1. Create a class named 'Member' having the following members:

Data members  
1 - Name  
2 - Age  
3 - Phone number  
4 - Address  
5 - Salary  
It also has a method named 'printSalary' which prints the salary of the members.  
Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

class Member {

String name;

int age;

String phoneNumber;

String address;

double salary;

void printSalary() {

System.out.println("Salary: " + salary);

}

}

class Employee extends Member {

String specialization;

Employee(String name, int age, String phoneNumber, String address, double salary, String specialization) {

super.name = name;

super.age = age;

super.phoneNumber = phoneNumber;

super.address = address;

super.salary = salary;

this.specialization = specialization;

}

}

class Manager extends Member {

String department;

Manager(String name, int age, String phoneNumber, String address, double salary, String department) {

super.name = name;

super.age = age;

super.phoneNumber = phoneNumber;

super.address = address;

super.salary = salary;

this.department = department;

}

}

public class Main {

public static void main(String[] args) {

// Create an Employee object

Employee employee = new Employee("Alice", 30, "123-456-7890", "123 Main St", 50000, "Software Engineer");

// Print employee details

System.out.println("Employee Details:");

System.out.println("Name: " + employee.name);

System.out.println("Age: " + employee.age);

System.out.println("Phone Number: " + employee.phoneNumber);

System.out.println("Address: " + employee.address);

System.out.println("Specialization: " + employee.specialization);

employee.printSalary();

// Create a Manager object

Manager manager = new Manager("Bob", 45, "987-654-3210", "456 Elm St", 80000, "IT Department");

// Print manager details

System.out.println("\nManager Details:");

System.out.println("Name: " + manager.name);

System.out.println("Age: " + manager.age);

System.out.println("Phone Number: " + manager.phoneNumber);

System.out.println("Address: " + manager.address);

System.out.println("Department: " + manager.department);

manager.printSalary();

}

}

2. You are developing a banking application in Java. Design a class hierarchy that represents different account types such as SavingsAccount, CheckingAccount, and LoanAccount.

Each account should have basic functionality like deposit, withdraw, and check balance.

Ensure that your design follows appropriate use of interfaces and inheritance.

interface Account {

void deposit(double amount);

void withdraw(double amount);

double getBalance();

}

abstract class AbstractAccount implements Account {

protected double balance;

public AbstractAccount(double initialBalance) {

this.balance = initialBalance;

}

@Override

public void deposit(double amount) {

balance += amount;

}

@Override

public void withdraw(double amount) {

balance -= amount;

}

@Override

public double getBalance() {

return balance;

}

}

class SavingsAccount extends AbstractAccount {

private final double interestRate;

public SavingsAccount(double initialBalance, double interestRate) {

super(initialBalance);

this.interestRate = interestRate;

}

// Add methods for calculating interest, etc.

}

class CheckingAccount extends AbstractAccount {

private final double overdraftFee;

public CheckingAccount(double initialBalance, double overdraftFee) {

super(initialBalance);

this.overdraftFee = overdraftFee;

}

@Override

public void withdraw(double amount) {

if (balance < amount) {

balance -= amount + overdraftFee;

} else {

super.withdraw(amount);

}

}

}

class LoanAccount extends AbstractAccount {

private final double interestRate;

private final int term;

public LoanAccount(double initialBalance, double interestRate, int term) {

super(initialBalance);

this.interestRate = interestRate;

this.term = term;

}

// Add methods for calculating loan payments, interest, etc.

}

3. You are tasked with designing a university enrollment system in Java. Implement a class hierarchy that includes a base class **Person**and two subclasses, **Student**and **Professor**and a**Course**class. Each class should have the necessary attributes. Each course should have a list of prerequisites and enrolled students.

Your tasks are as follows:

i) Students should only be enrolled if they have completed all the required prerequisites. In the course class, include logic for enrolling students.

ii) Display enrolled students in a particular with relevant information.

class Person {

protected String name;

protected String id;

public Person(String name, String id) {

this.name = name;

this.id = id;

}

}

class Student extends Person {

private List<String> completedCourses = new ArrayList<>();

public Student(String name, String id) {

super(name, id);

}

public boolean hasCompletedPrerequisites(List<String> prerequisites) {

return prerequisites.stream().allMatch(completedCourses::contains);

}

public void addCompletedCourse(String courseId) {

completedCourses.add(courseId);

}

}

class Professor extends Person {

private List<Course> assignedCourses = new ArrayList<>();

public Professor(String name, String id) {

super(name, id);

}

// Methods for assigning courses to the professor

}

class Course {

private String courseId;

private String courseName;

private Professor instructor;

private List<String> prerequisites = new ArrayList<>();

private List<Student> enrolledStudents = new ArrayList<>();

public Course(String courseId, String courseName, Professor instructor, List<String> prerequisites) {

this.courseId = courseId;

this.courseName = courseName;

this.instructor = instructor;

this.prerequisites = prerequisites;

}

public boolean enrollStudent(Student student) {

if (student.hasCompletedPrerequisites(prerequisites)) {

enrolledStudents.add(student);

return true;

} else {

return false;

}

}

public void displayEnrolledStudents() {

System.out.println("Enrolled Students in " + courseName + ":");

for (Student student : enrolledStudents) {

System.out.println("- " + student.name + " (" + student.id + ")");

}

}

}