Düzgün bir metot yazma ya da metot oluşturma yazılımdaki neredeyse en önemli konudur. Metotlar serisi boyunca bu konuya özen gösteriniz.

METODLAR
1. BÖLÜM

Metot Oluşturma

Oğuzhan Karagüzel

İçindekiler

ŞEKİLLER DİZİNİ

GİRİŞ

Metotlar ya da (fonksiyonlar) programlama dünyasının olmazsa olmaz elemanlarıdır. Tekrar tekrar kod yazmaktan bizi kurtarır. Bir kod bloğunu daha anlaşılabilir kılar. Hatta bir kod bloğuna resmen ad vermiş oluruz ve bu işi yapar diye tanımlamış oluruz. Bundan dolayı metotlar konusu oldukça önemlidir.

Ancak metotlar konusu oldukça geniş kapsamlıdır. Bundan dolayı bu konu size 4 ana bölümde anlatılacaktır. Bu konuya pür dikkat yaklaşmanızı tavsiye ediyorum. Bu konuyu iyi öğrenmek yazılım dünyasına atılacak en önemli adımlardan birisidir.

1. BÖLÜM METOT NEDİR, NASIL TANIMLANIR (+ STATİC ANAHTAR SÖZCÜĞÜ)

METOT (METHOD) NEDİR?

- **Metot**, belirli bir görevi gerçekleştirmek üzere oluşturulmuş, adı olan bir kod bloğudur.
- Bir programın içerisinde, tekrar tekrar kullanmamız gereken işlemleri bir metot içerisine koyarız.
- Böylece hem kod tekrarını azaltır, hem de programın okunabilirliğini ve bakımı kolaylaştırırız.

Bu zamana kadar değişkenleri öğrendiniz. İçlerinde sayısal ya da metinsel değer saklayabildiğimiz yapılardı. Metotları ise içlerinde kod blokları saklayabildiğimiz yapılar olarak düşünebilirsiniz. İnt sayi adlı değişkeni çağırdığımızda bize bir sayıyı geri döndürür. Metodu çağırdığımızda ise bize kod bloğunu geri döndürür gibi düşünebilirsiniz. Kod bloğu çalışır. ardından program akışına devam eder.

Neden Metot Kullanırız?

1. **Kod Tekrarını Azaltmak**: Bir işlemi farklı yerlerde kullanacaksak, metodu tek bir yerde yazar, ihtiyaç oldukça çağırırız.
2. **Okunabilirliği Artırmak**: Büyük programlar, metotlar aracılığıyla küçük parçalara bölünür. Hem geliştirme hem de hata ayıklama (debug) süreçleri kolaylaşır.
3. **Organize Olmak**: Programı parçalara (metotlara) böldüğümüzde, hangi blok ne yapıyor daha net belli olur.

METOT NASIL TANIMLANIR?

C# dilinde bütün metotlar **sınıfların (class)** içerisinde tanımlanır. Henüz nesne yönelimli programlama (OOP) konusuna girmemiş olsak da metot tanımlamayı şu şekilde görebiliriz:

[erişim_belirteci] [static veya diğer özellikler] [dönüş_tipi] MetotAdi([parametre_listesi])

```
{
    // Metot gövdesi (Bu kodlar, metot çağrıldığında çalışır.)
}
```

Bu şablondaki bileşenler şunlardır:

1. **Erişim Belirteci (Access Modifier)**: Metodun hangi alanlardan erişilebilir olacağını söyler.
    - Örn. public, private, internal gibi. Başlangıç seviyesinde genellikle public veya private kullanılır. Henüz nesne tabanlı programlama bilmediğinizden sizler şu anlık public anahtar kelimesini kullanacaksınız.
2. **static (veya diğer özellikler)**:
    - Bu dokümanda static üzerinde duracağız. static, metodu sınıf adını kullanarak direkt çağırabileceğimiz anlamına gelir (nesne oluşturmadan).
    - OOP'ye girdiğimizde virtual, override vb. başka anahtar sözcükler de öğreneceğiz.

    Şu an başlangıç aşamasındasınız. Daha öncede sizlere anlattığım üzere program sınıfının (class) içerisindeki "Main" metodu adı üzerinde bir metottur. Ancak statik bir metottur. Statik metodlar içerisinde sadece statik metotlar çalışmaktadır. (ya da bir sınıf üzerinden üretilen örnek – instance ile ulaşılan metotlar. Henüz sizin konunuz değil.) Şu aşamada class yapısını kullanamayacağınızdan ve yazacağınız metotları main metodu içerisinde çalıştıracağınızdan static anahtar kelimesini kullanmak zorundasınız.
3. **Dönüş Tipi (Return Type)**: Metodun geri döndürdüğü değerin türünü belirtir.
    - Örn. int, string, bool, double vb.
    - Eğer metot herhangi bir değer döndürmüyorsa void yazılır.

    Bu konuyu ilerleyen başlıklarda daha detaylı işleyeceğiz. İlk bölümde yazacağımız metotlar bir değer geri döndürmeyecektir. Bundan dolayı void anahtar kelimesini kullanacağız.
4. **Metot Adı (Method Name)**: Metodun ismi.
    - C#'ta metot isimleri genellikle **PascalCase** stilinde yazılır (örnek: MerhabaDe, Hesapla, PrintMessage).

    Bu konu size ödev olarak verilecektir. Yazım stili (case styles, naming conventions). Yazılımda oldukça önemli konudur. Düzgün adlandırma ve adları bir kurala göre yazma, programınızın diğer yazılımcılar tarafından anlaşılabilmesi için çok önemlidir.
5. **Parametre Listesi**: Parantez içinde, metot çalışırken dışarıdan almak istediğimiz değerler tanımlanır.
    - Örneğin: (int x, int y), (string mesaj) gibi.
    - Parametre yoksa parantezleri boş bırakırız.

Bu bölümde parametre konusuna kısaca değineceğiz, ama detaylı kullanımlar (örneğin ref, out gibi) sonraki bölümlerde anlatılacak.

Basit Bir Metot Örneği (Parametresiz ve void Dönüşlü)

```
internal class Program
{
    static void Main()
    {
        SelamVer(); // Metot çağrısı
    }
    public static void SelamVer() // Metot tanımlama
    {
        Console.WriteLine("Merhaba! Bu bir metot çağrısıdır."); // Metot içeriği
    }
}
```

Şekil 1 Metod tanımlama ve çağırma örneği

public static void SelamVer()

{

Console.WriteLine("Merhaba! Bu bir metot çağrısıdır.");

}

Bu örnekte:

- Erişim belirteci: public
- static: Metodu nesne oluşturmadan çağırmaya olanak sağlar. Ya da doğrudan başka bir statik metodun içerisinde çağırmaya yarar
- Dönüş tipi: void (herhangi bir değer döndürmez).
- Metot adı: SelamVer
- Parametre listesi: Boş.

Metot, çağrıldığında **ekrana "**Merhaba! Bu bir metot çağrısıdır.**"** yazar ve biter.

Burada bize sağlanan avantaj metodu tekrar tekrar istediğiniz yerde çağırıyor oluşunuzdur. Onlarca satır kodu tek satır ile çağırabilirsiniz. Örnek olarak.

```
internal class Program
{
    static void Main()
    {
        SelamVer(); // 1. Metot çağrısı
        SelamVer(); // 2. Metot çağrısı
        SelamVer(); // 3. Metot çağrısı
    }
    public static void SelamVer() // Metot tanımlama
    {
        Console.WriteLine("Merhaba! Bu bir metot çağrısıdır."); // Metot içeriği
    }
}
```

Şekil 2 Tekrar eden metod çağrıları

Ne oluyor?

- Program çalışınca Main metodu devreye giriyor.
- Main metodu içerisinde SelamVer() metodu üç kez çağrılıyor. Her çağrıda ekrana "Selamlar, ben bir metot içerisinden yazıyorum!" yazılıyor.
- SelamVer() metodu, yukarıdaki tanımı gereği herhangi bir değer döndürmüyor (çünkü void).
- Metot içerisinde yazdığımız kod bloğu, her çağrıda sırayla işleniyor.

Böylece basit bir metot tanımlamış ve çağırmış olduk.

Özet

- Bir metot, **tekrar kullanılabilir bir kod bloğudur**. Programı küçük, anlaşılır parçalara ayırmak ve kod tekrarını önlemek için kullanılır.
- C#'ta metotlar bir **sınıf** içerisinde tanımlanmak zorundadır.
- Metot tanımlarken:
    1. Erişim belirteci (örn. public, private)
    2. static veya başka özellikler (bu dokümanda static'i ele aldık)
    3. Dönüş tipi (örn. void, int, string vb.)
    4. Metot adı (örnek: SelamVer)
    5. Parametre listesi (boş olabilir ya da çeşitli parametreler içerebilir)
- static ifadesi, metodu sınıfın örneğini oluşturmadan (nesne yaratmadan) doğrudan çağırabilmemizi sağlar.
- Main metodu da çoğu zaman static tanımlanır; çünkü program başlarken henüz herhangi bir nesne oluşturulmamıştır.

Bu bölümde temel olarak **metot** kavramını ve **static** kullanımını anlattık. Bir sonraki bölümde, metotlara **parametre** vermeyi ve bu metotlardan **geri dönüş değeri** almayı öğreneceğiz.

Ek İpucu / Notlar

- Metot adlarının, yaptığı işi ifade etmesi faydalıdır. Mesela SelamVer ismi, "merhaba mesajı veriyor" olduğunu belli eder.
- Bir metot static olmasa bile, aynı şekilde tanımlanır ve çalışır. Tek fark, onu çağırırken bir sınıf örneği oluşturmak gerekebilir. (Bunu OOP konusuna girince daha net anlayacağız.)
- Metotlar içinde **diğer metotları** da çağırabiliriz. Bu da programı daha modüler hale getirir.
- Metot konusu ilerledikçe, parametre tipleri (int, string, bool vb.), geri dönüş tipleri (int, bool, string vb.) ve farklı varyasyonlar (örnek: ref, out) gündeme gelecek.

ÖDEV

1. Kendin bir metot yaz: Ekrana 3 satır yazı yazsın.
2. Main metodundan bu metodu birden fazla çağırarak sonuçları gözlemle.

Bu alıştırma, metot tanımlama ve çağırma sürecini pekiştirmenize yardımcı olabilir.

Case syles'ı araştırın ve hakkında kısa bir doküman oluşturun. Kebab-case, snake_case, camelCase ve PascalCase i mutlaka öğrenin. Nerelerde kullanıldıklarını öğrenin. Bazıları size yabancı gelecektir. İlerledikçe öğreneceksiniz. Bundan dolayı bilmediklerinizi de not almayı unutmayın.