Object Oriented Programming (OOPs) Concept in Java

Difficulty Level : Easy Last Updated : 22 Dec, 2021

As the name suggests, Object-Oriented Programming or OOPs refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Let us do discuss pre-requisites by polishing concepts of methods declaration and passing. Starting off with the method declaration, it consists of six components:

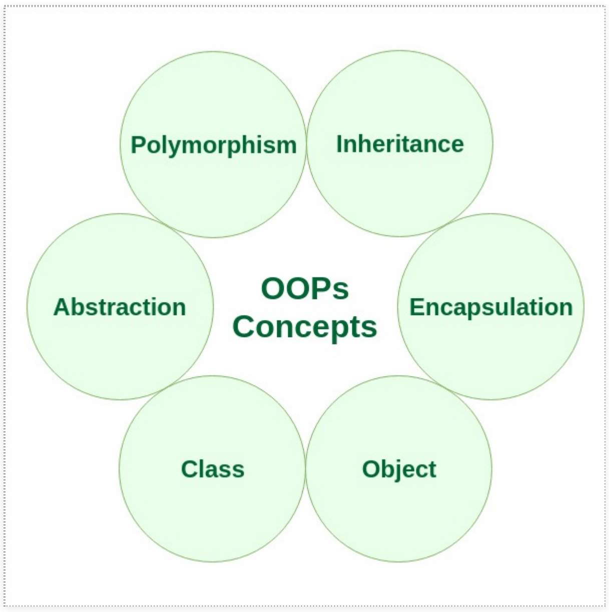
- **Access Modifier**: Defines **access type** of the method i.e. from where it can be accessed in your application. In Java, there 4 type of the access specifiers.
  - **public:** accessible in all class in your application.
  - **protected:** accessible within the package in which it is defined and in its **subclass(es)(including subclasses declared outside the package)**
  - **private:** accessible only within the class in which it is defined.
  - **default (declared/defined without using any modifier):** accessible within same class and package within which its class is defined.
- **The return type**: The data type of the value returned by the method or void if does not return a value.
- **Method Name**: the rules for field names apply to method names as well, but the convention is a little different.
- **Parameter list**: Comma separated list of the input parameters are defined, preceded with their data type, within the enclosed parenthesis. If there are no parameters, you must use empty parentheses ().
- **Exception list**: The exceptions you expect by the method can throw, you can specify these exception(s).
- **Method body**: it is enclosed between braces. The code you need to be executed to perform your intended operations.

Message Passing: Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

Now with basic prerequisite to step learning 4 pillar of OOPS is as follows. Let us start with learning about the different characteristics of an Object-Oriented Programming language

OOPs Concepts are as follows:

1. Class
2. Object
3. Method and method passing
4. Pillars of OOPS
    - Abstraction
    - Encapsulation
    - Inheritance
    - Polymorphism
        - Compile-time polymorphism
        - Run-time polymorphism



A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. **Modifiers**: A class can be public or has default access (Refer this for details).
2. **Class name:** The name should begin with a initial letter (capitalized by convention).
3. **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.

4. **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.
5. **Body:** The class body surrounded by braces, { }.

Object is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

1. **State** : It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity** : It gives a unique name to an object and enables one object to interact with other objects.
4. **Method:** A method is a collection of statements that perform some specific task and return result to the caller. A method can perform some specific task without returning anything. Methods allow us to **reuse** the code without retyping the code. In Java, every method must be part of some class which is different from languages like C, C++ and Python.
   Methods are **time savers** and help us to **reuse** the code without retyping the code.

Let us now discuss 4 pillars of OOPS:

Pillar 1: Abstraction

Data Abstraction is the property by virtue of which only the essential details are displayed to the user.The trivial or the non-essentials units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.
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.
Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

In java, abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces.

Pillar 2: Encapsulation

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.

Pillar 3: Inheritence

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

Let us discuss some of frequent used important terminologies:

- **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.

Pillar 4: Polymorphism

It refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities.

Note: *Polymorphism in Java are mainly of 2 types:*

1. *Overloading*
2. *Overriding*

Example

```
// 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));
    }
}
```

Output:

```
30
60
31.0
```