Department of Software Engineering

Java Object-Oriented Programming Practice Problem Set

Objective: Strengthen understanding of Object-Oriented Programming (OOP) concepts in Javathrough level-wise and topic-wise problem-solving.

Level	Focus Area
Level 1	Core OOP Building Blocks (Classes, Objects)
Level 2	Encapsulation & Constructors
Level 3	Inheritance, Polymorphism, Abstraction
Level 4	Interfaces, Composition, Access Modifiers, Exception Handling

Level 1: Beginner (Core OOP Building Blocks)

Topic: Classes and Objects

Create a **class** Student with attributes: name, id, and cgpa. Write methods to display student information.

Create a Calculator **class** with methods for addition, subtraction, multiplication, and division.

Create a **class** Rectangle with length and width as attributes. Include a method to calculate and **return** the area.

Level 2: Intermediate (Encapsulation & Constructors)

Topic: Encapsulation

Create a BankAccount **class** with **private** attributes and **public** methods deposit(), withdraw(), checkBalance().

Create a Grade **class** with **private** marks for three subjects. Provide methods to input marks, calculate average, and determine grade.

Topic: Constructors

Create an Employee **class** with attributes name, id, salary and both default and parameterized constructors.

Create a Product **class** with productId, productName, and price initialized using a constructor.

Level 3: Object Relationships (Inheritance, Polymorphism, Abstraction)

Topic: Inheritance

Create a base **class** Vehicle and a derived **class** Car to demonstrate single inheritance.

Create **classes** Person -> Employee -> Manager to demonstrate multilevel inheritance.

Create Shape, Circle, and Triangle **classes** to demonstrate hierarchical inheritance.

Topic: Method Overloading & Overriding

Create overloaded methods in a **class** AreaCalculator to calculate area for different shapes.

Create Animal, Dog, and Cat **classes** where Dog and Cat override makeSound().

Topic: Abstraction

Create an **abstract class** Payment and **subclasses** CreditCardPayment and PayPalPayment implementing processPayment().

Create an **abstract class** Shape and implement concrete Rectangle and Circle **classes**.

Level 4: Advanced (Interfaces, Composition, Access Modifiers, Exception Handling, Static & Final)

Topic: Interface

Create an **interface** Drivable with methods start() and stop() implemented by Car and Bike.

Create **interfaces** Flyable and Swimmable, implement both in a **class** Duck.

Topic: Composition (Has-A Relationship)

Create **classes** Professor and Department where Department has one or more Professors.

Create a Library **class** containing a list of Book objects.

Topic: Access Modifiers

Create a **class** Student with **private** fields (name, id, cgpa). Provide **public** getters and setters. Write a separate **class** StudentTest to verify that fields cannot be accessed directly.

Create a base **class** Person with a **protected** field *nationalId*. Create a subclass Employee in a different package and show that Employee can access *nationalId* through inheritance.

Topic: Exception Handling

Write a program that handles **ArithmeticException** when dividing by zero.

Write a program that reads a text file using **BufferedReader** and handles **IOException**.

Write a program that takes an integer input as a string and handles **NumberFormatException** if the input is not a valid number.

Subtopic: throw (Manual exception throwing)

Create a **class** MarksValidator with a method validate(int marks). If marks is outside 0-100, **throw** an **IllegalArgumentException** with a clear message.

Subtopic: throws (Declaring exceptions in method signature)

Create a method readFirstLine(String path) that uses **BufferedReader** and **throws** **IOException**. Call it from main using try-catch.

Suggested Practice Flow

1. Solve **Level 1** problems first to strengthen syntax & basic class design.
2. Move to **Level 2** for understanding constructors & encapsulation.
3. Focus on **Level 3** for mastering inheritance, polymorphism, and abstraction.
4. Finally, solve **Level 4** problems to gain confidence with advanced OOP features.