JAVA - OBJECT & CLASSES

Java is an Object-Oriented Language. As a language that has the Object Oriented feature, Java supports the following fundamental concepts:

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Parsing

In this chapter, we will look into the concepts Classes and Objects.

- **Object -** Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors -wagging, barking, eating. An object is an instance of a class.
- **Class -** A class can be defined as a template/blue print that describes the behaviors/states that object of its type support.

Objects in Java:

Let us now look deep into what are objects. If we consider the real-world we can find many objects around us, Cars, Dogs, Humans, etc. All these objects have a state and behavior.

If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging, running

If you compare the software object with a real world object, they have very similar characteristics.

Software objects also have a state and behavior. A software object's state is stored in fields and behavior is shown via methods.

So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

Classes in Java:

A class is a blue print from which individual objects are created.

A sample of a class is given below:

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

   void barking(){
   }

   void hungry(){
   }

   void sleeping(){
   }
}
```

A class can contain any of the following variable types.

- **Local variables:** Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables:** Instance variables are variables within a class but outside any method. These variables are instantiated when the class is loaded. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables:** Class variables are variables declared with in a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.

Below mentioned are some of the important topics that need to be discussed when looking into classes of the Java Language.

Constructors:

When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class the Java compiler builds a default constructor for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Example of a constructor is given below:

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

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

Java also supports Singleton Classes where you would be able to create only one instance of a class.

Creating an Object:

As mentioned previously, a class provides the blueprints for objects. So basically an object is created from a class. In Java, the new key word is used to create new objects.

There are three steps when creating an object from a class:

- **Declaration:** A variable declaration with a variable name with an object type.
- **Instantiation:** The 'new' key word is used to create the object.
- **Initialization:** The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Example of creating an object is given below:

```
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" );
```

```
    }
}
```

If we compile and run the above program, then it would produce the following result:

```
Passed Name is :tommy
```

Accessing Instance Variables and Methods:

Instance variables and methods are accessed via created objects. To access an instance variable the fully qualified path should be as follows:

```
/* 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();
```

Example:

This example explains how to access instance variables and methods of a class:

```
public class Puppy{

   int puppyAge;

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

If we compile and run the above program, then it would produce the following result:

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

Source file declaration rules:

As the last part of this section let's now look into the source file declaration rules. These rules are essential when declaring classes, *import* statements and *package* statements in a source file.

- There can be only one public class per source file.

- A source file can have multiple non public classes.
- The public class name should be the name of the source file as well which should be appended by **.java** at the end. For example : The class name is . *public class Employee{}* Then the source file should be as Employee.java.
- If the class is defined inside a package, then the package statement should be the first statement in the source file.
- If import statements are present then they must be written between the package statement and the class declaration. If there are no package statements then the import statement should be the first line in the source file.
- Import and package statements will imply to all the classes present in the source file. It is not possible to declare different import and/or package statements to different classes in the source file.

Classes have several access levels and there are different types of classes; abstract classes, final classes, etc. I will be explaining about all these in the access modifiers chapter.

Apart from the above mentioned types of classes, Java also has some special classes called Inner classes and Anonymous classes.

Java Package:

In simple, it is a way of categorizing the classes and interfaces. When developing applications in Java, hundreds of classes and interfaces will be written, therefore categorizing these classes is a must as well as makes life much easier.

Import statements:

In Java if a fully qualified name, which includes the package and the class name, is given then the compiler can easily locate the source code or classes. Import statement is a way of giving the proper location for the compiler to find that particular class.

For example, the following line would ask compiler to load all the classes available in directory java_installation/java/io :

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

A Simple Case Study:

For our case study, we will be creating two classes. They are Employee and EmployeeTest.

First open notepad and add the following code. Remember this is the Employee class and the class is a public class. Now, save this source file with the name Employee.java.

The Employee class has four instance variables name, age, designation and salary. The class has one explicitly defined constructor, which takes a parameter.

```
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);
   }
}
```

As mentioned previously in this tutorial, processing starts from the main method. Therefore in-order for us to run this Employee class there should be main method and objects should be created. We will be creating a separate class for these tasks.

Given below is the *EmployeeTest* class, which creates two instances of the class Employee and invokes the methods for each object to assign values for each variable.

Save the following code in EmployeeTest.java file

```
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();
   }
}
```

Now, compile both the classes and then run *EmployeeTest* to see the result as follows:

```
C :> javac Employee.java
C :> vi EmployeeTest.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
```

What is Next?

Next session will discuss basic data types in Java and how they can be used when developing Java applications.