Practice Questions

1. Shape Area Calculator
Create a class hierarchy for different shapes.

- Create a base class named **Shape** with a method called **calculateArea()**. This method can be empty or print a generic message.
- Create subclasses like **Circle, Rectangle, and Triangle** that extend Shape.
- Each subclass should override the **calculateArea()** method to compute and return the area specific to its shape. For example:
• Circle → π × r^2
• Rectangle → length × width
• Triangle → (1/2) × base × height
- In a **main** method, create a Shape reference and assign it objects of different subclass types. Call the **calculateArea()** method on the Shape reference to demonstrate runtime polymorphism.

2. Vehicle Class Hierarchy
Model different types of vehicles and their behaviors.

- Create a base class named **Vehicle** with a method called **getSpeed()**.
- Create subclasses such as **Car, Bicycle, and Motorcycle** that inherit from Vehicle.
- Each subclass should provide its own implementation of the **getSpeed()** method to return a different speed value.
- In the **main** method, create an array or list of Vehicle objects and add instances of Car, Bicycle, and Motorcycle. Iterate through the collection and call the **getSpeed()** method on each object to see polymorphism in action.

3. Animal Sound
This is a classic example to illustrate method overriding.

- Create a base class **Animal** with a method called **makeSound()**.
- Create two or more subclasses like **Dog, Cat, Bird**, etc., that extend Animal.
- Each subclass should override the **makeSound()** method to print the specific sound that animal makes ("Woof!", "Meow!", "Chirp!").
- In the **main** method, create objects of the subclasses and assign them to Animal reference variables. Call the **makeSound()** method on these references.

4. Employee Salary System
Design a system to calculate salaries for different types of employees.

- Create a base class **Employee** with a method **calculateSalary()**.
- Create subclasses like **Manager, Developer, and Intern**.
- Override the **calculateSalary()** method in each subclass to compute the salary based on different factors. For example:
• Manager → salary + bonus
• Developer → salary + performance bonus
• Intern → fixed stipend
- Demonstrate polymorphism by having a method that accepts an Employee object and prints its calculated salary, regardless of its specific type.

5. Bank Account Operations
Model a basic banking system with different account types.

- Create a base class **BankAccount** with a method **withdraw(double amount)**. This method can handle basic withdrawals.

- Create subclasses **SavingsAccount** and **CheckingAccount**.
- Override the **withdraw(double amount)** method in SavingsAccount to add a specific rule (e.g., a penalty for withdrawing too much, or a limit on the number of withdrawals per month).
- Override the **withdraw(double amount)** method in CheckingAccount to handle overdraft protection or a different fee structure.
- In the **main** method, create a BankAccount reference and use it to interact with both a SavingsAccount and a CheckingAccount object, showing how the same method call can lead to different behaviors.