Programming with Python and Java
(CS 29008)

School of Electronics Engineering, KIIT DU

Chapter 3

Lab 3: Class, Objects and Methods in Java

The class is at the core of Java. It forms the basis for object-oriented programming in Java. Any concept we wish to implement in a Java program must be encapsulated within a class.

A class is declared by use of the **class** keyword. A simplified general form of a **class** definition is shown below:

Listing 3.1: *class* syntax

```
class classname {
  dtype variable1;
  dtype variable2;
  //...
  dtype variableN;

  dtype methodname1 (parameter-list) {
     // body of method1
  }
  dtype methodname2 (parameter-list) {
     // body of method2
  }
  //...
  dtype methodnameN (parameter-list) {
     // body of methodN
  }
}
```

The data or variables defined within a **class** are called *instance-variables*. The code is contained within *methods*. Collectively, the methods and vari-

ables defined within a class are called *members* of the class. As a general rule, it is the methods that determine how a class data can be used.

Note that the general form of a class does not specify a **main()** method. Java classes do not need to have a **main()** method. We only specify one if that class is the starting point for the program. Further, some kinds of Java applications, such as applets, do not require a **main()** method at all.

3.1 A Simple Class

Let us illustrate the class with a simple example. Here, a class called **Box** defines three instance-variables: **width**, **height**, and **depth**. Currently, **Box** does not contain any methods (but will be added later).

Listing 3.2: An example class *Box*

```
class Box {
  double width;
  double height;
  double depth;
}
```

A class defines a new type of data. In this case, the new data type is called **Box**. We will use this name to declare objects of type **Box**. It is important to remember that a class declaration only creates a template; it does not create an actual object.

To actually create a **Box** object, we use a statement like the following:

```
Box mybox = new Box(); // create a Box object called mybox
```

mybox is now an instance of class **Box**. Thus, it will have "physical" reality. Every **Box** object will contain its own copies of the instance-variables **width**, **height**, and **depth**. To access these variables, we use the **dot (.)** operator. For example, to assign the **width** variable of **mybox** the value 100, we use the following statement:

```
mybox.width = 100;
```

Here is a Java program that uses the **Box** class.

```
class Box {
  double width;
  double height;
  double depth;
}
class BoxDemo {
```

```
    public static void main(String args[]) {
        Box mybox = new Box();
        double vol;

        // assign values to mybox's instance-variables
        mybox.width = 10;
        mybox.height = 20;
        mybox.depth = 15;

        // compute volume of box
        vol = mybox.width * mybox.height * mybox.depth;
        System.out.println("Volume:-" + vol);
    }
}
```

We should compile the file **BoxDemo.java** because the **main()** method is in the **BoxDemo** class. After compiling **BoxDemo.java**, Java compiler will automatically create two **.class** files- **Box.class** and **BoxDemo.class**. To run the program, we must execute **BoxDemo.class** which displays output **Volume: 3000.0**.

3.2 Declaring Objects

When we create a class, we are creating a new data type. We can use this type to declare objects of that type. To do this, we use **new** operator. The **new** operator dynamically allocates (that is, allocates at run-time) memory for an object and returns a reference to it. This reference is the address in memory of the object allocated by **new**. This reference is then stored in the variable. Thus, in Java, all class objects must be dynamically allocated.

In the above code, we declare an object of type **Box** in the following manner:

```
Box mybox = new Box();
```

Alternatively, we can write the above statement in the following manner:

```
Box mybox; // declare reference to object
mybox = new Box(); // allocate a Box object
```

3.3 Methods

Typically, a class consists of two things: instance-variables and methods. The general form of a method is:

```
dtype methodname(parameter-list) {
    // body of the method
}
```

Here, **dtype** refers the type of data returned by the method. The return type can be any valid built-in data type or a user-defined class type.

Listing 3.3: Adding methods to *Box* class

```
class Box {
  double width;
  double height;
  double depth;

  // compute and return volume
  double volume() {
      return width * height * depth;
  }
}
class BoxDemo {
  public static void main(String args[]) {
     Box mybox = new Box();
     double vol;

     // assign values to mybox's instance-variables
     mybox.width = 10;
     mybox.height = 20;
     mybox.depth = 15;

     // get volume of box
     vol = mybox.volume();
     System.out.println("Volume:-" + vol);
  }
}
```

Several methods need parameters. The example code shown below illustrates the use of parameterized methods.

Listing 3.4: Parameterized methods in class *Box*

```
class Box {
```

```
  double width;
  double height;
  double depth;

  // compute and return volume
  double volume() {
      return width * height * depth;
  }

  // sets dimensions of Box
  void setDim(double w, double h, double d) {
      width = w;
      height = h;
      depth = d;
  }
}
class BoxDemo {
  public static void main(String args[]) {
     Box mybox = new Box();
     double vol;

     // initialize mybox
     mybox.setDim(10, 20, 15);

     // get volume of box
     vol = mybox.volume();
     System.out.println("Volume:-" + vol);
  }
}
```

3.4 Nested Class in Java

An example of a nested class in Java is shown.

```java
import java.util.Scanner;
class Shape {
    String name;
    static class Square {
        int side;
        int sqArea() {
            return side * side;
        }
    }
    static class Rectangle {
        int width;
        int height;
        int recArea() {
            return width * height;
        }
    }
```

```java
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter shape name: ");
        String str = sc.nextLine();
        Shape sh = new Shape();
        sh.name = str;
        if(sh.name.equals("Square") || sh.name.equals("square")) {
            Square sq = new Square();
            System.out.print("Enter the side of square: ");
            sq.side = sc.nextInt();
            System.out.println("Area of square: " + sq.sqArea());
        }
        else if(sh.name.equals("Rectangle") || sh.name.equals("rectangle")) {
            Rectangle rec = new Rectangle();
            System.out.print("Enter width: ");
            rec.width = sc.nextInt();
            System.out.print("Enter height: ");
            rec.height = sc.nextInt();
            System.out.println("Area of rectangle: " + rec.recArea());
        }
        else {
            System.out.println("Invalid shape entered!");
        }
    }
}
```

3.5 Lab 3 Exercises

Objectives:

- To learn Java programs related to class, objects and methods.

Outcomes:

- After completing this, the students would be able to develop Java programs using class, objects and methods.

Lab Assignments

1. Modify the Java program **BoxDemo.java** to do the following:

   1. create two objects of class **Box**

   2. assign values to two objects using objects' instance-variables and get the volumes of two boxes

   3. set values to two objects using parameterized methods and get the volumes of two boxes

2. Write a Java program to calculate the area and perimeter of a rectangle using the concepts of class, objects and methods.

3. Create a Java class called **uniMember** which has instance-variables **name** and **gender**. Within this class, create two more classes, **Student** with instance-variable **roll_number** and **Faculty** with instance-variable **employee_id**. Write the Java methods to enter the details (name, gender, roll_number, employee_id) of a student and a faculty and display the same on the console.