BÖLÜM 4

Değişken Tipleri

Java da, tüm değişkenlerin kullanılmadan önce tanımlanması edilmesi gerekir. Bir değişken tanımlamanın temel gösterimi bu şekildedir:

```
type identifier [ = value][, identifier [= value] ...] ;
```

Belirtilen türde birden çok değişkeni tanımlamak etmek için, virgülle ayrılmış liste kullanın.

Burada çeşitli türde değişken tanımlamaları ile ilgi birkaç örnek gösterilmektedir. Bazılarının initialization(ilk değerini atama) içerdiğini unutmayın.

```
int a, b, c;          // declares three ints, a, b, and c.
int d = 3, e, f = 5; // declares three more ints, initializing
                      // d and f.
byte z = 22;          // initializes z.
double pi = 3.14159; // declares an approximation of pi.
char x = 'x';         // the variable x has the value 'x'.
```

Bu bölüm, Java dilinde mevcut olan çeşitli değişken tiplerini açıklamaktadır. Java da 3 çeşit değişken türü vardır.

- Yerel Değişkenler
- Instance(Oluşum) Değişkenleri
- Sınıf/static Değişkenleri

4.1 Yerel Değişkenler

Yerel değişkenler metotlar, constructorlar veya bloklar içinde tanımlanır.

Yerel değişkenler metotlar, constructor'lar veya bloklar girildiği zaman oluşturulur ve değişken, metot, constructor veya bloktan bir kez çıktığında yok edilecektir.

Access modifier'ları yerel değişkenler için kullanılamaz.

Yerel değişkenler, sadece yazıldığı metot, constructor veya block içinde görünürdür.

Yerel değişkenler için bir önceden tanımlanmış bir değer yoktur, bu yüzden yerel değişken tanımlanmalı ve başlangıç değeri, ilk kullanımdan önce atanmalıdır.

Örnek:

Burada, *age* bir yerel değişkendir. *pupAge()* metodu içinde tanımlanmıştır ve kapsamı sadece bu metotla sınırlıdır.

```
public class Test{
   public void pupAge(){
      int age = 0;
      age = age + 7;
      System.out.println("Puppy age is : " + age);
   }
}
```

```
public class example{

   public static void main(String args[]){
      Test test = new Test();
      test.pupAge();
   }
}
```

Bu aşağıdaki sonucu üretecektir:

```
Puppy age is: 7
```

Örnek:

Aşağıdaki örnek *age* değişkenini başlangıç değerini atamadan kullanmaktadır, bu yüzden derleme sürecinde bir hata verecektir.

```
public class Test{
   public void pupAge(){
      int age;
      age = age + 7;
      System.out.println("Puppy age is : " + age);
   }
}
```

```
public class example{
   public static void main(String args[]){
      Test test = new Test();
      test.pupAge();
   }
}
```

Derleme sırasında aşağıdaki hata üretilecektir:

```
Test.java:4:variable number might not have been initialized
age = age + 7;
          ^
1 error
```

4.2 Instance(Oluşum) Değişkenleri

Instance değişkenleri bir sınıf içinde fakat bir metot, constructor veya blok dışında deklare edilen değişkenlerdir.

Instance değişkenleri, bir nesne oluşturulduğu zaman 'new' anahtar sözcüğü kullanılarak oluşturulur ve bir nesne yok edildiği zaman yok edilir.

Instance değişkenlere, acces modifierlar verilebilir.

Instance değişkenleri bir sınıf içindeki bütün metotlar, constructorlar ve bloklar için görünürdür. Normalde, bu değişkenleri private olarak tanımlamak önerilmektedir(erişim düzeyinde)

Instance değişkenler, default değerlere sahiptirler. Sayılar için default değer 0, Boolean'lar için false, nesne referansları için null'dır. Değerler, tanımlama sürecinde veya constructor içinde atanabilir.

Instance değişkenlere, sınıf içindeki değişken ismini çağırarak doğrudan erişilebilir. Ancak farklı sınıflar içinde , tam nitelikli adı kullanılarak çağrılmalıdır.

```
ObjectReference.VariableName
```

Örnek:

```
import java.io.*;

public class Employee{
   // this instance variable is visible for any child class.
   public String name;

   // salary  variable is visible in Employee class only.
   private double salary;

   // The name variable is assigned in the constructor.
   public Employee (String empName){
      name = empName;
   }

   // The salary variable is assigned a value.
   public void setSalary(double empSal){
      salary = empSal;
   }

   // This method prints the employee details.
   public void printEmp(){
      System.out.println("name  : " + name );
      System.out.println("salary :" + salary);
   }
}
```

```
public class testclass{

   public static void main(String args[]){
      Employee empOne = new Employee("Ransika");
      empOne.setSalary(1000);
      empOne.printEmp();
   }
}
```

Bu aşağıdaki sonucu üretecektir:

```
name  : Ransika
salary :1000.0
```

4.3 Sınıf/static değişkenleri

Sınıf değişkenleri, ayrıca static değişkenler olarak da bilinir, *static* anahtar sözcüğü kullanılarak bir sınıf içinde fakat bir metot, constructor veya blok dışında tanımlanan edilen değişkenlerdir.

Bir sınıftan kaç tane nesne oluşturulmuş olursa olsun, sınıf değişkenlerinin her sınıf için sadece bir kopyası olacaktır.

Static değişkenler, constantları(sabitleri) tanımlarken kullanılır. Constantlar(sabitler), public/private, final ve static şeklinde tanımlanmış değişkenlerdir. Constant değişkenlerin ilk değerleri asla değişmez.

Static değişlenler, program başlatılınca oluşturulur ve durdurulunca yok edilir.

Görünülebilirlik, instance değişkenleri ile benzerdir. Fakat, çoğu static değişken sınıfın kullanıcıları için kullanılabilir olmak zorunda olduğundan, public olarak tanımlanır.

Static değişkenlere, sınıfın adı çağırılarak erişilebilir. `ClassName.VariableName`

Sınıf değişkenini, public static final olarak deklare ettiğimiz zaman, değişken isimleri(constantlar) büyük harflerle yazılmalıdır.

Örnek:

```
import java.io.*;

public class Employee{
   // salary  variable is a public static variable
   public static double salary;

   // DEPARTMENT is a constant
   public static final String DEPARTMENT = "Development ";

}
```

```
public class EmployeeTest{

public static void main(String args[]){

         Employee empOne = new Employee();
         empOne.salary = 1000;
        System.out.println(empOne.DEPARTMENT+"average salary:"+empOne.salary);
   }
}
```

:

Bu aşağıdaki sonucu üretecektir:

```
Development average salary:1000
```

Not: Eğer değişkenlere sınıfın dışından erişim varsa, constant'a Employee.DEPARTMENT olarak erişilmelidir.