MÜHENDİSLİK FAKÜLTESİ

BİL102 BİLGİSAYAR YAZILIMI II

C#

DERS NOTLARI

ANKARA

2010

1 C# Nedir?

C#, kendinden önce geliştirilen C ve C++ dillerinden türetilmiş java programlama dilinin güçlü özelliklerini barındıran bir programlama dilidir.

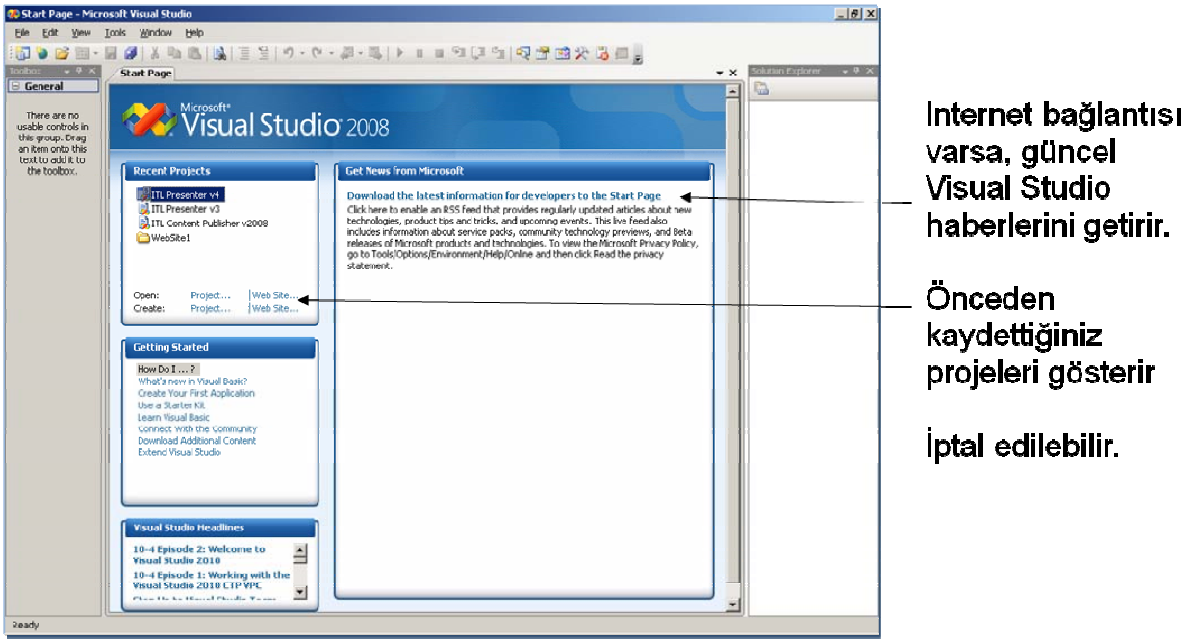
2 Değişken türleri

Değişken isimlendirme kuralları C ile aynıdır. C ve C++ dakine benzer değişken türleri bulunmaktadır. Aşağıda tablo halinde verilmektedir.

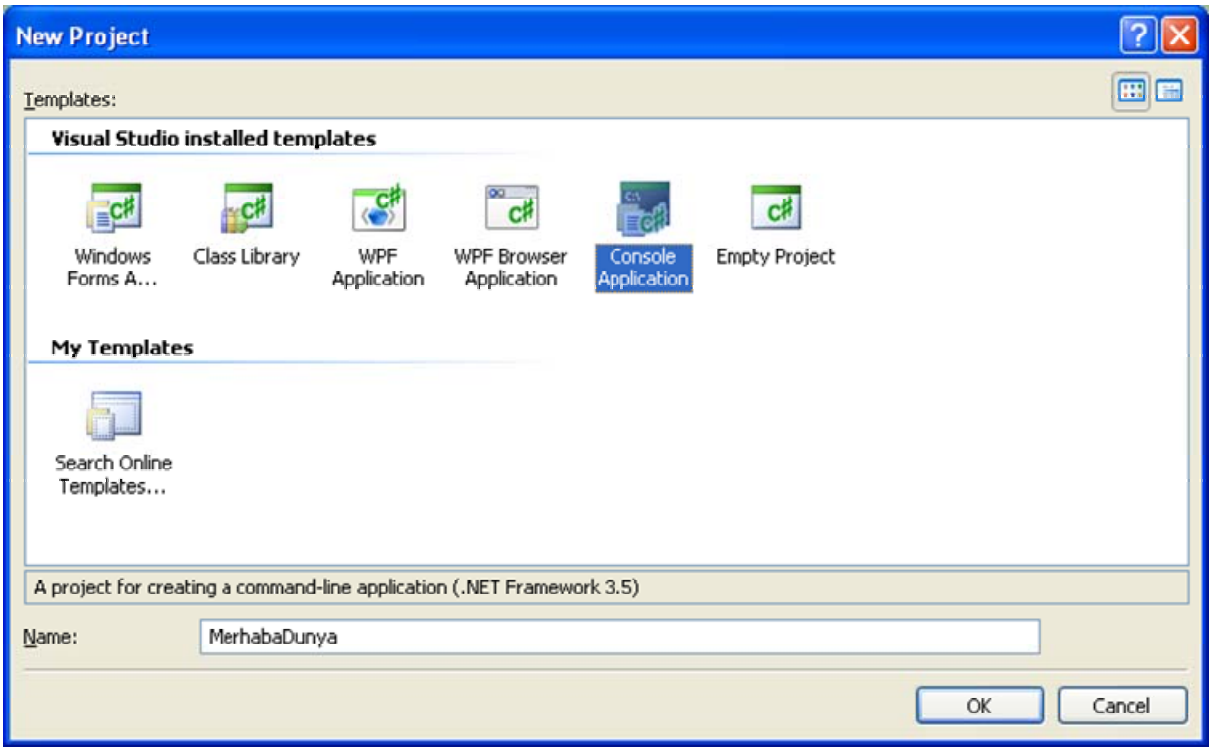
C# Tür	Uzunluk (Byte)	Value range
sbyte	**1**	– 128 … 127
byte	**1**	0 … 255
short	**2**	– 32768… 32767
ushort	**2**	0 … 65535
int	**4**	– 2147483648 … 2147483647
uint	**4**	0 … 4294967295
long	**8**	– 9223372036854775808 … +9223372036854775807
ulong	**8**	0 … 18446744073709551615
decimal	**12**	$1.0x10^{-28}$ - $7.9 x10^{28}$
float	**4**	$1.5x10^{-45}$ - $3.4x10^{38}$
double	**8**	$5.0 x 10^{-324}$ - $1.7 x 10^{308}$
char	**2**	(between 0 and 65535 (‘a’, ‘?’, ‘7’,’-’)
string	**-**	“Bil102”, “123”
bool	**1**	true ve false

3 İlk C# programımız

Burada geleneği bozmayıp yine “Merhaba Dünya” yazan program ile başlayacağız. C# programları, Microsoft un Visual Studio IDE si kullanılarak geliştirilir. Bu IDE nin açılış sayfası (start page) aşağıdaki gibidir.



İlk C# programımız bir konsol uygulaması olacak. Onun için öncelikle Dosya (file) mönüsünden Yeni Proje (New Project) emrini verin. Aşağıdaki gibi bir pencere ile karşılaşacaksınız. Burada “Console Application” ikonunu seçiniz. Visual Studio IDE sinde Solution: Bir ya da birden fazla proje içeren çalışma yapısı, Project: Bir uygulama için gerekli olan tüm dosyaları ve ayarlamaları içeren çalışma yapısı olarak tanımlanmıştır.

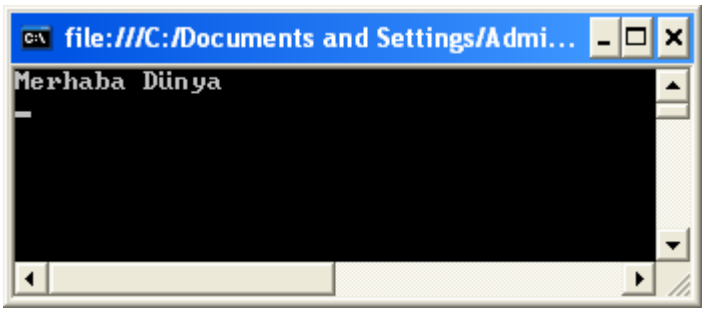


Gelen ortamda aşağıdaki kodları yazarak ilk programımızı F5 kısayol tuşu ile çalıştırabiliriz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MerhabaDunya
{
    class Program
    {
        static void Main(string[] args)          // C deki main ile aynı
        {
            Console.WriteLine("Merhaba Dünya"); //Printf karşılığı
            Console.ReadKey();     //Getch fonksiyonu gbi basılan tuşu okur
        }
    }
}
```

Programımızın çıktısı aşağıdaki gibi olacaktır.



ÖRNEK: Klavyeden girilen bir pozitif tamsayının faktöriyelini ekrana yazan programı yazınız . Yeni konsol projesi açarak gelen program penceresine aşağıdaki kodları yazalım. Açıklama satırlarına dikkat edelim.

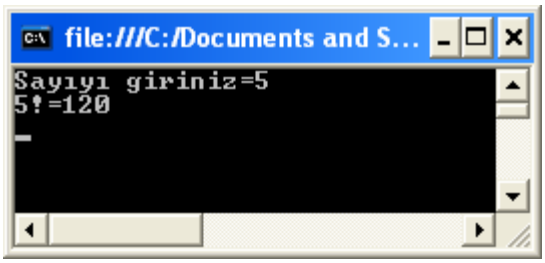
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Faktoriyel
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            long faktoriyel = 1;

            Console.Write("Sayıyı giriniz=");
            //Klavyeden girilen değer dizgi (string) olduğu için tamsayıya
            // dönüştürülerek atandı
            sayi = Convert.ToInt32(Console.ReadLine());
            if (sayi < 0)
            {
                Console.WriteLine("Negatif faktöriyel hesaplanamaz");
                Console.ReadKey();
```

```
                return ;
            }
            Console.Write(sayi + "!=");  // Alt satıra geçmez
            while (sayi > 0)
            {
                faktoriyel = faktoriyel * sayi;
                sayi--;
            }
            Console.WriteLine(faktoriyel);
            Console.ReadKey();
            return ;
        }
    }
}
```

Programın çıktıları aşağıdaki gibi olacaktır.



4 İlk Windows uygulaması

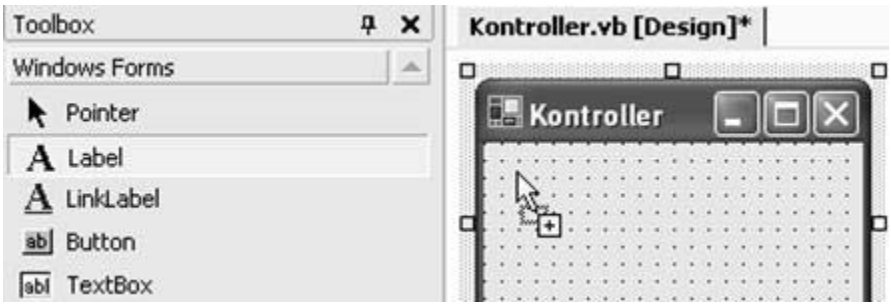
Tabii ki bu uygulamaların hepsini klasik C programlama dilinde de yapabiliyorduk. Programcıların C# programlama dilini kullanmasının en büyük sebebi görsel programlama (visual) olanaklarından faydalanabilmektir. Bu tür uygulamalar hakkında bilgi vermeden önce en çok kullanılan görsel programlama öğelerinden ve özelliklerinden bahsetmek yerinde olacaktır.

Windows ortamındaki kullanıcı etkileşimi ile oluşan olaylar aşağıdaki gibidir.

Click (Tık): Kontrol üzerine tıklandığı zaman tetiklenen olaydır. Windows tabanlı programlamada en sık kullanılan olaylardan biridir.
MouseDown (Mouse tuşu basıldığında)
Fare, kontrolün üzerindeyken herhangi bir tuşuna basıldığı zaman gerçekleşen olaydır. Bu olay, Click olayından önce çalışır.
MouseUp (Mouse tuşu bırakıldığında)
Fare, kontrolün üzerindeyken basılan tuş bırakıldığı zaman çalışır.
Enter (Girildiğinde)
Kontrol seçildiği veya üzerine odaklanıldığı zaman gerçekleşen olaydır.
Leave (Çıkıldığında)
Başka bir kontrol seçilmek üzere çıkıldığında, bu kontrolün Leave olayı tetiklenir.
VisibleChanged (Görünürlüğü değiştiğinde)
Kontrolün görünüp görünmediğini belirten Visible özelliği değiştiği zaman tetiklenir.

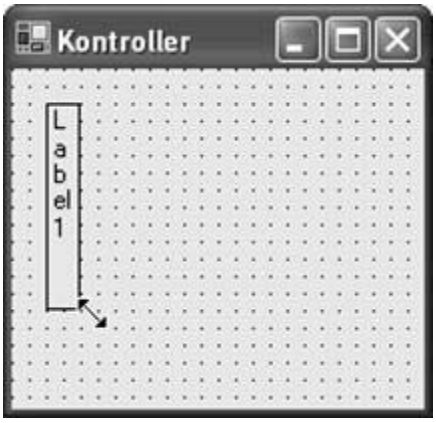
Windows tabanlı uygulamalar geliştirirken sıkça kullanacağımız bir grup nesne vardır. Form nesnesi

hariç diğer bütün kontroller Toolbox panelinden seçilir. Bu kontroller sürüklenip Form üzerine istenilen pozisyona bırakılır.



Kontroller, ToolBox panelinde üzerine çift tıklayarak da eklenebilir.

Kontrollerin tasarım anında büyüklükleri, yerleri Size ve Location özellikleri ile değiştirilebileceği gibi, fare ile de istenilen şekilde ayarlanabilir.



Form

Windows uygulamaları, Windows kontrollerinin tutulduğu pencereler olan formlardan oluşur. Bir Windows projesi açıldığı zaman Form kontrolü otomatik olarak eklenir. İkinci bir form eklemek için Project menüsünden Add Windows Form komutunu seçilir. Proje çalıştığı zaman başlangıç formu görüntülenir. Başlangıçta hangi formun açılacağını projenin özelliklerinden değiştirilir.

Visual Studio ortamında formlar, tasarım sayfası ve kod sayfası olmak üzere iki farklı sayfada görüntülenir. Tasarım sayfası, formun ve üzerindeki nesnelerin görünümlerini kolay bir şekilde değiştirmeyi sağlar. Visual Studio bu sayfada yapılan değişiklikleri kod sayfasında eş zamanlı olarak günceller. Örneğin bir Button kontrolünün genişliğini fare ile değiştirdiğimiz zaman, kod sayfasında bu kontrolün Width özelliği yapılan değişikliğe göre güncellenecektir. Aynı değişiklikler properties panelinde de görülebilir.

Formların, diğer kontrollerin özelliklerinden farklı bazı özellikleri vardır.
· **ControlBox** (Denetim Kutusu)
Form üzerindeki simge durumunda küçültme, ekranı kaplama ve formu kapama kutularının

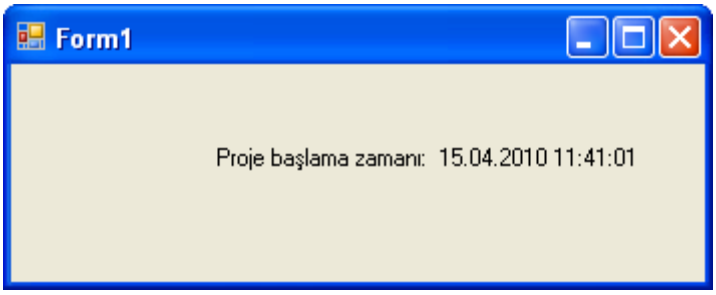
görünümünü ve erişebilirliğini kontrol eder.
NOT: Formun ControlBox özelliği False iken uygulama, Debug menüsünden Stop Debugging komutu seçilerek kapatılabilir.

· **StartPosition** (Başlagıç Pozisyonu)
Form açıldığı zaman nerede gözükeceğini belirler. CenterScreen seçeneği formu ekranın ortasında gösterir.

Formlar açıldığı zaman Load olayı gerçekleşir. Eğer form, başlangıç formu olarak seçilmişse, proje başladığı zaman çalıştırılmak istenen kodlar bu olayın yordamına yazılır.

```
private void Form1_Load(object sender, EventArgs e)
{
    label1.Text = "Proje başlama zamanı:  " + DateTime.Now;
}
```

Çıktısı aşağıdaki gibi olacaktır.



Button

Bir Windows düğmesini temsil eder. Button kontrolüne basıldığında Click olayı tetiklenir. Bu olay gerçekleştiği zaman yapılacak işlemler, ButtonIsmi_Click yordamında yazılır.

```
private void button1_Click(object sender, EventArgs e)
{
   button1.Top += 10; // Her bastığında düğme aşağı iner
}
```

Çıktısı aşağıdaki gibi olacaktır.

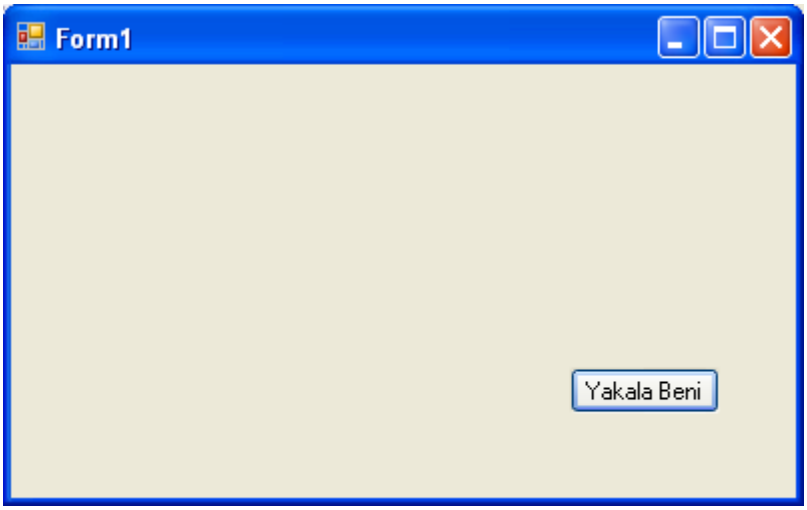
Gerçekleştirilen olayı değiştirmek için özellikler (properties) penceresinde üzerinde sarı şimşek olan şekle basarak o nesne ile kullanılabilecek olan tüm olaylar gözükecektir.

Aşağıdakine benzer bir kod yazıldığında butonun üzerine fare geldiği anda düğmenin yeri rastgele olarak değişmektedir.

```
private void btnYakala_MouseHover(object sender, EventArgs e)
{
   Random r= new Random(); //Rastgele sayı nesnesi

   btnYakala.Left = r.Next(1, 350);
   // 1 ile 350 arasında rastgele sayı üret
   btnYakala.Top = r.Next(1, 200);
   // 1 ile 200 arasında rastgele sayı üret
}
```

Programın çıktısı aşağıdaki gibi olacaktır.



TextBox

Bir Windows metin kutusunu temsil eder. Kullanıcıların değer girerek program ile haberleşmesini sağlamak amacıyla kullanılır. TextBox kontrolündeki yazı değiştiği zaman TextChanged olayı gerçekleşir.

```
private void textBox1_TextChanged(object sender, System.EventArgs e)
{
    // TextBox içindeki yazı değiştiği zaman
    // aşağıdaki kod çalışır.
    MessageBox.Show("Yazı değiştirildi: " + textBox1.Text);
}
```

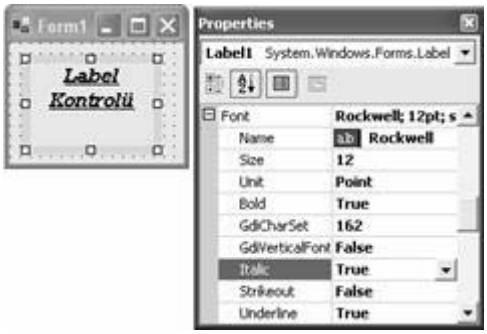
Label

Bir Windows etiketini temsil eder. Kullanıcıya, form üzerinde bir yazıyı göstermek amaçlı kullanılır. Bu yazının görünümü, Label kontrolünün bazı özellikleri ile değiştirilir.

· **TextAlign** (Yazı Hizalama)

Yazının Label kontrolü üzerinde nerede duracağını belirler.

· **Font** (Yazı Tipi)
Font özelliği birçok alt özellik taşır. Bunlardan bazıları en sık kullanılan özelliklerdir.
Name
Yazı tipinin ismini belirler. Varsayılan Microsoft Sans Serif seçilidir.
Size
Karakterlerin boyutunu belirler. Varsayılan büyüklük 8,5 değerini alır.
Bold (Kalın)
Yazının kalın tipte olmasını belirler.
Italic (Yatay)
Yazının italik tipte olmasını belirler.
UnderLine (Altı Çizgili)
Yazının altı çizgili olmasını belirler.



ComboBox

Bir Windows açılan kutusunu temsil eder. ComboBox kontrolü, kullanıcıların bazı değerleri açılan bir listeden seçmesini sağlar. Listeye tasarım anında veya çalışma anında öğe eklenebilir. Listeye öğe eklemek için kontrolün Items özelliğinden faydalanılır.
Tasarım anında öğe eklemek için Properties panelinden Items özelliği seçilir. String Collection Editor penceresinde, her öğenin değeri tek bir satırda yazılır.

Çalışma anında öğe eklemek için kod sayfasında, kontrolün Items özelliğinin Add metodu kullanılır.

```
private void Form1_Load(object sender, System.EventArgs e)
{
   comboBox1.Items.Add("Lise");
   comboBox1.Items.Add("Üniversite");
   comboBox1.Items.Add("Yüksek Lisans");
   comboBox1.Items.Add("Doktora");
}
```

ListBox

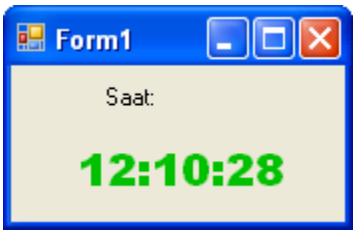
Bir Windows liste kutusunu temsil eder. Kontroldeki öğeler sabit bir liste olarak görüntülenir. ListBox kontrolüne öğe ekleme işlemi, ComboBox kontrolündeki işlemlere ile aynıdır. ComboBox kontrolünden farkı, birden fazla öğe seçilebilir olmasıdır.

Timer

Bir Windows sayacını temsil eder. Sayaç çalışmaya başladığı zaman, belirli zaman aralıklarında Tick olayı gerçekleşir. Timer kontrolünün Interval değeri, Tick olayının kaç milisaniyede bir gerçekleşeceğini belirler. Örneğin Interval değeri 2000 olan bir sayaç, Tick olayında yazılan kodları iki saniyede bir çalıştıracaktır.
Sayacı başlatmak için kontrolün Start metodu, durdurmak için ise Stop metodu kullanılır. Enabled özelliği, sayacın aktif olup olmadığını belirler.

```
private void Form1_Load(object sender, EventArgs e)
{
   timer1.Interval = 1000; // Her bir saniyede tick olayı çalışsın
   timer1.Start(); // Timer ı başlat
   label2.Text = DateTime.Now.ToLongTimeString();
   // Form açılır açılmaz zamanı bas
}
 private void timer1_Tick(object sender, EventArgs e)
{
   label2.Text = DateTime.Now.ToLongTimeString();
}
```

Ekran çıktısı aşağıdaki gibi olacaktır.



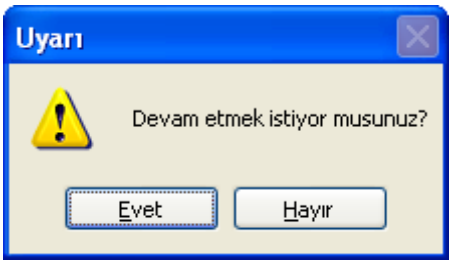
MessageBox

MessageBox, kullanıcıya bilgi göstermek için açılan mesaj kutusudur. Bu mesaj kutusu dört öğeden oluşur.

- **Text** (Yazı): Mesaj kutusunda verilmek istenen bilgiyi tutan yazıdır
- **Caption** (Başlık): Mesaj kutusunun başlığıdır
- **Buttons** (Düğmeler): Mesaj kutusunda hangi düğmelerin gösterileceğini belirler.
- **Icon** (Simge): Mesaj kutusunda gösterilecek olan simgeyi ve açıldığı zaman çıkartılacak sesi belirler.

```
MessageBox.Show("Devam etmek istiyor musunuz?", "Uyarı",
                MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
```

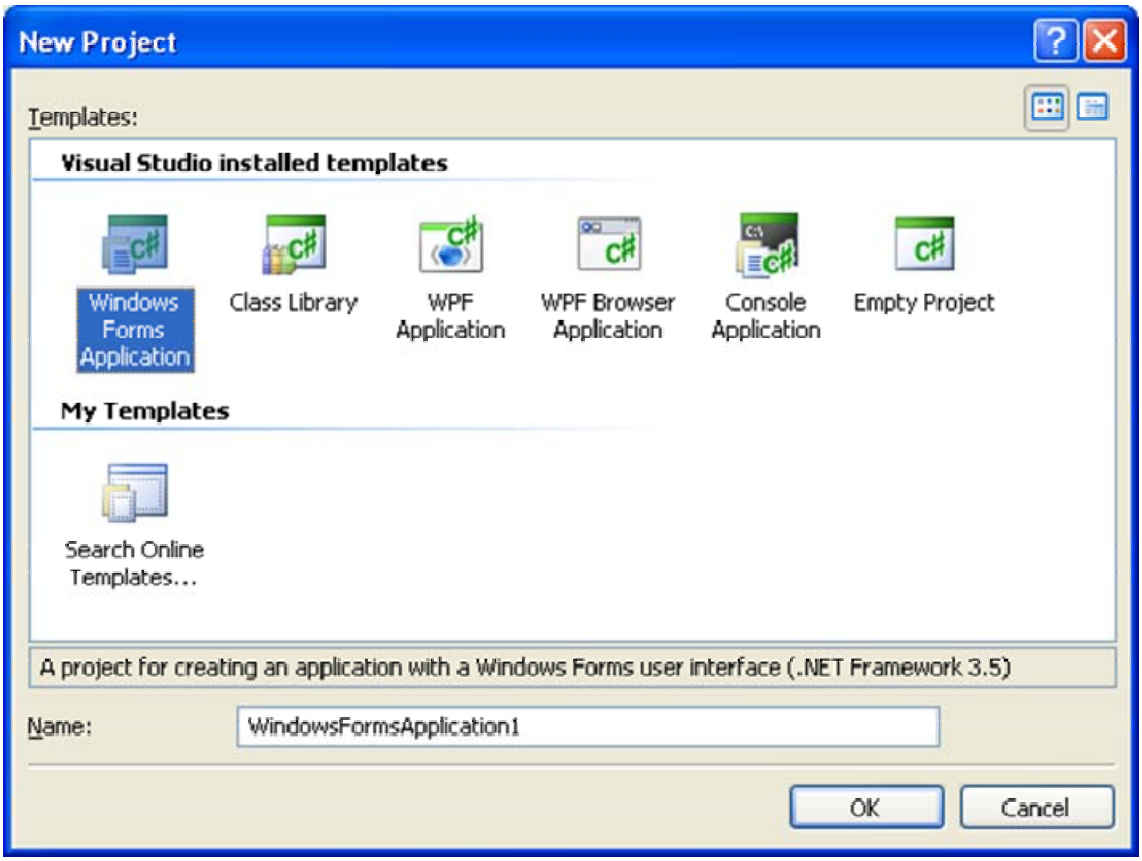
Komut çalıştığında çıktısı aşağıdaki gibi olacaktır.



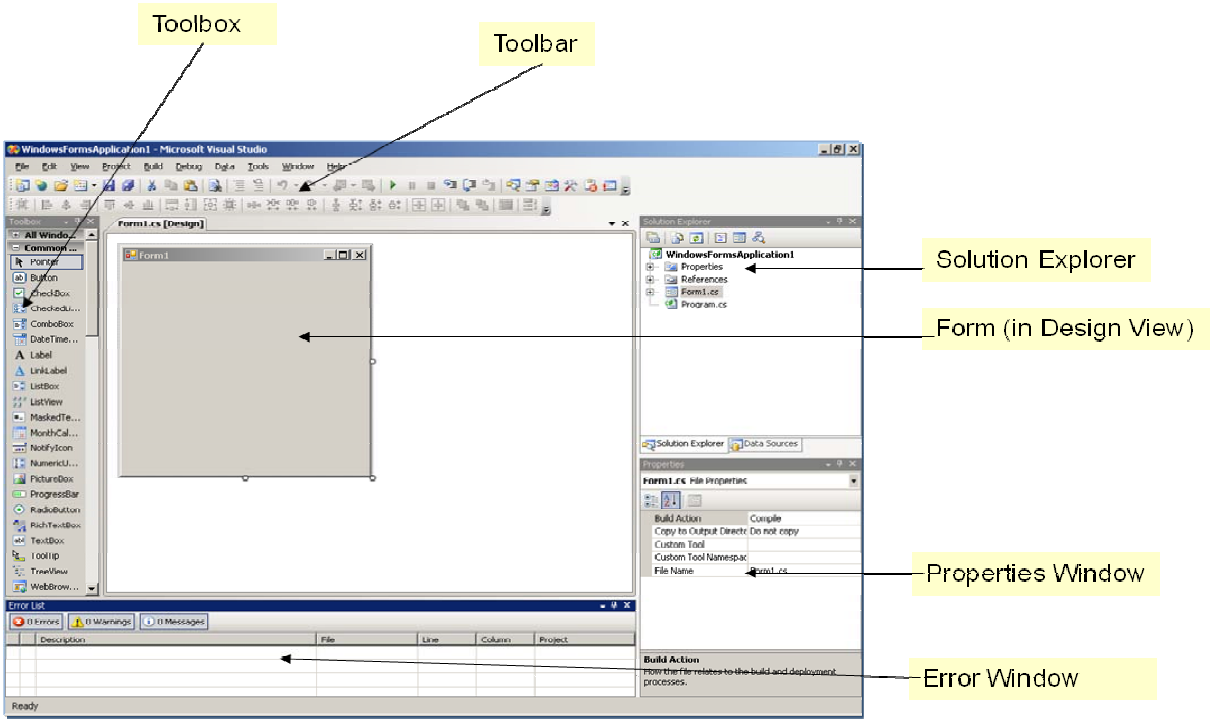
Mesaj kutusu, kapanırken hangi düğmenin basıldığını DialogResult nesnesi ile programcıya bildirir.

```
if (MessageBox.Show("Değişiklikler kaydedilsin mi?", "Kayıt",
                MessageBoxButtons.YesNoCancel) == DialogResult.Cancel)
{
    // İptal tuşuna basıldığı zaman
    // buraya girilir.
}
```

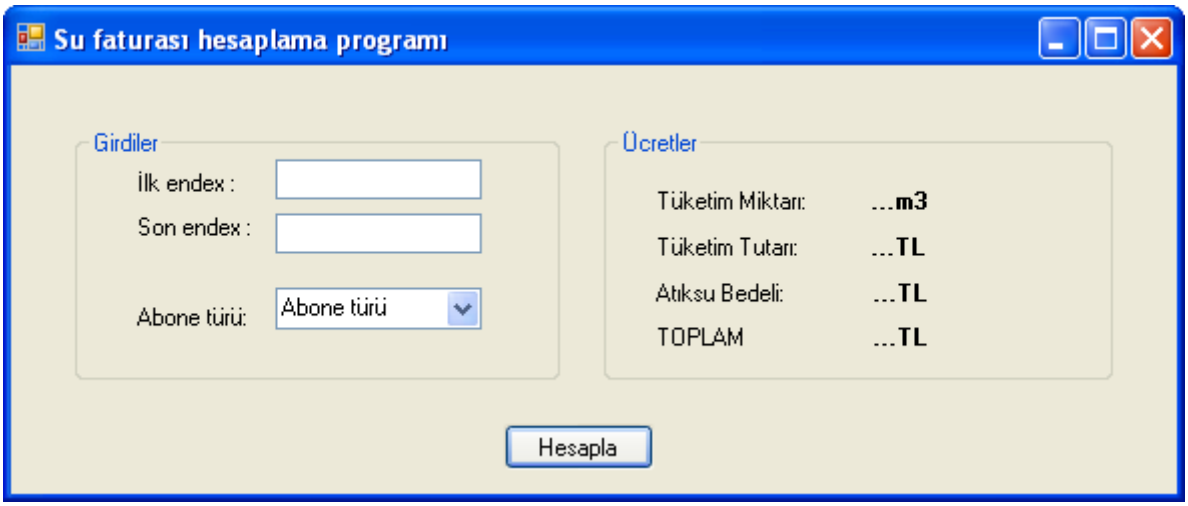
ÖRNEK : Girilen ilk gösterge ve son gösterge bilgilerine göre ödenecek su faturası ücretini hesaplayan bir program yazınız . Su bedeli evler için m3 başına 0.91827 TL, dükkan için 1.1827 TL olarak hesaplanacaktır. Öncelikle File->New Project mönü seçeneğinden Windows application seçeneğini işaretleyiniz.



Gelen arayüz aşağıdaki gibi olacaktır.



Formunuzun üstüne yukarıdaki tasarıma uyacak şekilde aşağıdaki isimlerde nesnelerinizi yerleştirin. Bu nesnelerin isimlerini anlamlı bir şekilde vermek kodun okunabilirliğini arttıracaktır. Burada en çok bilinen yöntem Macar (Hungarian) notasyonudur. Bu yöntemde ilk üç harf nesnenin ne olduğunu gösterir diğer kelimeler o nesnenin ismidir. Örnek : **`txtAtikSu`** (Atık su miktarının yazılacağı metin kutusu)




```
Label lblIlkendeks
Label lblSonEndeks
TextBox txtIlkEndeks
TextBox txtSonEndeks
Label lblSayacTip
ComboBox cmbAbone
Button btnHesapla
GroupBox groupBox1
```

```
Label lblTuketimMiktari
Label lblAtikSuBedeli
Label lblTuketimTutari
Label lblToplam
Label lblAtikSu
Label lblTutar
Label lblMiktar
Label lblToplamDeger
GroupBox groupBox2
```

Elbetteki tüm programlama dilinin özelliklerin tam olarak anlatmak mümkün değildir. Ancak en sık kullanılan özelliklerine değinilmiştir. Bundan sonrası sizin ilgi ve merakınıza kalmıştır.

Aşağıdaki kodda hesapla düğmesinin tıklama (Click) olayına yazılır.

```
int tuketim;
double tutar=0;
double atiksu;
double toplam;

if (txtIlkEndeks.Text == "")
{
   MessageBox.Show("İlk endeks bilgisini girmediniz","Hata");
   txtIlkEndeks.Focus();  // İlk endeks kutusuna imleci götürür
}
else if (txtSonEndeks.Text == "")
{
   MessageBox.Show("Son endeks bilgisini girmediniz");
   txtSonEndeks.Focus(); // Son endeks kutusuna imleci götürür
}
else if (Int32.Parse(txtSonEndeks.Text) < Int32.Parse(txtIlkEndeks.Text))
{
   MessageBox.Show("Son endeks ilk endeks ten küçük olamaz");
   txtSonEndeks.Focus(); // Son endeks kutusuna imleci götürür
}
else if (Int32.Parse(txtSonEndeks.Text) < Int32.Parse(txtIlkEndeks.Text))
{
   MessageBox.Show("Son endeks ilk endeks ten küçük olamaz");
   txtSonEndeks.Focus(); // Son endeks kutusuna imleci götürür
}
else if (cmbAbone.Text == "Abone türü")
{
   MessageBox.Show("Abone türünü seçmediniz");
   cmbAbone.Focus(); // Abone türü kutusuna imleci götürür
}
else
{
   tuketim = Int32.Parse(txtSonEndeks.Text) -
Int32.Parse(txtIlkEndeks.Text);
  // Metin kutularından gelen veriler tamsayıya dönmüştürülmelidir

  lblMiktar.Text = tuketim.ToString() + " m3";

   switch (cmbAbone.Text)
   {
      case "Ev":
         tutar = tuketim * 0.91827;
         break;
      case "Dükkan":
         tutar = tuketim * 1.1827;
         break;
      }
      lblTutar.Text = tutar.ToString() + " TL";
      atiksu = tutar / 2;
      lblAtikSu.Text = atiksu.ToString() + " TL";

      toplam = tutar + atiksu;
      lblToplam.Text = toplam.ToString() + " TL";
    }
}
```