Görsel Programlama BMT-214

C# Metotlar

Prof. Dr. O. Ayhan ERDEM

Hüseyin Kaan KIZILDAĞ

Tarih: 19.05.2020

Numara: 181816029

Şube: II. Şube

C# Metotlar

İçindekiler

Metot Nedir?

C# da olduğu gibi diğer nesne yönelimli programlama (OOP) [1]dillerinde C[2] dilinden aşina olduğumuz fonksiyon ismi yerine metot kullanılır. İsimleri farklı olsa da aralarındaki fark yok sayılabilecek kadar azdır.

C# dilinde değer döndürmeyen metotlar için Prosedür ya da Fonksiyon adı kullanılır. Bu küçük farkı açıkladıktan sonra metin bütünlüğü açısından yazının bundan sonrasında metot ifadesini kullanılacaktır.

C# ve diğer birçok dilde olduğu gibi ürettiğimiz proje içerinde mutlaka bulunması gereken bir metot vardır. Bu metot Main metottur. Metotlar ana fonksiyondaki iş yükünü azaltmak, okuması daha kolay kodlar yazmamızı sağlamak ve yazılan kodun kapladığı alanı küçülten yardımcı yapılardır.

Metot Tanımlama

Metot sınıflar içerinde tanımlanırken yazılması gerek bir takım anahtar kelimeler vardır. Bunlar erişim belirleyiciler (Access modifiers), değer döndürüyorsa döndürdüğü değerin tipi ve şart olmamakla birlikte nesne üzerinden çağrılmayacak özellik için static anahtar kelimesidir.

Değer Döndüren ve Döndürmeyen Metotlar

Metotlar kendi içinde işlemler yaptıktan sonra metot dışına değer döndürebildikleri gibi değer döndürmeden de sonlanabilirler. Değer döndürmek için return anahtar kelimesi kullanılır.

Değer Döndürmeyen Metotlar

Her metot değer döndürmek zorunda değildir. Bu tür metotlar için return kullanmak keyfidir. Değer döndürmeyen metotlarda return deyimi kullanılırsa metot o komut satırında sonlanacaktır. Aksi halde metodun tüm satırları bitinceye kadar işlemler devam edecektir. Bu metotlar tanımlanırken void anahtar kelimesiyle tanımlanır.

```
public void menu()
{
    Console.WriteLine("1_Para Yatırma");
    Console.WriteLine("2_Para Çekme");
    Console.WriteLine("3_SIM kart bloke kaldırma");
    Console.WriteLine("4_Hesap Bilgileri");
}
```

```
public void isimleSelam(string isim)
{
    if (isim.ToLower() == "ayhan")
    {
        Console.WriteLine("Hoşgeldiniz Ayhan Hocam");
    }
    else Console.WriteLine("Hoşgeldiniz " + isim);
}
```

[1] OOP Her işlevin nesneler olarak soyutlandığı bir programlama yaklaşımıdır. Yüksek seviyeli diller olarak da sınıflandırılabilir.

[2] C Dili bir yapısal programlama dili olup c# ve c++ gibi dillerin atası sayılmaktadır.

Değer Döndüren Metotlar

Metotlar sınıfları da değer olarak döndürebilir fakat biz genel veri tipleri için birkaç örnek göstereceğiz. C# dilinde metotlar tanımlanırken döndüreceği değerin tipi de belirtilmek zorundadır.

```
public double cemberAlani(float r)
{
    return Math.PI * Math.Pow(r,2);
}
```

```
public Class1 SinifDondur()
{
    Class1 YeniSinif = new Class1();
    return YeniSinif;
}
```

Erişim Belirleyiciler (Access modifiers)

Erişim belirleyiciler nesne yönelimli programlama dillerinin bir özelliği olan kapsüllemenin (encapsulation) temel taşlarındandır. Nesne içerisindeki bazı işlemlere erişim olması istenmeyebilir. Bu özelliklere sınıf dışından müdahalesinin engellenmesinde erişim belirleyiciler kullanılır.

ACCESS MODIFIER	AYNI ASSEMBLY İÇİNDEN			FARKLI ASSEMBLY'LER İÇİNDEN	
	TANIMLANDIĞI SINIFTAN	DİĞER SINIFLARDAN	TÜRETİLMİŞ SINIFLARDAN	DİĞER SINIFLARDAN	TÜRETİLMİŞ SINIFLARDAN
Private	Erişilebilir	**Erişilemez**	**Erişilemez**	**Erişilemez**	**Erişilemez**
Public	Erişilebilir	Erişilebilir	Erişilebilir	Erişilebilir	Erişilebilir
Protected	Erişilebilir	**Erişilemez**	Erişilebilir	**Erişilemez**	Erişilebilir
Internal	Erişilebilir	Erişilebilir	Erişilebilir	**Erişilemez**	**Erişilemez**
Protected Internal	Erişilebilir	Erişilebilir	Erişilebilir	**Erişilemez**	Erişilebilir

Public

Erişimi en açık erişim belirleyicidir. Kendi sınıfı içerisinden alt sınıfından paketinden ve global olarak her yerden ulaşılabilir.

Protected

Protected tanımlanan varlık tanımlandığı sınıfın içerisinde erişilebilirdir. Bu yönüyle private'a benzese de aralarındaki fark protected tanımlanan varlıklar o sınıftan türetilmiş sınıflar içerisinden de erişilebilir hale gelmektedir.

Internal

Internal olarak tanımlanan bir öğeye bulunduğu assembly [3] içerisinde her yerden erişilebilir. Ayrıca bir sınıfın erişim belirleyicisi tanımlanmazsa sınıfın erişim belirleyicisi varsayılan olarak Internal tanımlıdır.

Private

En katı erişim belirleyicidir. Belirttiği birimi tanımlandığı kod bloğu içerisinde görünür kılar. Bir sınıf içerisindeki üyelerin erişim belirleyicileri belirtilmezse varsayılan olarak private tanımlanır.

[3] Derlenmiş exe ve dll dosyasına assembly denir

Protected Internal

Inernal ve protected kavramlarının veya işlemine tabii tutulmuş halidir. protected internal tanımlanan bir öğeye tıpkı protected tanımlanmış gibi tanımlandığı sınıfın içerisinden ve o sınıftan türetilmiş sınıflardan erişilebilir. Ek olarak, aynı assembly içerisinde olmasalar dahi tanımlandığı sınıftan türetilmiş diğer sınıflar içerisinden de erişilebilir.

Static Anahtar Kelimesi

Şimdiye kadar pek çok kez kullandığımız static kelimesi fonksiyon tanımlamalarında da kullanılabilir. Nesne yönelimli programlama dillerinde işlemlerimizin çoğunu sınıflar içinde yaparız. Sınıf içerisinde oluşturulan bir değişkene ya da bir metoda erişebilmek için o sınıftan bir nesne oluşturulması gerekir. Fakat bir metodu veya değişkeni static olarak nitelendirirsek o sınıftan nesne oluşturmadan da o özelliği kullanabiliriz. Örneğin Math sınıfı üzerinden pi sayısına erişmek için yeni bir nesne oluşturmamıza gerek yoktur. Aşağıda Class1 isimli bir sınıftan c1 adında bir nesne oluşturulmuştur. static olarak tanımlanmayan selamla metoduna erişim ancak c1 nesnesi üzerinden olur. merhabaDe adlı metot ise static anahtar kelimesiyle nitelendirilmiştir. Metotların çağrılmasındaki fark aşağıda verilmiştir.

```
public void selamla()
{
    Console.WriteLine("Selamlar");
}

public static void merhabaDe()
{
    Console.WriteLine("Merhaba");
}
```

```
Class1 c1 = new Class1();
c1.selamla();

Class1.merhabaDe();
```

Parametreler ve Yerel Değişkenler

Programlama dillerinde metotların aldığı değerlere parametre denir. Metodun [4]gövdesi içinde tanımlanan değerler ise yerel (local) değişkenler olarak adlandırılır. Metot tanımı yaparken zorunlu olmamakla birlikte metodun parantezleri içinde parametre gönderilebilir. Burada dikkat edilmesi gereken husus gönderilen her parametrenin tipini de belirtme şartıdır.

```
public double delta (float a, float b, float c) //parametreler
{
    double delta;   //yerel değişken
    delta = Math.Sqrt(b) - 4 * a * c;
    return delta;
}
```

[4] Metot gövdesi küme parantezi '{ }' içindeki alana denir.

Metotlara Parametre Olarak Dizi Gönderilmesi

Bazı durumlarda metotlara kaç değer göndereceğimiz belli olmaz işte bu durumda params anahtar kelimesiyle metotlara dizi gönderebiliriz. foreach hazır metodu gönderilen dizi üzerinde gezmek için pratik bir yoldur. Burada dikkat edilmesi gereken nokta params anahtar kelimesiyle gönderilecek değerlerin fonksiyonun son parametresi olmasıdır. Aksi taktirde gönderilen her parametre yeni değerin params anahtar kelimesi dışında bir değer olduğu anlaşılamaz.

```
public int gelismisToplam(params int [] arr)
{
    int toplam = 0;
    foreach(int i in arr)
    {
        toplam += i;
    }
    return toplam;

}
```

Zaten bu durumda birçok geliştime ortamı uyarılar vermektedir. Fakat düşük seviyede bir derleyici kullanıyorsak veya not defteri üzerinden yazılım üretiyorsak bu uyarıları göremeyebiliriz.

```
static int toplam(params int [] dizi, int gecersizParametre)
{
    int toplam = 0;
    foreach(int i in dizi)
    {
        toplam += i;
    }
    return toplam;
}
```

```
static int toplam(int gecerliParametre, params int [] dizi)
{
    int toplam = 0;
    foreach(int i in dizi)
    {
        toplam += i;
    }
    return toplam;
}
```

Ref Anahtar Kelimesi

ref anahtar kelimesi metotlara gönderilen parametreler değer[5] tipli (value type) de olsa referans[6] (referece type) tipine zorlayarak gönderilir. Basit şekilde değer tanımı yaptığımızda değerler stack hafızada yer alır ve sadece değere ait bilgi tutar, adres bilgisi tutmazlar. Referans tipli değişkenler ise heap bellekteki bellek adresini, stack bellekte de veriyi tutarlar. Yani bir metoda değer tipiyle parametre yollarken sadece o değerin yerini tutan bir değişken gönderilir.

Heap bellekle stack bellek arasındaki temel fark stack bellekte oluşturulan veriler işi bittikten sonra kaybolur. Fakat heap bellekte işler bu şekilde yürümez. Heap bellekleri temizlemek için garbage collector [7]benzeri yapılar kullanılır. Alt seviyeli dillerde garbage collector yapıları olmadığı için bellek kullanımına dikkat edilmelidir. Aksi durumlarda taşma (overflow) hataları alınır. Referans tipi ve değer tipini en saf haliyle görebileceğimiz yer değiştirme metoduyla örneklendirelim.

```
static void yerDegistir(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
static void swap(ref int a, ref int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```
swap(ref x, ref y);
```

Not: swap metoduna parametre yollarken parametrelerin referans olduğunu belirtmeliyiz.

yerDegistir metodunda oluşturulan a ve b değerleri o fonksiyon içerisinde oluşturulur, metot bitince değerleri ve adresleri bellekten silinir. Yani yer değişimi ana metoda taşınamaz. Fakat swap metodunda parametreler referans tipli olduğu için bellek üzerindeki yansımaları metot içerisinde işleme tabi tutulur. Dolayısıyla metot tamamlandığında ana metot üzerinde bu metodun yansımasını görürüz.

```
ilk durumda x ve y değerleri
x: 10 y: 5

yerDegistir metodundan sonra x ve y değerleri
x: 10 y: 5

swap metodundan sonra x ve y değerleri
x: 5 y: 10
```

[5] Primitif tip (float int double bool) değişken tipleriyle tanımlanan değişkenler değer tipli (value type) olarak adlandırılır ve stack de tutulur.
[6] Tanımlanan bir değişkenin heap bellekteki adresi onun referansıdır.
[7]Garbage Collector java ve c# gibi üst seviyeli dillerde hiçbir yeri göstermeyen referansın silinmesini sağlayan yapılardır.

Out Anahtar Kelimesi

out ve ref anahtar kelimeleri hemen hemen aynı amaca hizmet ediyor. Ama aralarında temel bir fark bulunur. ref olarak metoda gönderilecek ifadenin başlangıç değeri metoda gönderilmeden önce atanmalıdır. Fakat out için bu zorunluluk yoktur. Metot içerisinde de başlangıç değeri atanabilir. Atama yapılmazsa out ile gönderilen değer metot içinde kullanılamaz ve derleyici hata verir.

```
static int outDeneme(out int x, int y)
{
    x = 0;
    x += y;
    return x;
}
```

Metotlara Default Parametre Göndermek

Default kelime anlamı olarak varsayılan demektir. Metotlara default parametre göndermek, C# ‘ a 5. nesilden sonra eklenen bir özelliktir. Metotlar çağrılırken gönderilmesi zorunlu olmayan parametrelerdir. Bu parametreler farklı bir değer gelmediği sürece başlangıçta tanımlanan değeri geçerli kılar. Burada dikkat edilmesi gereken default parametre veya parametrelerden sonra başka parametre yazarak tanımlama yapılmamasıdır. Bunun sebebi derleyicinin gönderilen hangi parametrenin default hangi parametrenin tanımlanması gerekli parametre olduğunu ayırt edemeyecek olmasıdır. Bu husus metotlara parametre yollarken de dikkat edilmesi gereken bir hususdur. Bu özelliği bir kdv hesaplama yaparken kullanalım. Katma Değer Vergisi aksi belirtilmediği sürece %18 olsun. Bu değer üzerinden kdv eklenmiş fiyatı bize döndürsün.

```
static float kdvHesapla(float fiyat, float kdv = 0.18F)
{
    return fiyat + fiyat * kdv;
}
```

```
static float kdvHesapla(float kdv = 0.18F, float gecersizTanimlama)
{
    return gecersizTanimlama + gecersizTanimlama * kdv;
}
```

Yukarıda yapılan tanımlama hatalıdır ve derleyici bize hata mesajı verir aşağıda ilgili hataya ait derleyici çıktısı verilmiştir.

```
CS1737  İsteğe bağlı parametreler tüm gerekli parametrelerden sonra yer almalıdır
CS7036  Program.kdvHesapla(float, float)' öğesinin gereken resmi 'gecersizTanimlama' parametresine karşılık gelen hiçbir bağımsız değişken yok
```

Metot İçinden Metot Çağırmak

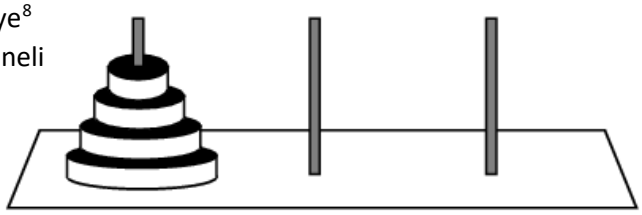
C# dilinde diğer programlama dillerinde de olduğu gibi metotlar birbirlerini çağırabilirler. Metotların kod tekrarından kurtardığı ve içinden çıkılamaz hale gelen problemleri çözmede güzel bir yöntem olduğu aşikardır. Bir kombinasyon hesabı yaparken birçok yerde faktöriyel hesabı yapmak gerekir. Bu gibi durumlarda metotlardan yardım almak kodun okunabilirliği açısından da bize avantaj sağlayacaktır. Ayrıca metotlar içerisinde de gerekli kontrolleri de yapmamı sağlar.

```
static long faktoriyel(int n)
{
    if(n == 0 || n == 1)
    {
        return 1;
    }
    else if(n < 0)
    {
        Console.WriteLine("negatif değerlerin faktöriyeli hesaplanamaz");
        return 0;
    }
    else
    {
        long tmp = 1;
        for(int i = n; i > 1; i--)
        {
            tmp *= i;
        }
        return tmp;
    }
}
```

```
static int kombinasyon(int n, int r)
{
    int komb = (int)(faktoriyel(n) / (faktoriyel(n - r) * faktoriyel(r)));
    return komb;
}
```

Özyineli Metotlar (Recursive Methods)

Yukarıdaki örnekte de görüldüğü gibi metotlar metot içerisinden de çağrılabilir. Özyineli metotlar da kendi kendilerini çağıran metotlardır. Bu metotlar gizli bir döngü oluşturur ve for, while, do-while döngüleri gibi bu döngü de sonsuz döngüye[8] girebilir. Birtakım yazılımcılar tüm işlemlerinde özyineli metotlarla çalışmayı kendilerine felsefe edinmiştir. Özyineli metotlar veri yapıları içinde dolaşmadan hanoi kulelerinin [9] çözümüne String sınıfının metotlarından, basit matematiksel işlemlere kadar birçok yerde karşımıza çıkar.



Daha önce tanımladığımız faktöriyel metodunu bu sefer de özyineli olarak tanımlayalım.

```
static long faktoriyel(int n)
{
    if (n == 1 || n == 0)
    {
        return 1;
    }
    return n * faktoriyel(n - 1);
}
```

Aşırı Yükleme (Overload)

Bir metot farklı parametrelerle olmak şartıyla aynı isimle birden fazla tanımlanabilir. Bu tür tanımlamalara aşırı yükleme denir. Aşırı yüklemeli metotları sınıf konusu içerisinde incelenmesine rağmen yapıcı(Constructor)[10] metotlar üzerinden örneklendirmek istiyorum. Bir sınıftan nesne oluşturulurken gerekli tanımlamaların yapıldığı ya da sadece bilgi amaçlı bir şeyler yazıldığı metotlardır. Sınıflardan nesne üretilmek isteniyorsa o sınıfın en az bir tane yapıcı metodu bulunmalıdır. Burada kullandığımız en az bir tane ifadesi bize ipucu veriyor. Bir sınıfın birden çok yapıcı metodu olabilir. Fakat nesne oluşurken bunlardan sadece biri çalışır.

[8] Sonlanması için gerekli denetimleri yapılmayan veya yanlış denetim sonucu true değer üreten döngülerdir.
[9] Hanoi kuleleri bir nevi puzzle oyunudur. Bazı kurallar çerçevesinde 3 kulenin birinde büyükten küçüğe olacak şekilde disk sıralamaya dayanır
[10] Yapıcı metot, bir sınıftan yeni bir nesne oluşturulurken çağrılan metottur. new anahtar kelimesiyle tetiklenir.

```
public Ucak(String kuyrukTscl, String marka, String model, int agirlik)
{
    this.kuyrukTscl = kuyrukTscl;
    this.marka = marka;
    this.model = model;
    this.agirlik = agirlik;

}
public Ucak(String kuyrukTscl)
{
    this.kuyrukTscl = kuyrukTscl;
}

public void bilgiler()
{
    Console.WriteLine("Kuyruk tescili: " + this.kuyrukTscl);
    Console.WriteLine("Marka: " + this.marka);
    Console.WriteLine("Model: " + this.model);
    Console.WriteLine("Ağırlık(kg): "+ this.agirlik.ToString());
    Console.WriteLine("******************************");
}
```

```
Ucak cess = new Ucak("TC-AHU", "Cessna", "172S", 757);
Ucak a400 = new Ucak("TC-OMD");
cess.bilgiler();
a400.bilgiler();
```

Örneğimizde sınıfımızın değişkenlerinden bazılarına nesne oluşturulurken atama yapmak zorunlu olsun. Atama yapmak zorunda olmadığımız değerleri de ayrı bir yapıcı metot içinde tanımlayalım ve hangi metodun hangi durumda çalıştığını inceleyelim.

_____Program Çıktısı_____

Kuyruk tescili: TC-AHU

Marka: Cessna

Model: 172S

Ağırlık(kg): 757

****

Kuyruk tescili: TC-OMD

Marka: Bilinmiyor

Model: Bilinmiyor

Ağırlık(kg): 0

****

Kaynakça

- Ders Notu – İnternet: http://cozvelioglu.com/c-erisim-belirleyiciler-ve-birkac-ufak-detay-access-modifiers/ Access Modifiers - Ceyhun ÇÖZVELİOĞLU
- Ders Notu – İnternet: https://www.w3schools.com/cs/cs_method_overloading.asp Overload
- Online Ders – İnternet: https://www.btkakademi.gov.tr/ Metotlara default parametre göndermek - Engin DEMİROĞ
- Kaynak – İnternet: http://www.canertosuner.com/post/C-Ref-Out-Kullan%C4%B1m%C4%B1 C# ref – out anahtar kelimesi - Caner TOSUNER
- Kaynak – İnternet: https://docs.microsoft.com/tr-tr/dotnet/csharp/language-reference/keywords/ref C# ref – out anahtar kelimesi – Microsoft
- Kaynak – İnternet: https://ilkaygenc.com.tr/metot-nedir-erisim-seviyeleri/ Yerel değişkenler – İlkay GENÇ
- Ders Notu : 4.hafta Fonksiyon Procedure Tanımlama - Prof. Dr. O. Ayhan ERDEM
- Kaynak – İnternet: https://www.kemalkefeli.com.tr/csharp-metotlar-ve-fonksiyonlar.html Metot ve Fonksiyon arasındaki farklar - Kemal KEFELİ
- Kaynak - İnternet: https://softwareengineering.stackexchange.com/questions/20909/method-vs-function-vs-procedure Metot ve Fonksiyon farkları - softwareengineering.stackexchange.com
- Görsel : https://www.pngindir.com/png-jssm7a/
- Görsel : https://mathworld.wolfram.com/TowerofHanoi.html
- Görsel : https://www.turkhackteam.org/c-j-vb-net-net-dilleri/1723531-c-acces-modifiers-erisim-belirleyiciler-ar-ge-kulubu.html