Metotlar

Java programlama dilinde alt programlara metot denir. Bazı programlama dillerindeki fonksiyon veya prosedür sözcüklerinin karşılığıdır. Bir metot, bağımsız olarak çalıştırılabilen bir program parçasıdır. Bir programı metotlara bölerek yazmak bazı faydalar sağlar:

- Programın kaynak kodu küçülür. Böylece byte kod da küçülür.

- Kaynak kodun okunabilirliği ve anlaşılabilirliği artar.

- Belirli kod parçalarının programın farklı yerlerinde yinelenmesi, programda yapılacak olası bir değişikliğin maliyetini azaltır. Ortak kod parçaları metotlar olarak yazıldığında yalnızca metotlarda yapılan değişiklikler yeterlidir.

- Programda hata arama daha kolay yapılır.

- Yazılan metotlar başka projelerde de kullanılabilir.

Nesne yönelimli programlama tekniğinde genel olarak programlar sınıflara bölünür ve sınıflar içerisindeki metotlarla yüksek seviyeli olarak geliştirilir.

Java programlama dilinde bir metot içerisinde metot bildirimi yapılamaz.

Örneğin:

```
package csd;

class App {
	public static void main(String [] args)
	{
	}
}

class Sample {
	public static int foo()
	{
		public static void bar() //error
		{

		}
	}
}
```

Metodların Geri Dönüş Değerleri

Bir metot çağrıldığı zaman onun kodu çalıştırılır. Metot sona erdiğinde ismine geri dönüş değeri (return value) denilen bir değer elde edilir. Bu değer ifadelerde kullanılabilir. Metotların geri dönüş değerlerinin türü metot bildirim işleminde metot isminin soluna yazılır.

Örneğin:

```
public static int foo()
{
	//...
}
```

Metotların geri dönüş değerleri return deyimiyle oluşturulur. return deyiminin genel biçimi şöyledir:

```
return [ifade];
```

return deyiminin iki işlevi vardır:

1. Geri dönüş değerini oluşturur.
2. Metodu sonlandırır. Yani onun aşağısına yazacağımız kodlar çalıştırılmaz.

Metodun geri dönüş değeri varsa kesinlikle return işlemi uygulanmak zorundadır. Örneğin:

```
package csd;

class App {
	public static void main(String [] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		//...

		kb.close();
	}
}

class Sample {
	public static int foo() //error: return etmek zorunda
	{
		//...
	}
}
```

Ayrıca metodun her mümkün akışında return anahtar sözcüğünün görülmesi gerekir. Geri dönüş değeri türünün yerine void anahtar sözcüğü yazılırsa bu durum metodun geri dönüş değeri olmadığı anlamına gelir. Geri dönüş değeri olmayan metotlar geri dönüş değeri varmış gibi kullanılamaz. Örneğin:

```
package csd;

class App {
	public static void main(String [] args)
	{
		int result;

		result = Sample.foo();	//error: foo metodunun geri dönüş degeri  yok
	}
}

class Sample {
	public static void foo()
	{
		System.out.println("foo");
	}
}
```

void bir metotta return deyimi kullanılabilir. Bu bir zorunluluk değildir. void metotlarda return deyimi kullanılırsa return deyiminin yanına ifade yazılamaz. Bu durumda return metodu sonlandırmak için kullanılır. void metotlarda return kullanılmamışsa metot ana blok bittiğinde biter. Bir metodun geri dönüş değerinin olması onu kullanmayı zorunlu hale getirmez. Yani geri dönüş değerleri programcı tarafından kullanılmayabilir.

Metotların geri dönüş değerleri önce derleyici tarafından yaratılan bir geçici değişkene aktarılır. Oradan alınarak kullanılır. İfade sonunda o geri dönüş değeri yok edilir. Örneğin:

```
a = foo() * 8
```

ifadesinde aslında şunlar olmaktadır:

temp = return ifadesi;
a = temp * 8;
temp yok ediliyor

Yani aslında return işlemi geçici değişkene yapılan bir atama işlemidir. Metotların geri dönüş değerlerinin türü aslında yaratılacak olan geçici değişkenin türünü belirler.

Metotların Parametre Değişkenleri

Metotların parametre değişkenleri metotların girdileri olarak düşünülebilir. Metotların parametre değişkenleri metot bildiriminde parametre parantezinin içerisinde belirtilir. Parametre değişkenleri metot bildirimi sırasında tür ve değişken ismi virgül atomu ile ayrılacak şekilde yapılır:

```
public static void foo(int a, double b)
{
	//...
}
```

Parametre değişkenleri aynı türden olsa bile tür belirten sözcük her bir parametre için yazılmalıdır.

```
public static void foo(int a, b) //error
{
	//...
}
```

Bu metodun şu şekilde bildirilmesi gerekir:

```
public static void foo(int a, int b)
{
	//...
}
```

Anahtar Notlar: Java programlama dilinde (hatta bir çok programlama dilinde) metot bildiriminde kullanılan değişkenlere parametre, metodun çağrılması sırasında parantez içerisinde verilen ifadelere de argüman denilmektedir.

Parametreli bir metot parametre sayısı kadar argümanla çağrılmak zorundadır. Argümanlar herhangi bir ifade olabilir. Örneğin:

```
foo(10 + 8, 11 * 7);
```

Bir metoda argüman olarak başka bir metodun geri dönüş değeri verilebilir. Bu durumda önce arügman olarak verilen metod çağrılır, daha sonra metodun geri dönüş değeri argüman olarak aktarılır. Örneğin:

```
package csd;

class App {
	public static void main(String [] args)
	{
		int a = 10;
		double pi = 3.14159;

		Sample.foo(Sample.square(a), pi * pi);
	}
}

class Sample {
	public static int square(int val)
	{
```

```
		return val * val;
	}
	public static void foo(int a, double b)
	{
		System.out.println(a);
		System.out.println(b);
	}
}
```

Parametreli bir metot çağrıldığında önce argümanların değerleri hesaplanır. Sonra argümanlardan parametre değişkenlerine otomatik atama yapılır. Yani metod çağrılması sırasında verilen argümalar da parametrelere yapılan atama işlemidir. Sonra da akış metoda geçer.

```
package csd;

class App {
	public static void main(String [] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci sayıyı giriniz");
		int val1 = Integer.parseInt(kb.nextLine());

		System.out.println("İkinci sayıyı giriniz");
		int val2 = Integer.parseInt(kb.nextLine());

		int result = Operation.add(val1, val2);

		System.out.println(result);


		kb.close();
	}
}


class Operation {
	public static int add(int a, int b)
	{
		return a + b;
	}
}
```

Anahtar Notlar: Java programlama dilinde aşağıdaki durumlar aslında atama işlemleridir:

- Atama operatörü ile yapılan atamalar:

- Bir metodun geri dönüş değeri: Geçici değişkene yapılan atama

- Parametreli bir metodun çağrılması: Argümanlardan parametrelere yapılan atama

Metotların parametre değişkenleri yalnızca o metot içeirisinde kullanılabilir. Metotların parametre değişkenleri adeta metodun ana bloğunun başında bildirilmiş yerel değişkenler gibidir. Dolayısıyla metot içerisinde bir parametre değişkeni ile aynı isimli başka bir yerel değişken bildirilemez.

Matematiksel Metotlar

Java programlama dilinde java.lang paketi içerisinde bulunan Math sınıfında faydalı işlemler yapan matematiksel metotlar

bulunmaktadır. Bu metotların yazılması için için belirli bir düzeyde matematiksel bilgi gerekir. Programcı bu detayları bilmek zorunda olmadan metotları çağırabilir. Çok kullanılan bazı metodlar şunlardır:

- Math sınıfının sqrt metodu parametre olarak aldığı sayının karaköküne geri döner.

- Math sınıfının abs metodları parametre olarak aldığı sayının mutlak değerine geri döner.

- Math sınıfının sin, cos, tan, asin, acos, atan gibi metotları trigonometrik işlemleri yapar. Bu metodlardaki parametre ya da geri dönüş değerleri radyan cinsindedir.

- Math sınıfında radyan değerini dereceye çeviren ve derece değerini radyana çeviren sırasıyla toDegrees ve toRadians metotları vardır.

- Math sınıfının pow metodu kuvvet alma işlemi yapar.

- Math sınıfının exp metodu paramtre olarak aldığı değeri kuvvet kabul ederek e sayısının o kuvvetine geri döner.

- Math sınıfının log, log10 ve log1p metotları sırasıyla e tabanına göre logaritma [ln(val)], 10 tabanına göre logaritma ve paramtre olarak aldığı değerin 1(bir) fazlasının e tabanına göre logaritma değerlerine [ln(val + 1)] geri dönerler. Herhangi bir tabana göre logaritma işlemi için programcı bu metotları kullanabilir.

- Math sınıfının signum metodu parametre olarak aldığı değerin işaretini belirleyen metottur. Sayı negatifse -1, pozitifse 1 ve sıfırsa 0(sıfır) değerine geri döner.