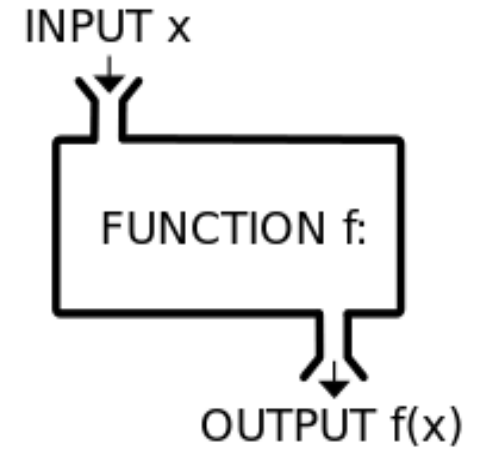


METOTLAR *(FONKSİYONLAR)*

Arif GÜNEL

METOTLAR (FONKSİYONLAR)



- *Herhangi bir sınıf içerisinde bir işi yapmak için tanımlanmış bölümlere metot denir.*
- *Bu arada metot ve fonksiyon aynı işi yapar sadece c# da metot kelimesini kullanıyoruz.*
- *Nesne yönelimli programla dili olan c# sınıf mantığı ile çalıştığından metotlar sınıfların altında çalışır.*

- *Şimdiye kadar incelediğimizde Main metodundan bahsetmiştik.*
- *Bu metot bir programda olmazsa olmaz metodumuzda program ilk çalıştığında bakılan kısımdı.*
- *Profesyonel anlamda yazılan programlar tek bir metodun altına toplanamaz. Bir çok metot tanımlanarak kod yazılır. Bu büyük bir problemi parçalara ayırarak çözmek gibidir.*

- ***Kullanım faydaları;***
- *Bu sayede uzun programlarda parça parça kodları oluşturmak kolaydır, sorun çıktığında düzeltmek kolaydır,*
- *Bir bölgedeki sorun diğer kısımları etkilemez.*
- *Programın daha anlaşılır olmasını sağlar.*

- *Metotlar parametre ile çalışıp sonucunda bir değer geri döndürebilir bu durumda geri dönüş tipi olur. Bazı metotlar çağrıldıkları yere geri değer döndürmeye bilir.*
- *C# da fonksiyon yerine metot terimi kullanılır.*
- *Metotlar kendi başlarına çalıştırılmazlar bir sınıfın(class) altında çalışırlar.*
- *Ana programdan çağrılmadıkları müddetçe çalışmazlar. Ana programın akışında ihtiyaç duyduğumuz zaman ilgili sınıfın içerisinden ihtiyaç duyulan metot çağrılır ve kullanılır ya da metoda bir değer verilir bu değer ile çalışıp bize değer olarak vermesinin ya da vermemesini sağlayabiliriz.*

- *Metot içerisinde başka metot tanımlanmaz.*
- *Main metodu programın başlangıç noktasını belirleyen metotdur.*
- *Kendi tanımladığımız metotlar ve sistem metotları bulunmaktadır.*
- *C# nesne yönelimli bir programlama dili dediğimizde aslında sınıfların altında metotlar bulunmaktadır.*
- *Çok kullandığımız ekrana yazdırma komutumuzda bir metotdur.*
- *Console sınıfının WriteLine metodudur.*

`Console.WriteLine(Topla(23,22));`

Metot'un kullanım şekli;

```
static void Main(string[] args)
```

- *static*: class ları incelerken göreceğiz
- *void*: geri değer döndürmediğini ifade ediyor.(mesela loglama metodu)(tam sayı, metin vb geri döndürebilir)
- *string[]*: metodun parametreleri(argümanları) yani çalışması için gerekli değerleri veriyoruz.Veriler tiplerini tanımlıyoruz.

Ornek: Metotlar

- Temel olarak iki sayının toplamını geri döndüren metot tanımlayalım

```
using System;
namespace Ornek16._0Metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            int sonuc;
            sonuc = Topla(25, 5); // metot
            // kullanılarak yapılacak. burada ekranda bir şey
            // görülmez.
            Console.WriteLine(sonuc);
            Console.WriteLine(Topla(23, 22));
            Console.Read();
        }
    }
}
```

```
static int Topla(int sayi1, int sayi2) //
int türünden değer verilerek
çalışacak. sonuçta int olarak gelecek.
{
    return sayi1 + sayi2;
}
}
```



- *NOT: Metotların imzası diye tanımlanan kavram;*
- *Metodun ismi ve parametrelerin sayısı ve veri tiplerinin aynı olduğu başka metot olamaz. (static
int Topla(int sayi1, int sayi2))*
- *Bu sayede metotlar bir birinden ayrılıyor. Aynı isimde olup parametre sayısı ve türlerinin farklı olduğu metotlar tanımlanabilir. Buna metotların aşırı yüklenmesi denir.*

Ornek: Metotlar(GERİ DEĞER DÖNDÜRMİYEN METOTLAR)

- *Gönderilen string ifadeyi ekrana yazdıran program,*
- *Metotları isimlendirirken değişken isimlendirme kuralları geçerli, sayı ile başlamaz özel karakter özel karakter kullanılmaz.*
- *Parametreler doğru girilmesi ve kullanılırken boş bırakılırsa hata verir.*
- *Geri değer döndürmeyen metotlar **return** anahtar kelimesi kullanılmaz.*
- *Metotlara açıklama yazmak için üç tane `///` işaretinden sonra açıklama yazıyoruz kullanırken kısa bilgilendirme yapar.*
- *Geri dönüş tipleri aynı değerden olmalı ama double, int döndürebilir ama int double geri döndürmez*

```

class Program
{
    static void Main(string[] args)
    {
        EkranaYaz("Metotlar değer
döndürmeyebilir. Sadece belirtilen bir işi
yapmak için oluşturulabilir.");
        EkranaYaz("ÖRNEK METİN
YAZMA");//parametrelili kullanım.
        EkranaYaz
        EkranaYaz();//parametresiz kullnımı
aşırı yükleme
        Console.ReadLine();
    }
    /// <summary>
    /// Bu metot ekrana srtring değer
yazdırır
    /// </summary>
    /// <param name="Metin">string tpinde bu
parametre ekrana yazılması istenen metni
taşıır.</param>

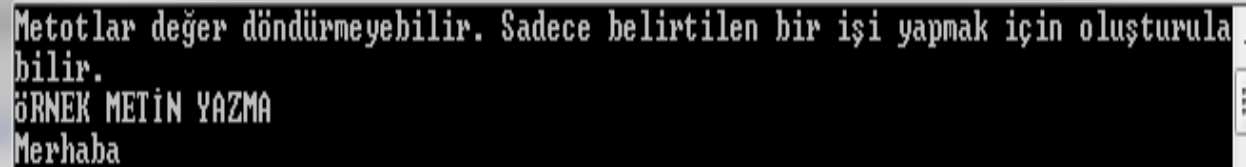
```

static void EkranaYaz(string Metin)//geri değer döndürmediğinden geri dönüş tipine void diyoruz.

```

{
    Console.WriteLine(Metin);
}
static void EkranaYaz();//parametresiz
metot tanımı
{
    Console.WriteLine("Merhaba");
}
}

```



```

Metotlar değer döndürmeyebilir. Sadece belirtilen bir işi yapmak için oluşturulabilir.
ÖRNEK METİN YAZMA
Merhaba

```

Ornek:Metotlar3(DEĞİŞKENLERİN YAŞAM ALANLARI)

- Değişkenlerin yaşam alanı ile ilgili örnek bir değişken sadece bir metot içerisinde tanımlandıysa tanımlandığı metot içerisinde geçerlidir.
- Metot tanımlayıp içerisinde değişkeler olsun.

```
static void Main(string[] args)
```

```
{
```

```
    decimal hesapsonucu = Ortalama(20,54);
```

```
    Console.WriteLine(hesapsonucu);
```

```
    Console.Read();
```

```
}
```

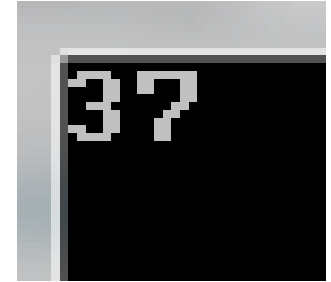
```
    static decimal Ortalama(int sayi1, int sayi2)//kendisine gönderilen iki sayının ortalamasını  
    alsın.geri dönüş tipi decimal virgüllü olabilir.
```

```
{
```

```
    decimal sonuc = (sayi1 + sayi2) / 2;// sonuc isminde değişken tanımlanıyoruz.main metodunun  
    altında aynı isimde değişken tanımlayabiliriz ama bunlar bir birinden bağımsızdır.
```

```
    return sonuc;
```

```
}
```



Örnek: KDV hesaplayan Metot

```
static void Main(string[] args)
{
    float sonuc;
    sonuc = kdv(23);
    Console.WriteLine(sonuc);
    Console.ReadLine();
}

static float kdv(float sayi1)
{
    float sayi2;
    sayi2 = sayi1 / 100;
    sayi2 = sayi2 * 18;
    return sayi1 + sayi2;
}
```

Örnek: Metotlar arası geçiş, bir metotdan diğerini çalıştırdık

```
static void Main(string[] args)
{
    int enBuyuk;
    enBuyuk = EnBuyukBul2(103,54,87);
    Console.WriteLine(enBuyuk);
    Console.ReadLine();
}
static int EnBuyukBul1(int sayi1, int sayi2)
{
    if (sayi1>sayi2)
    {
        return sayi1;
    }
    else
    {
        return sayi2;
    }
}
```

```
static int EnBuyukBul2(int sayi1, int
sayi2, int sayi3)
{
    int sonuc;

    sonuc = EnBuyukBul1(sayi1,
EnBuyukBul1(sayi2,sayi3));
    return sonuc;
}
```



103

DegiskenSayidaPArametreAlanMetotlar

- *Bundan önce yapılan metot örnekleri sabit değişkenli metot tanımlamalarıydı, yani 2 değişkenli 3 değişkenli gibi. 2 parametre tanımladıysak iki parametre ile kullanmak zorundaydık.*
- *Şimdi ise değişken sayıda parametre alan metotları inceleyeceğiz, sınırsız sayıda parametre alabilirler. 3 parametrelili metot tanımlayıp hiç parametre almadan da kullanılabilir.*
- *Toplama metodu yapıyoruz kendisine verilen sayıları toplayan metot,*
- *Eskiden olsa iki sayıyı toplayan metot olsa tanımlamasını;*
 - *`static int TopLa(int sayi1, int sayi2)`*
- *şeklinde yazıyorduk, ve sadece iki sayı olunca işlem yapıyorduk.*

- Parametre sayısı belli olmayan metotları tanımlarken **dizileri** kullanıyoruz. Değişken sayıda parametre olacak ise **Params** anahtar kelimesiyle birlikte kullanıyoruz

`static void Topla(params int[] sayılar)`

- Bu şekilde tanımlama ile istenilen sayıda sayıyı toplama işlemine tabi tutabiliyoruz.
- İlk olarak hiç değer döndürmeden sadece hesaplayıp ekrana yazdırmasını istiyorsak alt kısımdaki kodları yazıyoruz.


```
static void Main(string[] args)
```

```
{
```

```
    Topla(22, 32, 54, 6);
```

```
    Topla(2, 3);
```

```
    Topla(324);
```

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

```
}
```

```
static void Topla(params int[] sayilar)
```

```
{
```

```
    int Toplam=0;
```

```
    for (int i = 0; i < sayilar.Length; i++):
```

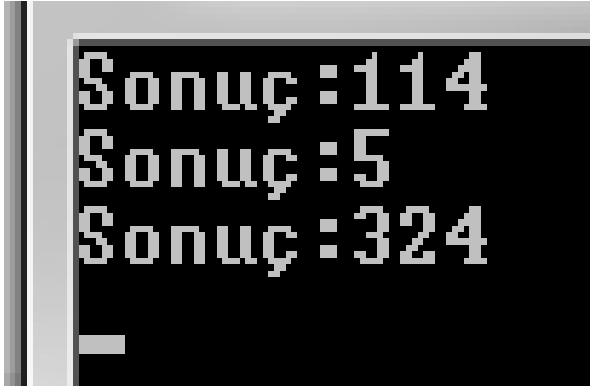
```
    {
```

```
        Toplam += sayilar[i];
```

```
    }
```

```
    Console.WriteLine("Sonuç:{0}", Toplam);
```

```
}
```



```
Sonuç:114  
Sonuç:5  
Sonuç:324  
-
```

- Bu örnekte tek bir metotla aynı türden olmak kaydı ile her defasında sayının önemli olmadığı değişkenleri toplama işlemini yaptık.

Örneğe devam edelim (DegiskenSayidaPArametreAlanMetotlar)

- *Params anahtar kelimesini kullanmadan da aynı değişken sayıda parametere ile çalışan metot tanımlayabilirdik. Bu şekilde kullanımda işlem yapılacak değerleri verirken dizi şeklinde vermeliyiz.*
- *Yeni bir dizi tanımlayalım aynı isimde olsun aşırı yükleme yapalım fakat isim ve parametre aynı olduğundan hata verir parametre değeri olarak Double seçelim.*
- *Kullanım olarak dizi tanımlamak sıkıntı olabilir ilk yaptığımız daha dinamikdir. İkincisinde dizi tanımlamak zorundayız.*

```

static void Main(string[] args)
{
    //static int Topla(int sayi1, int sayi2) sadece iki sayıyı toplayan parametre olsa bu
    şekilde tanımlayacaktık.
    Topla(22, 32, 54,6);
    Topla(2,3);
    Topla(324);

```

Ekle
nen

```

double[] dizi = {32,4,5,95,78 };
Topla(dizi);

```

```

Console.ReadLine();

```

```

static void Topla(params int[] sayilar)

```

```

{
    int Toplam=0;
    for (int i = 0; i < sayilar.Length; i++) //length sayilar dizisinin uzunluğunu simgeler
    {
        Toplam += sayilar[i];
    }

    Console.WriteLine("Sonuç:{0}",Toplam);
}

```

```

static void Topla(double[] sayilar)//bu kısımda params anahtar kelime kullanmadan sadece d
izi olarak yapıyoruz

```

E
k
l
e
n
e
n

```

{
    double Toplam = 0;
    for (int i = 0; i < sayilar.Length; i++)
    {
        Toplam += sayilar[i];
    }
    Console.WriteLine("Sonuç:{0}", Toplam);
}

```

```

}

```

```

Sonuç:114
Sonuç:5
Sonuç:324
Sonuç:214
_

```

file:///D:/ÖĞRETİM GÖREVLİLİĞİ/2015-

```

Sonuç:114
Sonuç:5
Sonuç:324
Sonuç:214

```

Seçimlik(Opsiyonel) Parametrelili Metot Kullanımı

- *C# 5.0 ile gelen bir özelliktir. Çok kolay ve anlaşılır. Kısa bir örnek yapalım. Biraz önceki Topla isminde yine bir metot oluşturalım. Bu konu bir önceki değişken sayıda parametre alan metotlara benzer gibi dursa da farklı bir kullanımdır. Örneğin 3 parametrelili bir metot tanımlayalım;*

```
static void Topla(int sayi1=0,int sayi2=0, string Mesaj="")
```

- Burada dikkat edilecek olursa her bir parametreye başlangıç değeri atıyoruz. Bu sayede bu parametreleri boş geçebiliriz. Metotun içerisine ekrana kullanıcı bir mesaj göndermiş ise bu mesaj ve sayıların toplamını mesaj göndermemiş ise sadece iki sayının toplamını yazacak;

```
Console.WriteLine(Mesaj + "TopLam:{0}",sayi1+sayi2);
```

Şimdi bu topla metodumuzu farklı parametre değerleri kullanarak çağırabiliriz

```
Topla(21);
```

//ilk parametre 21 ikinci parametre(sayi2) 0 olarak değerlendirilip işlem yapılır.

```
Topla(23,43);
```

```
Topla(13,32, "Sayıları toplandı. ");
```

```
Topla(sayi2:23,sayi1:98,Mesaj: "Parametere adıyla çağırma");
```

Programın tamamı;

```
class Program
{
    static void Main(string[] args)
    {
        Topla(21); //ilk parametre 21 ikinci parametre(sayi2) 0 olarak değerlendirilip işlem yapılır.
        Topla(23,43);
        Topla(13,32,"Sayıları toplandı. ");
        Topla(sayi2:23,sayi1:98,Mesaj:"Parametere adıyla çağırma");//parametre isimlerini de direk vererek sıra önemi olmadan değer verebiliriz. İsim vermez isek sıralı değer alır.

        Console.ReadLine();
    }

    static void Topla(int sayi1=0,int sayi2=0, string Mesaj="")
    {
        Console.WriteLine(Mesaj + "Toplam:{0}",sayi1+sayi2);
    }
}
```

```
Toplam:21
Toplam:66
Sayıları toplandı. Toplam:45
Parametere adıyla çağırmaToplam:121
```

Özyineli(*Recursive*) Metotlar

- *Recursive metotlar programlama dillerinde bir çok problemin çözümünde kullanılır. Çok hızlı sonuç verir. Metodun tanımlama bloğu içerisinde metodun kendisine referans verilip metodun kendisini çağırıyoruz.*
- *En bilinen örnek olan faktöriyel hesaplama metodu örneğini yapalım.*
- *Faktöriyel sayılan sayının kendisine kadar olan tüm sayıların çarpımı demektir.($n!=n.(n-1).....3.2.1$)*

```
static int Faktoriyel(int sayi)
```

```
{
```

```
    int sonuc;
```

```
    if (sayi==0)
```

```
    {
```

```
        return 1;
```

```
    }
```

```
    else
```

```
    {
```

```
        sonuc = sayi * Faktoriyel(sayi-1);
```

```
    }
```

```
    return sonuc;
```

5! Olduğunu düşünürsek

///*Faktoriyel(5)*

///*5*Faktoriyel(4)*

///*5*4*Faktoriyel(3)*

///*5*4*3*Faktoriyel(2)*

///*5*4*3*2*Faktoriyel(1)*

///*5*4*3*2*1*1*



Programın tamamı;

```
static void Main(string[] args)
{
    int hesap;
    hesap = Faktoriyel(5);
    Console.WriteLine(hesap);
    Console.ReadLine();
}
static int Faktoriyel(int sayi)
{
    int sonuc;
    if (sayi==0)
    {
        return 1;
    }
    else
    {
        sonuc = sayi * Faktoriyel(sayi-1);
    }
    return sonuc;
}
```