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.

**Member.java:**

**public class Member {**

**private String name;**

**private int age;**

**private String phoneNumber;**

**private String address;**

**private double salary;**

**public Member(String name, int age, String phoneNumber, String address, double salary) {**

**this.name = name;**

**this.age = age;**

**this.phoneNumber = phoneNumber;**

**this.address = address;**

**this.salary = salary;**

**}**

**public void printSalary() {**

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

**}**

**}**

**Employee.java:**

**public class Employee extends Member {**

**private String specialization;**

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

**super(name, age, phoneNumber, address, salary);**

**this.specialization = specialization;**

**}**

**}**

**Manager.java:**

**public class Manager extends Member {**

**private String department;**

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

**super(name, age, phoneNumber, address, salary);**

**this.department = department;**

**}**

**}**

**Main.java:**

**public class Main {**

**public static void main(String[] args) {**

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

**Manager manager = new Manager("Jane Smith", 40, "987-654-3210", "456 Elm St", 80000, "Sales");**

**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();**

**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.

**Account.java (interface):**

**public interface Account {**

**double getBalance();**

**void deposit(double amount);**

**void withdraw(double amount);**

**}**

**SavingsAccount.java:**

**public class SavingsAccount implements Account {**

**private double balance;**

**private double interestRate;**

**public SavingsAccount(double initialBalance, double interestRate) {**

**this.balance = initialBalance;**

**this.interestRate = interestRate;**

**}**

**@Override**

**public double getBalance() {**

**return balance;**

**}**

**@Override**

**public void deposit(double amount) {**

**balance += amount;**

**}**

**@Override**

**public void withdraw(double amount) {**

**if (balance >= amount) {**

**balance -= amount;**

**} else {**

**System.out.println("Insufficient funds");**

**}**

**}**

**public void addInterest() {**

**balance += balance \* interestRate;**

**}**

**}**

**CheckingAccount.java:**

**public class CheckingAccount implements Account {**

**private double balance;**

**private double overdraftFee;**

**public CheckingAccount(double initialBalance, double overdraftFee) {**

**this.balance = initialBalance;**

**this.overdraftFee = overdraftFee;**

**}**

**@Override**

**public double getBalance() {**

**return balance;**

**}**

**@Override**

**public void deposit(double amount) {**

**balance += amount;**

**}**

**@Override**

**public void withdraw(double amount) {**

**if (balance >= amount) {**

**balance -= amount;**

**} else {**

**balance -= overdraftFee; // Charge overdraft fee**

**System.out.println("Overdraft fee applied");**

**}**

**}**

**}**

**LoanAccount.java:**

**public class LoanAccount implements Account {**

**private double balance;**

**private double interestRate;**

**private double monthlyPayment;**

**public LoanAccount(double initialBalance, double interestRate, double monthlyPayment) {**

**this.balance = initialBalance;**

**this.interestRate = interestRate;**

**this.monthlyPayment = monthlyPayment;**

**}**

**@Override**

**public double getBalance() {**

**return balance;**

**}**

**@Override**

**public void deposit(double amount) {**

**// Disallow deposits for loan accounts**

**System.out.println("Deposits not allowed for loan accounts");**

**}**

**@Override**

**public void withdraw(double amount) {**

**// Disallow withdrawals for loan accounts**

**System.out.println("Withdrawals not allowed for loan accounts");**

**}**

**public void makePayment() {**

**balance -= monthlyPayment;**

**}**

**public void addInterest() {**

**balance += balance \* interestRate;**

**}**

**}**

**Main.java:**

**public class Main {**

**public static void main(String[] args) {**

**SavingsAccount savings = new SavingsAccount(1000, 0.05);**

**CheckingAccount checking = new CheckingAccount(500, 25);**

**LoanAccount loan = new LoanAccount(5000, 0.1, 200);**

**System.out.println("Savings Account Balance: " + savings.getBalance());**

**System.out.println("Checking Account Balance: " + checking.getBalance());**

**System.out.println("Loan Account Balance: " + loan.getBalance());**

**savings.deposit(500);**

**checking.withdraw(400);**

**loan.makePayment();**

**System.out.println("\nAfter transactions:");**

**System.out.println("Savings Account Balance: " + savings.getBalance());**

**System.out.println("Checking Account Balance: " + checking.getBalance());**

**System.out.println("Loan Account Balance: " + loan.getBalance());**

**}**

**}**

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.

import java.util.ArrayList;

import java.util.List;

class Person {

private String name;

private String id;

public Person(String name, String id) {

this.name = name;

this.id = id;

}

// Getters for name and id

}

class Student extends Person {

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

public Student(String name, String id) {

super(name, id);

}

public void addCompletedCourse(String courseCode) {

completedCourses.add(courseCode);

}

public boolean hasCompletedPrerequisite(String prerequisite) {

return completedCourses.contains(prerequisite);

}

}

class Professor extends Person {

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

public Professor(String name, String id) {

super(name, id);

}

public void addTeachingCourse(Course course) {

teachingCourses.add(course);

}

}

class Course {

private String code;

private String name;

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

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

private Professor instructor;

public Course(String code, String name, List<String> prerequisites) {

this.code = code;

this.name = name;

this.prerequisites = prerequisites;

}

public void enrollStudent(Student student) {

if (student.hasCompletedAllPrerequisites(prerequisites)) {

enrolledStudents.add(student);

System.out.println(student.getName() + " enrolled successfully in " + code);

} else {

System.out.println(student.getName() + " not eligible due to missing prerequisites.");

}

}

public void displayEnrolledStudents() {

System.out.println("Enrolled students in " + code + ":");

for (Student student : enrolledStudents) {

System.out.println("- " + student.getName() + " (" + student.getId() + ")");

}

}

// Getters and setters for instructor

}