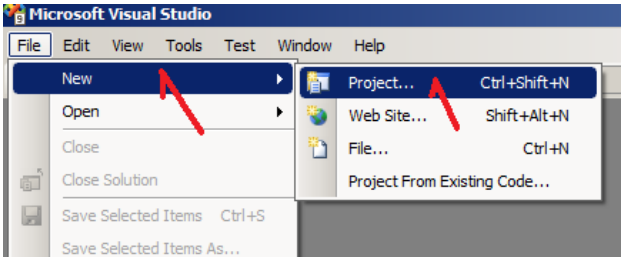
ASP.NET CORE 8.0 MVC DERS NOTLARI-1. HAFTA

İçindekiler

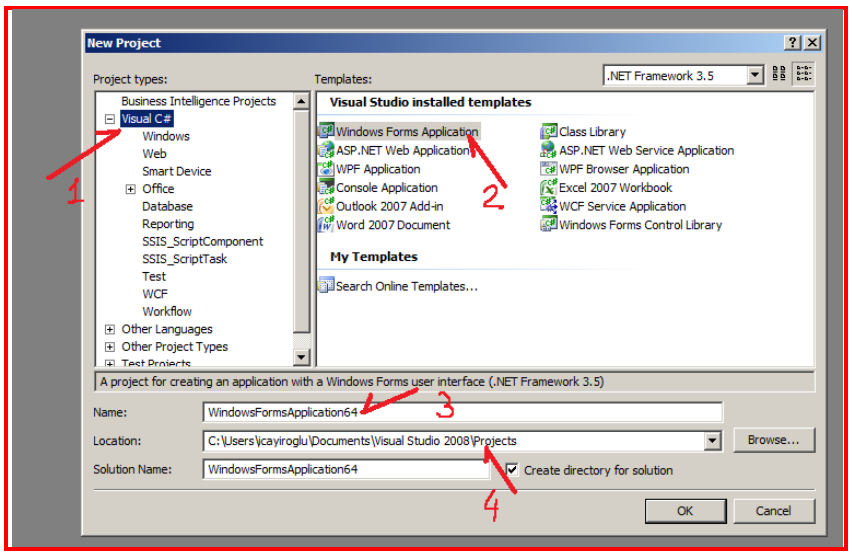
C# PROGRAMLAMANIN TEMELLERİ

VISUAL STUDIO'DA YENİ BİR PROJE OLUŞTURMA

Visual Studio (VS) programını çalıştırdığımızda karşımıza boş bir ekran gelir. Yeni bir proje oluştururken File>New>Project yolu kullanılarak yeni bir proje başlatırız.

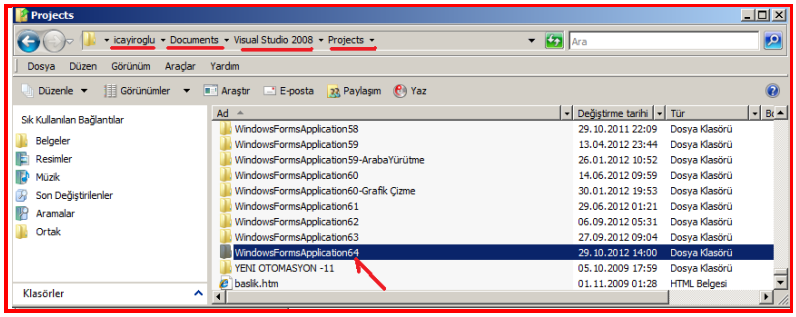


Karşımıza çıkan ekrandan C# dili seçili olmalı (1 nolu). Programımız masaüstü bir program olacağından ve Windows ortamında çalışan bir program olacağından "Window Forms Application" seçili olmalı. Projenin adı 3 nolu yerde gösterilen addır. Bu projenin bilgisayarımızda nerede kayıtlı olacağını gösteren yer ise 4 nolu yerdir. VS yi kapattıktan sonra hazırladığımız programı başka bir yere taşımak istiyorsak 4 nolu yere gidip orada 3 numara ile gösterilen ismin bulunduğu klasörü alıp kopyalayabiliriz.

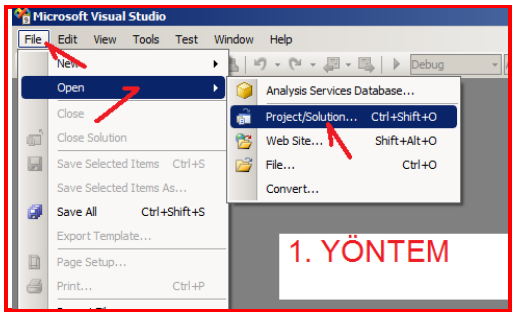


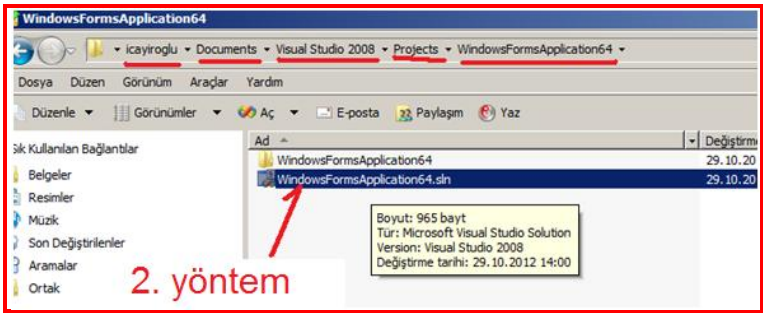
Ok dügmesine bastıktan sonra VS bize projemizin temelini oluşturan ilk yapıyı hazır olarak verecektir. Bunlarla ilgili ilk kodları projemizin içerisine atar ve bilgisayarımızdaki 4 numara ile gösterilen adresteki yere dosyaları kopyalar.

Şimdi bilgisayarımızdaki bu dosyaları görelim. Göreceğimiz gibi 64 numaralı proje oluşturulmuş durumda. Aynı proje VS içerisinde de şuan açık durumdadır. Biz VS içerisinde projenin dosyaları arasında gezerken VS nin kendi Gezgin (explorer) penceresini kullanırız. Buna "Solution Explorer" penceresi diyoruz.



Projemizde epey bir çalışma yaptığımızı düşünelim. Ertesi günü tekrar projemizi açmak istersek iki yolu kullanabiliriz. Ya VS nin içerisinden File>Open>Project yolunu kullanıp buradaki 64 numaralı projeyi açarız. Yada Windows'un kendi gezgin penceresinden gidip 64 nolu projenin ana dosyasına çift tıklayıp VS ile birlikte projenin açılmasını sağlayabiliriz.

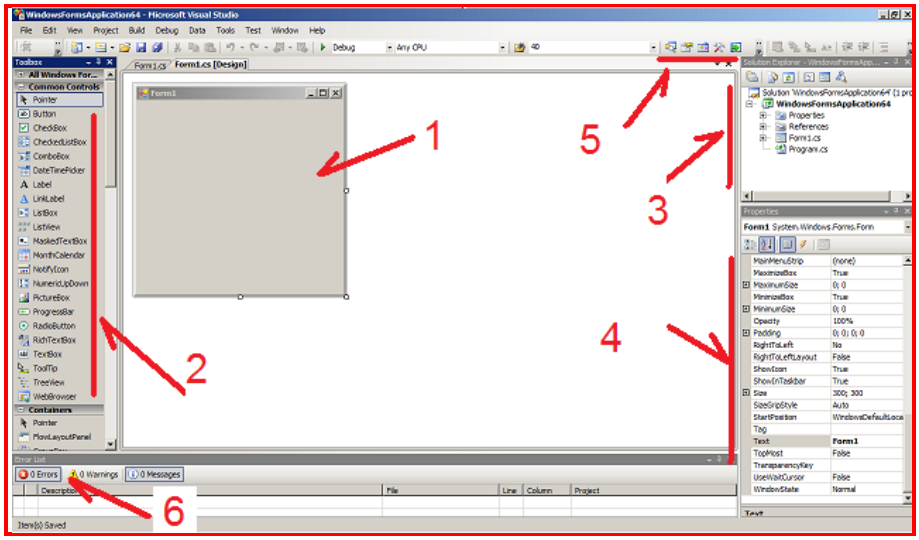




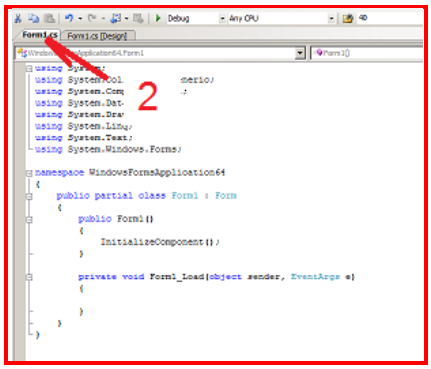
İlk projemizi New deyip oluşturduktan sonra karşımıza aşağıdaki gibi bir ekran gelecektir. Bu ekranda bize ilk olarak hazır bir nesne diyebileceğimiz Form nesnesi verilmiştir. Bu nesne programımızın zeminin oluşturan düz bir çerçevedir (1 nolu). Her nesnenin Özellikleri (properties) ve üzerinde gerçekleştirilebilecek olayları (events) vardır demiştik. Örneğin Form nesnesinin zemin rengini farklı bir renkte yapmak istiyoruz. Bunu nereden ayarlayabiliriz. Sağ taraftaki Properties penceresinden bu Bunu BackColor (arka renk) kısmında değiştirebiliriz. İşte bunun gibi projemize eklediğimiz hazır nesnelerin birçok özelliğini sağ taraftaki Preoperties penceresinde (4 nolu) ayarlayabiliriz.

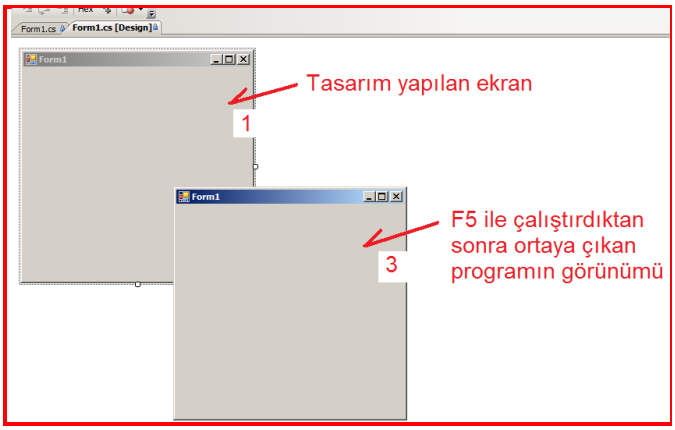
Dikkat edersek buradaki projemizde sadece bir tane hazır nesne (Form nesnesi) bulunmaktadır. Eğer istersek bu formun üzerine daha birçok hazır nesneleri ekleyebiliriz. Bu iş için sol taraftaki Toolbox (araçkutusu) (2 nolu) penceresini kullanabiliriz. Örneğin formun üzerine Textbox (yazı yazma kutucukları) button (buton) label (etiket) gibi daha bir çok nesneyi sol taraftan sürükleyip formun üzerine getirebiliriz.

Dikkat edersek ekranımızda birçok panel denilen pencereler bulunmaktadır. Bunları tekrar açıklarsak Toolbox hazır nesnelerin bulunduğu panel, Solution Explorer projemizin içinde bulunan dosyaları, Properties ise seçili olan nesnenin özelliklerini değiştirmeyi sağlayan kısımdır. Kodlarda bir hata var ise bu hatanın nerede olduğunu gösteren 6 nolu kısımda Errors panelimiz bulunmaktadır. İşte bütün bu paneller eğer ekranımızda gözükmeyip kaybolmuş ise 5 numara ile gösterilen yerdeki düğmeler tıklarsak ortaya çıkacaktır.



Projemizi hazırlarken üç tane ekran bizim için önemlidir. Bunlardan birincisi Design penceresi. Bu pencere projemizin nasıl görüneceğini bize gösterir (1 nolu) . Diğeri içeresine kod yazdığımız ekrandır. Bu ekranda C# kodlarını yazacağız (2 nolu). Birde Programı F5 ile çalıştırdıktan sonra Design penceresini çalışır halde gördüğümüz ekrandır (3 nolu).





C# da büyük küçük harf ayrımı vardır. Her satırın sonuna mutlaka ; işareti konulmalıdır.

DEĞİŞKENLER VE VERİ TİPLERİ

Verilerin tutulacağı değişkenlerin tanımlanması zorunludur. Değişken tanımlanırken hangi tip veri türü tutulacağı ve hangi aralıkta çalışacağına dikkat edilmelidir. Çalışacağı aralık içerisinde mümkün olduğunca en düşük hafıza tutan veri türünü tercih etmek gerekir. Değişkenlerin tanımlanması karmaşık program yapılarında bilgilerin karışmalarını engellemesi açısından ve en az ram kaynaklarını kullanmaya neden olduğu için kullanımı önemlidir. Değişken türleri aşağıdaki tabloda verilmiştir. Bir değişken hafızada tutmuş olduğu byte sayısı kadar bilgiyi tutabilir. Örneğin 2^8 = 1 byte = 256 kadar olan sayıları tutabilir. Yani 0 ile 256 arası sayıları tutabilir. Bu şekilde tanımlama işaretsiz (unsigned) tanımlama olur Eğer negatif bölgeye de geçiş yapılarsa bu sayı ikiye bölünür. -128 ile +128 arasında bilgiler tutulmuş olur. Bu tanımlama işaretli tanımlama olur.

Adı		**Hafıza (byte)**	**Sınır Değerleri**	**İşaretsiz (unsigned) Değerleri**		
Tamsayı	sbyte	1	-128 : + 127	byte	1	0 : + 255
	short	2	-32 768 : +32 767	ushort	2	0 : + 65 535
	int	4	-2 147 483 648 : + 2 147 483 648	uint	4	0 : +4 294 967 295

	long	8	-9 223 372 036 854 775 808 : +9 223 372 036 854 775 808	ulong	8	0 : +18 446 744 073 709 551 616
Ondalık	float	4	$\pm 3.6x10^{-38}$: $\pm 3.6x10^{+38}$ (tek duyarlık)	Yoktur		
	double	8	$\pm 1.8x10^{-308}$: $\pm 1.8x10^{+308}$ (çift duyarlık)	Yoktur		
	decimal	16	28 digit ondalık sayı tutar.	Yoktur		
metin	char	2	Unicode tipinde tek bir karakteri tutmak içindir. Tek tırnak içine yazılmalıdır (char sube='A') gibi			
	string	2x	Birden fazla karakteri tutmak içindir.			
	bool	1 (bit)	0 : 1 (false – true)			

Not: Ondalık sayıları tanımlarken üç tane tip tanımlıyoruz. Float daha küçük sayılar (10^38 gibi yine de büyük sayı). Bu sayıları tanımlarken sonun f harfi konulmalı. Yoksa double kabul eder. Double daha büyük sayıdır. (10^308 gibi çok büyük sayılar). Decimal ise daha hassas yani virgülden sonra hane sayısı 28 digite kadar giden bir sayı atarken kullanılır. Bunu tanımlarken sonuna m harfi konulmalıdır. Bir tanımlayı yaptığımızda üzerine gelip sağ tuşa tıklarsak değişkenin alabileceği üst ve alt sınırları sınıf kodlaı içinde görebiliriz.

```
int A = 10;
float B = 10.51f;
double C = 12.67;
decimal D = 13.78m;
```

```
public struct Single : IComparable, IFormattable, ICon
{
    public const Single MinValue = -3.40282347E+38F;
    public const Single Epsilon = 1.401298E-45F;
    public const Single MaxValue = 3.40282347E+38F;
    public const Single PositiveInfinity = 1F / 0F;
```

```
//     Represents the largest possible value of System.Decimal. This field is
//     and read-only.
public const Decimal MaxValue = 79228162514264337593543950335M;
public const Decimal MinValue = -79228162514264337593543950335M;
```

Aynı şekilde küçükten büyüğe doğru tam sayı tanımlamalarını da yazarsak

```
byte E = 123; //8 bit = 1 byte
short F = 12345; //16 bit = 2 byte
int G = 1234567890; //32 bit = 4 byte
long H = 1234567890123456789; //64 bit =8 byte
```

Tanmlamaların başına "u" harfini (unsigned=işaretsiz) getirsek eksi bölgede sayılar artı bölge üzerine eklenir ve 0 dan itibaren iki kat daha büyük sayıları tutabilir (detaylar için yukarıdaki tabloyu inceleyin).

Degişkenlerin Yaşam Süreleri (Geçerli oldukları aralıklar)

Degişkenlerin tanım aralıkları ve hafızada tuttukları yerin yanında yaşam süreleri yada geçerli oldukları bölge hakkında da bilgi sahibi olmamız gerekir. Buna göre tanımlanan değişkenler dört farklı şekilde açıklanabilir.

a) Local (yerel) değişkenler: Bu değişkenler sadece tanımlandıkları fonksiyon içinde geçerlidirler. Tanımlandıkları fonksiyon dışından ulaşılmak mümkün değildir. Fonksiyon çağrıldığında hafızada oluşturulurlar, fonksiyondan çıkıldığında ise tekrar hafızadan silinirler.

b) Global (genel) degişkenler: Tüm fonksiyonarın dışında tanımlanırlar. Dolayısı ile tüm fonksiyonlarda geçerli olurlar. Program çalışmaya başladığı anda hafızada yer alırlar ve program çalıştığı sürece hafızada kalırlar. Program sona erdiğinde hafızadan silinirler.

Doğru Değişken Yazımı

string 1isim10; (yanlış)
string isim10; (doğru)

string ad soyad; (yanlış)
string ad_soyad; (doğru)
string AdSoyad;
string true; (yanlış)

Not: 1 byte = 8 bit = 2^8 = 256 demektir. Eğer işaretsiz ise 0-255 arasındaki sayıları tutacak demektir. 2 byte= $(2^8)^2$ =2^{16} = 65536 demektir. Eğer işaretsiz ise 0-65536 arasındaki sayıları tutar. Şayet işaretli ise -32000 ile + 32000 küsür sayılar arasındaki rakamları tutar

4 byte =$(2^8)^4$ =4.294.967.296 sayısına karşılık gelir.

Nullable (null değer alabilir): String ve Char tipi değişkenler içerisine null (boş) alabilir. Yani başlangıçta herhangi bir şey atamazsak çalışır. Fakat sayısal ifadeler için başlangıçta içerisine 0 değerini atayabiliriz. Fakat yinede 0 da bir sayı olduğu için bunu atamak istemeyebiliriz. Bu durumda null değerini atmamız gerekir ama sayısal değişkenlerde bu hata verir. Bu değişkenlerin null değerini alabilmesi için nullable yapılması gerekir. Tanımın sonuna bir ? soru işareti konulmalı.

```
byte E = null;
short F = null;
int G = null;
long H = null;

float B = null;
double C = null;
decimal D = null;

char L = null;

string K = null;
```

```
byte? E = null;
short? F = null;
int? G = null;
long? H = null;

float? B = null;
double? C = null;
decimal? D = null;

char? L = null;

string K = null;
```

Bool gibi bir değişkeni cinsiyet için kullandığımızda true/false ifadelerini, erkek/kadın şeklinde değerlendirebiliriz. Eğer nullable yaparsak bu sefer cinsiyetini belirtmek istemiyor şeklinde de kullanabiliriz. Yani üç tane seçenek haline gelmiş olur.

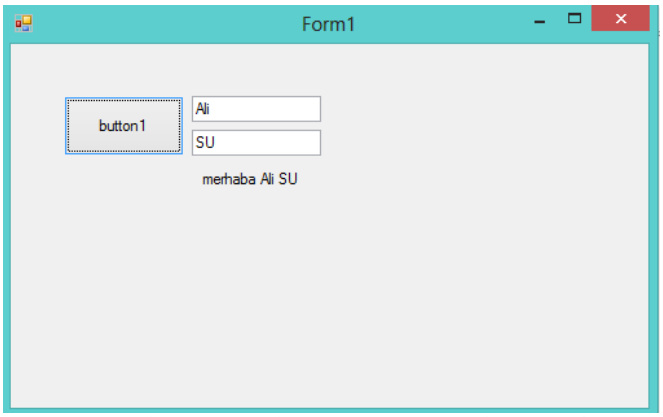
MATEMATİK (Math) KÜTÜPHANESİ

Matematik hesaplamalarında kullanılanılabilecek önemli fonksiyonlar aşağıda belirtilmiştir.

```
Math.E;          // e sayısını verir
Math.PI;         // pi sayısını verir
Math.Sin(b);     // b sayısının sin değerini alır
Math.Cos(b);     // b sayısının Cos değerini alır
Math.Tan(b);     // b sayısının Tan değerini alır
Math.Exp(b);     // e^b demektir
Math.Pow(b,c);   // b^c demektir
Math.Sqrt(b);    // Karekök değerini alır  daha fazla kök için a^(2/3) , Math.Pow(a,(2/3))
Math.Ceiling(b); // Ondalık sayıyı üste yuvarlar, b=10.3  , 11 çıkar
Math.Floor(b);   // Ondalık sayıyı aşağıya yuvarlar, b=10.3  , 10 çıkar
Math.Round(b);   // En yakın tamsayıya yuvarlar, b=10.3 , 10 çıkar, b=10.7 den 11 olur. b=10.49864 sayısı ,
                 Dikkat b=10.5  sayısını 10 yuvarlar.

Math.Min(b,c);   //b ve c sayısından en küçük sayıyı verir. b=3 , c=4 ise sonuç 3 çıkar
Math.Max(b,c);   //iki sayıdan en büyük olanını döndürür.
Math.Abs(b);     // sayının mutlak değerini alır, yani tüm sayılar pozitif çıkar.
Math.Log10(b);   // b sayısının 10 tabana göre logaritmasını alır. b=100 ise sonuç 2 çıkar b=10^2 => 2 çıkar.
Math.Log(b);     // b sayısının In'ini almaktadır. e tabanına göre logaritmasını alır.
```

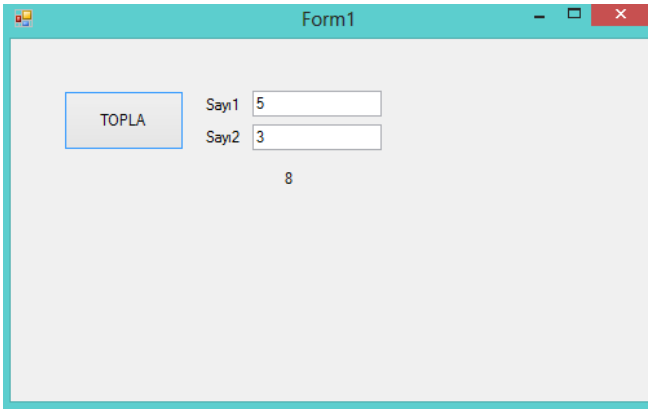
Math.Log(b,c); //c tabanında b sayısının logaritmasını alır. Örneğin b=8 ve c=2 ise sonuç 3 tür.

ÖRNEK



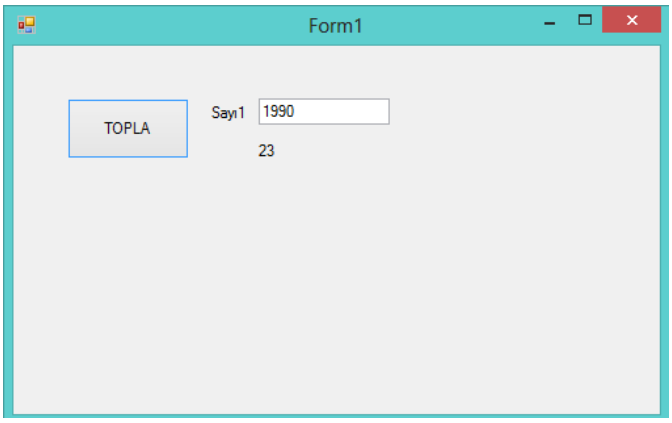

```
        private void button1_Click(object sender, EventArgs e)
        {
            string Ad, Soyad;
            int Yas;
            double Ortalama;

            Ad = textBox1.Text;
            Soyad = textBox2.Text;

            label1.Text = "merhaba " + Ad + " " + Soyad;

        }
```

ÖRNEK



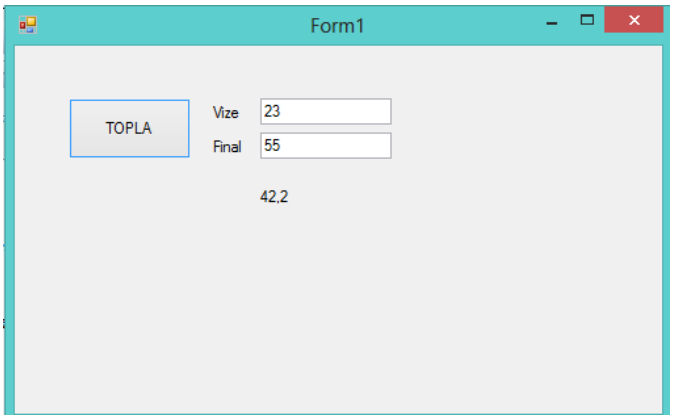

```
private void button1_Click(object sender, EventArgs e)
        {
            int Sayi1, Sayi2;

            Sayi1 = Convert.ToInt32(textBox1.Text);
            Sayi2 = Convert.ToInt32(textBox2.Text);

            label1.Text = (Sayi1 + Sayi2).ToString();

        }
```

ÖRNEK



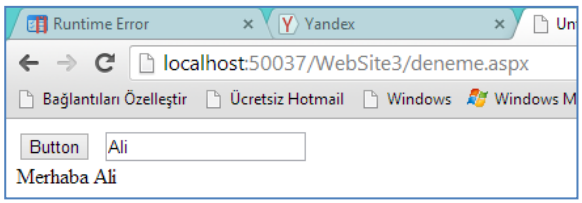

```
  private void button1_Click(object sender, EventArgs e)
        {
            int DogumTarihi, Yas;

            DogumTarihi = Convert.ToInt32(textBox1.Text);
            Yas = 2013 - DogumTarihi;

            label1.Text = Yas.ToString();

        }
```

ÖRNEK



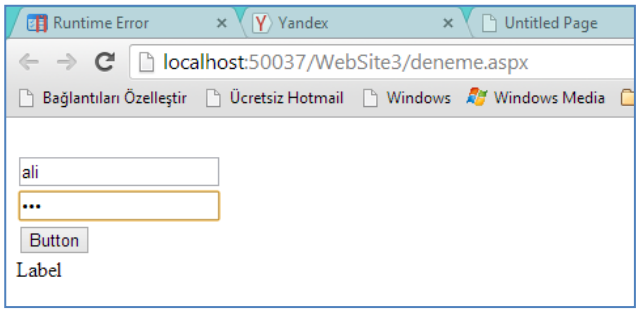

```
private void button1_Click(object sender, EventArgs e)
        {
            int Vize, Final;
            double Ortalama;

            Vize = Convert.ToInt32(textBox1.Text);
            Final = Convert.ToInt32(textBox2.Text);


            Ortalama = Vize * 0.40 + Final * 0.60;

            label1.Text = Ortalama.ToString();

        }
```

ÖRNEK



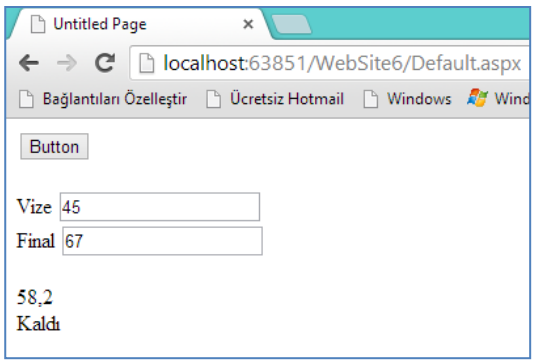

```
protected void Button1_Click(object sender, EventArgs e)
    {
        string Ad;
        Ad = TextBox1.Text;

        Label1.Text = "Merhaba " + Ad;
    }
```

ÖRNEK




```
protected void Button1_Click(object sender, EventArgs e)
    {
        string KullaniciAdi, Sifre;

        KullaniciAdi  = TextBox1.Text;
        Sifre = TextBox2.Text;

        if (KullaniciAdi == "ali" && Sifre == "123")
        {
            Label1.Text = "Hoş Geldiniz!..";
        }
        else
        {
            Label1.Text = "Şifre yanlıştır!..";
        }
    }
```

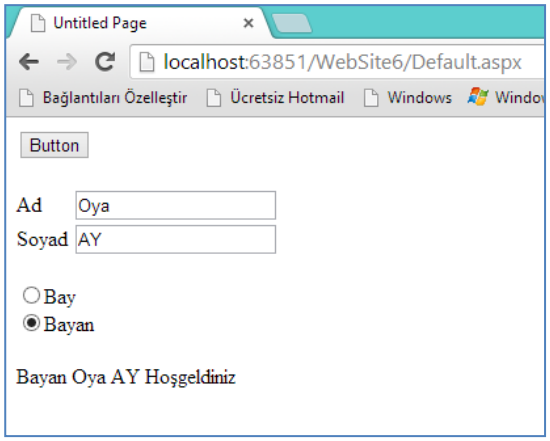
ÖRNEK




```
    protected void Button1_Click(object sender, EventArgs e)
    {
        double Vize, Final, Ortalama;
```

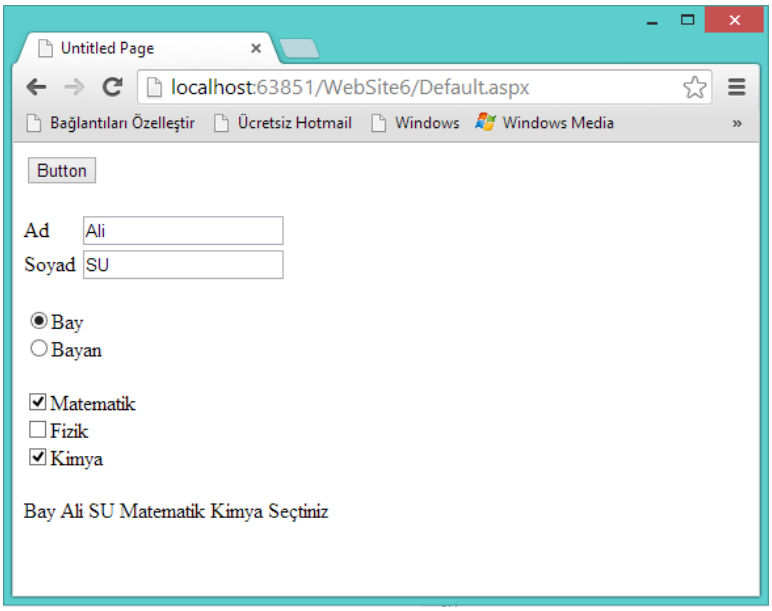
```
        Vize = Convert.ToDouble( TextBox1.Text);
        Final =Convert.ToDouble(  TextBox2.Text);

        Ortalama = Vize * 0.40 + Final * 0.6;

        Label1.Text = Ortalama.ToString();

        if (Ortalama >= 60 && Ortalama <= 100)
        {
            Label2.Text = "Geçti";
        }
        else if (Ortalama >= 0 && Ortalama < 60)
        {
            Label2.Text = "Kaldı";
        }
        else
        {
            Label2.Text = "HATALI NOT";
        }
    }
```

ÖRNEK



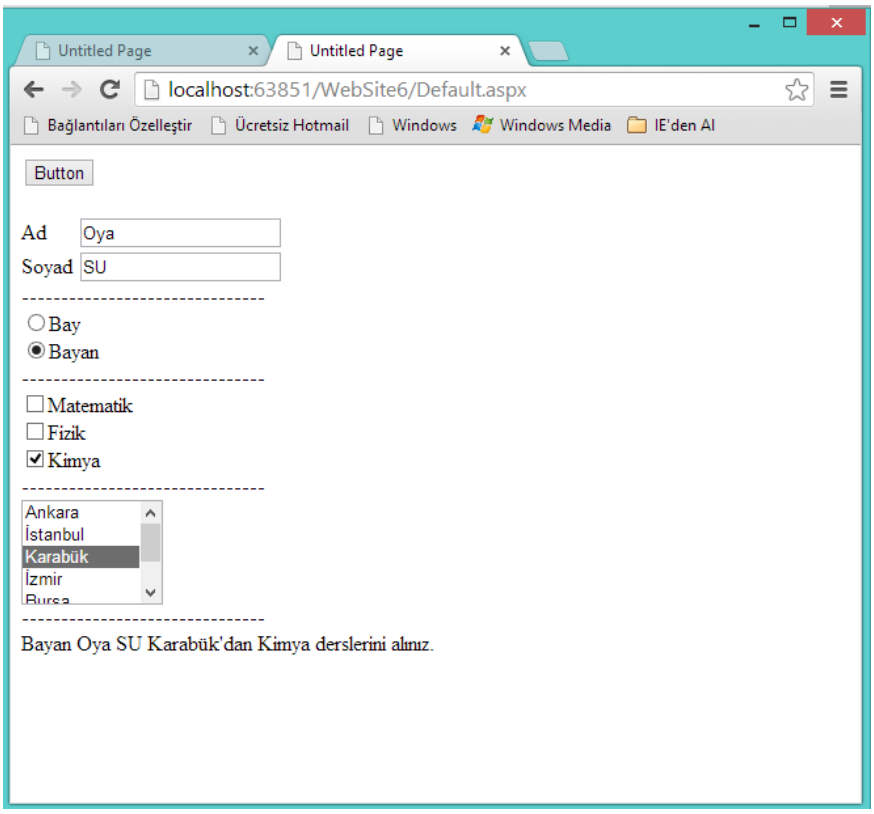

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        string  Ad, Soyad;
        string Cinsiyet =null;

        Ad = TextBox1.Text;
        Soyad =TextBox2.Text;

        if (RadioButton1.Checked ==true)
        {
            Cinsiyet = "Bay ";
        }
        else if (RadioButton2.Checked == true)
        {
            Cinsiyet = "Bayan ";
        }

        Label1.Text  = Cinsiyet + Ad + " " + Soyad + " Hoşgeldiniz";

    }
```

ÖRNEK




```
protected void Button1_Click(object sender, EventArgs e)
{
    string  Ad, Soyad;
    string Cinsiyet =null;
    string Dersler = null;

    Ad = TextBox1.Text;
    Soyad =TextBox2.Text;

    if (RadioButton1.Checked ==true)
    {
        Cinsiyet = "Bay ";
    }
    else if (RadioButton2.Checked == true)
    {
        Cinsiyet = "Bayan ";
    }

    if (CheckBox1.Checked == true)
    {
        Dersler =Dersler +  " Matematik ";
    }
    if (CheckBox2.Checked == true)
    {
        Dersler =Dersler + " Fizik ";
    }
    if (CheckBox3.Checked == true)
    {
        Dersler =Dersler + " Kimya ";
    }

    Label1.Text  = Cinsiyet + Ad + " " + Soyad +  Dersler + " Seçtiniz" ;
}
```

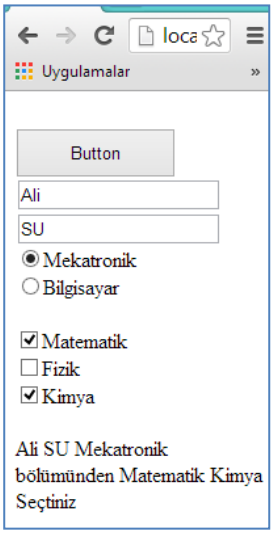
ÖRNEK




```
protected void Button1_Click(object sender, EventArgs e)
    {
        string  Ad, Soyad;
        string Cinsiyet =null;
        string Dersler = null;
        string Sehir = null;

        Ad = TextBox1.Text;
        Soyad =TextBox2.Text;

        if (RadioButton1.Checked ==true)
        {
            Cinsiyet = "Bay ";
        }
        else if (RadioButton2.Checked == true)
        {
            Cinsiyet = "Bayan ";
        }

        if (CheckBox1.Checked == true)
        {
            Dersler =Dersler +  " Matematik ";
        }
        if (CheckBox2.Checked == true)
        {
            Dersler =Dersler + " Fizik ";
        }
        if (CheckBox3.Checked == true)
        {
            Dersler =Dersler + " Kimya ";
        }

        Sehir = ListBox1.SelectedItem.Text;
```

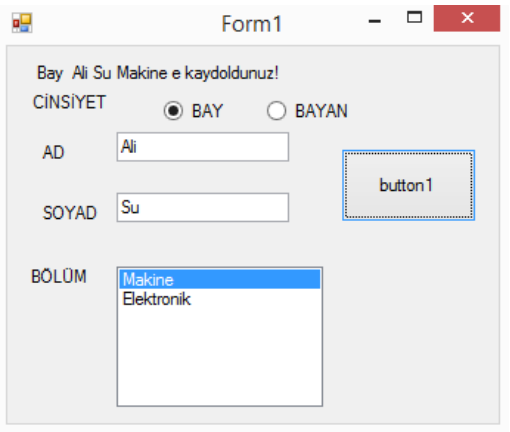
```
        Label1.Text = Cinsiyet + Ad + " " + Soyad + " " + Sehir + "'dan " + Dersler
+ " derslerini alınız.";

    }
```

ÖRNEK




```
protected void Button1_Click(object sender, EventArgs e)
    {
        string Ad, Soyad, Bolum=null, Dersler=null;

        Ad = TextBox1.Text;
        Soyad = TextBox2.Text;

        if (RadioButton1.Checked == true)
            Bolum = "Mekatronik ";
        else if (RadioButton2.Checked == true)
            Bolum = "Bilgisayar ";

        if (CheckBox1.Checked == true)
            Dersler = Dersler + " Matematik ";

        if (CheckBox2.Checked == true)
            Dersler = Dersler + " Fizik ";

        if (CheckBox3.Checked == true)
            Dersler = Dersler + " Kimya ";

        Label1.Text = Ad + " " + Soyad + " " +  Bolum + " bölümünden " + Dersler +
" Seçtiniz " ;

    }
```

ÖRNEK




```
string Cinsiyet = null;

if (radioButton1.Checked == true)
    Cinsiyet = "Bay ";
else if (radioButton2.Checked == true)
    Cinsiyet = "Bayan ";

string Ad = textBox1.Text;
string Soyad = textBox2.Text;

int SiraNo = listBox1.SelectedIndex;
string Bolum = listBox1.Items[SiraNo].ToString();

label4.Text = Cinsiyet + " " + Ad + " " + Soyad + " " +  Bolum  + " e kaydoldunuz!";
```

ÖRNEK

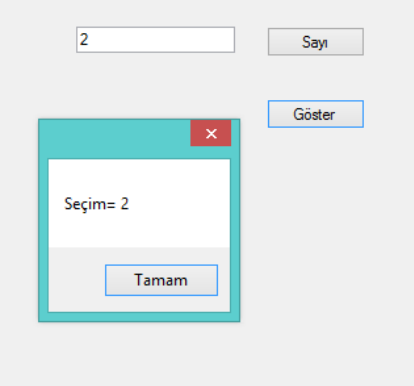

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication6
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int RastgeleSayi = 0;
        int sayac = 0;

        private void button1_Click(object sender, EventArgs e)
        {
            sayac = sayac + 1;

            int TahminSayisi =Convert.ToInt32(txtAd.Text);

            if (TahminSayisi > RastgeleSayi)
                label1.Text = " Aşağı";
            else if (TahminSayisi < RastgeleSayi)
                label1.Text = " Yukarı";
            else
                label1.Text = " Tebrikler" + sayac + "hakta bildiniz";
        }

          private void button2_Click(object sender, EventArgs e)
        {
            Random Rastgele = new Random();
            RastgeleSayi = Rastgele.Next(1, 100);

            label1.Text = "";
        }
    }
}
```

Switch Case Yapısı

Bu yapı If yapısı gibi koşullu durumlar için kullanılabilir. Burada koşulu sağlarken sadece eşit olma durumuna bakar. Büyük yada Küçük gibi koşulları kabul etmez. Değişken olarak kullanılacak ifadenin sadece integer olması gerekir. Yani blok yapıda kullanılacak değişkenini içeriği 1,2,3... gibi integer sayılar olmalıdır. Şu şekilde düşünebiliriz. Değişkenimizin içindeki değer 1 eşitse şu işlemi yap, 2 ye eşitse şu işlemi yap, gibi durumlar için kullanılır. Yapısı aşağıdaki şekildedir.

```
switch (Sayi)
{
   case 1:
      .....
         break;
   case 2:
      .....
         break;
   case 3:
      ....
         break;
   default:
      .....
         break;
}
```

Örnek:




```
int Sayi = 0;

private void button1_Click(object sender, EventArgs e)
{
   Sayi = Convert.ToInt32(textBox1.Text);
}

private void button2_Click(object sender, EventArgs e)
{
   switch (Sayi)
   {
      case 1:
         MessageBox.Show("Seçim= " + Sayi.ToString());
         break;
      case 2:
         MessageBox.Show("Seçim= " + Sayi.ToString());
         break;
```

```
            case 3:
                MessageBox.Show("Seçim= " + Sayi.ToString());
                break;
            default:
                MessageBox.Show("Geçersiz bir değer girdiniz" + Sayi.ToString());
                break;
        }
    }
```

Get ve Set Metodları

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

namespace set_get
{
    //dortgen clasının boy ve en özelliklerini tanımladık
    class dortgen {
        //private: oluşturulan değişkenlere sadece o class içinde ulaşılabilir.
        private int mBoy;
        private int mEn;

        //değişkene değer atama işlemleri
        //meodumuzu oluşturalımki ve private nesnelerimize değer atayabilelim
        public int En {
            get {
                return mEn;
            }

            //önce set metodu çalışacak. sonra get metodu ile değişken geri döndürülecek.
            set {
                mEn = value;
            }
        }

        //boy için metot oluşturalım
        public int boy{
            get {
                return mBoy;
            }
            set {
                mBoy = value;
            }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            //nesnemizi oluşturalım
            dortgen d1 = new dortgen();
```

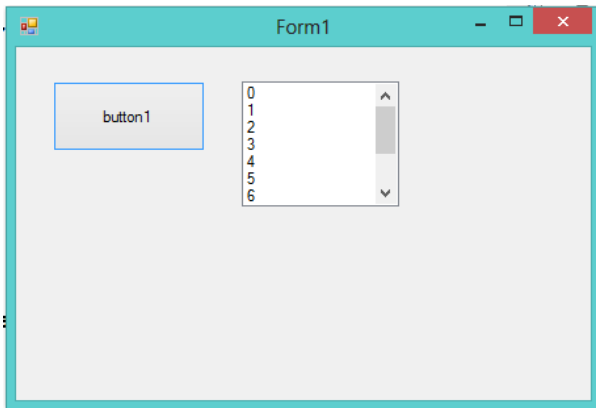
```
            //ve artık private olan değişkenlerimize değer atamaları yapabiliriz.. ve onlara istediğimiz gibi set ve get metodu ile ulaşabiliriz.
            d1.En = 52;
            d1.boy = 152;

            //ekrana yazdıralım
            Console.WriteLine("En : " + d1.En);
            Console.WriteLine("Boy : " + d1.boy);
            Console.ReadLine();
        }
    }
}
```

DÖNGÜLER (FOR, WHILE, DO-WHILE)

C# da döngüleri dört çeşit olarak sayabiliriz. Bunlar

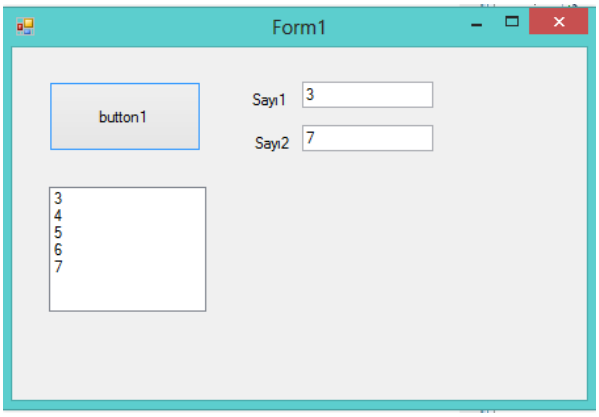
a) For döngüsü: Bu döngüde başlangıç ve bitiş sınırları verilerek kullanılır. Döngü tanım kısmında içinde bir de artan bir sayaç kullanılır.
b) While Döngüsü: Sonsuz döngüdür. Şart gerçekleştiği müddetçe döner. Döngüye daha girmeden şart başlangıçta kontrol edilir.
c) Do-While Döngüsü: Sonsuz döngüdür. Döngüye ilk girişte şart kontrol edilmez. Döngüden çıkarken şart kontrol edilir.
d) For-Each Döngüsü: Dizilerde kullanılan döngü tipidir. Dizi içinde kaç tane eleman varsa her biri için bir işlem yapacaksak bu döngü kullanılır. Bu döngü bir sonraki ders olan Diziler konusu içinde anlatılacaktır.

ÖRNEK



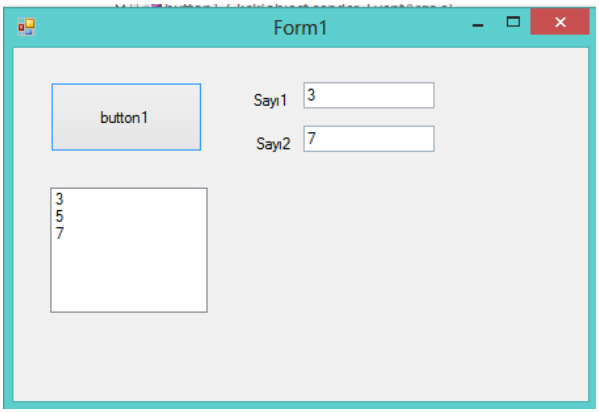

```
private void button1_Click(object sender, EventArgs e)
    {
        for (int i = 0; i <= 10; i++)
        {
            listBox1.Items.Add(i.ToString());

        }
    }
```

ÖRNEK



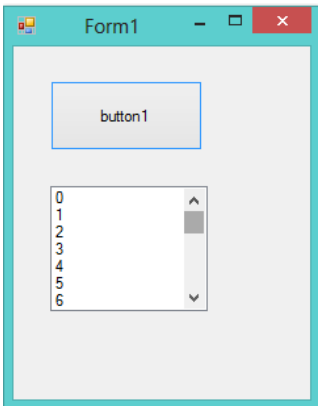

```
private void button1_Click(object sender, EventArgs e)
{
    int Sayi1, Sayi2;
    Sayi1 = Convert.ToInt32 (textBox1.Text);
    Sayi2 = Convert.ToInt32 (textBox2.Text);

    for (int i = Sayi1; i <= Sayi2; i++)
    {
        listBox1.Items.Add(i.ToString());

    }
}
```

ÖRNEK



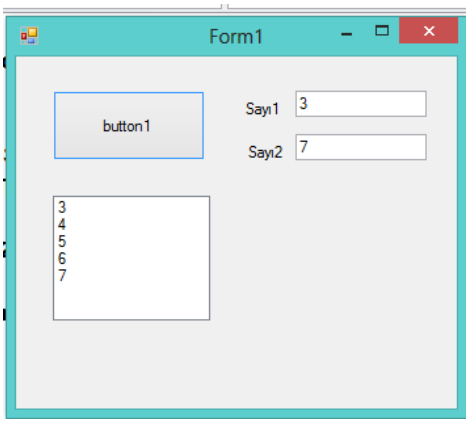

```
private void button1_Click(object sender, EventArgs e)
{
    int Sayi1, Sayi2;
    Sayi1 = Convert.ToInt32 (textBox1.Text);
    Sayi2 = Convert.ToInt32 (textBox2.Text);

    listBox1.Items.Clear();

    for (int i = Sayi1; i <= Sayi2; i = i + 2) // i++ => i=i+1   ,    i=i+2
    {
        listBox1.Items.Add(i.ToString());
    }
}
```

ÖRNEK



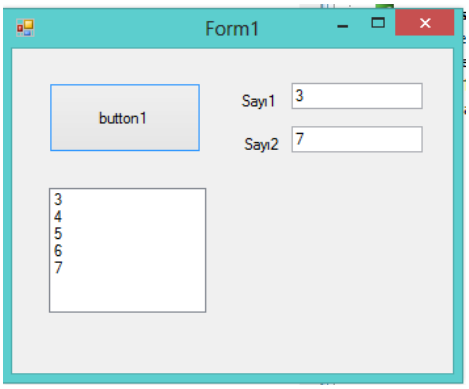

```
private void button1_Click(object sender, EventArgs e)
{
    int i=0;
    while (i <= 100)
    {
        listBox1.Items.Add(i.ToString());
        i++;
    }
}
```

ÖRNEK




```
private void button1_Click(object sender, EventArgs e)
    {
    int i=0, Sayi2;
    i = Convert.ToInt32(textBox1.Text);
    Sayi2 = Convert.ToInt32(textBox2.Text);

        while (i<=Sayi2)
        {
            listBox1.Items.Add(i.ToString());
            i++;
        }
    }
```

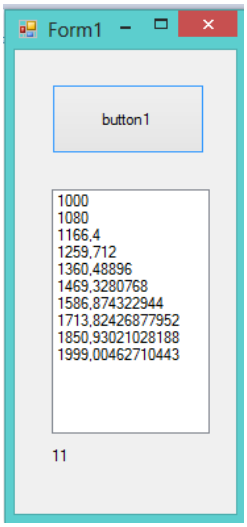
ÖRNEK




```
private void button1_Click(object sender, EventArgs e)
{
int i=0, Sayi2;
i = Convert.ToInt32(textBox1.Text);
Sayi2 = Convert.ToInt32(textBox2.Text);

do
{
    listBox1.Items.Add(i.ToString());
    i++;
} while (i <= Sayi2);

}
```

NOT : *Bir programı çalıştırırken değişkenlerin hangi değerleri aldığını görmek için F9 ile durmak istediğimiz yere bir çentik (durdurma işareti) atarız. Daha sonra F5 ile programı çalıştırdığımızda program oraya gelince duracaktır. Bu durumda mouse ile değişkenin üzerine geldiğimizde değişkenin içerisindeki değeri bize gösterecektir. Programı devam ettirmek için F11 ile adım adım devam edebiliriz. Yada sona kadar çalışmasını istersek F5 ile çalıştırabiliriz.*

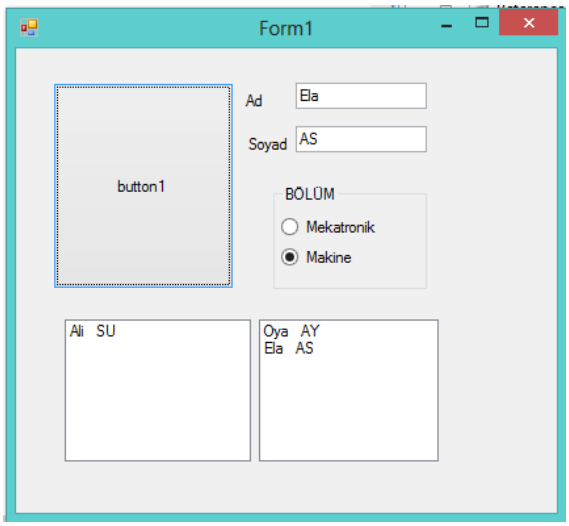
ÖRNEK:




```
private void button1_Click(object sender, EventArgs e)
    {
        double maas = 1000;
```

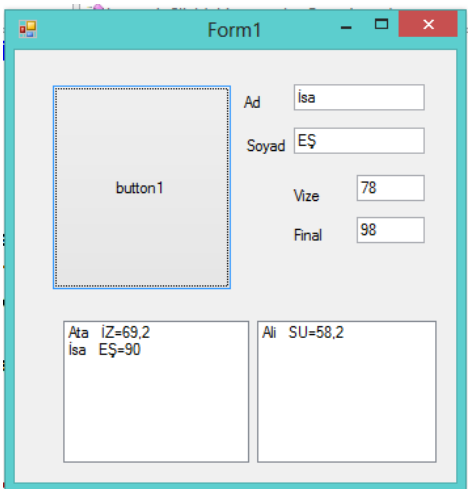
```
            int yil=1;

            while (maas <=2000)
            {
                listBox1.Items.Add(maas.ToString());
               yil++;
               maas = maas + maas * 0.08;
            }
            label3.Text = yil.ToString();

        }
```

ÖRNEK



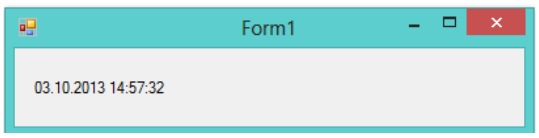

```
        private void button1_Click(object sender, EventArgs e)
         {
            string Ad, Soyad;
            Ad = textBox1.Text;
            Soyad = textBox2.Text;

            if (radioButton1.Checked == true)
            {
                listBox1.Items.Add(Ad + "    " + Soyad);
            }
            else if (radioButton2.Checked == true)
            {
                listBox2.Items.Add(Ad + "    " + Soyad);

            }
          }
```

ÖRNEK




```
private void button1_Click(object sender, EventArgs e)
{
    string Ad, Soyad;
    Ad = textBox1.Text;
    Soyad = textBox2.Text;

    double Vize, Final, Ortalama;
    Vize = Convert.ToDouble(txtVize.Text);
    Final = Convert.ToDouble(txtFinal.Text);

    Ortalama = Vize * 0.4 + Final * 0.6;

    if (Ortalama>=60)
    {
        listBox1.Items.Add(Ad + " " + Soyad + "=" + Ortalama);
    }
    else
    {
        listBox2.Items.Add(Ad + " " + Soyad + "=" + Ortalama);
    }
}
```

TARİH VE SAAT FONKSİYONLARI




```
private void Form1_Load(object sender, EventArgs e)
    {
        label7.Text = DateTime.Now.ToString();


    }
```

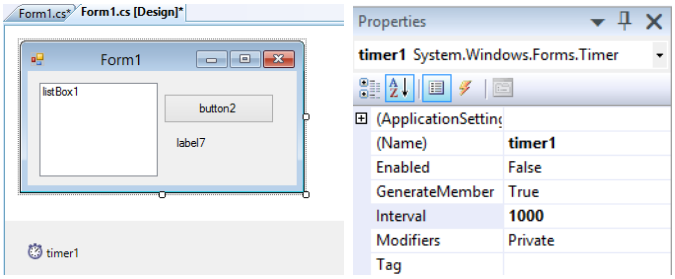
Örnek Kodlar

Aşağıdaki kodları deneyin yeni tarih saat formatları ve kullanımları bulun bunları deneyin.

```
DateTime Tarih = new DateTime(2008, 8, 23);
string YeniFormat = String.Format("{0:d}", Tarih);
Label1.Text = YeniFormat;

DateTime YeniTarih = DateTime.Today.Add(new TimeSpan(3, 12, 24)); //Yeni saat,dak,saniye.

string Dakika = DateTime.Now.Minute.ToString(); //Şimdiki zamanın daikası
string Saniye =  DateTime.Now.Second.ToString(); //Şimdiki zamanın saniyesi
```

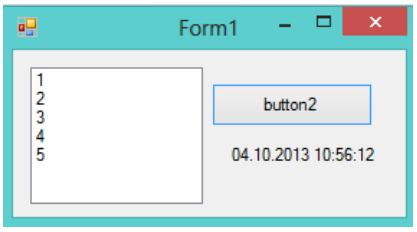
TIMER NESNESİ

Masaüstü program yazarken bilgisayarın belli aralıklar bir işi sürekli yapması istendiğinde bu nesne kullanılır. Nesnenin Properties penceresinden Interval özelliği süreyi belirler. Örneğin burada 1000 yazıyorsa her saniyede bir içerisindeki kodları çalıştıracak demektir. Bu nesne forma eklenirken arka planda kod olarak eklenir. Formün üzerinde gözükmez altındaki boş alanda kullanıcıya gösterilir.



Örnek:

Ekranda sistem saatini sürekli olarak her saniye göseren ve bu esnada bir listbox a geçen saniyeleri yazan bir program yazın.




```
int Sayac = 0; //global de tanımlanmalı

private void button2_Click(object sender, EventArgs e)
{
    timer1.Enabled = true;
}

private void timer1_Tick(object sender, EventArgs e)
{

    label7.Text = DateTime.Now.ToString();

    Sayac++;
    listBox1.Items.Add(Sayac.ToString());

}
```