JAVA

OOPs (Object-Oriented Programming System)

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

Object

Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the 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

Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

* **Super Class:**The class whose features are inherited is known as superclass(or a base class or a parent class).
* **Sub Class:** The class that inherits the other class is known as subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
* **Reusability:**Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Polymorphism

If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

*Polymorphism in Java are mainly of 2 types:*

1. [*Overloading*](https://www.geeksforgeeks.org/overloading-in-java/)
2. [*Overriding*](https://www.geeksforgeeks.org/overriding-in-java/)

// Java program to Demonstrate Polymorphism

// This class will contain

// 3 methods with same name,

// yet the program will

// compile & run successfully

public class Sum {

// Overloaded sum().

// This sum takes two int parameters

public int sum(int x, int y)

{

return (x + y);

}

// Overloaded sum().

// This sum takes three int parameters

public int sum(int x, int y, int z)

{

return (x + y + z);

}

// Overloaded sum().

// This sum takes two double parameters

public double sum(double x, double y)

{

return (x + y);

}

// Driver code

public static void main(String args[])

{

Sum s = new Sum();

System.out.println(s.sum(10, 20));

System.out.println(s.sum(10, 20, 30));

System.out.println(s.sum(10.5, 20.5));

}

}

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing. Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviours of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

In Java, we use abstract class and interface to achieve abstraction.

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines. It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

* Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.
* As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.
* Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.