Bölüm 11

Erişim Belirteçleri

(Access Modifiers)

Erişim Belirteci Nedir?

C# dili nesne yönelimli bir dil olduğundan, her şey sınıflar içinde tanımlanır. Sınıflara ve sınıf öğelerine erişim kısıtlanabilir ya da belli düzeylerde erişime izin verilebilir. Öğelere erişimi kıstlayan ya da yetki veren anahtar sözcüklere *Erişim Belirteçleri* (access modifiers) denir.

Erişim belirteçleri, bir sınıfa ya da bir sınıfa ait öğelere erişilebilme kısıtlarını veya yetilerini belirleyen anahtar sözcüklerdir. Esas olarak dört tane erişim belirteci vardır. Aşağıdaki tablonun beşinci satırında yazılan belirteç bir bileşiktir.

Erişim Belirteçleri

public	Erişim kısıtı yoktur; her yerden erişilir
protected	Ait olduğu sınıftan ve o sınıftan türetilen sınıflarından erişilebilir

internal	Etkin projeye ait sınıflardan erişilebilir, onların dışından erişilemez
private	Yalnız bulunduğu sınıftan erişilir, dıştaki sınıflardan erişilemez
protected internal	Etkin projeye ait sınıflardan ve onların türevlerinden erişilebilir

`Bir öğe`, protect internal hariç, öteki erişim belirteçlerinden yalnız birisini alabilir.

`namespace` erişim belirteci almaz; çünkü o daima `public` nitelemelidir.

`Sınıflar` public ya da internal nitelemesini alabilirler; ama protected ile private nitelemesi alamazlar.

`enum` erişim belirteci almaz; çünkü o daima `public` nitelemelidir.

`Bir öğe`, bildirildiği ortama bağlı olarak, yalnız izin verilen erişim belirteçlerini alabilir. Eğer erişim belirteci almamışsa, öndeğer (default) belirteç etkin olur.

Başkalarının içinde yuvalanmamış üst-düzey tipler ancak `internal` ve `public` nitelemesini alabilirler. Bu tipler için öndeğer (default) niteleme `public`'tir.

Oğul sınıflar, doğal olarak *ata*'nın niteleme kısıtlarına tabidir. İstenirse daha çok kısıt konabilir; ama kısıtlar azaltılamaz.

`İç-içe` yuvalanmış tipler aşağıdaki tabloda gösterilen erişim nitelemelerini alabilirler.

Aidiyet	Öndeğer (default erişim belirteci	Nitelenebilen erişim belirteci
enum	public	Hiç
class	private	public protected internal private protected internal
interface	public	Hiç
struct	private	public internal private

Aşağıdaki örnek public ve private öğelere erişimi göstermektedir.

Access01.cs

```
// Public ve private erişim
```

```
using System;

class Ev
{
    private int oda;    // private nitelemeli
    int dolap;          // doğal private erişimli
    public int kapı;    // public erişimli

    // oda ve dolap Ev sınıfından erişilebilir
    public void setOda(int a)
    {
        oda = a;
    }

    public int getOda()
    {
        return oda;
    }

    public void setDolap(int a)
    {
        dolap = a;
    }

    public int getDolap()
    {
        return dolap;
    }
}

public class Uygulama
{
    public static void Main()
    {
        Ev birEv = new Ev();

    /* oda ve dolap değişkenlerine

         ancak metotlarla erişilir */

      birEv.setOda(5);

      birEv.setDolap(7);

      Console.WriteLine("birEv.oda   : " + birEv.getOda());

      Console.WriteLine("birEv.dolap : " + birEv.getDolap());

      //  oda ve dolap değişkenlerine aşağıdaki gibi erişilemez:

      //  birEv.oda = 10;    // Hata! oda  private'dir!

      //  birEv.dolap = 9;   // Hata! dolap  private'dir!
```

```
        // kapı public olduğu için doğrudan erişilir.

        birEv.kapı = 8;

        Console.WriteLine("birEv.kapı  : " + birEv.kapı);
    }
}
```

Aşağıdaki örnek `protected` nitelemeli öğelere oğul'dan erişilebildiğini göstermektedir.

Access02.cs

```
/*
 * protected nitelemeli öğelere sınıf içinden ve
 * oğul'dan erişilebilir.
*/
using System;

class Ata
{
    protected int m, n;

  // m ile n Ata ye göre private gibidir,

  // ama onlara Oğul 'den erişilebilir

    public void Set(int a, int b)
    {
        m = a;
        n = b;
    }

    public void Göster()
    {
        Console.WriteLine(m + " " + n);
    }
}

class Oğul : Ata
{
    int çarpım; // private

    // Oğul den Ata'deki m ve n ye erişilir
    public void Set()
    {
        çarpım = m * n;
    }

    public void Yaz()
    {
        Console.WriteLine(çarpım);
    }
}

public class ProtectedDemosu
{
```

```
    public static void Main()
    {
        Oğul birD = new Oğul();

        birD.Set(2, 3);  // Oğul 'den erişiliyor
        birD.Göster();   // Oğul 'den erişiliyor

        birD.Set();      // Oğul 'nin öğesi
        birD.Yaz();      // Oğul 'nin öğesi
    }
}
```

Aşağıdaki programdaki `Hesap` sınıfı birisi `public`, ötekisi `private` nitelemeli iki metot içeriyor. Uygulama sınıfındaki `Main()` metodu her ikisini çağırıyor. Ama derleyici `private` olanın çağrılmasına izin vermiyor. Program derleme hatası veriyor.

Access03.cs

```
using System;
class Uygulama
{
    static void Main()
    {
        Hesap obj = new Hesap();
        System.Console.WriteLine("2 + 3 = {0}", obj.Topla(2, 3));
        //Bu deyim derlenemez!!
        System.Console.WriteLine("3 - 2 = {0}", obj.Çıkar(3, 2));
    }
}

class Hesap
{
    public long Topla(int a, int b)
    {
        return a + b;
    }

    private long Çıkar(int c, int d)
    {
        return c - d;
    }
}
```

```
Error   1   'Hesap.Çıkar(int, int)' is inaccessible due to its protection level..
```

Şimdi bu basit programı düzeltip (debug) çalışır duruma getirmek için neler yapabileceğimizi düşünelim. Bunları düşünüp denedikçe, erişim belirteçlerinin işlevlerini daha iyi kavrayacağız.

1. Derleyicinin itiraz ettiği erişim kısıtını kaldırabiliriz. `Çıkar()` metodunun nitelemesini `private` yerine `public` yaparsak, programın derlendiğini ve çalıştığını görebiliriz.

```
    public long Çıkar(int c, int d)
```

2. `internal` nitelemesi etkin olan programa ait sınıflardan (assembly) erişilmesine izin verdiğine göre, `Çıkar()` metodunun nitelemesini `private` yerine `internal` yaparsak, programın derlendiğini ve çalıştığını görebiliriz.

```
internal long Çıkar(int c, int d)
```

3. `protected internal` nitelemesi etkin projeye ait sınıflardan ve onların türevlerinden erişilmesine izin verdiğine göre, `Çıkar()` metodunun nitelemesini `private` yerine `protected internal` yaparsak, programın derlendiğini ve çalıştığını görebiliriz.

```
protected internal long Çıkar(int c, int d)
```

4. `protected internal` nitelemesi etkin projeye ait sınıflardan ve onların türevlerinden erişilmesine izin verdiğine göre, `Çıkar()` metodunun nitelemesini `private` yerine `protected internal` yaparsak, programın derlendiğini ve çalıştığını görebiliriz.

```
protected internal long Çıkar(int c, int d)
```

5. `protected internal` nitelemesi yalnızca ait olduğu sınıftan ve o sınıftan türetilen sınıflarından erişilmesine izin verdiğine göre, `Çıkar()` metodunun nitelemesini `private` yerine `protected` yaparsak, programın derlenemediğini görürüz.

```
protected long Çıkar(int c, int d)
```

6. `private` nitelemeli öğelere aynı sınıf içinden erişilebildiğini biliyoruz. O halde, `Main()` metodunu `Çıkar()` metoduyla aynı sınıfa koyarsak, programımız çalışır duruma gelecektir:

```
using System;

class Hesap
{
    public long Topla(int a, int b)
    {
        return a + b;
    }

    private long Çıkar(int c, int d)
    {
        return c - d;
    }

    static void Main()
    {
        Hesap obj = new Hesap();
        System.Console.WriteLine("2 + 3 = {0}", obj.Topla(2, 3));
        //Bu deyim derlenenir!!
        System.Console.WriteLine("3 - 2 = {0}", obj.Çıkar(3, 2));
    }
```

```
}
```

7. `Main()` metodunu `Çıkar()` metodunu içeren `Hesap` sınıfının bir alt sınıfına koyarsak, programımız çalışmaya devam edecektir:

```
using System;

class Hesap
{
    public long Topla(int a, int b)
    {
        return a + b;
    }

    private long Çıkar(int c, int d)
    {
        return c - d;
    }

    class Uygulama
    {
        static void Main()
        {
            Hesap obj = new Hesap();
            System.Console.WriteLine("2 + 3 = {0}", obj.Topla(2, 3));
            //Bu deyim derlenemez!!
            System.Console.WriteLine("3 - 2 = {0}", obj.Çıkar(3, 2));
        }
    }
}
```

Alıştırma

Aşağıdaki programda `Uygulama` sınıfındaki `Main()` metodu `AAA` sınıfındaki `bbb()` metodunu çağırmak istiyor. Ancak derleyici `bbb()` metoduna erişilemediği için hata iletisini veriyor.

```
using System;

class Uygulama
{
    public static void Main()
    {
        AAA.bbb();
    }
}
class AAA
{
    static void bbb()
    {
        System.Console.WriteLine("AAA.bbb()");
    }
```

```
    public static void ccc()
    {
        System.Console.WriteLine("AAA.ccc()");
        bbb();
    }
}
```

```
Error   1       'AAA.bbb()' is inaccessible due to its protection level ...
```

Programı debug edip çalışır duruma getirmek için aşağıdakilerden hangisi programı çalıştırır? Size göre, programı çalıştıran seçeneklerden hangisi ne zaman en uygundur?

1. `bbb()` metoduna public nitelemesini ekleriz:

```
static public void bbb()
```

2. `bbb()` metoduna internel nitelemesini ekleriz:

```
static internal void bbb()
```

3. `AAA` sınıfına ait bir nesne yaratıp, `bbb()` metodunu nesne içinden çağırırız. (Bunun için gerekli kodları yazınız.)

4. `Main()` metodunu `AAA` sınıfı içine alırız.

5. `Uygulama` sınıfını `AAA` sınıfı içine alırız.

6. `AAA` sınıfını `Uygulama` sınıfı içine alırız.

7. `bbb()` metodunun bildirimini `Uygulama` sınıfı içine alırız.

8. `bbb()` metodunun bildirimini `Main()` metodu içine alırız.