FONKSİYONLAR

Fonksiyonlarda Erişim Belirleyicileri

C# programlama dilinde fonksiyonlar, sınıflar içerisinde yer alırlar.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Main()
        {// Ana fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
}
```

Yukarıda, **Uzay1** isim uzayı içerisinde yer alan **Sinif1** isminde bir sınıf yer almaktadır. Bu sınıfın içerisinde ise **Main** isminde bir fonksiyon bulunmaktadır. Bu fonksiyon incelendiğinde:

- **Argüman almamaktadır.** Argüman alan fonksiyonlarda, fonksiyon isminin hemen sağında yer alan parantezlerin arasına, fonksiyonun aldığı argümanlar tip bilgileri ile birlikte yazılırlar. Argüman almayan fonksiyonlarda ise bu parantezlerin arası boş bırakılır.
- **Değer döndürmemektedir.** Fonksiyon isimlerinin hemen solunda, o fonksiyonun döndürdüğü değerin tipini belirten bir deyim bulunur. Değer döndürmeyen fonksiyonlarda bu deyim, "**void**" olarak kullanılmaktadır.
- **İçinde bulunduğu sınıfın nesneleri üzerinde kullanılmak üzere hazırlanmış bir fonksiyon olmaktan çok, bu sınıf içerisinde bulunan ve sınıf dışından çağrılmak üzere hazırlanmış bir fonksiyondur.** Fonksiyonun tanımlandığı satırın başında bulunan **static** deyimi, bir erişim belirleyicisidir ve bu fonksiyonun sınıf dışından çağrılabileceği anlamını taşımaktadır. **static** erişimli fonksiyonlar, aynı sınıf içerisinden de çağrılabilirler.

Programın başlangıç noktası olan ana fonksiyonlar, "**Main**" olarak isimlendirilirler ve erişimleri **static** olmalıdır. Bir sınıfın içerisinde tek bir fonksiyon bulunabileceği gibi, birden çok sayıda fonksiyon da bulunabilir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        void Main()
        {// hata : ana fonksiyon statik erisimli olmalidir
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
}
```

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        public void Main()
        {// hata : ana fonksiyon statik erisimli olmalidir
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
}
```

Yukarıdaki iki program derlenmeyecektir. Derleyici, programın giriş noktası olarak **statik** erişimli ve "**Main**" olarak isimlendirilmiş ana fonksiyona ihtiyaç olduğunu belirtecektir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Fonksiyon1()
        {
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
        static void Main()
        {// ana fonksiyon
            // Fonksiyon1' in cagrilmasi
            Fonksiyon1();
        }
    }
}
```

Yukarıdaki programda, **Sinif1** sınıfı içerisinde **Fonksiyon1** ve **Main** olmak üzere iki farklı fonksiyon bulunmaktadır. **Fonksiyon1**, argüman almayan ve değer döndürmeyen **statik** erişimli bir fonksiyondur. **Main** fonksiyonu da benzer şekilde argüman almayan ve değer döndürmeyen **statik** erişimli bir fonksiyondur, ek olarak programın giriş kapısı olma görevini üstlenmiştir. Burada **Fonksiyon1**, **Main** (ana fonksiyon) içerisinden çağrılmıştır.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        public void Fonksiyon1()
        { // statik olmayan fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
        static void Main()
        {
            // hata : nesne yaratilmadan cagrilan fonksiyon
            // statik erisimli olmalidir
            Fonksiyon1();
        }
    }
}
```

Yukarıdaki program derlenmeyecektir. Çünkü bir sınıfta bulunan ve statik olmayan bir fonksiyonu çağırabilmek icin o sinifin bir nesnesinin yaratılması ve bu nesne aracılığı ile çağrılması gerekir. Bu durum, çağıran ve çağrılan fonksiyonlar aynı sınıf içerisinde olduğunda dahi geçerlidir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        public void Fonksiyon1()
        { // statik olmayan fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
        static void Main()
        { // Sinif1 sinifinin bir nesnesi : snf1
            Sinif1 snf1 = new Sinif1();
            snf1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki programda ise **Fonksiyon1** fonksiyonu statik olmayan bir fonksiyondur. **Main** fonksiyonunun içerisinde ise önce **Fonksiyon1** fonksiyonunun (ve de **Main** fonksiyonunun) içerisinde bulunmakta olduğu **Sinif1** sınıfının "**snf1**" adında bir nesnesi yaratılmıştır. Sonrasında ise **Sinif1** sınıfının statik erişimli olmayan bu fonksiyonuna, yaratılmış olan bu **snf1** nesnesi üzerinden (bu nesneye mesaj göndererek) ulaşılmıştır.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Fonksiyon1()
        { // statik erişimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
        static void Main()
        {
            // hata : statik erişimli bir fonksiyona nesne
            // üzerinden erişim yapılamaz
            Sinif1 snf1 = new Sinif1();
            snf1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki program, derleme anı hatası verecektir. **Main** fonksiyonu içerisinde **Sinif1** sınıfına ait bir nesne yaratılarak bu nesne üzerinden **Fonksiyon1** fonksiyonu çağrılmıştır ancak **Fonksiyon1** statik erişimli bir fonksiyon olduğundan buna nesne üzerinden erişmek mümkün olmamaktadır.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Fonksiyon1()
        { // statik erişimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        { // hata : Fonksiyon1 bulunamiyor
            Fonksiyon1();
        }
    }
}
```

Yukarıdaki programda, aynı isim uzayı içerisinde bulunan iki farklı sınıf (**Sinif1** ve **Sinif2**) yer almaktadır. **Sinif1** içerisinde, statik erişimli bir fonksiyon olan **Fonksiyon1** bulunmaktadır. **Sinif2** içerisinde ise programın giriş kapısı olan **Main** fonksiyonu bulunmaktadır ve bu fonksiyon içerisinde **Fonksiyon1** çağrılmak istenmiştir. Ancak böylesi bir çağrıda çağrılan fonksiyonun sınıfı farklı olduğu için **Fonksiyon1** bulunamayacak, kod derlenmeyecektir. Görüldüğü gibi, fonksiyonların aynı isim uzayı içerisinde bulunuyor olmaları yeterli değildir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Fonksiyon1()
        { // statik erişimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        { // hata : Fonksiyon1 bulundu ama erisim izni yok
            Sinif1.Fonksiyon1();
        }
    }
}
```

Bir sınıfta bulunan bir fonksiyon içerisinden, başka bir sınıf içerisinde yer alan herhangi bir fonksiyona erişmek mümkündür. Eğer her iki sınıf da aynı isim uzayında yer alıyorsa, çağrılmak istenen fonksiyonun isminden önce nokta konarak bu noktanın soluna çağrılacak olan fonksiyonun yer aldığı sınıfın adı yazılmalıdır. Yukarıdaki programda **Fonksiyon1**' e ulaşılmak istenirken kullanılan "**Sinif1.Fonksiyon1**" ifadesi doğrudur ancak sınıflar farklı olduğu için, **Fonksiyon1**' in diğer sınıflar tarafından da erişilebilen bir fonksiyon olduğu ilan edilmeli, yani başına "**public**" erişim belirticisi konmalıdır. Program, bu haliyle derlenmeyecektir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        public static void Fonksiyon1()
        { // statik erişimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        { // dogru kullanim
            Sinif1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki program sorunsuz bir biçimde çalışacaktır. **Fonksiyon1**, **Sinif1** dışındaki sınıflar içerisinden de erişilebilir nitelikte ve statik bir fonksiyondur.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        public void Fonksiyon1()
        { // statik olmayan fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        {
            // hata : Fonksiyon1 bulundu ancak cagirabilmek
            // icin Sinif1 sinifinin nesnesini yaratmak gerekiyor
            Sinif1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki programın çalışmamasının nedeni, statik erişimli olmayan **Fonksiyon1** fonksiyonunun, statik bir fonksiyonmuş gibi, **Sinif1** sınıfının nesnesi yaratılmaksızın çağrılmaya çalışılmasıdır. Böylesi bir erişim mümkün değildir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        public void Fonksiyon1()
        { // statik olmayan fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        { // dogru kullanim
            Sinif1 snf1 = new Sinif1();
            snf1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki programda hata yoktur. **Sinif1** sınıfı içerisinde statik olmayan ve diğer sınıflardan erişilebilecek ("**public**") bir biçimde tanımlanan **Fonksiyon1** fonksiyonu, **Sinif2** sınıfında bulunan **Main** fonksiyonu içerisinde **Sinif1** nesnesi yaratılarak çağrılmıştır.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        void Fonksiyon1()
        { // private erisimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        { // hatali : erisim hakki yok
            Sinif1 snf1 = new Sinif1();
            snf1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki program hatalıdır. **Sinif1** sınıfı içerisinde yer alan **Fonksiyon1** fonksiyonu, **Sinif2** içerisinde yer alan **Main** fonksiyonu içerisinden **Sinif1** nesnesinin yaratılması yolu ile çağrılmıştır. **Fonksiyon1** fonksiyonunun yanında herhangi bir erişim belirleyici ifade bulunmamaktadır. Böylesi bir durumda **Fonksiyon1** fonksiyonunun erişim belirleyicisi aslında "**private**" olacaktır yani bu fonksiyona, sadece **Sinif1** sınıfı içerisinde yaratılan **Sinif1** nesneleri aracılığı ile erişilebilecektir. **Fonksiyon1** fonksiyonunu şu şekilde gerçeklemek de aynı anlama gelecektir:

```
.......

    class Sinif1
    {
        private void Fonksiyon1()
        {
            .......
```

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected void Fonksiyon1()
        { // protected erisimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
     }
    class Sinif2
    {
        static void Main()
        { // hatali : erisim hakki yok
            Sinif1 snf1 = new Sinif1();
            snf1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki program hatalıdır. **Sinif1** sınıfında bulunan **Fonksiyon1** fonksiyonunun erişim belirleyicisi "**protected**" olarak verilmiştir. Bu, **Fonksiyon1**' e:

- **Sinif1** içerisinde yaratılan **Sinif1** nesneleri aracılığı ile

ya da

- **Sinif1**' den kalıtlayan (miras alan) **SinifX** sınıfı içinde yaratılan **SinifX** nesneleri aracılığı ile

erişilebileceğini göstermektedir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected void Fonksiyon1()
        { // protected erisimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
     }
    class Sinif2:Sinif1
    { // Sinif1 den kalitlayan (miras alan) Sinif2 sinifi
        static void Main()
        {
            // hatali : kalitlanan degil kalitlayan
            // sinifin nesnesi uzerinden ulasilmaliydi
            Sinif1 snf1 = new Sinif1();
            snf1.Fonksiyon1();
        }
    }
}
```

Yukarıdaki program hatalıdır. **Sinif1**' de yer alan "**protected**" erişimli **Fonksiyon1** fonksiyonuna **Sinif2** sınıfından erişilebilmesi için **Sinif2**' nin **Sinif1**' den kalıtlaması (miras alması) ve fonksiyon çağrısının, mirası kabul eden sınıfa (yani **Sinif2**' ye) ait bir nesne üzerinden yapılması gerekmektedir. Bu programda her ne kadar **Sinif2** sınıfı **Sinif1** sınıfından kalıtlıyor olsa da **Fonksiyon1** çağrısı, kalıtlayan sınıfa (yani **Sinif2**' ye) ait bir nesne üzerinden yapılmamış, kalıtlanan sınıfa (yani **Sinif1**' e) ait bir nesne üzerinden yapılmıştır. Aşağıdaki program bu yönden doğru tasarlanmıştır:

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected void Fonksiyon1()
        { // protected erisimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
     }
    class Sinif2:Sinif1
    { // Sinif1 den kalitlayan (miras alan) Sinif2 sinifi
        static void Main()
        {
            // kalitlayan sinifin nesnesi olusturuluyor
            Sinif2 snf2 = new Sinif2();
            snf2.Fonksiyon1();
        }
    }
}
```

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected void Fonksiyon1()
        { // protected erisimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
     }
    class Sinif2:Sinif1
    {
        // Sinif1 den kalitlayan (miras alan)
        // Sinif2 sinifi
        void BosSatir()
        {
            Console.WriteLine();
        }
    }
    class Sinif3
    {
        static void Main()
        {
            // Sinif2 sinifinin nesnesi
            Sinif2 snf2 = new Sinif2();
            snf2.Fonksiyon1();
        }
    }
}
```

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected void Fonksiyon1()
        { // protected erisimli fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
     }
    class Sinif2:Sinif1
    {
        // Sinif1 den kalitlayan (miras alan)
        // Sinif2 sinifi
        void BosSatir()
        {
            Console.WriteLine();
        }
    }
    class Sinif3:Sinif2
    {
        // Sinif3 den kalitlayan Sinif2 sinifi
        static void Main()
        {
            // kalitlayan sinifin nesnesi
            Sinif3 snf3 = new Sinif3();
            snf3.Fonksiyon1();
        }
    }
}
```

Yukarıdaki programlardan soldaki hatalı, sağdaki hatasızdır. Soldaki programda, **Sinif3** içinde bulunan **Main** fonksiyonu içinde, **Sinif2** sınıfının bir nesnesi yaratılmış ve **Sinif1** içinde bulunan, "**protected**" erişimli **Fonksiyon1** fonksiyonu çağrılmak istenmiştir. **Sinif2**, **Sinif1**' den kalıtlamaktadır ve fonksiyon çağrısı, kalıtlayan sınıf olan **Sinif2**' ye ait bir nesne aracılığı ile yapılmıştır. Ancak bu **Sinif2** nesnesi, **Sinif2** sınıfı içerisinde değil, **Sinif1** ve **Sinif2** ile hiçbir kalıtımsal ilişkisi bulunmayan **Sinif3** sınıfı içerisinde yaratılmıştır. Bu yüzden program hatalıdır.

Sağdaki programda ise **Sinif3** **Sinif2**' den, **Sinif2** ise **Sinif1**' den kalıtlamaktadır. **Sinif3** sınıfı içerisinde bir **Sinif3** nesnesi yaratılarak **Fonksiyon1** çağrıldığında, kalıtımsal ilişki sayesinde sorunsuz bir biçimde program çalışacaktır. Tabi, **Sinif3**' ün **Sinif2**' den kalıtlıyor olması bize **Sinif3** içinde **Sinif2** nesnesi yaratarak **Fonksiyon1**' e erişme hakkı vermeyecektir.

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected static void Fonksiyon1()
        { // protected erisimli statik fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2
    {
        static void Main()
        {
            // Sinif1 sinifinda bulunan protected
            // erisimli statik bir fonksiyon olan
            // Fonksiyon1 cagriliyor
            Sinif1.Fonksiyon1(); // hatali
        }
    }
}
```

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        protected static void Fonksiyon1()
        { // protected erisimli statik fonksiyon
            Console.WriteLine("Merhaba dunya!");
            Console.ReadLine();
        }
    }
    class Sinif2 : Sinif1
    { // Sinif1 den kalitlayan Sinif2 sinifi
        static void Main()
        {
            Sinif1.Fonksiyon1(); // dogru: Sinif1'e ait
            // alttakiler de dogru, Sinif2' ye aitler
            // Fonksiyon1(); // dogru
            // Sinif2.Fonksiyon1(); // dogru
        }
    }
}
```

Yukarıdaki programlardan soldaki hatalı, sağdaki hatasızdır. Soldaki programda, **Sinif1** sınıfında bulunan "**protected**" erişimli **statik** bir fonksiyon olan **Fonksiyon1**, **Sinif2** içerisinden statik fonksiyon çağrısı yapılarak çağrılmıştır ancak **Fonksiyon1**, erişim hakkının "**protected**" olmasından dolayı **Sinif1** içerisinden ya da **Sinif1**' den kalıtlayan herhangi bir sınıf içerisinden çağrılabilir.

Sağdaki programda **Sinif2** sınıfı **Sinif1**' den kalıtlamakta olduğu için bu programda, diğer programdaki hatalı gerçekleştirimin düzeltilmiş olduğundan bahsedilebilir.

Buraya kadarki örneklerde, erişim belirleyicilerinin fonksiyonlar ve fonksiyon çağrıları üzerindeki etkileri sınıf ilişkileri ile birlikte anlatılmış olup, örneklerde pratiklik açısından argüman almayan ve değer döndürmeyen fonksiyonlar kullanılmıştır. Anlatılan tüm özellikler, argüman alan ve/veya değer döndüren fonksiyonlar söz konusu olduğunda da birebir aynı olmaktadır.

Fonksiyonlarda Argümanlar

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Merhaba()
        { // arguman almaz, deger dondurmez
            Console.WriteLine("Merhaba !");
        }

        static void ToplaYaz(int a, int b)
        { // arguman alir, deger dondurmez
            int sonuc = a + b;
            Console.WriteLine("Toplam : {0}", sonuc);
        }

        static int ToplaDondur(int a, int b)
        { // arguman alir, deger dondurur
            int sonuc = a + b;
            return sonuc;
        }

        static int TurkiyeYuzolcumu()
        { // arguman almaz, deger dondurur
            int yuzolcum = 814578;
            return yuzolcum;
        }

        static void Main()
        {
            Merhaba();
            ToplaYaz(60, 35);

            int td = ToplaDondur(25, 19);
            Console.WriteLine("ToplaDondur fonk. sonucu : {0}", td);

            int ty = TurkiyeYuzolcumu();
            Console.WriteLine("TurkiyeYuzolcumu fonk. sonucu : {0}", ty);
            Console.ReadLine();
        }
    }
}
```

C# programlama dilinde fonksiyonların aldıkları argümanlar, fonksiyon adının sağında yer alan parantezler arasına, virgüllerle ayrılarak ve tipleri belirtilerek yazılırlar (yukarıdaki **ToplaYaz** ve **ToplaDondur** fonksiyonlarında olduğu gibi). Argüman almayan fonksiyonlarda bu parantezlerin içi boş bırakılır (**Merhaba** ve **TurkiyeYuzolcumu** fonksiyonlarında olduğu gibi). Fonksiyonların dönüş değerlerinin tipi ise fonksiyon isimlerinin sol tarafına yazılır. Örneğin "**integer**" tipinden değer döndüren bir fonksiyonun adının sol tarafına "**int**" yazılmalıdır. Değer döndürmeyen fonksiyonların dönüş tipi ise "**void**" olarak ifade edilir. Değer döndüren fonksiyon gerçekleştirimlerinin içerisinde döndürülmek istenen ve fonksiyonun döndürdüğü değer

tipiyle aynı olan değişken ya da değer, "**return**" deyiminin sağına yazılarak değer döndürme işlemi yapılır (**ToplaDondur** ve **TurkiyeYuzolcumu** fonksiyonlarında olduğu gibi).

Argüman alan fonksiyonların çağrıları da, fonksiyon isminden sonra açılan parantezler arasına uygun tipteki değerler doğru sırayla verilerek yapılır. Argüman almayan fonksiyonlarda bu parantezlerin içi boş bırakılır. Değer döndüren fonksiyonların döndürdükleri değerlerden yararlanmak için, fonksiyon çağrısının yapıldığı satırda fonksiyon adının hemen soluna "=" işareti, onun soluna da fonksiyonun döndürdüğü değerin türündeki bir değişkenin adı yazılarak yapılır. **Değer döndüren fonksiyonların döndürdükleri değerleri herhangi bir değişkene atmadan da bu fonksiyonları çağırmak mümkündür (değer döndürmeyen fonksiyonların çağrılması gibi). Bu durumda, fonksiyonun döndürdüğü değer ihmal edilmiş olacaktır; örnek:**

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static int ToplaYazdirDondur(int a, int b)
        { // arguman alir, deger dondurur
            int sonuc = a + b;
            Console.WriteLine("Toplam : {0}", sonuc);
            return sonuc;
        }

        static void Main()
        {
            ToplaYazdirDondur(60, 35);
            Console.ReadLine();
        }
    }
}
```

Yukarıdaki örnekte **ToplaYazdirDondur** fonksiyonu, **Main** içinde çağrılmış ancak döndürdüğü değer ihmal edilmiştir. Bu durumda **ToplaYazdirDondur** fonksiyonu normal bir biçimde çalışacak; aldığı argümanların toplamını ekrana yazacak ve toplamı döndürecektir ancak döndürmüş olduğu değer kullanılmamış olacaktır.

Fonksiyonların Sınıf Nesneleri Üzerinde Kullanılması

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        // Sinif1 sinifinin uyeleri
        public int a;
        public int b;

        public Sinif1()
        { // varsayilan yapici metot
            a = -1;
            b = -1;
        }
        public Sinif1(int sayi1, int sayi2)
        { // 2 argumanli yapici metot
            a = sayi1;
            b = sayi2;
        }
        public int Topla1()
        { // sinif ici kullanim amacli Topla1 fonksiyonu
            return (a + b);
        }
        public static int Topla2(int s1, int s2)
        { // sinif disi kullanim amacli Topla2 fonksiyonu
            return (s1 + s2);
        }
    }
    class Sinif2
    {
        static void Main()
        {
            // varsayilan yapici metotla nesne yaratma
            Sinif1 nesne1 = new Sinif1();

            // 2 argumanli yapici metotla nesne yaratma
            Sinif1 nesne2 = new Sinif1(4, 7);

            int toplam1 = nesne1.Topla1(); // -2
            int toplam2 = nesne2.Topla1(); // 11

            int toplam3 = Sinif1.Topla2(23, 45); // 68

            Console.WriteLine("toplam1 : {0}", toplam1);
            Console.WriteLine("toplam2 : {0}", toplam2);
            Console.WriteLine("toplam3 : {0}", toplam3);
            Console.ReadLine();
        }
    }
}
```

Yukarıdaki programda **Sinif1** sınıfının üyesi olarak tamsayı türünde **a** ve **b** değişkenleri, argüman almayan ve 2 argüman alan iki farklı yapıcı metot, sınıf nesneleri üzerinde kullanıma yönelik **Topla1** fonksiyonu ve nesne yaratmadan kullanıma yönelik **Topla2** fonksiyonu bulunmaktadır.

Sinif1 sınıfının bir nesnesi yaratıldığında bu nesnenin bünyesinde **a** ve **b** değişkenlerini ve statik olmayan **Topla1** fonksiyonunu barındırdığı düşünülebilir. Bir sınıfın yapıcı metotları (fonksiyonları), o sınıfın üyesi olan ve o sınıfa ait yeni bir nesne yaratıldığında bu nesnenin içerisinde yer alan değişkenlere ilk değerlerinin verilmesi amacı ile kullanılan fonksiyonlardır. Yapıcı metotlar, üyesi oldukları sınıfın adını taşıyan ve sınıfın bir nesnesi yaratıldığında bu nesnenin alt alanlarını oluşturan değişkenlerin ilk değerlerini belirleyen fonksiyonlar olarak düşünülebilir.

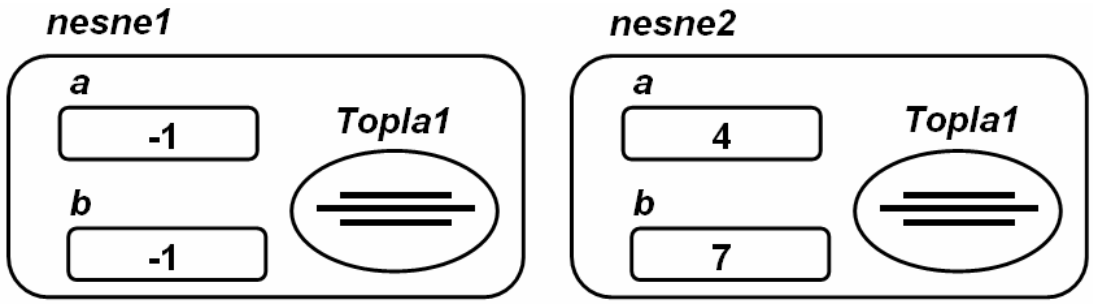
Örnek programda iki farklı yapıcı metot bulunmaktadır. Bunlardan biri hiç argüman almazken (varsayılan yapıcı metot *[default constructor]*) diğeri argüman olarak iki tamsayı almaktadır. Argüman almayan yapıcı metodu kullanmak için, **Sinif1**' e ait yeni bir nesne yaratılırken "**new Sinif1**" ifadesinin sağına (bitişik olarak) içi boş parantezler "**()**" yazılır. Bu durumda, verilen örnek için, yaratılan nesnenin **a** ve **b** alt alanlarının her ikisi de "-1" değerine sahip olacaktır. Örnek:

```
Sinif1 nesne1 = new Sinif1();
```

İki argüman alan yapıcı metot ise almış olduğu iki değeri, yaratılan nesnenin sırasıyla **a** ve **b** alt alanlarına ilk değer atamada kullanacaktır. Bu yapıcı metodu kullanmak için de, **Sinif1**' e ait yeni bir nesne yaratılırken "**new Sinif1**" ifadesinin sağına yazılan parantezlerin içerisine bu nesnenin sırasıyla **a** ve **b** alt alanlarının sahip olması arzu edilen değerler yazılacaktır. Örnek:

```
Sinif1 nesne2 = new Sinif1(4, 7);
```

Böylelikle, **Sinif1** sınıfına ait **nesne1** ve **nesne2** adında iki farklı nesne yaratılmış olacaktır. **nesne1** nesnesinin **a** ve **b** alt alanlarından her ikisi de **-1** değerini taşırken **nesne2** nesnesinin **a** ve **b** alt alanları sırasıyla **4** ve **7** değerlerine sahip olacaktır:



Sinif1 sınıfı içerisinde yer alan **Topla1** fonksiyonu, bu sınıfın nesneleri üzerinde kullanılmak için gerçeklenmiş, statik olmayan bir fonksiyondur. Bu fonksiyonu çağırmak için, **Sinif1**' e ait bir nesne adının sonuna nokta koyarak **Topla1** fonksiyonunu aşağıdaki gibi çağırmak gerekmektedir:

```
nesne1.Topla1();
nesne2.Topla1();
```

Topla1 fonksiyonunun gerçekleştirimine baktığımızda, bulunduğu sınıfın **a** ve **b** adlı ve tamsayı tipinde olan üyelerinin toplamını döndürdüğünü görmekteyiz. O halde bu fonksiyon, bulunduğu sınıfa ait bir nesneye mesaj gönderilerek çağrıldığında ise, bu nesnenin **a** ve **b** adlı alt alanlarının sahip olduğu değerlerin toplamını döndürecektir:

```
int toplam1 = nesne1.Topla1();
int toplam2 = nesne2.Topla1();
```

Yukarıdaki kutuda yer alan kodda, **Topla1** fonksiyonlarının döndürdüğü değerler **toplam1** ve **toplam2** değişkenlerine atılmaktadır.

Topla2 fonksiyonu ise statik bir fonksiyon olup, **Sinif1** sınıfının nesnelerine mesaj göndermek yolu yerine diğer sınıflardan erişim yolu ile kullanılmaktadır:

```
int toplam3 = Sinif1.Topla2(23, 45);
```

Eğer **Sinif1** sınıfında yer alan **Topla2** fonksiyonu statik olmasa idi,

```
.......
public int Topla2(int s1, int s2)
{ // statik olmayan Topla fonksiyonu
    return (s1 + s2);
}
.......
```

bu fonksiyona **Sinif1** sınıfının herhangi bir nesnesi üzerinden erişmek gerekecekti:

```
.......
Sinif1 nesne1 = new Sinif1();
.......
int toplam3 = nesne1.Topla2(23, 45);
.......
```

Ancak bu konuya nesne yönelimli tasarım penceresinden bakıldığında, **Topla2** fonksiyonunun **statik** olup – olmamasına karar vermek kolaylaşacaktır. **Topla2** fonksiyonu, kendisi **Sinif1** sınıfının bir üyesi olmasına rağmen bu sınıfın diğer üyeleriyle ilişkisi olmayan, sadece iki tamsayı argüman alarak bunların toplamını döndüren bir fonksiyondur. Dolayısı ile içinde bulunduğu sınıfın üyeleriyle ilişkisi olmayan, daha çok dış dünya ile (diğer sınıflar ile) etkileşim içerisinde olması beklenen bir fonksiyon tanımlanırken, bu fonksiyonun erişiminin **statik** olarak belirlenmesi, yani bu fonksiyonun kullanılabilmesi için bulunduğu sınıfın bir nesnesinin yaratılmasına lüzum kalmadan direkt sınıfın adı ile ulaşılabilir ve kullanılabilir kılınması, tasarım açısından izlenebilecek doğru bir yaklaşımdır.

Fonksiyonlara Argüman Olarak Referans Vermek

C# programlama dilinde fonksiyonlara argüman olarak, değer vermek gibi referans vermek de mümkündür. Önceki örneklerde yapılan fonksiyon çağrılarında, argüman alan fonksiyonlara argüman olarak değer verilmiştir. Bu durumda, fonksiyona dışarıdan verilen değişkenlerin değerlerinde herhangi bir değişiklik yapılmaz. Argüman olarak referans alan bir fonksiyon tanımlamak istediğimizde ise bu fonksiyonun aldığı argümanların tip belirticilerinin önüne **ref** deyimi koyulmalıdır. Böylesi bir fonksiyon çağrılırken de fonksiyona verilen argümanların önüne **ref** deyimi yazılır.

Çıktı:

```
En Basta                --> a = 3, b = 5
Takas_Deger(a, b)       --> a = 3, b = 5
Takas_Referans(a, b) --> a = 5, b = 3
```

```
using System;

namespace Uzay1
{
    public class Sinif1
    {
        public static void Takas_Deger(int s1, int s2)
        { // arguman olarak deger alan fonksiyon
            // hata: takas gerceklesmez
            int temp = s1;
            s1 = s2;
            s2 = temp;
        }
        public static void Takas_Referans(ref int s1, ref int s2)
        { // arguman olarak referans alan fonksiyon
            // takas gerceklesir
            int temp = s1;
            s1 = s2;
            s2 = temp;
        }
    }

    public class Sinif2
    {
        static void Main()
        {
            int a = 3, b = 5;
            Console.WriteLine("En Basta             --> a = {0}, b = {1}\n", a, b);

            Sinif1.Takas_Deger(a, b);
            Console.WriteLine("Takas_Deger(a, b)    --> a = {0}, b = {1}\n", a, b);

            Sinif1.Takas_Referans(ref a, ref b);
            // Sinif1.Takas_Referans(a, b); // hatali
            // Sinif1.Takas_Referans(ref a, ref 25); // hatali
            Console.WriteLine("Takas_Referans(a, b) --> a = {0}, b = {1}\n", a, b);

            System.Console.Read();
        }
    }
}
```

Yukarıdaki programda **Sinif1** sınıfı içerisinde statik erişimli ve değer döndürmeyen **Takas_Deger** ve **Takas_Referans** fonksiyonları yer almaktadır.

Takas_Deger fonksiyonu çağrıldığında, *int* tipindeki **s1** ve **s2** argümanları içerisinde, fonksiyona verilen değişkenlerin (**a** ile **b**) değerlerinin birer kopyasını tutmakta ve **s1** ile **s2** içerisinde tutulan değerleri birbiriyle takas etmektedir. Böylesi bir fonksiyon çağrıldığında fonksiyona argüman olarak verilen değişkenlerin değerleri üzerinde herhangi bir değişiklik yapılmamakta, fonksiyonun yerel değişkeni olarak da düşünülebilecek olan **s1** ve **s2** değişkenlerinin (aynı zamanda argümanlarının) değerleri değiştirilmektedir. Yani bu fonksiyonda **s1** ve **s2**, kendilerine ait verilerini bulunduran değer türünden birer değişken olarak düşünülebilir. Bu fonksiyon, **Sinif2** içerisinden *int* tipinden değişkenler olan **a** ve **b**' yi, kendisine ait olan **s1** ve **s2** değişkenlerine kopyalayarak tüm işlemleri **s1** ve **s2** üzerinde yapmaktadır.

Takas_Referans fonksiyonu ise birer *int referansı* olan **s1** ve **s2** argümanları içerisinde, fonksiyona verilen değişkenlerin (**a** ile **b**) adreslerini tutmakta (onları refere etmekte) olup, bu iki adreste tutulan değerleri birbiri ile takas etmektedir. Böylesi bir fonksiyon çağrıldığında, kendisine argüman olarak verilen değişkenlerin üzerinde değişiklik yapılması söz konusu olmaktadır.

Çıktı:

```
toplam degiskeninin ilk degeri : 0
toplam degiskeninin son degeri : 7
```

```
using System;

namespace Uzay1
{
    public class Sinif1
    {
        public static void Topla_Ref(ref int toplam, int sayi1, int sayi2)
        {
            toplam = sayi1 + sayi2;
        }
    }

    public class Sinif2
    {
        static void Main()
        {
            int a = 3, b = 4, toplam = 0;
            Console.WriteLine("toplam degiskeninin ilk degeri : {0}", toplam);

            Sinif1.Topla_Ref(ref toplam, a, b);
            Console.WriteLine("toplam degiskeninin son degeri : {0}", toplam);

            System.Console.Read();
        }
    }
}
```

Yukarıdaki program incelendiğinde, **Sinif1** içerisinde yer alan **Topla_Ref** fonksiyonunun hem değer, hem de referans türünden argüman aldığı görülecektir. Fonksiyon, değer türünden olan 2. ve 3. argümanları ile aldığı değerleri toplayarak, referans türünden olan 1. argümanı ile aldığı adrese kaydetmektedir.

Diziler de fonksiyonlara referans olarak verilebilirler:

Çıktı:

```
ONCE  : fib[2], fib[5] : 2, 8
SONRA : fib[2], fib[5] : 0, 0
```

```
using System;

namespace Uzay1
{
    public class Sinif1
    {
        public static void DiziSifirla(ref int[] dizi)
        {
            int uzunluk = dizi.Length;
            for (int i = 0; i < uzunluk; i++)
            {
                dizi[i] = 0;
            }
        }
    }
    public class Sinif2
    {
        static void Main()
        {
            int[] fib = { 1, 1, 2, 3, 5, 8, 11};

            Console.WriteLine("ONCE  : fib[2], fib[5] : {0}, {1}", fib[2], fib[5]);

            Sinif1.DiziSifirla(ref fib);
            Console.WriteLine("SONRA : fib[2], fib[5] : {0}, {1}", fib[2], fib[5]);

            System.Console.Read();
        }
    }
}
```

Fonksiyon Temsilcileri

C# programlama dilinde fonksiyonları temsil etmek (refere etmek) için "**delegate**" (temsilci) olarak adlandırılan sınıf üyelerinden yararlanılmaktadır. Temsilcilerin erişim alanlarının belirlenmesinde "public", "private" ve "protected" erişim belirleyicileri kullanılabilir.

Çıktı:

```
7 ile 9 un toplami : 16
7 ile 9 un carpimi : 63
```

```
using System;

namespace Uzay1
{
    public class Sinif1
    {
        public delegate int Temsilci1(int s1, int s2);
        // arguman olarak 2 tamsayi alan tum fonksiyonlari
        // temsil edebilecek bir temsilci

        public static int Fonksiyon1(int sayi1, int sayi2)
        { // Sinif1 e ait, toplama yapan  Fonksiyon1 fonk. (statik)
            return (sayi1 + sayi2);
        }

        public static int Fonksiyon2(int sayi1, int sayi2)
        { // Sinif1 e ait, carpma yapan  Fonksiyon2 fonk. (statik)
            return (sayi1 * sayi2);
        }
    }

    public class Sinif2
    {
        static void Main()
        {
            int sonuc1 = 0, sonuc2 = 0;

            // Sinif1' deki Temsilci1 tipinde 'topla' isminde bir temsilci
            // olustur ve bununla Sinif1' deki Fonksiyon1' i temsil et
            Sinif1.Temsilci1 topla = new Sinif1.Temsilci1(Sinif1.Fonksiyon1);
            // temsilciyi fonksiyon gibi kullan
            sonuc1 = topla(7, 9);
            Console.WriteLine("7 ile 9 un toplami : {0}\n", sonuc1);

            Sinif1.Temsilci1 carp = new Sinif1.Temsilci1(Sinif1.Fonksiyon2);
            sonuc2 = carp(7, 9);
            Console.WriteLine("7 ile 9 un carpimi : {0}", sonuc2);

            System.Console.Read();
        }
    }
}
```

Yukarıdaki programda, **Sinif1** içerisinde **Fonksiyon1** ve **Fonksiyon2** isminde, 2' şer tane tamsayı argüman alan iki farklı fonksiyon yer almakta olup, bu fonksiyonlardan ilki aldığı değerlerin toplamını döndürürken ikincisi ise çarpımını döndürmektedir. Ayrıca,

Temsilci1 isminde, **argüman olarak iki tamsayı alan herhangi bir fonksiyonu temsil edebilecek** nitelikte bir temsilci yer almaktadır.

Sinif2 içerisinde yer alan **Main** fonksiyonu içerisinde, ilk önce "**topla**" isminde ve **Sinif1**' deki **Temsilci1** temsilcisinin tipinde bir temsilci tanımlanmış ve bu temsilcinin, **Sinif1** içerisindeki **Fonksiyon1** fonksiyonunu temsil edeceği belirtilmiştir. Daha sonra **topla** temsilcisi tıpkı bir fonksiyon adı gibi kullanılarak, dolaylı yoldan **Fonksiyon1** fonksiyonu çağrılmıştır.

Sonrasında ise, önce "**carp**" isminde ve **Sinif1**' deki **Temsilci1** temsilcisinin tipinde başka bir temsilci tanımlanmış ve bu temsilcinin, **Sinif1** içerisindeki **Fonksiyon2** fonksiyonunu temsil edeceği belirtilmiştir. Daha sonra **carp** temsilcisi tıpkı bir fonksiyon adı gibi kullanılarak, dolaylı yoldan **Fonksiyon2** fonksiyonu çağrılmıştır.

```
using System;

namespace Uzay1
{
    public class Sinif1
    {
        public delegate int Temsilci1(int s1, int s2);
        // arguman olarak 2 tamsayi alan tum fonksiyonlari
        // temsil edebilecek bir temsilci

        public int Fonksiyon1(int sayi1, int sayi2)
        { // Sinif1 e ait, toplama yapan  Fonksiyon1 fonk.
            return (sayi1 + sayi2);
        }

        public int Fonksiyon2(int sayi1, int sayi2)
        { // Sinif1 e ait, carpma yapan  Fonksiyon2 fonk.
            return (sayi1 * sayi2);
        }
    }

    public class Sinif2
    {
        static void Main()
        {
            int sonuc1 = 0, sonuc2 = 0;
            Sinif1 Nesne1 = new Sinif1();

            // Sinif1' deki Temsilci1 tipinde 'topla' isminde bir temsilci
            // olustur ve bununla Sinif1' deki Fonksiyon1' i temsil et
            Sinif1.Temsilci1 topla = new Sinif1.Temsilci1(Nesne1.Fonksiyon1);
            // temsilciyi fonksiyon gibi kullan
            sonuc1 = topla(7, 9);
            Console.WriteLine("7 ile 9 un toplami : {0}\n", sonuc1);

            Sinif1.Temsilci1 carp = new Sinif1.Temsilci1(Nesne1.Fonksiyon2);
            sonuc2 = carp(7, 9);
            Console.WriteLine("7 ile 9 un carpimi : {0}", sonuc2);

            System.Console.Read();
        }
    }
}
```

Çıktı:

```
7 ile 9 un toplami : 16

7 ile 9 un carpimi : 63
```

Yukarıdaki programda ise, bir öncekinden farklı olarak **Fonksiyon1** ve **Fonksiyon2** fonksiyonları statik erişimli olmayacak şekilde tanımlanmıştır. Bu durumda temsilci kullanıldığında yapılacak tek değişiklik, **Fonksiyon1** ve **Fonksiyon2** fonksiyonlarını bir **Sinif1** nesnesi aracılığı ile çağırmak olacaktır.

Ek Bilgiler

Karakter Dizisi Fonksiyonları - Örnekler

Substring

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Main()
        {
            string sozcuk, s1, s2, s3, s4, s5;
            sozcuk = "televizyon";

            // 0. karakterden basla 4 karakter ilerisine kadar al
            s1 = sozcuk.Substring(0, 4);

            // 4. karakterden basla 6 karakter ilerisine kadar al
            s2 = sozcuk.Substring(4, 6);

            // 3. karakterden basla 2 karakter ilerisine kadar al
            s3 = sozcuk.Substring(3, 2);

            // 3. karakterden basla dizi sonuna kadar al
            s4 = sozcuk.Substring(3);

            // 5. karakterden basla 7 kar. ileri git: hatali
            // s5 = sozcuk.Substring(5, 7);

            Console.WriteLine("s1 : {0}", s1);
            Console.WriteLine("s2 : {0}", s2);
            Console.WriteLine("s3 : {0}", s3);
            Console.WriteLine("s4 : {0}", s4);

            Console.ReadLine();
        }
    }
}
```

Çıktı:

```
s1 : tele
s2 : vizyon
s3 : ev
s4 : evizyon
```

Split

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Main()
        {
            string gunler = "pzt,sal,car,per,cum,cts,paz";
            // Split fonksiyonu, String sinifinda yer alan ve statik
            // olmayan bir fonksiyondur. Bu sinifin bir nesnesi olan
            // (string temel turune sahip) bir nesneye mesaj gondermek
            // yolu ile cagirilabilir. alabilecegi arguman sayisi ve
            // cesidi degismekle birlikte, arguman olarak aldigi
            // karakteri ayirici olarak kullanir ve ilgili nesneyi
            // parcalayarak bir 'string' dizisine yerlestirir, bu
            // dizinin baslangic adresini dondurur.

            string[] str2 = gunler.Split(',');
            // str2 dizisinin icerigi su sekilde olacaktir:
            // {"pzt", "sal", "car", "per", "cum", "cts", "paz"}

            Console.WriteLine("Haftanin 1. gunu : {0}", str2[0]);
            Console.WriteLine("Haftanin 4. gunu : {0}", str2[3]);
            Console.ReadLine();
        }
    }
}
```

Çıktı:

```
Haftanin 1. gunu : pzt
Haftanin 4. gunu : per
```

Replace, Remove, ToUpper, ToLower

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Main()
        {
            string sozcuk, s1, s2, s3, s4, s5;
            sozcuk = "abcdeabcde";

            // sozcuk icindeki 'c' karakterlerinin tumunu
            // 'z' olarak degistir
            s1 = sozcuk.Replace('c', 'z');

            // sozcuk icindeki "de" alt dizilerinin tumunu
            // "yz" olarak degistir
            s2 = sozcuk.Replace("de", "yz");

            // sozcuk dizisindeki 2. karakter ve sonrasini at
            s3 = sozcuk.Remove(2);

            // sozcuk dizisindeki tum kucuk harf karakterlerini
            // buyukleriyle yer degistir
            s4 = sozcuk.ToUpper();

            // s4 dizisindeki ilk karakterden baslayarak
            // 2 karakter ilerisine kadar al ve kucult
            s5 = s4.Substring(0, 2).ToLower();

            Console.WriteLine("s1 : {0}", s1);
            Console.WriteLine("s2 : {0}", s2);
            Console.WriteLine("s3 : {0}", s3);
            Console.WriteLine("s4 : {0}", s4);
            Console.WriteLine("s5 : {0}", s5);

            Console.ReadLine();
        }
    }
}
```

Çıktı:

```
s1 : abzdeabzde
s2 : abcyzabcyz
s3 : ab
s4 : ABCDEABCDE
s5 : ab
```

Insert, IndexOf

```
using System;

namespace Uzay1
{
    class Sinif1
    {
        static void Main()
        {
            string sozcuk, s1, s2, s3, s4;
            sozcuk = "bilgisayar";

            // sozcuk dizisindeki 2. karakterin sonrasina
            // "XYZ" karakter dizisini yerlestir
            // sonuc : biXYZlgisayar
            s1 = sozcuk.Insert(2, "XYZ");

            // sozcuk icindeki "i" alt dizisinin yer aldigi
            // ilk indeksten baslayarak araya "KLM" ekle
            // "yz" olarak degistir
            // sonuc : bKLMilgisayar
            s2 = sozcuk.Insert(sozcuk.IndexOf("i"), "KLM");

            // "sozcuk dizisinin uzunlugu : " seklindeki karakter
            // dizisinin sonuna, sozcuk degiskeninin tamsayi
            // uzunlugunun karakter dizisine donusturulmus halini ekle.
            s3 = "sozcuk dizisinin uzunlugu : " + sozcuk.Length.ToString();

            Console.WriteLine("s1 : {0}", s1);
            Console.WriteLine("s2 : {0}", s2);
            Console.WriteLine("s3 : {0}", s3);

            Console.ReadLine();
        }
    }
}
```

Çıktı:

```
s1 : biXYZlgisayar
s2 : bKLMilgisayar
s3 : sozcuk dizisinin uzunlugu : 10
```