Bölüm 21

Delegeler (delegates)

`Delegeler` metot işaret eden referanslardır. Bunun ne anlama geldiğini örneklerle açıklayacağız.

Bir metodu çağırmak için, hiç bir aracıya (delege) gerekseme duymayan iyi bir yöntemimiz vardır. Şimdiye kadar yapageldiğimiz yöntemi aşağıdaki örnek üzerinde anımsayalım.

Aracısız Metot Çağırma

Delege01.cs

```
using System;

namespace Teknikler
{
    class Vergiler
    {
        public double BrütOku()
        {
            Console.WriteLine("Brüt Geliri Giriniz : ");
            return Double.Parse(Console.ReadLine());
        }

        public double GelirVergisi(double d)
        {
            if (d > 20000)
                return d * 0.40;
            else
                return d * 0.25;
```

```
        }
    }
    class Uygulama
    {
        public static void Main(string[] args)
        {
            Vergiler v = new Vergiler();
            Console.WriteLine(v.GelirVergisi(v.BrütOku()));
        }
    }
}
```

Bu programı çözümlemek kolaydır. `Vergiler` sınıfının iki metodu var. `BrütOku()` adlı metot, kullanıcının gireceği brüt geliri okuyor. `GelirVergisi` adlı metot ise, Brüt gelir 20000 den fazla ise %40 , değilse %25 gelir vergisi hesaplıyor. *Uygulama* sınıfındaki *Main()* metodu *Vergiler* sınıfının `v` adlı bir nesnesini yaratıyor. Sonra v nesnesi içinden `GelirVergisi()` metodunu çağırıyor. O da BrütOku() metodunu çağırıyor. Son satırda iç-içe yuvalanmış metotları çözümlemekte bir zorluk yoktur.

Şimdi bu metotları delegelerle çağırmayı öğreneceğiz. İlkönce delege bildirimini yapmalıyız.

Basit Bir Delege Bildirimi

Delege bildirimi metot imzası gibidir. Sözdizimi şöyledir:

```
delegate değer-bölgesi ad([parametreler]);
```

Açıklama

delegate	delege bildirimi için anahtar sözcük
değer-bölgesi	delegenin işaret edeceği metodun değer-bölgesi
ad	delegenin adı
parametreler	işaret edeceği metodun bağlı olduğu parametreler (varsa)

Örnekler:

```
public delegate void BasitDelege () ;
```

Bu bildirimde verilen *BasitDelege* adlı delege parametresiz ve *void* değer alan metotları işaret eder.

```
public delegate int DüğmeyeTıkla (object obj1, object obj2) ;
```

Bu bildirimde verilen `DüğmeyeTıkla` adlı delege object tipinden iki parametreye bağlı olan ve int değer alan metotları işaret eder.

Delege, hangi sınıftan hangi metodu çağıracağımızı bilmeden, çağrılacak metodun neye benzediğini belirlememizi sağlar. Biz, yalnızca, işaret etmesini istediğimiz metodun imzasını belirtiyoruz.

Delege bildirimi ve kullanımı üç aşamalıdır:

delegenin Bildirimi

Delegenin yaratılması

Delegenin çağrılması

Delege İle static Metot Çağırma

Aşağıdaki program delege ile metot çağırmanın çok basit bir örneğidir.

Delege02.cs

```
using System;

namespace Delegeler
{
    public delegate void BasitDelege();

    class Uygulama
    {
        public static void Topla()
        {
            Console.WriteLine("12 + 8 =" + (12+8));
        }

        public static void Main()
        {
            BasitDelege obj = new BasitDelege(Topla);
            obj();
        }
    }
}
```

Üçüncü satırda delege bildirimi yapılmıştır:

```
      public delegate void BasitDelege();
```

Uygulama sınıfındaki `Topla()` metodu 12+8 işlemini yapıp konsola yazmaktadır.

Main() metodunun ilk satırında delegeye ait obj adlı bir nesne yaratılmıştır. Bu nesne *Topla()* metodunu işaret etmektedir.

```
BasitDelege obj = new BasitDelege(Topla);
```

Son satırda ise yaratılan nesne çağrılmıştır. obj nesnesinin çağrılması demek, işaret ettiği metodun çağrılması demektir. *Obj* nesnesinin İşaret ettiği metot *Topla()* metodudur. Sonuç olarak, son satırdaki kod, *Topla()* metodunu çalıştırmaktadır.

Programda delegenin bildiriminin *Topla()* metodunun imzasına benzediğine dikkat ediniz.

```
public delegate void BasitDelege();
public static void Topla()
```

Her ikisinin değer kümesi `void` 'dir. Her ikisinin parametreleri yoktur. Bu demektir ki, delegenin değer kümesi ve parametreleri çağıracağı fonksiyonunkilerle aynı olmalıdır. Gerçekten, birisinin değer kümesi belirtecinde `void` yerine `int, string` gibi başka bir veri tipi koyarsanız, derleyici,

```
Error    1 'void Delegeler.Uygulama.Topla()' has the wrong return type...
```

hata mesajını verir. Bu mesaj, delege ile işaret edeceği `Topla()` metodunun değer kümelerinin uyuşmadığını söylemektedir. Benzer olarak, eğer delege bildiriminde `BasitDelege(int n)` biçiminde bir parametre koyarsanız, derleyici,

```
Error    1 No overload for 'Topla' matches delegate ...
```

hata mesajını verir. Bu mesaj, delege ile işaret edeceği `Topla()` metodunun parametrelerinin

uyuşmadığını söylemektedir.

Bu örnekte, `Topla()` metodu `static` nitelemeli olduğu için, onu, ait olduğu sınıfın bir nesnesini yaratmadan çağırabiliyoruz. Eğer `static` nitelemesini kaldırırsak, derleyici,

```
Error   1   An object reference is required for the non-static field, method, or property...
```

hata mesajını verir. Bu mesaj, static olmayan *Topla()* metodunun bellekteki adresini işaret eden bir referans gerektiğini söylemektedir. Başka bir deyişle, onu ancak ait olduğu `Uygulama` sınıfına ait bir nesneden çağırabiliriz. Aşağıdaki örnek bunu yapmaktadır.

Delege İle Dinamik Metot (member method) Çağırma

Delege03.cs

```
using System;

namespace Delegeler
{
    public delegate void BasitDelege();

    class Uygulama
    {
        public  void Topla()
        {
            Console.WriteLine("12 + 8 =" + (12+8));
        }

        public static void Main()
        {
            Uygulama u = new Uygulama();
            BasitDelege obj = new BasitDelege(u.Topla);
            obj();
        }
    }
}
```

İki programı satır satır karşılaştırarak, static nitelemesi kaldırılınca Topla() metodunun çağrılabilmesi için nasıl nesne yaratıldığını görünüz.

Delege İle Parametreli Metot Çağırma

Aşağıdaki programda parametreli delege ile parametreli metot çağrılmaktadır. Burada delege ile `Topla()` metodunun değer kümelerinin ve parametrelerinin aynı olduğuna dikkat ediniz. Son satırdaki `obj(12,8)` çağrısında kullanılan 12 ve 8 parametreleri `Topla()` metoduna geçmektedir.

Delege04.cs

```
using System;

namespace Delegeler
{
    public delegate void BasitDelege(int a, int b);

    class Uygulama
    {
```

```
        public  void Topla(int x, int y)
        {
            Console.WriteLine("{0} + {1} = {2} " , x ,y , x+y);
        }

        public static void Main()
        {
            Uygulama u = new Uygulama();
            BasitDelege obj = new BasitDelege(u.Topla);
            obj(12,8);
        }
    }
}
```

Aşağıdaki program, `void` yerine `int` koyarak yukarıdaki programı değiştirmiştir. Bunun için `Topla()` metoduna bir `return` değerini koymak yetmiştir.

Delege05.cs

```
using System;

namespace Delegeler
{
    public delegate int BasitDelege(int a, int b);

    class Uygulama
    {
        public  int Topla(int x, int y)
        {
            Console.WriteLine("{0} + {1} = {2} " , x ,y , x+y);
            return x + y;
        }

        public static void Main()
        {
            Uygulama u = new Uygulama();
            BasitDelege obj = new BasitDelege(u.Topla);
            obj(12,8);
        }
    }
}
```

Aşağıdaki programdaki `Delege01` adlı delege, `Vergiler` sınıfında tanımlanan `BrütOku` metodunu çağırmaktadır. Çağrılan metot, kullanıcının girdiği `double` değerini almaktadır. Aynı sınıfta tanımlı olan `GelirVergisi()` metodu `Main()` metodu tarafından ne doğrudan ne de delege ile çağrılmaktadır.

Delege06.cs

```
using System;

namespace Teknikler
{
    public delegate double Delege01();
    class Vergiler
    {
```

```
        public double BrütOku()
        {
            Console.WriteLine("Brüt Geliri Giriniz : ");
            return Double.Parse(Console.ReadLine());
        }

        public double GelirVergisi(double d)
        {
            if (d > 20000)
                return d * 0.40;
            else
                return d * 0.25;
        }
    }
    class Uygulama
    {
        public static void Main(string[] args)
        {
            Vergiler v = new Vergiler();
            Delege01 obj01 = new Delege01(v.BrütOku);
            Console.WriteLine(obj01());
        }
    }
}
```

Delege01 için yapılan aşamalar şunlardır:

Üçüncü satırda delege bildirimi yapılmıştır:

```
public delegate double Delege01();
```

Main() metodunun birinci satırı Vergiler sınıfının v adlı bir nesnesini yaratmıştır:

```
Vergiler v = new Vergiler();
```

Main() metodunun ikinci satırı Delege01 delegesinin obj01 adlı bir nesnesini yaratmıştır. Bu nesne v.BrütOku() metodunu işaret etmektedir:

```
Delege01 obj01 = new Delege01(v.BrütOku);
```

Main() metodunun son satırında obj01 nesnesinin işaret ettiği v.BrütOku metodu çağrılmıştır.

```
Console.WriteLine(obj01());
```

Şimdi, Vergiler sınıfındaki iki metodu çağıran birer delege oluşturalım. Delegelerin birisi parametresiz, diğeri double tipinden bir parametrelidir.

Delege07.cs

```
using System;

namespace Teknikler
{
    public delegate double Delege01();
    public delegate double Delege02(double dd);

    class Vergiler
    {
```

```
        public double BrütOku()
        {
            Console.WriteLine("Brüt Geliri Giriniz : ");
            return Double.Parse(Console.ReadLine());
        }

        public double GelirVergisi(double d)
        {
            if (d > 20000)
                return d * 0.40;
            else
                return d * 0.25;
        }
    }
    class Uygulama
    {
        public static void Main(string[] args)
        {
            Vergiler v = new Vergiler();
            Delege01 obj01 = new Delege01(v.BrütOku);
            Delege02 obj02 = new Delege02(v.GelirVergisi);
            Console.WriteLine(obj02(obj01()));
        }
    }
}
```

Bu programı çözümleyelim.

Üçüncü ve dördüncü satırda iki tane delege bildirimi yapılmıştır:

```
    public delegate double Delege01();
    public delegate double Delege02(double dd);
```

Vergiler sınıfında `BrütOku()` ve `GelirVergisi()` adlı iki metot tanımlanmıştır.

Main() metodunun birinci satırı *Vergiler* sınıfının `v` adlı bir nesnesini yaratmıştır:

```
Vergiler v = new Vergiler();
```

Main() metodunun ikinci satırı Delege01 delegesinin obj01 adlı bir nesnesini yaratmıştır. Bu nesne v.BrütOku() metodunu işaret etmektedir:

```
Delege01 obj01 = new Delege01(v.BrütOku);
```

Main() metodunun üçüncü satırı Delege02 delegesinin obj02 adlı bir nesnesini yaratmıştır. Bu nesne v.GelirVergisi metodunu işaret etmektedir:

```
Delege02 obj2 = new Delege02(v.GelirVergisi);
```

Main() metodunun son satırında obj02 nesnesi obj01 nesnesini parametre olarak almıştır; dolayısıyla v.BrütOku metodu çağrılmıştır. Onun getirdiği brüt miktar obj02 nesnesinin işaret ettiği GelirVergisi metoduna parametre olarak geçmiş ve istenen vergi hesabı yapılmıştır.

```
Console.WriteLine(obj02(obj01()));
```

Alıştırma

Aşağıdaki çok basit bir delege kullanımı örneğidir. Programı satır satır çözümleyiniz.

Delege08.cs

```
using System;
namespace Delegeler
{
    /*
     * Bu delege int tipi iki parametresi olan ve int
     * değer alan her metodu işaret edebilir
*/
    public delegate int Delegem(int x, int y);

    //Bu sınıf Delegem tarafından işaret edilebilecek metotlar içeriyor

    public class Hesap
    {
        public static int Topla(int a, int b)
        {
            return a + b;
        }
        public static int Çarpım(int a, int b)
        {
            return a * b;
        }
    }

    class Uygulama
    {
        static void Main(string[] args)
        {
            //Hesap.Topla() metodunu işaret eden Delegem nesnesi yarat
            Delegem delege1 = new Delegem(Hesap.Topla);

            //Delegate kullanarak  Topla() metodunu çağır
            int toplam = delege1(7, 8);
            Console.WriteLine("7 + 8 = {0}\n", toplam);

            // Hesap.Çarpım() metodunu işaret eden Delegem nesnesi yarat
            Delegem delege2 = new Delegem(Hesap.Çarpım);

            //Delegate kullanarak Çarpım() metodunu çağır
            int çarpım = delege2(7, 8);
            Console.WriteLine("7 X 8 = {0}", çarpım);
        }
    }
}
```

Çıktı

```
7 + 8 = 15

7 X 8 = 56
```