OOPS (JAVA)
Lec-13

Saif Nalband

Contents

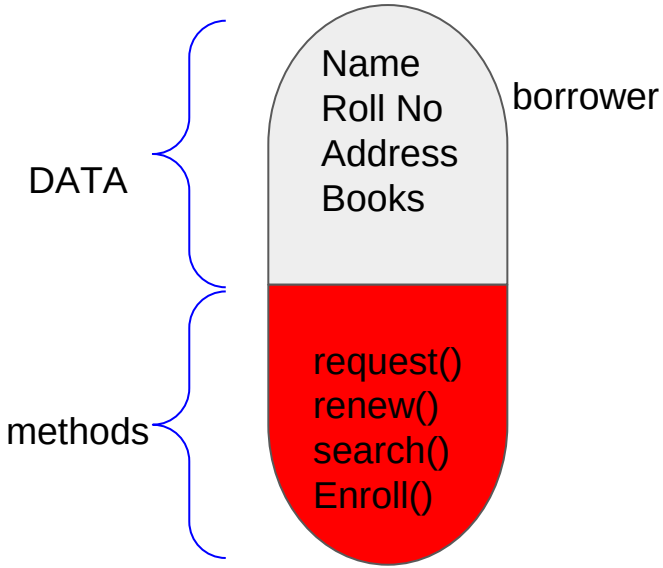
- Revisit previous class
- Constructors
- Methods
- Method vs constructors
- Constructor overloading
- Method overloading
- Garbage Collector

What is a Class ?

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

A class in JAVA can contain :

- Data
- Methods
- Constructors
- Blocks
- Nested Class(es) and interfaces



JAVA Class -- An Example

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

Methods with parameters Demo114

```
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 Demo114 {
   public static void main(String[]
args){
       Circle c1 = new Circle();
       Circle c2 = new Circle();
       // Initilise the circle
       c1.setCircle(1.0, 4.0, 5.0);
       c2.setCircle(3.0, 4.0, 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());
   }
}
```

Constructors 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 object in initialise themselves when they are created/ declared
3. This automatic initialization is performed through the **concept of constructors**

Constructors - Some properties

1. A constructors **initialises an object** immediately upon creation
2. Constructors in JAVA is a **method**
3. This method has the **same name** as the class in which it resides.
4. Once defined, the contractors is **automatically called** immediately after object is created.
5. Constructors is a method which has **no return type**
6. Infact, the implicit return type of a class constructors is the class itself.
7. Constructors initialize the internal state of an object

Constructors: An example Demp131/Demo131a

```
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 Demo120{
    public static void main(String arge[]){
        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());
    }
}
```

Difference Between Constructors and Method

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.

this keyword concept

1. this is used to reduce namespace collisions.
2. Sometimes a method will need to refer to the object that invoked
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 : Demo131

```
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){
        this.x = a; // Set center x-
coordinate
        this.y = b; // Set center y-
coordinate
        this.r = c; // Set radius
    }
}
```

```
class Demo121{
    public static void main(String args[]){
        Circle c2 = new Circle(-4.0,8.0,10.0);
        System.out.println("Circumference
Circle 2" + c2.circumference());
        System.out.println("Area Circle 2" +
c2.area());
            }
}
```

Constructor Overloading : Demo122

Small Real time Application : **DEMO123**

Method Overloading

- In JAVA, it is possible to define **two or more methods** within **same class** that share the **same name** considering that, their **parameters declaration are different**.
- This method is said to be **OVERLOAD**
- When an overloaded method is invoked, JAVA uses **type** and or **number of arguments** as its guide to determine *which version of overload method is called*.

Ways to perform Overloading Method

There are two ways to overload the method in java

- By changing number of arguments
- By changing the data type

Garbage Collector

- new keyword create dynamic objects.
- Destroyed by *garbage collector.*
- It works like when no reference to object is made, the object is assumed to be no longer needed.
- Different that C/C++

Thank you and
Stay Home and Stay safe