OBJECT ORIENTED PROGRAMMING WITH JAVA

Encapsulation in Java

Debasis Samanta
Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Class Concept in Java

Concept of encapsulation

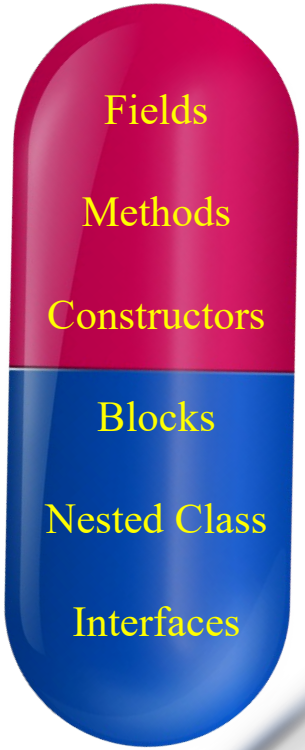
- The object is at the core of Java programming.
- Java provides the concept of class to build objects.
- A class defines the shape and working of an object.
- The concept of class is the logical construct upon which the entire Java language is built.

What is a class?

- A **class** is a group of objects, which have common properties.
- It is a template or blueprint from which objects are created.
- It is a logical entity.

A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class(es) and interface(s)

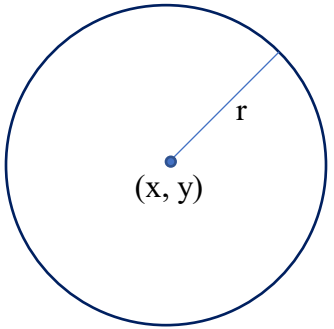


General structure of a class

```
class <class-name>{
        <type> <variable 1>;
        <type> <variable 2>;
        <type> <variable 3>;
        ...
        <type> <variable n>;

        <type> <method 1>(<parameter-list 1>){
                    // Body of the method 1
        }
        <type> <method 1>(<parameter-list 2>){
                    // Body of the method 2
        }
        ...
        <type> <method 1>(<parameter-list m>){
                    // Body of the method m
        }
}
```

Java class – An example

```
class Circle {
	double x, y; // The coordinates of the center
	double r;   // The radius
}
```



Adding methods to Circle class

```
class Circle {
      double x,y; // The coordinates of the center
      double r;   // The radius

      // Method that returns circumference
      double circumference(){
                  return 2*3.14159*r;
      }
      // Method that returns area
      double area(){
                  return (22/7)*r*r;
      }
}
```

Declaring object of type Circle class

```java
// A program that uses the circle class
// Call this file circledemo1.java
class Circle {
  double x,y; // The coordinates of the center
  double r;   // The radius

  // Method that returns circumference
  double circumference(){
    return 2*3.14159*r;
  }
  // Method that returns area
  double area(){
    return (22/7)*r*r;
  }
}
```

```java
//The following class declare an object of type Circle
class CircleDemo1 {
   public static void main(String args[]){
     Circle c = new Circle();
     c.x = 0.0;
     c.y = 0.0;
     c.r = 5.0;
     System.out.println("Circumference" + c.circumference());
     System.out.println("Area" + c.area());
   }
}
```

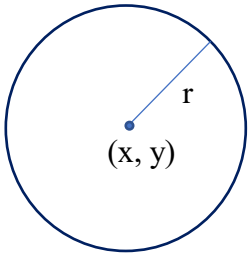
Multiple objects declaration

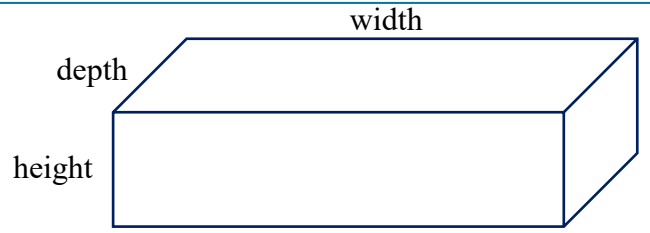
```
// A program that declares two objects of the Circle class
// Call this file CircleDemo2.java
class Circle {
   double x, y;
   double r;

   double circumference(){
     return 2*3.14159*r;
   }
   double area(){
     return (22/7)*r*r;
   }
}
```

```
//The following class declares multiple objects of type Circle
class CircleDemo2 {
   public static void main(String args[]){
      Circle c1 = new Circle();
      Circle c2 = new Circle();
      // Initialize the circles
      c1.x = 3.0;
      c1.y = 4.0;
      c1.r = 5.0;
      c2.x = -4.0;
      c2.y = -8.0;
      c2.r = 10.0;
      System.out.println("Circumference Circle 1" + c1.circumference());
      System.out.println("Area Circle 1" + c1.area());
      System.out.println("Circumference Circle 2" + c2.circumference());
      System.out.println("Area Circle 2" + c2.area());
   }
}
```

Multiple Classes Declaratio

Multiple classes in a program






```
class Circle {
   double x,y;
   double r;
   double circumference(){
      return 2*3.14159*r;
   }
   double area(){
      return (22/7)*r*r;
   }
}
```

```
class Box{
   double width;
   double height;
   double depth;
   double area(){
      double a;
       a = (width*height + height*depth + width*depth) * 2;
       return a;
   }
   double volume(){
       double v;
       v = width*height*depth;
       return v;
    }
}
```

Multiple class objects

```
// Declaring objects of type Circle and Box
class MulticlassDemo {
   public static void main(String args[]){
     Circle c = new Circle();
     Box b = new Box();
     // Initialize the circles
     c.x = 3.0; c.y = 4.0; c.r = 5.0;
     b.width = 3.0; b.height = 4.0; b.depth = 5.0;
     System.out.println("Circumference Circle" + c.circumference());
     System.out.println("Area Circle" + c.area());
     System.out.println("Area of Box" + b.area());
     System.out.println("Volume of Box" + b.volume());
   }
}
// Save this file as MulticlassDemo.java
```

Important notes

1. There should be a class which contains a method `main()`. This class is called main class.

2. There should be only one main class.

3. The name of the program file should be same as the name of the main class followed by .java as an extension.

4. If there is no main class, then there should be compilation error.

Java program without main class

```
class Circle {
   double x, y;
   double r;
   double circumference(){
      return 2*3.14159*r;
   }
   double area(){
      return (22/7)*r*r;
   }
}
class Box{
    double width;
    double height;
    double depth;
    double area(){
        double a;
        a = (width*height + height*depth + width*depth) * 2;
        return a;
     }
     double volume(){
        double v;
        v = width*height*depth;
        return v;
     }
}
```

Name the file as Test.java.

This program reports compilation error as follows.

Error: Main method not found in class Circle, please define the main method as:
public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

Method with parameters

```
class Circle {
    double x,y;
    double r;
    double circumference(){
       return 2*3.14159*r;
    }
    double area(){
       return (22/7)*r*r;
    }
    void setCircle(double a, double b, double c){
        x = a; // Set center x-coordinate
        y = b; // Set center y-coordinate
        r = c; // Set radius
    }
}
```

```
class CircleDemo3 {
   public static void main(String args[]){
      Circle c1 = new Circle();
      Circle c2 = new Circle();
      // Initialize the circles
      c1.setCircle(3.0,4.0,5.0);
      c2.setCircle(-4.0,8.0,10.0);
      System.out.println("Circumference Circle 1" + c1.circumference());
      System.out.println("Area of circle 1" + c1.area());
      System.out.println("Circumference Circle 2" + c2.circumference());
      System.out.println("Area of circle 2" + c2.area());
   }
}
```

Name the file as CircleDemo3.java

Constructors

Constructor for automatic object initialization

1. It can be tedious to initialize all of the variables in a class each time an object is instantiated.

2. Java allows objects to initialize themselves when they are created/ declared.

3. This automatic initialization is performed through the concept of constructor.

Constructors – Some properties

1. A constructor initializes an object immediately upon creation.
2. Constructor in Java is a method.
3. This method has the same name as the class in which it resides.
4. Once defined, the constructor is automatically called immediately after object is created.
5. Constructor is a method which has no return type.
6. In fact, the implicit return type of a class constructor is the class type itself.
7. Constructor initialize the internal state of an object.

Constructor : An example

```
class Circle {
     double x,y;
     double r;
     double circumference(){
         return 2*3.14159*r;
     }
     double area(){
         return (22/7)*r*r;
     }

     Circle (double a, double b, double c){
         x = a; // Set center x-coordinate
         y = b; // Set center y-coordinate
         r = c; // Set radius
     }
}
```

```
class CircleDemo4 {
     public static void main(String args[]){
        Circle c1 = new Circle(3.0,4.0,5.0);
        Circle c2 = new Circle(-4.0,8.0,10.0);
        System.out.println("Circumference Circle 1" + c1.circumference());
        System.out.println("Area Circle 1" + c1.area());
        System.out.println("Circumference Circle 2" + c2.circumference());
        System.out.println("Area Circle 2" + c2.area());
     }
}
```

The `this` Keyword

`this` keyword concept

1. `this` is used to reduce name-space collisions.
2. Sometimes a method will need to refer to the object that invoked it.
3. To allow this Java defines `this` keyword.
4. `this` can be used inside any method to refer to the current object.
5. That is, `this` is always a reference to the object on which the method is invoked.

Constructor : An example

```
class Circle {
    double x,y;
    double r;
    double circumference(){
       return 2*3.14159*r;
    }
    double area(){
        return (22/7)*r*r;
    }
    void setCircle(double a, double b, double c){
         x = a; // Set center x-coordinate
         y = b; // Set center y-coordinate
         r = c; // Set radius
    }
}
```

```
class Circle {
    double x,y;
    double r;
    double circumference(){
        return 2*3.14159*r;
    }
    double area(){
        return (22/7)*r*r;
    }
    Circle (double x, double y, double r){
       this.x = x; // Set center x-coordinate
       this.y = y; // Set center y-coordinate
       this.r = r; // Set radius
   }
}
```

```
class CircleDemo5 {
    public static void main(String args[]){
        Circle c1 = new Circle();
        c1.setCircle(3.0,4.0,5.0);
        Circle c2 = new Circle (-4.0,8.0,10.0);
        System.out.println("Circumference Circle 1" + c1.circumference());
        System.out.println("Area Circle 1" + c1.area());
        System.out.println("Circumference Circle 2" + c2.circumference());
        System.out.println("Area Circle 2" + c2.area())
    }
}
```

Name the file as
CircleDemo5.java

Multiple constructors

1. Sometimes, it is necessary to initialize an object in a number of ways.

2. Java allows his using the concept of constructor overloading.

3. In other words, Java allows to declare one or more constructor method with different lists of parameters and different method definition.

Constructor overloading : An example

```
class Circle {
    double x,y;
    double r;
    Circle (double x, double y, double r){
        this.x = x; this.y = y; this.r = r;
    }
    Circle (double r){
        x = 0; y=0; this.r = r;
    }
    Circle (Circle c){
        x = c.x;  y = c.y;   r = c.r;
    }
    Circle (){
        x = 0.0;  y = 0.0;   r = 1.0;
    }
    double circumference(){
       return 2*3.14159*r;
    }
    double area(){
       return (22/7)*r*r;
    }
}
```

```
class CircleDemo6 {
     public static void main(String args[]){
        Circle c1 = new Circle(3.0,4.0,5.0);
        Circle c2 = new Circle(5.0);
        Circle c3 = new Circle(c1);
        Circle c4 = new Circle();
        System.out.println("Circumference Circle 1" + c1.circumference());
        System.out.println("Area Circle 1" + c1.area());
        System.out.println("Circumference Circle 2" + c2.circumference());
        System.out.println("Area Circle 2" + c2.area());
        System.out.println("Circumference Circle 3" + c3.circumference());
        System.out.println("Area Circle 3" + c3.area());
        System.out.println("Circumference Circle 4" + c4.circumference());
        System.out.println("Area Circle 4" + c4.area());
    }
}
```

Name the file as CircleDemo6.java

`this` keyword again

1. There is a specialized use of `this` keyword that arises when a class has multiple constructors.

2. In that case, `this` can be used from one constructor to invoke one of the other constructor of the same class.

this with multiple constructors : An example

```
class Circle {
    double x, y;
    double r;
    Circle (double x, double y, double r){
       this.x = x; this.y = y; this.r = r;
    }
    Circle (double r){
       this(0.0, 0.0, r);
    }
    Circle (Circle c){
       this(c.x, c.y. c.r);
    }
    Circle (){
       this(0.0, 0.0, 1.0);
    }
    double circumference(){
       return 2*3.14159*r;
    }
    double area(){
       return (22/7)*r*r;
    }
}
```

Note:

1. There is a very important restriction on the `this` syntax.
2. It should appear only as the first statement in a constructor.

Questions to think…

- How Java developer makes the language simple compared to other language?
- Are there any specific things in Java, which a programmer should aware about those?


Thank You