

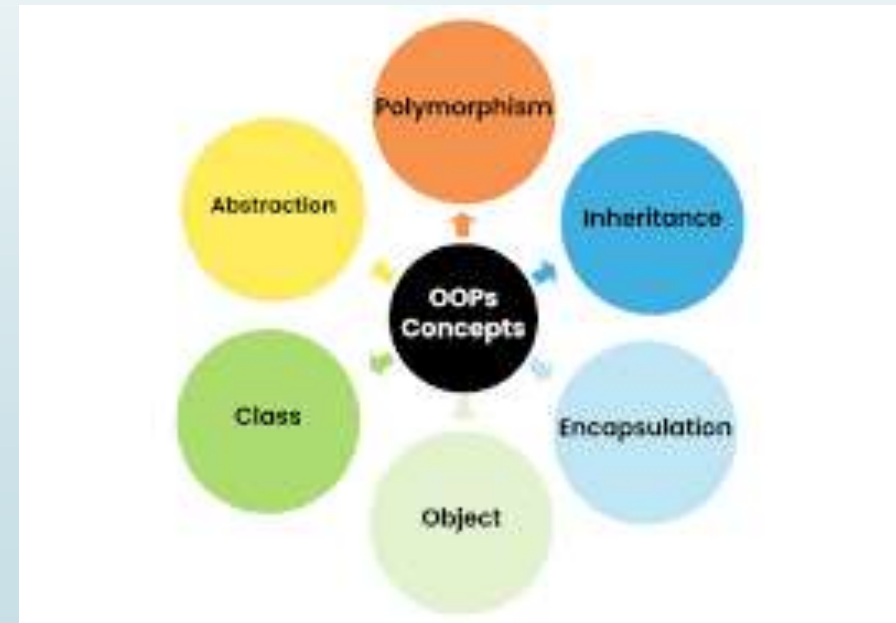
Java Object Oriented Programming, Class, Class Properties, Objects, new keyword, Object Creation and instanceof keyword


-K.L.Madhavi

Java Object Oriented Programming

Object oriented programming is a methodology to design a program using classes and objects. The main aim of object-oriented programming is to implement real-world entities. It simplifies software development and maintenance by providing some concepts:

- 1.Object
- 2.Class
- 3.Inheritance
- 4.Polymorphism
- 5.Abstraction
- 6.Encapsulation





➤ **Object:** Any entity that has state and behavior is known as object. It is an instance of class. An object contains an address and takes up some space in memory.

1.State: It is represented by the attributes of an object. It also reflects the properties of an object.

2.Behavior: It is represented by the methods of an object. It also reflects the response of an object to other objects.

Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

➤ **Class:** It is blueprint of an object. Without class we cannot create an object. A class contains no of objects. It does not consume any memory space.

➤ **Inheritance:** When one class(child class) acquires all the properties of another class(parent class) is known as inheritance. We are achieving inheritance by using **extends** keyword. It provides code reusability.

➤ **Polymorphism:** Performing same task in different ways is known as polymorphism. Therefore it can be achieved in java by:

1.Method overloading → compile time polymorphism

2.Method overriding → run time polymorphism

- 
- **Abstraction:** Showing essential parts and hiding implementation or internal details.

Ex:Phone call-here we don't know the internal processing,Android application etc.

- **Encapsulation:**Binding data(Variables) and code(Methods) into single unit or single entity.

Ex:

```
class student
{
    int rollno
    char name[]
    void read()
    {
        //body
    }
    void write()
    {
        //body
    }
}
```



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. For example, Student is a class while a particular student named Ravi is an object.

► Properties of Java Classes

1. Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
2. Class does not occupy memory.
3. Class is a group of variables of different data types and a group of methods.
4. A Class in Java can contain:
 - Data member
 - Method
 - Constructor
 - Nested Class
 - Interface

Declaration of class:

```
access_modifier class <class_name>
{
    data member;
    method;
    constructor;
    nested class;
    interface;
}
```

Components of class:

class declarations can include these components:

1. **Modifiers** : A class can be public or has default access.
2. **Class keyword**: class keyword is used to create a class.
3. **Class name**: The name should begin with an initial letter (capitalized by convention).
4. **Superclass(if any)**:The name of the parent class(super class), if any, preceded by the keyword extends. A class(sub class) can only extend one parent.
5. **Interfaces(if any)**: A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. **Body**: The class body is surrounded by braces, { }.



Object

- Definitions of an object:
 - An object is *a real-world entity*.
 - An object is *a runtime entity*.
 - The object is *an entity which has state and behavior*.
 - The object is *an instance of a class*.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.



Object Creation

In Java, we cannot execute any program without creating an object. There is various ways to create an object in Java:

- Using **new** Keyword
- Using **clone()** method
- Using **newInstance()** method of the **Class** class
- Using **newInstance()** method of the **Constructor** class
- Using **Deserialization**

Using New Keyword

Using the new keyword is the most popular way to create an object or instance of the class. When we create an instance of the class by using the new keyword, it allocates memory (heap) for the newly created object and also returns the reference of that object to that memory. The new keyword is also used to create an array.

The **syntax** for creating an object is:

```
ClassName object = new ClassName();
```

Process of creation of object using new keyword:

It involves 3 steps:

i.Declaration:A variable declaration with a variable name with an object type.

ii.Instantiation:The new keyword is used to create an object.

iii.Initialization:The new keyword is followed to call a constructor.This initializes the object.



Example Program:

```
public class CreateObjectExample1
{
    void show()
    {
        System.out.println("obj is created");
    }
    public static void main(String[] args)
    {
        //creating an object using new keyword
        CreateObjectExample1 obj = new CreateObjectExample1();
        //invoking method using the object
        obj.show();
    }
}
```

Instanceof keyword

- The **java instanceof keyword** is used to test whether the object is an instance of the specified class.
- The instanceof in java is also known as type *comparison operator* because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has null value, it returns false.

Example Program:

```
class Simple1
{
    public static void main(String args[])
    {
        Simple1 s=new Simple1();
        System.out.println(s instanceof Simple1);//true
        System.out.println(p instanceof Simple1);//false
    }
}
```