Object Oriented Programming

Why Study Object-Oriented Programming?

Modern software systems are large and complex. Writing programs as one long sequence of instructions becomes difficult to manage and maintain.
OOP helps developers:

Break problems into smaller, manageable parts

Reuse code instead of rewriting it

Model real-world systems more naturally

Most enterprise systems (banking, e-commerce, mobile money, web services) are built using OOP principles.

What is Programming?

Programming is the process of giving a computer instructions to perform a task. These instructions are written using programming languages such as Java.
Examples of problems solved using programs include:

- Calculating student grades
- Processing ATM withdrawals
- Managing user logins in applications

This course teaches how to solve such problems using an **object-oriented approach**.

Programming Paradigms

A programming paradigm is a style or way of writing programs.

Procedural Programming: Focuses on functions and steps (e.g., C)

Object-Oriented Programming: Focuses on objects and their interactions

Functional Programming: Focuses on functions and immutability

This course focuses on **Object-Oriented Programming**, which is the dominant paradigm in Java-based systems.

Functional Programming

Focus: Functions, immutability, and avoiding shared state

Common Languages

Haskell – Pure functional programming

Scala – Functional + OOP (runs on JVM)

Lisp – One of the oldest functional languages

Elixir – Scalable, concurrent systems

F# – Functional-first .NET language

JavaScript – Supports functional programming concepts

Python – Supports functional features (map, filter, lambda)

“Programs are written as combinations of functions without changing data.”

Use Case: Data processing, concurrency, distributed systems, analytics

Object-Oriented Programming (OOP)

Focus: Objects, classes, and interactions between objects

Common Languages

Java – Enterprise systems, Spring Boot, Android

C++ – High-performance systems, game engines

C# – .NET applications, Windows software

Python – OOP + scripting, data science, web apps

Kotlin – Modern OOP, Android, backend services

Ruby – Web applications (Ruby on Rails)

“Programs are built using objects that represent real-world entities.”

Use Case: Banking systems, web backends, mobile apps, enterprise software

Procedural Programming

Focus: Functions, procedures, and step-by-step execution

Common Languages

C – Low-level, fast, widely used in operating systems

Pascal – Designed for teaching structured programming

Fortran – Scientific and numerical computing

COBOL – Business and financial systems

BASIC – Beginner-friendly, early teaching language

“The program is a sequence of steps that operate on data.”

Use Case: Operating systems, embedded systems, legacy business applications

What is Object-Oriented Programming (OOP)?

Object-Oriented Programming is a programming approach where software is built using **objects**. An object represents a real-world entity and contains:

Attributes (data)

Methods (behavior)

OOP allows programmers to think about software in the same way they think about real-life systems.

Real-Life Analogy of OOP

Consider a **Car**:

Attributes: color, speed, fuel level

Methods: start(), accelerate(), brake()

In Java:

The **Car** is a class

A specific car (e.g., Toyota) is an object

Actions like starting or braking are methods.

Key Concepts of OOP

The main pillars of OOP are:

Class – blueprint of an object

Object – instance of a class

Encapsulation – hiding internal details

Inheritance – reusing and extending existing code

Polymorphism – one action behaving differently

Abstraction – focusing on what an object does, not how

These concepts will be studied in detail throughout the course.

What is Java?

Java is a:

- High-level programming language
- Object-oriented by design
- Platform-independent language

Java is widely used in:

- Enterprise applications
- Web backend systems
- Android applications

It is known for its stability, security, and scalability.

Why Java for OOP?

Java is an excellent language for learning OOP because:

- It strictly enforces object-oriented concepts
- It has clear syntax
- It is used widely in industry
- It prepares students for frameworks like **Spring Boot**

Learning OOP in Java builds strong programming fundamentals.

Java Platform Overview

Java consists of three main components:

JDK (Java Development Kit) – tools for writing and compiling Java code

JRE (Java Runtime Environment) – environment for running Java programs

JVM (Java Virtual Machine) – executes Java bytecode

How Java Programs Run

The Java program execution process:

Write source code (`.java`)

Compile using `javac` into bytecode (`.class`)

JVM executes the bytecode

This process enables Java's slogan: **Write Once, Run Anywhere**

Your First Java Program (Preview)

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Explanation:

`class` defines a blueprint

`main()` is the entry point of a Java program

`System.out.println()` outputs text to the screen