C# ile Görsel Programlama-I

Yazar.

Muhammed Furkan Ateş

Kaynaklar

Kitaplar

- Her yönüyle C#, Sefer Algan, Pusula Yayıncılık
  Kitap içeriği: C# dilini yapısal olarak konsol uygulama tabanlı anlatmakta
- C# Programlama Dili ve Yazılım Tasarımı: Cilt-2, Ahmet Kaymaz
  Kitap içeriği: C# dilinin daha çok veritabanı uygulamalarına yönelik anlatımı
- Programming in Visual C#, Julia Case Bradley, Anita Millspaugh, 2008
  Kitap içeriği: C# dilini görsel /form uygulamaları ile anlatmakta
- **C# Programming: From Problem Analysis to Program Design, Barbara Doyle, 2012.**

Visual Studio

Visual Studio, Microsoft tarafından geliştirilen bir tümleşik geliştirme ortamıdır (IDE- Integrated Development Environment). Başta Visual C++, Visual C#, Visual Basic olmak üzere Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework ve Microsoft Silverlight tarafından desteklenen tüm platformlar için Windows Form uygulamaları, web siteleri, web uygulamaları ve web servisleri ile birlikte konsol ve grafiksel/görsel kullanıcı ara yüzü uygulamaları geliştirmek için kullanılır.

- MS Visual Studio: ttp://msdn.microsoft.com/vstudio/

C# için Visual Studio Alternatifleri

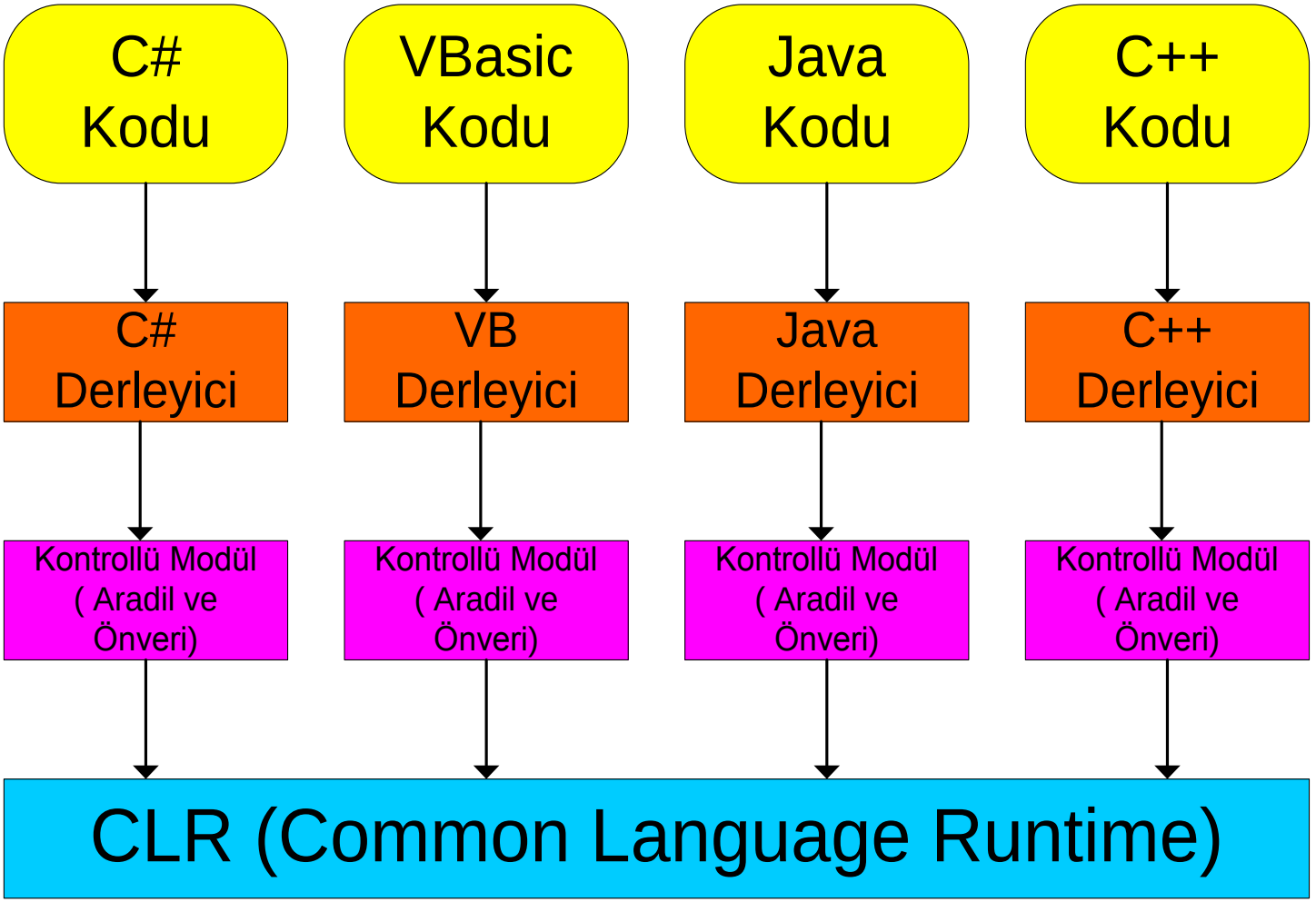
- **Mono**
  - Açık kaynaklı
  - Windows, Linux, Mac OS X, Solaris, Unix
  - Novell tarafından geliştirilmekte

- **Sharp Develop**
  - Açık kaynaklı, .NET SDK veya Mono kullanır
  - C# dilinde yazılmış

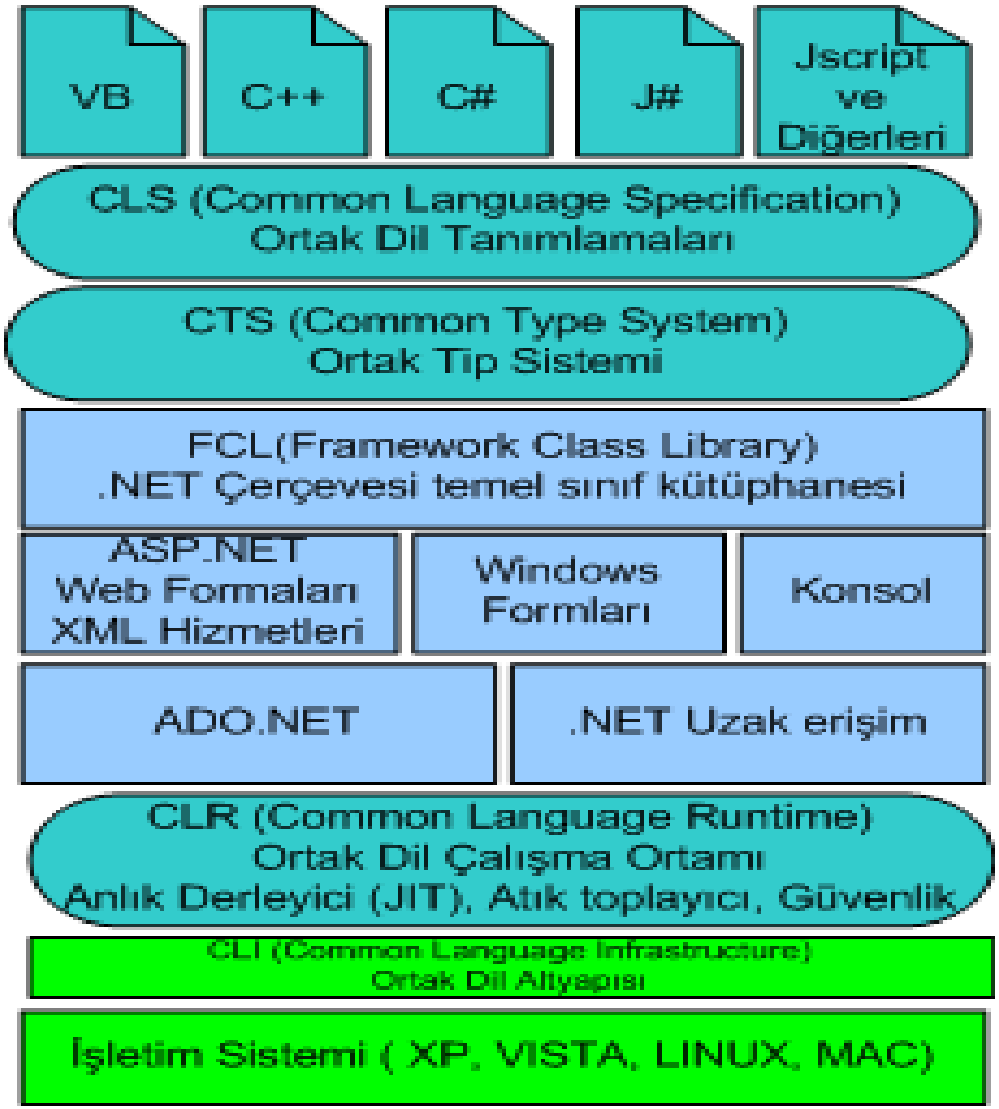
.NET Teknolojileri

- Windows Formları ve Konsol
- WEB Teknolojileri
  - ASP.NET
  - WEB Formları
  - WEB Hizmetleri (XML)
    - SOAP (Simple Object Access Protocol)
    - UDDI (Universal Description, Discovery and Integration)
    - WSDL (Web Services Description Language)
- Veritabanı Teknolojileri
  - ADO.NET
- Ve diğerleri…

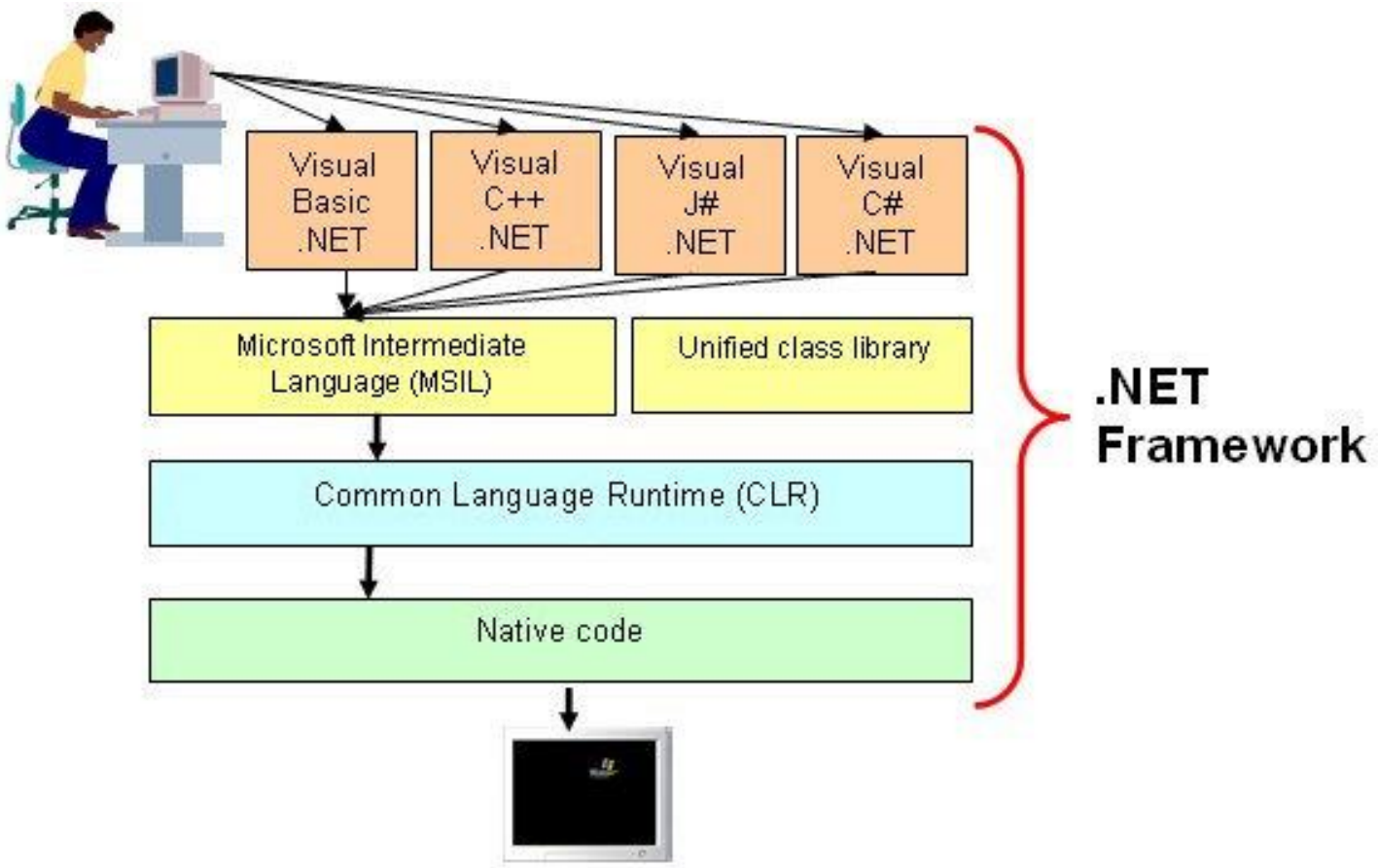
.NET Program Akışı



.NET Bütüncül Yapısı



.NET: Programcı-Program Akışı



.NET Özellikleri

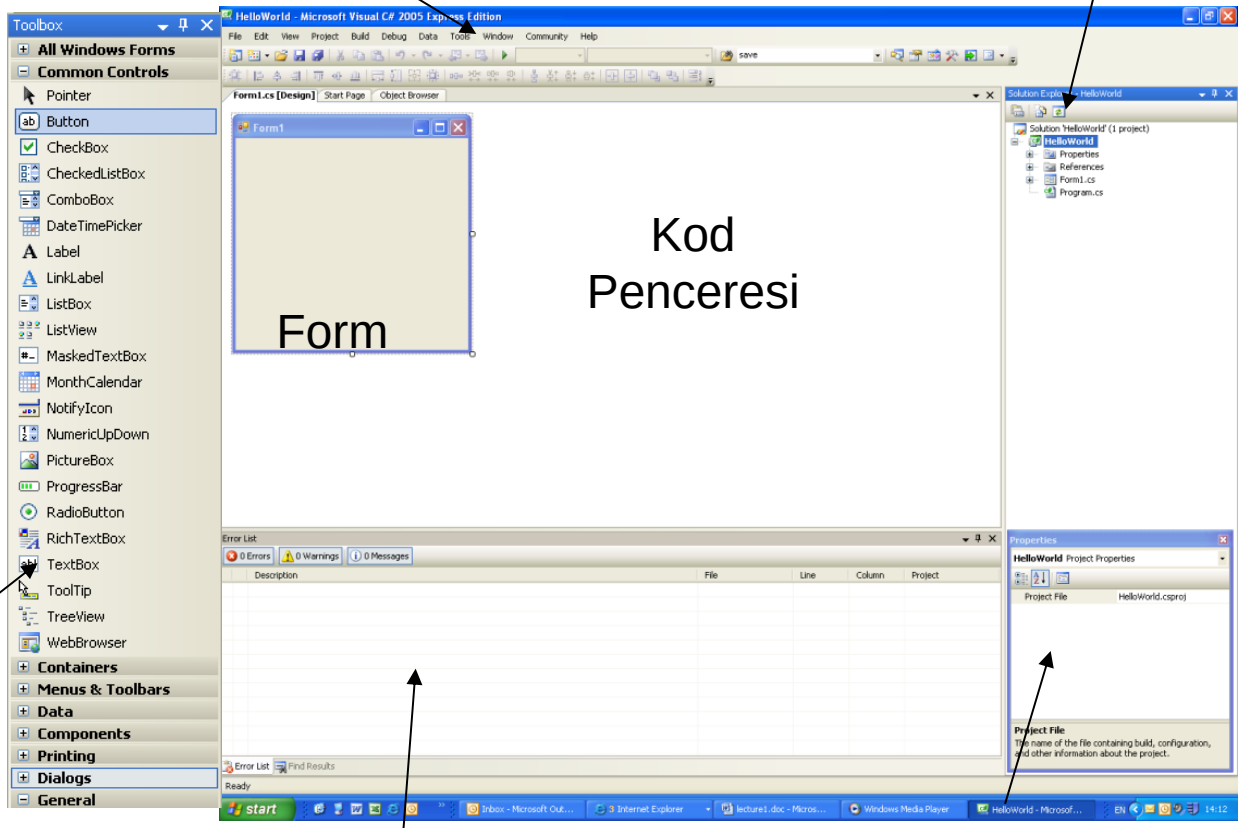
- CLR hangi dil kodunu çalıştırdığını bilmez
- Bütün diller IL (Ortak dil) koduna çevrilir
- IL kodlar her zaman managed'tir.
- IL code ve metadata ayrılamaz, aynı dosyada beraber üretilirler
- Sadece C++.NET ile hem managed hem de unmanaged (default) kod üretilebilir
- Üretilen dosya PE (Portable Executable)
- PE, CLR (.NET Framework) ile çalışır
- Managed Modül:
  - PE Başlık : GUI veya CUI veya DLL, built-time
    - Sadece IL Kod içerirse: PE başlığı ihmal edilir
    - Yerel CPU Kod: PE başlık CPU yerel makine dili kodları içerir
  - CLR Başlık : CLR modeli, bayraklar, Ana fonk. Adresi, module bilgileri
  - Metadata : **Tanımlanmış** veya **hazır** veri türleri, nesnelerin tabloları
  - IL Kod: Derleyici tarafından üretilen ortak kod, daha sonra CLR tarafından yerel CPU makine komutlarına çevrilir

C#

- Microsoft firması tarafından geliştirilen C# (si şarp okunur), C++ ve Java dili gramer yapısını kullanan, nesne yönelimli bir dildir. C#, C++ diline yeni eklentiler yapılarak **((C++)++)** bir adım daha ileriye götürülmüş ve C# dilinin isimlendirilmesinde, + karakterlerinin birbirlerine yakınlaşmış hali ve bir melodi anahtarı olan C# Major kullanılmıştır. C#' ın kaynak dosya uzantısı (soyadı) "**.cs**" dir.

C# Programlama: Design/Tasarım Ortamı

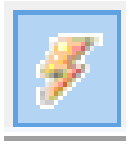
- Design Ortamı:



C# Windows Form Application

- C# Form Uygulamaları, olay tabanlı /hareket bağımlı (event driven) uygulamalardır.
- Geleneksel ya da “procedural/yapısal” uygulamalarda, uygulama kendisini belirli bir sıra ile kontrol etmekte ve işlemektedir. Bu tarzda uygulamalar ilk satırdan çalışmaya başlar ve belirli bir sırayla ilerler ve işlemler (procedure/function) gerekli oldukça çağırılarak çalıştırılır.
- Hareket/Olay-Bağımlı model uygulamalarda ise uygulama belirli doğrusal bir yol izleyerek çalıştırılmaz. Farklı hareketler için farklı işlemler çalıştırılır ve programın her çalıştırılmasıyla uygulama yeniden farklı farklı yollarla çalışmasını sürdürebilir.

OLAYLAR (EVENTS)



- Bir olay/hareket, kullanıcının faresine tıklaması ya da klavyeden herhangi bir tuşa basmasıyla, program içerisinden yapılan bir kontrol düzeneğiyle ya da başka bir pencerenin neden olacağı bir davranışı nedeniyle oluşmuş olabilir.
- Herhangi bir hareketin gerçekleştirilmesi, işletim sistemine bir mesaj gönderilmesine neden olur. Sistem mesajı işler ve diğer pencerelere yayımlar. Her pencere özelikle kendi yapısı ile ilgili (aynı zamanda uygun) mesajları yorumlayarak işler. Örneğin, başka bir pencere kendisi üzerinde yer alacaksa ilgili pencere kendi sınırlarını yeniden düzenler.

OLAYLAR (EVENTS)

- Uygulamanızda, kaynak kodunuz tanımlanmış tetikleme hareketleriyle kontrol edilecek ve çalıştırılacaktır.
- Örneğin bir textBox1'ın içeriğinin değişmesiyle birlikte **TextChanged olayı (**event procedure) 'ü gerçekleşecek ve bu değişikliğe bağlı işlem(ler) sırasıyla çalışacaktır.
  - void **textBox1_TextChanged**(object sender, EventArgs e)
    {

    }
- ***Bir butona tıklanınca (button1_Click) Ekrana "Merhaba " yazan program;***

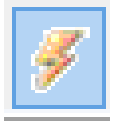
- void **button1_Click**(object sender, EventArgs e)
  {
  MessageBox.Show ("Merhaba");
  }

Özelik, Yöntem ve Hareket (Property, Method,Event)

- C# Form ve nesneleri(kontrolleri); özellikleri, yöntemleri ve olayları/hareketleri olan nesnelerdir.

Özellikler (Property) nesnenin ayırıcı nitelik ve nicelikleri,

- **Yöntemler (Methods)**eylem yetenekleri,



Olaylar (Hareketler)- ise tepkileri olarak düşünülebilir.

Özellikler – Metotlar- Olaylar Nasıl Tanımlanır?

Her bir nesne yada kontrol, özelliklere / properties, metotlara/methods, olaylara/events sahiptir.

Nesne**.Ozelligi =** deger;

```
titleLabel.Text = "Görsel Programlama";
Label.Text = "1234 Esentepe";
msgLabel.AutoSize = true;
sayac = 12;
```

Nesne**.Metot();**

```
helloButton.Hide();
messageLabel.Show();
```

Nesne**_Olay(…)**
{
}

```
private void Form1_Load(object sender, EventArgs e)

private void button1_Click(object sender, EventArgs e)
{

}
```

Örneğin günlük yaşamdan bir nesne BALON seçelim.

- ve bu balonun özeliklerini, yöntemlerini ve hareketlerini tanımlamaya çalışalım
- Özellikleri ;
- genişliği, yüksekliği, hacmi ve rengi, durumu hakkındaki bilgiler (patlak veya patlamamış olması gibi)
- Yöntemleri ;
- Bir balon gerçekleştirebileceği bir takım doğal yöntemlere veya yetilere sahip olacaktır mesela bunlardan (şişirme veya söndürme) bizim seçtiğimiz balonun yöntemleridir. Yine de ayrıca bütün balonlar bu yöntemlere sahiptir.
- Hareketleri ;
- Bir balon bir takım durumlara karşılık tepki verirler. İğne batırılırsa yada hava şiştikten sonra hava verilmeye devam edilirse patlayacaktır. Patlama balonun hareketidir.

- Eğer bir balon programı yapacak olsaydık yazacağımız C# kaynak kodumuz aşağıdaki gibi olacaktı;
- **Balonun özeliklerini belirlemek için**,
- Balon.Renk = Mavi;
- Balon.Basinc = 6;
- Balon.Patlak = False;
- **Balonun metotlarını belirlemek için**,
- Balon.Sis()
- Balon.Son()
- Balon.Buyu(5) *// 5 birim kadar büyüsün*

Balonun olası olayları için,

- void Balon_Patliyor()
- {
- Balon.Son(); *//balonun bir metotu*
- Balon.SesCikart(“paaatt”*);//balonun bir metotu*
- Balon.Patlak = True; *//balonun bir özellliği*
- Balon.Basinc = 1; *//balonun bir özelliği*
- }

C# ile uygulama geliştirmek için dört temel adım vardır.

- Kullanıcı arayüzü oluşturmak.
- Özelikleri atamak.
- Kod yazmak.
- Test etmek.

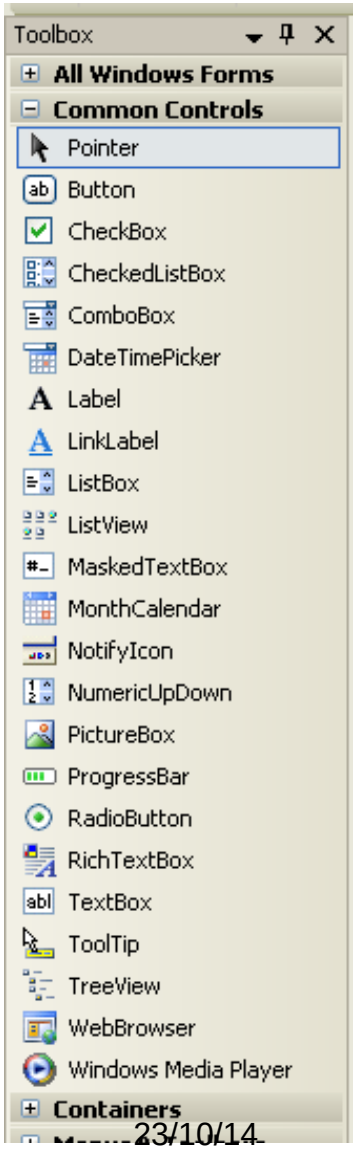
```
Private Sub Command1_Click()
Text1.Text = “Merhaba VB!”
End Sub
```

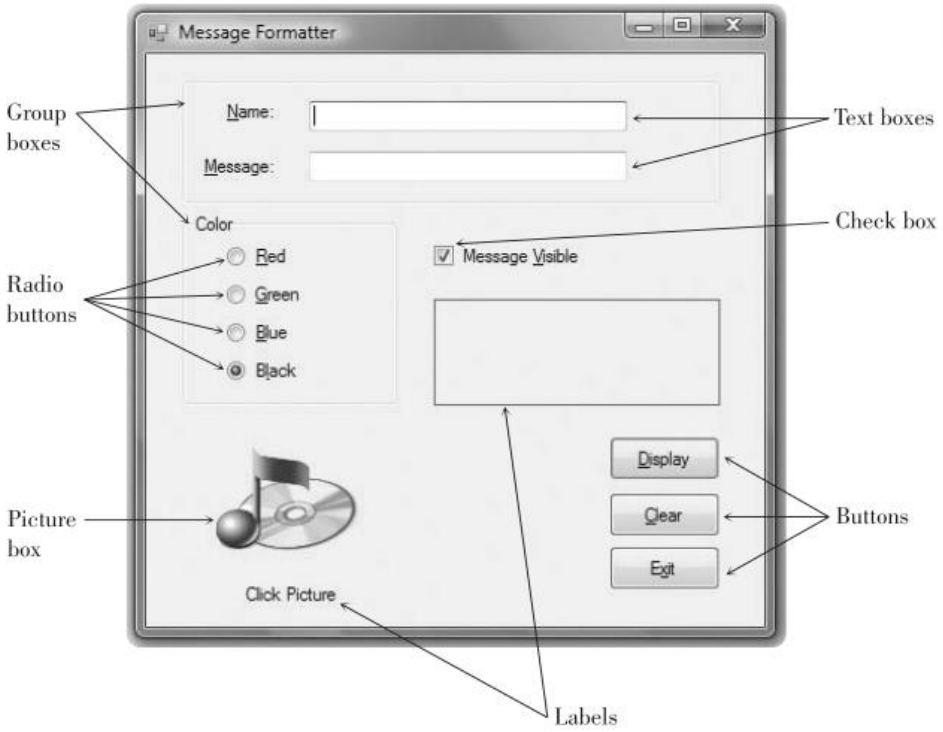
Şimdi klavyeden F9 tuşuna basarak yada C# IDE deki run tuşuna basarak uygulamamızı çalıştırıp test edebiliriz. İyi testler

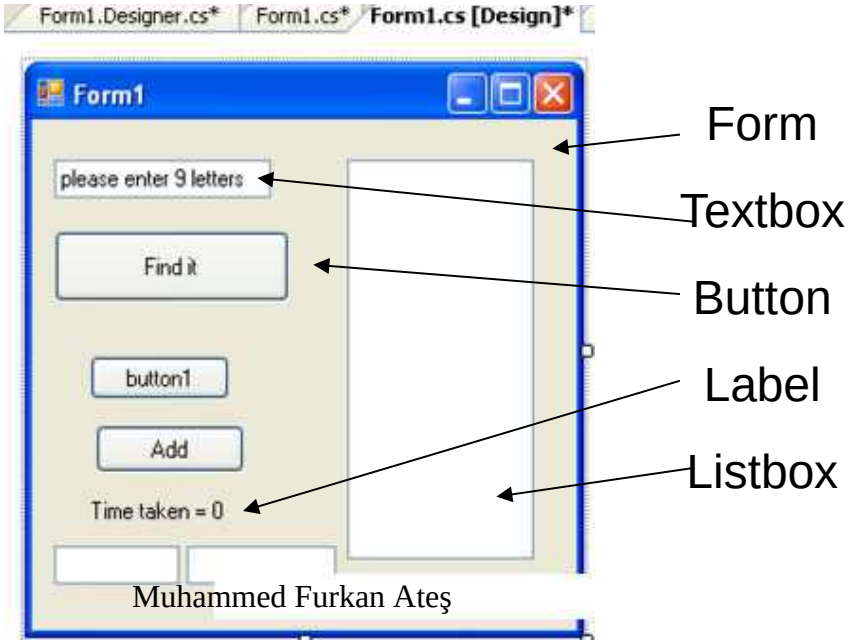
Bir Form Tasarlamak

- Text : Pencerenin başlığı. (window title)
- Icon : Pencerenin simgesi.
- MaxButton : Pencere büyütme tuşuna sahip mi sorusunun yanıtı.
- MinButton : Pencere küçültme tuşuna sahip mi sorusunun yanıtı.
- BorderStyle : Pencerenin tipi.
- Height : Pencerenin yüksekliği.
- Width : Pencerenin genişliği.
- Left : Pencere sol kenarı ile monitörün sol kenarı arasındaki uzaklık.
- Top : Pencere üst kenarı ile monitörün üst kenarı arasındaki uzaklık.
- WindowState : Pencere ilk göründüğünde durumu. (Küçük, büyük, normal)
- Name : Program kaynak kodunda penceremizi tanımamıza da yardımcı olacak isim.
- ***Formun Hareketleri***
- Resize : Pencerenin çerçevesinin büyüklüğünün değişmesi ile birlikte çağrılan hareket işlemi (event procedure).
- Activate : Pencerenin aktif hale gelmesiyle çağrılan hareket işlemi.
- Deactivate : Pencerenin pasif hale gelmesiyle çağrılan hareket işlemi.
- ***Formun Yöntemleri***
- Show : Pencerenin görünmesini sağlayan yöntem.
- Print : Pencere üzerine yazı bastırmamızı sağlayan yöntem.
- Line : Pencere üzerine çizgi çizmemizi sağlayan yöntem.
- Circle : Pencere üzerine çember çizmemizi sağlayan yöntem.
- Refresh : Pencerenin yeniden boyanmasını sağlayan yöntem.

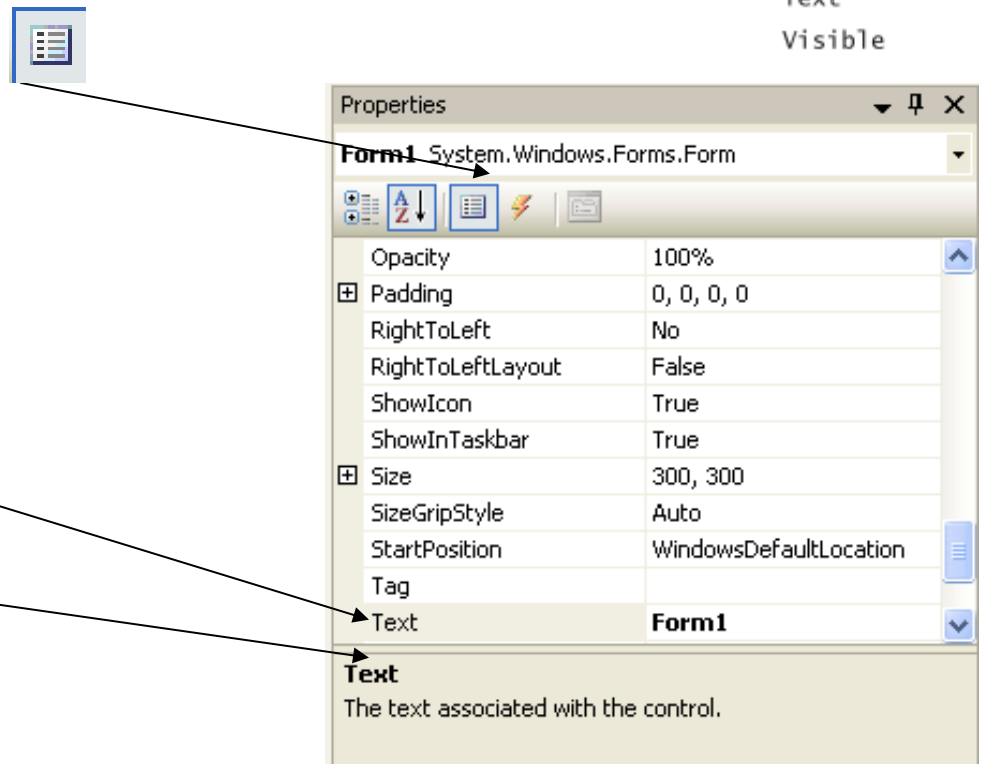
Nesneler / Kontroller





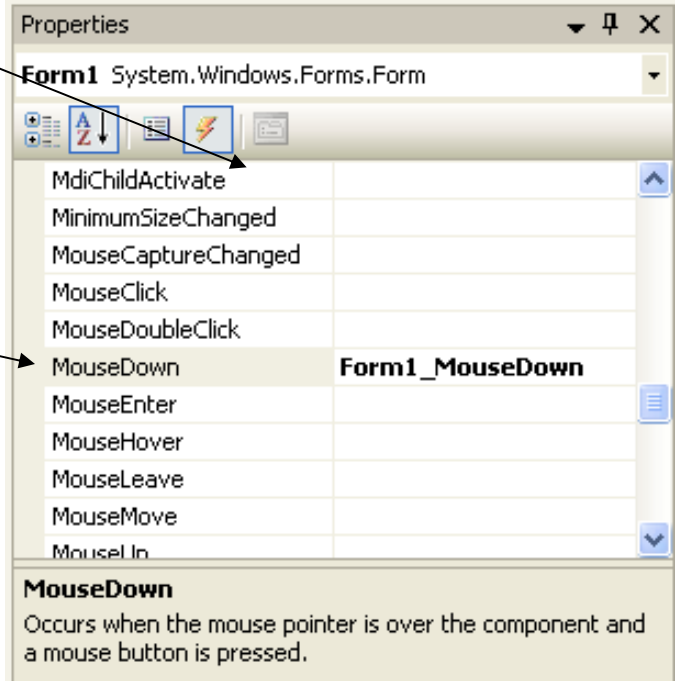


- **Properties / Özellikler penceresi**
- Her kontrol/nesne'nin
- bir özelliği vardır;
  - Name
  - Position (top and left)
  - Size (height and width)
  - Text
- Özelliğin açıklaması



Events/Olaylar Penceresi

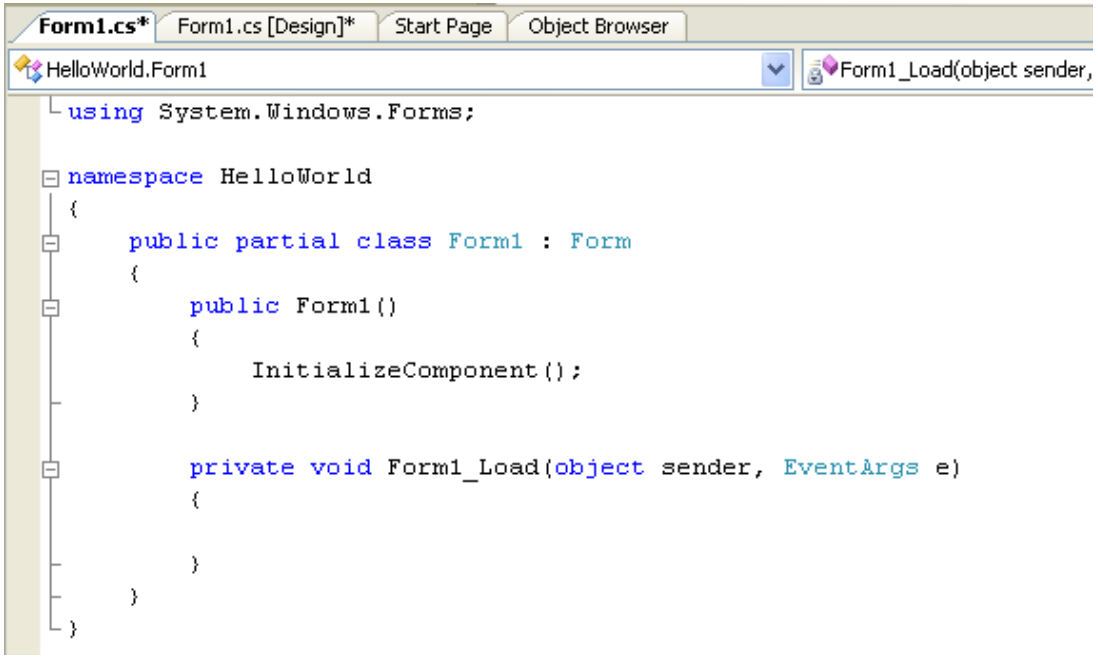
- Events(Olaylar):
  - Button click
  - KeyPress
  - MouseMove
  - MouseDown



- Her nesnenin kendine has bir
- Olayı olabilir
- – Form_load, Timer_tick () gibi

C# Programlama– Kod Editörü

- **Kod Editörü** – Bir nesneye veya forma click yaptığımızda gelen sayfa



Konsol ekranına "Merhaba" yazan örnek program kodu:

```
using System;

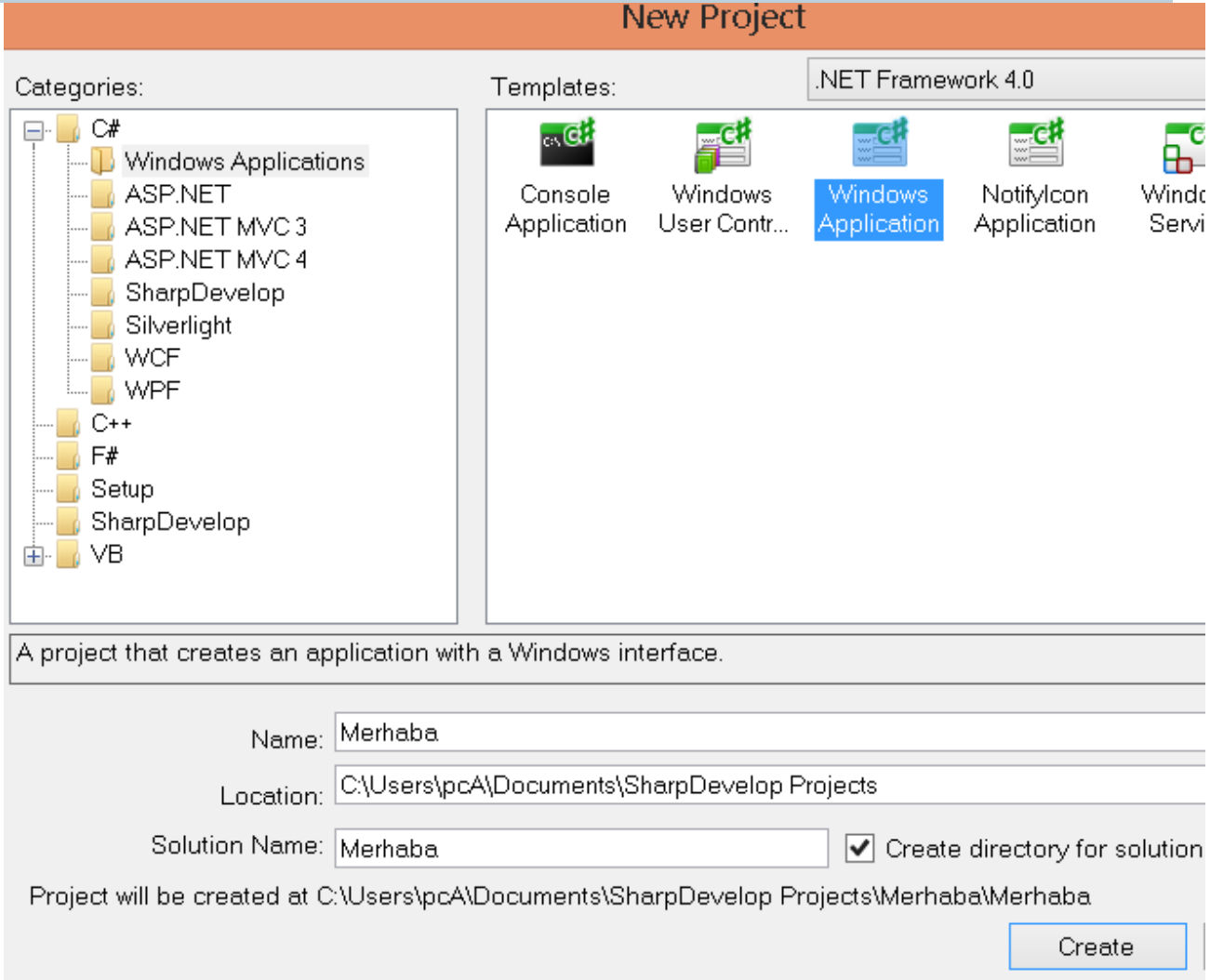
public class Merhaba
{
  public static void Main()
   {
    Console.WriteLine("Merhaba!");
   }
}
```

İsimuzayları (namespaces)

System	Bütün uygulamaların kullandığı temel veri türleri: int, long, float, string
System.Collections	Stack, Queue, Hashtable
System.Diagnostics	Debugging
System.Drawing	Web ve Windows formlarda kullanılan 2-Boyutlu grafik nesnleri
System.IO	Klasör, dosya ve stream nesneleri
System.Management	WMI ile çoklu bilgisayar yönetimi
System.Net	Ağ haberleşmesi
System.Security	Veri ve kaynak koruması
System.Text	Kodlama karakterleri: ASCII, Unicode
System.Threading	Eşzamansız işlemcikler, kaynaklara eşzamanlı erişim

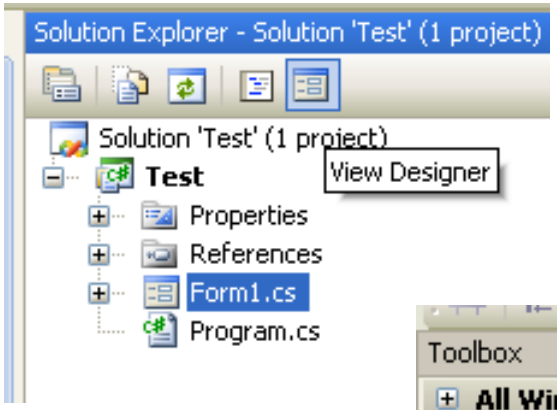
İlk Form uygulamamız

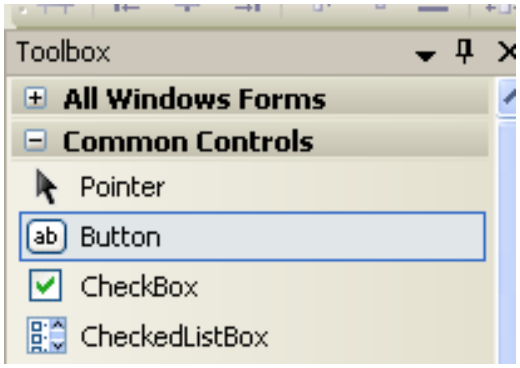
- Çalıştır C#'ı,
- File →New →Project /Solution penceresinden
- Windows Application seçilir.
- Name: Dosya ismi yazılır

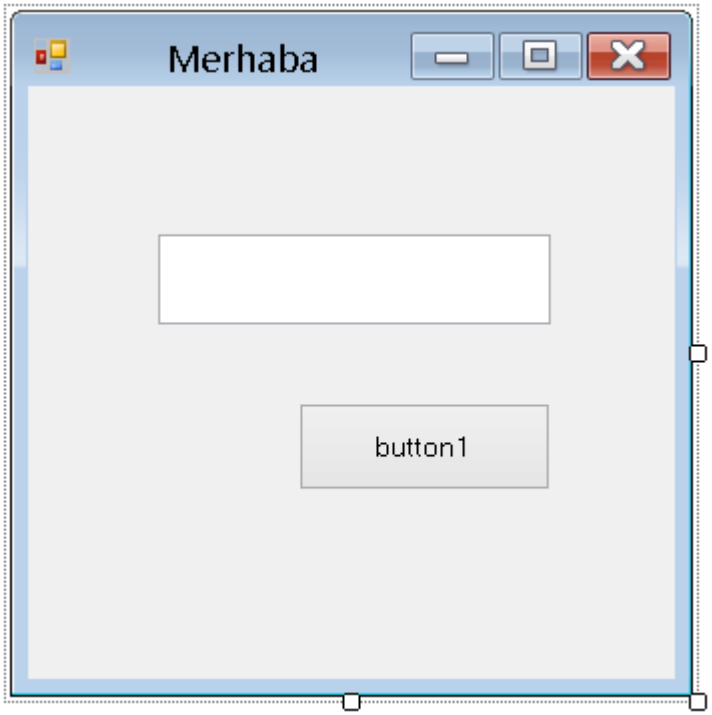


İlk Form Uygulamamız

- Design/Tasarım saçilir (click form1.cs[Design] tab).
- Toolbox tan projemize uygun nesne / kontroller seçilir

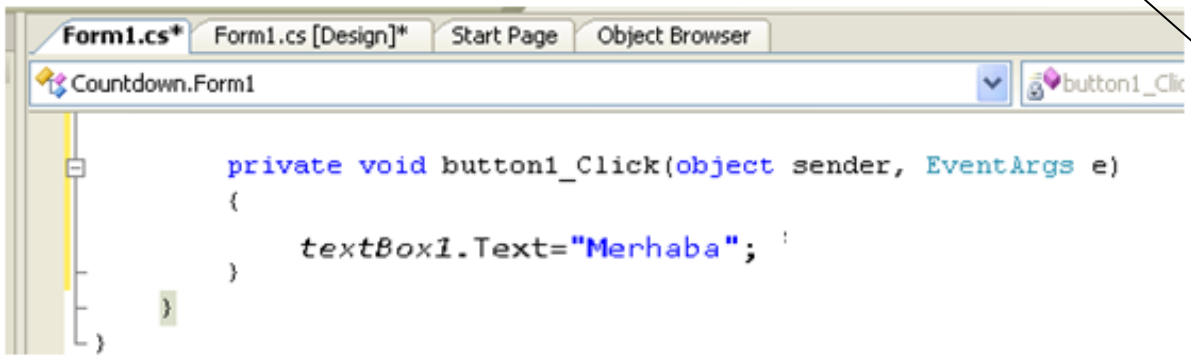


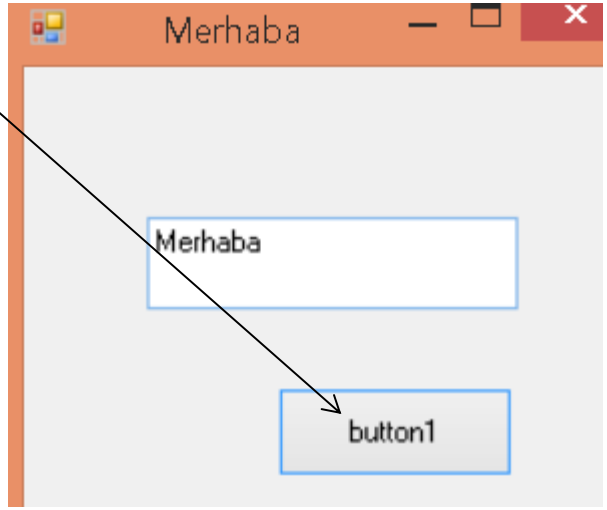




İlk Form Uygulamamız

- Form üzerindeki butona Double-click yapılarak button1'in click olayına kod yazılır.
- Yazılacak kod: textBox1.Text="Merhaba";
- Çalıştır/Run (F9) yapılır ve butona click yapılır





Veri Tipleri

C# type	Numeric range	Width in bits
byte	0 to 255	8-bit
sbyte	–128 to 127	8-bit
char	U+0000 to U+ffff	16-bit
int	–2,147,483,648 to 2,147,483,647	32-bit
uint	0 to 4,294,967,295	32-bit
long	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	64-bit
ulong	0 to 18,446,744,073,709,551,615	64-bit
short	–32,768 to 32,767	16-bit
ushort	0 to 65,535	16-bit

Veri Tipi	Doğrudan dönüşebileceği veri tipleri
byte	short, ushort, int, uint, long, ulong, float, double, decimal
sbyte	short, int, long, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
ulong	float, double, decimal
float	double
char	ushort, int, uint, long, ulong, float, double, decimal

Type	Numeric range	Precision	Size
float	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	7 digits	32-bit
double	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	15–16 digits	64-bit
decimal	1.0×10^{-28} to 7.9×10^{28}	28–29 significant digits	128 bits

Sabit Tanımlamaları

```
const int x = 0;
public const double gConstant = 6.673e-11;
private const string language = "Visual C#";
public const double x = 1.0, y = 2.0, z = 3.0;
```

- static kelimesi kullanılmaz
- *Tanımlandığında değer atanmalı*
- *readonly: atama constructor ‘da veya const gibi*

Sınır Aşımı

- **Sınır aşımı istisnai (exception) durum**

```
byte x=256;   // derleme hatası, sınır aşımı

byte z;
z=250;
z=z+10;              // derlenir ancak sonuç yanlış
```

Sayıları Yazdırma

```
byte x=10;
MessageBox.Show(x.ToString());

int y=905323345577
textbox1.Text=y.ToString("+## (###)### ## ##");

int z=15;
textbox2.Text=z.ToString("X");  //16 lık sistem
textbox3.Text=z.ToString("X4");  //16 lık sistem
```

Tip Dönüştürme:Parse

- Parse : string->sayı

```
string sayı1="1234";
int x=Int32.Parse(sayı1);

string sayı2="12.345";
int y=Int32.Parse(sayı2);    //derleme hatası

long s;
s=long.Parse(textbox1.Text);
```

Tip Dönüştürme:Convert

- **Convert:** string->sayı VE sayı->string

```
string sayı1="1234";
int x=Convert.ToInt32(sayı1);

string sayı2="12.345";
int y=Convert.ToInt32(sayı2);        //derleme hatası

long s;
S=Convert.Tolong(textbox1.Text);
```

Değişkenler

- C# ‘ta yeni türler: bool, decimal
- Herşey nesne: System.Object
- ***object*** tüm değişkenlerin ortak kalıbı
- Reference ve Value
  - Value tipler Stack (yığın) bellekte
  - Reference tipler Heap (öbek) bellekte

Boxing-Unboxing

- Boxing: Stack alanından Heap alanına taşıma

```
object x;
int i=10;
x=i;
```

- Unboxing: Heap alanından Stack alanına taşıma

```
object x;
int i=10, j=20;
x=j;
i=(int)x;
```

Değişken Kapsama Alanları

- Global değişkenler metotların dışında tanımlanır, değer ataması hemen yapılmalıdır ve sınıf içindeki tüm metotlardan erişilir
- Yerel değişkenler metotların içinde tanımlanır, sadece metot içerisinden erişilir
- **public static**: Diğer sınıflardan da erişilir, tüm kopyalarda ortak kullanılır
- **private**: Sadece tanımlandığı sınıftan erişilir
- **protected:** Sadece ilgili sınıftan ya da o sınıftan türetilen sınıflardan erişilir
- **internal**: Sadece aynı isim uzayından

Dizi Tanımlama ve Kullanma

- Tek Boyutlu dizi tanımlamaları
  - `int [] dizi=new  int[10];`
  - `int  [] dizi;  dizi=new int[10];`
  - `string []dizi={“SAU”, “Mek”, “Teknik”};`
  - `int[]dizi={1,2,3,4,5};`
  - `char [] s=“merhaba”;`

Tek Boyutlu Dizi (Array) Elemanlarını Ekrana Yazma

```
int[] myArray = new int[10];
for (int i=0;i<10; i++)
  Console.Write ("{0} ",myArray[i]);
  Console.WriteLine();
```

Çok Boyutlu Diziler

- Düzenli Çok Boyutlu Dizi
  - int [,]dizi=new [3,3];
  - int [,] dizi={{1,2},{3,4},{10,11}};
  - int [,,]dizi=new dizi[5,5,5];
- Düzensiz Çok Boyutlu Diziler (Jagged Arrays)
  - int [][] dizi=new int[3][]; //satır sayısı belli
    - dizi[0]=new int[3];
    - dizi[1]=new int[4];
    - dizi[2]=new int[2];

Düzensiz Dizi İşlemleri

```
class jagged
{
    static void Main()
    {
        int [][]dizi=new int [3][];
        dizi[0]=new int []{1,2,3};
        dizi[1]=new int []{6,7,8,9};
        dizi[2]=new int []{10,11};
        for(int i=0;i<dizi.GetLength(0);i++)
              for(int j=0;j<dizi[i].GetLength(0);j++)
                    Console.WriteLine("dizi[{0}][{1}]={2}",i,j,dizi[i][j]);
     }
}
```

Array Sınıfı Metotları

- **`Array.Copy ()`**
- //Diziyi kopyalar
- **`Array.Sort()`**
- // Dizi elemanlarını küçükten büyüğe sıralar
- **`Array.BinarySearch()`**
- //Dizi elemanları içerisinde ikili arama yapar
- **`Array.Reverse()`**
- //Diziyi ters çevirir.
- **`Array.Clear()`**
- //Diziyi temizler.

Array Sınıfı Metotları

- **`Array.Copy ()`**
- //Diziyi kopyalar
- **`Array.Sort()`**
- // Dizi elemanlarını küçükten büyüğe sıralar
- **`Array.BinarySearch()`**
- //Dizi elemanları içerisinde ikili arama yapar
- **`Array.Reverse()`**
- //Diziyi ters çevirir.
- **`Array.Clear()`**
- //Diziyi temizler.

Döngüler

Java-C-C++ daki gibi

- ```
  while (a > 0)
        a--;
  ```
  - ```
    do
      a++;
    while (a < 10);
    ```
  - ```
    for (int i=1; i<=10; i++) {a--;}
    ```
- Eğer **a Boolean** ise şu şekilde döngü tanımı yapılabilir

```
Boolean a=true;
while (a)
{
    MessageBox.Show("selam");
    a = false;
}
```
```

Karar Yapıları

Java-C++ daki gibi

```
if (a >= 10)
  {
   fonksiyon1();
   fonksiyon2();
  }
else
  ...
```

```
switch (a)
{
  case 10:
   fonksiyon1(); break;
  ……
  default:
  ...
}
```

Çift seçimli yapı→ ?: operatörü

1. Eğer ‘j<k ise m=j değilse m=k dir’ önermesi aşağıdaki gibi yazılabilir.

```
int m = j < k ? j : k;
```

2. Eğer ‘j>k’ ise n=j+k değilse n=j*k işlemini yap’ önermesi aşağıdaki gibi yazılabilir.

```
int n = j > k ? j + k : j * k;
```

Örnek Soru: Windows Hesap Makinesi Tasarımı-1

```
private void btn8_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "8";
}

private void btn9_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "9";
}

private void btnce_Click(object sender, EventArgs e)
{
    TextBox1.Clear();
}

private void btn4_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "4";
}

private void btn5_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "5";
}

private void btn6_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "6";
}

private void btn7_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "7";
}

private void btn0_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "0";
}

private void btn1_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "1";
}

private void btn2_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "2";
}

private void btn3_Click(object sender, EventArgs e)
{
    TextBox1.Text = TextBox1.Text + "3";
}
```

Örnek Soru: Windows Hesap Makinesi Tasarımı-2

Global Değişkenler;

```
int sayi1, sayi2;
char islec;
double sonuc=0;
```

```
private void btnarti_Click(object sender, EventArgs e)
{
sayi1 = Int32.Parse(TextBox1.Text);
TextBox1.Clear();
islec = '+';
}

private void btneksi_Click(object sender, EventArgs e)
{
    sayi1 = Int32.Parse(TextBox1.Text);
    TextBox1.Clear();
    islec = '-';
}

private void btncarpı_Click(object sender, EventArgs e)
{
    sayi1 = Int32.Parse(TextBox1.Text);
    TextBox1.Clear();
    islec = '*';
}

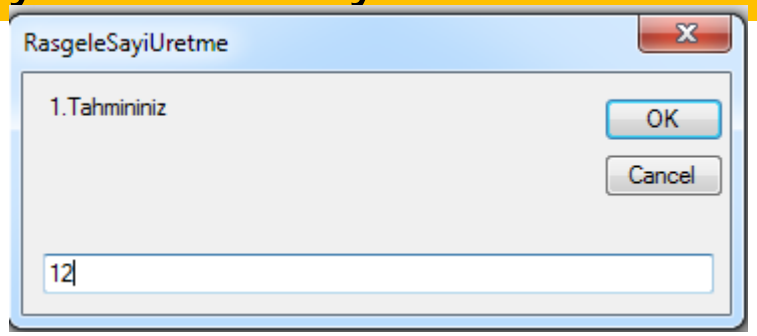
private void btnbolu_Click(object sender, EventArgs e)
{
    sayi1 = Int32.Parse(TextBox1.Text);
    TextBox1.Clear();
    islec = '/';
}
```

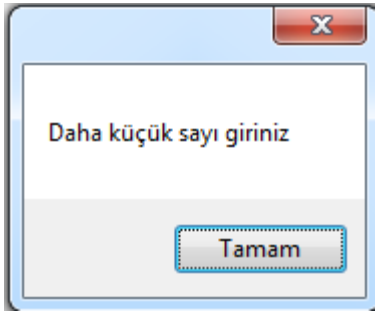
```
private void btnesit_Click(object sender, EventArgs e)
{
 sayi2 = Int32.Parse(TextBox1.Text);
 switch(islec)
 {
     case '+' : sonuc = sayi1 + sayi2; break;
     case '-' : sonuc = sayi1 - sayi2; break;
     case '*' : sonuc = sayi1 * sayi2; break;
     case '/' : sonuc = (double) sayi1 / sayi2; break;
 }
TextBox1.Text = sonuc.ToString();
}
```

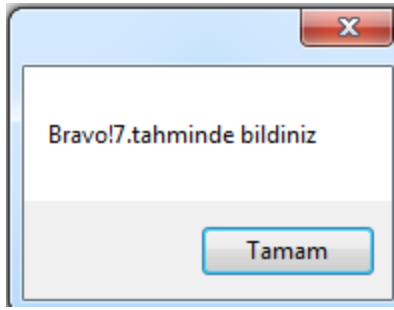
Örnek Soru: Rasgele Sayı Üretimi ve Sayı Tahmin Oyunu

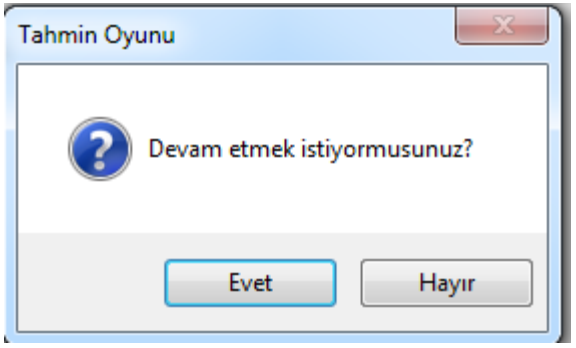
```
int tutulan, sayac = 0, tahmin;
public Form1()
{
    InitializeComponent();
}
void AnaProgram()
{
    Random sayi = new Random();
    tutulan = sayi.Next(1, 100);
    do
    {
        tahmin = int.Parse(Interaction.InputBox(sayac.ToString() + ".Tahmininiz"));
        if (tahmin == tutulan)
        {
            MessageBox.Show("Bravo!" + sayac + ".tahminde bildiniz");
        }
        else if (tahmin < tutulan)
        {
            MessageBox.Show("Daha büyük sayı giriniz");
            sayac++;
        }
        else if (tahmin > tutulan)
        {
            MessageBox.Show("Daha küçük sayı giriniz");
            sayac++;
        }

    } while (tutulan != tahmin);
}
private void Form1_Load(object sender, EventArgs e)
{
    DialogResult cvp;
    do
    {
        sayac = 0;
        AnaProgram();
        cvp = MessageBox.Show("Devam etmek istiyormusunuz?", "Tahmin Oyunu", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    } while (cvp == DialogResult.Yes);
    this.Close();
}
```





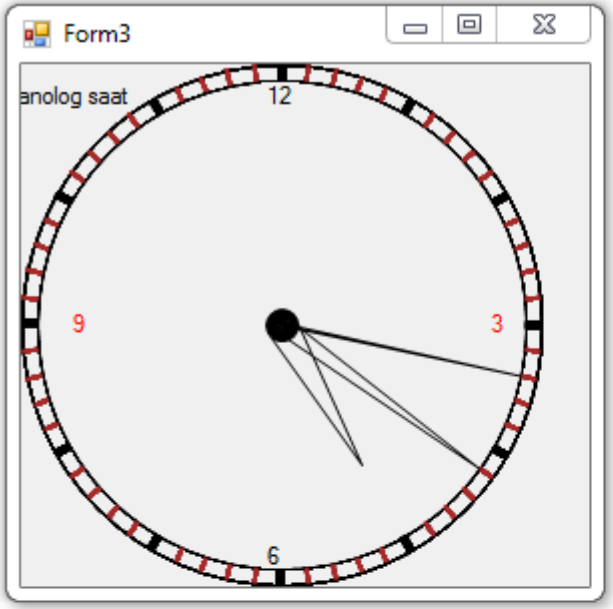


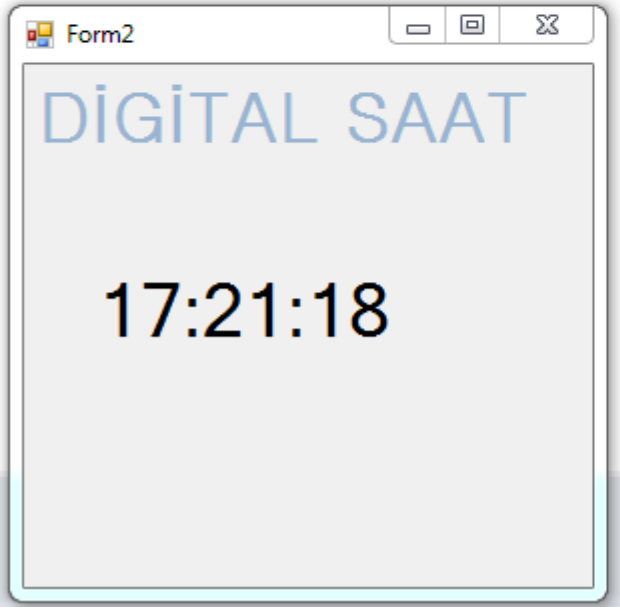


Grafik Komutları

Method
`Object.Clear();`
`Object.Dispose();`
`Object.DrawArc(Pen, x1Integer, y1Integer, x2Integer, y2Integer, widthInteger, heightInteger);` `Object.DrawArc(Pen, Rectangle, startAngleFloat, angleLengthFloat);`
`Object.DrawLine(Pen, x1Integer, y1Integer, x2Integer, y2Integer);` `Object.DrawLine(Pen, Point1, Point2);`
`Object.DrawEllipse(Pen, xInteger, yInteger, widthInteger, heightInteger);` `Object.DrawEllipse(Pen, Rectangle);`
`Object.DrawRectangle(Pen, xInteger, yInteger, widthInteger, heightInteger);` `Object.DrawRectangle(Pen, Rectangle);`
`Object.DrawPie(Pen, xInteger, yInteger, widthInteger, heightInteger, angleStartInteger, angleLengthInteger);` `Object.DrawPie(Pen, Rectangle, angleStartFloat, angleLengthFloat);`
`Object.DrawString(textString, Font, Brush, xFloat, yFloat);` `Object.DrawString(textString, Font, Brush, PointF);`
`Object.FillEllipse(Brush, xInteger, yInteger, widthInteger, heightInteger);` `Object.FillEllipse(Brush, Rectangle);`
`Object.FillPie(Brush, xInteger, yInteger, widthInteger, heightInteger, angleStartInteger, angleLengthInteger);` `Object.FillPie(Brush, Rectangle, angleStartFloat, angleLengthFloat);`
`Object.FillRectangle(Brush, xInteger, yInteger, widthInteger, heightInteger);` `Object.FillRectangle(Brush, Rectangle);`

Uygulama: Saat





Uygulama: Form1

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

    private void Form1_Load(object sender, EventArgs e)
    {
        radioButton1.Checked = false;
        radioButton2.Checked = false;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        Form2 frm2 = new Form2();
        Form3 frm3 = new Form3();

        if (radioButton1.Checked == true)
        {
            frm2.Show();

        }
        if (radioButton2.Checked == true)
        {
            frm3.Show();
        }
    }
```

```
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Interval = 1000;
    timer1.Enabled = true;
    //Formun iç yüksekliği ile iç genişliğini eşitleme
    this.Height = this.Width + this.ClientSize.Width - this.ClientSize.Height;
}

private void Form1_Resize(object sender, EventArgs e)
{
    //Form boyutlandığında içeriği sil
    Graphics g;
    g = this.CreateGraphics();
    g.Clear(this.BackColor);
    g.Dispose();
}
```

Uygulama: Form2:Digital Saat

```
public partial class Form2 : Form
{
    public Form2()
    {
        InitializeComponent();
    }

    private void Form2_Load(object sender, EventArgs e)
    {
        timer1.Enabled = true;
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        label1.Text = DateTime.Now.ToLongTimeString();
    }
}
```

Uygulama: Form3:Analog Saat

```
private void timer1_Tick(object sender, EventArgs e)
 {
     Graphics g;
     g = this.CreateGraphics();
     float xorta, yorta, yarı_çap, sx, sy;
     if (this.ClientSize.Width > this.ClientSize.Height)
         yarı_çap = this.ClientSize.Height / 2;
     else
         yarı_çap = this.ClientSize.Width / 2;
     g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
     //Saatin etrafındaki ait dairelerin çizimi
     g.DrawEllipse(new Pen(Color.Black), 0, 0, 2 * yarı_çap, 2 * yarı_çap);
     g.DrawEllipse(new Pen(Color.Black), 8, 8,
      2 * yarı_çap - 16, 2 * yarı_çap - 16);
     g.FillEllipse(new SolidBrush(this.BackColor), 9, 9,
      2 * yarı_çap - 18, 2 * yarı_çap - 18);
     xorta = yarı_çap;
     yorta = yarı_çap;
     yarı_çap -= 8;
     int açı, saniye, dakika, saat;
     int kx1, kx2, ky1, ky2;
     //Saati ve dakikayı gösteren çizgileri oluşturma
     for (açı = 0; açı <= 360; açı += 6)
     {
         kx1 = (int)(xorta + (yarı_çap + 8) * Math.Cos(açı * 3.1415 / 180));
         ky1 = (int)(yorta - (yarı_çap + 8) * Math.Sin(açı * 3.1415 / 180));
         kx2 = (int)(xorta + (yarı_çap) * Math.Cos(açı * 3.1415 / 180));
         ky2 = (int)(yorta - (yarı_çap) * Math.Sin(açı * 3.1415 / 180));
         if ((açı % 30) == 0)
             //Her 30 derecede bir saat başı bulunur. Bunları kalın çizme
             g.DrawLine(new Pen(Color.Black, 4), kx1, ky1, kx2, ky2);
         else
             //Her 6 derecede bir dakika bulunur. Bunları ince çizme
             g.DrawLine(new Pen(Color.Brown, 2), kx1, ky1, kx2, ky2);
     }
```

```
//Orta noktayı bulup Bizim Saat yazma
kx1 = (int)(xorta - (g.MeasureString("y",
 new Font("Tahoma", 8, FontStyle.Regular)).Width) / 2);
ky1 = 30;

//Saat, Dakika, Saniye çizgilerini hesapla
System.Drawing.Drawing2D.GraphicsPath p = new System.Drawing.Drawing2D.GraphicsPath();
saat = DateTime.Now.TimeOfDay.Hours; //saati al
açı = -saat * 30 + 90; //her saat 30 derecedir.
//360 derece / 12 saat = 30
sx = (int)(xorta + yarı_çap * 2 / 3 * Math.Cos(açı * 3.1415 / 180));
sy = (int)(yorta - yarı_çap * 2 / 3 * Math.Sin(açı * 3.1415 / 180));
kx1 = (int)(xorta - 8 * Math.Cos(açı * 3.1415 / 180 + Math.PI / 2));
kx2 = (int)(xorta + 8 * Math.Cos(açı * 3.1415 / 180 + Math.PI / 2));
ky1 = (int)(yorta + 8 * Math.Sin(açı * 3.1415 / 180 + Math.PI / 2));
ky2 = (int)(yorta - 8 * Math.Sin(açı * 3.1415 / 180 + Math.PI / 2));
//Akrebi üçgen şeklinde çizme
p.AddLine(kx1, ky1, sx, sy);
p.AddLine(kx2, ky2, sx, sy);
p.AddLine(kx1, ky1, kx2, ky2);

dakika = DateTime.Now.TimeOfDay.Minutes; //dakikayı saatten alma
açı = -dakika * 6 + 90; //her bir dakika 6 derecedir.360 derece/60dakika
sx = (float)(xorta + yarı_çap * Math.Cos(açı * 3.1415 / 180));
sy = (float)(yorta - yarı_çap * Math.Sin(açı * 3.1415 / 180));
kx1 = (int)(xorta - 4 * Math.Cos(açı * 3.1415 / 180 + Math.PI / 2));
kx2 = (int)(xorta + 4 * Math.Cos(açı * 3.1415 / 180 + Math.PI / 2));
ky1 = (int)(yorta + 4 * Math.Sin(açı * 3.1415 / 180 + Math.PI / 2));
ky2 = (int)(yorta - 4 * Math.Sin(açı * 3.1415 / 180 + Math.PI / 2));
//Yelkovanı üçgen şeklinde çizme
p.AddLine(kx1, ky1, sx, sy);
p.AddLine(kx2, ky2, sx, sy);
p.AddLine(kx1, ky1, kx2, ky2);

saniye = DateTime.Now.TimeOfDay.Seconds; //saniyeyi saatten alma
açı = -saniye * 6 + 90; //her bir saniye 6 derecedir.360 derece/60saniye
sx = (float)(xorta + yarı_çap * Math.Cos(açı * 3.1415 / 180));
sy = (float)(yorta - yarı_çap * Math.Sin(açı * 3.1415 / 180));
kx1 = (int)(xorta - 1 * Math.Cos(açı * 3.1415 / 180 + Math.PI / 2));
kx2 = (int)(xorta + 1 * Math.Cos(açı * 3.1415 / 180 + Math.PI / 2));
ky1 = (int)(yorta + 1 * Math.Sin(açı * 3.1415 / 180 + Math.PI / 2));
ky2 = (int)(yorta - 1 * Math.Sin(açı * 3.1415 / 180 + Math.PI / 2));
//Saniyeyi üçgen şeklinde çizme
p.AddLine(kx1, ky1, sx, sy);
p.AddLine(kx2, ky2, sx, sy);
p.AddLine(kx1, ky1, kx2, ky2);
//Ortadaki yuvarlak
p.AddEllipse(xorta - 8, yorta - 8, 16, 16);
//Akrep, Yelkovan ve Saniye çizimlerini yapma
g.DrawPath(new Pen(Color.Black, 1), p);
//Ortadaki yuvarlağı boyama
g.FillPath(new SolidBrush(Color.Black), p);
g.Dispose();
p.Dispose();
```