Object-Oriented Programming (OOP) Practice Notes

Key Topics to Practice

1. Classes and Objects

Definition: A class is a blueprint for creating objects, and an object is an instance of a class.

Practice:

- Create a class to represent a real-world entity (e.g., a Car or a Student).
- Define properties (fields) and behaviors (methods) for the class.
- Create multiple objects and interact with their fields and methods.

2. Encapsulation

Definition: The process of wrapping data (fields) and methods into a single unit (class).

Practice:

- Create private fields and use public getter and setter methods to access them.
- Write a program where a class restricts direct access to its fields.

3. Static vs. Non-Static

Definition:

- Static: Belongs to the class and is shared by all objects.
- Non-Static: Belongs to individual objects.

Practice:

- Create a static field and method in a class and access them without creating an object.
- Define non-static fields and methods and access them through an object.

4. Inheritance

Definition: A mechanism where a class (child) inherits properties and methods from another class (parent).

Practice:

- Create a parent class with some fields and methods.
- Create a child class that extends the parent class and add additional fields or methods.
- Use super to call parent class constructors and methods.

5. Polymorphism

Definition: The ability of a method to perform different tasks based on the object that calls it.

- Compile-Time Polymorphism (Method Overloading): Same method name but different parameter lists.
- Runtime Polymorphism (Method Overriding): Child class provides a specific implementation of a method i

Practice:

- Write overloaded methods with different parameter types or counts.
- Override a parent class method in a child class and demonstrate method overriding.

6. Constructors

Definition: Special methods used to initialize objects.

- Default Constructor: No parameters.
- Parameterized Constructor: Accepts parameters to initialize fields.

Practice:

- Create a class with both default and parameterized constructors.
- Use constructors to initialize fields when creating objects.

7. Real-World Projects

Design simple applications using OOP principles, such as:

- Library Management System
- Employee Management System
- Banking System

Sample Questions

1. Classes and Objects:

- Create a class Book with fields like title, author, and price. Add a method to display book details.

2. Encapsulation:

- Create a class Account with private fields accountNumber and balance. Provide getter and setter method

3. Static and Non-Static:

- Create a static field companyName in a class Employee and access it without creating an object. Define r

4. Inheritance:

- Create a parent class Animal with a method makeSound(). Create child classes Dog and Cat that overrid

5. Polymorphism:

- Write overloaded methods calculateArea() in a class Shape to calculate the area of a circle and a rectang

- Override a method draw() in a parent class Shape in child classes Circle and Rectangle.

6. Constructors:

- Create a class Student with fields name and rollNumber. Use constructors to initialize these fields.

7. Project Example:

- Build a small application to manage a library with features like adding books, issuing books, and viewing a

Tips for Practice

- Start with simple examples to understand the concepts.

- Gradually increase the complexity of the problems.

- Focus on writing clean and modular code.

- Debug and trace your programs to understand the flow of execution.