Programming in Java

Lecture 5: Classes, Objects, Methods and Constructors

Contents

What is a class?

- A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support.
- A class is the blueprint from which individual objects are created.
- A class defines a new data type which can be used to create objects of that type. Thus, a class is a template for an object, and an object is an instance of a class.

- A class is declared by using **class** keyword.

```
class classname
          {
          type instance-variable1;
          type instance-variable2;
          …
          type instance variable N;
          type methodname1(parameter-list) {
                    // body of method
                    }
          type methodname2(parameter-list) {
                    // body of method
                    }
           ...
          type methodnameN(parameter-list) {
                    // body of method
                    }
          }
```

- The data, or variables, defined within a class are called ***instance variables*** because each instance of the class (that is, each object of the class) contains its own copy of these variables.

- The code is contained within ***methods.***

- The methods and variables defined within a class are called ***members of the class.***

- Java classes do not need to have a **main( ) method** till JDK 1.6**.** We only specify one main() if that class is the starting point for your program.

Defining Classes

- The basic syntax for a class definition:

```
      class  ClassName
      {
                  [fields declaration]
                  [methods declaration]
}
```

- Bare bone class – no fields, no methods

```
public class Circle {
        // my circle class
}
```

Object

- An object is an instance of the class which has well-defined attributes and behaviors.
- Obtaining objects of a class is a two-step process.
- First, you must declare a variable of the class type. This variable does not define an object. Instead, it is simply a variable that can *refer to an object.*
- Second, you must acquire an actual physical copy of the object and assign it to that variable. You can do this using the **new operator.**

- Similar to variables we can define object of the class.

  class_name Obj;

  Obj= new class_name();

- It can be rewritten like

  class_name Obj= new class_name();

- The *new* operator dynamically allocates memory for an object.

  class var = new classname( );

- A class creates a logical framework that defines the relationship between its members while An object has physical reality. (That is, an object occupies space in memory.)

Method

- A method is a construct for grouping statements together to perform a function.
- A method that returns a value is called a *value retuning method*, and the method that does not return a value is called *void method*.

- In some other languages, methods are referred to as *procedures* or *functions*.
- A method which does not return any value is called a procedure.
- A method which returns some value is called a function.

Methods

- A class with only data fields has no life. Objects created by such a class cannot respond to any messages.
- Classes usually consist of two things: instance variables and methods. General form of a method is:

  ```
  type name(parameter-list)
  {
  // body of method
  }
  ```

- Methods that have a return type other than void return a value to the calling routine using the following form of the return statement:

  ```
  return value;
  ```

Invoking Methods

- We've seen that once an object has been instantiated, we can use the *dot operator* to invoke its methods

```
count = title.length()
```

- A method may *return a value*, which can be used in an assignment or expression

- A method invocation can be thought of as asking an object to perform a service

```
class Box
{   double width;
    double height;
    double depth;
    double volume()
    {       return width * height * depth;
}   }
class BoxDemo4
{   public static void main(String args[])
    {       Box mybox1 = new Box();
            double vol;
            mybox1.width = 10;
            mybox1.height = 20;
            mybox1.depth = 15;
            vol = mybox1.volume();
            System.out.println("Volume is " + vol);
}   }
```

Adding a Method That Takes Parameters

- While some methods don’t need parameters, most do. Parameters allow a method to be generalized.

```
int square(int i)
{
return i * i;
}
```

- square( ) will return the square of whatever value it is called with.

java.util.Scanner

- byte nextByte()
- short nextShort()
- int nextInt()
- long nextLong()
- float nextFloat()
- double nextDouble()
- String next()
- String nextLine()

Let’s Do It

- Create a class Student having attributes name, fatherName, rollNo, section, college and address. Write a menu driven program to enter and display the details of Students.

Brainstorming Questions

```
class Demo
{
public static void main(String[] args) {
                int x = 5, y;
                while (++x < 7) {
                          y = 2;
                }
                System.out.println(x + y);
      }
}
```

- A) 7
- B) 8
- C) 9
- D) a compilation error

- Ans-D

Brainstorming Questions

- class Demo
- {
- public static void main(String... args) {
-                         System.out.println("JavaChamp");
-          }
- }
- A) The program will compile and run fine printing JavaChamp as output
- B) The program will compile fine but won't run correctly, a NoSuchMethodError exception would be thrown
- C) There is a compilation error at declaring the main() argument, should be an array of String instead
- D) Runtime error

- Ans-A

Let’s Do It

- Create a class named MyTriangle that contains the following two methods:

  //Return true if the sum of any two sides is greater than the third side.

  public static boolean isValid(double side1, double side2, double side3)
- //Return the area of the triangle.

  public static double area(double side1, double side2, double side3)
- Write a test program that takes three sides for a triangle as i/p and computes the area if the input is valid. Otherwise, it displays that the input is invalid.

Constructors

- A **constructor** is a special method that is used to **initialize a newly created object.**
- It has the same name as the class in which it resides and is syntactically similar to a method.
- Once defined, the constructor is automatically called immediately after the object is created, before the new operator completes.
- It can be used to initialize the objects ,to **required, or default values** at the time of object creation.
- It is **not mandatory** for the coder to write a constructor for the class.

Default Constructor

- If no user defined constructor is provided for a class, compiler initializes member variables to its default values.
  - numeric data types are set to 0
  - char data types are set to null character(‘’)
  - reference variables are set to null

- In order to create a Constructor observe the following rules:
  - It has the **same name** as the class
  - It should not return a value not even ***void***

```
class Box
{   double width;   double height; double depth;
    Box() {width = 10;       height = 10;        depth = 10;}
double volume()
    {      return width * height * depth;
}   }
class BoxDemo4
{   public static void main(String args[])
    {      Box mybox1 = new Box();
           double vol;
           vol = mybox1.volume();
           System.out.println("Volume is " + vol);
}   }
```

Parameterized Constructors

```
class Box
{   double width; double height; double depth;
Box(double w, double h, double d) { width = w; height = h; depth = d;}
double volume()
    {       return width * height * depth;
}   }
class BoxDemo4
{   public static void main(String args[])
    {       Box mybox1 = new Box(10, 20, 15);
            double vol;
            vol = mybox1.volume();
            System.out.println("Volume is " + vol);
}   }
```