Assignment 4

1) Problem Statement: Employee Management System Using ArrayList in Java

Objective:

Develop a Java program that manages employee records using an **ArrayList** of `Employee` objects. The system should support various operations such as adding, updating, deleting, and displaying employee details. Additionally, implement a **business logic** to determine employees eligible for promotion based on their experience.

Requirements:

1. **Create an `Employee` class** with the following attributes:
    - `id` (int)
    - `name` (String)
    - `designation` (String)
    - `salary` (double)
    - `experience` (int)
2. **Implement the following operations using an `ArrayList` of `Employee` objects:**
    - **Add Employee** – Add a new employee to the list.
    - **Update Employee** – Modify an employee's details based on their ID.
    - **Delete Employee** – Remove an employee from the list based on their ID.
    - **Search Employee** – Retrieve employee details using their ID.
    - **Display All Employees** – Print all employee records.
3. **Business Logic:**
    - Implement a method `getPromotionEligibleEmployees()` that checks all employees with experience **greater than or equal to 5 years** and prints their details as eligible for promotion.

---

2) Problem Statement: Student Management System Using HashMap in Java

Objective:

Develop a Java program that manages student records using a **HashMap** where each student's **ID** is the key, and the corresponding **Student object** is the value. The system should support various operations such as adding, updating, deleting, searching, and displaying student details. Additionally, implement a **business logic** to determine students eligible for a scholarship based on their marks.

---

1. **Create a `Student` class** with the following attributes:
    - `id` (Integer) → Unique student ID

- `name` (String) → Student's full name
- `course` (String) → Course enrolled
- `marks` (Double) → Average marks obtained

2. **Implement the following operations using a `HashMap<Integer, Student>`:**
    - **Add Student** – Add a new student to the HashMap.
    - **Update Student** – Modify a student's details using their ID.
    - **Delete Student** – Remove a student using their ID.
    - **Search Student** – Retrieve student details using their ID.
    - **Display All Students** – Print all student records stored in the HashMap.
3. **Business Logic:**
    - Implement a method `getScholarshipEligibleStudents()` that checks all students with **marks greater than or equal to 85** and prints their details as eligible for a scholarship.

Problem Statement: Employee Management System Using File Handling in Java

Objective:

Develop a Java program that manages employee records using **file handling**. The system should store employee objects in a file and support operations such as adding, updating, deleting, searching, and displaying employee details. Additionally, implement a **business logic** to determine employees eligible for a promotion based on their experience.

Requirements:

1. **Create an `Employee` class** with the following attributes:
    - `id` (Integer) → Unique employee ID
    - `name` (String) → Employee name
    - `designation` (String) → Employee role
    - `salary` (Double) → Employee salary
    - `experience` (Integer) → Years of experience
2. **Implement the following operations using file handling (`ObjectOutputStream` & `ObjectInputStream`)**:
    - **Add Employee** – Store new employee details in the file.
    - **Update Employee** – Modify employee details in the file.
    - **Delete Employee** – Remove an employee entry from the file.
    - **Search Employee** – Retrieve employee details from the file using their ID.
    - **Display All Employees** – Read and print all employee records from the file.
3. **Business Logic:**

- Implement a method `getPromotionEligibleEmployees()` that reads all employees from the file and displays those with **experience greater than or equal to 5 years** as eligible for promotion.

Problem Statement: Employee Payroll Management System with Total Compensation Calculation

Objective:

Develop a Java program that models an **Employee Payroll Management System** using **Object-Oriented Programming (OOP) principles** such as **classes and objects, constructors, inheritance (hierarchical), abstract classes, and interfaces**. The system should manage different types of employees (permanent and contract-based) and include a **business logic** to:

1. **Calculate total salary expenses (compensation) for the company.**
2. **Determine employees eligible for a bonus.**

Requirements:

1. **Create an abstract class `Employee`** with the following attributes and methods:
    - **Attributes:**
        - `empId` (Integer) → Unique Employee ID
        - `name` (String) → Employee name
        - `designation` (String) → Job role
        - `salary` (Double) → Monthly salary
    - **Constructor** to initialize the attributes.
    - **Abstract Method:** `calculateSalary()` → Must be implemented by child classes.
2. **Create two subclasses `PermanentEmployee` and `ContractEmployee`** that inherit from `Employee`:
    - **`PermanentEmployee`**: Has additional benefits such as medical allowance.
    - **`ContractEmployee`**: Receives payment based on the number of working hours.
    - **Both subclasses should override the `calculateSalary()` method.**
3. **Implement an interface `BonusEligibility`** that has a method:
    - `isEligibleForBonus()` → Determines if an employee is eligible for a bonus if `salary > 50000`.
    - Both `PermanentEmployee` and `ContractEmployee` should implement this interface.
4. **Demonstrate Hierarchical Inheritance:**
    - **Superclass `Employee`** has two **subclasses `PermanentEmployee` and `ContractEmployee`**, demonstrating hierarchical inheritance.
5. **Business Logic:**

- Implement a method `getBonusEligibleEmployees()` that filters out employees **eligible for a bonus (`salary > 50000`)** and displays them.
- Implement a method `calculateTotalCompensation()` that calculates the **total salary expenses for the company** by summing up the salaries of all employees.