# FUNDAMENTALS OF OBJECT-ORIENTED PROGRAMMING WITH JAVA

**Course Duration: 12 Weeks**

## Course Description

This course introduces students to **Object-Oriented Programming (OOP) using Java**. It covers fundamental programming concepts, Java syntax, and core OOP principles such as encapsulation, inheritance, polymorphism, and abstraction. Students will develop problem-solving skills through hands-on coding exercises, culminating in a real-world project.

By the end of the course, students will have a solid foundation in **Java programming** and the ability to build simple **console-based and GUI-based applications** using **Java and databases**.

---

## Course Objectives

By the end of the course, students should be able to:

1. Understand Java syntax and structure.
2. Write Java programs using variables, loops, and control statements.
3. Apply Object-Oriented Programming (OOP) principles in Java applications.
4. Implement Java classes, objects, methods, and constructors.
5. Work with **Encapsulation, Inheritance, Polymorphism, and Abstraction**.
6. Use **Exception Handling and File Handling** in Java programs.
7. Develop applications using **Java Collections and Multithreading**.
8. Connect Java applications to a database using **JDBC**.
9. Develop a **Student Management System** as a capstone project.

---

## Week-by-Week Breakdown

### Week 1: Introduction to Java and Development Environment

- Overview of Java and its applications
- Setting up Java Development Kit (JDK) and Integrated Development Environment (IDE)
- Writing and running a simple Java program
- Understanding Java syntax, variables, and data types
- **Practical:** Write a simple Java program to print user details

## Week 2: Control Structures and Functions

- Conditional statements (`if-else`, `switch`)
- Looping structures (`for`, `while`, `do-while`)
- Methods (functions) and method overloading
- **Practical:** Create a program to calculate student grades based on user input

## Week 3: Introduction to Object-Oriented Programming (OOP)

- Principles of OOP: Encapsulation, Inheritance, Polymorphism, and Abstraction
- Classes and objects
- Creating and using constructors
- **Practical:** Create a `Student` class with attributes and methods for displaying student details

## Week 4: Encapsulation and Data Hiding

- Access modifiers (`private`, `public`, `protected`)
- Getters and setters
- **Practical:** Implement a `BankAccount` class with deposit and withdrawal methods

## Week 5: Inheritance and Polymorphism

- Extending classes using `extends`
- Method overriding and method overloading
- **Practical:** Implement a class hierarchy for employees (`Employee`, `Manager`, `Developer`)

## Week 6: Abstraction and Interfaces

- Abstract classes and abstract methods
- Implementing interfaces
- **Practical:** Design a system with an interface `Vehicle` and multiple concrete implementations (`Car`, `Bike`, `Bus`)

## Week 7: Exception Handling and File Handling

- Try-catch-finally block
- Custom exceptions
- Reading and writing files using Java I/O
- **Practical:** Implement a simple file-based contact manager

## Week 8: Working with Collections Framework

- Lists (`ArrayList`, `LinkedList`)
- Maps (`HashMap`, `TreeMap`)
- Sets (`HashSet`, `TreeSet`)
- **Practical:** Implement a student database system using collections

### Week 9: Multithreading and Concurrency

- Thread creation (`Thread` class and `Runnable` interface)
- Synchronization
- **Practical:** Create a program that simulates multiple users booking movie tickets

### Week 10: JDBC and Database Connectivity

- Introduction to Java Database Connectivity (JDBC)
- Connecting Java to MySQL/PostgreSQL
- CRUD operations using JDBC
- **Practical:** Build a simple student registration system with a database backend

### Week 11: Mini-Project Implementation

- Students start implementing their final projects
- Instructor review and feedback
- **Practical:** Guide students on debugging and improving their projects

### Week 12: Project Presentation and Evaluation

- Final project submission and presentations
- Peer code review and feedback
- Course recap and career guidance

---

# Final Project: Student Management System

## Project Overview

Students will develop a **Student Management System** using Java and Object-Oriented Programming principles. The system should:

- Allow student registration (Name, ID, Course, Grade)
- Implement encapsulation using private fields and public getter/setter methods
- Store student records using an `ArrayList` or database (optional for advanced students)
- Use exception handling for invalid inputs
- Implement file handling to save student records
- Use multithreading to simulate multiple users accessing the system

**Bonus Features:**

- Implement an interactive console-based menu
- Use GUI (JavaFX or Swing) for an advanced version