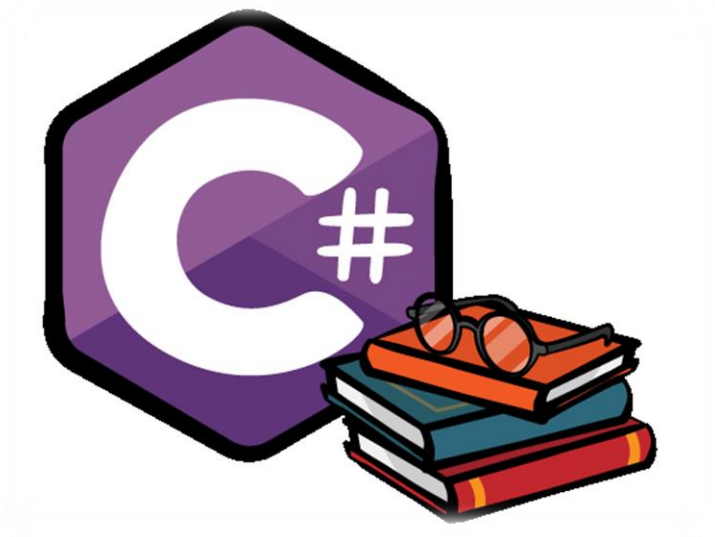


CONSOLE EKRANI-8

CONSOLE EKRANI

- ✓ *Metotlar*

KAZANIMLAR

- ✓ *Metot çeşitlerini açıklar ve öğrenir.*
- ✓ *Kendi metodunu oluşturabilir.*

METOTLAR

Programların hazırlanması esnasında, aynı işlemi gerçekleştiren program parçalarına programın birçok yerinde ihtiyaç duyulabilir. Eğer metotlar kullanılmazsa; programda aynı kodu defalarca yazmamız gerekebilir ve program kodlarının okunması zorlaşır. Aynı zamanda kaynak kodun gereksiz uzamasına sebep olur. Bunun için programın birçok yerinde ihtiyaç duyulan ve aynı işlemleri yapan program parçaları metotlar olarak hazırlanırlar.

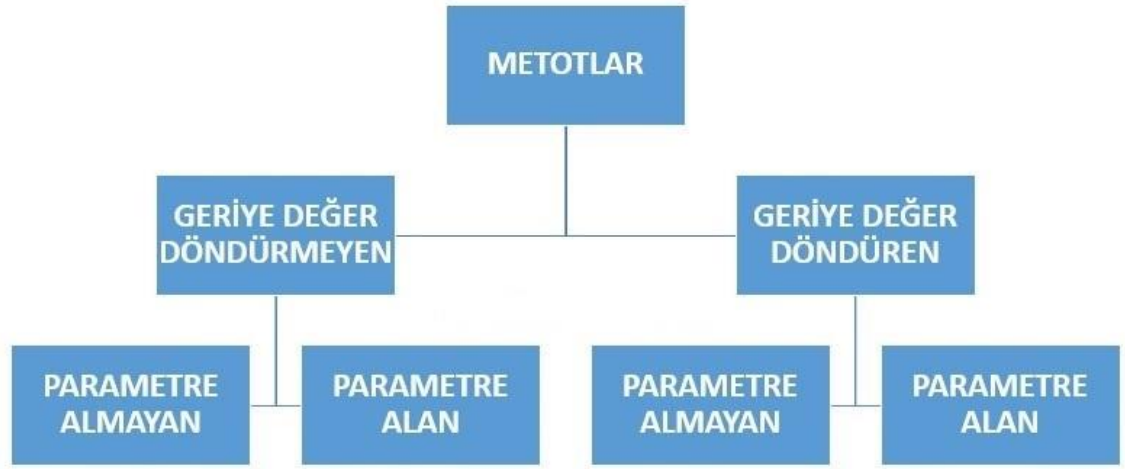
Metot Kavramı

Programların herhangi bir yerinde kullanılmak için belirli bir işi yerine getirmek amacıyla tasarlanmış alt programlara metot denilir. Metotlar belirli bir işi yapması için geliştirilirler. Programın yapısal olmasını sağlamak ve birbiriyle ilgili komutları veya programın bir bölümünü istenen isim altında toplamaktır. Bu şekilde programın okunması kolaylaşmakta ve yapısal bir görünüm kazanmaktadır.

Metot Tanımlama

*Her metodun bir ismi vardır ve program içerisinde metot çağrılırken bu isim kullanılır. Bir metodun iş yapabilmesi için kendi çağıran metottan aldığı bilgilere parametre, kendisini çağıran fonksiyona döndürdüğü değere de metot **geri dönüş değeri** (return value) denir. Metotlar genellikle şu şekilde tanımlanırlar;*

```
erişim dönüş-tipi isim(parametre-listesi)
{
// metodun gövdesi;
}
```



***Erişim:** Bu metoda, programın diğer bölümlerinin nasıl erişebileceğini belirleyen bir erişim niteleyicisidir. Bunun kullanımı isteğe bağlıdır. Eğer herhangi bir erişim belirteci kullanılmazsa varsayılan olarak sınıfa özel **(private)** olarak belirlenir. Private olarak kullanıldığında yalnızca metodun yazıldığı sınıf içerisinden çağrılabilmesini öngörür. Eğer programın içerisinde bulunan diğer kodlar içerisinden de bu metot çağrılabilsin isteniyorsa, erişim belirteci **public** olarak belirtilmelidir. Nesne yönelimli programlama dillerinde metotlar, tanımlandıkları sınıf adı ile birlikte çağrılırken eğer metot, programın ana metodu (Main()) içerisinden çağrılacaksa **static** olarak tanımlanır ve sınıf adını yazmaya gerek kalmadan çağrılır.*

Dönüş-tipi: *Bu metodun çalıştırıldıktan sonra programda çağrıldığı noktaya döndürdüğü verinin tipini belirlediğimiz kısımdır. Eğer metot bir değer döndürmeyecekse dönüş-tipi* ***void*** *olarak belirtilmelidir.*

İsim: *Metodunun isminin belirtildiği kısımdır. Metodumuza isim verirken yapacağı iş ile alakalı bir isim vermek hem metodun ne işe yaradığıyla ilgili bize bilgi verecektir, hem de bizden başka aynı programı kodlayacak kimselere yol gösterecektir. Metoda isim verirken aynı değişken isimleri tanımlarken kullandığımız kuralları yine göz önünde bulundurmalıyız. Geri dönüş tiplerinin veya parametre-listesinin farklı olması durumunda aynı isme sahip birden fazla metot olabilir.*

Parametre-Listesi: *Virgül (,) ile ayrılmış tip ve tanımlayıcı çiftlerden oluşan bir listedir. Parametreler, metot çağrıldığında, metodun kullanması için gönderilen bilgilerdir. Eğer metot hiç parametre kullanmayacaksa parametre listesi de boş olur.*

Örneklerle metot tanımlamalarını inceleyelim;

Örnek: *Geri dönüş değeri ve parametre-listesi boş olan, ekrana "Merhaba Dünya" yazdıran metodu tanımlayıp program içerisinde kullanımına bir örnek veriniz.*

```
    class Program
    {
        static void MerhabaDunyaYazdir()
        {
            Console.WriteLine("Merhaba Dünya");
        }
        static void Main(string[] args)
        {
            MerhabaDunyaYazdir();
            Console.ReadKey();
        }
    }
}
```

```
Merhaba Dünya
```

Örnek:*Klavyeden girilen bir tam sayının karesini bulan metodu ve bu metodun program içerisinde kullanımını gösteren programın kodunu yazınız.*

```
class Program
    {
        static int KareAl(int sayi)
        {
            int karesi = sayi * sayi;
            return karesi;
        }
        static void Main(string[] args)
        {
            Console.Write("Bir sayı giriniz: ");
            int s1, sonuc;
            s1 = Convert.ToInt32(Console.ReadLine());
            sonuc = KareAl(s1);
            Console.WriteLine("{0} sayısının karesi: {1}", s1, sonuc);
            Console.ReadKey();
    }
    }
```

Örnek: Klavyeden girilen bir mesajı ekrana 10 defa yazdıran metodun kodunu yazınız.

```
class Program
{
    static void MesajYaz(string msj)
    {
        for (int i = 1; i <= 10; i++)
            Console.WriteLine(msj);
    }
    static void Main(string[] args)
    {
        Console.Write("Mesajınızı giriniz: ");
        string mesaj;
        mesaj = Console.ReadLine();
        MesajYaz(mesaj);
        Console.ReadKey();
    }
}
```

Yukarıdaki kodları incelediğimiz zaman; klavyeden girilen yazı mesaj isimli değişken içerisine aktarılmakta ve daha sonra MesajYaz() isimli metota gönderilmektedir. MesajYaz() metodu ise kendisine parametre olarak verilen string türdeki mesajı ekrana 10 defa yazmaktadır.

Metotlarda Parametre Kullanımı

Parametrenin tanımını ve kullanımını daha önce metotların tanımlanması sırasında parametre listelerini oluştururken gördük. Parametre-listeleri, tek bir türde verileri içeren bir liste olabileceği gibi, farklı türlerde de veriler içerebilen listelerdir. Parametreler veri türünde olabileceği gibi nesneler de parametre olarak bir metoda gönderilebilirler. Her bir parametre aralarına virgül kullanılarak birbirinden ayrılırlar. Aynı veri türüne sahip parametrelerin her biri için değişken isimlerinden önce ayrı ayrı veri türleri de yazılmak zorundadır.

Çeşitli veri türlerini parametre olarak metotlarımıza nasıl gönderdiğimizi örneklerle inceleyelim

```
static bool AsalMi(int s)
        {
            bool durum = false;
            for (int i = 2; i < s / 2 + 1; i++)
            {
                if (s % 2 == 0)
                    durum = false;
                else
                    durum = true;
            }
            return durum;
        }
        static void Main(string[] args)
        {
            int sayi = 0;
            bool drm;
            Console.Write("Bir sayı giriniz: ");
            sayi = Convert.ToInt32(Console.ReadLine());
            drm = AsalMi(sayi);
            if (drm == true)
                Console.WriteLine("{0} sayısı asaldır.", sayi);
             else
                Console.WriteLine("{0} sayısı Asal değildir.", sayi);
            Console.ReadKey();
        }
```

Örnek: Parametre olarak gönderilen kullanıcı adı ve şifreyi kontrol eden, önceden belirlenmiş olan bir kullanıcı adı ve şifreyle karşılaştıran metodun kodlarını yazınız.

```
class Program
    {
        static void KullaniciKontrol(string kAdi, string psw)
        {
            if ((kAdi == "Admin") || (kAdi == "ADMİN") || (kAdi == "admin"))

              {
                if (psw == "123rty")
                    Console.WriteLine("Tebrikler Kullanıcı ve Şifreniz Doğru");
                else
                    Console.WriteLine("Şifrenizi Hatalı Girdiniz");
            }
           else
            {
                Console.WriteLine("Kullanıcı adınız hatalı.");
            }
        }
        static void Main(string[] args)
        {
            string kullaniciAdi, sifre;
            Console.Write("Lütfen kullanıcı adınızı giriniz: ");
            kullaniciAdi = Console.ReadLine();
            Console.Write("Lütfen şifrenizi giriniz: ");
            sifre = Console.ReadLine();
            KullaniciKontrol(kullaniciAdi, sifre);
            Console.ReadKey();
        }

    }
```

Örnek : *Klavyeden girilen değerler arasında rastgele sayı üreten ve bu değerleri 10 elemanlı bir dizi içerisine atayan* ***SayiUret( )*** *isimli bir metot yazınız. Dizinin elemanlarını ekrana yazdıran* ***DiziYazdir( )*** *isimli bir metot daha yazarak elemanları ekrana yazdırınız. Daha sonra bu dizi içerisindeki en büyük sayı değerini bulan* ***EnBuyuk( )*** *isimli, en küçük değeri bulan* ***EnKucuk( )*** *isimli iki metot daha yazınız. EnBuyuk ve EnKucuk metotlarından dönen sayıları ekrana yazdıran programın kodlarını yazınız.*

Bu kısımda rastgele sayılar üretilip parametre olarak gönderilen dizi isimli diziye değerler aktarılıyor ve dizi ana programa geri döndürülüyor;

```
        static int[] SayiUret(int bas, int bit, int[] dizi)
        {
            int tutulan = 0;
            Random rnd = new Random();

        for (int i = 0; i < 10; i++)
            {
                tutulan = rnd.Next(bas, bit);
                dizi[i] = tutulan;
            }
            return dizi;
        }
    }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki değerler ekrana yazdırılıyor;

```
static void DiziYazdir(int[] dizi1)
        {
            Console.WriteLine("----------------");
            Console.WriteLine("Tutulan sayılar:");
            foreach (int i in dizi1)
                Console.WriteLine(i);
            Console.WriteLine("----------------");
        }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki en büyük değer bulunup ana programa geri döndürülüyor;

```
  static int EnBuyuk(int[] dizi2)
        {
            int ebs = 0;//en küçük değer
            foreach (int s in dizi2)
            {
                if (s > ebs) //eğer sayı ebs'den büyükse
                    ebs = s; //yeni ebs, sayının değeri olur
            }
            return ebs;
        }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki en küçük değer bulunup ana programa geri döndürülüyor;

Ana programımız;

```
        static void Main(string[] args)
        {
            int[] sayilar = new int[10];
            int baslangic, bitis;
            Console.Write("Başlangıç değerini giriniz: ");
            baslangic = Convert.ToInt32(Console.ReadLine());
        Tekrar: Console.Write("Bitiş değerini giriniz: ");
            bitis = Convert.ToInt32(Console.ReadLine());
            if (bitis <= baslangic)
            {
                Console.WriteLine("Bitiş değeri başlangıçtan ({0}) küçük ya da eşit olamaz
tekrar deneyiniz.",baslangic);
            goto Tekrar;
            }
            //Rastgele sayılar üretilip diziye aktarılıyor
            sayilar = SayiUret(baslangic, bitis, sayilar);
            //Dizi ekrana yazdırılıyor
            DiziYazdir(sayilar);
            //En büyük değer bulunuyor
            int mak = EnBuyuk(sayilar);
            //En küçük değer bulunuyor
            int min = EnKucuk(sayilar);
            //Sonuçlar ekrana yazdırılıyor
            Console.WriteLine("En büyük sayı: " + mak);
            Console.WriteLine("En küçük sayı: " + min);
        }

    }
```

Metotlarla ilgili Temel Özellikleri

Metotlarla ilgili bilinmesi gereken bazı önemli özellikler şunlardır;

- ✓ *Metotlara isim verilirken aynı değişkenlere isim verirken uyduğumuz kurallara uymamız gerekir. Main( ) ismi programımızın çalışmasını başlatan ana metodun ismi olduğu için bu ismi metot ismi olarak veremeyiz.*
- ✓ *Aynı isime sahip farklı geri dönüş tiplerine veya farklı parametre-listesine sahip metotlar oluşturabiliriz.*

```
class Program
   {
        static int Topla(int sayi1, int sayi2, int sayi3)
        {
            int toplam;
            toplam = sayi1 + sayi2 + sayi3;
            return toplam;
        }
        static int Topla(int sayi1, int sayi2)
        {

            int toplam;
            toplam = sayi1 + sayi2;
            return toplam;
        }
        static void Topla(int sayi1)
        {
            Console.Write("Parametresiz metodun sonucu= {0}", sayi1);
        }
        static void Main(string[] args)
        {
            int sonuc, s1, s2, s3;
            s1 = 43;
            s2 = 16;
            s3 = 66;
            sonuc = Topla(s1, s2, s3);
            Console.WriteLine("3 parametreli metodun sonucu= {0}",
            sonuc);
            sonuc = Topla(s1, s2);
            Console.WriteLine("2 parametreli metodun sonucu= {0}",
            sonuc);
            Topla(s1);
            Console.ReadKey();
        }

    }
```

Bu yöntem pek de tavsiye edilen bir yöntem değildir. Bu şekilde aynı isme sahip farklı metotlar oluştururken çok dikkatli olmalıyız.

- ✓ *Metotlar çağrılırken, başlangıçta belirlenen parametre sayısından ne az ne de çok sayıda parametre girmeliyiz. Eğer metodumuz 2 parametre ile işlem yapıyorsa, biz bu metoda 1 veya 3 adet parametre gönderemeyiz. Aksi takdirde hata mesajı alırız.*

```
static int Topla(int sayi1, int sayi2)
        {
            int toplam;
            toplam = sayi1 + sayi2;
            return toplam;
        }
        static void Main(string[] args)
        {
            int sonuc, s1, s2, s3;
            s1 = 43;
            s2 = 16;
            s3 = 66;
            sonuc = Topla(s1, s2, s3);
            sonuc = Topla(s1);
            Console.WriteLine("Sonuç= {0}", sonuc);
               }
```

Yukarıdaki kodları incelediğimizde; geri dönüş değeri bulunmayan Topla isimli metot, - koyu renkli satırdan da görüleceği üzere - sonuc isimli değişkene atama işlemi yapılmaya çalışılırsa hata mesajı alırız

- ✓ *Metotların geri dönüş değerleri herhangi bir veri türünde olabilir. Metot içerisindeki bir değer return anahtar sözcüğüyle metodun çağrıldığı yere geri döndürülür. Burada metodun geri dönüş tipine uyumlu bir değişken içerisine atanmalıdır. Aksi takdirde tür uyumsuzluğundan dolayı hata mesajı alırız.*

```
class Program
    {
        static float Topla(int sayi1, int sayi2)
        {
            float toplam;
            toplam = sayi1 + sayi2;
            return toplam;
        }
        static void Main(string[] args)
        {
            int sonuc, s1, s2, s3;
            s1 = 43;
            s2 = 16;
            s3 = 66;
            sonuc = Topla(s1, s2);
            Console.WriteLine("Sonuç= {0}", sonuc);
               }
```

Yukarıdaki kodları incelediğimizde; int türünde tanımlanmış olan sonuc değişkeni içerisine float türünde tanımlanmış bir metodun geri dönüşü atanmaya çalışılırsa hata mesajı alırız.