# **Object Oriented Programming (OOPs) in Java**

Object-Oriented Programming or OOPs refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code. Objects are seen by the viewer or user, performing tasks assigned by you. 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.

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
     + Runtime polymorphism

# 

# **Classes and Objects in Java**

In Java, classes and objects are basic concepts of Object Oriented Programming (OOPs) that are used to represent real-world concepts and entities. The class represents a group of objects having similar properties and behavior. For example, the animal type **Dog** is a class while a particular dog named **Tommy** is an object of the **Dog** class.

## **Java Classes**

A class in Java is a set of objects which shares common characteristics/ behavior and common properties/ attributes. It is a user-defined blueprint or prototype from which objects are created. 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

### **Class Declaration in Java**

*access\_modifier* **class** <*class\_name*>

{

data member;

method;

constructor;

nested class;

interface;

}

## **Java Objects**

An object in Java is a basic unit of Object-Oriented Programming and represents real-life entities. Objects are the instances of a class that are created to use the attributes and methods of a class. 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 the 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.

Example of an object: dog



*When we create an object which is a non primitive data type, it’s always allocated on the heap memory.*

### **Declaring Objects (Also called instantiating a class)**

When an object of a class is created, the class is said to be **instantiated**. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

| **Class** | **Object** |
| --- | --- |
| Class is the blueprint of an object. It is used to create objects. | An object is an instance of the class. |
| No memory is allocated when a class is declared. | Memory is allocated as soon as an object is created. |
| A class is a group of similar objects. | An object is a real-world entity such as a book, car, etc. |
| Class is a logical entity. | An object is a physical entity. |
| A class can only be declared once. | Objects can be created many times as per requirement. |
| An example of class can be a car. | Objects of the class car can be BMW, Mercedes, Ferrari, etc. |

# **Java Methods**

The **method in Java** or Methods of Java is a collection of statements that perform some specific task and return the result to the caller. A Java method can perform some specific task without returning anything. Java Methods allow us to **reuse** the code without retyping the code. In Java, every method must be part of some class that is different from languages like C, C++, and Python.

1. A method is like a function i.e. used to expose the behavior of an object.  
2. It is a set of codes that perform a particular task.

## **Syntax of Method**

<access\_modifier> <return\_type> <method\_name>( list\_of\_parameters)

{

//body

}

**Advantage of Method**

* Code Reusability
* Code Optimization

*Methods are time savers and help us to reuse the code without retyping the code.*

### **Method Declaration**

In general, method declarations have 6 components:

**1. Modifier:** It defines the **access type** of the method i.e. from where it can be accessed in your application. In Java, there are 4 types of access specifiers.

* **public:** It isaccessible in all classes in your application.
* **protected:** It is accessible within the class in which it is defined and in its subclass/es
* **private:** It isaccessible only within the class in which it is defined.
* **default:** It is declared/defined without using any modifier. It is accessible within the same class and package within which its class is defined.

***Note:*** *It is* ***Optional*** *in syntax.*

**2. The return type:** The data type of the value returned by the method or void if does not return a value. It is **Mandatory** in syntax.

**3. Method Name:** the rules for field names apply to method names as well, but the convention is a little different. It is **Mandatory** in syntax.

**4. Parameter list:** Comma-separated list of the input parameters is defined, preceded by their data type, within the enclosed parentheses. If there are no parameters, you must use empty parentheses (). It is **Optional** in syntax.

**5. Exception list:** The exceptions you expect by the method can throw, you can specify these exception(s). It is **Optional** in syntax.

**6. Method body:** it is enclosed between braces. The code you need to be executed to perform your intended operations. It is **Optional** in syntax.



## **Types of Methods in Java**

There are two types of methods in Java:

### **1. Predefined Method**

In Java, predefined methods are the method that is already defined in the Java class libraries and is known as predefined methods. It is also known as the standard library method or built-in method. We can directly use these methods just by calling them in the program at any point.

### **2. User-defined Method**

The method written by the user or programmer is known as a user-defined method. These methods are modified according to the requirement.

### **2 Ways to Create Method in Java**

There are two ways to create a method in Java:

**1. Instance Method:** Access the instance data using the object name.Declared inside a class.

// Instance Method

**void** method\_name(){

body // instance area

}

**2. Static Method:** Access the static data using class name. Declared inside class with **static** keyword.

//Static Method

**static** **void** method\_name(){

body // static area

}

### **Method Signature**

It consists of the method name and a parameter list (number of parameters, type of the parameters, and order of the parameters). The return type and exceptions are not considered as part of it.

Method Signature of the above function:

max(int x, int y) Number of parameters is 2, Type of parameter is int.

## **How to Name a Method?**

A method name is typically a single word that should be a **verb** in lowercase or a multi-word, that begins with a **verb** in lowercase followed by an **adjective, noun…..** After the first word, the first letter of each word should be capitalized.

**Rules to Name a Method**

* While defining a method, remember that the method name must be a **verb** and start with a **lowercase** letter.
* If the method name has more than two words, the first name must be a verb followed by an adjective or noun.
* In the multi-word method name, the first letter of each word must be in **uppercase** except the first word. For example, findSum, computeMax, setX, and getX.

Generally, a method has a unique name within the class in which it is defined but sometimes a method might have the same name as other method names within the same class as method overloading is allowed in Java.