**Object-Oriented Programming (OOP)**

Zakariya Ibrahim AL Hadhrami

21/07/2025

**Definition of OOP :**

Object-oriented programming models a system as a group of objects, each representing a specific part of that system. These objects contain both data and methods, exposing only a public interface while keeping their internal state hidden from the rest of the system [1].

Object-oriented programming (OOP) is a programming paradigm that uses objects—data structures consisting of data fields and methods—together with their interactions to design applications and computer programs[2].

**Advantages of OOP:**

* OOP is faster and easier to execute
* OOP provides a clear structure for the programs
* OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
* Better representation of real-world entities: OOP allows developers to model real-world concepts and relationships directly in their code, making it more intuitive and easier to reason about.

**Classes and Objects in OOP:**

1. **Class:**

* Think of a class as a blueprint, template, or a user-defined data type. It defines the properties (attributes) and behaviors (methods) that objects of that class will possess. No memory is allocated when a class is defined; it's just a specification.
* Example: A Car class might have **attributes** like **color**, **make**, **model**, and **methods** like **start\_engine(), accelerate(), brake()**

1. **Object:**

* An object is an instance of a class. When you create an object, memory is allocated, and it has its own unique set of attribute values based on the class's blueprint.
* Example: **my\_car = new Car("red", "Toyota", "Camry")** creates an object **my\_car** that is an instance of the **Car class**.

**Core pillars of OOP:**

1. **Encapsulation:**

* is to make sure that "**sensitive**" data is hidden from users. To achieve this, you must:
* declare class **variables/attributes** as **private**
* provide public **get** and **set** methods to access and update the value of a private variable

1. **Inheritance:**

* Inheritance allows a new class (**subclass or child class**) to derive properties and characteristics (**attributes and methods**) from an existing class (**superclass or parent class**)
* A **SportsCar class** could inherit from the **Car class**, inheriting all its basic car functionalities and then adding specific features like **turbo\_boost().**

1. **Polymorphism:**
   * allows objects of different classes to be treated as objects of a common type. It enables a single interface to represent different underlying forms or behaviors.
   * If you have a **Shape class** with a **draw()** method, **Circle** and **Square** **classes** (inheriting from **Shape**) can each implement their own **draw()** method to draw themselves uniquely. You could then have a list of **Shape** objects and call **draw()** on each, and the correct drawing method for each specific shape would be executed.
2. **Abstraction:**
   * Abstraction focuses on showing only the essential features of an object and hiding the complex implementation details. It allows you to create a generalized representation of something without revealing its inner workings.
   * When you use a remote control to change channels on a TV, you're interacting with an abstract interface. You don't need to know the complex electronics inside the TV to change the channel.

# Reference

1. <https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Advanced_JavaScript_objects/Object-oriented_programming>
2. <https://docs.oracle.com/javase/tutorial/java/concepts/index.html>
3. <https://www.w3schools.com/java/java_oop.asp>
4. Gemini
5. <https://www.geeksforgeeks.org/dsa/introduction-of-object-oriented-programming/>