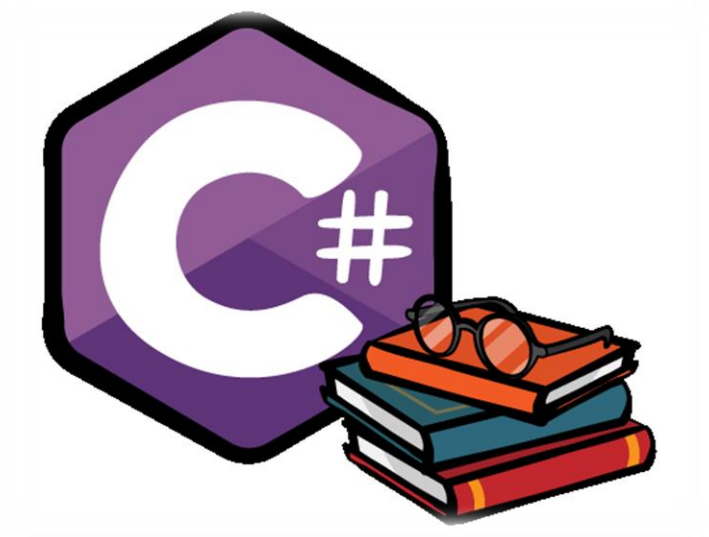


FORM EKRANI-6

FORM EKRANI

- ✓ **Class**

KAZANIMLAR

- ✓ *Class yapısını öğrenir ve kavrar.*
- ✓ *Nesneleri program kod bloğuna göre sıralar*
- ✓ *Nesnelerin ortak olay, metot ve özelliklerini kullanır*
- ✓ *Programa göre nesneleri tasarlar.*
- ✓ *Program ihtiyaçlarını analiz eder.*

SINIFLAR

Gerçek hayatta karşılaştığımız bir problemi çözerken nasıl adım adım ilerlersek program geliştirirken de bu şekilde ilerleğiz. Problemi önce küçük parçalara böler daha sonra her birini ayrı ayrı geliştiririz. Sonrada her bir parçaya birleştirerek bir bütün haline gelmesini sağlarız.

Sınıfları da programlama dünyasında tam olarak bu amaçla kullanırız. Programın her bir aşaması için ayrı sınıflar yazarız. Hatta bazen önceden bir başkası tarafından hazırlanmış bir sınıfı kullanırız. Daha sonrasında da ayrı ayrı geliştirilen bu sınıfları birleştirerek ortaya yeni bir program çıkartırız.

Bu şekilde gerçekleştirilen görev dağılımı (böl ve birleştir) nesne tabanlı programlamanın da ana mantığıdır.

Geliştirilen programı mümkün olduğu kadar küçük parçalara bölerek, her parçayı bir türde nesne haline getirip ve daha sonra bu nesneleri gerekli yerlerde çağırarak kullanmaktır. Nesneleri yazmak için önce bir şablon (kalıp) oluşturulur. Daha sonra bu şablondan istenilen sayıda nesne çıkarılır. Programlama dilinde bu şablonlara ***class,*** *nesnelere de* ***object*** *adı verilir.*

Hadi gelin beraber ilk sınıfımızı oluşturalım.

```
Class Ev

{

}
```

Ev Sınıfın ismidir. İstediğimiz adı verebiliriz. { } bloğu sınıfın gövdesidir. Şimdi Ev sınıfımız işe yarar hale getirmeye çalışalım. Bir ev ile ilgili bilgileri tutan değişkenleri ve o bilgilerle işlem yapan fonksiyonları Ev sınıfının gövdesine yerleştiricez. Örneğin bir evin adresi, kapı numarası, oda sayısı, evin sahibi vb. bilgilerden istediklerimizi bu gövdeye yazalım

```
Class Ev

{

İnt kapıNo;

String sokakAdı;

}
```

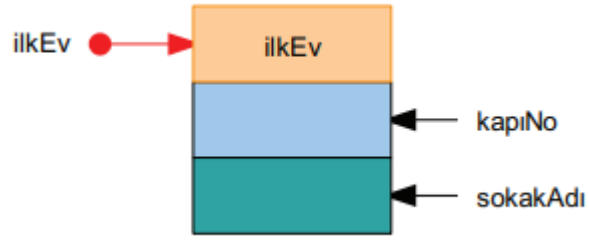
Sınıftan Nesne Oluşturma

Sınıftan nesne oluştururken Main () metodundan yararlanıcaz.

Ev ilkEv;

ilkEv → *null*

ilkEv=new Ev();



Ana bellekte yandaki gibi yer ayrılıyor. Yani Ev sınıfının iki öğesini tanımlamıştık: KapıNo ve SokakAdı Bellekte de yaratılan nesneye göre bu iki öğemize yer ayrılmış oluyor. Bir sınıfa ait nesne(object) yaratmak demek, ana bellekte sınıfa ait static olmayan bütün öğelere birer yer ayırmak demek. Bu ayrılan yerlere başka değerler yazılamaz; ve bu yaratılan nesnenin öğelerine daha değer atamadık. Yani daha kiracı taşınmadı. C# bellekte yaratılan her değişkene bir öndeğer (default value) atar.Veri tipine göre bu değişir.Örneğin; KapıNo değişeni int tipi olduğu için öndegeri 0,sokakAdı ise string olduğu için öndegeri “”(boş string)

Bir sınıftan bir nesne yaratırken söyle tanımlama yaparız:

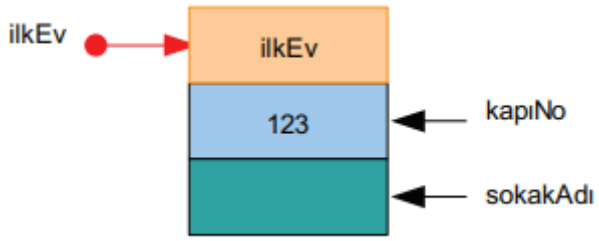
Sınıfın_adı nesne_adı=new sınıf_adı();

```
Class Uygulama
{
static public void Main(string[]arg)
{
Ev ilkEv=new Ev();
}
}
```

Nesnenin Öğelerine Erişim

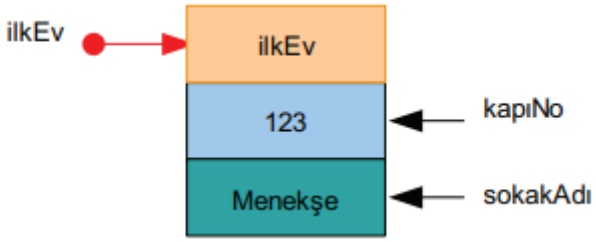
ilkEv.kapıNo=123;

Bu atamadan sonra ilkEv nesnesinin durumu aynen söyle



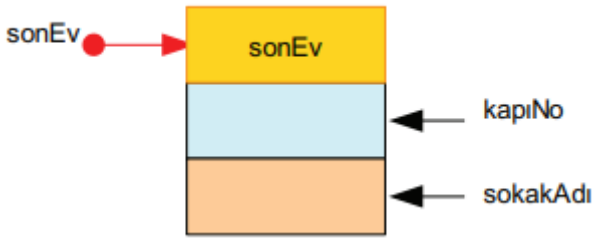
Şimdide bunu yapalım

ilkEv.sokakAdı=”Menekşe”;

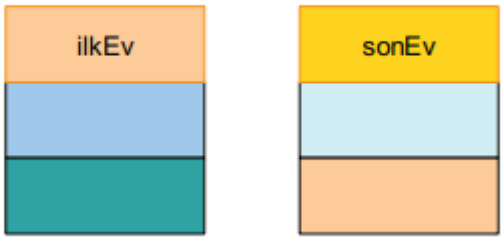


Nesne içindeki değişkenlere yapılan atama deyimlerinde,değişken adlarının önüne nesne adını yazıyoruz.Diyelim ki;

Ev sonEv=new Ev();



Nesne yaratıcını kullanarak bir başka nesne yarattık.Ozaman,bellekteki iki nesne ve her birinde kapıNo ve sokakAdı bileşenleri ayrı ayrı yer alacaktır.



kapıNo=123;

sokakAdı=”Menekşe”;

atama deyimlerini yazarsak,derleyici,bu değerleri hangi ev için atayacağını bilemez,hata iletisi verir.Bu nedenle,önce nesneyi, sonra değişkeni yazıyoruz.Tabi aralarına(.)koyamayı unutmayalım.

ilkEv.kapıNo

ifadesi,ilkEv nesnesi içindeki kapıNo değişkeninin adıdır.Benzer sekilde de

sonEv.kapıNo

Dikkat: *Görüldügü gibi farklı nesneler içinde aynı adı taşıyan değişkenlere farklı değerler atanabilir.Bir sınıftan istediğimiz kadar farklı nesne türetebiliriz.Hadi program adımlarımzı şimdi birleştirelim..*

```
using System;
class Ev
{
Public int kapıNo;
Public string sokakAdı;
}
Class Uygulama
{
static public void Main()
{
Ev ilkEv=new Ev();
ilkEv.kapıNo=123;
ilkEv.sokakAdı=”Menekşe”;
Console.WriteLine(“ADRES: ” + ilkEv.sokakAdı + “ Sokak,No: “ +ilkEv.kapıNo);
```

}

Cıktı

ADRES:Menekşe Sokak, No: 123

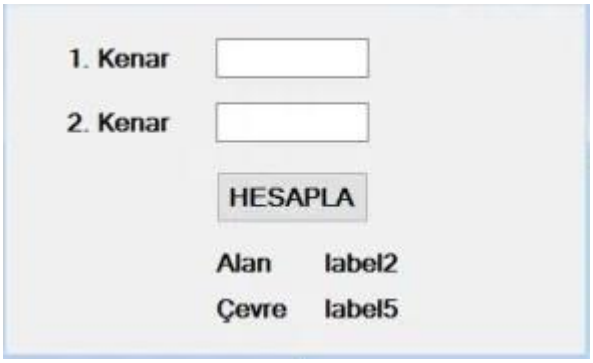
Örnek: *Form ekranına 2 buton yerleştirelim. Birinci butona tıkladığımızda mesaj versin ikinci butona tıkladığımızda da yeni bir mesaj verelim. Bunu da öğrendiğimiz class yapısını kullanarak yaparım.*

Kod Blokları

```
namespace WindowsFormsApp6
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            mesajlar msj = new mesajlar();

            msj.mesaj1();
        }

        private void button2_Click(object sender, EventArgs e)
        {

            mesajlar msj = new mesajlar();

            msj.mesaj2();

        }
        class mesajlar

        {
            public void mesaj1()

            {
                MessageBox.Show("Hello World");
            }

            public void mesaj2()

            {
```

```
        }

    }
}
```

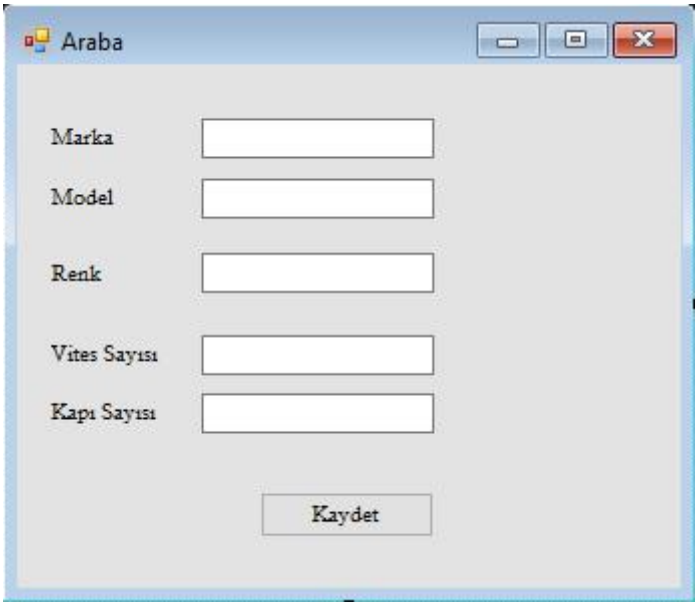
Örnek: *Bu örneğimizde C# ile **Ucgen** isimli bir **Class** oluşturarak 2 dik kenarı girilen bir üçgenin alanını ve çevresini hesaplayacağız. Örneğimizde kullanacağımız yöntem **Ucgen Sınıfı** içinde dik kenarlar için **Property** oluşturmak ve **AlanHesapla()** isimli ve **CevreHesapla() metotlar** ile dik üçgenin alanını ve çevresini hesaplamak olacak.*




```
namespace Class_Ucgen
{
    class Ucgen
    {
        double kenar1;
        double kenar2;
        public double Kenar1
        {
            get
            {
                return kenar1;
            }

            set
            {
                kenar1 = value;
            }
        }
        public double Kenar2
        {
            get
            {
                return kenar2;
            }

            set
            {
                kenar2 = value;
            }
        }
        public double AlanHesapla()
        {
            return (kenar1 * kenar2) / 2;
```

```
        public double CevreHesapla()
        {
            double hipotenus = Math.Sqrt(Math.Pow(kenar1, 2) + (Math.Pow(kenar2, 2)));
            return (kenar1 + kenar2 + hipotenus);
        }
private void button1_Click(object sender, EventArgs e)
        {
            Ucgen dikucgen = new Ucgen();
            dikucgen.Kenar1 = Convert.ToDouble(textBox1.Text);
            dikucgen.Kenar2 = Convert.ToDouble(textBox2.Text);
            label2.Text = dikucgen.AlanHesapla().ToString();
            label5.Text = dikucgen.CevreHesapla().ToString();
        }
    }
}
```

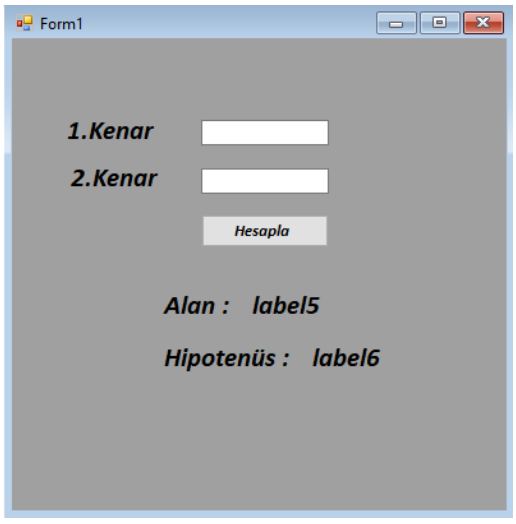
Örnek: Araba nın özellikleri girildiği bir class örneği yapalım.




```
public partial class Form1 : Form
{
    class araba
    {
        public string Marka { get; set; }
        public string Model { get; set; }
        public string Renk { get; set; }
        public int VitesSayısı { get; set; }
        public int KapıSayısı { get; set; }
    }
    public Form1()
    {
        InitializeComponent();
    }
```

```
    private void button1_Click(object sender, EventArgs e)
    {
        int b;
        int a;
        araba araba1 = new araba();
        araba1.Marka = textMarka.Text;
        araba1.Model = textModel.Text;
        araba1.Renk = textRenk.Text;

        if (int.TryParse(textVites.Text, out a) == false ||
int.TryParse(textKapı.Text, out b) == false)
        {
            DialogResult diyalog1 = MessageBox.Show("Lütfen Vites ve kapı Değerini
Doğru giriniz", "Dikkat", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);

            if (diyalog1 == DialogResult.OK)
            {
                MessageBox.Show("Akıllı ol :)");
            }
        }
        else
        {
            araba1.VitesSayısı = a;
            araba1.KapıSayısı = b;
            MessageBox.Show("İçerik Girildi\n" + "Araba Markası: " + araba1.Marka +
                "\n" + "Araba Modeli: " + araba1.Model + "\nAraba Rengi: " +
                araba1.Renk + "\nVites Sayısı: " + araba1.VitesSayısı + "\nAraba Kapı
Sayısı: " + araba1.KapıSayısı, "Arabanız", MessageBoxButtons.OK,
                MessageBoxIcon.Information);

        }

    }
```

Örnek: Kenarları girilen üçgenin alan ve hipotenüs değerlerini class la hesaplayalım



Kod Blokları

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

namespace ClassUcgen
{
    class Ucgen
    {
        double kenar1;

        public double Kenar1
        {
            get { return kenar1; }
            set
            {
                if (value <= 0) // Kenar 0 ve daha küçükse 1 yapıyoruz.
                {
                    kenar1 = 1;
                }
                else
                    kenar1 = value;
            }
        }
        double kenar2;

        public double Kenar2
        {
            get { return kenar2; }
            set
            {
                if (value <= 0) // Kenar 0 ve daha küçükse 1 yapıyoruz.
                { kenar2 = 1; }
                else
                    kenar2 = value;
            }
        }

        public double Alan()
        {
            double ucgenAlan = 0;
            ucgenAlan = (kenar1 * kenar2) / 2;
            return ucgenAlan;
        }
        public double Hipotenus()
        {
            double ucgenHipotenus = 0;
            ucgenHipotenus = Math.Sqrt((Math.Pow(kenar1, 2) + Math.Pow(kenar2, 2)));
            return ucgenHipotenus;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Ucgen dikUcgen = new Ucgen();
            dikUcgen.Kenar1 = Convert.ToDouble(textBox1.Text);
            dikUcgen.Kenar2 = Convert.ToDouble(textBox2.Text);
            label3.Text = "Alan = " + dikUcgen.Alan();
            label4.Text = "Hipotenüs = " + dikUcgen.Hipotenus();
```

```
            }
        }
    }
```