

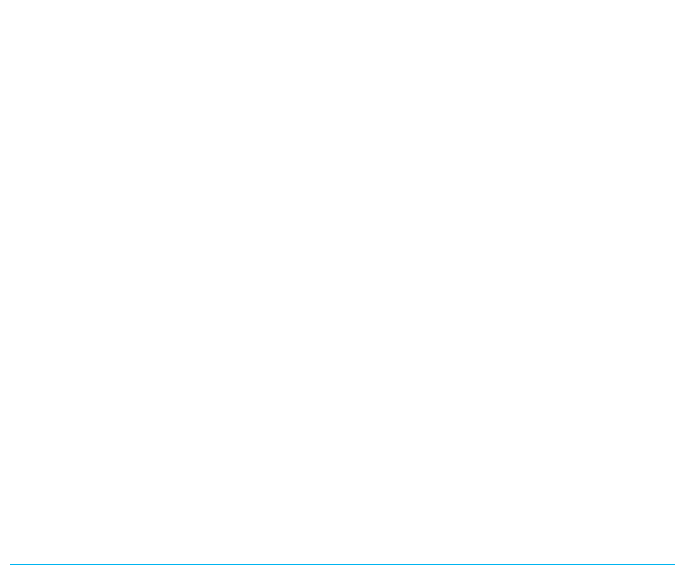
Java OOPs Concepts

In this page, we will learn about the basics of OOPs. Object-Oriented Programming is a paradigm that provides many concepts, such as **inheritance**, **data binding**, **polymorphism**, etc.

Simula is considered the first object-oriented programming language. The programming paradigm where everything is represented as an object is known as a truly object-oriented programming language.

Smalltalk is considered the first truly object-oriented programming language.





The popular object-oriented languages are Java, C#, PHP, Python, C++, etc.

The main aim of object-oriented programming is to implement real-world entities, for example, object, classes, abstraction, inheritance, polymorphism, etc.

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

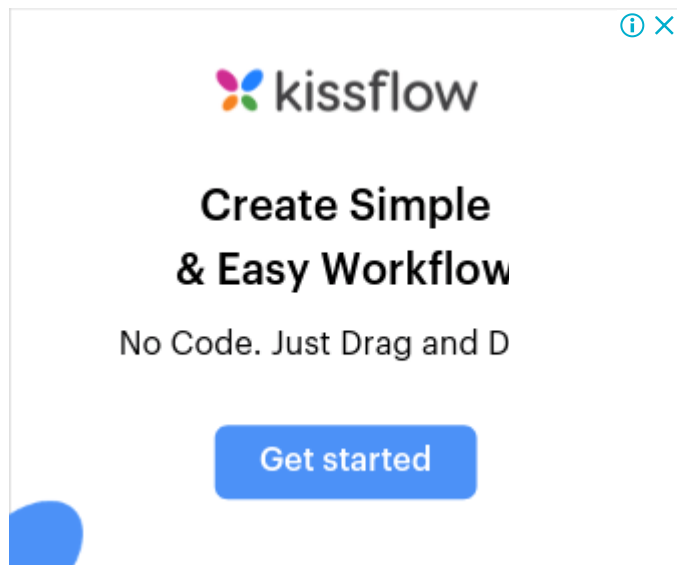
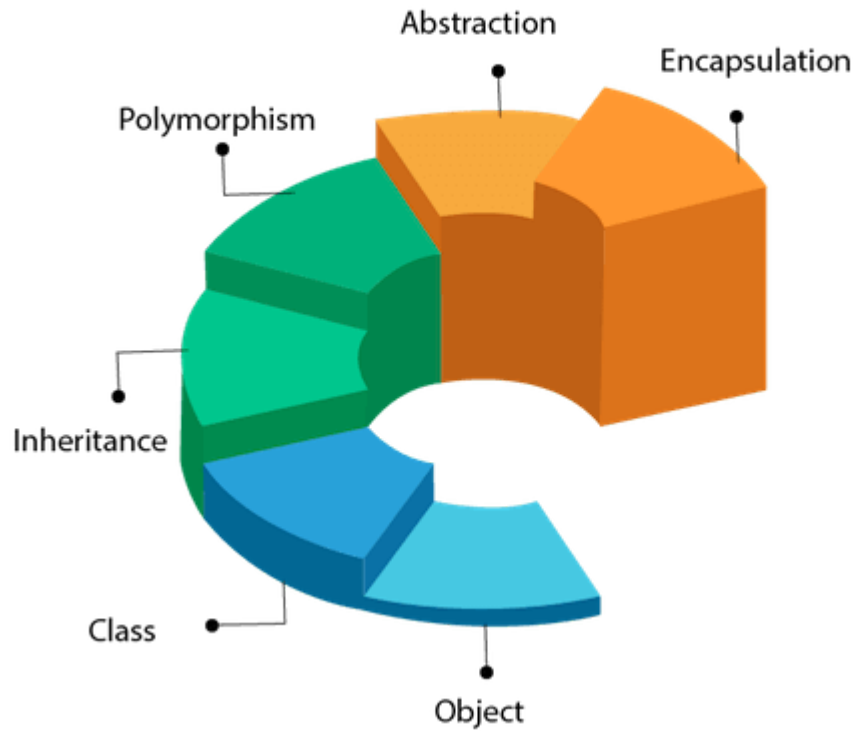


Apart from these concepts, there are some other terms which are used in Object-Oriented design:

- Coupling
- Cohesion
- Association
- Aggregation
- Composition



OOPs (Object-Oriented Programming System)



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.



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.

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.



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



Capsule

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.

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

Coupling

Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other. If a class has the details information of another class, there is strong coupling. In Java, we use private, protected, and public modifiers to display the visibility level of a class, method, and field. You can use interfaces for the weaker coupling because there is no concrete implementation.

Cohesion

Cohesion refers to the level of a component which performs a single well-defined task. A single well-defined task is done by a highly cohesive method. The weakly cohesive method will split the task into separate parts. The java.io package is a highly cohesive package because it has



I/O related classes and interface. However, the `java.util` package is a weakly cohesive package because it has unrelated classes and interfaces.

Association

Association represents the relationship between the objects. Here, one object can be associated with one object or many objects. There can be four types of association between the objects:

- One to One
- One to Many
- Many to One, and
- Many to Many

Let's understand the relationship with real-time examples. For example, One country can have one prime minister (one to one), and a prime minister can have many ministers (one to many). Also, many MP's can have one prime minister (many to one), and many ministers can have many departments (many to many).

Association can be unidirectional or bidirectional.

Aggregation

Aggregation is a way to achieve Association. Aggregation represents the relationship where one object contains other objects as a part of its state. It represents the weak relationship between objects. It is also termed as a *has-a* relationship in Java. Like, inheritance represents the *is-a* relationship. It is another way to reuse objects.

Composition

The composition is also a way to achieve Association. The composition represents the relationship where one object contains other objects as a part of its state. There is a strong relationship between the containing object and the dependent object. It is the state where



containing objects do not have an independent existence. If you delete the parent object, all the child objects will be deleted automatically.

Advantage of OOPs over Procedure-oriented programming language

- 1) OOPs makes development and maintenance easier, whereas, in a procedure-oriented programming language, it is not easy to manage if code grows as project size increases.
- 2) OOPs provides data hiding, whereas, in a procedure-oriented programming language, global data can be accessed from anywhere.

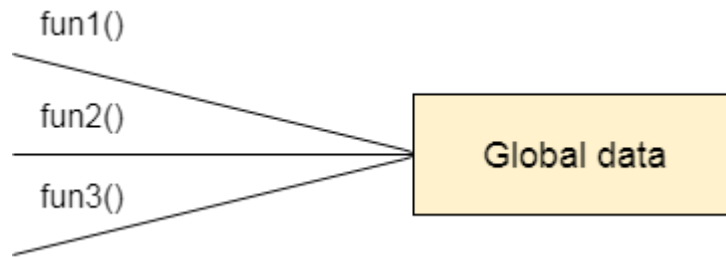


Figure: Data Representation in Procedure-Oriented Programming

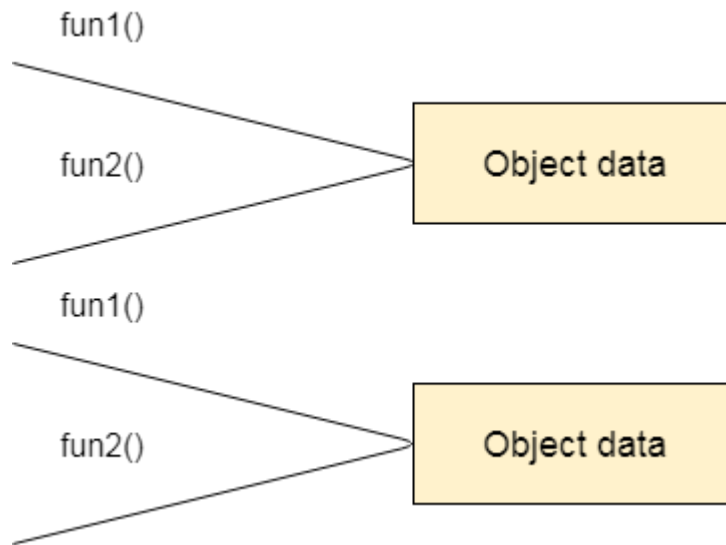


Figure: Data Representation in Object-Oriented Programming

3) OOPs provides the ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

What is the difference between an object-oriented programming language and object-based programming language?

Object-based programming language follows all the features of OOPs except Inheritance. JavaScript and VBScript are examples of object-based programming languages.

Do You Know?

- Can we overload the main method?
- A Java Constructor returns a value but, what?



- Can we create a program without main method?
- What are the six ways to use this keyword?
- Why is multiple inheritance not supported in Java?
- Why use aggregation?
- Can we override the static method?
- What is the covariant return type?
- What are the three usages of Java super keyword?
- Why use instance initializer block?
- What is the usage of a blank final variable?
- What is a marker or tagged interface?
- What is runtime polymorphism or dynamic method dispatch?
- What is the difference between static and dynamic binding?
- How downcasting is possible in Java?
- What is the purpose of a private constructor?
- What is object cloning?

What will we learn in OOPs Concepts?

- Advantage of OOPs
- Naming Convention
- Object and class
- Method overloading
- Constructor
- static keyword
- this keyword with six usage



- Inheritance
- Aggregation
- Method Overriding
- Covariant Return Type
- super keyword
- Instance Initializer block
- final keyword
- Abstract class
- Interface
- Runtime Polymorphism
- Static and Dynamic Binding
- Downcasting with instanceof operator
- Package
- Access Modifiers
- Encapsulation
- Object Cloning

[< prev](#)[next >](#)

Help Others, Please Share

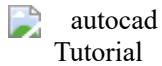




Workflow Automation at its Simplest.

[Get started](#)

Learn Latest Tutorials



autocad
Tutorial

AutoCad



kubernetes
Tutorial

Kubernetes



Openpyxl
Tutorial

Openpyxl



Tally
Tutorial

Tally



Godot
Tutorial

Godot



Spring Boot
Tutorial

Spring Boot



Gradle
Tutorial

Gradle



UML
Tutorial

UML



Artificial
Neural
Network
Tutorial

ANN



ES6 Tutorial
ES6



Flutter
Tutorial

Flutter



Selenium
Python

Selenium Py



Preparation



Aptitude

Aptitude



Logical
Reasoning

Reasoning



Verbal
Ability

Verbal A.



Interview
Questions

Interview



Company
Interview
Questions

Company

Trending Technologies



Artificial
Intelligence
Tutorial

AI



AWS
Tutorial

AWS



Selenium
tutorial

Selenium



Cloud
tutorial

Cloud



Hadoop
tutorial

Hadoop



ReactJS
Tutorial

ReactJS



Data Science
Tutorial

D. Science



Angular 7
Tutorial

Angular 7



Blockchain
Tutorial

Blockchain



Git Tutorial

Git



Machine
Learning
Tutorial



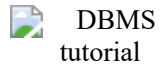
DevOps
Tutorial

DevOps



ML

B.Tech / MCA



DBMS
tutorial

DBMS



Data
Structures
tutorial

DS



DAA
tutorial

DAA



Operating
System tutorial

OS



Computer
Network
tutorial

C. Network



Compiler
Design tutorial

Compiler D.



Computer
Organization
and
Architecture

COA



Discrete
Mathematics
Tutorial

D. Math.



Ethical
Hacking
Tutorial

E. Hacking



Computer
Graphics
Tutorial

C. Graphics



Software
Engineering
Tutorial

Software E.



html tutorial
Web Tech.



Cyber
Security
tutorial

Cyber Sec.



Automata
Tutorial

Automata



C Language
tutorial

C



C++ tutorial

C++



Java tutorial
Java



.Net
Framework
tutorial



Python
tutorial

Python



List of
Programs

Programs



.Net



Control
Systems
tutorial

Control S.



Data Mining
Tutorial

Data Mining

