

# C# Metotlar

**Adem AKKUŞ**

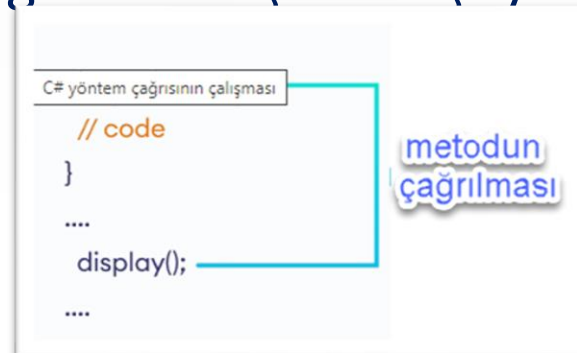
Bilgisayar Mühendisi  
Uzman Bilişim Teknolojileri Öğretmeni  
Eğitmen

# Metod Nedir ?

- Programlarda iş yapan en temel parçalar fonksiyonlardır.
- **Fonksiyonlar belirli bir işi yapan kodlardır.**
- İş yapan bu fonksiyonlar çeşitli şekillerde paketlenerek başkaları tarafından da kullanılabilir.
- Fonksiyonların ve bazı özelliklerin paketlenerek yeniden kullanılabilen hale gelmesi sınıf (**class**) dediğimiz yapıları oluşturur.
- Bir metod, yalnızca çağrıldığında çalışan ve bir dizi ifade içeren kod bloktur. Yazılım dünyasında bir metod, sınıf içinde yapılacak işlerin veya operasyonların tanımlanmasını sağlar.
- Metotlara parametreler aracılığıyla ana programdan veriler gönderilebilir ve metotlar çalışmasını bitirdikten sonra ana programa değer döndürülebilir
- Bu bölümde sınıfların en önemli yapısı olan metotları (fonksiyonları ) göreceğiz.

# Metod Nedir ?

- **Metotlar**, bir program içerisinde aynı işi gerçekleştiren satırları belirli düzende sadece bir kez oluşturarak gerektiğinde tekrar tekrar kullanabilmemizi sağlayan alt programlardır.
- **Metotlar**, programın herhangi bir yerinde kullanılmak için ,belli bir işi, görevi yerine getirmek için tasarlanmış alt programlardır. Metotlar tek başlarına çağrılabilen yapılar değildir. Ancak metot, kendisini çağıran ana programa faydalı işler yapar. Yani çağrılmayan metot kod kalabalığından başka bir şey değildir.



# Metod Nedir ?

- Bazı programlama dillerinde (Python, Javascript,PHP gibi) **fonksiyon** olarak ifade edilen yapılar bazı programlama dillerinde (Basic, Pascal,SQL gibi) **prosedür** olarak anılmaktadır.
- Günümüz OOP yaklaşımında bu yapıya **metot** (method) denilmektedir. C# 'da fonksiyon yerine metod daha çok kullanılır.
- Şuana kadar yaptığımız uygulamalarda sadece bir metod kullandık. O da her projede sadece bir tane olması gereken Main() metodudur.

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
}
```

*Not: Main() metodu, programın başlangıç (start) noktasıdır.*



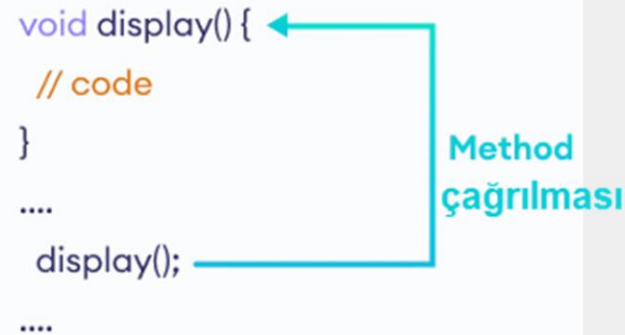
# Metod Nedir ?

- Kapsamlı ve karmaşık uygulamalar yaparken ,bütün işlemler Main() metodu içerisinde yapılmaz.
- Bu şekilde yazılım geliştirme hem kötü bir teknik hem de geliştirme açısından zorluklar barındırır.
- Programlar modüllere bölünür. Her modül sadece kendi kendisine atanan görevi yerini getirir.
- Metoda mümkün olduğunca az görev verilmelidir. **Bir metod tek bir işi az ve öz biçimde yapmalıdır.**

Örnek, bir hem dizi elemanlarını sıralama hem de ekrana yazdırma görevi olmamalıdır. Ayrı ayrı metotlar tarafından yapılmalıdır.

# Metod Nedir ?

- Metod tekrar çağrılana kadar bir iş yapmadan bekler.
- Metodlar, çağrılan metod tarafından bir takım bilgiler alır, alınan bilgiler çeşitli işlemlerden geçirildikten sonra metodu çağıran metoda yeni değeri gönderir.



```
void display() {  
    // code  
}  
....  
display();  
....
```

Method çağırılması

- Metod tanımlamada parantez içerisine yazılan değişkenlere **parametre**,
- Çağrılırken parantez içerisine yazılan değişkenlere/değerlere **argüman**, denir

# Metod Bildirimi (Tanımlanması)

**Erişim Belirleyici:** Metoda nerelerden erişilebileceğini tanımlar.

**Dönüş Tipi:** Metotlar değer döndürebilir. Dönüş tipi, metodun döndüreceği değer tipini tanımlar (int, string vb.). Değer döndürmek için **return** anahtar kelimesi kullanılır. Metot değer döndürmeyecekse dönüş tipi olarak **void** anahtar kelimesi kullanılır.

**Metot Adı:** Metodun adını tanımlamada kullanılır. İsim+fiil şeklinde olabilir  
Örnek; *OrtalamaHesapla()*, *EkranaYazdir()*, *BilgileriGetir()* gibi

**Parametre Listesi:** Parantez içinde verilen parametreler, bir metoda değer göndermek veya metottan değer almak için kullanılır. Parametrelerin türü, sayısı ve sırası parametre listesi olarak adlandırılır.

Bunlardan **metodun adı** ve **parametre listesi** *metodun imzası*, küme parantezi { } arasına yazılan kodlar da **metodun gövdesi** olarak adlandırılır. Metot imzaları, bir sınıf içinde her metotta farklı olmak zorundadır.

**Metot adı + Parametre listesi = Metodun imzası**

# Metod Bildirimi (Tanımlanması)

[erişim belirleyici] dönüş-tipi metotadi ( [parametre listesi ] )

{

//metot içerisinde gerçekleştirilecek işlemler ;

}

erişim belirleyici :public, private, protected,internal, protected internal

dönüş tipi :void, bool, int, string, char, herhangi bir referans tipi

Metoda verilecek isim :EkranaYazdir, Listele, IkiSayiTopla

Not : [erişim belirleyici] zorunlu değil. Varsayılan olarak private 'dir.

[parametre listesi ] ,parametre yoksa parantez içerisinde hiçbir şey yazılmaz.



# Metod Bildirimi (Tanımlanması)

```
class Program
{
    static void MerhabaYaz()
    {
        Console.WriteLine("Merhaba");
    }
    static void Main(string[] args)
    {
        MerhabaYaz(); // metodun çağrılarak çalıştırılması
        Console.ReadKey();
    }
}
```

**Not:** Sınıf içerisindeki metodların birisi `static` ise diğer metodların da erişim belirleyicisi `static` olmalıdır. Çünkü sınıf artık `static` olmuştur.

# Metod Türleri

Yapılarına göre metotlar 4 'e ayrılır:

1. Değer Döndürmeyen Metotlar
2. Değer Döndüren Metotlar
3. Parametre Almayan Metotlar
4. Parametre Alan Metotlar

# Metod Türleri

## 1. Değer Döndürmeyen Metotlar

Değer döndürmeyen metotlar tanımlanırken void anahtar kelimesi kullanılır.

```
1 public void DegerDondurmeyenToplama(int sayi1, int sayi2)
2 {
3     int sonuc = sayi1 + sayi2;
4     Console.WriteLine(sonuc);
5 }
6
7 static void Main(string[] args)
8 {
9     int a=5, b=2;
10    DegerDondurmeneyenToplama(a,b);
11 }
```

# Metod Türleri

## 2. Değer Döndüren Metotlar

Değer döndürme işlemi **return** parametresi ile yapılabilir. **return** ifadesinden sonra yazılan değişken veya değer, aşağıda gösterilen *DegerDondurenToplama()* metodunun çağrıldığı yer olan *Console.WriteLine()* içine değer gönderilmiş olur.

```
public int DegerDondurenToplama(int sayi1, int sayi2)
{
    int sonuc = sayi1 + sayi2;
    return sonuc;
}

static void Main(string[] args)
{
    int a=5, b=2;
    Console.WriteLine(DegerDondurenToplama(a,b));
}
```

Aslında değer döndürmeyen, yani *void* ile tanımlı metotlarda da **return** komutuna rastlanabilir. Ancak bu tür durumlarda **return** komutu aşağıdaki gibi kullanılır:

```
1 return;
```

Bu kullanımın amacı geriye değer döndürmek değil metottan çıkılmasını sağlamaktır.



# Geriye Değer Döndüren Metodlar

```
class Program
{
    static string MerhabaGonder()
    {
        return "Merhaba";
    }
    static void Main(string[] args)
    {
        string mesaj=MerhabaGonder(); // metodun çağrılarak çalıştırılması
        Console.WriteLine(mesaj);
        Console.ReadKey();
    }
}
```

# Parametre Alan Metodlar

```
class Program
{
    static string MesajGonder(string mesaj)
    {
        return mesaj;
    }
    static void Main(string[] args)
    {
        string mesaj=MesajGonder("Merhaba"); // metodun çağrılarak çalıştırılması
        Console.WriteLine(mesaj);
        Console.ReadKey();
    }
}
```

# Metodların Aşırı Yüklenmesi

```
class Program
{
    static int Topla(int s1,int s2)
    {
        return s1+s2;
    }
    static int Topla(int s1,int s2,int s3)
    {
        return s1+s2+s3;
    }

    static void Main(string[] args)
    {
        string mesaj=MesajGonder("Merhaba"); // metodun çağrılarak çalıştırılması
        Console.WriteLine(mesaj);
        Console.ReadKey();
    }
}
```

# Recursive (öz yinelemeli) Metodlar

- Kendisini çağıran metodlardır.
- Yavaş çalışırlar.
- Sondan başa doğru çözülürler.



# Recursive (öz yinelemeli) Metodlar

- Kendisini çağıran metodlardır.
- Yavaş çalışırlar.
- Sondan başa doğru çözülürler.

## Örneğin

- $2^3$  sayısını hesaplamak istersek

$$2^3 = 2^2 * 2$$

$$2^2 = 2^1 * 2$$

# Üs Alma İşlemi (Recursive)

```
static int UsAl(int us,int taban)
{
    if (us == 0)
        return 1;
    return taban * UsAl(us-1,taban);
}
```

```
static void Main(string[] args)
{
    Console.Write("Tabanı Giriniz : ");
    int taban = Int32.Parse(Console.ReadLine());
    Console.Write("Üssü Giriniz : ");
    int us = Int32.Parse(Console.ReadLine());
    Console.WriteLine("{0} üssü {1}={2}", taban, us, UsAl(us,taban) );
    Console.ReadKey();
}
```

## Faktoriyel Hesaplama (Recursive)

///kodu ekle



## Metotlarda params Kullanımı

///`kodu ekle`