



K.R. Mangalam University
School of Engineering & Technology

**Fundamentals Of Java Programming Lab
(ENCA203) Assignment 2
Student Management System**

Submitted by:

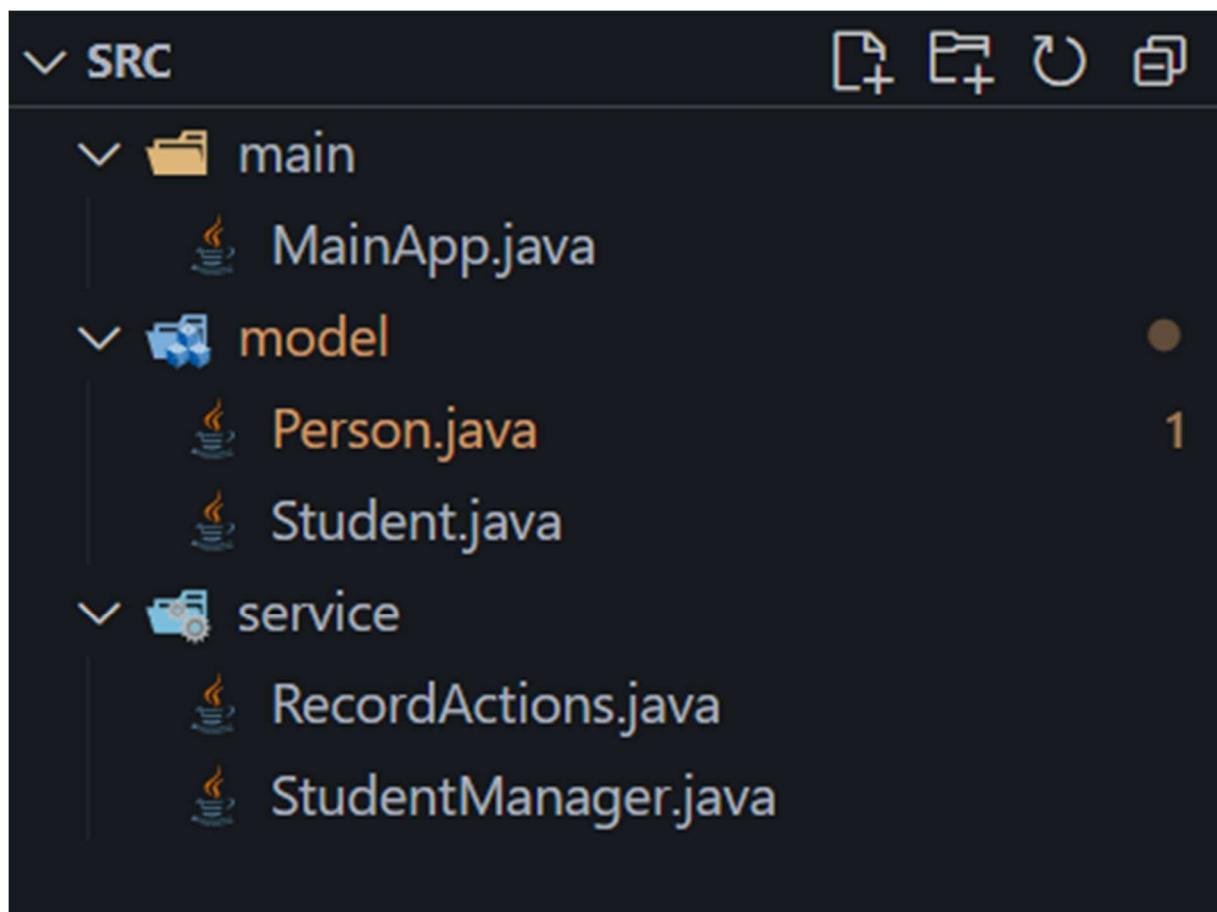
Name: RAKESH G

Roll No: 2401201064

Course: BCA (AI & DS)

Section: B

Structure of the files:



SRC/MainApp.java:

```
main > MainApp.java > ...
1 package main;
2
3 import model.Student;
4 import service.StudentManager;
5
6 import java.util.Scanner;
7
8 public class MainApp {
9     Run | Debug
10    public static void main(String[] args) {
11        Scanner sc = new Scanner(System.in);
12        StudentManager manager = new StudentManager();
13
14        int choice;
15        do {
16            System.out.println("\n===== Student Management Menu =====");
17            System.out.println("1. Add Student");
18            System.out.println("2. Delete Student");
19            System.out.println("3. Update Student");
20            System.out.println("4. Search Student");
21            System.out.println("5. View All Students");
22            System.out.println("6. Exit");
23            System.out.print("Enter choice: ");
24            choice = sc.nextInt();
25            sc.nextLine();
26
27            switch (choice) {
28                case 1:
29                    System.out.print("Enter Roll No: ");
30                    int roll = sc.nextInt();
31                    sc.nextLine();
32                    System.out.print("Enter Name: ");
33                    String name = sc.nextLine();
34                    System.out.print("Enter Email: ");
35                    String email = sc.nextLine();
36                    System.out.print("Enter Course: ");
37                    String course = sc.nextLine();
38                    System.out.print("Enter Marks: ");
39                    double marks = sc.nextDouble();
```

```
39
40         Student s = new Student(roll, name, email, course, marks);
41         manager.addStudent(s);
42         break;
43
44     case 2:
45         System.out.print(s:"Enter Roll No to delete: ");
46         int delRoll = sc.nextInt();
47         manager.deleteStudent(delRoll);
48         break;
49
50     case 3:
51         System.out.print(s:"Enter Roll No to update: ");
52         int updRoll = sc.nextInt();
53         sc.nextLine();
54         System.out.print(s:"Enter field to update (course/marks): ");
55         String field = sc.nextLine();
56         System.out.print(s:"Enter new value: ");
57         Object newVal = field.equalsIgnoreCase("marks") ? sc.nextDouble() : sc.nextLine();
58         manager.updateStudent(updRoll, field, newVal);
59         break;
60
61     case 4:
62         System.out.print(s:"Enter Roll No to search: ");
63         int searchRoll = sc.nextInt();
64         Student found = manager.searchStudent(searchRoll);
65         if (found != null) found.displayInfo(showMarks:true);
66         else System.out.println(x:"Student not found.");
67         break;
68
69     case 5:
70         manager.viewAllStudents();
71         break;
72
73     case 6:
74         System.out.println(x:"Exiting program. Goodbye!");
75         break;
```

```
76
77         default:
78             System.out.println(x:"Invalid choice.");
79         }
80     } while (choice != 6);
81
82     sc.close();
83 }
84 }
```

Code:

```
package main;

import model.Student; import
service.StudentManager;

import java.util.Scanner;

public class MainApp {    public static
void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    StudentManager manager = new StudentManager();

    int choice;
    do {
        System.out.println("\n===== Student Management Menu =====");
        System.out.println("1. Add Student");
        System.out.println("2. Delete Student");
        System.out.println("3. Update Student");
        System.out.println("4. Search Student");
        System.out.println("5. View All Students");
        System.out.println("6. Exit");

        System.out.print("Enter choice: ");
        choice = sc.nextInt();      sc.nextLine();

        switch (choice) {
            case 1:
                System.out.print("Enter Roll No: ");
```

```

int roll = sc.nextInt();

sc.nextLine();

System.out.print("Enter Name: ");

String name = sc.nextLine();

System.out.print("Enter Email: ");

String email = sc.nextLine();

System.out.print("Enter Course: ");

String course = sc.nextLine();

System.out.print("Enter Marks: ");

double marks = sc.nextDouble();

Student s = new Student(roll, name, email, course, marks);

manager.addStudent(s); break;

case 2:

System.out.print("Enter Roll No to delete: ");

int delRoll = sc.nextInt();

manager.deleteStudent(delRoll);

break;

case 3:

System.out.print("Enter Roll No to update: ");

int updRoll = sc.nextInt(); sc.nextLine();

System.out.print("Enter field to update (course/marks): ");

String field = sc.nextLine();

System.out.print("Enter new value: ");

Object newVal = field.equalsIgnoreCase("marks") ? sc.nextDouble() : sc.nextLine();

manager.updateStudent(updRoll, field, newVal);

break;

```

```
case  
4:  
    System.out.print("Enter Roll No to search: ");  
    int searchRoll = sc.nextInt();  
    Student found = manager.searchStudent(searchRoll);  
    if (found != null) found.displayInfo(true);           else  
        System.out.println("Student not found.");  
    break;  
  
case  
5:  
    manager.viewAllStudents();  
    break;  
  
case  
6:  
    System.out.println("Exiting program. Goodbye!");  
    break;  
  
default:  
    System.out.println("Invalid choice.");  
}  
} while (choice != 6);  
  
sc.close();  
}  
}
```

SRC/model/Person.java:

```
model > Person.java > ...
1 package model;
2
3 public abstract class Person {
4     protected String name;
5     protected String email;
6
7     public Person(String name, String email) {
8         this.name = name;
9         this.email = email;
10    }
11
12    // Abstract method
13    public abstract void displayInfo();
14
15    // Final method (cannot be overridden)
16    public final void finalMethodExample() {
17        System.out.println("This is a final method in Person class.");
18    }
19
20    // finalize example
21    @Override
22    protected void finalize() throws Throwable {
23        System.out.println("Finalize method called before object is garbage collected.");
24        super.finalize();
25    }
26 }
```

Code:

```
package model;
```

```
public abstract class Person {  
    protected String name;    protected  
    String email;  
  
    public Person(String name, String email) {  
        this.name = name;      this.email = email;  
    }  
  
    // Abstract method    public  
    abstract void displayInfo();  
  
    // Final method (cannot be overridden)  
    public final void finalMethodExample() {  
        System.out.println("This is a final method in Person class.");  
    }  
  
    // finalize example    @Override  
    protected void finalize() throws Throwable {  
        System.out.println("Finalize method called before object is garbage collected.");  
        super.finalize();  
    }  
}
```

SRC/model/Student.java:

```
model > Student.java > ...
1 package model;
2
3 public class Student extends Person {
4     private int rollNo;
5     private String course;
6     private double marks;
7     private char grade;
8
9     public Student(int rollNo, String name, String email, String course, double marks) {
10        super(name, email);
11        this.rollNo = rollNo;
12        this.course = course;
13        this.marks = marks;
14        calculateGrade();
15    }
16
17    // Overloaded constructor (without marks)
18    public Student(int rollNo, String name, String email, String course) {
19        this(rollNo, name, email, course, marks:0.0);
20    }
21
22    // Method overloading for display
23    public void displayInfo(boolean showMarks) {
24        displayInfo();
25        if (showMarks) {
26            System.out.println("Marks: " + marks);
27            System.out.println("Grade: " + grade);
28        }
29    }
30
31    @Override
32    public void displayInfo() {
33        System.out.println("Roll No: " + rollNo);
34        System.out.println("Name: " + name);
35        System.out.println("Email: " + email);
36        System.out.println("Course: " + course);
37    }
38
39    private void calculateGrade() {
```

```
39     private void calculateGrade() {
40         if (marks >= 90) grade = 'A';
41         else if (marks >= 75) grade = 'B';
42         else if (marks >= 50) grade = 'C';
43         else grade = 'D';
44     }
45
46     // Getters and setters
47     public int getRollNo() { return rollNo; }
48     public void setCourse(String course) { this.course = course; }
49     public void setMarks(double marks) { this.marks = marks; calculateGrade(); }
50 }
```

Code:

```
package model;
```

```
public class Student extends Person {  
    private int rollNo;    private String  
    course;    private double marks;  
    private char grade;
```

```
    public Student(int rollNo, String name, String email, String course, double marks) {  
        super(name, email);    this.rollNo = rollNo;    this.course = course;  
        this.marks = marks;    calculateGrade();  
    }
```

```
// Overloaded constructor (without marks)    public Student(int  
rollNo, String name, String email, String course) {    this(rollNo,  
name, email, course, 0.0);  
}
```

```
// Method overloading for display    public  
void displayInfo(boolean showMarks) {  
    displayInfo();  
    if (showMarks) {  
        System.out.println("Marks: " + marks);  
        System.out.println("Grade: " + grade);  
    }  
}
```

```
@Override public void  
displayInfo() {  
    System.out.println("Roll No: " + rollNo);  
    System.out.println("Name: " + name);  
    System.out.println("Email: " + email);  
    System.out.println("Course: " + course);  
}  
  
private void calculateGrade() {  
    if (marks >= 90) grade = 'A';  
    else if (marks >= 75) grade = 'B';  
    else if (marks >= 50) grade = 'C';  
    else grade = 'D';  
}  
  
// Getters and setters public int getRollNo() { return rollNo; } public void  
setCourse(String course) { this.course = course; } public void  
setMarks(double marks) { this.marks = marks; calculateGrade(); } }
```

SRC/service/RecordActions.java:

```
service > RecordActions.java > ...
1 package service;
2
3 import model.Student;
4
5 public interface RecordActions {
6     void addStudent(Student s);
7     void deleteStudent(int rollNo);
8     void updateStudent(int rollNo, String field, Object newValue);
9     Student searchStudent(int rollNo);
10    void viewAllStudents();
11 }
```

Code:

```
package service;
```

```
import model.Student;
```

```
public interface RecordActions {    void addStudent(Student s);
void deleteStudent(int rollNo);    void updateStudent(int rollNo,
String field, Object newValue);    Student searchStudent(int
rollNo);    void viewAllStudents();
}
```

SRC/service/StudentManager.java:

```
service > StudentManager.java > ...
1 package service;
2
3 import model.Student;
4 import java.util.HashMap;
5
6 public class StudentManager implements RecordActions {
7     private HashMap<Integer, Student> students = new HashMap<>();
8
9     @Override
10    public void addStudent(Student s) {
11        if (students.containsKey(s.getRollNo())) {
12            System.out.println("Error: Student with this roll number already exists!");
13        } else {
14            students.put(s.getRollNo(), s);
15            System.out.println("Student added successfully.");
16        }
17    }
18
19    @Override
20    public void deleteStudent(int rollNo) {
21        if (students.remove(rollNo) != null) {
22            System.out.println("Student removed successfully.");
23        } else {
24            System.out.println("No student found with this roll number.");
25        }
26    }
27
28    @Override
29    public void updateStudent(int rollNo, String field, Object newValue) {
30        Student s = students.get(rollNo);
31        if (s == null) {
32            System.out.println("No student found with this roll number.");
33            return;
34        }
35
36        switch (field.toLowerCase()) {
37            case "course":
38                s.setCourse((String)newValue);
39                break;
```

```
40     case "marks":  
41         s.setMarks((Double)newValue);  
42         break;  
43     default:  
44         System.out.println(x:"Invalid field.");  
45         return;  
46     }  
47     System.out.println(x:"Student record updated.");  
48 }  
49  
50 @Override  
51 public Student searchStudent(int rollNo) {  
52     return students.get(rollNo);  
53 }  
54  
55 @Override  
56 public void viewAllStudents() {  
57     if (students.isEmpty()) {  
58         System.out.println(x:"No student records available.");  
59     } else {  
60         for (Student s : students.values()) {  
61             s.displayInfo(showMarks:true);  
62             System.out.println(x:"-----");  
63         }  
64     }  
65 }  
66 }
```

Code:

```
package service;
```

```
import model.Student; import
```

```
java.util.HashMap;
```

```
public class StudentManager implements RecordActions {    private  
HashMap<Integer, Student> students = new HashMap<>();
```

```
    @Override    public void  
addStudent(Student s) {        if  
(students.containsKey(s.getRollNo())) {  
            System.out.println("Error: Student with this roll number already exists!");  
        } else {  
            students.put(s.getRollNo(), s);  
            System.out.println("Student added successfully.");  
        }  
    }
```

```
    @Override    public void  
deleteStudent(int rollNo) {        if  
(students.remove(rollNo) != null) {  
            System.out.println("Student removed successfully.");  
        } else {  
            System.out.println("No student found with this roll number.");  
        }  
    }
```

```
    }

}

@Override    public void updateStudent(int rollNo, String field,
Object newValue) {      Student s = students.get(rollNo);      if (s ==
null) {
    System.out.println("No student found with this roll number.");
    return;
}

switch (field.toLowerCase()) {
case "course":
    s.setCourse((String)newValue);
    break;
case "marks":
    s.setMarks((Double)newValue);
    break;
default:
    System.out.println("Invalid field.");
    return;
}

System.out.println("Student record updated.");
}
```

```
@Override public Student  
searchStudent(int rollNo) {      return  
students.get(rollNo);  
  
}  
  
  
@Override public void  
viewAllStudents() {      if  
(students.isEmpty()) {  
    System.out.println("No student records available.");  
} else {      for (Student s :  
students.values()) {  
    s.displayInfo(true);  
    System.out.println("-----");  
}  
}  
}
```

Output:

```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 1
Enter Roll No: 1
Enter Name: gameonas
Enter Email: gameonas@gmail.com
Enter Course: bca
Enter Marks: 99
Student added successfully.
```

```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 2
Enter Roll No to delete: 2
Student removed successfully.
```

```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 3
Enter Roll No to update: 1
Enter field to update (course/marks): course
Enter new value: btech
Student record updated.
```

```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 4
Enter Roll No to search: 1
Roll No: 1
Name: gameonas
Email: gameonas@gmail.com
Course: btech
Marks: 99.0
Grade: A
```

===== Student Management Menu =====

1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit

Enter choice: 5

Roll No: 1

Name: gameonas

Email: gameonas@gmail.com

Course: btech

Marks: 99.0

Grade: A

```
===== Student Management Menu =====
```

- 1. Add Student
- 2. Delete Student
- 3. Update Student
- 4. Search Student
- 5. View All Students
- 6. Exit

```
Enter choice: 6
```

```
Email: gameonas@gmail.com
```

```
Course: btech
```

```
Marks: 99.0
```

```
Grade: A
```

```
===== Student Management Menu =====
```

- 1. Add Student
- 2. Delete Student
- 3. Update Student
- 4. Search Student
- 5. View All Students
- 6. Exit

```
Enter choice: 6
```

Explanation of the Code

This program implements a **Student Management System** using key **OOP principles** — inheritance, abstraction, interfaces, and polymorphism.

An **abstract class Person** holds common fields like name and email. The **Student class** extends it and adds specific fields such as rollNo, course, marks, and grade. It also **overrides** the displayInfo() method and **overloads** it to show compile-time polymorphism.

An **interface RecordActions** defines operations like addStudent(), deleteStudent(), updateStudent(), searchStudent(), and viewAllStudents().

The **StudentManager class** implements this interface and manages student data using a **HashMap** to prevent duplicate roll numbers.

The code demonstrates:

- **Inheritance** (Student extends Person)
- **Abstraction & Interfaces** (through Person and RecordActions)
- **Polymorphism** (method overloading and overriding)
- **Encapsulation** (data handled through methods)
- **Modular Design** (organized using packages like model and service)

Overall, the program provides a structured, reusable, and object-oriented approach to managing student records efficiently.