METOTLAR

1.1. Metot nedir?

Bir veya birden fazla komutun pratik bir şekilde tekrar tekrar çalıştırılabilecek şekilde bir araya getirilmiş haline ***metot*** denir. Başka bir deyişle metot, tekrar tekrar kullanılabilecek komutlar kümesidir.

Metot yerine ***fonksiyon*** kelimesi de kullanılabilir. Ancak terminoloji açısından C# dili için metot kelimesinin kullanılması daha uygundur.

Herhangi bir metot çağrıldığında, çağrılan metottaki komutlar işletilir ve daha sonra program çalışmaya kaldığı yerden devam eder.

1.2. Metot yazma

C# dilinde bir metot yazmak için aşağıdaki yazım şekli kullanılır.

```
Erişim Şekli Dönüş Tipi Metot Adı (Parametreler) {
     Metot çağrıldığında çalıştırılacak komutlar
}
```

Erişim şekli metodun nereden çağrılabileceğini belirler. Erişim şekli olarak ***private***, ***public*** veya ***protected*** ifadeleri kullanılabilir. Bu üç erişim şeklinin birbirinden farklarına bu dokümanda değinilmeyecek, konunun devamında verilecek örneklerde erişim şekli olarak "private" kullanılacaktır.

Dönüş tipi olarak int, double vs. gibi C# dilinde geçerli herhangi bir veri tipi yazılabilir. Eğer metot değer döndürmeyecekse bu kısma ***void*** yazılır. Bu konuya birazdan değinilecektir.

Metot adı olarak herhangi bir isim yazılabilir. Program içinden metot çağrılacağı zaman bu isim kullanılacaktır. Metot adı belirlenirken *değişken tanımlama kurallarına uyulmalıdır*.

Parametreler metot adından sonra parantez içinde yazılır ve isteğe bağlıdır. Yani gerektiğinde hiç parametre yazılmayabilir. Bu durumda parantezlerin içi boş bırakılır. İhtiyaca göre bir ya da birden fazla parametre yazılabilir. Parametre sayısı ile ilgili herhangi bir sınırlama yoktur.

Metot çağrıldığında çalıştırılacak olan komutlar parametrelerden sonra kıvrımlı parantezler { } içine yazılır.

Örnek olarak, MerhabaYaz isimli bir metot yazalım. Bu metot çağrıldığında "Merhaba" mesajı göstersin. Yazacağımız bu metodun erişim şeklinin "private" olduğunu varsayalım.

Metot herhangi bir değer döndürmeyeceği için dönüş tipi olarak “void” yazıyoruz. Metot adı olarak “MerhabaYaz” yazıyoruz. Metoda herhangi bir parametre göndermeyeceğimiz için metot adından sonra parantezlerin arasına hiçbir şey yazmıyoruz. Son olarak; kıvrımlı parantezler arasına, “Merhaba” mesajı göstermek için gerekli olan kodları yazıyoruz.

```
private void MerhabaYaz() {
     MessageBox.Show("Merhaba");
}
```

Yukarıda örnek olarak verilen MerhabaYaz komutu, aşağıdaki şekilde çağrılabilir.

```
MerhabaYaz();
```

Metotları kendi oluşturduğumuz yeni komutlara benzetebiliriz. Bir metot yazarken, o komutu çalıştırdığımızda neler yapılması gerektiğini belirleriz ve yazdığımız bu yeni komutu istediğimiz yerde, istediğimiz zaman, istediğimiz kadar kullanabiliriz.

Metotlar, tıpkı diğer komutlar gibi peş peşe veya programın farklı yerlerinde **istendiği kadar çağrılabilir**. Örneğin; bir butona basıldığında “MerhabaYaz” metodunu peş peşe iki kere çağırırsak, peş peşe iki kere Merhaba mesajı gösterilecektir.

```
private void merhabaButton_Click(object sender, EventArgs e) {
     MerhabaYaz();
     MerhabaYaz();
}
private static void MerhabaYaz() {
     MessageBox.Show("Merhaba");
}
```

1.3. Metotlara parametre gönderme

Metotlar çağrılırken bazı değerler göndererek metotların bu değerlere göre işlem yapması sağlanabilir. Yukarıdaki örnekteki “MerhabaYaz” metodu “Merhaba” mesajı gösteriyordu. Ancak biz istediğimiz herhangi bir mesajı gösteren bir metot yazmak isteyebiliriz. Bu durumda, mesajda göstermek istediğimiz metni parametre olarak gönderebileceğimiz bir metot yazmalıyız.

Daha önce de bahsedildiği gibi, metotların parametreleri metot adından sonra parantez içinde yazılır. Parametre yazılırken önce gönderilecek değerin tipi sonra da gönderilen değerin saklanacağı değişkenin adı yazılır.

Örnek olarak, "EkranaYaz" isimli bir metot yazalım. Bu metoda parametre olarak bir metin (karakter katarı – string) göndereceğimiz için parametre tipi olarak "string" yazmamız gerekir. Değişken adı olarak da "metin" yazalım. Bu metot, kendisine parametre olarak gönderilen metni ekrana yazsın.

```
private void EkranaYaz(string metin) {
     MessageBox.Show(metin);
}
```

Yukarıdaki metot aşağıdaki gibi çağrılabilir.

```
EkranaYaz("Selam.");
```

Yukarıdaki örnekte, metot çağrılırken parametre olarak "Selam." değeri gönderilmiştir. Ancak gönderilecek değer programcıya bağlıdır. Yani programcı string tipli olmak kaydıyla istediği herhangi bir değeri gönderebilir.

Yukarıdaki örnekten devam edersek; "EkranaYaz" metoduna "Selam." değeri gönderildiğinde metottaki "metin" değişkenine "Selam." değeri atanır. Daha sonra metodun içinde bu değişkenin değeri istenildiği gibi kullanılabilir. Yukarıdaki örnekte, gönderilen değer mesaj olarak gösterilmektedir.

Parametre gönderilerek çalıştırılan metotlara hep aynı değer gönderilmek zorunda değildir. Metoda her çağrılışında parametre olarak farklı bir değer gönderilebilir.

```
private void mesajButton_Click(object sender, EventArgs e) {
     EkranaYaz("Selam.");
     EkranaYaz("Benim adım Sinan.");
     EkranaYaz("Şu anda C# öğreniyorum.");
}
private void EkranaYaz(string metin) {
     MessageBox.Show(metin);
}
```

Yukarıdaki program derlenip çalıştırıldığında ekranda önce “Selam.” mesajı, sonra “Benim adım Sinan.” mesajı, sonra da “Şu anda C# öğreniyorum.” mesajı gösterilecektir.

Daha önce de belirtildiği gibi, metotlara parametre olarak birden fazla değer gönderilebilir. Bu durumda, her bir parametre birbirinden virgül ile ayrılır. Örnek olarak, iki adet int tipli parametresi olan ve bu sayıların toplamını mesaj olarak gösteren “ToplaYaz” isimli bir metot yazalım. Metot çağrılırken iki adet tamsayı değer gönderilecektir.

```
private void ToplaYaz(int sayi1, int sayi2) {
     var toplam = sayi1 + sayi2;
     MessageBox.Show(toplam);
}
```

Yukarıdaki metot aşağıdaki gibi çağrılabilir.

```
ToplaYaz(3, 5);
```

Yukarıdaki örnekte, metot çağrılırken parametre olarak 3 ve 5 değerleri gönderilmiştir. Ancak daha önce de belirtildiği gibi, gönderilecek değerler programcıya bağlıdır. Yani programcı her ikisi de int tipli olmak kaydıyla istediği değerleri gönderebilir.

Yukarıdaki örnekten devam edersek; “ToplaYaz” metoduna 3 ve 5 değerleri gönderildiğinde metottaki “sayi1” değişkenine 3 değeri, “sayi2” değişkenine de 5 değeri atanır. Daha sonra metodun içinde bu değişkenlerin değerleri toplanır ve yeni tanımlanan “toplam” isimli değişkene atanır. Daha sonra da toplam değişkeninin değeri mesaj olarak gösterilir.

Yukarıdaki örnekte, 3 ve 5 değerlerinin toplamı hesaplanıp 8 sonucu bulunacak, “toplam” değişkenine 8 değeri atanacak ve “toplam” değişkeninin değeri, yani 8 mesaj olarak gösterilecektir. Eğer parametre olarak 3 ve 5’ten farklı değerler gönderilirse, gönderilen değerler toplanacak, ekrana bu toplama işleminin sonucu yazdırılacaktır.

```
private void hesapButton_Click(object sender, EventArgs e) {
     ToplaYaz(3, 5);
}
private void ToplaYaz(int sayi1, int sayi2) {
     var toplam = sayi1 + sayi2;
     MessageBox.Show(toplam);
}
```

Yukarıdaki program derlenip çalıştırıldığında mesaj olarak “8” değeri gösterilecektir.

1.4. Metotlardan değer döndürme

Tıpkı metotlara parametre olarak değerler gönderilebildiği gibi, metotlar da çalıştırıldıktan sonra geriye değer gönderebilirler. Tek farkı, metotlara istenildiği kadar parametre gönderilebilirken, **metotlar geriye en fazla bir değer döndürebilirler**.

Yukarıdaki örneklerde yazdığımız metotlarda dönüş değeri olarak hep “void” ifadesini kullandık. Bu ifade, yazdığımız metodun geriye hiçbir değer döndürmeyeceğini bildiriyordu. Ancak void yerine bir değişken tipi yazarsak, metodun geriye o tipte bir değer döndüreceği anlamına gelir.

```
private int Topla(int sayi1, int sayi2) {
     var toplam = sayi1 + sayi2;
     return toplam;
}
```

Yukarıdaki örnekte de görüldüğü gibi, metot eğer bir değer geri döndürecekse, bu işlem ***return*** anahtar kelimesi ile yapılır. Değer geri döndürüleceği zaman önce “return” sonra da yanına döndürülecek değer (ya da bu değerin saklandığı değişkenin adı) yazılır.

Bu örnekte, parametre olarak gönderilen değerler sırayla “sayi1” ve “sayi2” değişkenine atanır. İlk satırda “sayi1” ve “sayi2” değişkenlerinin değerleri toplanır ve sonuç “toplam” değişkenine atanır. Daha sonra da “toplam” değişkeninin değeri return ifadesi ile geri döndürülür.

Bu metotta ekranda hiçbir şeyin gösterilmediğine dikkat edilmelidir. Metodun görevi sadece toplama işlemini yapmak ve sonucu geri döndürmektir.

Yukarıdaki metot şu şekilde çağrılabilir.

```
var sonuc = Topla(4, 7);
```

Burada, “Topla” isimli metot çağrılırken 4 ve 7 değerleri parametre olarak gönderilmiş, metodun geri döndürdüğü değer, yeni tanımlanan “sonuc” isimli değişkenine aktarılmıştır. Daha sonraki ifadelerde “sonuc” değişkeninin değer istenildiği gibi kullanılabilir.

Aşağıdaki örnek programda “sonuc” değişkeninin değeri mesaj olarak gösterilmektedir.

```
private void hesapButton_Click(object sender, EventArgs e) {
     var sonuc = Topla(4, 7);
     MessageBox.Show(sonuc);
}
private int Topla(int sayi1, int sayi2) {
     var toplam = sayi1 + sayi2;
     return toplam;
}
```

Yukarıdaki program derlenip çalıştırıldığında mesaj olarak “11” gösterilir.

Bir metodun dönüş tipi olarak “void” haricinde bir tip yazılmışsa, o metodun mutlaka bir değer döndürmesi gerekir. Değer döndürme işlemi “return” ifadesi ile yapıldığından dolayı, değer döndürmesi gereken metotlarda “return” ifadesinin kullanılması zorunludur.

Örnek olarak, aşağıdaki metot *hatalı* olacaktır.

```
private int Topla(int sayi1, int sayi2) {
     var toplam = sayi1 + sayi2;
     MessageBox.Show(toplam);
}
```

Benzer şekilde, eğer dönüş tipi olarak “void” yazılmışsa, o metotta “return” ifadesinin kullanılması hatalı olacaktır. Çünkü metodun herhangi bir değer döndürmemesi bekleniyorken, “return” ifadesi ile değer döndürmeye çalışmak anlamsız olacaktır.

Örnek olarak, aşağıdaki metot *hatalı* olacaktır.

```
private void Topla(int sayi1, int sayi2) {
     var toplam = sayi1 + sayi2;
     return toplam;
}
```

Bir metotta dönüş tipi olarak ne yazılmışsa, “return” ifadesinden sonra yazılacak değerin veya değişkenin tipi dönüş tipi ile aynı olmalıdır. Örneğin, dönüş tipi olarak “int” yazılmışsa “return” ifadesinden sonra mutlaka bir tamsayı ya da tamsayı tipli bir değişken yazılmalıdır. Ya da dönüş tipi olarak “string” yazılmışsa “return” ifadesinden sonra mutlaka bir karakter katarı

(çift tırnak içinde yazılmış bir veya birden fazla karakterden oluşan metin) ya da “string” tipli bir değişken yazılmalıdır.

Örnek olarak, aşağıdaki metot *hatalı* olacaktır.

```
private int Topla() {
     return "Selam.";
}
```

ÇALIŞMA SORULARI

1. Metot nedir?
2. Bir metot en fazla kaç kez çağrılabilir?
3. Metotlarda dönüş tipi olarak neler yazılabilir?
4. Hangi durumda metodun dönüş tipi olarak void yazılır?
5. Metot yazarken parametreler nereye yazılır?
6. Metotlara kaç adet parametre gönderilebilir?
7. return ifadesi ne işe yarar?
8. return ifadesinden sonra ne yazılır?
9. Hangi durumlarda return ifadesinin kullanılması zorunludur?
10. Bir metotta return ifadesi kullanılmamışsa bu ne anlama gelir?