**PRACTICE PROBLEMSON ABSTRACT CLASSES**



Create an abstract class “BankAccount” with abstract methods “deposit()” and “withdraw()”. Implement two subclasses “SavingsAccount” and “CheckingAccount” which extend “BankAccount” and implement the abstract methods. Create a “Customer” class which contains a customer of “BankAccount” objects. Add methods to the “Customer” class to display account balances, deposit/withdraw money, etc. Create objects of all classes and test their behavior.

Create an abstract class “Employee” with abstract methods “calculateSalary()” and “displayEmployeeDetails()”. Implement two subclasses “Manager” and “Worker” which extend “Employee” and implement the abstract methods. Create a “SalesPerson” class which extends “Manager” and overrides the necessary methods. Create objects of all classes and test their behavior.

Create an abstract class “Account” with abstract method “calculateInterest()”. Implement two subclasses “SavingsAccount” and “CurrentAccount” which extend “Account” and implement the “calculateInterest()” method. Create objects of both classes and test their behavior.

An abstract class called “Marks” is needed to calculate the percentage of marks earned by students A in three subjects (with each subject out of 100) and student B in four subjects (with each subject out of 100). This class must contain the abstract method “getPercentage,” which two other classes, “A” and “B,” will inherit. The method “getPercentage,” which provides the percentage of students, is shared by classes “A” and “B.”

The constructor of class ‘A’ will accept the marks obtained in three subjects as its parameters and the constructor of class ‘B’ will accept the marks obtained in four subjects as its parameters. To test the implementation, objects for both the classes need to be created and the percentage of marks for each student should be printed.

Create an abstract class “Vehicle” with abstract methods “start()” and “stop()”. Implement two subclasses “Car” and “Motorcycle” which extend “Vehicle” and implement the abstract methods. Create a “Driver” class which contains a list of “Vehicle” objects. Add methods to the “Driver” class to start and stop all vehicles. Create objects of all classes and test their behavior.

Create an abstract class “Shape” with abstract methods “getArea()” and “getPerimeter()”. Implement two subclasses “Rectangle” and “Circle” which extend “Shape” and implement the abstract methods. Create a “Square” class which extends “Rectangle” and overrides the necessary methods. Create objects of all classes and test their behavior.

**1.** Write a Java program to create an interface Shape with the getArea() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getArea() method for each of the three classes.

**2.** Write a Java program to create a Animal interface with a method called bark() that takes no arguments and returns void. Create a Dog class that implements Animal and overrides speak() to print "Dog is barking".

**3.** Write a Java program to create an interface Flyable with a method called fly\_obj(). Create three classes Spacecraft, Airplane, and Helicopter that implement the Flyable interface. Implement the fly\_obj() method for each of the three classes.

**4.** Write a Java programming to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw, calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods.

**5.** Write a Java program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.

**6.** Write a Java program to create an interface Drawable with a method draw() that takes no arguments and returns void. Create three classes Circle, Rectangle, and Triangle that implement the Drawable interface and override the draw() method to draw their respective shapes.

**7.** Write a Java program to create an interface Sortable with a method sort() that sorts an array of integers in ascending order. Create two classes BubbleSort and SelectionSort that implement the Sortable interface and provide their own implementations of the sort() method.

**8.** Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

**9.** Write a Java program to create an interface Searchable with a method search(String keyword) that searches for a given keyword in a text document. Create two classes Document and WebPage that implement the Searchable interface and provide their own implementations of the search() method.

**10.** Write a Java program to create an interface Encryptable with methods encrypt (String data) and decrypt (String encryptedData) that define encryption and decryption operations. Create two classes AES and RSA that implement the Encryptable interface and provide their own encryption and decryption algorithms.

**11.** Write a Java program to create an interface Sortable with a method sort (int[] array) that sorts an array of integers in descending order. Create two classes QuickSort and MergeSort that implement the Sortable interface and provide their own implementations of the sort() method.