**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.**

**CODE :**

class Member {

    String name;

    int age;

    String phoneNumber;

    String address;

    double salary;

    public void printSalary() {

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

    }

}

class Employee extends Member {

    String specialization;

}

class Manager extends Member {

    String department;

}

public class Main {

    public static void main(String[] args) {

        Employee emp = new Employee();

[emp.name](http://emp.name/) = "John";

        emp.age = 30;

        emp.phoneNumber = "1234567890";

        emp.address = "123 Main St, City";

        emp.salary = 50000.0;

        emp.specialization = "Java Programming";

        Manager manager = new Manager();

[manager.name](http://manager.name/) = "Alice";

        manager.age = 35;

        manager.phoneNumber = "9876543210";

        manager.address = "456 Park Ave, Town";

        manager.salary = 80000.0;

        manager.department = "Human Resources";

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

        System.out.println("Name: " + [emp.name](http://emp.name/));

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

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

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

        emp.printSalary();

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

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

        System.out.println("Name: " + [manager.name](http://manager.name/));

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

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

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

        manager.printSalary();

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

    }

}

**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.**

**CODE :**

interface Account {

    void deposit(double amount);

    void withdraw(double amount);

    double checkBalance();

}

abstract class BankAccount implements Account {

    private double balance;

    public BankAccount(double balance) {

        this.balance = balance;

    }

    public void deposit(double amount) {

        balance += amount;

        System.out.println(amount + " deposited. New balance: " + balance);

    }

    public void withdraw(double amount) {

        if (balance >= amount) {

            balance -= amount;

            System.out.println(amount + " withdrawn. New balance: " + balance);

        } else {

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

        }

    }

    public double checkBalance() {

        return balance;

    }

}

class SavingsAccount extends BankAccount {

    public SavingsAccount(double balance) {

        super(balance);

    }

}

class CheckingAccount extends BankAccount {

    public CheckingAccount(double balance) {

        super(balance);

    }

}

class LoanAccount extends BankAccount {

    public LoanAccount(double balance) {

        super(balance);

    }

}

public class Main {

    public static void main(String[] args) {

        SavingsAccount savings = new SavingsAccount(1000);

        savings.deposit(500);

        savings.withdraw(200);

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

        CheckingAccount checking = new CheckingAccount(2000);

        checking.deposit(1000);

        checking.withdraw(500);

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

        LoanAccount loan = new LoanAccount(5000);

        loan.deposit(200);

        loan.withdraw(100);

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

    }

}

**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.**

**CODE :**

import java.util.ArrayList;

import java.util.List;

class Person {

    private String name;

    private int age;

    public Person(String name, int age) {

[this.name](http://this.name/) = name;

        this.age = age;

    }

    public String getName() {

        return name;

    }

    public int getAge() {

        return age;

    }

}

class Student extends Person {

    private int studentId;

    public Student(String name, int age, int studentId) {

        super(name, age);

        this.studentId = studentId;

    }

    public int getStudentId() {

        return studentId;

    }

}

class Professor extends Person {

    private String expertise;

    public Professor(String name, int age, String expertise) {

        super(name, age);

        this.expertise = expertise;

    }

    public String getExpertise() {

        return expertise;

    }

}

class Course {

    private String courseName;

    private List<Course> prerequisites;

    private List<Student> enrolledStudents;

    public Course(String courseName) {

        this.courseName = courseName;

        this.prerequisites = new ArrayList<>();

        this.enrolledStudents = new ArrayList<>();

    }

    public void addPrerequisite(Course prerequisite) {

        prerequisites.add(prerequisite);

    }

    public void enrollStudent(Student student) {

        if (hasCompletedPrerequisites(student)) {

            enrolledStudents.add(student);

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

        } else {

            System.out.println(student.getName() + " cannot be enrolled in " + courseName +

                    ". Prerequisites not completed.");

        }

    }

    private boolean hasCompletedPrerequisites(Student student) {

        for (Course prerequisite : prerequisites) {

        }

        return true;

    }

    public List<Student> getEnrolledStudents() {

        return enrolledStudents;

    }

}

public class UniversityEnrollmentSystem {

    public static void main(String[] args) {

        Student student1 = new Student("Alice", 20, 1001);

        Student student2 = new Student("Bob", 21, 1002);

        Professor professor = new Professor("Dr. Smith", 45, "Computer Science");

        Course javaCourse = new Course("Java Programming");

        Course dataStructures = new Course("Data Structures");

        javaCourse.addPrerequisite(dataStructures);

        javaCourse.enrollStudent(student1);

        javaCourse.enrollStudent(student2);

        List<Student> enrolledStudents = javaCourse.getEnrolledStudents();

        System.out.println("Enrolled Students in Java Programming:");

        for (Student student : enrolledStudents) {

            System.out.println("Name: " + student.getName() + ", ID: " + student.getStudentId());

        }

    }

}