Java - Introduction to Programming
Lecture 9

OBJECT ORIENTED PROGRAMMING SYSTEMS (OOPS)

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts defined below :

Class is a user-defined data type which defines its properties and its functions. Class is the only logical representation of the data. For example, Human being is a class. The body parts of a human being are its properties, and the actions performed by the body parts are known as functions. The class does not occupy any memory space till the time an object is instantiated.

Object is a run-time entity. It is an instance of the class. An object can represent a person, place or any other item. An object can operate on both data members and member functions.

Example 1:

```
class Pen{
    String color;
    String type;

    public void write(){
        System.out.println("writing something");
    }
    public void printColor(){
        System.out.println(this.color);
    }
}

public class OOPS {
    public static void main(String[] args) {
        Pen pen1 = new Pen();
        pen1.color = "blue";
        pen1.type = "gel";

        Pen pen2 =new Pen();
        pen2.color = "black";
        pen2.type = "ballpoint";

        pen1.printColor();
        pen2.printColor();
    }
}
```

'this' keyword : 'this' keyword in Java that refers to the current instance of the class. In OOPS it is used to:

1. pass the current object as a parameter to another method
2. refer to the current class instance variable

Example 2:

```
class Student{
    String name;
    String age;

    public void printInfo(){
        System.out.println(this.name);
        System.out.println(this.age);
    }
}

public class OOPS {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Aman";
        s1.age = "24";

        s1.printInfo();
    }
}
```

Note: When an object is created **using** a new keyword, then space is allocated for the variable in a heap, and the starting address is stored in the stack memory.

Constructor: Constructor is a special method, which is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally.

- Constructors have the same name as class or structure.
- Constructors do not have a return type. (Not even void)
- Constructors are only called once, at object creation.

There can be **three types** of constructors in Java.

1. Non-Parameterized constructor: A constructor, which has no argument, is known as non-parameterized constructor (or no-argument constructor). It is invoked at the time of creating an object. If we don't create one then it is created by default by Java.

```java
class Student{
    String name;
    String age;

    public void printInfo(){
        System.out.println(this.name);
        System.out.println(this.age);
    }
    Student(){
        System.out.println("Constructor Called");
    }
}

public class OOPS {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Aman";
        s1.age = "24";
        s1.printInfo();
    }
}
```

2. Parameterized constructor: Constructor, which has parameters, is called a parameterized constructor. It is used to provide different values to distinct objects.

```java
class Student{
    String name;
    int age;

    public void printInfo(){
        System.out.println(this.name);
        System.out.println(this.age);
    }
    Student(String name, int age ){
        this.name = name;
        this.age = age;
    }
}

public class OOPS {
    public static void main(String[] args) {
        Student s1 = new Student("Aman",45);
        s1.printInfo();
    }
}
```

3. Copy Constructor: A Copy constructor is an **overloaded** constructor used to declare and initialize an object from another object. There is only a user defined copy constructor in Java(C++ has a default one too).

```
class Student{
    String name;
    int age;

    public void printInfo(){
        System.out.println(this.name);
        System.out.println(this.age);
    }
    Student(Student s2){
        this.name = s2.name;
        this.age = s2.age;
    }
    Student(){

    }
}

public class OOPS {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "aman";
        s1.age = 24;

        Student s2 = new Student(s1);
        s2.printInfo();

    }
}
```

Note : Unlike languages like C++, Java has no Destructor. Instead, Java has an efficient garbage collector that deallocates memory automatically.