Week 1: Introduction to OOP Basics

Project 1: Creating a Simple Employee Management System

- **Description:** Introduce OOP principles by building an employee management system. Define an **Employee** class with attributes like name, ID, position, and salary. Implement methods to display employee details and update salary.
- **Tasks:**
  - Create the **Employee** class.
  - Implement methods to display employee details and update salary.
  - Test the system by creating employee instances and performing operations.

Week 2: Classes and Objects in Depth

Project 1: Developing a Car Dealership Inventory System

- **Description:** Explore class relationships by building a car dealership inventory system. Define classes for cars, dealerships, and customers. Include methods for adding cars to inventory, displaying available cars, and simulating customer purchases.
- **Tasks:**
  - Design classes for cars, dealerships, and customers.
  - Implement methods for managing inventory and handling customer transactions.
  - Test the system with various car additions and customer interactions.

Inheritance and Polymorphism

Project 2: Creating a Banking System with Account Types

- **Description:** Implement inheritance and polymorphism concepts by extending the bank account system. Create different types of accounts (e.g., savings, checking) that inherit from a base **BankAccount** class. Implement polymorphic methods and demonstrate their usage.
- **Tasks:**
  - Extend the **BankAccount** class to include specialized account types.
  - Implement methods specific to each account type.
  - Test polymorphic behavior by performing transactions on different account types.

Week 3: Encapsulation and Abstraction

Project1: Building a Hospital Management System

- **Description:** Focus on encapsulation and abstraction by designing a hospital management system. Create classes for patients, doctors, and medical records. Use encapsulation to protect sensitive data and abstraction to manage medical records.
- **Tasks:**
  - Define classes for patients, doctors, and medical records.
  - Encapsulate sensitive patient data and implement methods for record management.
  - Test the system by adding patients, assigning doctors, and managing medical records.

Advanced OOP Concepts

Project2: Implementing a University Course Registration System

- **Description:** Explore advanced OOP concepts like interfaces and abstract classes. Design classes for courses, students, and registration. Implement interfaces for enrollment and abstract classes for different course types.
- **Tasks:**
  - Define classes for courses, students, and registration mechanisms.
  - Implement interfaces for enrollment and abstract classes for different course types.
  - Test the system by registering students for various courses.

Week4: Real-world Application

Project: Developing a Task Management Application

- **Description:** Apply OOP concepts to a real-world scenario. Create classes for tasks, users, and task assignments. Implement functionalities for task creation, assignment, status tracking, and user management.
- **Tasks:**
  - Design classes for tasks, users, and task assignments.
  - Implement functionalities for task management and user interactions.
  - Test the application with various task scenarios and user interactions.

This 4-week program gradually introduces and reinforces Object-Oriented Programming concepts through practical projects, helping learners gain a comprehensive understanding of OOP principles and their applications in software development