YAPICI METOT

Bir sınıf oluşturduğumuzda bu sınıf üyesi olan değişkenlere nesne tanımlandığında doğrudan değer verebilmek için yapıcı metot kullanılır. Bu değer verme işlemi değişkenleri boş bırakma veya belli bir değer atama şeklinde yapılabilir. Değer verme işlemi Sınıf içerisinde bir metot tarafından verilebilir. Bunu yanında nesneyi oluştururken de istenen değerler başlangıç değeri olarak verilebilir.

Yapıca metotlarda dikkat edilmesi gereken nokta metodun dönüş tipinin olmadığı ve metodun adının sınıfın adı ile aynı olması gerektiğidir.

```
class Program
{
    static void Main(string[] args)
    {
        yapici bilgi1 = new yapici(); //ilk yapıcı çalıştı

        yapici bilgi2 = new yapici("İstanbul","Bakırköy",1256000); //ikinci yapıcı çalıştı

        Console.WriteLine(bilgi1.il + " " + bilgi1.ilce + " " + bilgi1.nufus);

        Console.WriteLine(bilgi2.il + " " + bilgi2.ilce + " " + bilgi2.nufus);

        Console.ReadKey();
    }
}

class yapici
{
    public string il;
    public string ilce;
    public int nufus;

    //Değerleri sınıf içerisinde atanarak oluşturulan yapıcı
    public yapici()
    {
        il = "Sinop";
        ilce = "Merkez";
        nufus = 53000;
    }

    //Değerleri parametre olarak alan yapıcı
    public yapici(string il,string ilce,int nufus)
    {
        //burada "this" deyimi ile kullanılan değişken sınıf üyesi olan değişkendir.
        //diğer değişken parametre olan (metoda dışarıdan gelen) değişkendir.
        this.il = il;
        this.ilce = ilce;
        this.nufus = nufus;
    }
}
```

YIKICI METOT

Belleğin dinamik kullanımı için sınıf içerisindeki metotlar çalışıp işleri bittikten sonra belleği terk ederler ve bellekten kalan verilerini temizlerler. Bu belleği terk etme ve temizleme işini yapan yıkıcı metotlardır. C#, Java gibi dillerde bu iş için Garbage Collector bulunur. Bu nedenle ayrıca bir yıkıcı metot tanımı gerekmez.

```
class yikici
{
    //yıkıcı metodun adı sınıf adı ile aynıdır. Önüne ~ operatörü alır.

    ~yikici()    //Class'ın işi bittiğinde bellek temizleniyor.
    {
        Console.WriteLine("program sonlandı, Bellek boşaldı");
    }
}
```

DEĞİŞKEN SAYIDA PARAMETRE ALAN METOD

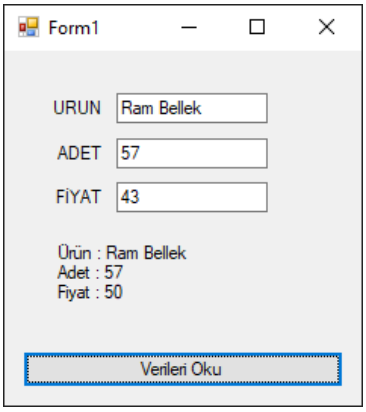
Bazen üzerinde işlem yapılacak veri sayısı belli olmayabilir. Çalışılacak veri adedi değişiklik gösterebilir. Bu durumda klasik bir dizi ile çalışamayız. Çünkü dizilerin eleman sayısı bellidir. Eğer belli olmayan sayıda veri ile işlem yapacaksak (veriler aynı veri türünde olmak koşulu ile) değişken sayıda parametre alan metot oluşturmamız gerekir. Bunun için kullandığımız anahtar kelime belirsiz sayıda veriden bir dizi oluşturmak için kullanılan **params** deyimidir. Oluşturulacak olan bu dizinin türü kullanım amacına göre istenilen şekilde belirtilebilir.

```
class Program
{
     static void Main(string[] args)
     {
         sayilar ornek1 = new sayilar();
         double eb=ornek1.buyukbul(8);
         Console.WriteLine("En Büyük Değer ="+eb);
         Console.ReadKey();
     }
}

class sayilar
{
     private int enbuyuk;

     //Değişken sayıda değer alan metod
     public int buyukbul(params int[] degerler)
     {
         enbuyuk = degerler[0];

         foreach(int max in degerler)
         {
             if(max>enbuyuk)
             {
                 enbuyuk = max;
             }
          }
            return enbuyuk;
      }
}
```

PROPERTY

Sınıf içerisindeki Private olan verilerin değerlerine erişmek ya da bu verilere değer atamak istediğimizde kullandığımız özel yapılardır. İçerisinde get ve set olamak üzere iki bölüm bulunur. Property'nin get metodu ilgili private verinin değerini isteyene bildirir. Set metodu ise private olan veriye değer ataması yapmak için kullanılır.

Değer ataması yaparken istenirse veri çeşitli kontroller ile alınabilir. Gelen veri istenilen kriterlere göre sınanarak alınabilir ve istenirse gelen değerden bir başka değer veriye atanabilir.




```
public partial class Form1 : Form
{
    private void button1_Click(object sender, EventArgs e)
    {
        ozellik uret = new ozellik();

        uret.Urun = textBox1.Text;
        uret.adet = Convert.ToInt32(textBox2.Text);
        uret.fiyat = Convert.ToDouble(textBox3.Text);

        label4.Text = "Ürün : " + uret.Urun + "\nAdet : " +
uret.adet + "\nFiyat : " + uret.fiyat;
    }
}

class ozellik
{
    public string Urun;
    private int Adet;
    private double Fiyat;
    //Propertylerin metotlar gibi metot başlığında parantezleri bulunmaz.

    public int adet //Adet değişkeni için
    {
        get
        {
            return Adet;
        }
        set
        {
            Adet = value;
        }
    }

    public double fiyat //Fiyat değişkeni için
    {
        get
        {
            return Fiyat;
        }
        set
        {   /*Gelen veri kontrol ediliyor.
            Eğer veri 50'den küçükse verinin değeri 50 yapılıyor. */
            if (value < 50)
                Fiyat = 50;
            else
                Fiyat = value;
        }
    }
}
```