Bölüm 18

Kapsülleme (Encapsulation)

Kavram

Nesne Yönelimli Programlamada, *kapsülleme* eylemi, verinin veya metodun başka yerden görünmeyecek biçimde üstünün sınıf içinde örtülmesi demektir. Genellikle, kapsüllenmiş nesneye soyut veri tipi (abstract data type) denilir.

Gerekseme

Aslında, kapsülleme yapmamızın nedeni her programcının yaptığı basit yanlışlarla verinin veya kodların bozulmasını önlemektir. Başka bir deyişle, duyarlı verilerimizi, onu işleyen metotlrla bir araya getirip paketleriz ve yanlışlıkla dışarıdan erişimi engelleriz.

Korumak istediğimiz verileri `public` nitelemek yerine `private` niteleriz. Private nitelenmiş veriler direkt olarak ancak iki yolla işlenebilir. Birinci yöntem alışılmış `accessor` ve `mutator` (get/set) metotlarını kullanmaktır. Öteki yöntem ise `özgen` (property) kullanmaktır. Hangisini kullanırsak kullanalım, verilerimiz güvende olacaktır.

Şimdi bu yöntemleri örnekler üzerinde görelim. Aşağıdaki örnekte, Mutator (set metodu) ve Accessor (get metodu) , sırasıyla, `private` nitelenmiş department `özgen`'ine değer atar ve atanmış değeri okurlar.

```
using System;
public class Bölüm
{
    private string bölümAdı;
```

```
    // Accessor (Getter)
    public string GetBölümAdı()
    {
        return bölümAdı;
    }
    // Mutator (Setter).
    public void SetBölümAdı(string a)
    {
        bölümAdı = a;
    }
}

class Uygulama
{
    public static void Main(string[] args)
    {
        Bölüm d = new Bölüm();
        d.SetBölümAdı("MUHASEBE");
        Console.WriteLine("Bölüm adı  :" + d.GetBölümAdı());

        //private bölümAdı'na nesne ile erişilemez
        Bölüm e = new Bölüm();
        // e.bölümAdı = "ARAŞTIRMA"; // Hata
        // Console.WriteLine("Bölüm adı  :" + e.bBölümAdı()); // Hata
    }
}
```

Bu örnekte, sınıfa ait bir nesne ile private nitelemeli `bölümAdı` alanına ulaşamayız. Ona değer atamak için `SetBölümAdı` metodunu, atanan değeri okumak için `GetBölümAdı()` metodunu kullanıyoruz. Bu iki metot accessor (getter) ve mutator (setter) adlarıyla da anılır.

Özgen (property) Kullanılarak Kapsülleme Yapma

Özgen kavramı C# ile ortaya konan yeni bir programlama kavramıdır. Şimdilik, bu özellik ancak bir kaç dil tarafından destekleniyor. Özgen'in görevi alana veri yazmak ve yazılan veriyi okumaktır. Alana başka türlü erişilemediği için, oradaki veri *kapsüllenmiş* sayılır. Yukarıda anlattığımız yöntem de bu işi iyi yapar. Ama C# bunu daha nezih bir yolla yapmaktadır. Bunu bir örnek üzerinde gösterelim.

Kapsülleme01.cs

```
using System;
public class Fakülte
{
    private string bölüm;
    public string Bölüm
    {
        get
        {
            return bölüm;
        }
        set
        {
            bölüm = value;
        }
    }
```

```
}
public class Uygulama
{
    public static void Main(string[] args)
    {
        Fakülte  d = new Fakülte();
        d.bölüm = "Matematik";
        Console.WriteLine("Fakülte :{0}", d.Bölüm);
    }
}
```

Çıktı

```
Error    1    'Fakülte.bölüm' is inaccessible due to its protection level ...
```

Görüldüğü gibi, program sınıfa ait bir özgene atanan değeri okuyamıyor; yani ona dışarıdan erişilemiyor. Bu programı çözümleyerek özgen ile kapsüllemenin yapılışını öğreneceğiz. Özgen'in iki metodu var: `Get()` ve `Set()`. `Get()` metodu alandaki veriyi getirir. `Set()` metodu alana veri yollar; yani değişkene değer atar. Bu işi yaparken, C# dilinin bir anahtar sözcüğünü kullanır: "`value`". Eğer `Set()` metodu konulmazsa, özgen *yalnız-okunur* (read-only) kılınmış olur.

Yalnız-okunur Özgen (Read Only Property)

Kapsülleme02.cs

```
using System;
public class Fakülte
{
    private string bölüm;
    public Fakülte(string str)
    {
        bölüm = str;
    }
    public string Bölüm
    {
        get
        {
            return bölüm;
        }
    }
}
public class Uygulama
{
    public static void Main(string[] args)
    {
        Fakülte obj = new Fakülte("Ekonomi");
        Console.WriteLine("Bölüm: {0}", obj.Bölüm);
        //obj.bölüm = "ist";
    }
}
```

Bu program, sınıfa ait `bölüm` adlı özgenin değerini okur; yani ona erişebilir. Ama ona değer atayamaz. Sınıfın tek parametreli bir kurucusu var:

```
    public Fakülte(string str)
    {
        bölüm = str;
```

```
    }
```

`Main()` metodu bu kurucuyu

```
Fakülte obj = new Fakülte("Ekonomi");
```

deyimi ile çağırınca `Fakülte` sınıfının `obj` adlı bir nesnesi kurulur ve

```
bölüm = "Ekonomi";
```

ataması yapılır.

```
Console.WriteLine("Bölüm: {0}", obj.Bölüm);
```

deyimi atanmış olan bu değeri konsola yazar.

```
    public string Bölüm
    {
        get
        {
            return bölüm;
        }
    }
```

blokuna bakarsak, Bölüm özgeni yalnızca `get` metodunu içeriyor. Bu metot bölüm değişkeninin değerini veriyor. `set` metodu olmadığı için, bölüme bir değer atanamaz. Gerçekten, programda etkisiz kılınan

```
//obj.bölüm = "ist";
```

deyimindeki // simgelerini kaldırıp deyimi etkili kılarsak, derleyici şu hata iletisini verecektir.

```
Error   1   'Fakülte.bölüm' is inaccessible due to its protection level ...
```

Yalnız-yazılır Özgen (Write Only Property)

Yalnız-okunur özgene benzer olarak *yalnız-yazılır özgenler* de yaratılabilir. Bunu yapmak için `get/set` metot ikilisinden yalnızca `set` metodunu kullanmak yetecektir. Aşağıdaki program o işi yapıyor.

```
using System;
public class Fakülte
{
    private string bölüm;
    public string Bölüm
    {
        set
        {
            bölüm = value;
            Console.WriteLine("Bölüm :{0}", bölüm);
        }
    }

    public class Uygulama
    {
        public static void Main(string[] args)
        {
            Fakülte d = new Fakülte();
            d.bölüm = "Biyoloji";
        }
    }
}
```