Java - Objek dan Kelas

Java adalah Bahasa Berorientasi Objek. Sebagai bahasa yang memiliki fitur Berorientasi Objek, Java mendukung konsep dasar berikut :

- Polimorfisme
- Inheritance / Warisan
- Encapsulation /Enkapsulasi
- Abstraksi
- Kelas
- Objek
- Instance
- metode
- Message Passing

Dalam bab ini, kita akan melihat konsep - Kelas dan Objek.

- Objek - Objek memiliki status dan perilaku. Contoh: Seekor anjing memiliki keadaan - warna, nama, ras serta perilaku - mengibaskan ekor, menggonggong, makan. Objek adalah turunan dari kelas.
- Kelas - Sebuah kelas dapat didefinisikan sebagai template / cetak biru yang menggambarkan perilaku / keadaan yang didukung oleh objek jenisnya.

Objek di Java

Sekarang mari kita lihat lebih dalam apa itu objek. Jika kita mempertimbangkan dunia nyata, kita dapat menemukan banyak benda di sekitar kita, mobil, anjing, manusia, dll. Semua benda ini memiliki keadaan dan perilaku.

Jika kita menganggap seekor anjing, maka statusnya adalah - nama, ras, warna kulit, dan perilakunya adalah - menggonggong, mengibas-ngibaskan ekor, berlari.

Jika Anda membandingkan objek perangkat lunak dengan objek dunia nyata, mereka memiliki karakteristik yang sangat mirip.

Objek perangkat lunak juga memiliki status dan perilaku. Status objek perangkat lunak disimpan dalam field dan perilaku/behavior ditampilkan melalui metode.

Jadi dalam pengembangan perangkat lunak, metode beroperasi pada keadaan internal suatu objek dan komunikasi objek-ke-objek dilakukan melalui metode.

Kelas di Java

Kelas adalah cetak biru dari mana objek individu dibuat.

Berikut adalah contoh kelas.

Contoh :

```
public class Dog {
   String breed;
   int age;
   String color;

   void barking() {
   }

   void hungry() {
   }

   void sleeping() {
   }
}
```

Kelas dapat berisi salah satu tipe variabel berikut.

- Variabel lokal - Variabel yang didefinisikan di dalam metode, konstruktor atau blok disebut variabel lokal. Variabel akan dideklarasikan dan diinisialisasi dalam metode dan variabel akan dimusnahkan ketika metode telah selesai.
- Variabel instans - Variabel instans adalah variabel dalam kelas tetapi di luar metode apa pun. Variabel-variabel ini diinisialisasi saat kelas dibuat. Variabel instance dapat diakses dari dalam metode, konstruktor, atau blok apa pun dari kelas tertentu itu.
- Variabel kelas - Variabel kelas adalah variabel yang dideklarasikan di dalam kelas, di luar metode apa pun, dengan kata kunci statis.

Kelas dapat memiliki sejumlah metode untuk mengakses nilai berbagai jenis metode. Dalam contoh di atas, menggonggong(), lapar() dan tidur() adalah metode.

Berikut adalah beberapa topik penting yang perlu didiskusikan saat mempelajari kelas Bahasa Java.

Konstruktor

Saat membahas tentang kelas, salah satu sub topik terpenting adalah konstruktor. Setiap kelas memiliki konstruktor. Jika kita tidak secara eksplisit menulis konstruktor untuk kelas, kompilator Java akan membangun konstruktor default untuk kelas itu.

Setiap kali objek baru dibuat, setidaknya satu konstruktor akan dipanggil. Aturan utama konstruktor adalah mereka harus memiliki nama yang sama dengan kelasnya. Sebuah kelas dapat memiliki lebih dari satu konstruktor.

Berikut adalah contoh konstruktor -

Contoh

```
public class Puppy {
   public Puppy() {
   }

   public Puppy(String name) {
      // This constructor has one parameter, name.
   }
}
```

Java juga mendukung Kelas Singleton di mana Anda hanya dapat membuat satu instance kelas.

Catatan - Kami memiliki dua jenis konstruktor. Kami akan membahas konstruktor secara rinci di bab-bab selanjutnya.

Membuat Objek

Seperti yang disebutkan sebelumnya, kelas menyediakan cetak biru untuk objek. Jadi pada dasarnya, sebuah objek dibuat dari sebuah kelas. Di Java, kata kunci baru digunakan untuk membuat objek baru.

Ada tiga langkah saat membuat objek dari kelas -

- Deklarasi - Deklarasi variabel dengan nama variabel dengan tipe objek.
- Instansiasi - Kata kunci 'baru' digunakan untuk membuat objek.
- Inisialisasi - Kata kunci 'baru' diikuti dengan panggilan ke konstruktor. Panggilan ini menginisialisasi objek baru.

Berikut adalah contoh membuat objek -

Contoh

```
public class Puppy {
   public Puppy(String name) {
      // This constructor has one parameter, name.
      System.out.println("Passed Name is :" + name );
   }

   public static void main(String []args) {
      // Following statement would create an object myPuppy
      Puppy myPuppy = new Puppy( "tommy" );
   }
}
```

Jika kita melakukan compile dan menjalankan program diatas, maka akan menghasilkan hasil sebagai berikut:

Keluaran

```
Passed Name is :tommy
```

Mengakses Variabel dan Metode Instans

Variabel dan metode instance diakses melalui objek yang dibuat. Untuk mengakses variabel instance, berikut adalah jalur yang sepenuhnya memenuhi syarat -

```
/* First create an object */
ObjectReference = new Constructor();

/* Now call a variable as follows */
ObjectReference.variableName;

/* Now you can call a class method as follows */
ObjectReference.MethodName();
```

Contoh

Contoh ini menjelaskan cara mengakses variabel instan dan metode kelas.

```
public class Puppy {
   int puppyAge;

   public Puppy(String name) {
      // This constructor has one parameter, name.
      System.out.println("Name chosen is :" + name );
   }

   public void setAge( int age ) {
      puppyAge = age;
   }

   public int getAge( ) {
      System.out.println("Puppy's age is :" + puppyAge );
      return puppyAge;
   }

   public static void main(String []args) {
      /* Object creation */
      Puppy myPuppy = new Puppy( "tommy" );

      /* Call class method to set puppy's age */
```

```
        myPuppy.setAge( 2 );

        /* Call another class method to get puppy's age */
        myPuppy.getAge( );

        /* You can access instance variable as follows as well
*/
        System.out.println("Variable Value :" + myPuppy.puppyAge
);
    }
}
```

Jika kita melakukan compile dan menjalankan program diatas, maka akan menghasilkan hasil sebagai berikut:

Keluaran

```
Name chosen is :tommy
Puppy's age is :2
Variable Value :2
```

Aturan Deklarasi File Sumber

Sebagai bagian terakhir dari bagian ini, sekarang mari kita lihat aturan deklarasi file sumber. Aturan ini penting saat mendeklarasikan kelas, pernyataan *import*, dan pernyataan *package* dalam file sumber.

- Hanya boleh ada satu kelas publik per file sumber.
- File sumber dapat memiliki beberapa kelas non-publik.
- Nama kelas publik harus menjadi nama file sumber juga yang harus ditambahkan oleh .java di akhir. Misalnya: nama kelasnya adalah *kelas publik Employee {}* maka file sumbernya adalah Employee.java.
- Jika kelas didefinisikan di dalam sebuah paket, maka pernyataan paket harus menjadi pernyataan pertama dalam file sumber.
- Jika ada pernyataan import, maka itu harus ditulis di antara pernyataan paket dan deklarasi kelas. Jika tidak ada pernyataan paket, maka pernyataan import harus menjadi baris pertama di file sumber.
- Pernyataan impor dan paket akan menyiratkan ke semua kelas yang ada di file sumber. Tidak mungkin mendeklarasikan import dan / atau pernyataan paket yang berbeda ke kelas yang berbeda di file sumber.

Kelas memiliki beberapa tingkat akses dan terdapat berbagai jenis kelas; kelas abstrak, kelas akhir, dll. Kami akan menjelaskan tentang semua ini di bab pengubah akses.

Selain jenis kelas yang disebutkan di atas, Java juga memiliki beberapa kelas khusus yang disebut kelas dalam dan kelas Anonymous.

Package / Paket Java

Dengan kata sederhana, ini adalah cara mengkategorikan kelas dan antarmuka. Saat mengembangkan aplikasi di Java, ratusan kelas dan antarmuka akan ditulis, oleh karena itu mengkategorikan kelas-kelas ini adalah suatu keharusan dan juga membuat hidup lebih mudah.

Pernyataan Import

Di Java, jika nama yang sepenuhnya memenuhi syarat, yang menyertakan paket dan nama kelas diberikan, maka kompilator dapat dengan mudah menemukan kode sumber atau kelas. Pernyataan import adalah cara untuk memberikan lokasi yang tepat bagi compiler untuk menemukan kelas tertentu tersebut.

Misalnya, baris berikut akan meminta kompilator untuk memuat semua kelas yang tersedia di direktori java_installation / java / io -

```
import java.io.*;
```

Studi Kasus Sederhana

Untuk studi kasus akan membuat dua kelas. Mereka adalah Employee dan EmployeeTest.

Pertama buka notepad dan tambahkan kode berikut. Ingat ini adalah kelas Karyawan dan kelas tersebut adalah kelas umum. Sekarang, simpan file sumber ini dengan nama Employee.java.

Kelas Karyawan memiliki empat variabel contoh - nama, usia, penunjukan dan gaji. Kelas memiliki satu konstruktor yang didefinisikan secara eksplisit, yang mengambil parameter.

Contoh

```
import java.io.*;
public class Employee {

   String name;
   int age;
   String designation;
   double salary;

   // This is the constructor of the class Employee
```

```
   public Employee(String name) {
      this.name = name;
   }

   // Assign the age of the Employee  to the variable age.
   public void empAge(int empAge) {
      age = empAge;
   }

   /* Assign the designation to the variable designation.*/
   public void empDesignation(String empDesig) {
      designation = empDesig;
   }

   /* Assign the salary to the variable salary.*/
   public void empSalary(double empSalary) {
      salary = empSalary;
   }

   /* Print the Employee details */
   public void printEmployee() {
      System.out.println("Name:"+ name );
      System.out.println("Age:" + age );
      System.out.println("Designation:" + designation );
      System.out.println("Salary:" + salary);
   }
}
```

Seperti yang disebutkan sebelumnya dalam tutorial ini, pemrosesan dimulai dari metode utama. Oleh karena itu, agar kita dapat menjalankan kelas Karyawan ini, harus ada metode utama dan objek harus dibuat. Kami akan membuat kelas terpisah untuk tugas-tugas ini.

Berikut adalah kelas *EmployeeTest* , yang membuat dua instance kelas Employee dan memanggil metode untuk setiap objek untuk menetapkan nilai untuk setiap variabel.

Simpan kode berikut di file EmployeeTest.java.

```
import java.io.*;
public class EmployeeTest {

   public static void main(String args[]) {
      /* Create two objects using constructor */
      Employee empOne = new Employee("James Smith");
      Employee empTwo = new Employee("Mary Anne");
```

```
        // Invoking methods for each object created
        empOne.empAge(26);
        empOne.empDesignation("Senior Software Engineer");
        empOne.empSalary(1000);
        empOne.printEmployee();

        empTwo.empAge(21);
        empTwo.empDesignation("Software Engineer");
        empTwo.empSalary(500);
        empTwo.printEmployee();
    }
}
```

Sekarang, kompilasi kedua kelas tersebut lalu jalankan *EmployeeTest* untuk melihat hasilnya sebagai berikut -

Keluaran

```
C:\> javac Employee.java
C:\> javac EmployeeTest.java
C:\> java EmployeeTest
Name:James Smith
Age:26
Designation:Senior Software Engineer
Salary:1000.0
Name:Mary Anne
Age:21
Designation:Software Engineer
Salary:500.0
```