Java OOP Practice Problems (Medium Level)

1. Vehicle Management System

Concepts: Inheritance, Polymorphism, Method Overriding

Design a system to manage different types of vehicles in a garage:

- `Vehicle` is the base class with attributes like `brand`, `model`, and `fuelType`.
- Subclasses: `Car`, `Bike`, and `Truck`, each with different methods like `calculateServiceCost()`.

Requirements:

- Implement a method `printServiceDetails()` that behaves differently for each vehicle.
- Use polymorphism to manage a list of vehicles and call their specific service methods.

2. Online Store Billing System

Concepts: Inheritance, Abstract Classes, Polymorphism

Create a billing system for an online store:

- Base class: `Product` with `price` and `name`.
- Subclasses: `Electronics`, `Clothing`, `Grocery`.

Requirements:

- Each subclass must override a method `calculateDiscount()` (e.g., groceries have no discount, electronics 10%, clothing 20%).
- Use an abstract method in `Product` to enforce implementation of discount calculation.

3. Educational Platform (Course Management)

Concepts: Inheritance, Runtime Polymorphism

Design a course management system where:

- Base class: `User` with subclasses `Student`, `Instructor`.
- Common method: `showDashboard()`.

Requirements:

- Show different dashboard content based on user type using runtime polymorphism.
- Implement method overriding for personalized dashboards.

4. Hotel Room Booking System

Concepts: Inheritance, Abstract Classes, Interface

Design a system to manage room bookings:

- Base class: `Room` with subclasses `DeluxeRoom`, `SuiteRoom`, and `StandardRoom`.

Requirements:

- Each room type overrides `calculatePrice(int nights)`.
- Use interface `Bookable` with method `bookRoom()` for booking logic.
- Allow polymorphic handling of different room types when booking.

5. Bank Account Management

Concepts: Inheritance, Method Overriding, Polymorphism

Design a banking system with:

- Base class: `BankAccount`
- Subclasses: `SavingsAccount`, `CurrentAccount`, `FixedDepositAccount`

Requirements:

- Override `calculateInterest()` in each subclass.
- Use a `BankSystem` class to manage accounts and process monthly interest using polymorphism.

6. School Timetable Generator

Concepts: Inheritance, Polymorphism

Design a class hierarchy for generating school timetables:

- Base class: `Employee` with `generateTimetable()`
- Subclasses: `Teacher`, `LabAssistant`, `Coach`

Requirements:

- All must override `generateTimetable()` with their specific subject/periods.
- Maintain a list of employees and call `generateTimetable()` for each using polymorphism.

7. Game Character System

Concepts: Inheritance, Method Overriding, Interface

Design a character system for a game:

- Base class: `Character`

- Subclasses: `Warrior`, `Archer`, `Mage`

Requirements:

- Each character must override `attack()` and `defend()`.

- Use interface `PlayableCharacter` with method `performSpecialMove()`.

8. Shipping System

Concepts: Inheritance, Polymorphism

Build a shipping system where:

- `Parcel` is a base class with weight and dimensions.

- Subclasses: `StandardParcel`, `ExpressParcel`, `InternationalParcel`

Requirements:

- Each parcel overrides `calculateShippingCost()`.

- Use a `ShippingService` class to process cost for a list of parcels using polymorphism.