Java Programlama Dilinin Temel Türleri

Tür bir değişkenin bellekte kaç byte yer kapladığını belirten ve onun içersine atanacak değerlerin hangi aralıkta ve nasıl bir formatta tutulduğunu anlatan bir bir bilgidir. Java da temel türler şu şekildedir:

Tür Belirten Anahtar Sözcük	**Uzunluk (Byte)**	**Sınır Değerler**
short	2	[-32768, +32767]
int	4	[-2147483648, +2147483647]
long	8	[-9223372036854775808, +9223372036854775807]
byte	1	[-128, +127]
float	4	[+-3.6*10^-38, +-3.6*10^+38]
double	8	[+-1.8*10^-308, +-1.8*10^+308]
boolean	-	true, false
char	2	[0, +65535]

- int işaretli bir tamsayı türüdür ve int türden bir değişken bu türün sınırları içerisindeki pozitif ya da negatif tamsayıları tutabilmektedir.

- Java da işaretsiz tamsayı türleri yoktur. Ancak Java 8 den itibaren 4(dört) byte'lık işaretsiz tamsayı işlemleri yapılabilmektedir. Bu konu ileride ele alınacaktır.

- short türü int türünün yarı uzunluğunda bir tamsayı türüdür.

- long türü en uzun tamsayı türüdür.

- byte türü bir byte lık bir tamsayı türüdür.

- float ve double türleri değerleri kayan noktalı (floating point) formatta tutan türlerdir. float ve double türleri sayıları sırasıyla tek hassasiyetli (single-precision) ve çift hassasiyetli (double-precision) olarak IEEE 754 standardına göre tutmaktadır. Bu türlerde bazı sayılar tam olarak ifade edilemezler. O sayıya yakın bir sayı ile ifade edilebilirler. Bir noktalı sayının ifade edilemeyip ona yakın olan bir sayının ifade edilmesiyle oluşan hataya yuvarlama hatası (rounding error) denilmektedir. Yuvarlama hataları değer ilk kez yerleştirilirken oluşabileceği gibi, bir işlemin sonucunda da oluşabilmektedir.

- float ve double türleri işaretli türlerdir.

- float türünün yuvarlama hatalarına karşı direnci zayıftır. Bu nedenle double türü daha fazla tercih edilmektedir.

- Yuvarlama hataları sebebiyle özellikle finansal uygulamalarda hatalı değerler oluşabilmektedir. Bunun için Java' da java.math paketi içerinde BigDecimal isimli bir sınıf bulunmaktadır. Bu tarz uygulamalarda BigDecimal sınıfı kullanılmalıdır. Bu konu uygulma kursunda ele alınacaktır.

- char türü Java' da UNICODE tablo dikkate alınarak belirlenmiştir. Bu nedenle char türü 2(iki)

byte'dır. Yani char türden bir değişkenin içerisine herhangi bir karakterin UNICODE tablosundaki sıra numarası yerleştirilebilir. UNICODE tablo dünyadaki neredeyse bütün dillerin karakterlerini ve pek çok özel karakteri ifade edebilme yeteneğine sahiptir. (www.unicode.org)

- Yazılar aslında her bir karakteri bir sayı olacak biçimde bellekte kodlanmaktadır. Hangi sayının hangi karakteri temsil ettiği bir tabloyla belirlenmektedir. ASCII tablosu 1(bir) byte lık yani 256 satırdan oluşan bir tabloyken UNICODE 2(iki) byte'lık yani 65536 satırdan oluşan bir tablodur.

Anahtar Notlar: ASCII tablosunun ilk hali 7(yedi) bittir. Daha sonra 8(sekiz) bitlik ASCII tablosu oluşturulmuştur.

- Yalnızca ASCII ve UNICODE değil pek çok karakter tabloları ve kodlamaları vardır. Buna "character encoding" denilmektedir. Bir yazı hangi tabloyla kodlanmışsa o tabloyla yorumlanmalıdır. Aksi halde asıl yazıyla alakası olmayan bir yazı görüntülenebilir. Karakter kodlama (character encoding) konusu ileride ele alınacaktır.

- boolean türü, yalnızca true ve false değerlerini tutabilen bir türdür. Java standartlarında boolean türünün uzunluğu belirtilmemiştir.

Java' da en çok kullanılan tamsayı türü int, en çok kullanılan gerçek sayı türü de double' dır. Programcılar öncelikle bu türleri tercih etmelidir. Eğer geçerli bir neden varsa diğer türleri kullanmalıdır.

Anahtar Notlar: Büyük ölçüde Java programlama dilinin özellikleri alınarak geliştirilmiş C# programlama dilinde tamsayı türlerinin ayrıca işaretsiz versiyonları da bulunmaktadır.

Anahtar Notlar: C# programlama dilinde decimal isimli bir tür bulunmaktadır. Bu tür doğal bir tür değildir. Microişlemcilerin decimal türü ile işlemler yapabilecek komutları yoktur. Bu işlemler dolaylı olarak yüzlerce komut ile yapılabilmektedir. Java' daki BigDecimal sınıfının karşılığı olarak C#' da decimal türü kullanılmaktadır.

Bildirim İşlemi

Kullanılmadan önce bir değişkenin derleyiciye tanıtılması işlemine bildirim (declaration) denir. Değişkenler bildirilmeden kullanılamazlar.

Bildirim işleminin genel biçimi şöyledir:

```
<tür> <degişken listesi>;
```

Örneğin:

```
int a, b, c;
double a;
float k;
```

Aynı türden değişkenler aralarına ","(virgül) konularak bildirilebilirler (paramtre değişkenleri için farklıdır). Örneğin:

```
int a, b, c;
```
Burada a, b ve c değişkenleri int türdendir. Yukarıdaki işlem şu şekilde de yapılabilir:

```
int a;
int b;
int c;
```

Java programlama dilinde değişken bildirimleri 3(üç) yerde yapılabilir:

1. Metotların içerisinde: Metotların içerisinde bildirilen değişkenlere yerel değişkenler (local variables) denilmektedir.
2. Metotların parametre parantezleri içerisinde: Bu tür değişkenlere parametre değişkenleri denilmektedir.
3. Metotların dışında fakat sınıfların içerisinde: Bu tür değişkenlere "sınıfların veri elemanları" denilmektedir.

Yerel Bildirimler

Metodların içerisinde yapılan bildirimlere yerel bildirimler denilmektedir. Bu biçimde bildirilen değişkenlere de yerel değişkenler denilmektedir. Yerel değişkenler metodun içerisinde herhangi bir yerde bildirilebilirler.

Anahtar Notlar: Bir değişkenin içerisindeki değerler print ve println metotlarıyla yazdırılabilirler. Örneğin:

```
int a;

a = 8;

System.out.println(a);
```

Değişkenlere İlkdeğer Verilmesi

Bildirim işleminin bir parçası olarak değişkenlere değer atanmasına ilk değer verme (initialization) denilmektedir. Örneğin:

```
int a = 8;
double x, y = 3.14, z;
```

İlk değer verme ve ilk kez değer verme işlemleri çoğu zaman işlevleri aynı olsa da teknik olarak aynı değildir.

```
int a = 8; // İlkdeğer verme işlemi

int b;

b = 11; // İlk kez değer verme işlemi
```

Değişkenlerin Faaliyet Alanları

Bildirilen bir değişkenin derleyici tarafından tanınabildiği (görülebildiği) program aralığına faaliyet alanı (scope) denir. Bir değişken yalnızca faaliyet alanı içerisinde kullanılabilir. Yani değişkenin ismi yalnızca o faaliyet alanında geçerlidir.

Yerel Değişkenlerin Faaliyet Alanları

{ ve } arasında kalan bölgelere blok denilmektedir. Bir metodun ana bloğu olmak zorundadır. O ana bloğun içerisinde istenildiği kadar iç içe ya da ayrık bloklar oluşturulabilir. Bir yerel değişken bildirildiği noktadan sonra ve bildirildiği bloğun sonuna kadarki bölgede faaliyet gösterir. Yani yerel değişkenlerin faaliyet alanı bildirildikleri yerden bildirildikleri bloğun sonuna kadardır. Yalnızca o bölgede kullanılabilirler:

```
package csd;

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

		{
			int b;

			{

				a = 20; // geçerli

				b = 8; // geçerli
			}
		}
		b = 34; // geçersiz
	}
}
```

Java'da bir yerel bir degiskenin bildirildigi noktadan itibaren ayni faaliyet alani icerisinde ayni isimde bir yerel degisken bildirilemez:

```
package csd;

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

		{
			double a; //error:

		}
	}
}
```

Ancak ayrık bloklarda aynı isimde değişken bildirimi yapılabilir:

```
package csd;

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

		{
			int a;
			//...
		}

		{
			float a;
			//...
		}

	}
}
```

Metot gövdeleri de aslında birer blok olduğundan, farklı metotlarda da aynı isimde değişkenler bildirilebilir. Bunlar aslında farklı değişkenlerdir. Örneğin:

```
package csd;

class App {
	public static void foo()
	{
		int a;
		// ...

	}

	public static void main(String [] args)
	{
		int a;
		//...



	}
}
```

Buna göre Java programlama dilinde aşağıdaki gibi bildirimler geçerlidir:

```
package csd;

class App {

	public static void main(String [] args)
	{

		{
			int a;
```

```
                a = 10;

                System.out.println(a);
        }

        float a;

        a = 34;

        System.out.println(a);

    }
}
```

Burada float türünde bildirilmiş a değişkeni int türünde bildirilmiş a değişkeni aynı faaliyet alanı içerisinde olmadığından herhangi bir sorun oluşmaz.

Anahtar Notlar: C# programlama dilinde Java programlama dilinden farklı olarak önce iç blokta sonra dış blokta da aynı isimli değişkenler bildirilemez. Aşağıdaki program C#' da geçersiz (error) Java' da geçerlidir:

```
package csd;

class App {

    public static void main(String [] args)
    {

        {
            int a;

            a = 10;

            System.out.println(a);
        }

        float a;

        a = 34;

        System.out.println(a);

    }
}
```

Java programlama dilinde değer atanmamış yerel değişkenlerin değerleri program içerisinde kullanılamaz:

```
package csd;

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

```
		int a;

		System.out.println(a);	// error: Deger atanmamış
	}
}
```

Değişken İsimlendirme Kuralları

Java' da değişken isimlendirme kuralları şu şekildedir:

- Değişken isimleri sayısal karakterle başlatılamaz. Ancak alfabetik, _(alttire) ya da $ karakterleri ile başlatılabilir. Bundan sonra nümerik karakterler olabilir.

- Değişken isimlendirmede UNICODE karakterler kullanılabilir. Yani isimler Türkçe karakterler içerebilirler.

- Java büyük harf küçük harf duyarlılığı olan (case sensitive) bir programlama dilidir. Yani büyük harfler ve küçük harfler tamamen farklı karakterler olarak ele alınmaktadır.

- Değişken isimlerinin anlamlı, telaffuz edilebilir çok da uzun olmayacak biçimde seçilmesi tavsiye edilmektedir. Bu bir sentaks kuralı değildir ancak programın okunabilirliği ve anlaşılabilirliği açısından önemlidir.

- Değişken isimlendirmede boşluk karakterleri de kullanılamaz.

- Değişken isimlerinde $ karakteri sentaks olarak kullanılabilmesine karşın tavsiye edilmez. Bazı program kodunu kendisi yazan araçlar nadir de olsa $ karakterini kullanabilmektedir. Fakat programcı kullanmamalıdır.

- Değişken isimleri doğrudan anahtar sözcüklerden oluşamaz. Java’nın yeni versiyonlarında bu kurala ilişkin bazı istisnalar bulunabilmektedir

- Değişken isimlerinin maksimum uzunluğu hakkında Java standartlarında açıkça birşey söylenmemiştir. Yeterince uzun olabilmektedir.

Klavyeden Değer Okunması

Java'da klavyeden değer okumak için birden fazla yöntem bulunmaktadır. Ancak burada basit olarak Scanner sınıfı kullanılacaktır. Scanner sınıfı Java 5 den sonra standart kütüphaneye eklenmiştir ve java.util paketi içerisinde bulunmaktadır.

Klavyeden int türden bir değer okuma işlemi aşağıdaki gibi yapılabilir:

```
package csd;

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

		System.out.println("Bir sayı giriniz");
```

```
		int val = Integer.parseInt(kb.nextLine());

		System.out.println(val * val);
	}
}
```

Klavyeden long türden bir değer okuma işlemi aşağıdaki gibi yapılabilir:

```
package csd;

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

		System.out.println("Bir sayı giriniz");
		long val = Long.parseLong(kb.nextLine());

		System.out.println(val * val);

	}
}
```

Klavyeden double türden bir değer okuma işlemi aşağıdaki gibi yapılabilir:

```
package csd;

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

		System.out.println("Bir sayı giriniz");
		double val = Double.parseDouble(kb.nextLine());

		System.out.println(val * val);
	}
}
```

İfade Kavramı

Değişkenlerin sabitlerin ve operatörlerin herhangi bir kombinasyonuna ifade (expression) denilmektedir. Örneğin:

```
a * b
x * 8 + 11
10
y
```

birer ifadedir.

Tek başına bir değişken ifade belirtir. Tek başına bir sabit de ifade belirtir. Ancak tek başına bir operatör

ifade belirtmez.