Nesne Yönelimli Programlama

5.Hafta

Konu: Sınıflardan Nesne Yaratma, Get ve Set Metotları

Dr. Öğr. Üyesi Güncel SARIMAN
E-posta: guncelsariman@mu.edu.tr

Sınıf Bildirimi

```
class Ev
{
}
```

Sınıf gövdesi yukarıdaki gibi boş olabilir. Ama boş gövdeli sınıfın bir işe yaramayacağı açıktır. Onun içe yarar olabilmesi için içine sabit, değişken ve fonksiyon bildirimleri (tanımları) eklenecektir. Onlara sınıfın öğeleri (class member) denilir.

Sınıf Bildirimi

Bir ev ile ilgili bilgileri tutan değişkenleri ve o bilgilerle işlem yapan fonksiyonları Ev sınıfının gövdesine yerleştireceğiz. Örneğin, bir evin adresi, kapı numarası, oda sayısı, evin sahibi, emlak vergisi, fiyatı vb bilgilerden istediklerimizi sınıfın gövdesine koyabiliriz.

Her evin bir kapı numarası vardır. Bu numara bir tamsayıdır. byte, short, int veya long tipinden olabilir.

```
class Ev

{ int kapıNo ; string sokakAdı ;

}
```

Sınıf Bildirimi

Bir ev yapmak istiyoruz. Önce evin mimari tasarımını yaparız. O tasarım evin konumunu, odalarını, mutfağını, banyosunu, pencerelerini, nasıl ısıtılacağını, nasıl havalanacağını, suyun, elektriğin, doğal gazın nerelerden nasıl geleceğini, vb belirler.

Bu tasarım, büyük bir yapı için uzmanların tasarladığı ve çizdiği karmaşık bir mimari proje olabileceği gibi, bir köylünün kendi kafasında tasarladığı basit bir kulübe de olabilir. İster büyük, karmaşık bir yapı, ister bir kulübe olsun, önce ortada bir tasarım vardır. İster kâğıt üzerine çizilsin, ister birisinin kafasında tasarlanmış olsun, o tasarım somut bir ev, bir nesne değildir.

Evin mimari tasarımını, C# dilinde bir sınıfa (class) benzetebiliriz. Mimari tasarımdan inşa edilen somut bir evi de C# dilinde bir sınıftan elde edilen bir nesneye (object) benzetebiliriz. Bir sınıf bir tasarımdır. Onu somut olarak kullanamayız. Onun için sınıftan nesne(ler) kurmalıyız

Sınıf Bildirimi

Bir mimari tasarımdan aynı tipte istediğimiz kadar ev inşa edebiliriz. Örneğin bir sitedeki veya bir apartmandaki evlerin hepsi bir tek tasarımdan üretilir.

Ancak, o evlerin her birisi kendi başına somut bir varlıktır, her biri uzayda farklı bir yer işgal eder.

Evler aynı yapıya sahiptirler, ama her bir evin boyası, badanası, içindeki eşyalar, insanlar bir başka evin içindekilerden farklıdır. Bu öznitelikler, bir evi ötekinden ayırır.

Benzer olarak, bir sınıftan istedigimiz kadar nesne kurabiliriz. O nesnelerin yapıları aynıdır, ama öznitelikleri farklıdır.

new Operatörü ile Nesne Yaratmak

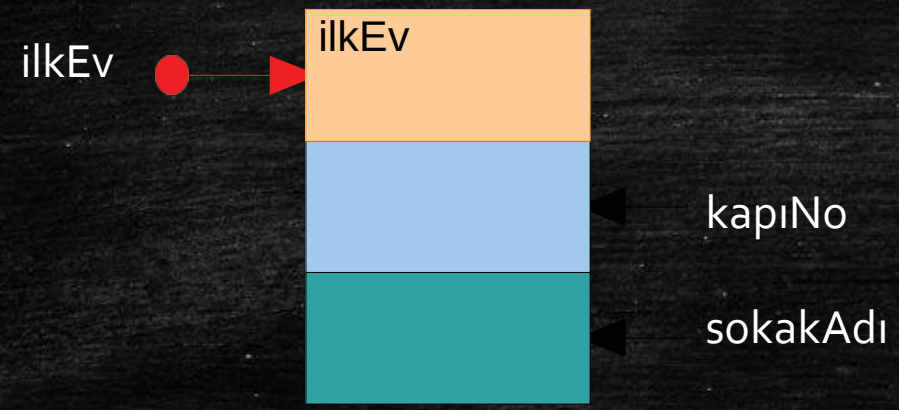
```
class Uygulama
{           static public void Main()
            {
            Ev  yeniev ;
            }

}
```

Bu deyim derleyiciye Ev sınıfına ait bir nesnenin ana bellekteki adresini gösterecek bir referans bildirimidir. Çeşitli kaynaklarda buna pointer, gösterici, işaretçi, referans gibi adlar verilir. C# dili 'reference' sözcüğünü kullanır. Biz de ona uyarak 'referans' veya 'işaretçi' sözcüklerini eş anlamlı olarak kullanacağız. Bunun tek ve önemli görevi Ev sınıfına ait bir nesnenin bellekteki adresini işaret etmektir.

new Operatörü ile Nesne Yaratmak

İşaretçimizin işaret edeceği somut bir Ev inşa etmeliyiz. Somut bir Ev'i inşa etmek demek, ana bellekte Ev'in bütün ögelerinin sığacagı bir yeri tahsis etmek demektir. Bu yere Ev'in bir nesnesi (object, instance) denir. Yapılan bu işe de sınıfa ait bir nesne yaratmak (instantiate) denilir.

```
yeniEv = new Ev() ;
```



new Operatörü ile Nesne Yaratmak

Ev sınıfının iki ögesini tanımlamıştık: kapıNo ve sokakAdı. Bellekte yaratılan nesneye bakarsak, orada kapıNo ve sokakAdı için ayrı ayrı yerler açılmış olduğunu görüyoruz. Bir sınıfa ait nesne (object) yaratmak demek, ana bellekte sınıfa ait statik olmayan bütün ögelere birer yer ayırmak demektir. Ayrılan bu yerlere başka değerler yazılamaz; o nesne bellekte durduğu sürece, yalnızca ait oldukları ögelerin değerleri girebilir. Bunu, bellekte yer kiralama gibi düşünebiliriz.

Bize göre kapıNo ve sokakAdı değişkenleri için ayrılan hücreler boştur. Ama, C# bellekte yaratılan her değişkene kendiliğinden bir ön değer (default value) atar. Ön değer veri tipine göre değişir. Aşağıdaki tablo başlıca veri tiplerine atanan ön değerleri göstermektedir.

Veri Tipi	Deger
byte, short, Int, long	o
float, double	o,o
bool	False
char	'\o' (null karakter)
string	"" (boş string)
nesne (object)	null

new Operatörü ile Nesne Yaratmak

Ev=Yukarıda tasarladığımız Ev sınıfıdır;

ilkEv=Tasarımdan üretilecek olan somut evin yerini işaret eder. Bu nedenle, ilkEv'i işaret ettiği evin adıymış gibi de düşünebiliriz. Bundan böyle işaretçi (referans) adı ile işaret ettiği nesneyi aynı adla anacağız. Söylemlerimizde, kastedilen şeyin referans mı, nesne mi olduğu belli olacaktır. Ancak çok gerektiğinde, referans oluşuna ya da nesne oluşuna vurgu yapacağız.

= Atama operatörü

new= Sınıftan nesne yaratan operatör (nesne yaratıcı) Ev() Yaratılacak nesnenin tasarımı

new Operatörü ile Nesne Yaratmak

Şu ana kadar iki sınıf tanımladık. Ev adlı sınıf bir evin kapı numarası ile bulunduğu sokağı tutacak iki değişkene sahiptir. Ama Ev sınıfı bir tasarımdır, kendi başına bir iç yapamaz.

Uygulama adlı sınıfta Main() metodu tanımlıdır ve bu metot Ev sınıfının bir nesnesini yaratacak olan nesne yaratıcıyı çalıştırmaktadır. Nesne yaratıldıktan sonra, onun öğeleriyle ilgili işleri yapmaya başlayabiliriz.

Nesnenin Öğelerine Erişim

İlk işimiz nesnenin ögelerine (degişken) birer değer atamak olmalıdır. ilkEv nesnesi içindeki kapıNo ve sokakAdı bileşenleri birer değişkendir. Dolayısıyla, degişkenlerle yapılan her iç ve içlem, onlara da uygulanabilecektir. Birincisi int tipinden, ikincisi string tipinden olduğuna göre, onlara tiplerine uygun birer değer atayabiliriz:

```
yeniEv.kapıNo = 123;

yeniEv.sokakAdi="Menteşe"
```

Nesnenin Öğelerine Erişim

```
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 = "Menekçe";
Label1.Text="ADRES: " + ilkEv.sokakAdı + "
Sokak, No: " + ilkEv.kapıNo;
}
}
```

Örnek

Örneğin bir araba fabrikasında elimizde motor, tekerlek vb.(değişken) veriler var. Eğer biz bu verilerle araba nesnelerini oluşturmazsak bunların hiçbir anlamı yoktur. Aynı verilerle birbirinden farklı birçok nesne oluşturabiliriz.

```
Tasit otomobil = new Tasit();
```

Artık otomobil adında gerçek bir nesnemiz oluşmuş durumda

Bir sınıfa ait bir nesne oluşturduğumuzda, her nesne de sınıfta tanımlanan örnek değişkenlerin birer kopyası oluşur. Yani her nesnede renk, marka, hiz, yakitTipi değişkenlerinin kopyaları bulunmaktadır. Bu aynı zamanda nesnelerin içindeki değişkenlerin farklı değer alabilecekleri anlamına gelmektedir. Bu değişkenlere nokta(.) operatörü ile erişilebilir.

```
otomobil.hiz = 220;
```

Örnek

İlk kısımda Tasit sınıfı tipinde otomobil adında bir referans oluşturuyoruz. Yani ortada fiziksel bir nesne yok. İkinci kısımda ise fiziksel olarak ram de nesne yer almış olur. otomobil 'in değişkenine nesnenin referansı atanır ve nesne ile ilişkilendirilmiş olur. Burada new operatörünün görevi, dinamik olarak bellekte yer ayırmak ve referans döndürmektir. Bu referans değişkeninin (otomobil) içinde saklanır.

```
Tasit motor = 5;

// motor nesneyi değil referansını taşır,

bu yüzden herhangi bir atama yapamayız.
```

Örnek

```
public class Tasit{
public String yakit;// Taşıtın yakıt tipi
public int hiz; // Taşıtın Maximum hızı
public String renk; // Taşıtın rengi
public String marka; // Taşıtın markası
 // Taşıtın bilgilerini ekrana yazdıran metot
        public void tasitInfo()
        {

        String tasit = "Taşıtın markası: " + marka + " rengi: " + renk + " yakıt tipi: " + yakit + " maximum hızı: " + hiz;

        System.Console.WriteLine (tasit);
        }
    }
```

Örnek

```
class MainClass
        {
                public static void Main (string[] args)
                {
                //iç satırlar
                }
}
```

Örnek

```
// Taşıt tipinden otomobil nesnesini oluşturuyoruz
Tasit otomobil = new Tasit ();
Tasit motor = new Tasit ();

// nesnemize özellik değerlerini giriyoruz
otomobil.hiz = 220;
otomobil.yakit = "LPG";
otomobil.renk = "Siyah";
otomobil.marka = "Renault";
```

```
// nesnemize özellik değerlerini giriyoruz
motor.hiz = 200;
motor.yakit = "Benzin";
motor.renk = "Metalik Gri";
motor.marka = "Honda";
 /* Yukarıda dikkat edilmesi gereken bir nokta var. Oluşturduğumuz iki nesnenin örnek değişkenlerine farklı değerler atıyoruz. Yani her nesne oluşturulduğu sınıfın örnek değişkenlerinden bir kopya taşır, kendisini değil. Bu yüzden her nesne örnek değişkenlere farklı değerler verip kullanabilir.
*/
// Bilgileri ekrana yazdıracak metodu çağırıyoruz
otomobil.tasitInfo();
motor.tasitInfo ();
```

Properties

Properties, private bir alanın değerini okumak, yazmak veya hesaplamak için esnek bir mekanizma sağlayan bir üyedir. Properties(Özellikler), public veri üyeleri gibi kullanılabilir, ancak aslında accessors adı verilen özel yöntemleri içerirler.

Bir Properties erişimcisi, ilgili alanı almaya (okuma veya hesaplama) veya ayarlamaya (yazmaya) yardımcı olan yürütülebilir ifadeleri içerir. Erişici bildirimleri, get erişimci, set erişimci veya her ikisini de içerebilir.

Properties

```
class Person
{
 private string name; //field
 public string Name //property
 {
 get { return name; }
 set { name = value; }
 }
}
```

Person sınıfı hem set hem de get accessors'a sahip bir Name özelliği içerir.

Set erişimcisi, name değişkenine bir değer atamak için kullanılır; get name değişkeninin değerini döndürmek için kullanılır.

value, set erişimcisini kullanarak bir propertye atadığımız değeri temsil eden özel bir anahtar kelimedir.

property adı istediğiniz herhangi bir şey olabilir, ancak kodlama kuralları gereği propertylerin adları büyük harfle başlamalı ve özel alanla aynı ada sahip olmalıdır.

Properties

```
class Person
{
 private string name;
 public string Name
 {
 get { return name; }
 set { name = value; }
 }
}
```

```
static void Main(string[] args)
{
 Person p = new Person();
 p.Name = "Bob";
 Console.WriteLine(p.Name);
}
```

Get ve Set Metotları

Set ve Get metotlarını birer kontrol mekanizması olarak düşünebiliriz. Olası problemleri önlemek, işlemleri güvenilir ve kontrollü bir şekilde gerçekleştirmek için Set ve Get metotlarını kullanırız.

Basit bir senaryo üzerinden konumuzu açıklamaya devam edelim. Otel otomasyonu için müşteri bilgilerini tutan bir sınıf tasarladığımızı düşünelim. Müşterinin ad-soyad, TC kimlik numarası ve oda numarası bilgilerinin tutulması için gerekli tasarım;

```
class Musteri
{
   public string AdSoyad;
   public ulong TCNo;
   public int OdaNo;
}
```

Get ve Set Metotları

Musteri sınıfının üyeleri public olarak bildirildiği için bu üyelere doğrudan erişilip değerler atanabilir. İşte bu noktada kontrolü elimize almamız lazım aksi taktirde TC kimlik numarası eksik/fazla girilebilir veya 120 odalı bir otelde oda numarası negatif veya 120'den büyük girilebilir.

Amacımız dikkatsizlik sonucu yaşanabilecek olası sorunların önüne geçmek. Bu yüzden üyelere doğrudan erişimi engelleyip (Private), Get ve Set metotları ile kontrollü bir erişim sağlayacağız

Get ve Set Metotları

```
class Musteri
{    private string AdSoyad;private ulong TCNo;private int OdaNo;public string adsoyad
   {
    get
      {return AdSoyad;}
      set
      {AdSoyad = value;}
   }
```

Get ve Set Metotları

```
public ulong tcno
   {get  {return TCNo;}
    set
    {if (value.ToString().Length == 11)
            TCNo = value;
     else
            Console.WriteLine("HATA! TC Kimlik Numarası 11 Haneli Olmalıdır.");
      }
   }
```

Get ve Set Metotları

```
public int odano
{get{
return OdaNo;
}
set {
 if (value > 0 && value <= 120)
OdaNo = value;
else
Console.WriteLine("HATA! Oda Numarası 1-120 Aralığında Olmalıdır. ");
 }
}
```

Get ve Set Metotları

```
class Program
{
    static void Main(string[] args)
    {
        Musteri m1 = new Musteri();
        m1.adsoyad = "Güncel Sarıman";  // adsoyad özelliğinin SET metodu çalıştı.
        m1.tcno = 99999999999;        // tcno özelliğinin SET metodu çalıştı.
        m1.odano = 110;               // odano özelliğinin SET metodu çalıştı.
        // adsoyad, tcno ve odano özelliklerinin GET metodu çalıştı.
        Console.WriteLine("Ad/Soyad:{0} - Tc No:{1} - Oda No:{2}", m1.adsoyad, m1.tcno, m1.odano);

    }
}
```

Get ve Set Metotları

Sınıfın AdSoyad, TCNo, OdaNo üyelerini diğer sınıfların erişimine kapattık (private) ve bu üyelere olan erişimi adsoyad, tcno, odano özellikleri üzerinden kontrollü bir şekilde sağladık.

GET Metodu

Bir özelliğin değeri okunmak istenildiğinde o özelliğe ait GET metodu çalışır.

Console.WriteLine(m1.adsoyad);

m1 nesnesinin içerisindeki adsoyad özelliğinin Get metodu çalışacaktır.

Get ve Set Metotları

SET Metodu

Bir özelliğe atama yapılmak istenildiğinde o özelliğe ait SET metodu çalışır. Atanan değere SET metodu içerisinde value anahtar sözcüğü ile erişilir. value anahtar sözcüğünün önceden belirlenmiş herhangi bir türü yoktur. Özelliğe atanacak değer hangi türden ise value da o türden olur.

m1.adsoyad=”Güncel Sarıman”;

m1 nesnesinin içerisindeki adsoyad özelliğinin SET metodu çalışacaktır ve value ifadesi “Güncel Sarıman” değerini alacaktır.