Java Lab Assignment 2

Inheritance, Interfaces, and Modular Design

Problem Statement

Design and implement a **Student Management System** using inheritance, polymorphism, and interfaces. The system should consist of an abstract class Person with common fields such as name and email, and a concrete class Student that extends Person with additional fields like rollNo, course, marks, and grade. Implement an interface RecordActions with methods to **add, delete, update**, and **view** student records. Use a StudentManager class to handle the operations on student records, ensuring that duplicate roll numbers are prevented. The system should demonstrate **method overriding, method overloading**, and the use of **abstract methods**.

Objective:
Implement key object-oriented principles such as inheritance, interfaces, and abstract classes.

Learning Outcomes

Upon completion of this assignment, the student will be able to:

1. Use inheritance, method overloading, and method overriding.
2. Understand and apply abstract classes and interfaces.
3. Organize Java programs into multiple packages for modular design.
4. Work with **polymorphism** (static and dynamic).

Class Hierarchy & Data Types

Class Hierarchy:

1. **Person** (abstract class)
    - Fields: name, email
    - Method: displayInfo()
2. **Student** (extends **Person**)
    - Fields: rollNo, course, marks, grade
    - Method: displayInfo()
3. **RecordActions** (interface)

   - Methods: addStudent(), deleteStudent(), updateStudent(), searchStudent(), viewAllStudents()
4. **StudentManager** (implements **RecordActions**)
   - Methods: Implementations of CRUD operations

Data Types:

- String: for name, email, course
- int: for rollNo
- double: for marks
- List<Student>: for student storage
- Map<Integer, Student>: for student management

Detailed Instructions

1. **Create Abstract Class Person**: Define an abstract class with common fields.
2. **Create Student Class**: Implement Student by extending Person and overriding displayInfo().
3. **Interfaces and Methods**: Create the **RecordActions** interface and implement it in **StudentManager**.
4. **Method Overloading and Polymorphism**: Demonstrate method overloading in **Student** and method overriding in **StudentManager**.

Expected Output

```
Student Info:
Roll No: 101
Name: Ankit
Email: ankit@mail.com
Course: B.Tech
----------------------

Student Info:
Roll No: 102
Name: Riya
Email: riya@mail.com
Course: M.Tech
Research Area: AI
----------------------

[Note] Overloaded display method:
Student Info:
Roll No: 101
Name: Ankit
Email: ankit@mail.com
```

Course: B.Tech
This is a final method in a final class.
Finalize method called before object is garbage collected.

Guidelines to Students

1. **Abstract Classes and Inheritance**: Use inheritance to create the Student class extending Person.
2. **Interface Implementation**: Implement the RecordActions interface in **StudentManager**.
3. **Use of Packages**: Organize classes into packages (model, service).

Improvements/Adjustments

1. **Extend Interface**: Add more operations like sorting and updating records in RecordActions.
2. **More Complex Data Types**: Use **HashMap** for more efficient student management.

Submission Guidelines

1. Submit **Java code** with all necessary classes and interfaces.
2. Ensure the code is properly organized with packages.

Performance Metrics (Out of 10 Marks)

Criteria	**Marks**
Inheritance and Method Overloading	3
Interface Implementation	2
Abstract Class and Polymorphism	2
Code Organization (Packages)	2
Code Quality and Documentation	1

Dr. Manish Kumar