uzayına değinilecektir. Son olarak ADO.NET nesneleri kullanılarak SQL Server veritabanında okuma,ekleme, güncelleme ve silme işlemleri yapılacaktır. Bu ünitede amaçlanan ADO.NET teknolojisini anlamak ve bu teknoloji ile Visual Studio.NET'ten temel veritabanı işlemlerinin nasıl yapıldığını göstermektir. Ünite içerisinde verilen örneklerde SQL Server veri sağlayıcısı ve Visual Basic.NET dili kullanılmıştır.

ADO.NET MİMARİSİ

ADO.NET ile nesne yönelimli veritabanlarına, ilişkisel veritabanlarına, Excel, CSV, metin ve XML dosyalarına erişim mümkündür.

Yoğun veri akışının olduğu günümüzde birçok program verilerini saklamak ve gerektiğinde bu verilere ulaşmak için artık veritabanları ile çalışılmaktadır. Hatta bazı büyük yazılım projelerinde birden fazla farklı veri tabanı kullanım ihtiyacı doğabilmektedir.

Özellikle getirmiş olduğu kolay ve esnek yöntemler ile veritabanı ile çalışmayı kolaylaştıran ADO.NET teknolojisi, Visual Studio.NET'in ilk sürümünden itibaren programlama dünyasında büyük yankı uyandırmış ve programcılar tarafından kullanımı kısa sürede yaygınlaşmıştır.

ADO.NET, veri kaynağındaki (Veritabanı, metin, XML Dosyaları vb.) verilere erişmek veya bu verileri değiştirmek için geliştirilmiş bir veri erişim teknolojisidir. ADO.NET teknolojisi ile Access, MSSQL ve Oracle gibi farklı veritabanlarına erişmek ve bu veritabanları üzerinde okuma, yazma, güncelleme ve silme gibi işlemler kolaylıkla gerçekleştirilebilmektedir. Ayrıca veri kaynağına sunmuş olduğu farklı erişim yöntemleri ile performans, güvenlik ve güncel veri gibi kriterlere göre programcının ihtiyaca göre farklı alternatifler kullanmasına olanak tanır. Örneğin kaynakların sınırlı olduğu mobil uygulamalarda masaüstü uygulamalarına göre veri kaynağına erişim için farklı yöntemler kullanır.

ADO.NET teknolojisi, veri işlemek için iki yöntem(model) sunar. Bu yöntemler bağlantılı (connected) erişim ve bağlantısız (disconnected) erişim olarak adlandırılmaktadır. Bağlantılı erişimde uygulama, veritabanına veri işlemleri gerçekleştiği sürece bağlı kalır. Bağlantısız erişimde ise uygulama, veri kaynağı sadece veri alış verişi yapılırken açılır ve işlem bittikten sonra kapatılır. Bağlantısız veri erişiminde veri kaynağından alınan veriler bellekte belirli bir süre ve belirli işlemlere tabi olarak tutulabilir. Böylece sürekli veritabanına bağlanmaya gerek kalmaz.

Özetle ADO.NET, veri depolamasına ve erişimine yönelik artan ihtiyaçlara çevrim içi ve çevrim dışı çalışabilen mekanizmalar ile cevap vermektedir. Çeşitli veritabanlarına erişmek ve işlem yapabilmek için Managed Provider isimli sınıf kütüphaneleri kullanılır. Bu kütüphaneler ADO.NET'in bağlantılı ve bağlantısız erişim yöntemlerin de de kullanılırlar.

ADO.NET'te kullanıma hazır gelen iki managed provider vardır. Bu providerların isimleri ve destekledikleri veritabanları Tablo 1'de belirtilmiştir:

Tablo 1. Managed Providerlar ve Destekledikleri Veritabanları

Managed Provider		Genel Bilgi
OleDB.NET Data Provider	→	OLE DB protokolünü destekleyen tüm veritabanlarına erişim için kullanılır. SQL Server, Access, Oracle bu protokolü destekleyen veritabanlarındandır.
SQL Server.NET Data Provider	→	Microsoft SQL Server 7.0 ve üzeri sürümlerine erişim için kullanılır.

ADO.NET SINIFLARI

ADO.NET, veri kaynağına bağlanmanın yanında bu veri kaynağında birtakım işlemlerin yapılmasına da imkân vermektedir. Örneğin veri ekleme, güncelleme, silme, Batch Update (Toplu güncelleme) asenkron veri tabanı işlemleri bu işlemlerden birkaçıdır. Bütün bu işlemler managed providerların içerdikleri sınıflar vasıtasıyla gerçekleştirilir. Tablo 2'de bir veri kaynağına erişmek için kullanılan OLEDB.NET Data Provider'a ait bazı sınıflar verilmiştir.

Veri kaynağına bağlantıyı sağlamak için veri kaynağına göre OleDbConnection veya SqlConnection sınıfları kullanılır.

Tablo 2. OleDb.NET Sınıfları ve Sql Server.Net Sınıfları

OleDb.Net Sınıfı		Genel Bilgi
OleDbConnection	→	Veri kaynağına bağlanmak ve bu bağlantıyı yönetmek için gerekli fonksiyonlara ve özelliklere sahiptir.
OleDbCommand	→	İşlem yapılmak istenen veri kaynağı üzerinde çeşitli sorgu ve komutların çalıştırılmasını sağlayan fonksiyonlara ve özelliklere sahiptir.
OleDbDataAdapter	→	Bağlantılı ve bağlantısız özelliği ile verinin iletilmesinde bir köprü görevi görür.
OleDbDataReader	→	Veri kaynağından sırasıyla ve salt okunabilir(read-only) şekilde veri okumayı sağlar.

OleDb.NET Data Provider'in sahip olduğu bu sınıfların Sql Server .Net Data Provider'daki karşılıkları Tablo 3'te belirtilmiştir.

Tablo 3. OleDb.NET Sınıfları

OleDb.Net Sınıfı		Sql Server.Net Sınıfı
OleDbConnection	→	SqlConnection
OleDbCommand	→	SqlCommand
OleDbDataAdapter	→	SqDataAdapter
OleDbDataReader	→	SqlDataReader

Bir veri kaynağına erişmek için öncelikle veri kaynağı ile bağlantının sağlanması gerekmektedir. Veri kaynağına bağlantıyı sağlamak için veri kaynağına göre OleDbConnection veya SqlConnection sınıfları kullanılır. OleDbConnection hem Sql Server'a hem de diğer veri tabanı sistemlerine bağlanırken kullanılabileceği için, bu ünitede vereceğimiz örneklerde OleDb provider'ını kullanacağız.

OleDbConnection Sınıfı

ADO.NET kütüphanesinde yer alan Connection sınıfları, veritabanına bağlantı kurma ve bu bağlantıya ait bir takım kurallar veya özellikler belirlemek gibi işlemlerin yapılmasını yarayan fonksiyon ve sınıfları içermektedir. Genel olarak OleDbConnection ve SqlConnection sınıflarının içermiş olduğu fonksiyonlar ve özellikler aynı olup sadece isimleri farklıdır.

OleDbConnection sınıfı System.Data.OleDb isim uzayı altında yer alır. Bu yüzden OleDb provider'ına ait sınıflar kullanılırken System.Data.OleDb isim uzayı çağrılmalıdır. Aşağıda verilen örnekte Bir Windows Form'da yer alan Button'a tıklandığında bir OleDbConnection nesnesi oluşturularak SqlServer'da bulunan bir veritabanına bağlantı sağlayan kod örneği görülmektedir:

```
Imports System.Data.OleDb
Public Class Form1
  Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
    Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
    Try
      baglanti.Open()
      MessageBox.Show("Bağlantı Başarılı")
      baglanti.Close()
    Catch ex As Exception
      MessageBox.Show("Bağlantı Başarısız")
    End Try
  End Sub
    End Class
```

OleDb provider'ına ait sınıflar kullanılırken System.Data.OleDb isim uzayı çağrılmalıdır.

Kodlar incelendiğinde ilk olarak "baglanti" isimli bir OleDbConnection nesnesi türetildiği görülmektedir. Sonrasında "baglanti" isimli nesnenin *ConnectionString özelliğine, bağlanılacak veritabanı ile ilgili bilgiler tanımlanır.* Verilen ConnectionString lokal bir bilgisayarda kurulu olan Sql Server 2012 sürümündeki bir veritabanına bağlanmak için yazılmıştır.

baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"

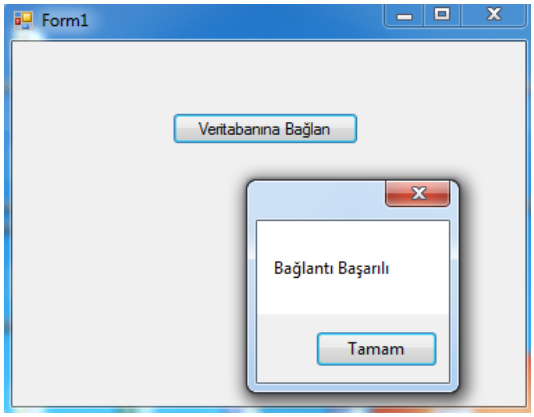
ConnectionString bağlantı ile ilgili bilgileri içerir ve *(;)* karakteri kullanılarak verilen bilgiler bir anlamda sınıflandırılır. Örneğimizdeki ConnectionString'de Provider erişimde kullanılan provider'ın adını, *Data Source* veri tabanı sunucusunun ismini veya adresini, *Initial Catalog* bağlanılan veritabanının ismini, *Integrated Security* ise isim ve şifreye gerek kalmadan bağlanabilmek için kullanılan ConnectionString nitelikleridir. Bütün bu nitelikler (;) karakteri ile birbirinden ayrılmışlardır. ConnectionString belirlendikten sonra veritabanına bağlantı açılabilir. Bunun için OleDbConnection sınıfının *Open()* metotu kullanılır. Bu metot, ConnectionString'de belirlenen veritabanına bağlantı oluşturur. Eğer daha önceden bu nesne için veritabanına bir bağlantı oluşturulmuşsa Open() metodunun kullanılması Connection Pooling özelliği aktif değil ise hataya yol açabilir. Bu yüzden Open metodu kullanılmadan önce ConnectionState özelliğine bakılarak bağlantı durumu kontrol edilmeli ve buna göre bağlantı yapılmalıdır. ConnectionState özelliğinin değeri aşağıda verilen kod ile kontrol edilir:

```
Dim baglanti As New OleDbConnection
 baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
 Try
   If baglanti.State = ConnectionState.Closed Then
      baglanti.Open()
      MessageBox.Show("Bağlantı Başarılı")
      baglanti.Close()
   End If
 Catch ex As Exception
   MessageBox.Show("Bağlantı Başarısız")
End Try
```

CLOSE() metodu açık olan bağlantıyı kapatmaya yarar ve böylece bağlantı için ayrılan kaynaklar iade edilir.

Örnekte dikkat edilmesi gereken son önemli nokta "baglati" nesnesine ait olan *Close()* metotudur. Bu metot açık olan bağlantıyı kapatmaya yarar ve böylece bağlantı için ayrılan kaynaklar iade edilir.

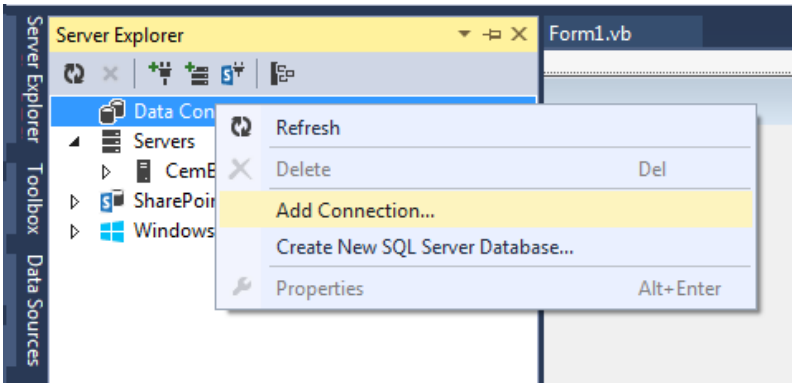
Yukarıdaki kodlar çalıştırıldığında eğer veritabanına erişim sağlanırsa "Bağlantı Başarılı" mesajı kullanıcıya gösterilecektir.(Resim 1)



Resim 1. Bağlantı Kurulduğunda Oluşan Ekran Görüntüsü

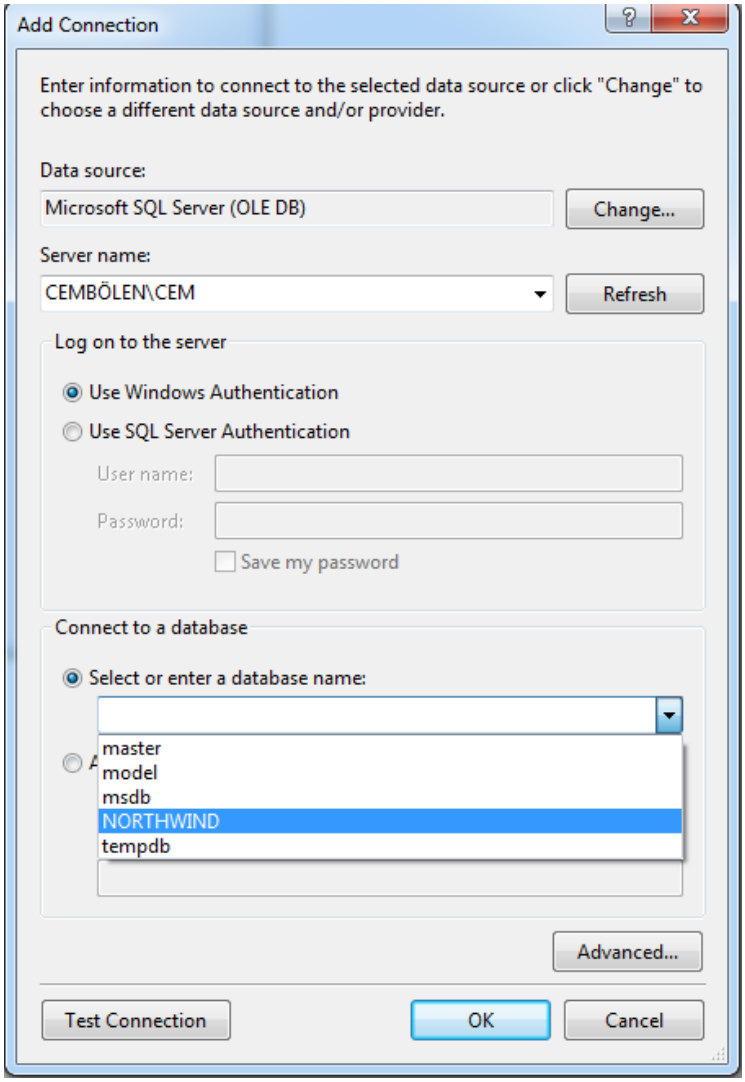
Özellikle ADO.NET'i öğrenmeye yeni başlayanların ConnectionString bağlantısını elle yazarken bir noktalama işaretinin unutulması ya da bir harfin yanlış yazılması muhtemeldir. Bu yüzden Visual Studio.Net arayüzünden bir sihirbaz (wizard) yardımıyla Connection/ConnectionString'i oluşturmak hataların giderilmesinde ve ConnectionString'in yazım (syntax)'ına alışmak için yararlı olabilir. Bunun için yapılaması gerekenler sırayla şu şekildedir:

Server Explorer menüsünden Data Connections seçeneğinin üzerine sağ tıklayarak Add Connection komutunu verelim.



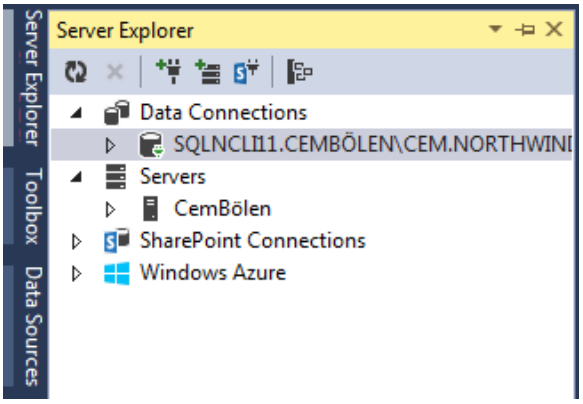
Resim 2. Server Explorer Penceresi

1) Karşımıza gelen pencerede bağlanılacak veritabanı ve kullanılacak provider seçilir:



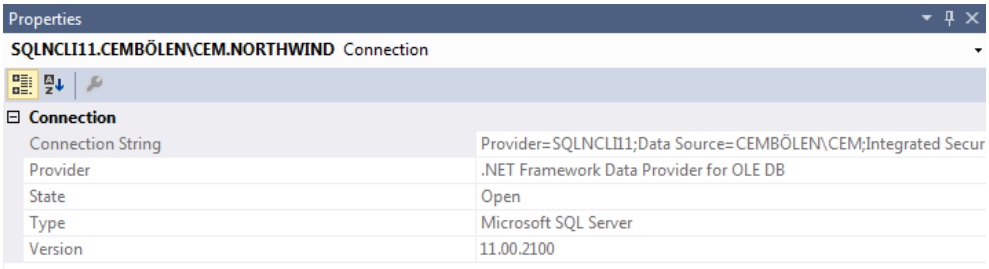
Resim 3. Bağlantı Ekleme Penceresi

2) Bu işlemleri gerçekleştiridikten sonra Test Connection butonuna basarak bağlantıyı test edebiliriz. Bağlatı başarılı şekilde kurulursa OK butonuna basarak işlemimizi tamamlarız. Bundan sonra Server Explorer'da yer alan Data Connections sekmesi altına oluşturduğumuz bağlantının eklendiğini görürüz (Resim 4).



Resim 4. Data Connections Sekmesi Altına Eklenen Bağlantı

3) Bu bağlantıda veritabanına başarılı şekilde eriştiği için ConnectionString'ini alıp kullanabiliriz. Oluşan ConnectionStringi almak için bağlatının özelliklerine giderek (Resim 5).

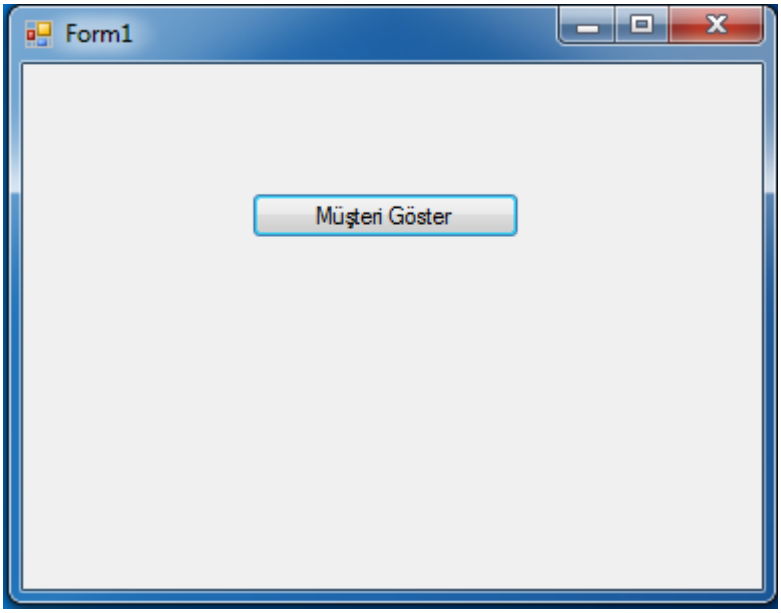


Resim 5. Eklenen Bağlantı Özellikleri

OleDbCommand Sınıfı

Veri kaynağına erişim sağladıktan sonra OleDbCommand sınıfı kullanılarak veri kaynağı üzerinde çeşitli komutları çalıştırılır. Bu komutlar genelde SQL adı verilen veritabanı sorgulama ve manipulasyon dilinden oluşmakta olup OleDbCommand sınıfında yer alan CommandText özelliği kullanılarak veri tabanına gönderilir.

OleDbCommand sınıfının kullanım şekli OleDbConnection'a benzerdir. OleDbCommand sınıfını kullanmak için *System.Data.OleDb* isim uzayını çağırmak gereklidir. OleDbCommand sınıfını daha iyi anlamak için yapacağımız örnekte bir Windows Forms'da yer alan buttona tıklandığında veritabanına erişim sağlanacak ve veritabanındaki müşteri tablosundaki ilk müşteri MessageBox ile ekrana yazdırılacaktır. Kullanacağımız örnek veritabanı NORTHWIND isminde olup Microsoft'un sitesinden indirilebilir. Resim 6'deki gibi basit bir ekran tasarımı yaptıktan sonra artık kod yazmaya başlayabiliriz.



Resim 6. Örnek Uygulama Ekran Tasarımı

Aşağıdaki kodlar incelendiğinde OleDbCommand'dan türetilen bir Command nesnesi kullanılarak NORTHWIND isimli veritabanında yer alan Customers tablosundaki bütün kayıtları getiren sorgunun nasıl yazıldığı görülmektedir:

```
Dim baglanti As New OleDbConnection
baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
Dim komut As New OleDbCommand
komut.Connection = baglanti
    komut.CommandText = "Select TOP 1 * from CUSTOMERS"
```

CommandText özelliğinde verilen sorgu, *ExecuteReader()* fonksiyonu kullanılarak çalıştırılır.

Burada öncelikle bir baglanti nesnesinin oluşturulduğu, sonrasında ise OleDbCommand sınıfından komut isimli bir nesnenin türetildiği görülmektedir. Komut nesnesinin hangi bağlantıyı kullacağı Connection özelliğine atanarak belirlenmiştir. Örnekte OleDbConnection sınıfından türetilen "baglanti" nesnesi Connection özelliğine atanmıştır. Kodun son satırına geldiğimizde CommandText özelliğine veritabanına gönderilecek sorgunun atandığını görmekteyiz. CommandText özelliğinde verilen sorgu ise *ExecuteReader()* fonksiyonu kullanılarak çalıştırılır. Ancak ExecuteReader() fonksiyonundan dönen değer OleDbDataReader sınıfı türünden bir referans değeri döndürdüğü için bu fonksiyonun kullanımına OleDbDataReader başlığı altında devam edilecektir.

OleDbCommand sınıfının diğer temel fonksiyonları Tablo 4'te belirtilmiştir.

Tablo 4. OleDbCommand Sınıfının Başlıca Metotları

OleDbCommand Sınıfı		Genel Bilgi
ExecuteNonQuery	→	CommandText özelliğinde belirtilen, herhangi bir kayıt döndürmeyen (Insert,Update,Delete gibi) sorguların çalıştırılmasını sağlar.
ExecuteScalar	→	CommandText özelliğinde belirtilen sorguyu çalıştırır ve geriye ilk kaydın ilk kolonunu döndürür.
ExecuteReader	→	CommandText özelliğinde belirtilen sorguyu çalıştırır ve geriye OleDbDataReader sınıfı türünden bir nesne referansı döndürür. Özellikle SELECT komutu içerdiği durumlarda sıklıkla kullanılır.

OleDbDataReader Sınıfı

OleDbCommand sınıfını kullanarak veri kaynağına çeşitli işlemleri gerçekleştirmek üzere komutlar göndeririz. OleDbDataReader sınıfı ise OleDbCommand sınıfı ile gönderilen komutların sonuçlarını okumak için kullanılır. *Bir OleDbDataReader nesnesinin Read() metotu kullanılarak veri kaynağından gelen kayıtlar sırayla okunur ve tüm kayıtlar okunduktan sonra veri kaynağı ile bağlantı kesilir.*

OleDbDataReader sınıfından bir nesne türetilirken "NEW" anahtar kelimesi kullanılmaz.

OleDbDataReader sınıfından bir nesne türetilirken "new" anahtar kelimesi kullanılmaz. Bunun yerine OleDbCommand nesnesinden ExecuteReader() fonksiyonundan dönen OleDbDataReader sınıfı nesnesinin referansı kullanılır. Aşağıda verilen örnekte OleDbDataReader sınıfına ait bir nesnenin nasıl türetildiği görülmektedir:

```
Dim ord As OleDbDataReader
ord = komut.ExecuteReader
```

DataReader nesnesi bu şekilde türetildikten sonra Read() metotu ile okuma yapılarak veritabanından dönen kayıt kümesi içerisinde dolaşılabilir. Artık DataReader kullanımını öğrendiğimize göre örneğimizi tamamlayabiliriz. Buna göre örneğimizdeki kodların tamamlanmış hâli aşağıdaki gibidir:

```
Imports System.Data.OleDb
Public Class Form1
  Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
    Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
    Dim komut As New OleDbCommand
    komut.Connection = baglanti
    komut.CommandText = "Select TOP 1 * from CUSTOMERS"
    Try
      If baglanti.State = ConnectionState.Closed Then
        baglanti.Open()
        Dim ord As OleDbDataReader
```

```
            ord = komut.ExecuteReader
                        While ord.Read
                MessageBox.Show(ord("CUSTOMERID").ToString())
                End While
                baglanti.Close()
            End If
        Catch ex As Exception
            MessageBox.Show("Bağlantı Başarısız")
        End Try
    End Sub
End Class
```

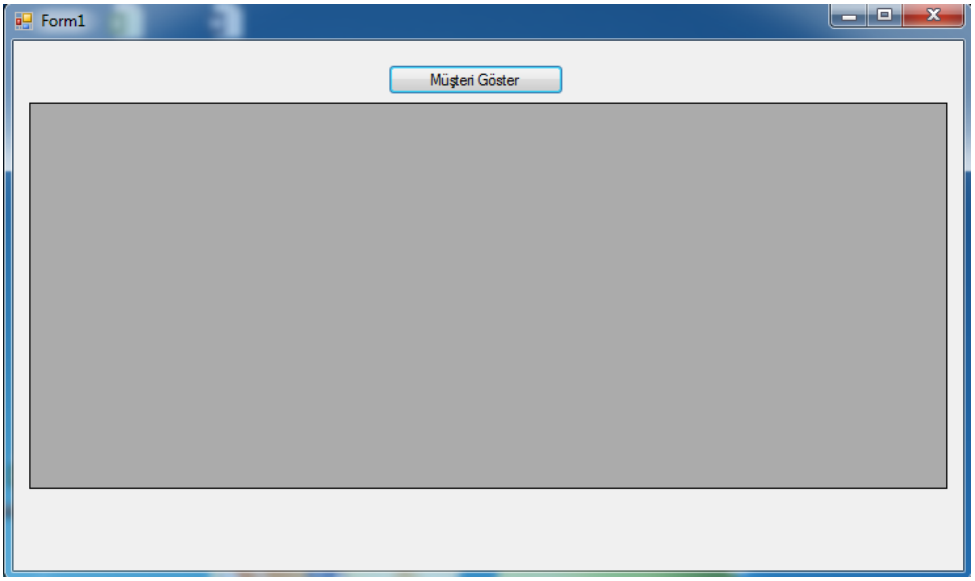
Kodlar incelendiğinde *Read()* metotu kullanıldıktan sonra DataReader'dan değerini okumak istediğimiz kolonun adını yazarak aldığımız görülmektedir. Ayrıca Read() metotunun while döngüsü ile kullanıldığı dikkat çekmektedir. Bunun sebebi Read() metotunun çalışma mantığının sıralı bir okuma işlemi ile aynı olmasıdır. Read() ile son kayıt okunduktan sonra artık okuma yapılmaz ve while döngüsünün dışına çıkılır.

DataAdapter verikaynağı ile uygulama arasında köprü görevi gören bir yapıdır.

OleDbDataAdapter

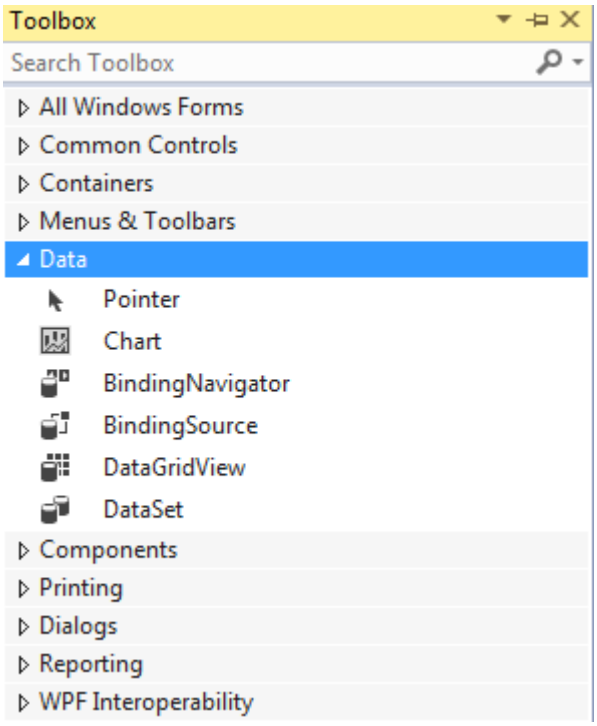
Bazı durumlarda her zaman veri kaynağına bağlantı kesintisiz sağlanamayabilir. Buna rağmen veriler üzerinde işlem yapmaya gerek duyulabilir. Hatta günümüzde birçok uygulamada bu sıklıkla karşılaşılan bir ihtiyaçtır. Örneğin internete erişimin maliyetli ve düşük hızda olduğu, kullanılan cihazların donanım özelliklerinin çok gelişmiş olmadığı mobil uygulamalarda veri üzerindeki işlemler bağlantısız ortamlarda yürütülebilir. Veri kaynağına bağlanılıp veriler çekildikten sonra bağlantı kesilse dahi daha önceden çekilmiş veriler üzerinde işlemler yapılarak uygulama çalışmaya devam eder. Üzerinde değişiklik yapılan veriler sonrasında veri kaynağına gönderilebilir. İşte bütün bu esneklikler ADO.NET'in sağlamış olduğu DataAdapter ve DataSet sınıfları ile mümkün olmaktadır.

DataAdapter veri kaynağı ile uygulama arasında köprü görevi gören bir yapıdır. Her ne kadar bağlantılı ve bağlantısız erişimde kullanılabilen bir yapı olsa da özellikle bağlantısız erişimi sağlamak için OleDbDataAdapter kullanılır. Aslında OleDbDataAdapter, OleDbDataReader'in bağlantılı ortamda yapmış olduğu işi yapmaktadır. OleDb ve SqlServer managed provider'larına özel DataAdapter sürümleri bulunmaktadır. Aşağıda verilen örnekte OleDbDataAdapter sınıfı kullanılarak NORTHWIND veritabanındaki Customers tablosundaki tüm müşteri kayıtlarının görütülenmesi işlemi gerçekleştirilmiştir (Resim 7).



Resim 7. Örnek Uygulama Ekran Tasarımı

Öncelikle bir Windows Forms uygulaması açalım. Form'a bir button bir de DataGridView kontrolü sürükleyelim. DataGridView kontrolüne ToolBox'ta Data sekmesinin altından ulaşabilirsiniz (Resim 8).



Resim 8. DataGridView ToolBoxta Data Sekmesi Altında Yer Alır.

Ekran tasarımını bitirdikten sonra kod aşamasına geçebiliriz. İlk olarak System.Data.OleDb isim uzayını çağırmamız gereklidir. Sonrasında OleDbConnection ve OleDbDataAdapter nesneleri oluşturacağız.

```
Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial
Catalog=NORTHWIND"
        Dim oadapter As New OleDbDataAdapter("Select * from
CUSTOMERS", baglanti)
```

Yukarıda görüldüğü üzere OleDbDataAdapter sınıfından nesne türetirken içerisinde gönderilecek Select sorgusu ve veri kaynağına erişmek için kullanılacak bağlantıyı belirleyebiliriz. Bunu yaptıktan sonra veriler çekilir ve çekilen veriler OleDbDataAdapter'in içerisine doldurulur. Gelen verileri bellekte tutmak için, yani veri kaynağı bağlantısını kapatarak bağlantısız ortamda çalışmak için ADO.NET'in sunmuş olduğu *DataTable* ya da *DataSet* sınıflarını kullanırız. Bu sınıfların kullanımını ilerde daha detaylı açıklayacağımız için şu an genel hatlarıyla bilmemiz yeterlidir. Aşağıdaki kod örneğinde bir DataSet nesnesi OleDbDataAdapter'ın Fill()metotu kullanılarak DataSet'e veri aktarımının nasıl yapıldığı görülmektedir.

Veri kaynağı bağlantısını kapatarak bağlantısız ortamda çalışmak için ADO.NET'in sunmuş olduğu DataTable ya da DataSet sınıfları kullanılır.

```
Dim ds As New DataSet
    baglanti.Open()
    oadapter.Fill(ds)
    baglanti.Close()
```

Burada dikkat edilmesi gereken noktalardan biri de *Fill()* metodu çalıştırılmadan önce baglantinin Open() metodu ile açılmasıdır. DataSete Fill() metodu ile DataAdapterdaki veri aktarıldığında artık geriye sadece DataGridView'de verileri görüntülemek kalmaktadır. Bunun için DataGridView'in DataSource özelliğine DataSetteki tabloyu bağlarız.

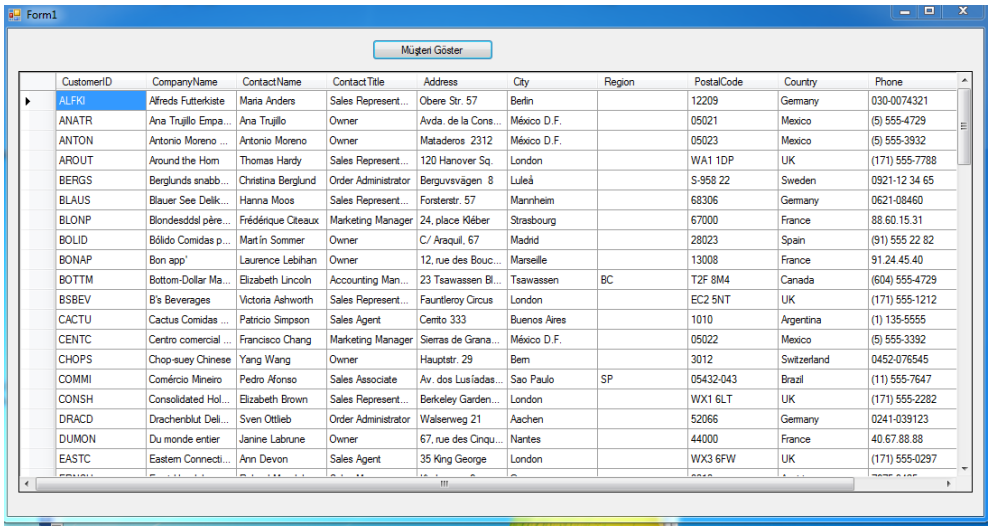
```
DataGridView1.DataSource = ds.Tables(0)
```

Yukarıdaki kod satırını da ekledikten sonra Customers tablosundan verileri getirip DataGridView'de görüntüleyen kodlarımızın bütünü şu şekilde olacaktır:

```
Imports System.Data.OleDb
Public Class Form1
  Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
    Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
    Dim oadapter As New OleDbDataAdapter("Select * from CUSTOMERS",
baglanti)
    Dim ds As New DataSet
    Try
      If baglanti.State = ConnectionState.Closed Then
        baglanti.Open()
        oadapter.Fill(ds)
        baglanti.Close()
        DataGridView1.DataSource = ds.Tables(0)
```

```
        End If
    Catch ex As Exception
        MessageBox.Show("Bağlantı Başarısız")
    End Try
End Sub
    End Class
```

Proje derlenip "müşterileri getir" butonuna basıldığında ekran görüntüsü Resim 9'teki gibi olacaktır.



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Represent...	Obere Str. 57	Berlin		12209	Germany	030-0074321
ANATR	Ana Trujillo Empa...	Ana Trujillo	Owner	Avda. de la Cons...	México D.F.		05021	Mexico	(5) 555-4729
ANTON	Antonio Moreno ...	Antonio Moreno	Owner	Mataderos 2312	México D.F.		05023	Mexico	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	Sales Represent...	120 Hanover Sq.	London		WA1 1DP	UK	(171) 555-7788
BERGS	Berglunds snabb...	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå		S-958 22	Sweden	0921-12 34 65
BLAUS	Blauer See Delik...	Hanna Moos	Sales Represent...	Forsterstr. 57	Mannheim		68306	Germany	0621-08460
BLONP	Blondesddsl père...	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg		67000	France	88.60.15.31
BOLID	Bólido Comidas p...	Martín Sommer	Owner	C/ Araquil, 67	Madrid		28023	Spain	(91) 555 22 82
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouc...	Marseille		13008	France	91.24.45.40
BOTTM	Bottom-Dollar Ma...	Elizabeth Lincoln	Accounting Man...	23 Tsawassen Bl...	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729
BSBEV	B's Beverages	Victoria Ashworth	Sales Represent...	Fauntleroy Circus	London		EC2 5NT	UK	(171) 555-1212
CACTU	Cactus Comidas ...	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires		1010	Argentina	(1) 135-5555
CENTC	Centro comercial ...	Francisco Chang	Marketing Manager	Sierras de Grana...	México D.F.		05022	Mexico	(5) 555-3392
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern		3012	Switzerland	0452-076545
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas...	Sao Paulo	SP	05432-043	Brazil	(11) 555-7647
CONSH	Consolidated Hol...	Elizabeth Brown	Sales Represent...	Berkeley Garden...	London		WX1 6LT	UK	(171) 555-2282
DRACD	Drachenblut Deli...	Sven Ottlieb	Order Administrator	Walserweg 21	Aachen		52066	Germany	0241-039123
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinqu...	Nantes		44000	France	40.67.88.88
EASTC	Eastern Connecti...	Ann Devon	Sales Agent	35 King George	London		WX3 6FW	UK	(171) 555-0297

Resim 9. Customers Tablosunun DataGridView'da Gösterimi

Bir DataSette birden fazla DataTable tutulabilir.

DataSet ve DataTable

ADO.NET bağlantısız erişim modelinde veri kaynağından gelen kayıtların tablo hâlinde bellekte depolanmasını sağlayan sınıf, DataSettir. Özellikle aralarında ilişki bulunan tabloların ilişkileriyle birlikte bellekte temsil edilmesi gereken durumlarda DataSet sınıfı kullanılır.

Veritabanından gelen tablolar DataSet nesnesine Tables kolleksiyonun bir elemanı olarak eklenir. Gelen tablolara bir kolleksiyonun elemanlarına nasıl erişiliyorsa öyle erişilir. Örneğin DataSet içerisinde 3 tablo varsa ve ikinci tabloya erişmek isteniyorsa DataSet.Tables(1) ya da DataSet.Tables("Tablo Adı") yazılmalıdır. Kısacası Tables kolleksiyonu elemanlarına erişim indeks numarası ya da erişilmek istenen tablonun adı ile olmaktadır.

DataTable sınıfı ise veritabanından gelen bir tabloyu bellekte saklar. *DataSetin bir alt kümesi şeklinde de değerlendirilebilecek olan DataTable, veritabanından gelen kayıtları tıpkı veritabanı tablosu gibi satırlarda tutar. Bu* satırlar DataTable içerisinde DataRow sınıfı türünde olan Rows isimli bir kolleksiyonda bulunmaktadır. DataTable içerisindeki her bir satıra bu koleksiyon kullanılarak erişilir.

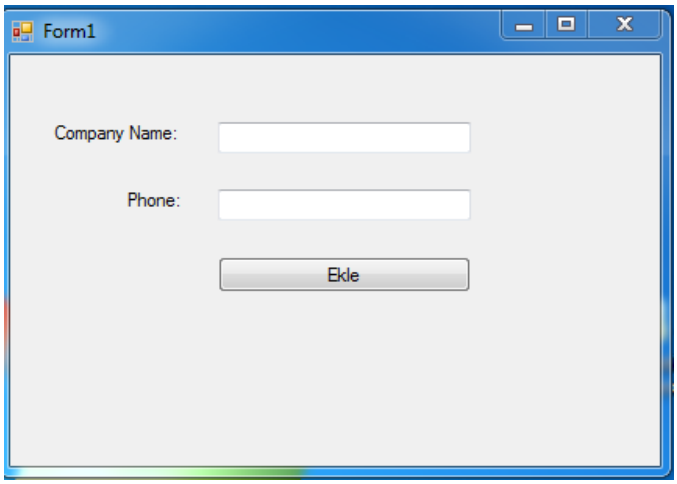
DataSet ve DataTable yapıları sadece veri tabanından gelen kayıtları tutmak için değil, aynı zamanda programın çalışma zamanında oluşacak kayıtların tutulması veya dönen kayıtların XML'e dönüştürülmesi gibi birçok farklı amaçlar ile de kullanılabilir. Bir sonraki ünitede DataSet ve DataTable kavramlarının kullanım şekilleri ve farklı özelliklerine daha detaylı bir şekilde değinilecektir.

ADO.NET İLE VERİ İŞLEME

Şu ana kadar ADO.NET sınıflarını kısaca tanıyarak basit veri sorgulama işlemleri yaptık. ADO.NET'te tıpkı veri seçiminde olduğu gibi veri ekleme,silme ve güncelleme de basit birkaç satır kod yazılarak yapılabilir. Örneklerimizde yine NORTHWIND veritabanını ve OleDb sınıflarını kullanacağız.

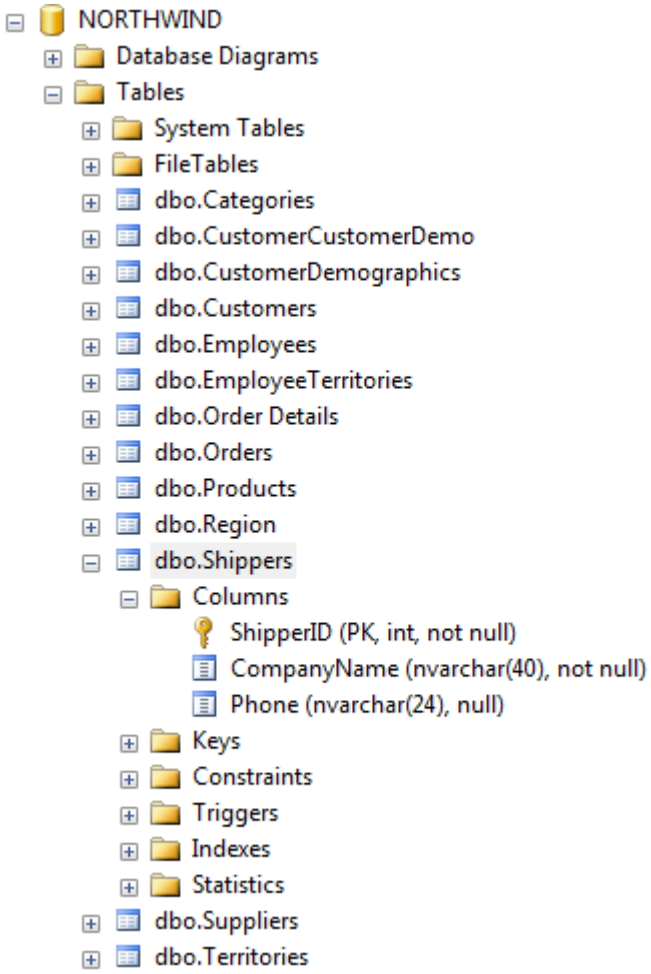
Veri Ekleme

OleDbCommand sınıfını anlatırken ExecuteNonQuery() isimli bir metottan kısaca söz etmiştik. Bu metot ile CommandText'te belirtilen ve herhangi bir kayıt döndürmeyen Insert, Update, Delete gibi SQL komutları çalıştırılabilmektedir. Aşağıda verilen örnekte NORTHWIND veritabanındaki Shippers tablosuna kayıt eklenmiştir.



Resim 10. Örnek Uygulama Ekran Tasarımı

Öncelikle ekranı Resim 10'daki görüldüğü gibi tasarlayalım. Veritabanında Company Name ve Phone alanlarına veri ekleyeceğimiz için sadece 2 TextBox koymamız yeterlidir. Resim 11'de NORTHWIND veritabanı ve Shippers tablosu görülmektedir.



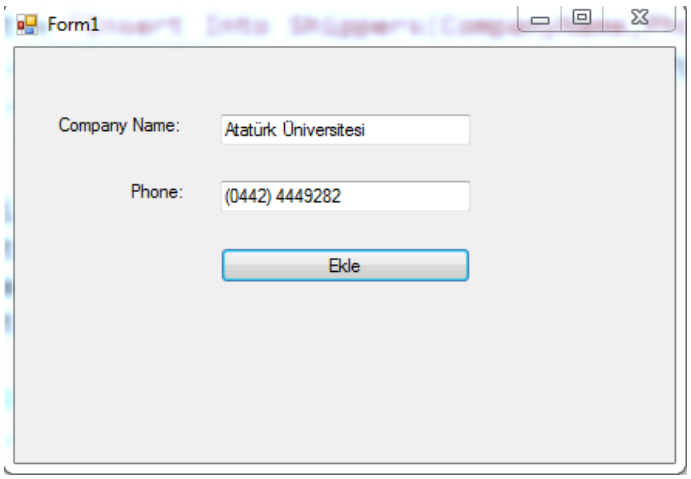
Resim 11. Northwind Veritabanı

ShipperID kolonu değeri otomatik artan "Primary Key" bir alan olduğu için bu kolona dışarıdan bir değer vermemiz söz konusu olamaz. Ekran tasarımını bitirip hangi kolonlara değer ekleneceğini belirledikten sonra artık kağıt ekleme için gerekli kodlarımızı yazabiliriz:

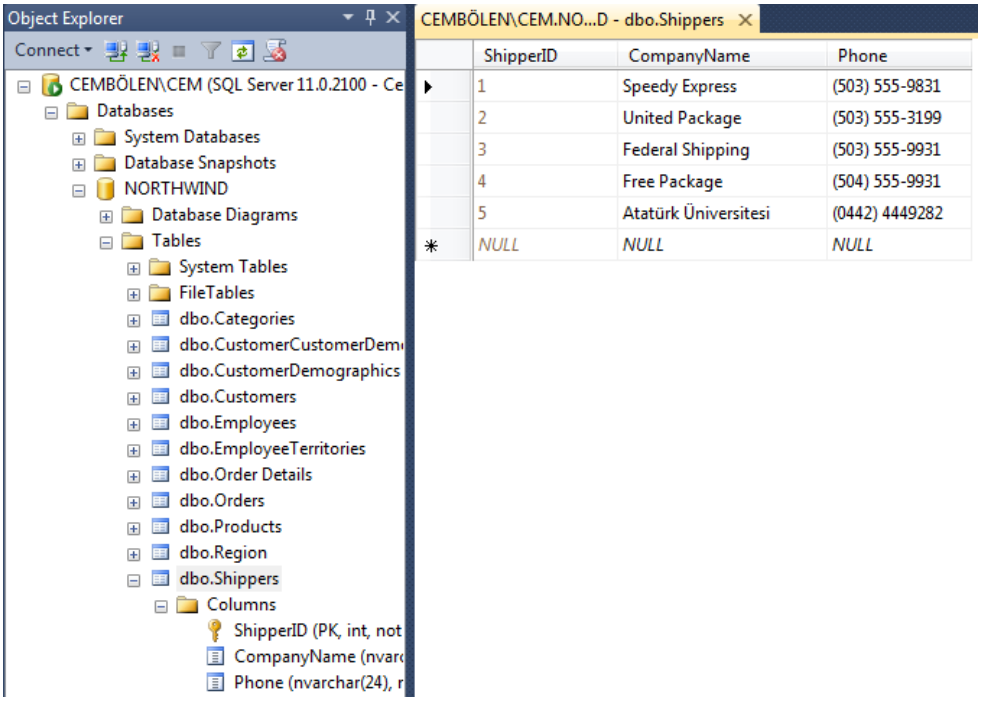
```
        Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLNCLI11;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
    Dim cmd As New OleDbCommand
    cmd.CommandText = "Insert Into Shippers(CompanyName,Phone) Values
(?,?)"
    cmd.Parameters.AddWithValue("@pCompanyName", txtCompanyName.Text)
    cmd.Parameters.AddWithValue("@pPhone", txtPhone.Text)
    cmd.Connection = baglanti
    Try
      If baglanti.State = ConnectionState.Closed Then
        baglanti.Open()
                cmd.ExecuteNonQuery()
              MessageBox.Show("Eklendi")
```

```
            baglanti.Close()
        End If
    Catch ex As Exception
        MessageBox.Show("Bağlantı Başarısız")
        End Try
```

CommandText özelliğinde Insert sorgumuzu yazdıktan sonra ? karakteri ile belirtilen kolonlara sırasıyla eklenecek değerler Parameters.AddWithValue ile gönderilmiştir. Son olarak ExecuteNonQuery() metodu ile sorgu çalıştırılarak veritabanına kayıtlar eklenmiştir.(Resim 12 ve Resim 13)



Resim 12. Eklenecek Veriler

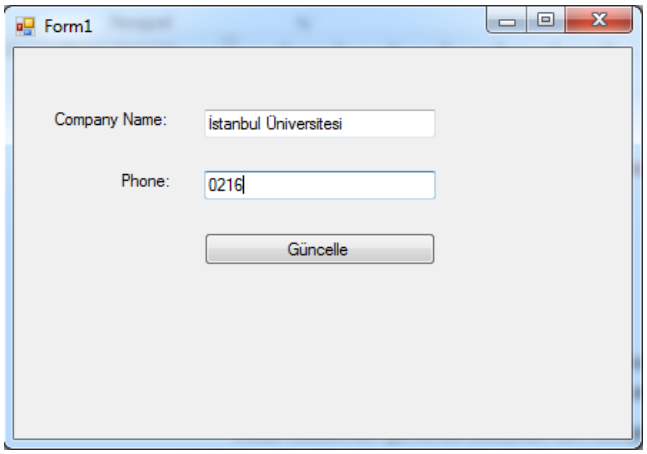


Resim 13. Shippers Tablosundaki Kayıtların Görünümü

VERİ GÜNCELLEME VE SİLME

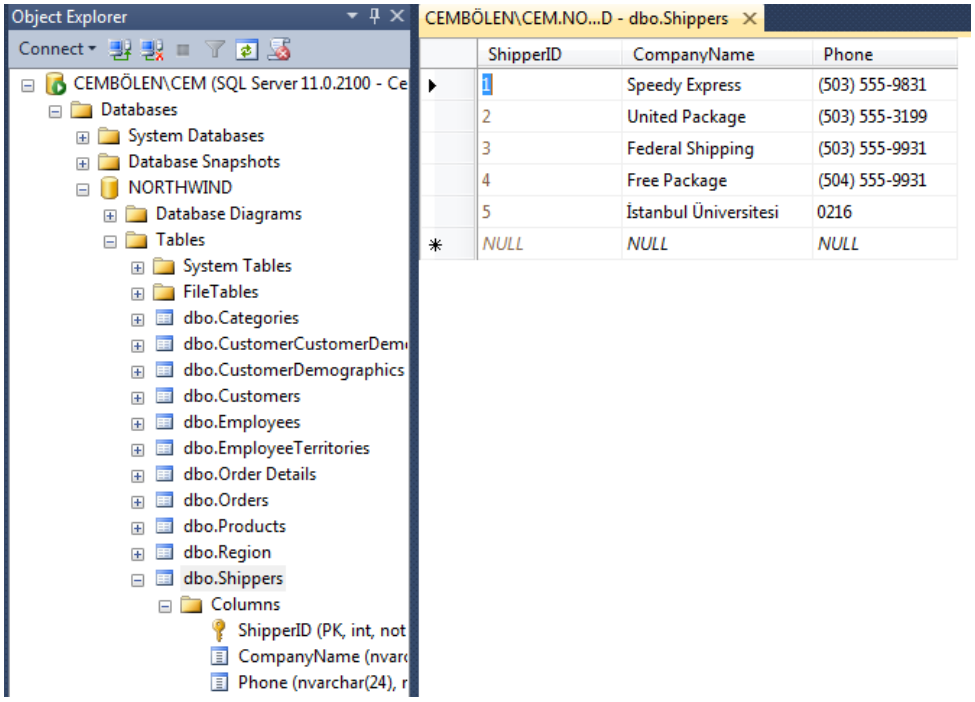
ADO.NET ile veri güncelleme ve silmenin, veri eklemeden en büyük farkı CommandText'e atanan komuttur. Aşağıda verilen kod örneğinde ShipperID değeri 5 olan kayıt güncellenmiştir.

```
Imports System.Data.OleDb
Public Class Form2
  Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
    Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLNCLI11;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
    Dim cmd As New OleDbCommand
    cmd.CommandText = "Update Shippers Set CompanyName=?,Phone=? Where
ShipperID=5"
    cmd.Parameters.AddWithValue("@pCompanyName", txtCompanyName.Text)
    cmd.Parameters.AddWithValue("@pPhone", txtPhone.Text)
    cmd.Connection = baglanti
    Try
      If baglanti.State = ConnectionState.Closed Then
        baglanti.Open()
        cmd.ExecuteNonQuery()
        MessageBox.Show("Güncellendi")
        baglanti.Close()
      End If
    Catch ex As Exception
      MessageBox.Show("Bağlantı Başarısız")
    End Try
  End Sub
    End Class
```



Resim 14. Güncellenecek Verilerin TextBoxlarda Görünümü

Ekrandan Resim 14'teki veriler girilip güncelle butonuna basıldığında veritabanındaki ShipperID 5 değerine sahip kayıt güncellenir. (Resim 15)

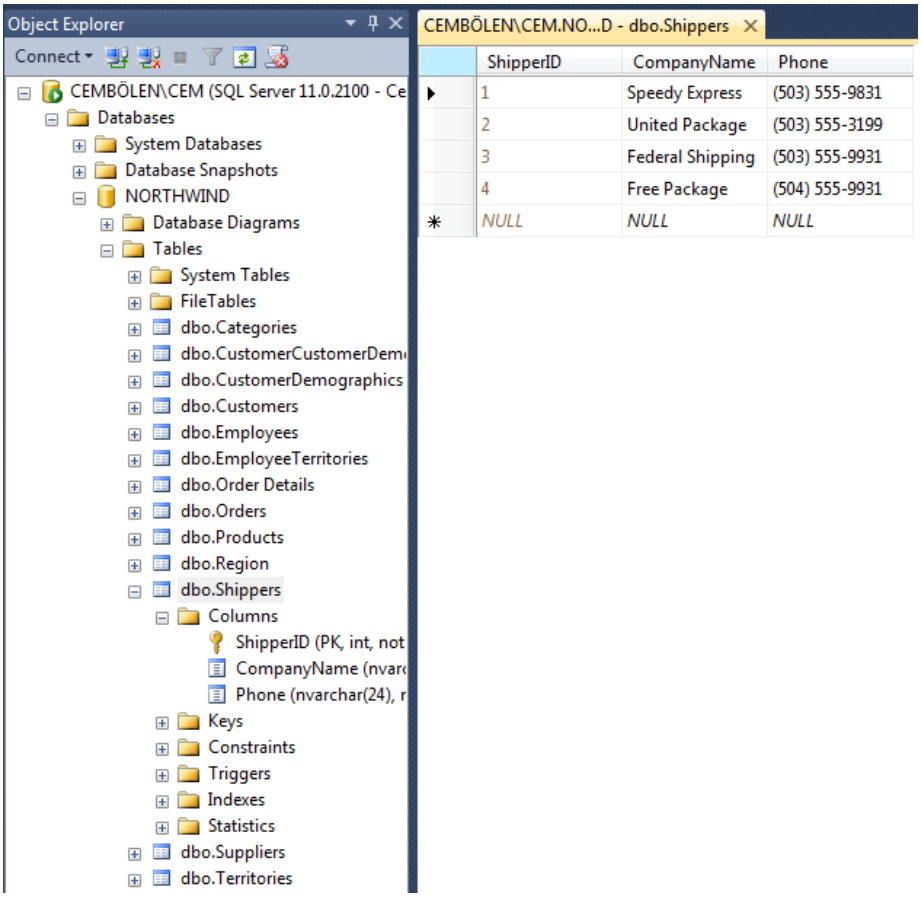


Resim 15. Güncelleme İşleminden Sonra Shippers Tablosundaki Kayıtların Görünümü

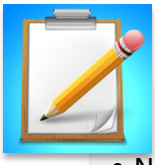
Son olarak ShipperID 5 değerine sahip kaydı veritabanından silelim. Bunun için aşağıdaki kodları yazmamız gerekecektir:

```
 Dim baglanti As New OleDbConnection
     baglanti.ConnectionString = "Provider=SQLNCLI11;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
     Dim cmd As New OleDbCommand
     cmd.CommandText = "Delete from Shippers Where ShipperID=5"
     cmd.Connection = baglanti
     Try
        If baglanti.State = ConnectionState.Closed Then
           baglanti.Open()
           cmd.ExecuteNonQuery()
           MessageBox.Show("Kayıt Silindi")
           baglanti.Close()
        End If
     Catch ex As Exception
        MessageBox.Show("Bağlantı Başarısız")
  End Try
```

Bu kodları çalıştırdıktan sonra veritabanından ilgili kaydın silindiğini göreceksiniz.(Resim 16)



Resim 16. Silme İşleminden Sonra Shippers Tablosundaki Kayıtların Görünümü



Bireysel Etkinlik

- Northwind veritabanını bilgisayarınıza indirip SQL Server'ınıza yükleyiniz. Bir Windows Form oluşturarak Northwind veritabanına bağlanınız ve Northwind veritabanı tabloları üzerinde ekleme, güncelleme ve silme işlemleri yapınız.

Özet

- ADO.NET, veri kaynağındaki (Veritabanı, metin, XML Dosyaları *vb.*) verilere erişmek veya bu verileri değiştirmek için geliştirilmiş bir veri erişim teknolojisidir.
- ADO.NET teknolojisi veriyi işlemek için iki yöntem sunar. Bu yöntemler bağlantılı (connected) erişim ve bağlantısız (disconnected) erişim olarak adlandırılmaktadır.
- Çeşitli veritabanlarına erişmek ve işlem yapabilmek için Managed Provider isimli sınıf kütüphaneleri kullanılır. Bu kütüphaneler, ADO.NET'in bağlantılı ve bağlantısız erişim yöntemlerinde de kullanılırlar.
- ADO.NET kütüphanesinde yer alan Connection sınıfları veritabanına bağlantı kurma ve bu bağlantıya ait bir takım kurallar veya özellikler belirlemek gibi işlemlerin yapılmasını yarayan fonksiyon ve sınıfları içermektedir
- Veri kaynağına erişim sağladıktan sonra OleDbCommand sınıfı kullanılarak veri kaynağı üzerindeki çeşitli komutlar çalıştırılır.
- OleDbDataReader sınıfı OleDbCommand sınıfı ile gönderilen komutların sonuçlarını okumak için kullanılır. Bir OleDbDataReader nesnesinin Read() metotu kullanılarak veri kaynağından gelen kayıtlar sırayla okunur ve tüm kayıtlar okunduktan sonra veri kaynağı ile bağlantı kesilir.
- DataAdapter, veri kaynağı ile uygulama arasında köprü görevi gören bir yapıdır.
- OleDbDataAdapter, OleDbDataReader'in bağlantılı ortamda yapmış olduğu işi yapmaktadır. OleDb ve SqlServer Managed Provider'larına özel DataAdapter sürümleri bulunmaktadır.
- ADO.NET bağlantısız erişim modelinde, veri kaynağından gelen kayıtların bellekte tablo hâlinde depolanmasını sağlayan sınıf DataSettir
- DataSetin bir alt kümesi şeklinde de değerlendirilebilecek olan DataTable, veritabanından gelen kayıtları tıpkı veritabanı tablosu gibi her bir kaydı satırlarda olmak üzere tutar.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "Bölüm Sonu Testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. **ADO.NET sınıfları aşağıdaki hangi isim uzayı altında yer almaktadır?**
   a) System.Data
   b) System.IO
   c) System.Security
   d) System.Printing
   e) System

2. **Aşağıdakilerden hangisi.NET içerisindeki managed providerlara özel sınıflardan biri değildir?**
   a) OleDbConnection
   b) OleDbCommand
   c) OleDbDataReader
   d) DataSet
   e) SqlDataAdapter

3. **Belirlenen bir veritabanına bağlanmak için aşağıdaki sınıflardan hangisi kullanılabilir?**
   a) OleDbConnection
   b) OleDbCommand
   c) SqlDataReader
   d) DataTable
   e) DataSet

4. ConnectionString içerisinde yer alan …… niteliğinde bağlanılacak veritabanının lokasyonu ya da dosya ismi verilir?

   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) Database
   b) Provider
   c) User ID
   d) Password
   e) DataSource

5. ……. sınıfı bağlantılı ve bağlantısız erişim modellerinde kullanılabilir ve veri transferinde bir köprü görevi görür.

   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?

   a) OleDbDataAdapter
   b) OleDbDataReader
   c) OleDbCommand
   d) DataTable
   e) DataSet

6. Bir OleDbDataReader nesnesinin …… metodu kullanılarak veri kaynağından gelen kayıtlar sırayla okunur ve tüm kayıtlar okunduktan sonra veri kaynağı ile bağlantı kesilir.

   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?

   a) NextResult()
   b) IsDbNull()
   c) Read()
   d) GetName()
   e) GetOrdinal()

7. **CommandText özelliğinde belirtilen, herhangi bir kayıt döndürmeyen (Insert,Update,Delete gibi), sorguların çalıştırılmasını sağlayan OleDbCommand sınıfı metodu aşağıdakilerden hangisidir?**

   a) ExecuteScalar()
   b) ExecuteNonQuery()
   c) ExecuteReader()
   d) Cancel()
   e) Prepare()

8. **OleDbDataAdapter'ın hangi metodu kullanılarak DataTable ya da DataSet'e veri aktarımı yapılır?**

   a) GetInsertCommand()
   b) Update()
   c) Dispose()
   d) Fill()
   e) GetType()

9. **Bir OleDbConnection nesnesinin veri kaynağına bağlı olup olmadığı aşağıdaki özelliklerden hangisi ile kontrol edilir?**
   a) ServerVersion
   b) PacketSize
   c) ConnectionTimeout
   d) State
   e) StaticsEnabled

10. **Bir OleDbCommand nesnesinin hangi özelliğine komut atanarak veri kaynağına gönderilir?**
   a) CommandType
   b) Connection
   c) CommandText
   d) CommandTimeout
   e) Transaction

Cevap Anahtarı

1.A, 2.D, 3.A, 4.E, 5.A, 6.C, 7.B, 8.D, 9.D. 10.C

YARARLANILAN KAYNAKLAR VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

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

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.

Grundgeiger, Dave. (2002). Programming Visual Basic. NET. O'Reilly, California.

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

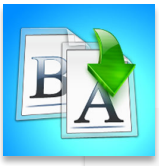
Schneider, D. I. (2013). An Introduction to Programming Using Visual Basic 2012. Prentice Hall Press,New Jersey.

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.

Taşdelen,Aykut,(2010),C# ile Veritabanı Programlama ve Ado.Net, Pusula Yayıncılık, İstanbul.

ADO.NET ile VERİ GÖRÜNTÜLEME



İÇİNDEKİLER

- ADO.NET ile Veriye Erişim
- Veritabanına Bağlantı
- DataGridView Kontrolü

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Sihirbazları kullanarak veritabanına kod yazmadan bağlantı kurabilecek,
  - ADO.NET sınıflarını kullanıp, kod yazarak veritabanına erişebilecek,
  - DataGridView kontrolünü tanıyacak,
  - Veritabanında varolan kayıtları arayüzden görüntüleyebileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Arş. Gör. Mehmet Cem BÖLEN

ÜNİTE

10

GİRİŞ

Bu ünitede veritabanında tutulan bilgilerin ADO.NET teknolojisi ve Visual Studio.NET ortamında yer alan hazır kontroller kullanılarak kullanıcıya nasıl sunulduğu üzerinde durulacaktır. Bu kapsamda .NET tarafından sunulan veri bağlama mimarisi kısaca incelenecek ve veri görüntülemede sıklıkla tercih kullanılan DataGridView kontrolü ile ilgili bilgi verilecektir.

ADO.NET İLE VERİYE ERİŞİM

Test Connection butonuna basarak veritabanına bağlanılıp bağlanılamadığınızı test edebilirsiniz.

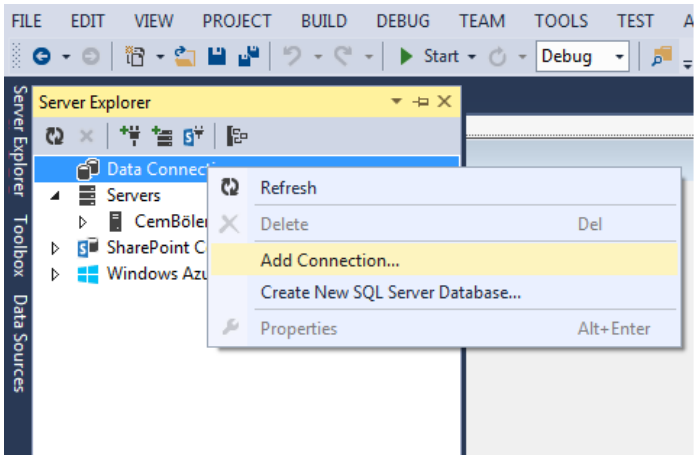
Veritabanında tutulan kayıtların görüntülenmesi sıkça karşılaşılan bir ihtiyaçtır. Bu ihtiyacı gidermeye yönelik .NET kütüphanesi tarafından Windows Forms.NET uygulamalarında veri bağlamanın yönetimi için standartlaştırılmış bir "veri bağlama mimarisi" (Data Binding Architecture) sunulmaktadır.

Visual Studio.Net tarafından veritabanına erişim iki genel kategoride incelenebilir. Bunlardan ilki sihirbazlar yardımıyla kod yazmadan veriye erişmek ve bu veriyi görüntülemektir. İkinci yöntem ise bağlantıyı sağlayacak komutlardan tutun da veritabanına gönderilecek sorgulara kadar herşeyin programcı tarafından oluşturulmasıdır. İkinci yöntem genellikle çok katmanlı mimariye sahip orta ve büyük ölçekli uygulamalarda kullanılırken, ilk yöntem çok kısıtlı zaman dilimlerinde geliştirilmesi istenen küçük projelerde tercih edilmektedir. Bu yöntemler ASP.NET ve Windows Forms.NET uygulamalarında kullanım şekli yönünden benzerdirler. Biz ünitemizde Windows Forms.NET uygulamalarında bu yöntemlerin kullanımını inceleyeceğiz.

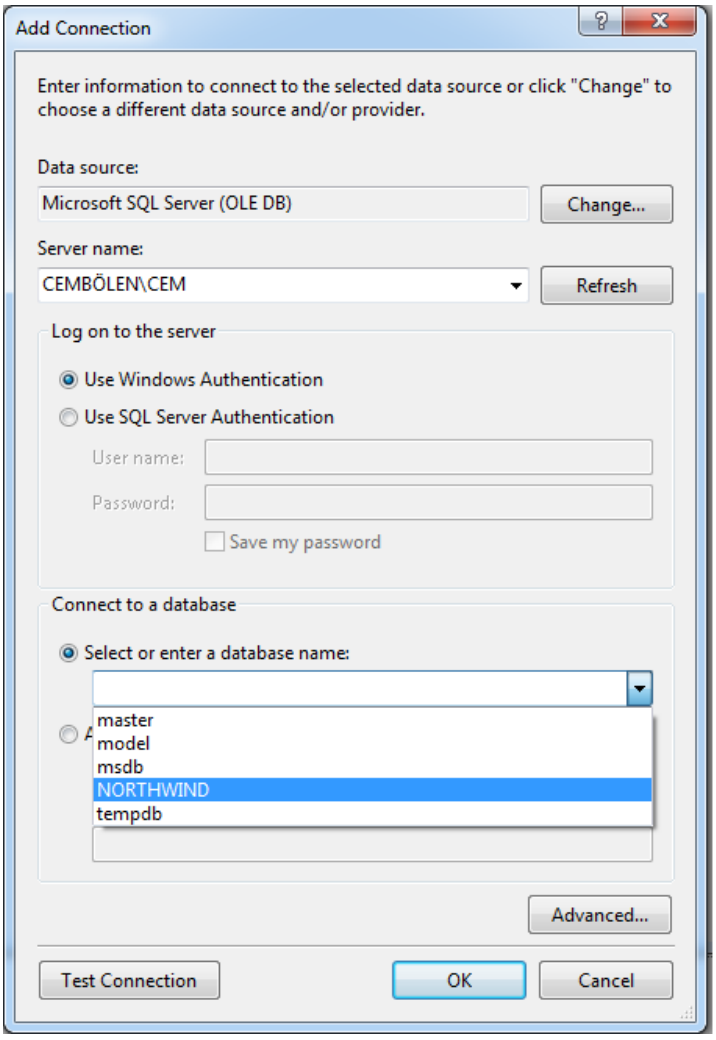
Veritabanına Bağlantı

Sihirbazları kullanarak kod yazmadan Microsoft SQL Server'da bulunan bir veritabanına bağlanmak için şu adımları takip etmek gereklidir.

- Server Explorer'da bulunan Data Connections seçeneğine sağ tıklayıp Add Connection komutunu verin (Resim 1). Eğer Server Explorer menüsünü göremiyorsanız View menüsünden Server Explorer alt menüsünü seçiniz ya da Ctrl+Alt+S kısayol tuş kombinasyonunu kullanınız.
- Karşınıza gelen ekrandan bağlanmak istediğiniz veritabanını seçiniz. Burada ayrıca veritabanına bağlanırken Data source kısmından hangi provider ile bağlanabileceğinizi belirleyebilirsiniz (Resim 2). Veritabanını seçtikten sonra Test Connection butonuna basarak veritabanına bağlanılıp bağlanılamadığını test edebilirsiniz.

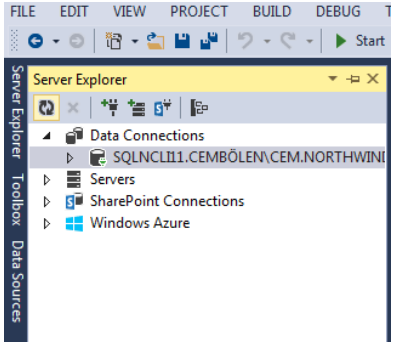


Resim 1. Add Connection Menü Seçeneği



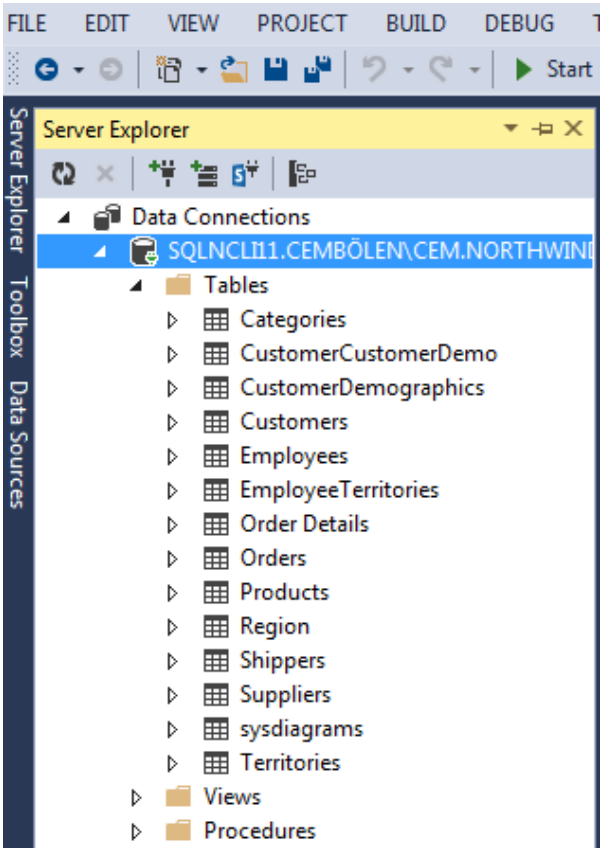
Resim 2. Penceresi Yerine Diyalog Kutusu

- Bu işlemler yapıldıktan sonra Data Connections seçeneğinin altında eklemiş olduğumuz bağlantını adı gelir (Resim 3).



Resim 3. Eklenen bağlantı Data Connections Altında Görülür

- Artık Norhwind veritabanına ait tablo, view ve procedurelere bağlantının altından ulaşabiliriz (Resim 4).



Resim 4. Northwind Veritabanı Tabloları Vs.Net Arayüzünden Görüntülenebilir.

Northwind veritabanı tabloları vs.Net arayüzünden görüntülenebilir.

Bu işlemleri gerçekleştirdikten sonra artık veriye erişebiliriz. Tabloların yanındaki ok işaretine tıklayarak alanları görüntüleyebilir ya da tablonun üstüne sağ tıklayıp Retrieve Data diyerek tablo içerisindeki verileri görüntüleyebiliriz. Ancak programcı olarak bu ekrandan bizim gördüğümüz verileri yazdığımız

programı kullananların da görmesi için çeşitli kontroller kullanırız. Verilere eriştikten sonra yapmamız gereken bunları ihtiyaca göre seçilen bir kontrolle kullanıcılara sunmaktır.

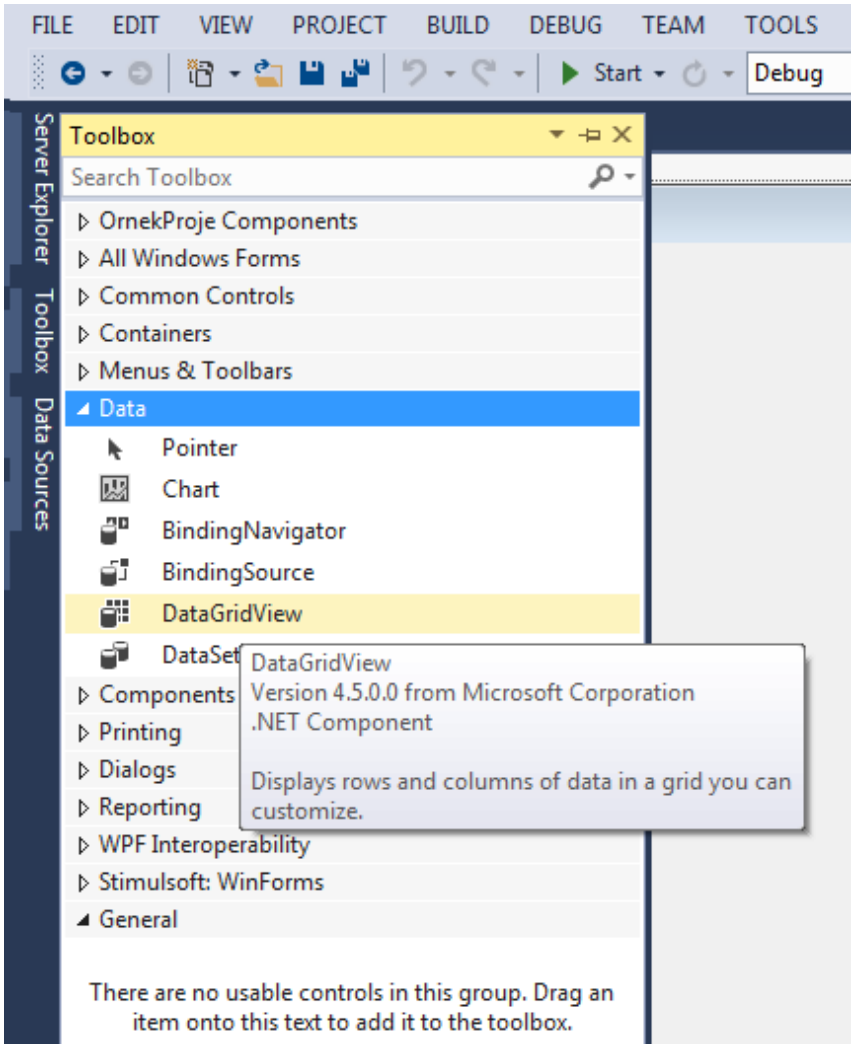
Datagridview Kontrolü

Veriyi satır ve sütun hâlinde görüntülemek için .NET kontrollerinden DataGridView kullanılır. DataTable, DataSet gibi yapılar DataGridView'e kolayca bağlanarak Excel dosyası formatına benzer bir şekilde görüntülenir.

DataGridView kontrolüne kod yazarak (programatik) verileri bağlayabileceğimiz gibi bu kontrolü forma sürükleyip bırakarak önceden oluşturulmuş bir veri bağlantısında yer alan tablolardaki verileri sihirbaz yardımıyla da bağlayabiliriz. Sihirbaz yardımıyla verileri bağlamak için aşağıdaki adımları takip etmemiz yeterlidir:

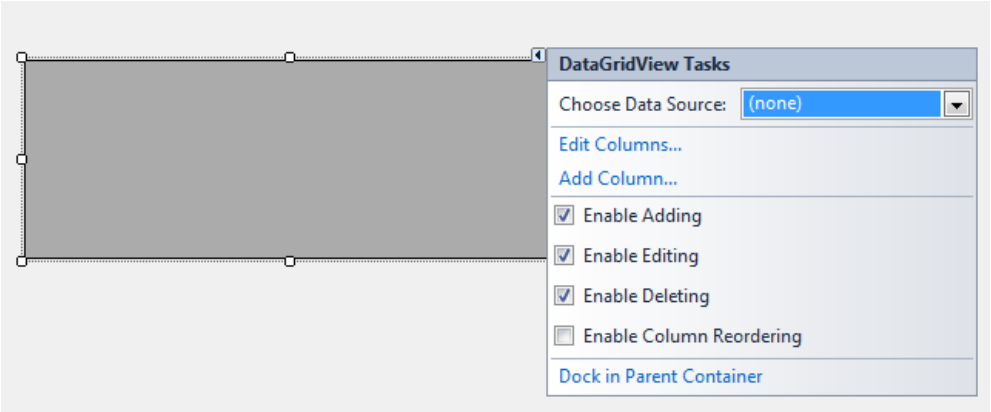
- Öncelikle Toolbox'ta Data sekmesi altında bulunan DataGridView kontrolünü formun üzerine sürükleyiniz. (Resim 5)

DataGridView, Data sekmesi altındadır.



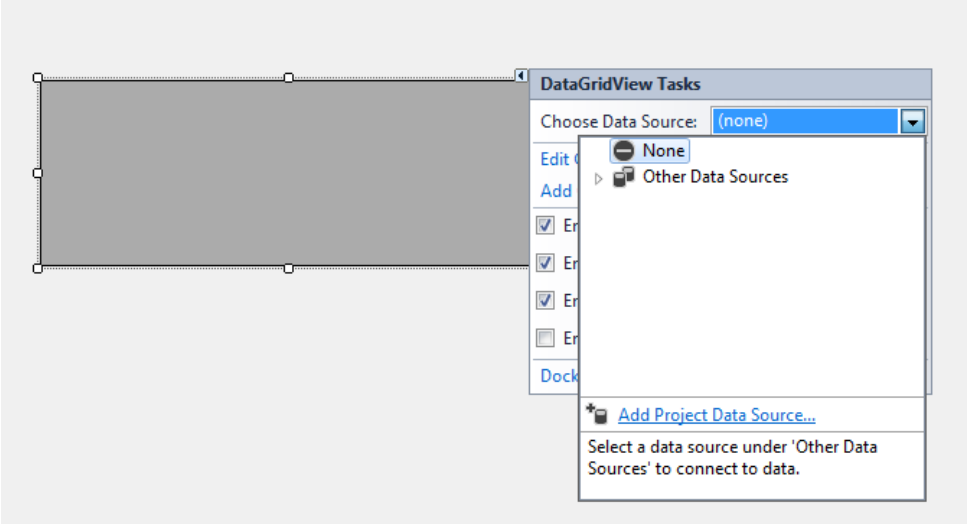
Resim 5. DataGridView Data sekmesi altındadır.

- DataGridView üzerindeki küçük ok işaretine tıklayarak DataGridView Tasks penceresini açınız (Resim 6).



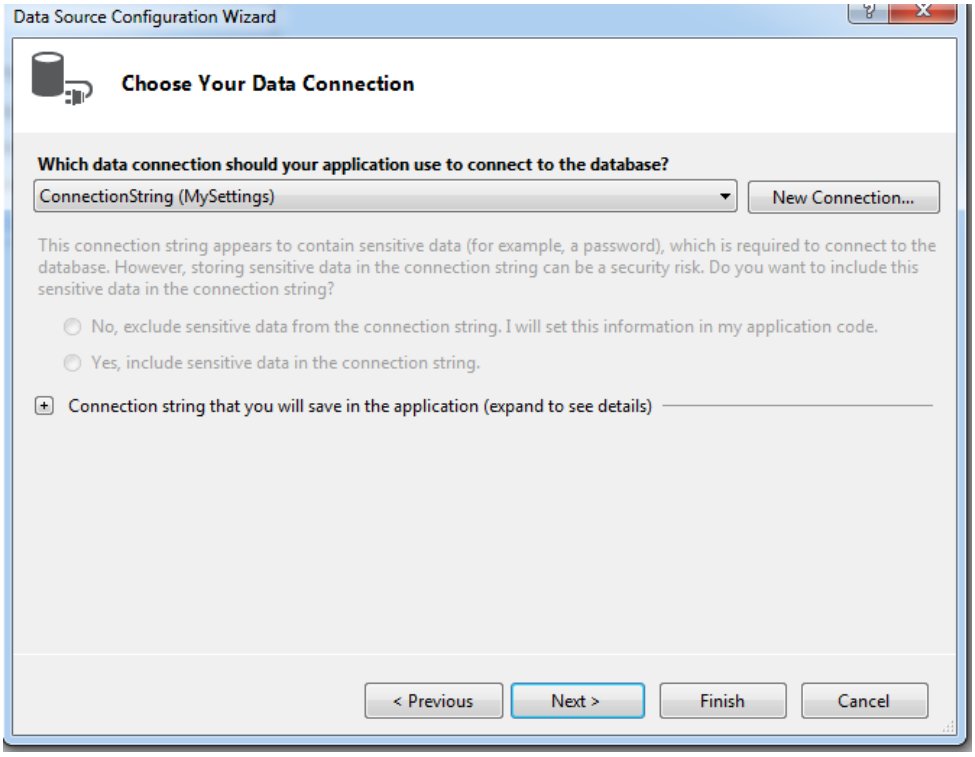
Resim 6 DataGridView Görev Çubuğu

- Bu pencerede Choose Data Source seçeneğinden Add Project Datasource seçeneğine tıklayınız (Resim 7).



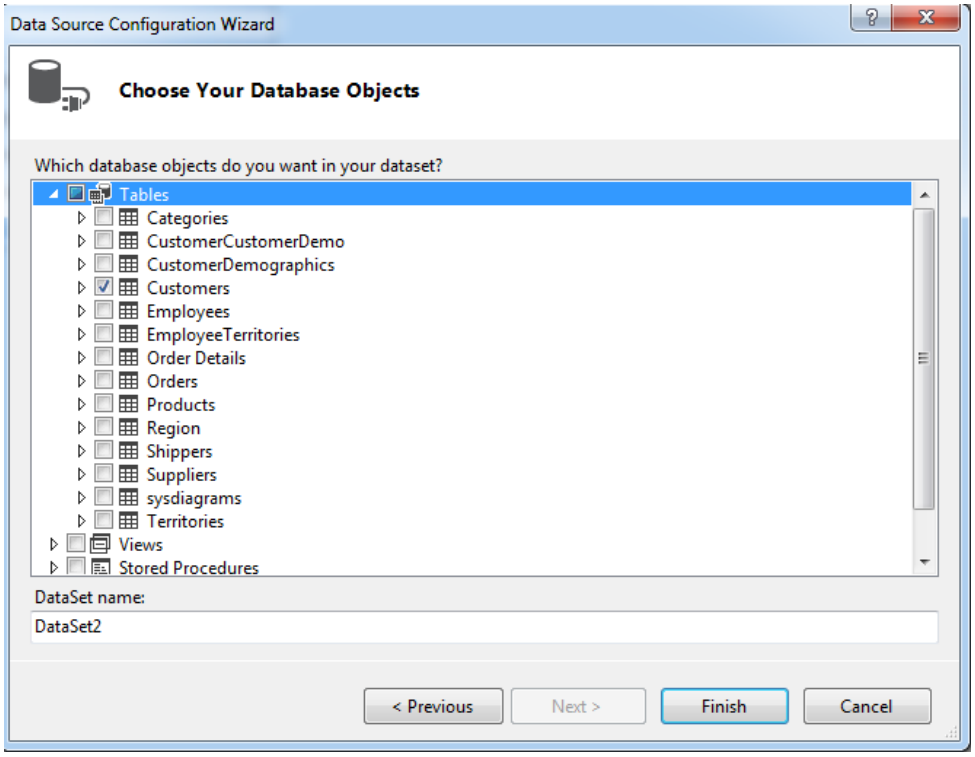
Resim 7 DataGridView Data Source Seçim Penceresi

- Bu işlemden sonra karşımıza çıkan pencereden sırasıyla Database, Dataset seçeneklerini seçip Next butonuna basıyoruz. Karşımıza biraz önce Data connectionsa eklemiş olduğumuz bağlantının seçili olarak geldiğini ve yanında eğer yeni bir bağlantı kurmamızı sağlayan new connection butonun yer aldığı ekran gelecektir. Bu ekrandan var olan bağlantıyı seçip next butonuna basalım. (Resim 8)



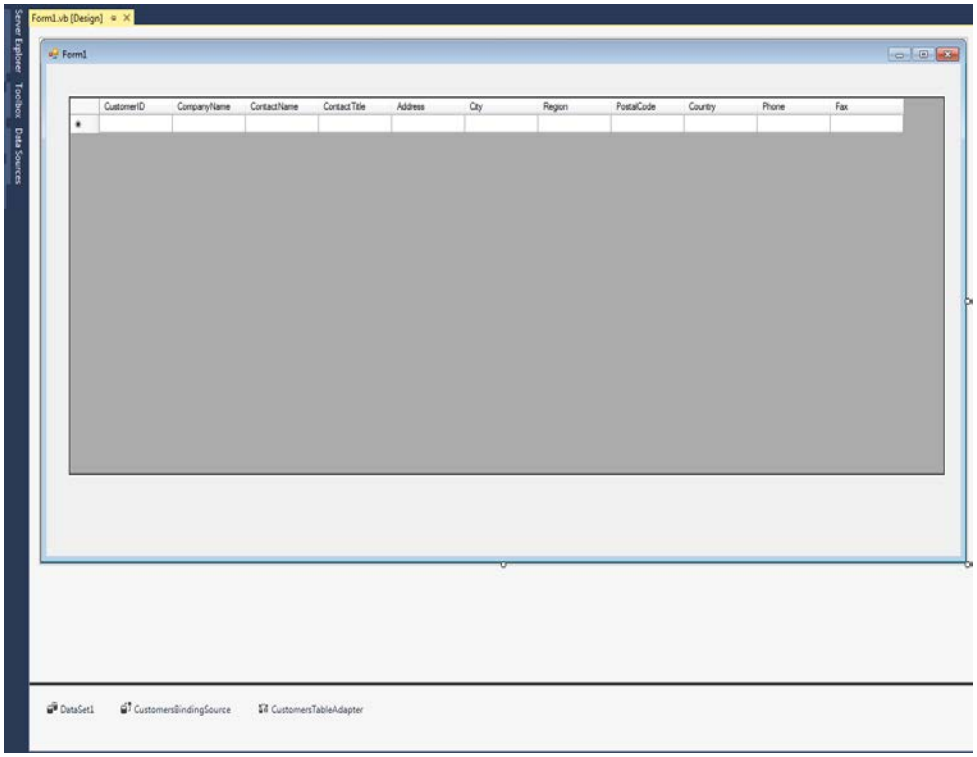
Resim 8 Bağlantı Seçim Ekranı

- Next butonuna basıldığında karşımıza veritabanında bulunan tablo, view ve procedurelerin seçiminin yapıldığı ekran çıkacaktır. Bu ekrandan DataGridView'da görüntülemek istediğimiz tabloları seçebiliriz. (Resim 9). Örneğimizde Customers tablosunu DataGridView'da görüntüleyeceğiz. Bu yüzden Customers tablosunu seçip Finish butonuna basarak işlemimizi tamamlıyoruz.
- Customers tablosundan gelen veriler otomatik olarak DataSet nesnesinde tutulacaktır.



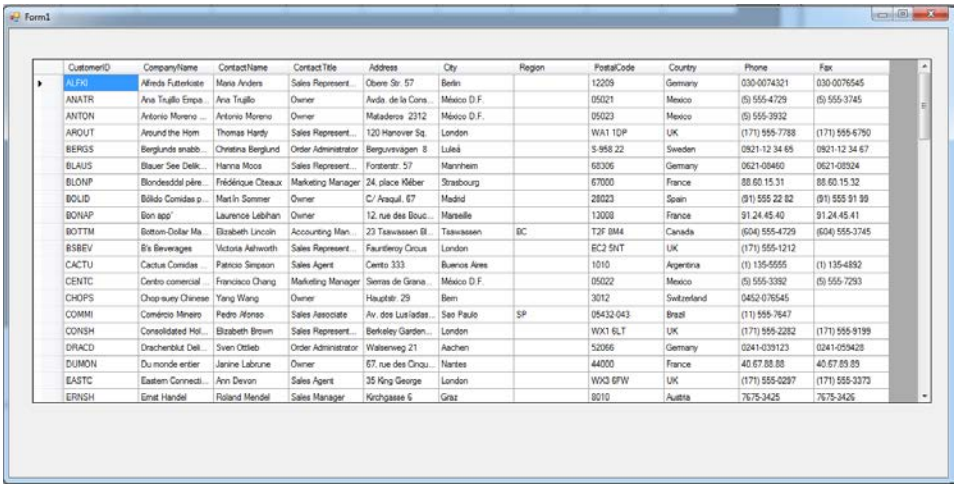
Resim 9 DataSete Eklenecek Tabloların Seçildiği Pencere

- Bu işlemler gerçekleştikten sonra DataGridView'de Customers tablosunun alanları görülür. Ayrıca forma BindingSource, TableAdapter ve DataSet nesneleri eklenir (Resim 10).



Resim 10 BindingSource, TableAdapter, DataSet, DataGridView Nesneleri

- Proje derlendikten sonra ekran görüntüsü şu şekilde olur. (Resim 11)



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Represent...	Obere Str. 57	Berlin		12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Empa...	Ana Trujillo	Owner	Avda. de la Cons...	México D.F.		05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno ...	Antonio Moreno	Owner	Mataderos 2312	México D.F.		05023	Mexico	(5) 555-3932	
AROUT	Around the Horn	Thomas Hardy	Sales Represent...	120 Hanover Sq.	London		WA1 1DP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabb...	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå		S-958 22	Sweden	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delik...	Hanna Moos	Sales Represent...	Forsterstr. 57	Mannheim		68306	Germany	0621-08460	0621-08924
BLONP	Blondesddsl père...	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg		67000	France	88.60.15.31	88.60.15.32
BOLID	Bólido Comidas p...	Martín Sommer	Owner	C/ Araquil, 67	Madrid		28023	Spain	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouc...	Marseille		13008	France	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dollar Ma...	Elizabeth Lincoln	Accounting Man...	23 Tsawassen Bl...	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745
BSBEV	B's Beverages	Victoria Ashworth	Sales Represent...	Fauntleroy Circus	London		EC2 5NT	UK	(171) 555-1212	
CACTU	Cactus Comidas ...	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires		1010	Argentina	(1) 135-5555	(1) 135-4892
CENTC	Centro comercial ...	Francisco Chang	Marketing Manager	Sierras de Grana...	México D.F.		05022	Mexico	(5) 555-3392	(5) 555-7293
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern		3012	Switzerland	0452-076545	
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas...	Sao Paulo	SP	05432-043	Brazil	(11) 555-7647	
CONSH	Consolidated Hol...	Elizabeth Brown	Sales Represent...	Berkeley Garden...	London		WX1 6LT	UK	(171) 555-2282	(171) 555-9199
DRACD	Drachenblut Deli...	Sven Ottlieb	Order Administrator	Walserweg 21	Aachen		52066	Germany	0241-039123	0241-059428
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinqu...	Nantes		44000	France	40.67.88.88	40.67.89.89
EASTC	Eastern Connecti...	Ann Devon	Sales Agent	35 King George	London		WX3 6FW	UK	(171) 555-0297	(171) 555-3373
ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6	Graz		8010	Austria	7675-3425	7675-3426

Resim 11 DataGridView'e verilerin bağlanmış hali

Konunun başında da belirttiğimiz gibi küçük projelerde sihirbaz yardımıyla bu tip hızlı çözümler uygulanabilir. Ancak birçok kurumsal, orta ve büyük ölçekteki projelerde veri görüntüleme işlemi kod yazarak gerçekleştirilir. ADO.NET'in sunmuş olduğu sınıflar ile sadece birkaç satır kod yazarak DataGridView'de veritabanındaki kayıtlar görüntülenebilir. Aşağıda verilen örnekte bağımsız veri erişim modeli kullanılarak Northwind veritabanındaki Customers tablosunda bulunan kayıtlar DataGridView kontrolünde gösterilmiştir.

```
Imports System.Data.OleDb
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim baglanti As New OleDbConnection
        baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
        Dim oadapter As New OleDbDataAdapter("Select * from CUSTOMERS",
baglanti)
        Dim ds As New DataSet
        baglanti.Open()
        oadapter.Fill(ds)
        baglanti.Close()
        DataGridView1.DataSource = ds.Tables(0)

    End Sub
End Class
```

DataSet ve DataTable sınıflarını kullanmak için System.Data isim uzayını çağırmak gereklidir. Eğer System.Data.SqlClient veya System.Data.OleDb isim uzayları çağrılmışsa System.Data'yı bir daha çağırmaya gerek kalmaz.

Verilen kodları daha iyi anlamak için satır satır kodları inceleyelim.

Öncelikle Windows Formun üzerine DataGridView kontrolünü sürükleyip bıraktıktan sonra formun boş bir yerine iki kez tıklayarak Load olayının içine girelim. Önce Public Class yazan yerine üzerinde System.Data.OleDb isim uzayını çağıran aşağıdaki kodu yazalım.

```
Imports System.Data.OleDb
```

İsim uzayını çağırdıktan sonra OleDbConnection sınıfından bir nesne türeterek ConnectionString özelliğine bağlanılacak veritabanı bilgilerini atayalım.

```
    Dim baglanti As New OleDbConnection
    baglanti.ConnectionString = "Provider=SQLOLEDB.1;Data
Source=CEMBÖLEN\CEM;Integrated Security=SSPI;Initial Catalog=NORTHWIND"
```

Baglanti isimli nesnemizi tanımlayıp, ConnectionString özelliğine değer atadıktan sonra sıra OleDbDataAdapter nesnemizi tanımlamaya geldi. OleDbDataAdapter veri kaynağından alınan verinin DataSet ya da DataTable'a taşınmasına yaradığı için mutlaka tanımlanması gerekmektedir. OleDbDataAdapter'ı aşağıdaki gibi tanımlarız.

```
    Dim oadapter As New OleDbDataAdapter("Select * from CUSTOMERS",
baglanti)
```

Parantez içerisinde odapter nesnesinin çalıştıracağı sorgu ve kullanılacağı bağlantı atanır. Buna göre Select * from CUSTOMERS sorgusu baglanti isimli nesnede belirtilen veritabanına gönderilecektir.

Bu işlemin ardından sıra, bir DataSet tanımlamaya geldi. DataSet ve DataTable gelen verileri bellekte tutan ve içerisine veriler aktarıldıktan sonra veritabanına bağlantıya gerek kalmadan tıpkı veritabanında işlem yapıyormuş gibi

veri üzerinde düzenlemeler yapmaya imkân tanıyan sınıflardır. DataAdapter ile verileri aldıktan sonra bunları DataSet nesnesine aktararak verileri bellekte tutup, veritabanına bağlantıyı keseceğiz. Böylece veritabanı için kullanmış olduğumuz kaynakları sisteme iade etmiş olacağız. DataSet tanımlaması aşağıdaki gibi yapılır.

```
Dim ds As New DataSet
```

DataSet tanımladıktan sonra artık bağlantı açıp DataAdapter ile verileri alarak bu verileri DataSet'e aktarabiliriz. Bunun için aşağıda verilen kodları yazıyoruz.

```
baglanti.Open()
oadapter.Fill(ds)
baglanti.Close()
```

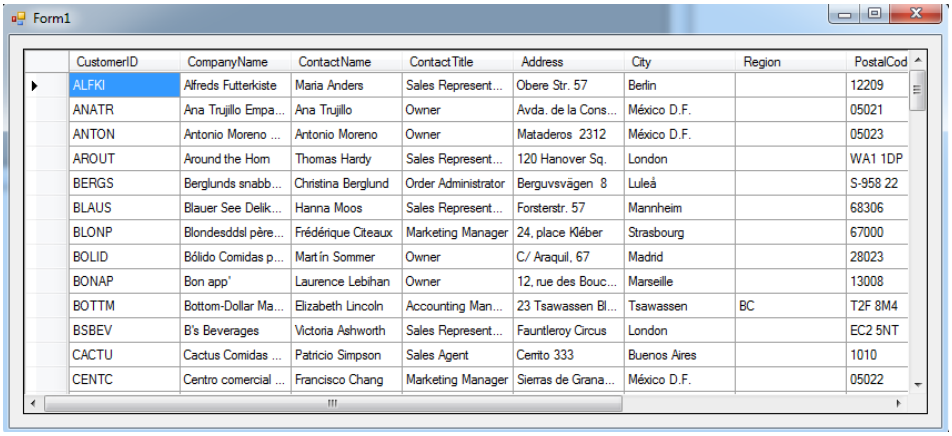
Open() metodu ile veritabanına bağlantı açıp DataAdapter'ın Fill metodu ile DataSet'e verileri dolduruyoruz. Sonrasında baglantıyı Close() metodu ile kapatıyoruz.

Bu işlemleri yaptıktan sonra geriye DataGridView nesnesine DataSet'te bulunan verileri bağlamak kaldı. Bunun için DataGridView'ın DataSource özelliğini kullanıyoruz.

```
DataGridView1.DataSource = ds.Tables(0)
```

Bu örnekte DataSet'in içerisinde tek tablo yer aldığı için ds.Tables(0) yazarak DataSource özelliğine tabloyu atıyoruz. DataSet yapısı gereği içerisinde birden fazla tablo bulundurabileceği için bağlamak istediğimiz tabloyu bir indeks numarası ile belirtmekteyiz.

Projeyi çalıştırdığımızda ekran görüntüsü şu şekilde olur. (Resim 12)



Resim 12. Uygulama Çalıştığında DataGridView

DataGridView her ne kadar alternatifleri bulunsa da veri görüntülemede programcılar tarafından en çok tercih edilen kontroldür. Bunda sağlamış olduğu farklı özelliklerin ve kullanım kolaylığının etkisi büyüktür. Aşağıda verilen tabloda DataGridView kontrolüne ait birtakım özellikler ve açıklamaları bulunmaktadır.

Tablo 1. DataGridView Kontrolü Özellikleri

Özellik		Açıklama
AllowUser ToAdd Rows	→	Kullanıcının yeni satır ekleyip eklemeyeceği iznini verir. True ya da False değeri alır.
AllowUser ToDelete Rows	→	Kullanıcıya DataGridView'den satır silme iznini verir. True ya da False değeri alır.
BackgroundColor	→	Arkaplan renginin belirlenmesi için kullanılır.
Cursor	→	Fare imleci üzerine geldiği zaman alacağı şekli belirler.
DefaultCellStyle	→	Hücrenin rengi, fontu gibi stil özelliklerinin belirlenmesi için kullanılır.
Multiselect	→	Çoklu satır seçme olanağı tanır.
Readonly	→	DataGridView'ı sadece okunabilir hâle getirir.

Özet

- Visual Studio.Net tarafından veritabanına erişim iki genel kategoride incelenebilir. Bunlardan ilki sihirbazlar yardımıyla kod yazmadan veriye erişmek ve bu veriyi görüntülemektir. İkinci yöntem ise bağlantıyı sağlayacak komutlardan tutun da veritabanına gönderilecek sorgulara kadar herşeyin programcı tarafından oluşturulmasdır.
- Veriyi satır ve sutun hâlinde görüntülemek için .NET kontrollerinden DataGridView kullanılır.
- DataGridView'e DataSet ve DataTable sınıflarından türetilen nesneler bağlanabilir.
- DataAdapter'ın Fill() metodu ile DataSet ya da DataTable doldurulur.
- DataGridView'ın DataSource özelliği sayesinde DataSet ve DataTable veri kaynakları bağlanır.
- DataSet ve DataTable sınıflarına erişmek için System.Data isim uzayını çağırmak gereklidir.

DEĞERLENDİRME SORULARI

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "Bölüm Sonu Testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

1. Visual Studio.Net'te sihirbazlar yardımıyla bir bağlantı eklenmek istendiğinde aşağıdaki menülerden hangisi kullanılır?
   a) Server Explorer
   b) Team Explorer
   c) Object Browser
   d) Solution Explorer
   e) Code Definition Window

2. Bir tabloda yer alan bütün kayıtları çekmek için aşağıdaki SQL komutlarından hangisi yazılmalıdır?
   a) Select * from tablo adı
   b) Select * from tablo adı where id=1
   c) Select * from tablo adı where id>1
   d) Select Top 1 * from tablo adı where id>1
   e) Select Top 1 * from tablo adı

3. Verinin tablo şeklinde görüntülendiği ve kullanıcı tarafından üzerinde düzenlemelerin de yapılabileceği .NET kontrolü aşağıdakilerden hangisidir?
   a) TextBox
   b) DataGridView
   c) DataView
   d) ListView
   e) Label

4. I. DataTable
   II. DataSet
   III. Datarow
   Yukarıdakilerden hangisi ya da hangileri DatGridView'e datasource olarak verilebilir?
   a) Yalnız I
   b) Yalnız II
   c) I ve II
   d) II ve III
   e) I, II ve III

5. …… sınıfı DataSet ve DataTable'a veri aktarmaya yarar.
   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) SqlConnection
   b) OleDbConnection
   c) OleDbCommand
   d) DataGridView
   e) OleDbDataAdapter

6. DataGridView kontrolünün ……. özelliği kullanılarak veriler bağlanır.
   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) ReadOnly
   b) Anchor
   c) Visible
   d) DataSource
   e) Cursor

7. Aşağıdaki kontrollerden hangisi veri görüntüleme için kullanılmaz?
   a) ListView
   b) DataGridView
   c) ListBox
   d) ReportViewer
   e) NotifyIcon

8. OleDbDataAdapter'ın ……. metodu kullanılarak DataTable ya da DataSet'e veri aktarımı yapılır.
   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) GetInsertCommand()
   b) Update()
   c) Dispose()
   d) Fill()
   e) GetType()

9. Bir OleDbConnection nesnesinin veri kaynağına bağlantı açmasını sağlamak için aşağıdaki metotlardan hangisi kullanılır?
   a) Close()
   b) CreateCommand()
   c) Open()
   d) Dispose()
   e) GetSchema()

10. DataTable ve DataSet hangi isim uzayı altında bulunur?
    a) System.Drawing
    b) System.Data
    c) System.Dynamic
    d) System.Deployment
    e) System.Configuration

Cevap Anahtarı

1.A,2.A,3.B,4.C,5.E,6.D,7.E,8.D,9.C,10.B

YARARLANILAN KAYNAKLAR VE BAŞVURULABİLECEK DİĞER KAYNAKLAR

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

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

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

Grundgeiger, Dave. (2002). Programming Visual Basic. NET. O'Reilly, California.

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

Schneider, D. I. (2013). An Introduction to Programming Using Visual Basic 2012. Prentice Hall Press,New Jersey.

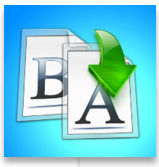
Sharp, John, (2009). Microsoft Visual Studio 2008 Step By Step, çev. Ümit Tezcan, Arkadaş Yayınevi, Ankara.

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

Taşdelen, Aykut, (2010), C# ile Veritabanı Programlama ve Ado.Net, Pusula Yayıncılık, İstanbul.

Türkoğlu, Tansu, (2009). Profesyonel Programlama Teknikleri .Net, Kalitte Bilgi Teknolojileri Basım ve Yayıncılık, Ankara.

XML İŞLEMLERİ



İÇİNDEKİLER

- XML'e Giriş
- XML Belgesi Oluşturma
- Visual Studio.NET ve XML
- XmlTextReader Sınıfı
- XmlTextWriter Sınıfı

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - XML teknolojisini tanıyacak,
  - XML belgesi oluşturabilecek,
  - Visual Studio.NET ile XML belgeleri üzerinde işlemler yapabileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Uzm. Orhan ÇELİKER

ÜNİTE

11

GİRİŞ

Teknolojinin gelişmesine paralel olarak ortaya çıkan farklı yazılım çözümlerinin temel problemlerinden biri de birbirileriyle verialış verişinde ortaya çıkan uyumsuzluktur. Birbirleriyle uyumsuz sistemler arasında veri alış verişini sorunsuzca gerçekleştirmek ve veri depolamayı kolaylaştırmak geliştirilmiş olan genişletilmiş işaretleme dilidir. Sağlamış olduğu kolaylık ve esnekliklerden ötürü kısa sürede kabul gören ve W3C(World Wide Web Consortium) tarafından tavsiye edilen bu işaretleme dili, Visual Studio.Net'ten Java'ya, SQL Server'dan Oracle'a kadar birçok farklı platform tarafından desteklenmektedir. Bu ünitede XML dilinin yapısı ve bu dil ile yapılacabilecek temel işlemler incelenecektir. Ayrıca kısaca Visual Studio.NET ile entegrasyonuna değinilecek ve örnek kodlar paylaşılacaktır.

"Extensible Markup Language" İngilizce ifadesinin baş harflerinden oluşan XML birçok kaynakta Türkçe'ye genişletilebilir işaretleme dili olarak çevrilmiştir.

XML NEDİR

"Extensible Markup Language" İngilizce ifadesinin baş harflerinden oluşan XML birçok kaynakta Türkçe'ye genişletilebilir işaretleme dili olarak çevrilmiştir. Yazımı ve sentaksı (syntax)'ı HTML'e benzeyen bu dil, hem insanların hem de bilgisayar sistemleri tarafından kolaylıkla okunabilecek dosyalar oluşturma ihtiyacından ötürü ortaya çıkmıştır. *XML'in sağlamış olduğu en büyük avantaj herhangibir platform ya da teknolojiye bağımlı kalmadan farklı sistemler arasında veri alış verişi yapma imkânı sunmasıdır.*

XML bir programlama dili değildir. XML bir işaretleme dilidir.

Ortaya çıkmasından sonra kısa sürede kabul gören XML, bugün birçok farklı alanda kullanılmaktadır. Bu alanlardan bazıları şunlardır:

- Veritabanı çözümleri
- E-posta iletileri
- Finansal verilerin alışverişi
- Web servisler ile veri iletişimi
- Yasal belgeleri toplamak ve e-ortamda paylaşıma sunmak

XML ve HTML birbirine benzer kavramlar olmakla birlikte ortaya çıkış ve kullanım amaçları tamamiyle farklıdır. Bu farkı en basit olarak şu şekilde anlatabiliriz. HTML belgelerin görsel özellikleri ile ilgilenirken, XML içerikleri ile ilgilenir. Ayrıca HTML Web teknolojileri için kullanılırken, XML'in kullanım alanı internet ile sınırlı değildir. XML ve HTML arasında daha birçok fark bulunmasına rağmen başlangıç seviyesinde bu temel farkları bilmek yeterlidir. Son olarak şuana kadar yaptığımız tanımlamalardan sonra XML teknolojisini tanımlarken özetle şu noktalara dikkat edilmelidir.

- XML aralarında teknolojiye yön veren firmaların bulunduğu W3C konsosiyumu tarafından geliştirilmiştir. Tamamen ücretsizdir.
- XML bir programlama dili değildir. XML bir işaretleme dilidir.
- Belirli standartları ve kuralları vardır. Bu durum geliştiriciler için daha belirgin bir alanda çalışmayı sağlar.

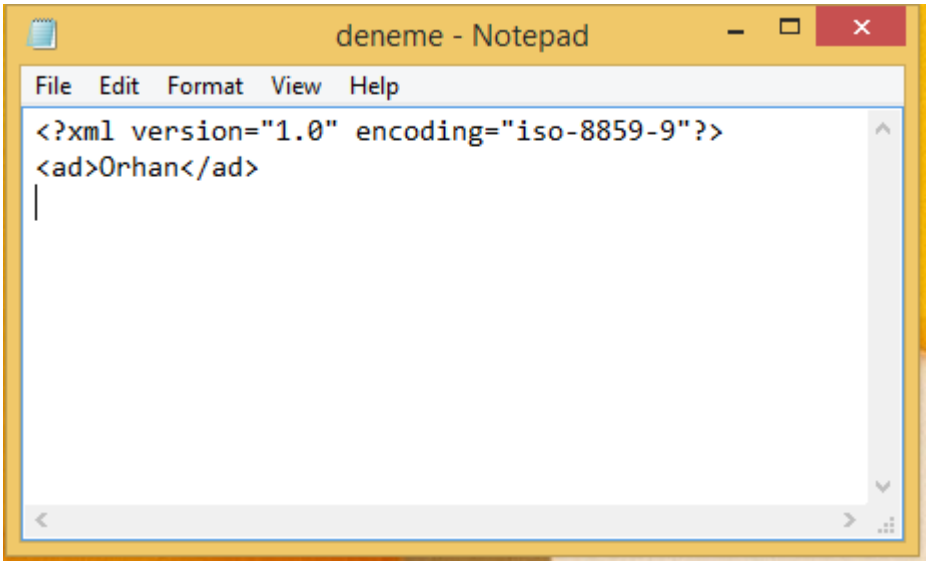
XML BELGESİ OLUŞTURMA

Bir XML belgesi oluşturmak için şu adımları takip etmeniz yeterlidir.

1) Belgenin hangi yapıda olacağını başında belirtirsiniz. Burada XML ve versiyon bilgisi açıkça belirtilir.
2) XML belgesinde hangi dil standardına göre kodlama yapılacağı belirtilir. Türkçe karakterlerinin dahil olduğu standart ISO-8859-9'dur.
3) İyi biçimlendirilmiş (well-formatted) bir belge oluşturmak için kök, etiket ve element yapısı doğru şekilde tanımlanmalıdır.

Bu adımları takip edip kolayca bir XML belgesi oluşturulabilir. Şimdi boş bir metin dosyası kullanarak XML belgesi oluşturalım.

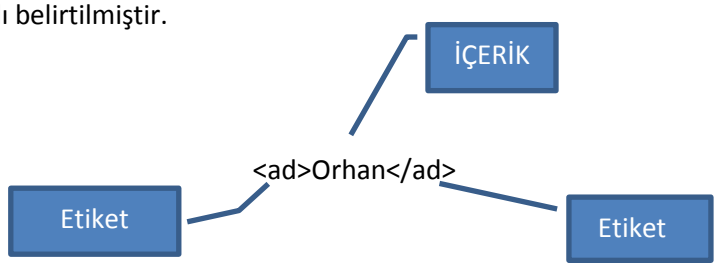
İlk olarak deneme isimli bir metin dosyası açalım ve Resim 1'deki gibi içini dolduralım.



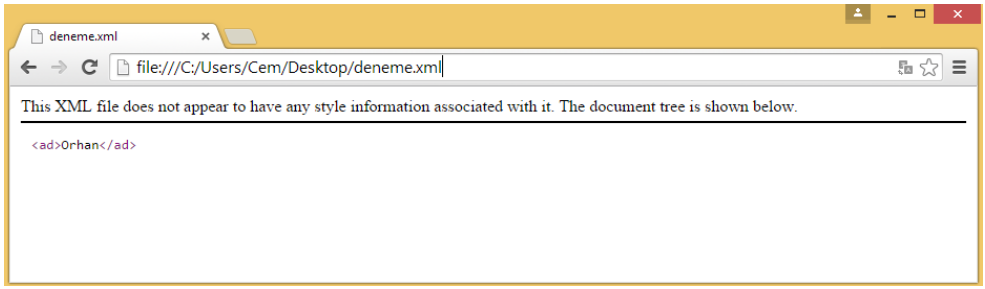
Resim 1. Metin dosyasından XML belgesi oluşturma

İlk satırda XML belgesinin versiyonu numarası ve hangi dil standartına göre kodlanacağı bilgisi yazılır. Bu yüzden bu satır giriş bölümü olarak adlandırılır. İkinci satırda ise etiketler (tag) arasına yazılmış olan veri bulunmaktadır. Etiket ve

veriden oluşan bu XML birimine element adı verilir. Bu birim aşağıda görsel olarak daha detaylı belirtilmiştir.



Bu işlemleri gerçekleştirdikten sonra sıra dosyayı XML olarak kaydetmeye geldi. Notepad'da açmış olduğumuz metin dosyasını üst menüden farklı kaydet seçeneğini seçerek uzantısını xml olarak değiştirip kaydedelim. Sonrasında bu dosyası herhangi bir tarayıcıda ya da xml editöründe açtığımızda aşağıdaki gibi görüntülenir. (Resim 2)



Resim 2. Google Chrome'da XML belgesi görünümü

XML belgeleri her zaman yukarıda oluşturduğumuz kadar basit içeriğe sahip olmayabilir. Daha açık ifade etmek gerekirse bazı durumlarda veriler kategorilere ya da gruplara ayrılmak istenebilir. Bu durum için XML'in sunmuş olduğu bazı yapılar kullanılır. Şimdi bu yapıları kısaca tanıyalım.

XML belgesi içerisinde kategori oluşturmak için kök elementi oluşturmak gereklidir.

KÖK (ROOT) OLUŞTURMA

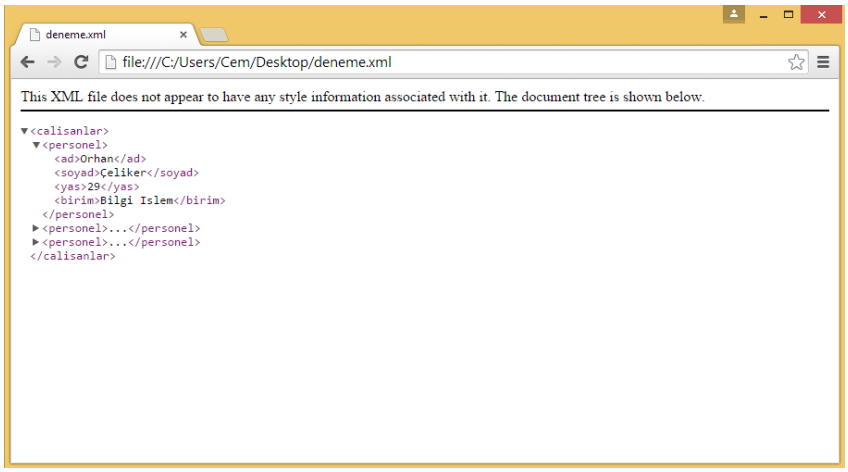
XML belgesi içerisinde kategori oluşturmak için kök elementi oluşturmak gereklidir. Kök elementini ağaç yapısına benzetebiliriz. Bir kök elementi başka elementler barındırabilir. Aşağıda personel isimli bir kök elementi oluşturulmuş ve bu element içerisinde personele ait ad, soyad, yas ve birim bilgileri saklanmıştır.

```
<personel>
  <ad>Orhan</ad>
  <soyad>Çeliker</soyad>
  <yas>29</yas>
  <birim>Bilgi Islem</birim>
    </personel>
```

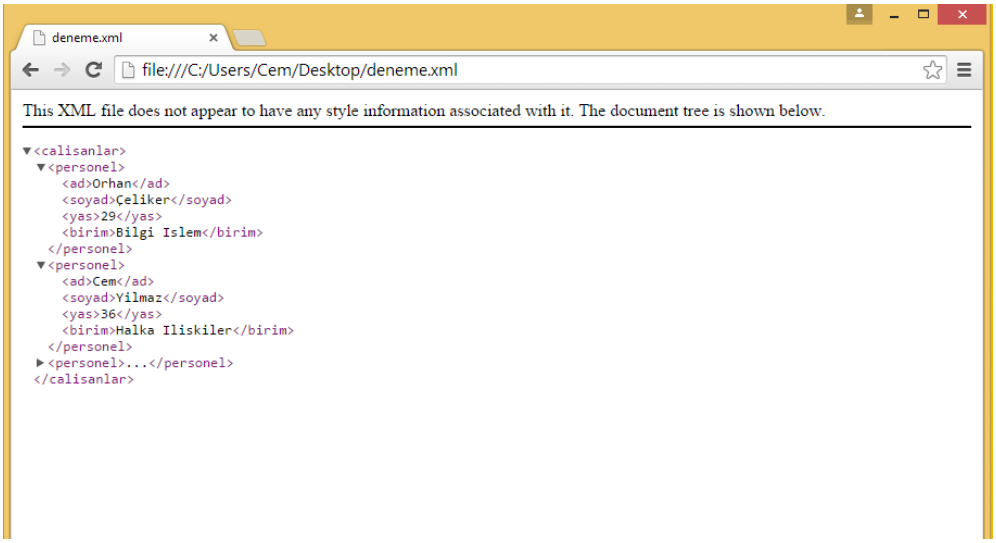
Yukarıdaki belgede ad, soyad, yas ve birim elementleri personel elementi altında toplanmıştır. Birden fazla personel olduğunu düşünelim. Bunları da calisanlar isimli başka bir kök altında toplayalım. Bu sefer XML belgesini şu şekilde tanımlamamız gerekecektir.

```
<calisanlar>
  <personel>
    <ad>Orhan</ad>
    <soyad>Çeliker</soyad>
    <yas>29</yas>
    <birim>Bilgi Islem</birim>
  </personel>
  <personel>
    <ad>Cem</ad>
    <soyad>Yilmaz</soyad>
    <yas>36</yas>
    <birim>Halka Iliskiler</birim>
  </personel>
  <personel>
    <ad>Ahmet</ad>
    <soyad>Arif</soyad>
    <yas>32</yas>
    <birim>Pazarlama</birim>
  </personel>
   </calisanlar>
```

XML belgesi bu şekilde oluşturulduktan sonra tarayıcıda şu şekilde görüntülenir (Resim 3 ve Resim 4).



Resim 3 İlk Sıradaki Personel Kökü



Resim 4. Deneme.xml dosyası görünümü

NİTELİK TANIMLAMA

Bir XML belgesinde açılan etiket içerisinde nitelik (attribute) tanımlanarak element ile ilgili açıklayıcı bilgilere yer verilebilir.

Bir XML belgesinde açılan etiket içerisinde nitelik (attribute) tanımlanarak element ile ilgili açıklayıcı bilgilere yer verilebilir. Aşağıda verilen örnekte tanımlanan elemente eğitim durumu nitelik olarak eklenmiştir.

```
<calisanlar>
  <personel egitim="Doktora">
    <ad>Orhan</ad>
    <soyad>Çeliker</soyad>
    <yas>29</yas>
    <birim>Bilgi Islem</birim>
  </personel>
  <personel egitim="Lisans">
    <ad>Cem</ad>
    <soyad>Yilmaz</soyad>
    <yas>36</yas>
    <birim>Halka Iliskiler</birim>
  </personel>
  <personel egitim="Lise">
    <ad>Ahmet</ad>
    <soyad>Arif</soyad>
    <yas>32</yas>
    <birim>Pazarlama</birim>
  </personel>
   </calisanlar>
```

Verilen XML kodlarında egitim isimli nitelik tanımlanmış ve niteliğe sırasıyla Doktora, Lisans ve Lise değerleri girilmiştir. Nitelik değerleri sayesinde element ile ilgili ek bilgiler verilebilir.

Nitelik tanımlarken dikkat edilmesi gereken bazı kurallar vardır. Bunlar;

1) Bir etiket içerisinde birden fazla aynı nitelik adı kullanılmamalıdır.
2) Nitelik değerleri çift tırnak ya da tek tırnak arasına yazılmalıdır.
3) Nitelik değerleri < ve & karakterlerini içermemelidir.

Bu kurallara dikkat edilmediği takdirde XML dosyası görüntülenemez ve hata verir.

VS.NET İLE XML DOSYASI İŞLEMLERİ

Ünitenin başında XML'in birçok farklı platform ve teknoloji tarafından desteklendiğini belirtmiştik. Visual Studio.NET de XML teknolojisine destek vermekte olup XML belgeleri ile çalışabilmek için çeşitli sınıflar sunulmaktadır. Bu sınıflar System.XML isim uzayı (alanı) altında yer almaktadır. Bu sınıflardan en çok kullanılanları şu şekildedir:

XmlReeader: XML belgelerinden veri okumak için kullanılan sınıftır. Tıpkı DataReader'da olduğu gibi sadece ileri yönlü okuma yapılır.

XmlTextReader: XML belgelerinden sadece ileri yönde okuma yapabilir ve XML verilerine bir stram gibi erişmemizi sağlar. XmlReader sınıfından türemiştir.

XmlWriter: XML belgesi oluşturmak ve XML belgesine veri kaydetmek için gerekli metotları ve özellikleri içinde barındırır.

Bir XML belgesindeki her bir satır bir düğümü ifade etmektedir.

XmlTextWriter: XML streamlerini dosya sisteminde oluşturmak için kullanılır. XMLWriter sınıfından türemiştir.

XmlDocument: Bir XML belgesinin belleğe ağaç yapısı şeklinde yüklenmesi ve bu yapı üzerinde çeşitli işlemler yapılabilmesini sağlar.

XmlNode: XML dosyasında tek bir XML düğümünü(node) elde etmek için kullanılır.

Bir XML belgesindeki her bir satır bir düğümü ifade etmektedir. Düğümlerin bazıları kök, bazıları ise başlangıç ya da sonlandırma düğümleridir. Bu düğümlerin bazılarında nitelik bilgisi varken bazılarında yoktur.

Visual Studio.NET ile XML belgelerinden veri okumanın birkaç yolu vardır. Bu ünitede Visual Studio tarafından XML belgelerinde okuma-yazma yapmak için sunulan XMLTextWriter ve XMLTextReader sınıflarını inceleyelim.

XmlTextReader ve XmlTextWriter

System.XML isim alanı altında bulunan XmlTextReader, bir XML belgesinden okuma yapmamızı sağlar.

System.XML isim alanı altında bulunan XmlTextReader, bir XML belgesinden okuma yapmamızı sağlar. XmlTextReader sınıfı ile aşağıdaki yöntemlerle XML belgesinden okuma yapılabilir.

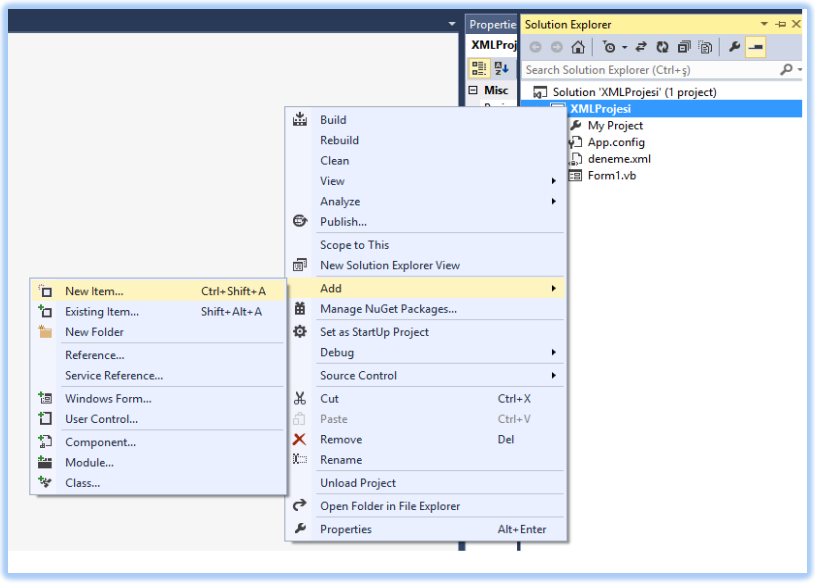
```
Dim dosyaadi As String = "deneme.xml"
Dim okuyucu As New XmlTextReader(dosyaadi)
```

Ya da

```
Dim dosyaadi As String = "deneme.xml"
Dim stream As New FileStream(dosyaadi,
FileMode.Open)
Dim okuyucu As New XmlTextReader(stream)
```

XmlTextReader nesnesi oluşturulduktan sonra Read() metodu kullanılarak XML belgesinden okuma yapılır. XmlTextReader sınıfının Read() metodu çalışma algoritması olarak DataReader'ın Read() metotuna benzerdir. Sadece ileri yönlü bir okuma yapar ve okuma bitince False değeri döndürür. Read() metotu ile okuma yaparken XML belgesinden her bir düğüm ayrı ayrı ele alınır. Read() metotunu daha iyi anlamak için aşağıdaki örneği inceleyelim.

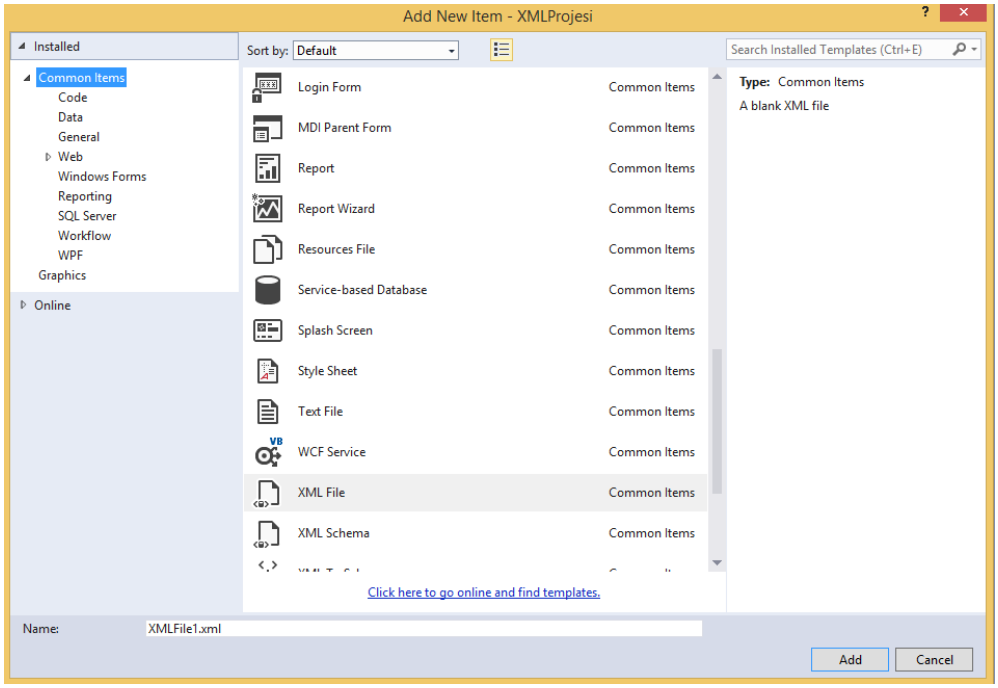
İlk olarak Visual Studio.NET ortamında bir Windows Forms projesi oluşturalım. Sonrasında projenin üstüne sağ tıklayıp Add seçeneğini seçtikten sonra New Item komutuna tıklayalım (Resim 5).



Resim 5. Deneme.xml dosyası görünümü

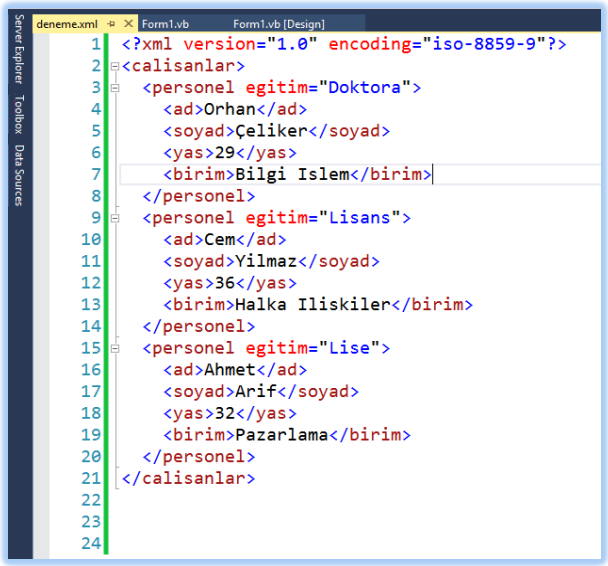
.NET Framework konfigürasyon dosyalarının birçoğu XML tabanlıdır.

Karşımıza açılan pencerede XML File nesnesini seçerek projemize XML belgemizi ekliyoruz. (Resim 6)



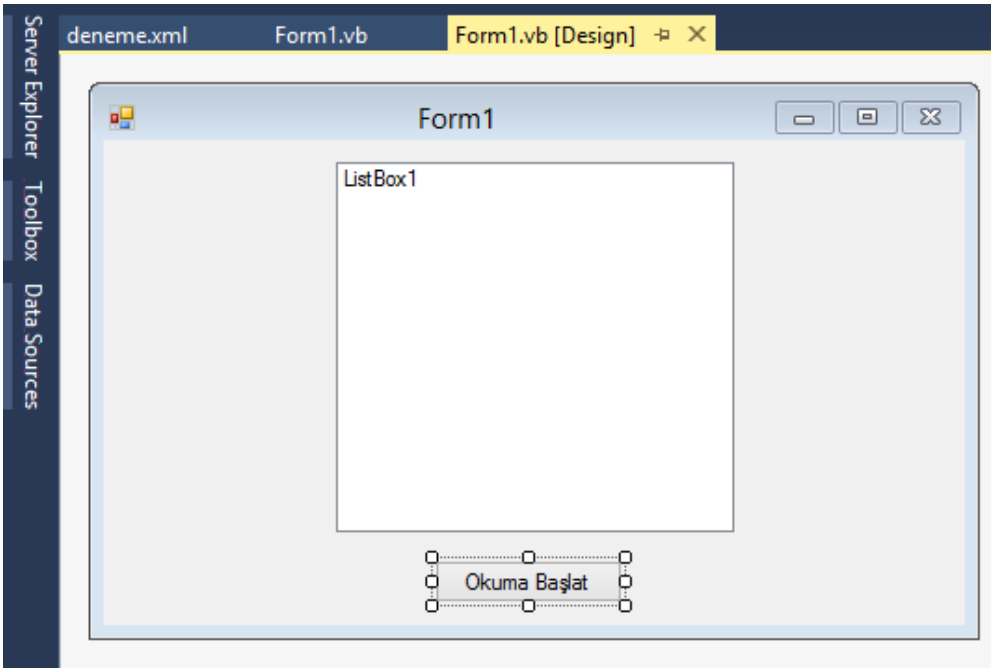
Resim 6. XML Dosyası Ekleme Penceresi

Bunun ardından XML dosyasımızı önceki örneklerdeki gibi oluşturuyoruz. Visual Studio.NET editörü XML desteği verdiği için XML ile çalışırken Notepad'a göre daha kullanıcı dostu bir arayüzle çalışmaktayız (Resim 7).



Resim 7. VS.NET'te XML Dosyası Görünümü

Xml belgesini projeye ekledikten sonra projenin bulunduğu dosya dizinleriyle uğraşmamak için C sürücüsüne yapıştırıyoruz. Sonrasında Windows Formunu aşağıdaki gibi tasarlıyoruz (Resim 8).



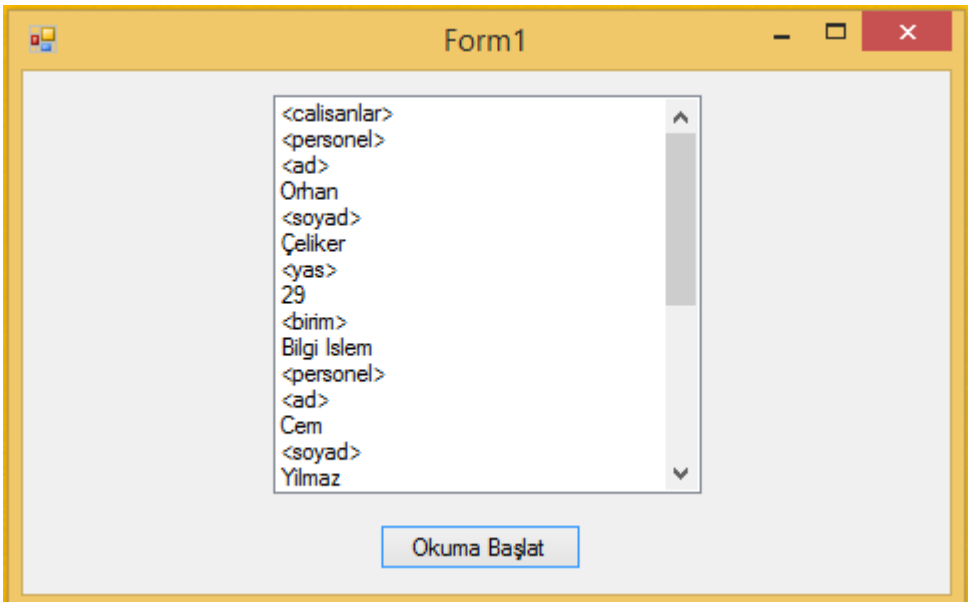
Resim 8. Form Tasarımı

Artık XML belgesini okuyacak kodu yazabiliriz. Okuma Başlat butonuna tıklandığında XML elementlerini ve bu elementlerin içinde yer alan verileri okuyup, listBox kontrolünde görüntüleyen kodlar aşağıdaki gibidir.

```
    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        Dim dosyaadi As String = "C:\deneme.xml"
        Dim okuyucu As New XmlTextReader(dosyaadi)
        While okuyucu.Read
            Select Case okuyucu.NodeType
                Case XmlNodeType.Element
               ListBox1.Items.Add("<" + okuyucu.Name + ">")
                    Exit Select
                Case XmlNodeType.Text
               ListBox1.Items.Add(okuyucu.Value)
                    Exit Select
                Case XmlNodeType.EndElement
                    Exit Select
            End Select
        End While
        End Sub
```

XmlTextWriter ve XmlTextReader sınıfları ile çalışırken System.Xml isim alanı(namespace) çağrılmalıdır.

Okuma Başlat butonuna tıklandığında programın çıktısı aşağıdaki gibidir (Resim 9).



Resim 9. Program Çıktısı

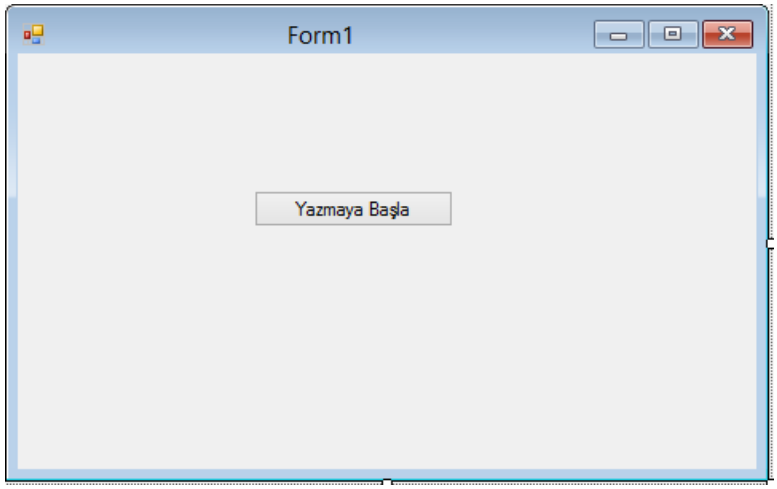
Programın çıktısı incelendiğinde listBox'a deneme.xml dosyasındaki tüm elemanların element adları ve veri içerikleri eklendiği görülmektedir. XmlTextReader sınıfının NodeType özelliğini kullanarak düğümün element ya da içerik olup olmadığına karar veriyoruz (Select Case döngüsüne dikkat ediniz). Ayrıca düğüm elemanı son eleman ise (EndElement) NodeType'da bunu belirtip istediğimiz işlemi yapma şansımız vardır.

XmlTextReader sınıfının Name özelliği düğümün etiket ismini verirken, Value özelliği ise ilgili düğümün değerini vermektedir.

XmlTextReader sınıfının Name özelliği düğümün etiket ismini verirken, Value özelliği ise ilgili düğümün değerini vermektedir. Örnekte de bu özellikler kullanılarak XML okuyucudan etiket ve veriler listBox'a eklenmiştir. Belgedeki tüm ögeler okunduktan sonra Read() metodu false değeri döndürür ve while döngüsünden çıkılır.

Eğer programlama ile XML belgesi oluşturmak istersek, XmlTextWriter sınıfını kullanabiliriz. Tıpkı XmlTextReader sınıfı gibi XmlTextWriter sınıfının da kullanımı basittir. Aşağıdaki örnekte SortedList kullanılarak bir sözlük tablosu oluşturulmuş ve bu tablo XML belgesi olarak kaydedilmiştir. Örnek boyunca XmlTextWriter sınıfının sunmuş olduğu çeşitli metot ve özellikler kullanılmıştır.

İlk olarak bir Windows Forms projesi oluşturarak Resim 10'daki gibi basit bir ekran tasarımı yapalım.



Resim 10. XML Yazdırma Ekranı

Bu işlemden sonra Yazmaya Başla butonunun Click olayına aşağıda verilen kodu yazalım.

```
  Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        Dim sozluk As New SortedList
        sozluk.Add("Computer", "Bilgisayar")
        sozluk.Add("Software", "Yazılım")
        sozluk.Add("Hardware", "Donanım")

        Dim yazici As New
XmlTextWriter("C:\Sozluk\sozluk.xml",
System.Text.Encoding.GetEncoding("windows-1254"))
        yazici.Formatting = Formatting.Indented
        yazici.WriteStartDocument()
        yazici.WriteComment("Sözlük")
        yazici.WriteStartElement("Sozluk")

        Dim i As Integer = 1
        Dim kelime As DictionaryEntry
        For Each kelime In sozluk
            yazici.WriteStartElement("kelime")
            yazici.WriteAttributeString("No",
i.ToString())
            '*/İngilizce Kelimeleri yazıyoruz  /*
            yazici.WriteStartElement("İngilizce")
            yazici.WriteString(kelime.Key.ToString())
            yazici.WriteEndElement()
            '*/Türkçe Karşılıklarını yazıyoruz  /*
            yazici.WriteStartElement("Türkçe")
            yazici.WriteString(kelime.Value.ToString())
```

```
            yazici.WriteEndElement()
            yazici.WriteEndElement()
            i += 1
        Next
        yazici.WriteEndElement()
        yazici.WriteEndDocument()
        yazici.Close()

        End Sub
```

Kodları incelediğimizde SortedList nesnesine 3 İngilizce-Türkçe kelime çiftinin eklendiğini görmekteyiz. Foreach döngüsü kullanılarak bu kelime çiftleri *WriteStartElement()* metodu ile oluşturulan Sozluk isimli kök altında *WriteStartElement()* metodu ile sozluk isimli XML dosyasına yazdırılmıştır. Burada dikkat edilmesi gereken noktalardan biri C:\ sürücüsü altında Sozluk isimli bir klasörün mutlaka oluşturulması gerektiğidir. Eğer böyle bir klasör oluşturulmaz ise XmlTextWriter nesnesi yazılacak belgenin yolunu bulamaycak ve program hata verecektir. Kodlarda dikkat çeken bir başka nokta da *Formatting=Formatting.Indented* deyimidir. Bu deyim sayesinde XML belgesi bizim rahat okuyabileceğimiz bir formatta oluşturulur. Bu deyim kullanılmadığı taktirde tek satırda yazılmış, okunması zor bir belge oluşturulur.

Kodlarda kullanılan *WriteStartDocument()* metodu XML belgesine verisyon bilgisi yazarak, belgenin giriş kısmını oluşturur. Sonrasında kullanılan *WriteComment()* metodu ise XML belgesine bir yorum satırı eklemektedir. Örnekte kullanılan for each döngüsü ile SortedList nesnesinde kelime çiftleri bir kelime etiketi oluşturacak şekilde XML belgesine yazdırılmaktadır. Her bir elemana ait yazdırma işlemi bittikten sonra *WriteEndElement()* metodu ile kelime etiketi kapatılır. Döngü içerisinde kullanılan *WriteAttributeString()* kelime isimli etikete numara vermektedir. Bunu da "No" ve niteliğin alacağı değer şeklinde yapmaktadır. Son olarak sık kullanılan bir metot olan *WriteString()* ise bir eleman düğümü için değer yazmaktadır. SortedList'te yer alan İngilizce ve Türkçe kelimeler bu metot ile yazdırılmaktadır. Program çalıştırıldığında elde edilen XML belgesi çıktısı şu şekildedir:

```
<?xml version="1.0" encoding="windows-
1254"?>
<!--Sözlük-->
<Sozluk>
  <kelime No="1">
    <İngilizce>Computer</İngilizce>
    <Türkçe>Bilgisayar</Türkçe>
  </kelime>
```

```
  <kelime No="2">
    <İngilizce>Hardware</İngilizce>
    <Türkçe>Donanım</Türkçe>
  </kelime>
  <kelime No="3">
    <İngilizce>Software</İngilizce>
    <Türkçe>Yazılım</Türkçe>
  </kelime>
   </Sozluk>
```

XmlTextReader ve XmlTextWriter sınıflarının birçok metot ve özelliği vardır. Ünitemizde sadece temel seviyede değinilen bu sınıflar ile ilgili detaylı bilgiyi MSDN kütüphanesinde bulabilirsiniz.

Özet

- XML hem insanların hem de bilgisayar sistemleri tarafından kolaylıkla okunabilecek dosyalar oluşturma ihtiyacından ötürü ortaya çıkmıştır.
- XML ve HTML birbirine benzer kavramlar olmakla birlikte ortaya çıkış ve kullanım amaçları tamamıyla farklıdır. Bu farkı en basit olarak şu şekilde anlatabiliriz. HTML, belgelerin görsel özellikleri ile ilgilenirken, XML içerikleri ile ilgilenir.
- VS.NET XML belgeleri ile çalışabilmek için çeşitli sınıflar sunulmaktadır. Bu sınıflar System.XML isim uzayı (alanı) altında yer almaktadır.
- XmlTextReader nesnesi oluşturulduktan sonra Read() metodu kullanılarak XML belgesinden okuma yapılır.
- XML streamlerini dosya sisteminde oluşturmak için kullanılır. XMLWriter sınıfından türemiştir.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. **XML; dili, sentaksı ve yapısı itibariyle aşağıdakilerden hangisine daha çok benzemektedir?**
   a) C++
   b) C#
   c) Visual Basic.Net
   d) C
   e) HTML

2. **Aşağıdakilerden hangisi yanlıştır?**
   a) XML aralarında teknolojiye yön veren firmaların bulunduğu W3C konsersiyumu tarafından geliştirilmiştir.
   b) XML bir programlama dilidir.
   c) Belirli standartları ve kuralları vardır.
   d) Visual Studio.Net tarafından desteklenir.
   e) Veri depolama için kullanılabilir.

3. XML dosyasında tek bir XML düğümünü elde etmek için …… kullanılır.
   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?
   a) XmlTextWriter
   b) XmlNode
   c) XmlDocument
   d) XmlTextReader
   e) XmlReader

4. **Türkçe karakterlerin dâhil olduğu standart aşağıdakilerden hangisidir?**
   a) ISO-8859-9
   b) ISO-8859-2
   c) ISO-8858-9
   d) ISO-8852-9
   e) ISO-8759-9

5. XmlWriter ve XmlReader sınıfların hangi isim alanı altındadır?

a) System.Data
b) System.IO
c) System.XML
d) System.Drawing
e) System.Print

6. …… XML belgesi oluşturmak ve XML belgesine veri kaydetmek için gerekli metotları ve özellikleri içinde barındırır.
Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

a) XmlNode
b) XmlReader
c) XmlTextReader
d) XmlTextWriter
e) XmlDocument

7. XmlTextReader sınıfının …… özelliğini kullanarak düğümün element ya da içerik olup olmadığına karar verilir.
Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

a) Name
b) Value
c) NodeType
d) XmlLang
e) XmlSpace

8. Aşağıdaki metotlardan hangisi XML belgesine verisyon bilgisi yazarak, belgenin giriş kısmını oluşturur?

a) Read()
b) Close()
c) WriteStartDocument()
d) WriteEndElement()
e) WriteString()

9. Her bir elemana ait yazdırma işlemi bittikten sonra …… metotu ile kelime etiketi kapatılır.
Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

a) WriteEndElement()
b) WriteStartDocument()
c) WriteString()
d) WriteComment()
e) Write()

10. …… bir XML belgesinin belleğe ağaç yapısı şeklinde yüklenmesi ve bu yapı üzerinde çeşitli işlemler yapılabilmesini sağlar.
Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

a) XMLDocument
b) XmlNode
c) XmlTextWriter
d) XmlTextReader
e) XmlWriter

Cevap Anahtarı

1.E,2.B,3.B,4.A,5.C,6.D,7.C,8.C,9.A,10.A

YARARLANILAN KAYNAKLAR VE BAŞVURULABİLECEK DİĞER 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.

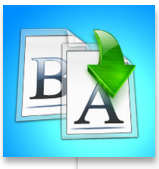
AKTAŞ, Volkan, (2013), *Her Yönüyle C# 5.0,* KODLAB, İstanbul.

DEMİRKOL, Zafer (2002), *XML*, Pusula Yayıncılık, İstanbul.

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

http://tr.wikipedia.org/wiki/XML , Erişim Tarihi:01.02.2015

İSİM UZAYI HAZIRLAMAK VE KULLANMAK



İÇİNDEKİLER

- İsim Uzayı Nedir?
- İsim Uzayı Bildirimi
- İsim Uzayı Kullanma
- İç İçe İsim Uzayı Yapısı

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - İsim uzayı kavramını öğrenebilecek,
  - İsim uzayı oluşturabilecek,
  - İsim uzayını farklı sayfalardan ve dosyalardan çağırabilecek,
  - İç içe isim uzayı bildirimi yapabileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA-II

Arş. Gör. Mehmet Cem BÖLEN

ÜNİTE

12

GİRİŞ

Yazılım projelerinin giderek büyümesi bilgisayar bilimcileri yazılan kodların organizasyonu ve alt programlar arasında erişimi ile ilgili farklı yöntemler aramaya itmiştir. Bu çabaların sonucunda ortaya çıkan yaklaşımlardan biri de isim uzayı (Namespace) adı verilen yöntemdir. Bu ünitede öncelikle isim uzayının ne olduğu ve ne işe yaradığı üzerinde durulacaktır. Ardından Visual Basic.Net ortamında bir isim uzayının nasıl tanımlanacağı ve kullanılacağı verilen kod örnekleriyle açıklanacaktır.

İSİM UZAYI NEDİR

İsim uzayının temel amacı sınıf ve yapı(structure) kullanımlarında ortaya çıkabilecek isim karmaşıklığının önüne geçmektir.

Yazılım projelerinin boyutlarının ve kapsamlarının büyümesi ile birlikte yeni programlama teknikleri ve yöntemleri ortaya çıkmıştır. Bu tekniklerden biri de isim uzayı (Namespace) mekanizmasıdır. Özellikle büyük yazılım projelerinde yüzlerce yazılımcının birarada çalışma gerekliliği düşünülünce projede kodların organizasyonunun (değişken isimlendirmeleri, sınıf yapısı *vb.*) doğru planlanması ihtiyacı üzerine ortaya çıkmıştır.

İsim uzayı, birbiri ile mantıksal ya da işlevsel olarak ilişkili olan farklı sınıfların fonksiyonların ya da metotlarının birbiriyle karışmamasını sağlayan bir kapsam alanı (scope) belirleyicidir. Bir başka ifadeyle benzer işleri yapan sınıfları bir arada toplayan bir taşıyıcı olarak da düşünülebilir.

Visual Studio.Net içerisinde buluanan tüm kütüphaneler, amaçlarına ve işlevlerine göre isim alanı yapısı kullanılarak sınıflandırılmışlardır. Örneğin Net Framework içerisinde veri tabanına erişim ile ilgili ihtiyaç duyulan bütün sınıflar System.Data isim uzayı altında yer almaktadır. Görsel işlemler (şekil, resim çizimi *vb.*) yapmak için ihtiyaç duyulan sınıflar ise System.Drawing isim uzayı altında bulunur. Dolayısıyla yapılacak işleme göre bir projede birçok farklı isim alanı ile çalışmak gerekebilir.

Visual Studio.Net'te isim uzayı (namespace)'nın kullanım mantığı Java'daki paket (package) ile hemen hemen aynıdır .

Visual Studio.Net içerisinde onlarca isim uzayı vardır. Ancak çoğu zaman .Net ortamında hazır sunulan isim uzayları dışında programcının kendi oluşturduğu isim uzaylarını da kullanması gerekebilir. Özellikle orta ve büyük ölçekli projelerde programcılar tarafından oluşturulan sınıflar, arayüzler ve modüller belirli isim uzayları çatısı altında toplanır ve gerekli görülen yerlerde çağrılarak kullanılır.

İsim uzaylarının ne olduğunu ve ne için kullanıldığını teorik olarak ne kadar uzun anlatırsak anlatalım, tam anlamıyla anlamak için kod örneklerini incelemek ve pratik yapmak gereklidir. Bu yüzden öncelikle isim uzayının bildiriminin nasıl yapıldığını inceleyelim.

İsim Uzayı Bildirimi

Oluşturduğu bir sınıfları ya da modülleri bir isim uzayı altında toplamak için öncelikle isim uzayının bildirimini (tanımını) yapmamız gerekmektedir. İsim uzayı *Namespace* anahtar kelimesi ile tanımlanır. Bu anahtar kelimeden sonra oluşturulmak istenen isim uzayının adı belirtilir. Aşağıda verilen örnekte bir isim alanı bildirimi yapılmıştır.

```
Namespace Meyve
End Namespace
```

Verilen örnekte görüldüğü üzere Namespace anahtar kelimesinden sonra bir isim verilmiş ve End Namespace ifadesiyle de namespace bloğu kapatılmıştır. İsim uzayını tanımlarken sınıf (Class), arayüz (Interface), numaralandırma (Enum) gibi yapıları içerebileceğini belirtmiştik. İsim uzayının içereceği bütün bu ögeler *Namespace-End Namepace* bloğu içerisinde tanımlanır. Aşağıdaki örnekte Meyve isimli isim uzayının içinde elma ve armut isimli iki sınıfın nasıl oluşturulduğu görülmektedir.

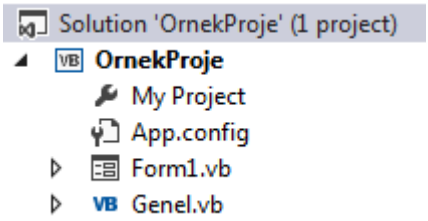
```
Namespace Meyve
  Public Class Elma
    Private _fiyat As Double
    Private _uretimyeri As String
    Property Fiyat() As Double
      Get
        Return _fiyat
      End Get
      Set(ByVal Value As Double)
        _fiyat = Value
      End Set
    End Property
    Property UretimYeri() As String
      Get
        Return _uretimyeri
      End Get
      Set(ByVal Value As String)
        _uretimyeri = Value
      End Set
    End Property

    Public Sub ElmaBilgileriGoster()
      MessageBox.Show("Elmanın Fiyatı:" & _fiyat.ToString() & " Elmanın Üretim
Yeri:" & _uretimyeri)
    End Sub
  End Class

 Public Class Armut
    Private _fiyat As Double
```

```
    Private _uretimyeri As String
    Property Fiyat() As Double
      Get
        Return _fiyat
      End Get
      Set(ByVal Value As Double)
        _fiyat = Value
      End Set
    End Property
    Property UretimYeri() As String
      Get
        Return _uretimyeri
      End Get
      Set(ByVal Value As String)
        _uretimyeri = Value
      End Set
    End Property
    Public Sub ArmutBilgileriGoster()
      MessageBox.Show("Armutun Fiyatı:" & _fiyat.ToString() & " Armutun
Üretim Yeri:" & _uretimyeri)
    End Sub

  End Class
End Namespace
```

Verilen örnekte Meyve isimli isim uzayının altına Elma ve Armut isimli iki sınıf tanımlanmıştır. Sonrasında bu sınıflara ait Fiyat ve UretimYeri isimli iki özellik ve ElmaBilgileriGoster() adında bir alt program(sub) tanımlaması yapılmıştır. Burada dikkat edilmesi gereken en önemli nokta sınıf tanımlamalarının Namespace-End Namespace kod bloğu arasına yapılmış olduğudur. Böylece Meyve isimli isim uzayı altında bu sınıflara erişim sağlanabilecektir.

İsim Uzayı Kullanımı

İsim uzayı altında tanımlanmış olan ögelere dışarıdan (Form, sınıf *vb.*) erişmek için farklı yöntemler bulunmaktadır. Örneğin basit bir Windows Forms projesi oluşturduğumuzu düşünelim. Bu projede Genel isimli bir sınıf (class) dosyasımız ve Form1 isimli bir Windows Formumuz olsun (Resim 1).



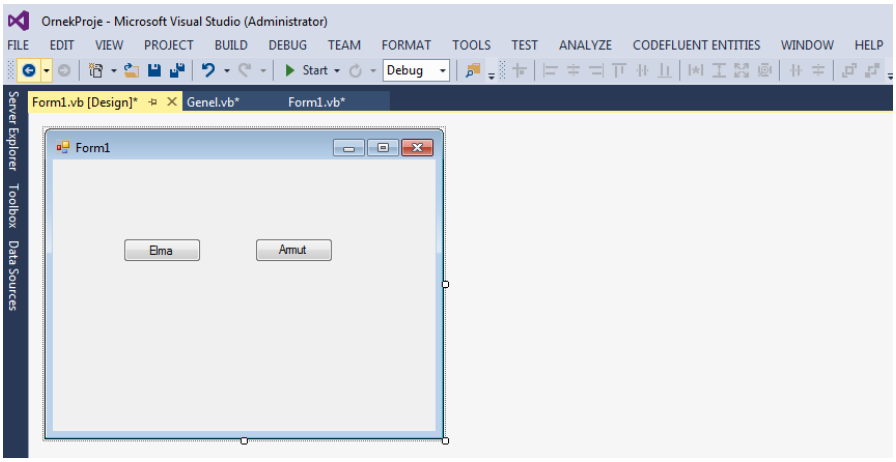
Resim 1. Örnek Proje Dosyalar

Genel isimli sınıf dosyasına isim uzayında verdiğimiz örnekteki kodların aynısı yapıştıralım. Bu işlemi yaptıktan sonra Genel isimli sınıf dosyasının görünümü aşağdaki gibi olacaktır (Resim 2).

```
Namespace Meyve
    Public Class Elma
        Private _fiyat As Double
        Private _uretimyeri As String
        Property Fiyat() As Double
            Get
                Return _fiyat
            End Get
            Set(ByVal Value As Double)
                _fiyat = Value
            End Set
        End Property
        Property UretimYeri() As String
            Get
                Return _uretimyeri
            End Get
            Set(ByVal Value As String)
                _uretimyeri = Value
            End Set
        End Property
        Public Sub ElmaBilgileriGoster()
            MessageBox.Show("Elmanın Fiyatı:" & _fiyat.ToString() & " Elmanın Üretim Yeri:" & _uretimyeri)
        End Sub
    End Class
    Public Class Armut
        Private _fiyat As Double
        Private _uretimyeri As String
        Property Fiyat() As Double
            Get
                Return _fiyat
            End Get
            Set(ByVal Value As Double)
                _fiyat = Value
            End Set
        End Property
        Property UretimYeri() As String
            Get
                Return _uretimyeri
            End Get
            Set(ByVal Value As String)
                _uretimyeri = Value
            End Set
        End Property
        Public Sub ArmutBilgileriGoster()
            MessageBox.Show("Armutun Fiyatı:" & _fiyat.ToString() & " Armutun Üretim Yeri:" & _uretimyeri)
        End Sub
    End Class
End Namespace
```

Resim 2. Genel .Vb Dosyasının Görünümü

Bu işlemleri gerçekleştirdikten sonra Form1 isimli Windows Form'a iki button sürükleyip bırakalım. Bu buttonlardan biri elma bilgilerini diğeri ise armut bilgilerini ekrana getirecektir (Resim 3).

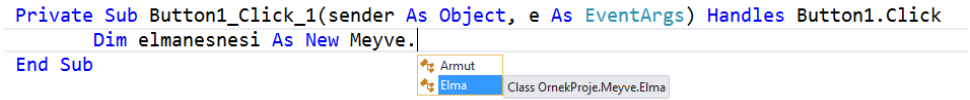


Resim 3. Form 1 Ekran Tasarımı

Öncelikle elma bilgilerini getirecek kodları yazmayı başlayalım. Meyve isim uzayı altında bulunan elma isimli sınıfta ElmaBilgileriGoster() isimli alt program ekrana bu sınıfa ait Fiyat ve UretimYeri isimli özelliklere girilen değerleri MessageBox ile gösterecektir. Elme sınıfı içerisindeki bu ögelere erişmemiz için yapmamız gereken Elma türümden bir nesne türetmektir. Elma türünden bir nesne türetmek için ise Elma sınıfına erişmemiz gereklidir. Elma sınıfı Meyve isim uzayının altında olduğu için aşağıdaki şekilde Elma türünden bir nesne türetilir.

```
Dim elmanesnesi As New Meyve.Elma
```

Burada dikkat çeken en önemli nokta, önce isim uzayının adı yazılıp nokta operatöründen sonra sınıf adının yazılmasıdır. İsim uzayı yazıldıktan sonra nokta operatöründen sonra ilgili isim uzayında dışarıya erişimi olan (Public) sınıflar listelenir. (Resim 4)



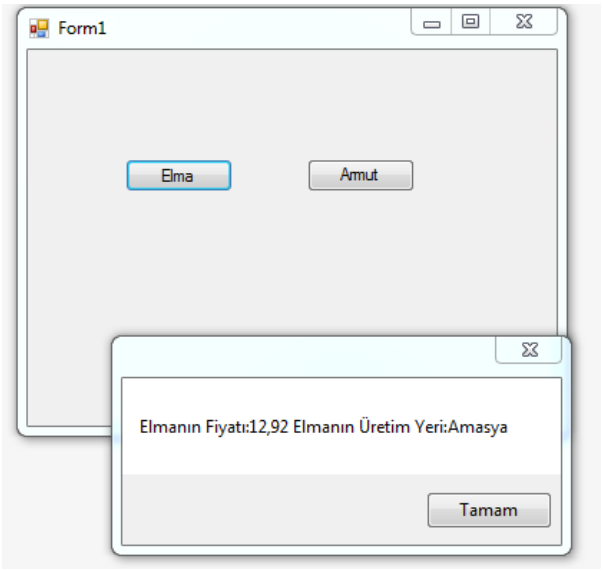
Resim 4. Meyve isim uzayı altındaki sınıflara erişim

Yukarıda görüldüğü gibi Meyve yazılıp nokta denildiğinde Armut ve Elma isimli bu isim uzayı altından tanımlanan Public sınıflar listelenir. Elma sınıfından elmanesnesi isimli nesne türetildiğinde bu sınıf içerisinde tanımlanan public sınıf ve özelliklere erişildiği görülmektedir.

```
Dim elmanesnesi As New Meyve.Elma
    elmanesnesi.Fiyat = 12.92
    elmanesnesi.UretimYeri = "Amasya"
    elmanesnesi.ElmaBilgileriGoster()
```

Yukarıdaki kodlar çalıştırıldığında Fiyat ve UretimYeri özelliklerine değer atanacak ve ElmaBilgileriGoster() metodu ile de kullanıcıya Fiyat ve UretimYeri özelliklerine girilen değerler kullanıcıya gösterilecektir. (Resim 5)

İstediğiniz kadar aynı isimli isim uzay oluşturabilirsiniz.



Resim 5. Elma butonuna basıldığında görüntülenen mesaj

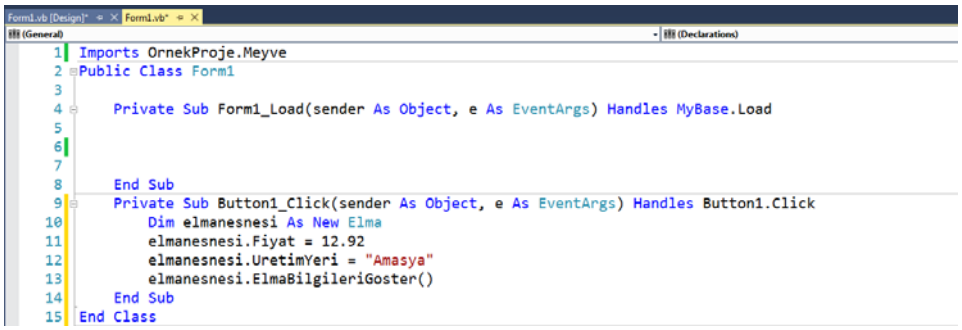
Bir isim alanı içerisine birden fazla isim alanı tanımlanabilir. İç içe isim uzaylarının kullanılması özellikle kod organizasyonunda büyük esneklik sağlar. Aşağıda verilen örnekte içe içe isim uzayı kullanımı görülmektedir.

```
Namespace Yiyecek
  Namespace Meyve
    Namespace YazlikMeyve

    End Namespace

    Namespace KislikMeyve

    End Namespace
  End Namespace

  Namespace Sebze
    Namespace YazlikSebze

    End Namespace
    Namespace KislikSebze

    End Namespace
  End Namespace
  Namespace Tatli
    Namespace SerbetliTatli

    End Namespace
    Namespace SerbetsizTatli

    End Namespace
```

```
    End Namespace
End Namespace
```

Visual Basic.Net içerisinde temel veri tiplerini kullanmak için System isim uzayını çağırmaya gerek yoktur.

.*Net Framework içerisinde kullanmış olduğumuz sınıf ve metotlar isim uzayları içine tanımlanmıştır*. Bazı sınıflar tek bir isim uzayı altına tanımlanmışken (System gibi) bazıları iç içe isim uzayları arasına tanımlanmıştır. (System.Data.SqlClient gibi). Özellikle iç içe isim uzaylarına tanımlanan sınıfları çağırırken her seferinde isim uzayının adını yazmak hem gereksiz hem de kodun okunabilirliğini azaltan bir durumdur. Bu durumu önlemek için *Imports* anahtar kelimesi kullanılır. *Imports anahtar kelimesi ile çağrılan isim uzaylarının ögelerine proje içerisinde doğrudan erişilebilir*. Aşağıda verilen örnekte önceden oluşturduğumuz Meyve isim uzayının Imports anahtar kelimesi kullanılarak projeye nasıl eklendiği görülmektedir (Resim 6).



```
Imports OrnekProje.Meyve
Public Class Form1

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



    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim elmanesnesi As New Elma
        elmanesnesi.Fiyat = 12.92
        elmanesnesi.UretimYeri = "Amasya"
        elmanesnesi.ElmaBilgileriGoster()
    End Sub
End Class
```

Resim 6. Form1'in Code Behind görünümü

Resim 6'da görüldüğü üzere programın en başına Imports anahat kelimesi ile kullanılacak isim uzayı dâhil edilmiştir. OrnekProje adlı proje içerisinde bulunan Meyve isim uzayı bir kere çağrıldıktan sonra bu isim uzayı altında yer alan Elma sınıfına erişmek için bir daha Meyve yazılmasına gerek kalmamıştır. Kısacası sadece kullanılacak sınıfın isminin yazılması yeterli olmuştur.

İsim uzayı tanımlanırken aşağıdaki kurallara dikkat edilmesi hem kodun okunabilirliğini artıracak hem de umulmadık problemlerin ortaya çıkmasını engelleyecektir.

1) İsim uzayı adı fazla uzun olmamalıdır.
2) İsim uzayı adında Türkçe karakter kullanılmamaya çalışılmalıdır.
3) İsim uzayı adı anlaşılabilir olmalı ve içerdiği öğelerin işlevlerine uyumlu olmalıdır.
4) İsim uzayında yalnızca sınıf (class), temsilci (delegate), numaralandırma (enum), arayüz (interface) ya da yapı (structure) bildirimi yapılabilir.

Verilen bu kurallara uymak özelliklere büyük yazılım projelerinde büyük kolaylıklar sağlayacaktır.

Aşağıdaki tabloda Visual Studio.Net içerisinde en çok kullanılan isim uzayları hakkında bir özet sunılmuştur.

Tablo 1. İsim Uzayları ve Genel Bilgiler

İsim Uzayı(Namespace)		Genel Bilgi
System	→	Referans edilen veri türlerini ve ortak kullanıma sahip değerleri içinde bulundurur.
System.Data	→	Veritabanı ile ilgili birçok işlem bu isim uzayı altında yer alan sınıflar vasıtasıyla gerçekleştilir.
System.IO	→	Metin ve dosya okuma/yazma işlemleri için kullanılan sınıfları barındırır.
System.Web	→	Web projelerinde ihtiyaç duyulan sınıfları barındırır.
System.Security	→	.Net Framework içerisinde güvenlik ilgili (izinler *vb.*) ihtiyaç duyulan sınıfları bulundurur.
System. Globalization	→	Dil, para birimi, takvim gibi ülkelere/yörelere özel bilgileri sağlamada ya da çevirmede kullanılan sınıfları içerisinde bulundurur.
System.Windows.Forms	→	Windows tabanlı uygulamaların çalışmak için ihtiyaç duyduğu sınıfları barındırır.

Bireysel Etkinlik

- Visual Basic.Net ortamında Canlılar isimli bir isim uzayı oluşturunuz. Bu isim uzayında insan, bitki ve hayvan isimli 3 farklı sınıf oluşturarak bu sınıflardan Windows Forms'ta 3'er farklı nesne türetiniz.
- Not:İnsan,bitki ve hayvan sınıflarının temel özellikleri sizin hayal gücünüz ile sınırlıdır.

Özet

- Yazılım projelerinin boyutlarının ve kapsamlarının büyümesi ile birlikte yeni programlama teknikleri ve yöntemleri ortaya çıkmıştır. Bu tekniklerden biri de isim uzayı (Namespace) mekanizmasıdır.
- Visual Studio.Net içerisinde buluanan tüm kütüphaneler, amaçlarına ve işlevlerine göre isim uzayı yapısı kullanılarak sınıflandırılmışlardır
- . İsim uzayı *Namespace* anahtar kelimesi ile tanımlanır.
- İsim uzayı adı fazla uzun olmamalıdır.
- İsim uzayı adı anlaşılabilir olmalı ve içerdiği öğelerin işlevlerine uyumlu olmalıdır.
- İsim uzayında yalnızca sınıf (class), temsilci (delegate), numaralandırma (enum), arayüz (interface) ya da yapı (structure) bildirimi yapılabilir.
- Imports anahtar kelimesi ile çağrılan isim uzaylarının ögelerine proje içerisinde doğrudan erişilebilir.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "Bölüm Sonu Testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. Visual Basic.Net içerisinde isim uzayı aşağıdaki anahtar kelimelerden hangisi ile tanımlanır?
   a) Imports
   b) New
   c) Dim
   d) Namespace
   e) Create

2. Aşağıdakilerden hangisi isim uzayı içerisindem tanımlanamaz?
   a) Sınıf
   b) Arayüz
   c) Numaralandırma
   d) Metot
   e) Yapı

3. İsim uzaylarına proje içerisinde ……. anahtar kelimesi kullanılarak doğrudan erişilir.
   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) Namespace
   b) Imports
   c) Public
   d) New
   e) Create

4. Temel tür yapıları hangi uzayın altında bulunur?
   a) System
   b) System.Printing
   c) System.Data
   d) System.IO
   e) System.Security

5. Visual Studio.Net içerisinde buluanan tüm kütüphaneler amaçlarına ve işlevlerine göre ……. yapısı kullanılarak sınıflandırılmışlardır.
   Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) Metot
   b) Sınıf
   c) İsim uzayı
   d) Numaralandırma
   e) Değişken

6. Aşağıdakilerden hangisi yanlıştır?
   a) İsim uzayı kodların organizasyonunu sağlar.
   b) System.Data Visual Basic.Net içerisinde bulunan bir isim uzayıdır.
   c) .Net içerisindeki isim uzayı yapısı Java'daki paket(Package) yapısı ile benzerdir.
   d) Bir isim uzayı içerisinde başka bir isim uzayı tanımlanamaz.
   e) İsim uzayı içerisinde sınıf tanımlanabilir.

7. Visual Basic.Net iç içe tanımlanan isim uzayları …… operatörü ile …… anahtar sözcüğü kullanılarak Windows Forms'dan çağrılabilir.

   Cümlede boş bırakılan yere sırasıyla aşağıdakilerden hangisi getirilmelidir?
   a) ! - Imports
   b) . - Get
   c) * - Namespace
   d) . - Namespace
   e) . - Imports

8. Aşağıdakilerden hangisi isim uzayı tanımlarken dikkat edilecek kurallar arasında yer almaz?
   a) İsim uzayı adı büyük harfle başlamalıdır.
   b) İsim uzayı adı anlaşılabilir olmalı ve içerdiği ögelerin işlevlerine uyumlu olmalıdır
   c) İsim uzayı adı fazla uzun olmamalıdır.
   d) İsim uzayı adında Türkçe karakter kullanılmamasına özen gösterilmelidir.
   e) İsim uzayında yalnızca sınıf (class), temsilci (delegate), numaralandırma (enum), arayüz (interface) ya da yapı (structure) bildirimi yapılmasına dikkat edilmelidir.

9. Veritabanı ile ilgili işlemleri gerçekleştirmek için hangi isim uzayını çağırmamız gerekmektedir?
   a) System.IO
   b) System**.** Globalization
   c) System.Data
   d) System.Security
   e) System.Web

10. .Net Framework içerisinde güvenlik ilgili (izinler *vb.*) ihtiyaç duyulan sınıfları bulundurur.

    Yukarıda özelliği verilen isim uzayı hangisidir?

    a) System.Security
    b) System.Net
    c) System.Data
    d) System.IO
    e) System.Web

Cevap Anahtarı

1.D, 2.D, 3.B, 4.A, 5.C, 6.D, 7.E, 8.A, 9.C, 10.A

YARARLANILAN KAYNAKLAR VE BAŞVURULABİLECEK DİĞER 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.

Clark, D., & Sanders, J, (2011). Beginning C# object-oriented programming, Apress.

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

Grundgeiger, Dave. (2002). Programming Visual Basic. NET. O'Reilly, California.

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

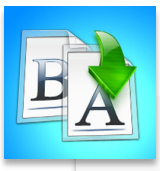
Schneider, D. I. (2013). An Introduction to Programming Using Visual Basic 2012. Prentice Hall Press,New Jersey.

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.

Taşdelen, Aykut, (2010). C# ile Veritabanı Programlama ve Ado.Net, Pusula Yayıncılık, İstanbul.

KURULUM DOSYASI OLUŞTURMA



İÇİNDEKİLER

- Giriş
- Kurulum Dosyası Oluştumak
  - File System Editor
  - Registry Editor
  - User İnterface Editor
  - Custom Actions Editor
  - Launch Condions Editor
  - File Types Editor
- Kurulumun Test Edilmesi

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Kurulum dosyasını açabilecek,
  - Kurulum dosyasında gerekli olan düzenleyicileri kullanabilecek,
  - Kurulum dosyanıza Framework ekleyebilecek,
  - Kurulumunuzu test edebilecek,
  - Hazırladığınız projeyi dağıtabileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AÖF

GÖRSEL PROGRAMLAMA II

Okt. Daha ORHAN

ÜNİTE

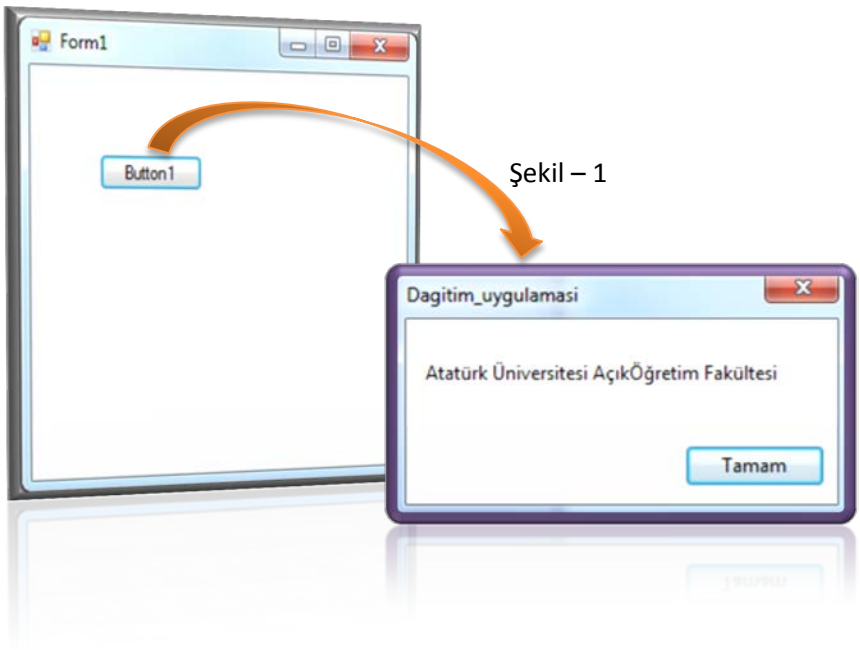
13

GİRİŞ

Bu ünitede Visual Studio 2010 üzerinden kurulum dosyası hazırlama, derleme ve dağıtma işlemlerinden bahsedeceğiz. 2010 sürümünün seçilmesinin nedeni daha alt versiyonları kullananların anlayabilmesini kolaylaştırmak. 2013 versiyonu üzerinden anlatılmamasının sebebi ise Windows installer metodunu kullanırken ek kurulum dosyasına ihtiyaç duyulmaktadır. Bunun için de http://learn.flexerasoftware.com/content/IS-EVAL-InstallShield-Limited-Edition-Visual-Studio adresinden kayıt yapılarak kurulum için gerekli dosyayı yüklememiz gerekiyor. Bu işlemi yaptıktan sonra bahsedeceğimiz adımlar birbiriyle aynı özelliğe sahiptir.

Hazırladığımız projeleri yayımlamak için ClickOnce ve Windows Installer metotlarından herhangi birini kullanabiliriz. Bu ünite içinde Windows Installer metodu kullanılacaktır.

Kurulum dosyası hazırlama başlığına geçmeden önce dağıtımını gerçekleştireceğimiz basit bir uygulama hazırlayalım. Uygulamamız, bir Windows form üzerine yerleştirilen buton yardımıyla "Atatürk Üniversitesi AçıkÖğretim Fakültesi" mesajını verdirecek şekilde ayarlanmış olsun. Programın çalışması esnasındaki olaylar Şekil 1'de gösterilmiştir.



Şekil – 1

Kurulum dosyası hazırlanmadan önce programın en az bir kere çalıştırılmış olması gerekmektedir.

Bunu yapmamızın sebebi, kurulum dosyası hazırlanmaya başlamadan önce programın hazırlanıp en az bir kere çalıştırılmış olması gerekir. Çünkü derlenen dosyaları kullanacağımız alanlar bulunmaktadır.

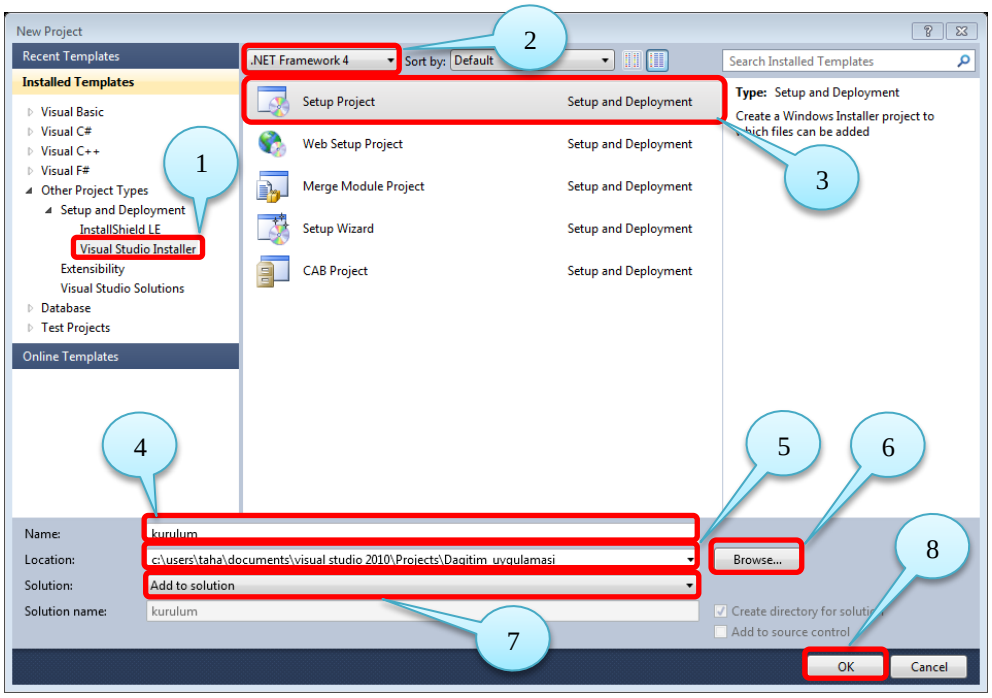
Artık Dagitim_uygulamasi olarak hazırladığımız programın kurulum dosyasını oluşturmak için yapacağımız adımlara geçebiliriz.

KURULUM DOSYASI OLUŞTURMAK

Hazırladığımız programların Visual Studio yüklü olmayan başka bilgisayarlarda çalışabilmesi için kurulum dosyasının oluşturulması gerekmektedir.

Hazırladığımız programın Visual Studio yüklü olmayan başka bilgisayarlarda da çalışabilmesi için kurulum (setup veya installation) dosyasının oluşturulması gerekmektedir. Bu işleme deployment adı verilmektedir. Yani diğer bilgisayarlar üzerinde ana programın bir kopyasını çalıştırma işlemidir.

Şimdi Dagitim_uygulamasi adı ile hazırladığımız projemizi açalım. Burada menü yolu olarak File → New → Project veya kısayol tuşu olarak Ctrl + Shift + N adımlarından herhangi birini uygulayalım. Karşımıza Şekil 2'de gösterildiği gibi New Project penceresi açılacaktır.



Şekil 2

Bu pencere üzerinde dikkat etmemiz gereken hususlar kırmızı kutu içine alınarak sırasıyla numaralandırılmıştır. Bu pencerede ilk olarak sol taraftaki Installed Templates alanından sırasıyla Other Project Types, Setup and Deployment ve Visual Studio Installer seçeneğini 1 numaralı adımda gösterildiği gibi seçmemiz gerekmektedir.

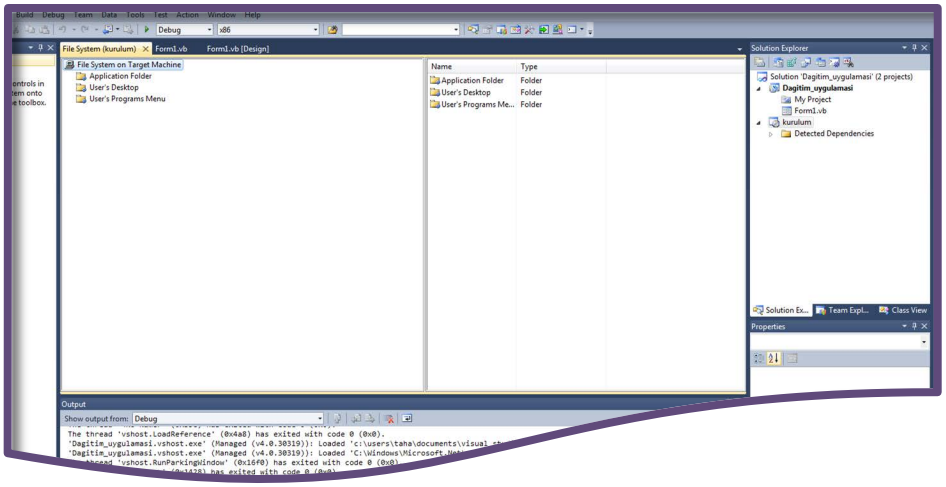
2 numaralı alan üzerinde değişiklik yapmamıza gerek yoktur. Fakat şunu belirtmekte fayda görüyorum, kurulum dosyasının yükleneceği bilgisayarda "Framework" kütüphanesinin bulunması gerekmektedir. Bilgisayarlara işletim sisteminin özelliğine bağlı olarak Framework kütüphaneleri yüklenmektedir. Tabi sürüm farklılıkları olabilir. İşte burada hazırladığımız Framework sürümü ile kurulum yapılacak olan bilgisayardaki framework sürümleri birbirlerini desteklemek zorundadır. Aksi takdirde projemizin çalışması esnasında problem yaşayabilir. Bu kütüphaneleri ekleyeceğimiz alanlardan ünitenin ilerleyen kısımlarında bahsedilecektir.

Visual Studio Installer seçeneğini işaretledikten sonra karşımıza gelen seçeneklerden 3 numaralı alanda belirtilen Setup Project seçeneğini işaretleyerek

işlemlerimize devam ediyoruz. 4 numaralı alanda kurulum yapacağımız .exe uzantılı dosyamızın adını belirtiyoruz. 5 numaralı alanda ise kurulum dosyamızın bulunduğu yol gösterilmektedir. Bu yolu değiştirmek için 6 numaralı alanda gösterilen “Browse” butonuna tıklayarak konum bilgisi güncellenebilir.

New Project penceresini açtığımızda 7 numaralı alanda varsayılan seçenek olarak “Create New Solution” bulunmaktadır. Biz bunu “Add to Solution” olarak değiştiriyoruz. Son olarak 8 numaralı alanda gösterilen ok butonuna tıklıyoruz.

Bu işlemleri gerçekleştirdikten sonra Visual Studio platformu Şekil 3’teki gibi karşımıza gelmektedir.



Şekil 3

Kurulumun yapılacağı bilgisayarda programın nasıl çalışacağı ile ilgili seçenekler file system editorden gerçekleştirilmektedir.

Şayet bu alan boş olarak gelirse sağ tarafta bulunan Solution Explorer panelindeki kurulum dosyasının ismi üzerine fare ile geldikten sonra sağ tuşa basılarak açılan menüden sırasıyla View ve File System seçenekleri tıklanır. Şimdi buradaki seçenekleri daha yakından inceleyelim.

File System Editor

Hazırlanan uygulamanın kurulacak olan bilgisayar üzerinde nasıl çalışacağı ile alakalı seçeneklerin bulunduğu alandır. Program kullanıcı bilgisayarında nereye kurulacak, masaüstünde veya göre çubuğunda simgesi olacak mı gibi ayarlar buradan gerçekleştirilir.

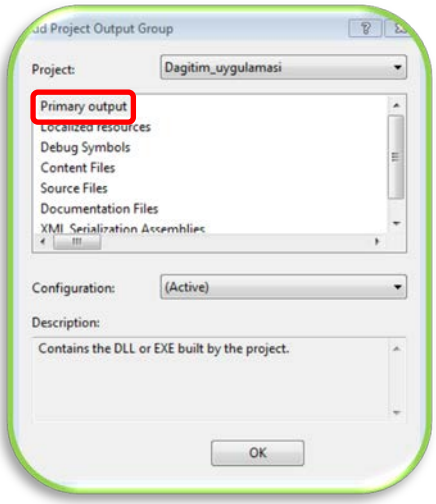
File system editör üzerinde üç adet klasör otomatik olarak karşımızda görüntülenmektedir. Bunlar:

- Application Folder
- User’s Desktop
- User’s Programs Menü klasörleridir.

Bunların haricindeki klasörleri eklemek için bu alan üzerinde boş bir yere farenin sağ tuşu ile tıklarız. Karşımıza Add Special Folder menüsü altında seçenekler sunulacaktır. Bunlardan bize uygun olanlarını ekleme imkânına sahibiz.

Application Folder: bu klasör sayesinde projemizde kullandığımız referansları, harici dosyaları, resim ve ses dosyaları gibi işlemlerin yapılmasını sağlayabiliriz. Projemizde kullanacağımız birkaç seçeneğin nasıl eklenebileceğini gösterelim.

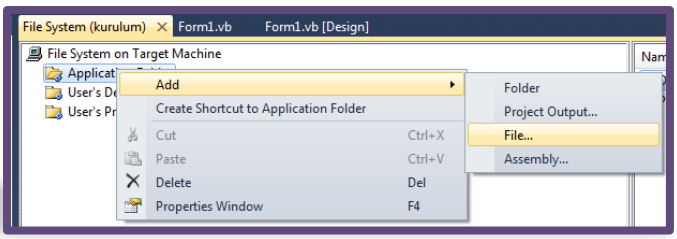
Application Folder klasörü üzerinde farenin sağ tuşuna basarak açılan menüden sırasıyla Add ve Project Output seçeneklerini işaretleyelim. Karşımıza Şekil 4'te gösterilen pencere gelecektir.



Şekil 4

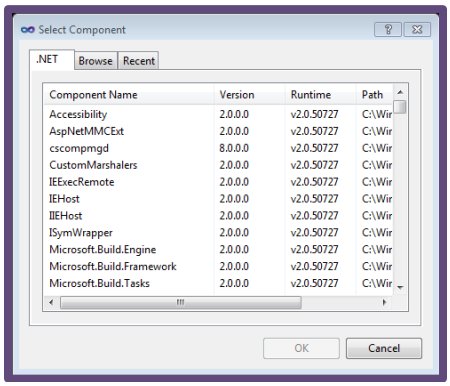
Programımızın .exe uzantılı dosyasını Application folder klasörüne eklememiz gerekmektedir.

Bu alan üzerinde kırmızı kutucuk ile gösterilen seçeneği işaretleyip "OK" butonuna tıklayalım. Primary Output seçeneğini ekledikten sonra programımızın .exe uzantılı dosyasını ve programın simgesi için kullanacağımız .ico uzantılı dosyayı buraya ekleyelim. Programımızın .exe uzantılı dosyasını eklemek için Application Folder üzerinde farenin sol tuşuna basılarak, Şekil 5'te gösterildiği gibi açılan pencereden sırasıyla Add ve File… komutlarını veririz.



Şekil 5

Karşımıza gelecek olan pencereden, bilgisayarımızda hazırladığımız projenin yolu içindeki Debug klasöründe bulunan Dagitim_uygulamasi.exe uzantılı dosyayı işaretleyip aç butonuna tıklarız. Böylelikle proje içinde kullandığımız referanslarda otomatik olarak eklenecektir. Şayet bunların dışında bir referans varsa bu defa Şekil 5'te gösterilen Add komutundan sonra File… komutu yerine Assembly… komutunu veririz.



Şekil 6

User's desktop masaüstünde programın kısayolunun oluşturulmasını sağlayan ayarları bize sunar.

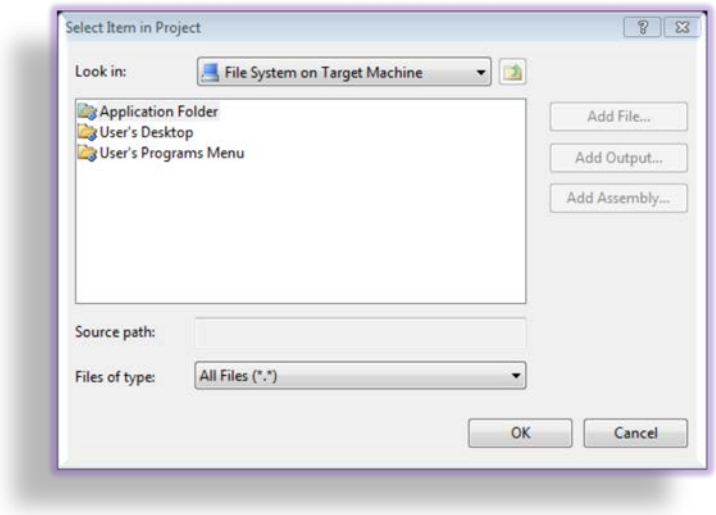
Şekil 6'da gösterildiği gibi karşımıza gelen Select Component penceresinden istediğimiz seçenekleri işaretleyip "OK" butonuna tıklayarak işlemimizi tamamlamış oluruz.

Programımızın görselliğini artırmak amacıyla kullanacağımız, bilgisayarımıza daha önceden yüklediğimiz veya oluşturduğumuz .ico uzantılı resimleri Şekil 5'te gösterilen Add ve File… komutlarını sırasıyla vererek açılan pencereden yolunu belirterek seçebiliriz.

User's Desktop: Bize birçok imkân sunan kurulum ortamı ile bu klasör sayesinde kullanıcının masaüstünde bir klasör oluşturabiliyoruz.

Not: Bu ayarlamayı yapmak zorunda değiliz. Tamamen kişisel tercihe bağlı bir seçenektir.

Ayrıca User's Desktop sayesinde kullanıcını masaüstünde bir kısayol oluşturma imkânına da sahibiz. Bu işlemi gerçekleştirebilmek için User's Desktop seçeneğini işaretledikten sonra sağ taraftaki boş alan üzerinde farenin sağ tuşuna tıklanır. Açılan menüden Creat New Shortcut komutu verilerek Şekil 7'de gösterilen Select Item in Project penceresi karşımıza gelir.



Şekil 7

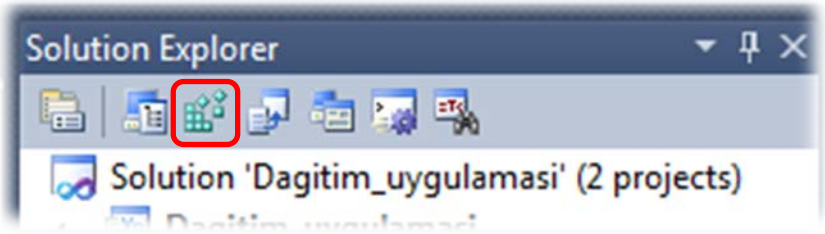
Bu pencereden de daha önce Application Folder içine eklemiş olduğumuz Dagitim_uygulamasi.exe uzantılı dosyayı gösterip, “OK” butonuna tıklarız. Programımızın görselliğini artırmak amacıyla Application Folder içine eklediğimiz .ico uzantılı dosyayı masaüstüne koyduğumuz bu kısayol için simge olarak belirleyebiliriz. Bu işlemi de eklediğimiz kısayolu seçtikten sonra sağ alt köşede bulunan properties panelindeki Icon seçeneğine Browse komutunu vererek gerçekleştirebiliriz.

User’s Programs Menu: bu klasörde kullanıcının bilgisayarında bulunan programlar menüsü içine kısayol eklemek ve simgesini belirlemek için kullanılır. User’s Desktop klasöründe gerçekleştirdiğimiz işlemlerin aynısı bunun içinde uygulamamız mümkündür. Yine belirtmek gerekirse bu ayarlama da zorunlu değil ve tamamen isteğe bağlıdır.
File System Editör altındaki bu üç klasörün özelliklerinden bahsettikten sonra Solution Explorer panelindeki “kurulum” adlı setup dosyamızı işaretleyerek bununla alakalı birçok seçeneği properties panelinde bulabiliriz.

Registry Editor

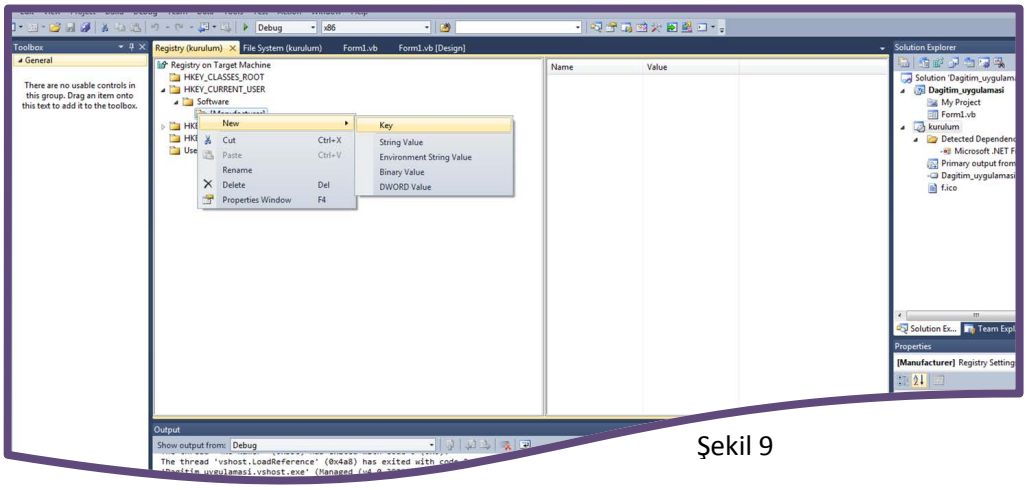
Registry editörü açarak programımızla ilgili yapacağımız kayıt düzenlemelerini gerçekleştirebiliriz.

Kurulum sırasında kayıt defteri editörü yardımıyla kullanıcı bilgisayarının kayıt defterine yönelik girdilerin belirlenmesini sağlayan bir alandır. Registry editörünü açmak için Solution Explorer penceresindeki ilgili butona tıklamamız gerekmektedir. Şekil 8’de kırmızı kutu içinde gösterilmiştir.



Şekil 8

Bu alana tıkladığımız zaman Visual Studio ekran görüntüsü Şekil 9’da gösterilmiştir. Buradaki düğümlerden herhangi birine girdi eklemek için girdi eklenilmek istenen klasörün üzerinde farenin sağ tuşuna basılır ve yine Şekil 9’da gösterildiği gibi açılan menüden sırasıyla New ve Key komutları verilir.



Şekil 9

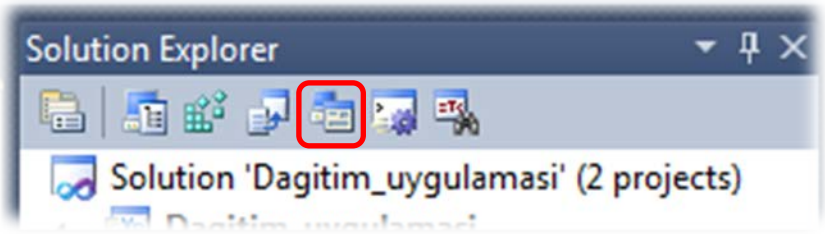
Eklenen bu girdi için yeni bir değer oluşturmak istiyorsak: Registry Editörünün sağ tarafında bulunan boş alanda farenin sağ tuşuna tıklar ve açılan menüden sırasıyla New ve String Value komutlarını veririz. Gelen değerin adını değiştirmek için fare ile üzerine gelinerek sağ tuş tıklanır ve ardından Rename komutu verilir. Diğer alakalı seçenekler properties panelinde bulunmaktadır.

Not: Yukarıda girdinin içinde string bir değer oluşturduk. Bu değer isteğe bağlı olarak açılan menü içinde farklı bir tercihte de olabilir.

User Interface Editor

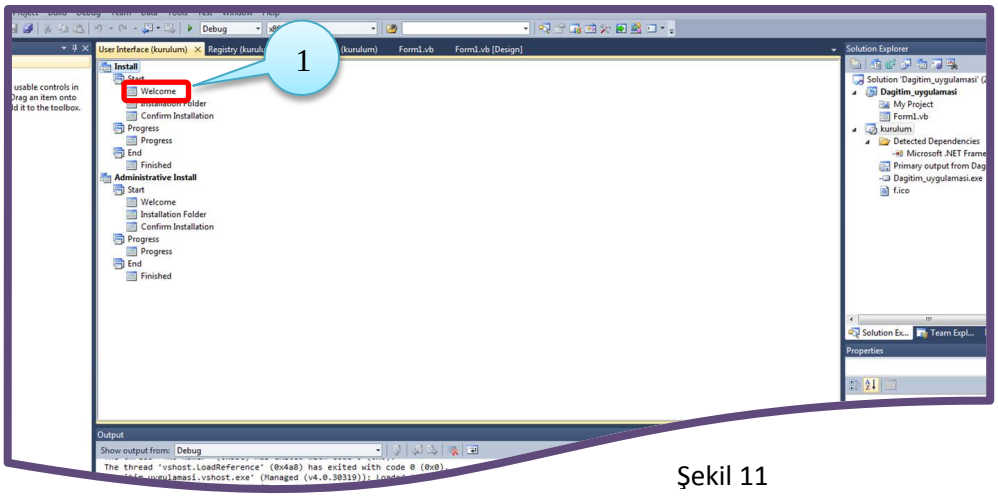
Kullanıcı bilgisayarı üzerindeyken, kurulum esnasında kurulum sihirbazı ile ilgili düzenlemelerin yapıldığı editördür. User interface editorünü açmak için Solution Explorer penceresindeki ilgili butona tıklamamız gerekmektedir. Şekil 10'da kırmızı kutu içinde gösterilmiştir.

User interface editörü kurulum sihirbazının özelleştirilmesini sağlar.



Şekil 10

Bu alana tıkladığımız zaman Visual Studio ekran görüntüsü Şekil 11'de gösterilmiştir. Bu alan üzerinde kurulum esnasında kullanıcıya yardımcı olacak bilgiler yer almaktadır.



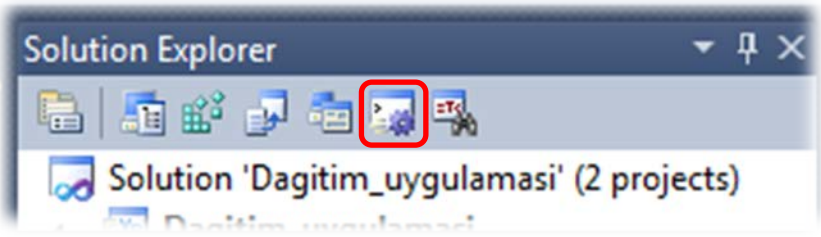
Şekil 11

Burada kırmızı kutu ile gösterilmiş alanda 1 numaralı seçeneği işaretledikten sonra properties panelinden Welcome Text özelliğine istediğimiz bir metin girebiliriz. Örneğin "Kurulum işlemini başlattınız." Şeklinde bir mesaj yazabiliriz. Buna benzer olarak CopyrightWarning özelliğine istediğimiz bir telif hakkıyla ilgili mesaj

verdirebiliriz. Kendi içinde iki ana başlığa ayrılmıştır. Bunlar Install ve Administrative Install seçenekleridir. Bu seçenekler de kendi içlerinde alt başlıklardan oluşmaktadır. Hazır olarak gelen Welcome Text diyalog kutusundan ve özelliklerinin nasıl değişebileceğinde bahsettik. Aynı işlemleri diğer diyalog kutuları içinde gerçekleştirebilirsiniz. Var olan diyalog kutularının haricinde yeni bir diyalog kutusu eklemek mümkündür. Start alt başlığı için örneklendirelim. Fare imleci ile start alt başlığının üzerine geldikten sonra sağ tuş tıklanır ve açılan menüden Add Dialog komutu verilir. Akabinde Add Dialog penceresinden istenilen bir seçim yapılır ve "OK" butonuna tıklayarak yeni diyalog kutumuz eklenmiş olur. Bu pencerede kurulumlarda sıklıkla karşılaştığımız seçim kutuları, tercih kutuları bulunmaktadır. Hazırlayacağımız projeye istinaden bir veya birden fazla diyalog kutusu ekleyebiliriz.

Custom Actions Editor

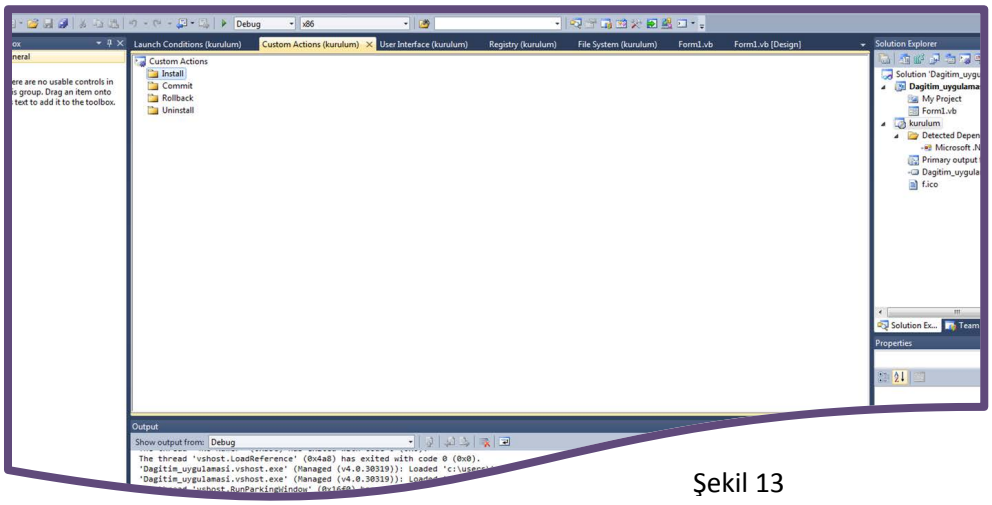
Kullanıcı bilgisayarında kurulum esnasında ek eylemlerin belirlenmesini ve ek dosyaların çağrılmasını sağlayan bir alandır. Custom actions editorünü açmak için Solution Explorer penceresindeki ilgili butona tıklamamız gerekmektedir. Şekil 12'de kırmızı kutu içinde gösterilmiştir.



Şekil 12

Custom editör install, commit, rollback ve uninstal olmak üzere dört bölümden oluşur.

Bu alana tıkladığımız zaman Visual Studio ekran görüntüsü Şekil 13'te gösterilmiştir. Bu alan dört bölümden oluşmaktadır. Bunlar: kurulum, işle, geri al ve kaldır seçenekleridir. Kurulum esnasında gerçekleştirilen veya kurulumun hatasız veya kurulum esnasında karşılaşılan güçlükler veya uygulamanın kaldırıldıktan sonraki işlemlerini kapsayan özel eylem seçenekleridir.

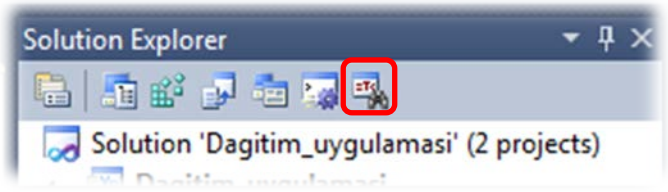


Şekil 13

Bir özel eylem eklemek için, eklenilmek istenen bölüm seçilir ve farenin sağ tuşu ile açılan menüden Add Custom Action… komutu verilir. Konumu belirtilerek Add File butonuna tıklanır ve kurulum programına eklenilmek istenen dosya belirtilir.

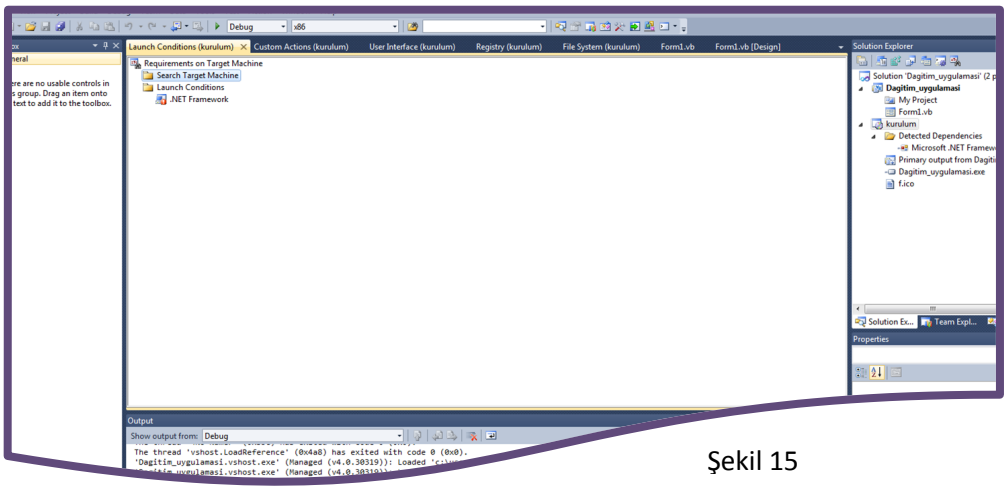
Launch Conditions Editor

Bu alan hazırladığımız Dagitim_uygulamasi adlı programın kurulacağı bilgisayardaki ayarlamaları yapmamızı ve özel olarak istenen bir gereklilik varsa onu belirtmemizi sağlar. Launch conditions editorünü açmak için Solution Explorer penceresindeki ilgili butona tıklamamız gerekmektedir. Şekil 14'te kırmızı kutu içinde gösterilmiştir.



Şekil 14

Bu alana tıkladığımız zaman Visual Studio ekran görüntüsü Şekil 15'te gösterilmiştir. Bu alan iki bölümden oluşmaktadır. Bunlar Search Target Machine ve Launch Conditions seçenekleridir.

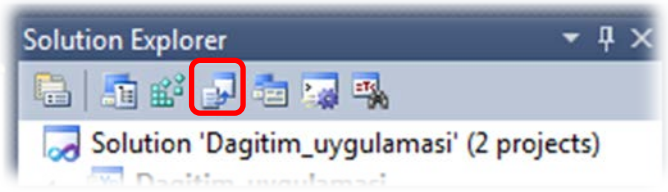


Şekil 15

File types editör üzerinde yeni eklenen eylemin özellikleri properties panelinden ayarlanabilir.

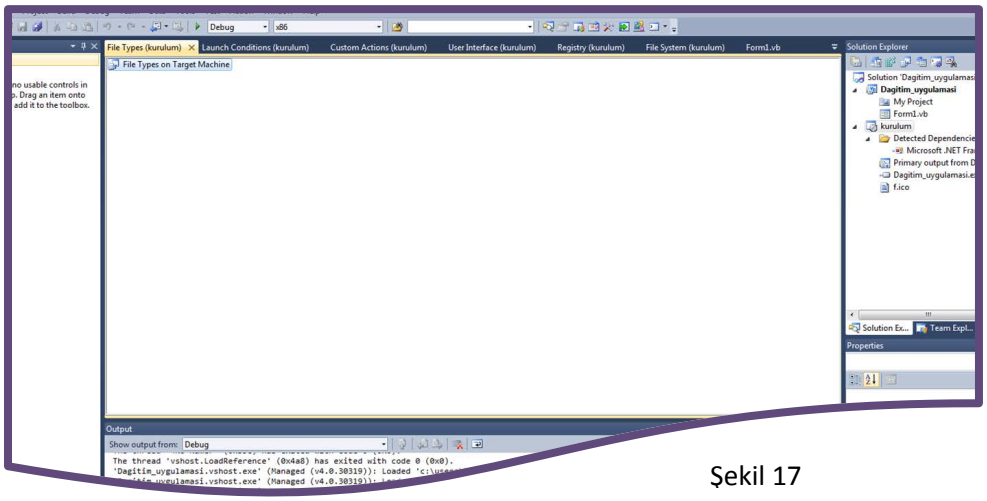
File Types Editor

Farklı uzantılardaki dosyaların birtakım uygulama veya eylemlerle ilişkilendirilmesini sağlayan alandır. File types editorünü açmak için Solution Explorer penceresindeki ilgili butona tıklamamız gerekmektedir. Şekil 16'da kırmızı kutu içinde gösterilmiştir.



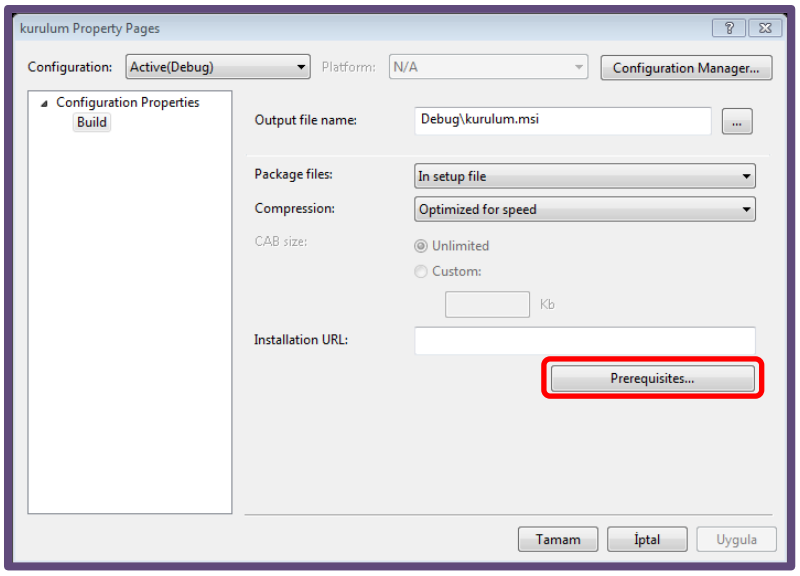
Şekil 16

Bu alana tıkladığımız zaman Visual Studio ekran görüntüsü Şekil 17'de gösterilmiştir. Yeni bir dosya tipi eklemek için: File types on target machine seçeneği üzerine farenin sağ tuşu ile tıklayarak Add File Type kmutunu vermemiz gerekir. Yeni eklenen dosya tipine ait Open seçeneği otomatik olarak gelmektedir.



Şekil 17

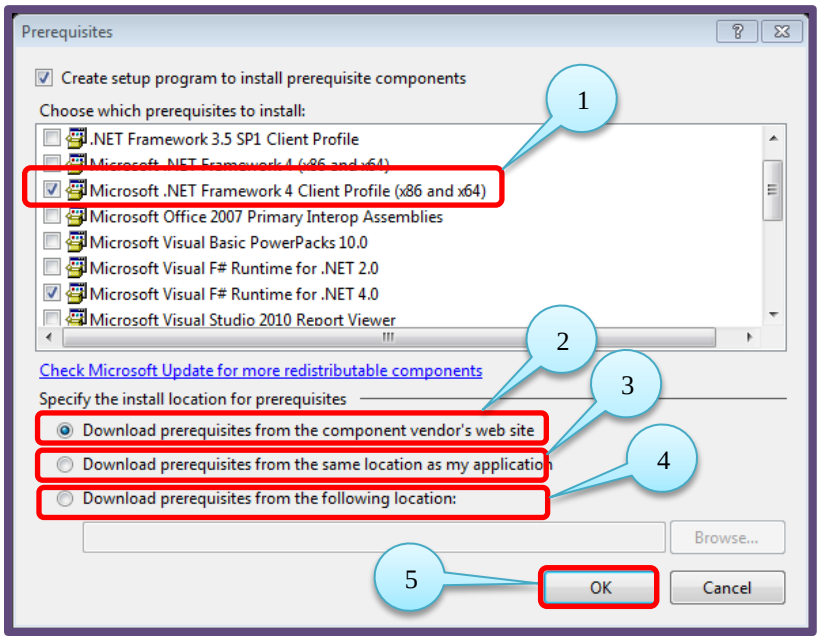
Şimdiye kadar kurulum dosyasının özelliklerini ayarlayabileceğimiz editörlerden bahsettik. Şimdi ise framework seçeneklerine ulaşacağımız seçeneklerin bulunduğu alana değinilecektir. Solution Explorer panelinde "kurulum" adlı setup dosyamız üzerine gelip farenin sağ tuşuna tıkladıktan sonra açılan menüden properties komutunu veriyoruz. Böylelikle karşımıza Şekil 18'deki kurulum Property Pages penceresi gelecektir. Bu pencere üzerinden programımız için gerekli olan yeterlilikleri ayarlayacağız.



Şekil 18

Projemizi geliştirirken framework'un hangi sürümünü kullanıyorsak kurulum dosyasına onu eklememiz gerekir.

Bu pencere üzerinden kırmızı kutucuk içinde gösterilen "Prerequisites…" butonuna tıklıyoruz. Karşımıza şekil – 19'daki pencere açılır ve bu pencere üzerinden 1 numaralı adımda gösterilen kırmızı kutucukla kullanacağımız framework versiyonunu belirtiyoruz.

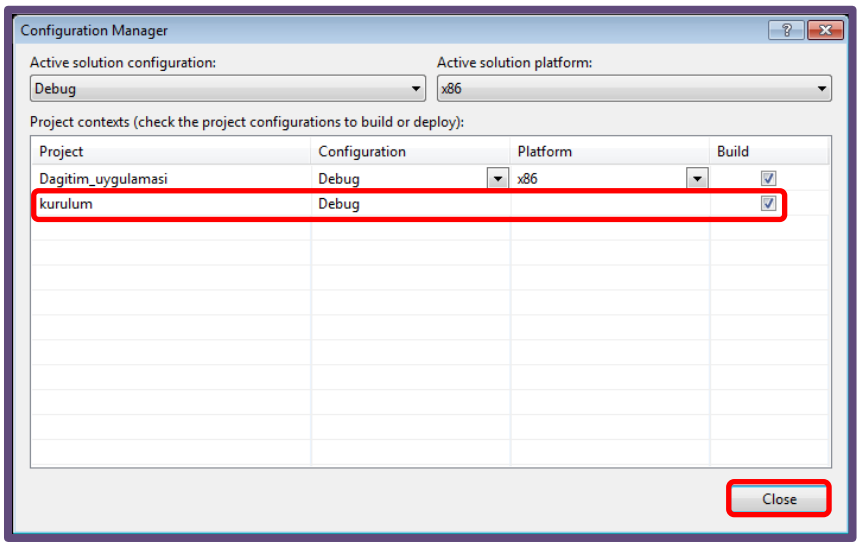


Şekil 19

Şekil – 19'da Specify the install location for prerequisites alanında üç farklı yükleme seçeneğimiz bulunmaktadır. Bunlardan ilki olan 2 numaralı seçeneğimiz yukarıdaki alandan seçtiğimiz dosyaların bilgisayarda olmaması hâlinde internet üzerinden yükleme yapmamızı sağlıyor. 3 numaralı seçenek ile kurulum dosyasına eklenmiş programlar sayesinde yüklenmesini sağlıyor. 4 numaralı seçenek işaretli olduğu takdirde de browse butonu aktifleşerek yolu gösterilen klasörden yükleme yapılması sağlanıyor. Bu işlemleri gerçekleştirdikten sonra 5 numaralı "OK" butonuna tıklayarak uyguladığımız adımları onaylıyoruz.

Kurulum dosyamızı tamamlamadan önce build edeceğimiz alanda kurulum projemizin seçili olup olmadığını kontrol etmeliyiz.

Kurulum dosyamızı tamamlamadan önce Build →Congfiguration Manager menü yolunu takip ederek karşımıza gelen Şekil 20'deki pencereden kurulum projemizin seçili olup olmadığını kontrol ediyoruz ve "Close" butonuna tıklıyoruz.



Şekil 20

Artık kurulum işlemimizi sonlandırabiliriz. Solution Explorer panelinde Dagitim_uygulamasi ve kurulum seçeneklerinin üzerine gelip yani proje adının üzerindeyken farenin sağ tuşu ile tıklandıktan sonra "Build" komutunu vererek kurulumumuzu tamamlamış oluruz.

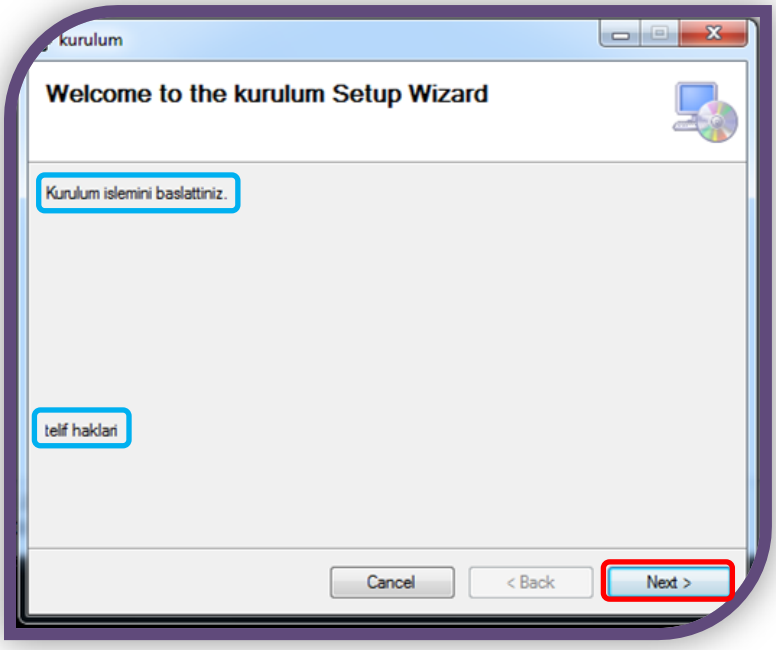
Visual Studio programını kapatıp hazırladığımız uygulamayı kaydettiğimiz alanı açtığımız zaman Debug klasörünün içerisinde .msi uzantılı kurulum paketi ve .exe uzantılı setup dosyasını görmek mümkündür.

KURULUMUN TEST EDİLMESİ

Bu başlık altında hazırlamış olduğumuz kurulum dosyasını çalıştırarak dört aşamadan oluşan kurulum sihirbazının çalışıp çalışmadığını kontrol edeceğiz. Aşamalar alt başlıklar hâlinde belirtilecektir. Burada dikkat edilmesi gereken noktalar mavi kutucuklar içinde gösterilecektir. Bunlar bizim kurulum dosyamızı hazırlarken üzerinde değişiklik yaptığımız alanlardır. Kırmızı kutucuklar ise kullanıcının tıklaması gereken butonları göstermektedir.

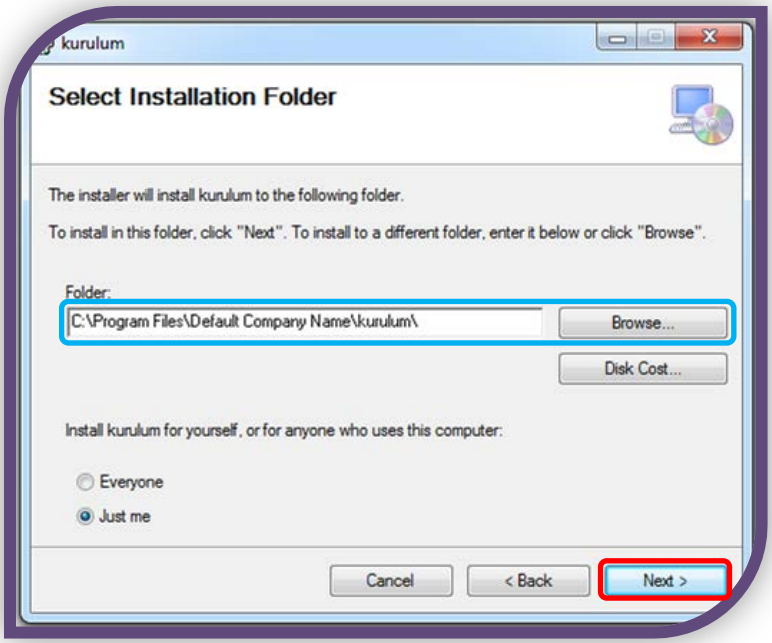
Şimdi projemizi kaydettiğimiz yolun içindeki "Debug" klasörüne erişelim. Akabinde setup.exe uzantılı dosyaya çift tıklayalım. Bundan sonra karşımıza gelecek olan pencereler aşamalar hâlinde aşağıda gösterilmektedir.

1. **Aşama**



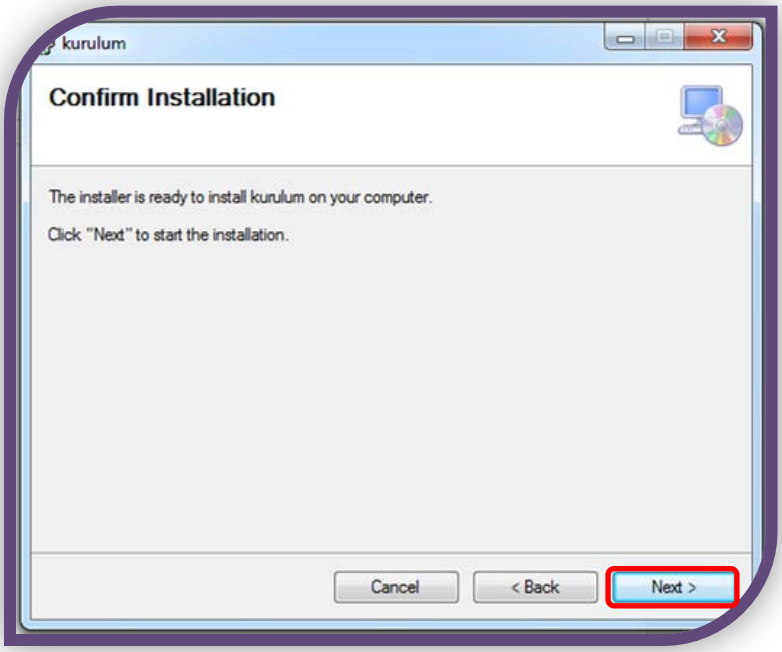
Şekil 21

2. Aşama



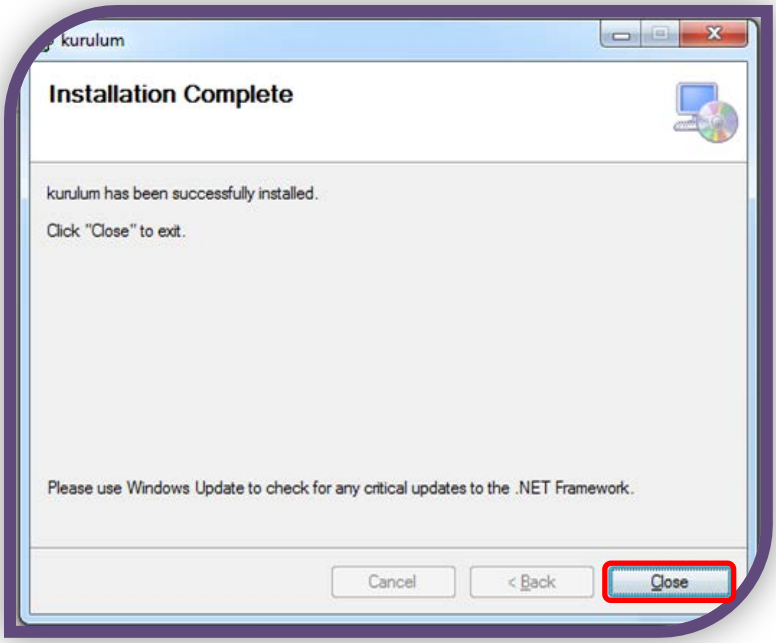
Şekil 22

3. Aşama



Şekil 23

4. Aşama



Şekil 24

Kurulum dosyanızı tamamladıktan sonra test etmek dağıtıma geçmeden önce faydalı olacaktır.

Bu dört aşamayı tamamladıktan sonra masaüstünde Dagitim_uygulamasi adlı kısayol dosyasını göreceğiz. Bu dosyaya çift tıkladığımız zaman programımız açılarak sorunsuz bir kurulum gerçekleştirdiğimizi onaylamış oluyoruz. Artık kurulum adlı klasörünüzü CD veya internet yoluyla kullanıma sunabilirsiniz.

Özet

- Visual Studio .NET ile hazırladığımız projelerin kurulum dosyasını oluşturmak için bize imkân tanımaktadır.
- Kurulum dosyamızı oluşturabilmek için hazırladığımız projeyi en az bir kere çalıştırmamız gerekmektedir. Bunun sebebi hem hata olup olmadığını görmek hem de projemizi bilgisayarımız üzerinde derlemektir.
- Şimdi File menüsünden new project komutunu vererek kurulum dosyasını oluşturabilmek için gerekli olan editörlerin bulunduğu alanı açalım. Bu işlemi yaptıktan sonra editörler üzerinden eklenilmesi ve yolunun belirtilmesi gereken dosya, klasör, referans, resim ve ses seçeneklerimizi oluşturalım.
- Son olarak bütün kontrollerimizi yaparak projemizi build edelim. Böylelikle hazırladığımız proje kullanıma hazır hâle gelmiştir.

DEĞERLENDİRME SORULARI

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

1. File → New → Project menü yolunun kısayol tuşu aşağıdakilerden hangisidir?

a) Ctrl + Shift + M
b) Ctrl + Shift + O
c) Ctrl + Alt + N
d) Ctrl + Shift + N
e) Ctrl + Alt + M

2. New Project penceresinde "Add to Solution" seçeneğini ayarlayacağımız alan hangi etiket yanında bulunmaktadır?

a) Solution
b) Name
c) Location
d) Solution Name
e) Installed templates

I. Application Folder
II. User's Desktop
III. User's Program Menu

3. Yukarıdaki kütüphanelerden hangisi veya hangileri file system editör içinde bulunmaktadır?

a) Yalnız I
b) I, II ve III
c) II ve III
d) I ve III
e) I ve II

4. Application folder klasörüne referans eklemek için hangi seçeneği kullanırız?

a) Folder
b) Project Output
c) Assembly
d) File
e) User's

5. Kurulacak programla ilgili dosya türlerinin düzenlendiği editör aşağıdakilerden hangisidir?

a) Launch conditions editor
b) User interface editor
c) Registry editor
d) File system editor
e) File types editor

6. Programın kurulacağı bilgisayar ile ilgili ayarlamaların yapıldığı editör aşağıdakilerden hangisidir?

a) User interface editor
b) Launch conditions editor
c) Registry editor
d) File system editor
e) File types editör

7. User's interface editör için aşağıda verilen başlıklardan hangisi yanlıştır?

a) Open
b) Start
c) Progress
d) End
e) Welcome

8. Launch conditions editör iki bölümden oluşmaktadır. Bu bölümler aşağıdakilerin hangisinde doğru olarak verilmiştir?

a) Search target machine – Progress
b) Search target machine – Welcome
c) Progress - Launch conditions
d) Welcome - Launch conditions
e) Search target machine – Launch conditions

9. Congfiguration Manager seçeneğine hangi menüden ulaşabiliriz?
a) File
b) Project
c) Debug
d) Build
e) Data

10. Prerequisites penceresinde yapabileceğimiz ayarlar aşağıdakilerin hangisinde doğru olarak verilmiştir?

a) Kullanıcının masaüstünde simge oluşturmamıza yarar.
b) Referans ekleyebildiğimiz penceredir.
c) Kullanacağımız framework versiyonunu seçeriz.
d) Kurulum dosyamızı build ederiz.
e) Kayıt defterini düzenleriz.

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

Aktaş, V. (2010). Visual Basic .Net 10, 1. Baskı, İSTANBUL: Kodlab Yayıncılık.

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

Demirli, N. ve İnan Y. (2010). Visual Basic .Net 2008 Ado .Net 3.0 &Sql Server 2008, ANKARA: Palme Yayıncılık.

Yazıcı, Ü. (2011). Visual Basic 2008 ile Windows Uygulamaları Geliştirmek, 1. Baskı, İSTANBUL: Pusula Yayıncılık.

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

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

Yanık, M. (2009). Visual Studio 2008 ile Microsoft Visual Basic 9.0 for .NET Frame 3.5, 1. Baskı, ANKARA: Seçkin Yayıncılık.

ÖRNEK PROGRAM UYGULAMALARI

İÇİNDEKİLER

- Giriş
  - Örnek Kodların Yazılması
  - Yazılan Kodların Visual Basic . Net Üzerinde Çalışma Hâlinin Gösterilmesi

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Karşılaşılan problemler üzerinde olası çözümler üretebilecek,
  - Kodların çalışma mantığını daha iyi bir şekilde kavrayabilecek,
  - Üretilen çözümlerin sonuçlarını bilgisayar ortamında gözlemleyebileceksiniz.

ATATÜRK ÜNİVERSİTESİ

ATA-AOF

GÖRSEL PROGRAMLAMA II

Okt. Daha ORHAN

ÜNİTE

14

GİRİŞ

Ezber sadece geçici çözümler üretir.

Bu ünitede örnek uygulamalara yer verilerek, önceki ünitelerde anlatılan konuların daha iyi kavratılması amaçlanmıştır. Örnekler ünitelerin sırasına göre belirlenmiş ve üniteyi kapsayacak ölçüde tercih edilmiştir.

Uygulamalı derslerde yeni öğrenilen bir ders veya konunun en iyi şekilde anlaşılması için; konu veya dersle alakalı çok sayıda alıştırma yapmak gerekmektedir. Bu sayede konunun veya dersin belirli özellikleri her yeni alıştırma ile daha rahat bir şekilde anlaşılmış olur.

Hazır kodları okuyup geçmenin yanı sıra bu kodları çalıştırıp olası sonuçları test etmek her zaman için bir programcıya yardımcı olacaktır. Bu açıdan örneklerin ilgili kodları Visual Studio üzerinde yazılarak çalışması görüntülenebilir. Örnekler hazırlanırken yapılması gereken adımlar tek tek belirtilip kodlar olduğu gibi verilmiştir. Örnekleri adım adım yapmanızı tavsiye ederiz.

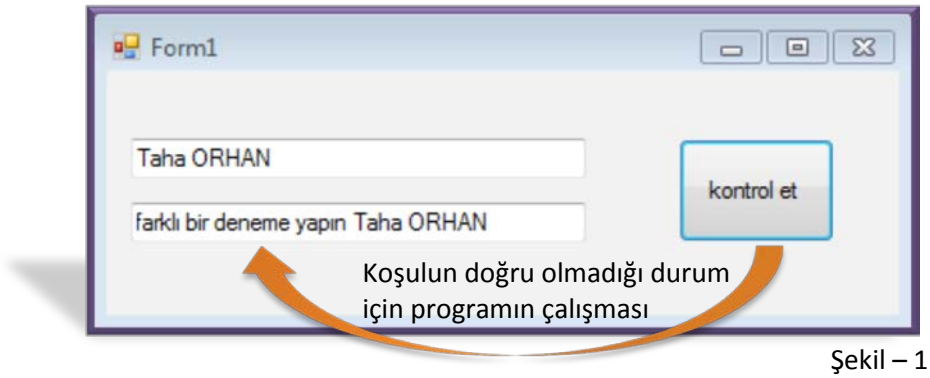
Örnekler üzerinde çalışırken ilk önce kendiniz çözüm yolu üretin ve hazır olan çözüm yoluyla kıyaslayın. Bununla beraber programınızı çalıştırdığınızda farklı değerler üzerinde birkaç deneme gerçekleştirmenizde fayda bulunmaktadır. Çünkü programda yazmış olduğunuz adımların mantığını daha rahat kavrayarak gelecekte karşılaşacağınız problemlere transfer etmeniz açısından fayda sağlayacaktır. Kodları ezberlemeniz sadece geçici çözüm sağlayacaktır. Bu durum dikkate alınarak ünitedeki örnekler hazırlanmıştır.

Örnek 1

•Windows form üzerine yerleştirilen bir textbox'a girilen değerin ilk harfi ve son harfi birbirine eşitse, girilen değerin küçük harflerini büyük harfe dönüştürüp diğer textboxa yazdıran; ilk harf ve son harf birbirine eşit değilse bu değer içine "farklı bir deneme yapın" mesajını ekleyip diğer textboxa yazdıran programın kodları ve örnek çıktısı.

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim ilkh, sonh As String
        ilkh = Microsoft.VisualBasic.Left(TextBox1.Text, 1)
        sonh = Microsoft.VisualBasic.Right(TextBox1.Text, 1)
        If ilkh = sonh Then
            TextBox2.Text = TextBox1.Text.ToUpper
        Else
            TextBox2.Text = TextBox1.Text.Insert(0, "farklı bir
deneme yapın ")
        End If
    End Sub
```

İnsert metodu kullanılırken eklenilmek ifadenin nereye ekleneceği sayısal olarak belirtilmelidir.

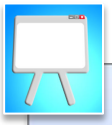


Şekil – 1

Bu örnekte birinci ünitede anlatılan karakter özellikli hazır metotlardan faydalanılmıştır. Örnek incelendiği zaman ilk ulaşmamız gereken özellik ifadenin baştaki ve sondaki değeridir. Buna göre karşılaştırma yapılmış ve doğru olup olmama durumuna göre istenilen işlemler gerçekleştirilmiştir.

Bireysel Etkinlik

• Örnek 1'e benzer şekilde girilen metnin baştan ikinci karakteri ile sondan ikinci karakterini karşılaştırıp doğru olup olmama durumları için test ediniz.



Örnek 2

•Bir muhasebe programında işçilerin ad soyad ve yaş bilgileri alındıktan sonra o işçiye ait maaş bilgisini buton yardımıyla hesaplatmak amacıyla bir program hazırlanmaktadır. Bu hesap işlemi için: işçinin işe başlangıç tarihini sistem zamanından alıp ayın onbeşinden önce işe başlayanların alacakları maaşlarını ve ayın 15'den sonra işe girenlerin kesintili maaşlarını 2000tl üzerinden hesaplayan program.

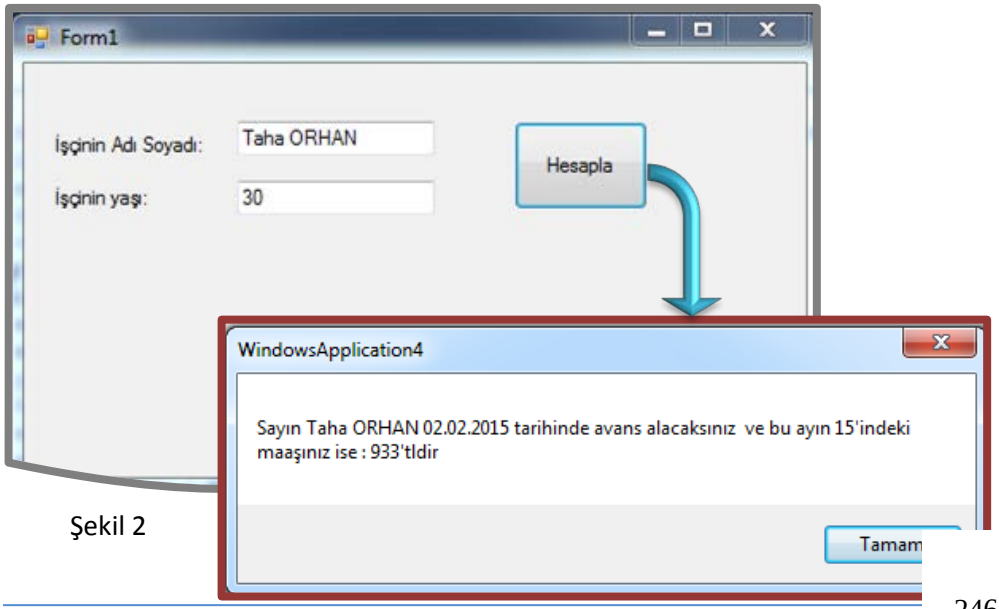
Hesap yapılırken dikkat edilecek kurallar:

- Kesirli olarak yatan para bir üst değere yuvarlanır.
- Çalışan işçinin yaşının karekökü hesaplanır ve çıkan sonuç ile 0 arasında rastgele bir değer belirlenerek eklenen gün sayısına göre avans alacağı gün belirlenir.

Random metodu kullanılırken istenilen aralıkta bir sayı üretilmesi bekleniyorsa next özelliğine istenilen parametre girilmelidir.

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim isegiris As DateTime = Today
        Dim avansgunu As DateTime
        Dim isegirisgun, maas, yaskarekok, tam, avans As
Integer
        Dim rastgele As New Random
        isegirisgun = isegiris.Day
        If isegirisgun < 15 Then
            maas = 2000 / 30 * (15 - isegirisgun)
            maas = Math.Round(maas, 1)
            yaskarekok = Math.Sqrt(Val(TextBox3.Text))
            tam = Int(yaskarekok)
            avans = rastgele.Next(tam)
            avansgunu = isegiris.AddDays(avans)
            MsgBox("Sayın " & TextBox1.Text & " " &
avansgunu & " tarihinde avans alacaksınız " & " ve bu ayın
15'indeki maaşınız ise : " & maas & "'tldir")
        ElseIf isegirisgun > 15 Then
            maas = 2000 / 30 * (isegirisgun - 15)
            maas = Math.Round(maas, 1)
            yaskarekok = Math.Sqrt(Val(TextBox3.Text))
            tam = Int(yaskarekok)
            avans = rastgele.Next(tam)
            avansgunu = isegiris.AddDays(avans)
            MsgBox("Sayın " & TextBox1.Text & " " &
avansgunu & " tarihinde avans alacaksınız " & " ve
önümüzdeki ayın 15'inde maaşınızdan yapılacak kesinti: -" &
maas & "'tldir")
        Else
            MsgBox("tam maas alacaksınız")
        End If
    End Sub
```

Bu örnek hazırlanırken 2. Ünitede anlatılan sayısal özellikli metotlardan ve zaman fonksiyonlarından faydalanılmıştır. Şekil 2 'de programın çalışma hâli görüntülenmiştir. Sizin de sistem tarihinizi değiştirerek olası sonuçları görüntülemenizde fayda bulunmaktadır.



Şekil 2

Örnek 3

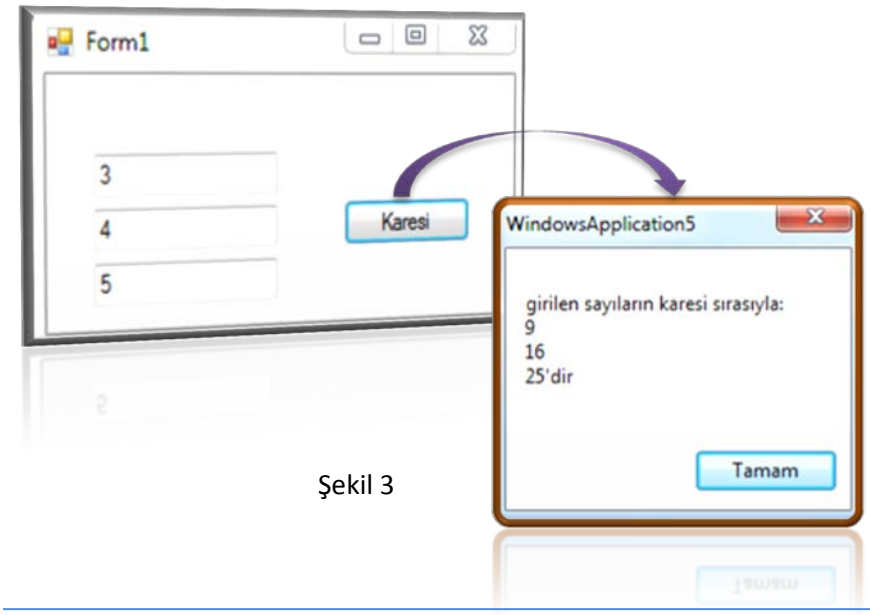
- Windows form üzerine yerleştirilen üç textbox içindeki sayıların karesini hesaplayıp mesaj yoluyla gösteren programın örnek kod yazılımı ve çalışması.

Prosedür ve fonksiyonlar: Kod tekrarının önlenmesi, daha okunaklı programların oluşturulmasını, program geliştirmenin kolaylaştırılmasını ve diğer projelerde kullanılması gibi avantajlar sağlar.

Bu örnekte üçüncü ünitede anlatılan fonksiyonlardan faydalanılmıştır. Fonksiyonları kullanmamızın sebebi her sayı için ayrı ayrı kare hesaplama işlemi yapmadan tek bir işlem gerçekleştirerek fazla kod yazımından programımızı arındırmaktır. Kod tekrarı önlenerek, daha okunaklı ve geliştirilmesi kolay bir program tasarlamış oluruz.

```
Public Class Form1
    Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim x, y, z As Integer
        x = Val(TextBox1.Text)
        y = Val(TextBox2.Text)
        z = Val(TextBox3.Text)
        MsgBox("girilen sayıların karesi sırasıyla: " &
vbNewLine & kare(x) & vbNewLine & kare(y) & vbNewLine & kare(z)
& "'dir")
    End Sub
    Function kare(ByVal a As Integer) As Integer
        kare = a * a
    End Function
End Class
```

Yukarıdaki örneğimizi incelediğimiz zaman fonksiyon tanımındaki “a” değişkeninin hesap işlemi gerçekleştirilecek değer olduğu ve döndürülen ifadenin “kare” değişkeni ile sağlandığını görmek aşikârdır. Programı çalıştırdığımızda Şekil 3’teki ekran görüntüsünü elde edebiliriz.



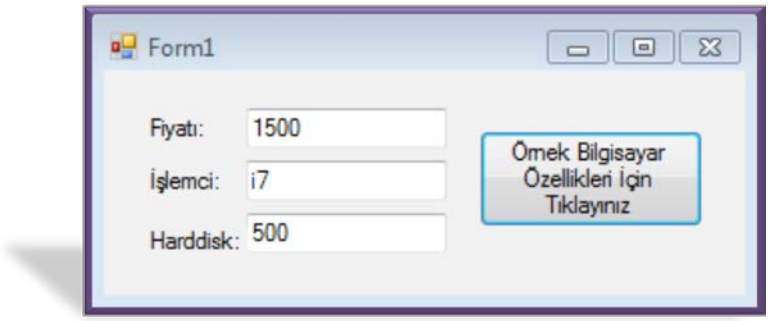
Şekil 3

Örnek 4

•Bilgisayar adında bir sınıf ve bu sınıfa ait farklı veri tiplerine sahip işlemci, kapasite ve fiyat olmak üzere özellikler oluşturalım ve bunlar içinde fiyat özelliğini salt okunur olarak ayarlayalım. Bu sınıf içinde tanımlanan bir örnek bilgisayar oluşturup özelliklerine değer atayalım.

Var olan bir projeye class eklemek için Project menüsünde Add Class seçeneği tıklanır.

Bu örnekte dördüncü ünitede anlatılan sınıf yapısından faydalanılmıştır. Form üzerine yerleştirilen textboxlara buton yardımı ile değer ataması gerçekleştirilmiştir. Programın çalışması Şekil 4'te gösterilmiştir.



Şekil 4

Fakat burada dikkat edilmesi gereken bir nokta bulunmaktır. "fiyat" nesnesinin özelliği sadece okunabilir olarak ayarlanmıştır. Şayet biz bu özelliğe butonun tıklanması esnasında değer atayacak olursak programımızı çalıştıracağımız zaman bu özelliğin salt okunabilir olarak ayarlandığı ve değiştirilemeyeceğine dair hata mesajı verecektir.

```
Public Class bilgisayar
    Private _islemci As String
    Private _kapasite As String
    Private _fiyat As Integer
    ReadOnly Property fiyat() As Integer
        Get
            _fiyat = 1500
            Return _fiyat
        End Get
    End Property
    Property kapasite() As Integer
        Get
            Return _kapasite
        End Get
        Set(ByVal Value As Integer)
            _kapasite = Value
        End Set
    End Property
    Property islemci() As String
        Get
            Return _islemci
        End Get
        Set(ByVal Value As String)
            _islemci = Value
        End Set
    End Property
End Class
```

Yukarıda verilen kodlar project menüsünden add class seçeneği tıklanarak karşımıza açılan sınıf özelliklerine; aşşağıda verilen kodlar ise dizaynını yaptığımız windows form içindeki butona yazılmalıdır.

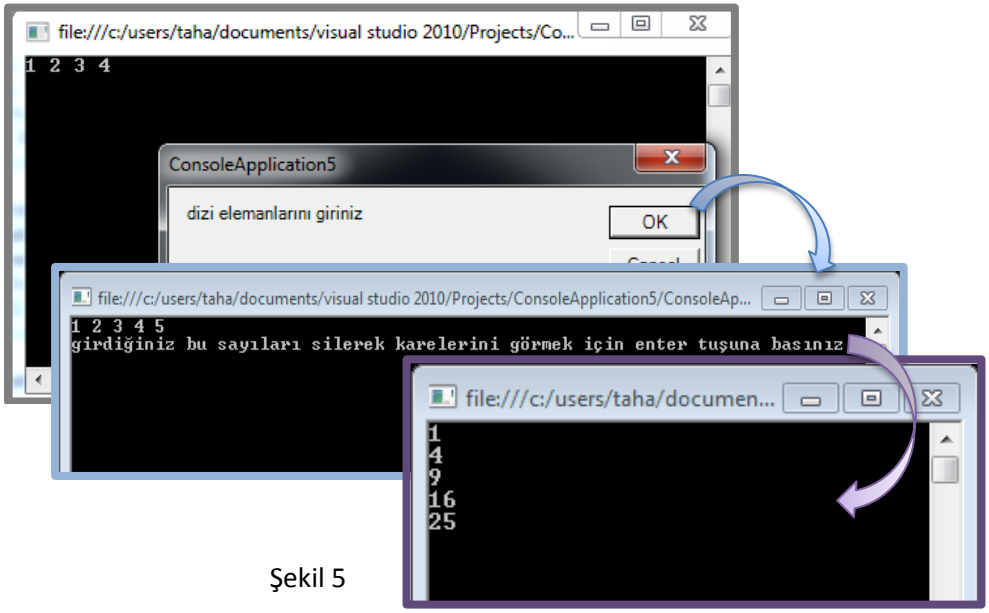
```
Public Class Form1
    Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim ornekpc As New bilgisayar
        ornekpc.islemci = "i7"
        ornekpc.kapasite = 500
        TextBox1.Text = ornekpc.fiyat
        TextBox2.Text = ornekpc.islemci
        TextBox3.Text = ornekpc.kapasite
    End Sub
End Class
```

Örnek 5

•Konsol ekranında klavyeden girilen beş elemanlı bir a dizisinin elemanlarını aralarında bir boşluk bırakarak yan yana yazdırdıktan sonra ekranı temizleyerek bu dizi elemanlarının karelerini alt alta yazdıran program.

Konsol ekranında WriteLine metodu ile mesaj yazdırıldıktan sonra satır sonlanır ve diğer seçenekler bir sonraki satırdan devam eder.

```
Module Module1
    Sub Main()
        Dim a(4) As Integer
        Dim i, k As Integer
        For i = 0 To 4
            a(i) = InputBox("dizi elemanlarını giriniz")
            Console.Write(a(i) & " ")
        Next
        Console.Write(vbNewLine & "girdiğiniz bu sayıları
silerek karelerini görmek için enter tuşuna basınız")
        Console.ReadLine()
        Console.Clear()
        For i = 0 To 4
            k = a(i) * a(i)
            Console.WriteLine(k)
        Next
        Console.ReadLine()
    End Sub
End Module
```



Şekil 5

Bu örnek beşinci ünitede anlatılan konsol uygulamalarından faydalanılarak yapılmıştır. Programın çalışma hâlini Şekil 5'i inceleyerek görebilirsiniz. Benzer şekilde uygulamalar gerçekleştirmeniz sizin hayal güzünüze bağlıdır.

Örnek 6

•Kullanıcıdan bilgisayar özelliklerini almak isteyen bir programın örnek kod yazılımı ve ekran görüntüsü.

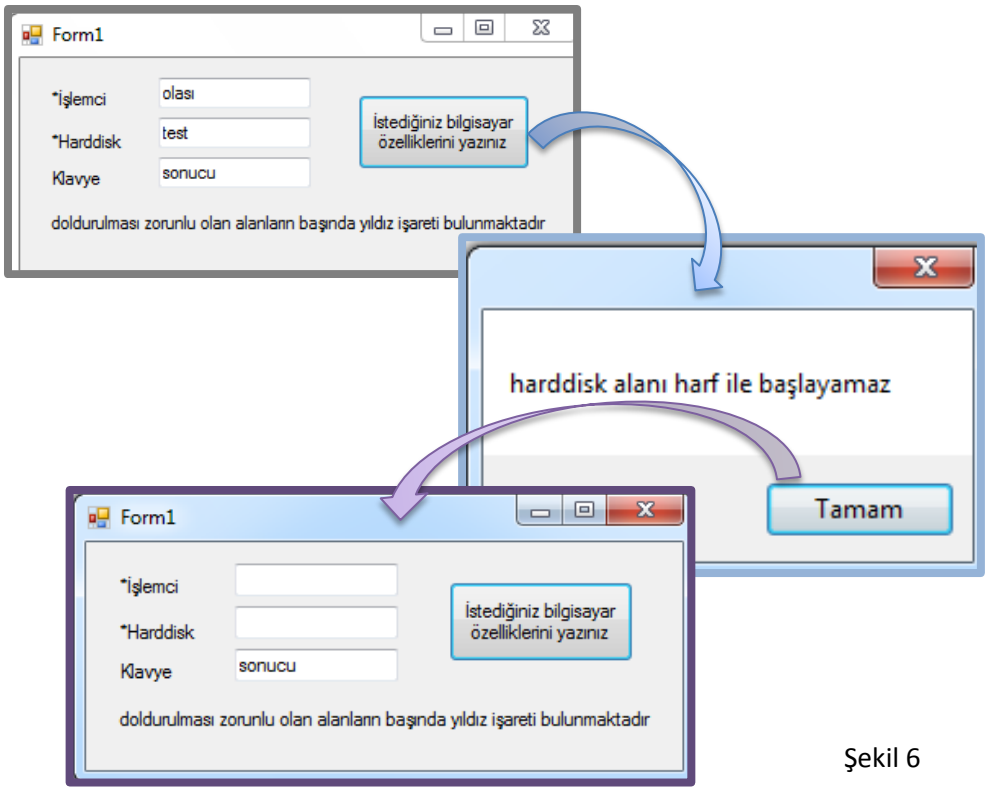
Özellikler girildikten sonra dikkat edilecek kurallar:

- İşlemci ve harddisk alanları boş bırakılmayacak.
- İşlemci alanı harflerden, harddisk alanı ise rakamlardan oluşması gerekecektir.

Throw satırından sonraki kod satırı işletilmez.

```
Public Class Form1
    Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim islemci, klavye As String
        Dim harddisk As String
        islemci = TextBox1.Text
        harddisk = Val(TextBox2.Text)
        klavye = TextBox3.Text
        If ((islemci <> "") And (harddisk <> "")) Then
            Try
                islemci = Convert.ToString(TextBox1.Text)
                harddisk = Convert.ToInt32(TextBox2.Text)
                klavye = Convert.ToString(TextBox3.Text)
            Catch uyarı As FormatException
                MessageBox.Show("harddisk alanı harf ile
başlayamaz")
            Catch uyarı As Exception
                MessageBox.Show("islemci alanı rakam ile
başlayamaz")
            Finally
                TextBox1.Text = Nothing
                TextBox2.Text = Nothing
            End Try
        Else
            Throw New Exception("Doldurulması zorunlu olan
alanları boş bırakmayınız")
        End If

    End Sub
End Class
```

Bu örnek hazırlanırken altıncı ünitede anlatılan hata yakalama ve ayıklama özelliklerinden faydalanılmıştır. Hazırlanan programın çalışır durumdaki hâli ve olası bir sonuç üzerindeki ekran görüntüsü Şekil 6'da gösterilmiştir.



Şekil 6

Örnek 7

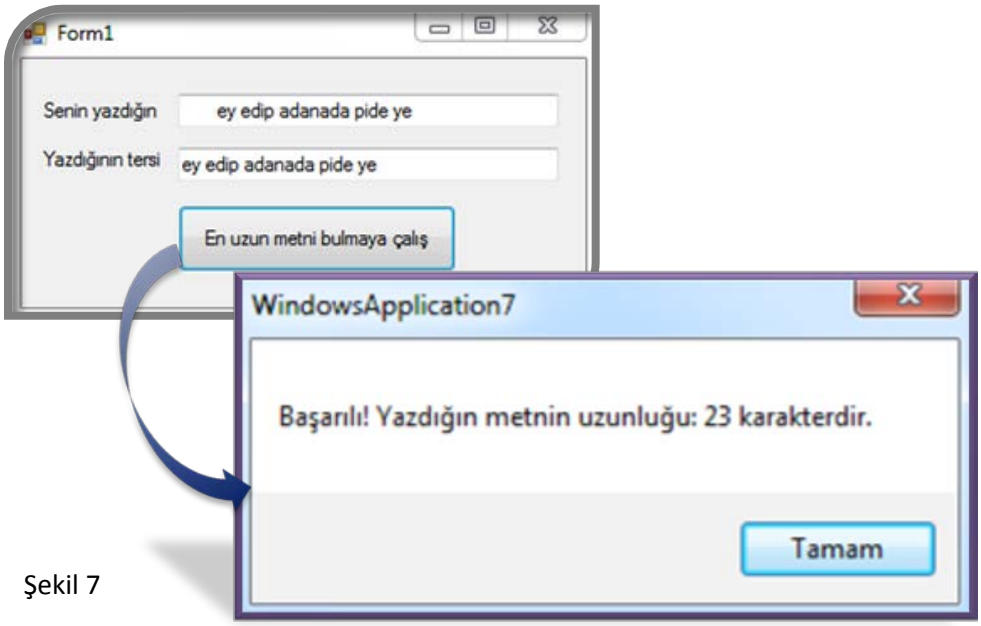
•Tersten yazılışı ile aynı olan en uzun metni bulmak için hazırlanan bir programın örnek kodu ve ekran görüntüsü.

Metin girildikten sonra başında veya sonunda boşluk varsa bu boşluklar silinerek kontrol işlemi gerçekleştirilecek. Metnin uzunluğu mesaj yoluyla belirtilecektir.

Trim metodu ile var olan ifadenin hem sağındaki hem de solundaki boşluklar silinir.

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim yazdigin, tersi As String
        Dim uzunluk As Integer
        yazdigin = Trim(TextBox1.Text)
        tersi = StrReverse(yazdigin)
        TextBox2.Text = String.Copy(tersi)
        If tersi = yazdigin Then
            uzunluk = Len(yazdigin)
            MsgBox("Başarılı! Yazdığın metnin uzunluğu: " &
uzunluk & " karakterdir.")
        Else
            MsgBox("Üzgünüm! Yazdigin ile tersi aynı değil.")
        End If
    End Sub
```

Bu örnekte ünite yedide anlatılan temel string işlemlerinden faydalanılmıştır. Yukarıda yazdığımız kodlara göre hazırlanan örnek incelendiği zaman Şekil 7'deki gibi ekran görüntüsünü görmek mümkündür.



Şekil 7

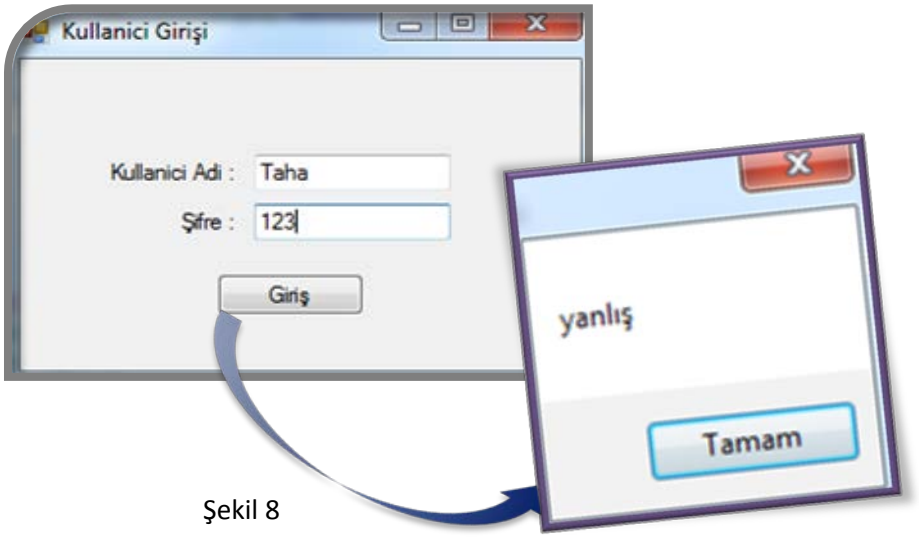
Örnek 8

- İki formdan oluşan bir projede başlangıç formunda kullanıcı adı ve şifre istenilmektedir. Textboxlara girilen değer ilişkilendirilmiş olan local veri tabanından sorgulanarak diğer forma geçişi sağlayan veya hata mesajı veren bir projenin çalışması ve örnek kod yazılımı.

Bu örnek, sekizinci, dokuzunca ve onuncu ünitelerdeki bilgiler ışığında yapılmıştır. Veri tabanı olarak Access programı tercih edilmiştir. Buna istinaden her kod satırının yanında açıklaması verilerek konunun daha iyi anlaşılması

amaçlanmıştır. İlk olarak Şekil 8'de verilen programın çalışmasına bir göz atalım ve hemen altında verilen kodları inceleyelim.

Verileri saklamak, bu veriler üzerinde işlemler gerçekleştirmek ve istenildiği takdirde bu verilere ulaşmak için veritabanları kullanılır.



Şekil 8

```
Imports System.Data.OleDb
Public Class Form1
    Public baglanti As New
OleDbConnection("Provider=Microsoft.Ace.OLEDB.12.0;Data
Source=Taha.accdb")
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Text = "Kullanici Girişi" 'burada başlangıç
formumuzun ismini değiştiriyoruz
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim a, b As String
        Dim c As Integer
        Dim i As Integer
        a = TextBox1.Text 'a kullanıcı adını tutan değişken
        b = TextBox2.Text 'b kullanıcı şifresini tutan
değişken
        baglanti.Open() 'veritabanı bağlantısını açıyoruz
        Dim komut As New OleDbCommand("SELECT
kullaniciAd,kullaniciSifre FROM kullanici_giris", baglanti)
'yaptığımız bağlantıdan sorgu işlemini gerçekleştiriyoruz
        Dim k As New OleDbDataAdapter(komut) 'sorgudan gelen
verileri tutuyoruz
        Dim t As New DataTable() 'yeni bir tablo veri
tablosu oluşturmamızı sağlıyor
        k.Fill(t) 'bu tabloyu adaptor ile dolduruyoruz
        Dim f As New DataSet ' bir veri seti oluşturuyoruz
        f.Tables.Add(t) 'bu veri setine oluştuğumuz t
tabloyu atıyoruz
        BindingSource1.DataSource = f 'tabloyu
ilişkilendiriyoruz
        BindingSource1.DataMember = f.Tables(0).TableName
'tablonun indis numarasını veriyoruz ve hangi tabloyu
alacağını belirtiyoruz
        c = BindingSource1.Count() 'tablodaki kayıt sayısını
c değişkeninin içine aktarıyoruz
        Dim oku As OleDbDataReader 'tablomuzdaki satırları
incelemek için oku diye bir değişken tanımlıyoruz
        oku = komut.ExecuteReader() 'okuma işlemini
başlatıyoruz
        While oku.Read() 'okuma işlemini döngü ile yapmamız
gerekiyor
            i = i + 1 'burda bir sayaç koyduk
            If (a = oku("kullaniciAd")) And (TextBox2.Text =
oku("kullaniciSifre")) Then
                Form2.Show()
                Exit While
            End If
            If c = i Then
                MessageBox.Show("yanlış")
                Exit While
            End If
        End While
        baglanti.Close()
    End Sub
End Class
```

Örnek 9

•Üniversite isim uzayı altında eğitim ve mühendislik olmak üzere iki isim uzayı daha oluşturup bunların öğrenci sayılarını textboxlardan alıp mesaj yoluyla karşımızda gösteren programın örnek kod yazılımı ve ekran görüntüsü.

İçiçe isim uzayları oluştururken hiyerarşik yapıya dikkat etmek gerek hataları ayıklarken gerekse programı geliştirirken kolaylık sağlayacaktır.

Bu kodlar eklenen classa olduğu gibi yazılacaktır.

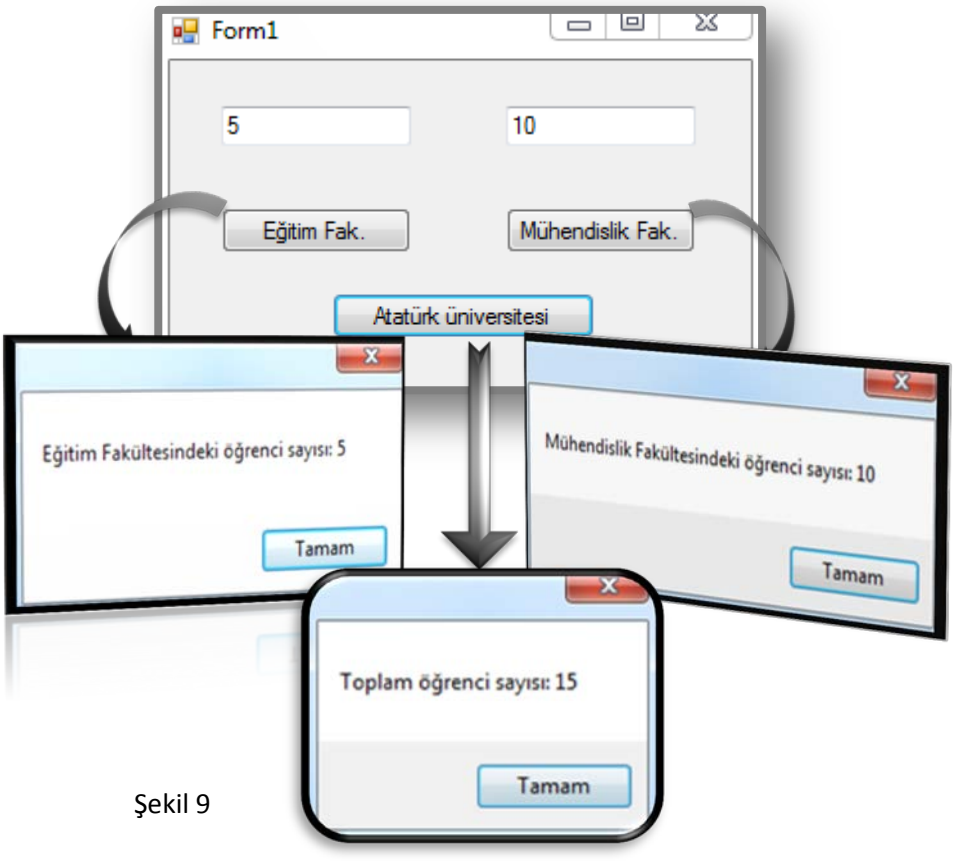
```
Namespace uni
    Public Class atauni
        Private _ogrsay As Integer
        Property ogrsay() As Integer
            Get
                Return _ogrsay
            End Get
            Set(ByVal Value As Integer)
                _ogrsay = Value
            End Set
        End Property
        Public Sub kacogr()
            MessageBox.Show("Toplam öğrenci sayısı: " &
_ogrsay.ToString())
        End Sub
    End Class
    Namespace egitim
        Public Class atauni_e
            Private _ogrsay_e As Integer
            Property ogrsay_e() As Integer
                Get
                    Return _ogrsay_e
                End Get
                Set(ByVal Value As Integer)
                    _ogrsay_e = Value
                End Set
            End Property
            Public Sub kacogr_e()
                MessageBox.Show("Eğitim Fakültesindeki
öğrenci sayısı: " & _ogrsay_e.ToString())
            End Sub
        End Class
    End Namespace
```

Bu alanı “eğitim” isim uzayına benzer şekilde siz doldurunuz.

```
    End Namespace
End Namespace
```

```
   Bu kodlar ise Windows form içine yerleştirilen butonlara
                        yazılacaktır.

Public Class Form1
    Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        Dim uninesnesi As New uni.atauni
        uninesnesi.ogrsay = Val(TextBox1.Text) +
Val(TextBox2.Text)
        uninesnesi.kacogr()
    End Sub
    Private Sub Button2_Click(sender As System.Object, e As
System.EventArgs) Handles Button2.Click
        Dim egitimnesnesi As New uni.egitim.atauni_e
        egitimnesnesi.ogrsay_e = Val(TextBox1.Text)
        egitimnesnesi.kacogr_e()
    End Sub
    Private Sub Button3_Click(sender As System.Object, e As
System.EventArgs) Handles Button3.Click
        Dim muhendisliknesnesi As New uni.muhendislik.atauni_m
        muhendisliknesnesi.ogrsay_m = Val(TextBox2.Text)
        muhendisliknesnesi.kacogr_m()
    End Sub
End Class
```

Bu örnek, İsim Uzayı Hazırlamak ve Kullanmak adlı 12. Üniteden faydalanılarak yapılmıştır. Yukarıdaki kodlar detaylıca incelendikten sonra programın çalışmasını Şekil 9'a bakarak görebiliriz.



Şekil 9

Özet

- Ünitenin tamamında, önceki ünitelerde anlatılan konulara yönelik örnekler bulunmaktadır. Bu özgün örnekler özenle seçilerek sizlerin anlayabileceği düzeyde basamak basamak izah edilmiştir.
- Siz de bu basamakları ve tavsiye edilen bireysel etkinlikleri farklı seçenekler kullanarak tekrar tekrar çalıştırdığınız takdirde öğrendiğiniz bilgilerin kalıcılığı artacaktır.
- Ünitede hazırlanan örneklerin kodları ve çalıştırıldığı takdirde ekran görüntüleri şekillerle gösterilmiştir.

Değerlendirme sorularını sistemde ilgili ünite başlığı altında yer alan "bölüm sonu testi" bölümünde etkileşimli olarak cevaplayabilirsiniz.

DEĞERLENDİRME SORULARI

1. " Erzurum" ifadesindeki "Erz" değerini alarak bir değişkene aktarabilmemizi veya üzerinde işlem yapabilmemizi sağlayan kod yapısı aşağıdakilerden hangisidir?

a) Microsoft.VisualBasic.Right("Erzurum",3)
b) Microsoft.VisualBasic.Left("Erz",1)
c) Microsoft.VisualBasic.Insert("Erzurum",3)
d) Microsoft.VisualBasic.Left("Erzurum",3)
e) Microsoft.VisualBasic.Left("Erzurum",1)

2. İçinde tarih tutan "TD" isimli bir değişkene 7 gün daha eklemek için kullanacağımız kod parçası aşağıdakilerden hangisidir?

a) TD.AddDays(7)
b) TD.AddMounts(7)
c) TD + 7
d) TD.AddYears(7)
e) TD.AddDays.Insert(7)

3.
```
Function kare(ByVal a As Integer) As Integer
    kare = a * a
End Function
```
Yukarıdaki kod satırında yapılan işlem hangisinde doğru açıklanmıştır?

a) Ana programdan gelen bir verinin tür dönüşümü yapılmıştır.
b) Ana programdan gelen integer verinin karesi hesaplanarak aynı türden geri döndürülmüştür.
c) Form üzerine kare şekli çizilmiştir.
d) Ana programdan gelen integer verinin karekökü hesaplanarak aynı türden geri döndürülmüştür.
e) Hatalı bir kod satırıdır.

I. Public
II. Private
III. Protected

4. Yukardakilerden hangisi ya da hangileri Visual Basic.Net içerisinde kullandığımız erişim belirleyici anahtar kelimelerindendir?

a) I ve II
b) II ve III
c) I, II ve III
d) Yalnız I
e) Yalnız II

5. Aşağıdaki kodlardan hangi konsol ekranında bulunan yazıların temizlenmesi amacıyla kullanılır?

a) Console.Write()
b) Console.WriteLine("")
c) Console.Read()
d) Console.ReadLine()
e) Console.Clear()

6. Aşağıdaki hata yakalama ve ayıklama yöntemlerinden hangisi hata olsun veya olmasın kod bloğunun sonundaki satırın işletilmesini sağlar?

a) Try – Catch
b) Try – Catch – Finally
c) Throw
d) Exception
e) Delete

7. Aşağıdakilerden hangisinde TextBox1'e girilen string ifadenin solundaki boşluklar silinerek tersten yazılışı "TY" değişkenine aktarılır?

a) TY = StrReverse(LTrim(TextBox1.Text))
b) TY = StrReverse(RTrim(TextBox1.Text))
c) TY = LTrim (StrReverse (TextBox1.Text))
d) StrReverse(LTrim(TextBox1.Text)) = TY
e) LTrim(TextBox1.Text) = TY

I. Copy
II. Len
III. ToLower
IV. Mid

8. Yukarıda verilen string metotlarından hangi ikisi kopya almak ve uzunluk ölçmek için kullanılır?

a) III ve IV
b) II ve III
c) I ve IV
d) I ve III
e) I ve II

I. OleDbDataReader
II. DataSet
III. SqlDataAdapter

9. Yukarıda verilenlerden hangisi veya hangileri .NET içindeki managed providerlara özel sınıflardandır?

a) Yalnız I
b) I ve III
c) II v III
d) I ve III
e) I, II ve III

10. İsim uzayları hangi anahtar kelime ile tanımlanır?

a) Public Class
b) Private
c) Namespace
d) Property
e) Get

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

Aktaş, V. (2010). Visual Basic .Net 10, 1. Baskı, İSTANBUL: Kodlab Yayıncılık.

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

Demirli, N. ve İnan Y. (2010). Visual Basic .Net 2008 Ado .Net 3.0 &Sql Server 2008, ANKARA: Palme Yayıncılık.

Yazıcı, Ü. (2011). Visual Basic 2008 ile Windows Uygulamaları Geliştirmek, 1. Baskı, İSTANBUL: Pusula Yayıncılık.

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

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

Yanık, M. (2009). Visual Studio 2008 ile Microsoft Visual Basic 9.0 for .NET Frame 3.5, 1. Baskı, ANKARA: Seçkin Yayıncılık.