Object-Oriented Programming (OOP) Coding Questions

Problem 1: Basic Inheritance (Vehicle Hierarchy)

Create a basic inheritance structure with a Vehicle superclass and a Car subclass. **Task** - Create a Vehicle class with the following: - A protected string variable brand. - A method honk() that prints "Tuut, tuut!". - Create a Car class that extends Vehicle: - A private string variable modelName. - A constructor that sets the brand (from the Vehicle class) and the modelName. - A main method to create a Car object and call its honk() method and display its brand and model name. **Example Output** ``` Tuut, tuut! Brand: Ford, Model: Mustang ```

Problem 2: Method Overriding (Animal Sounds)

Task - Create an Animal class with a method makeSound() that prints "The animal makes a sound". - Create a Dog class that extends Animal and override the makeSound() method to print "The dog barks". - Create a Cat class that extends Animal and override the makeSound() method to print "The cat meows". **Example Output** ``` The animal makes a sound The dog barks The cat meows ```

Problem 3: Using the super Keyword (Employee Details)

Task - Create a Person class with: - private instance variables for name (String) and age (int). - A constructor that accepts and initializes name and age. - A displayDetails() method that prints the person's name and age. - Create an Employee class that extends Person. - A private instance variable for employeeId (int). - A constructor that accepts name, age, and employeeId. It should use super() to pass the name and age to the Person constructor. - Override the displayDetails() method. This method should first call the super.displayDetails() method and then print the employee's ID. **Example Output** ``` Name: John Doe Age: 30 Employee ID: 12345 ```

Problem 4: Polymorphism (Shapes) ■

Task - Create a Shape class with a method calculateArea() that returns 0.0. - Create a Rectangle class that extends Shape with width and height properties, a constructor, and an overridden calculateArea() method. - Create a Circle class that extends Shape with radius property, a constructor, and an overridden calculateArea() method. **Main Method** - Create a Shape variable myShape. - Assign a new Rectangle object to myShape and print its area. - Assign a new Circle object to myShape and print its area. This shows how the same variable (myShape) can hold different object types and how the correct calculateArea() method is called each time.

Advanced / Frequently Asked Coding Questions

Problem 1: Abstract Classes (Payment Processing)

Task ■ - Create an abstract class named PaymentMethod with: - A private double variable for amount. - A constructor to set amount. - A concrete method getTransactionDetails() that prints the transaction amount. - An abstract boolean method processPayment(). - Create two concrete classes, CreditCardPayment and PayPalPayment, that extend PaymentMethod. - Implement processPayment() in each class with appropriate print statements. **Why it's asked:** Tests understanding of abstraction and forcing subclasses to implement specific logic.

Problem 2: Interfaces (Data Exporting)

Task ■■■ - Create an interface Exportable with method: void exportToCSV(). - Create UserReport and SalesData classes implementing Exportable. - Implement exportToCSV() in both with appropriate print statements. - In main, create a list of Exportable objects and call exportToCSV() polymorphically. **Why it's asked:** Tests understanding of interfaces for creating loosely coupled systems.

Problem 3: Composition over Inheritance (Car and Engine)

Task ■■ - Create an Engine class with start() and stop() methods. - Create a Car class that has an Engine instance (composition). - Implement startCar() and stopCar() to delegate to Engine's methods. - Demonstrate usage in main. **Why it's asked:** Tests understanding of 'Composition over Inheritance' principle, a core OOP design guideline.