K.R. Mangalam University

School of Engineering & Technology

Fundamentals Of Java Programming Lab

(ENCA203) Assignment 2

Student Management System

Submitted by:

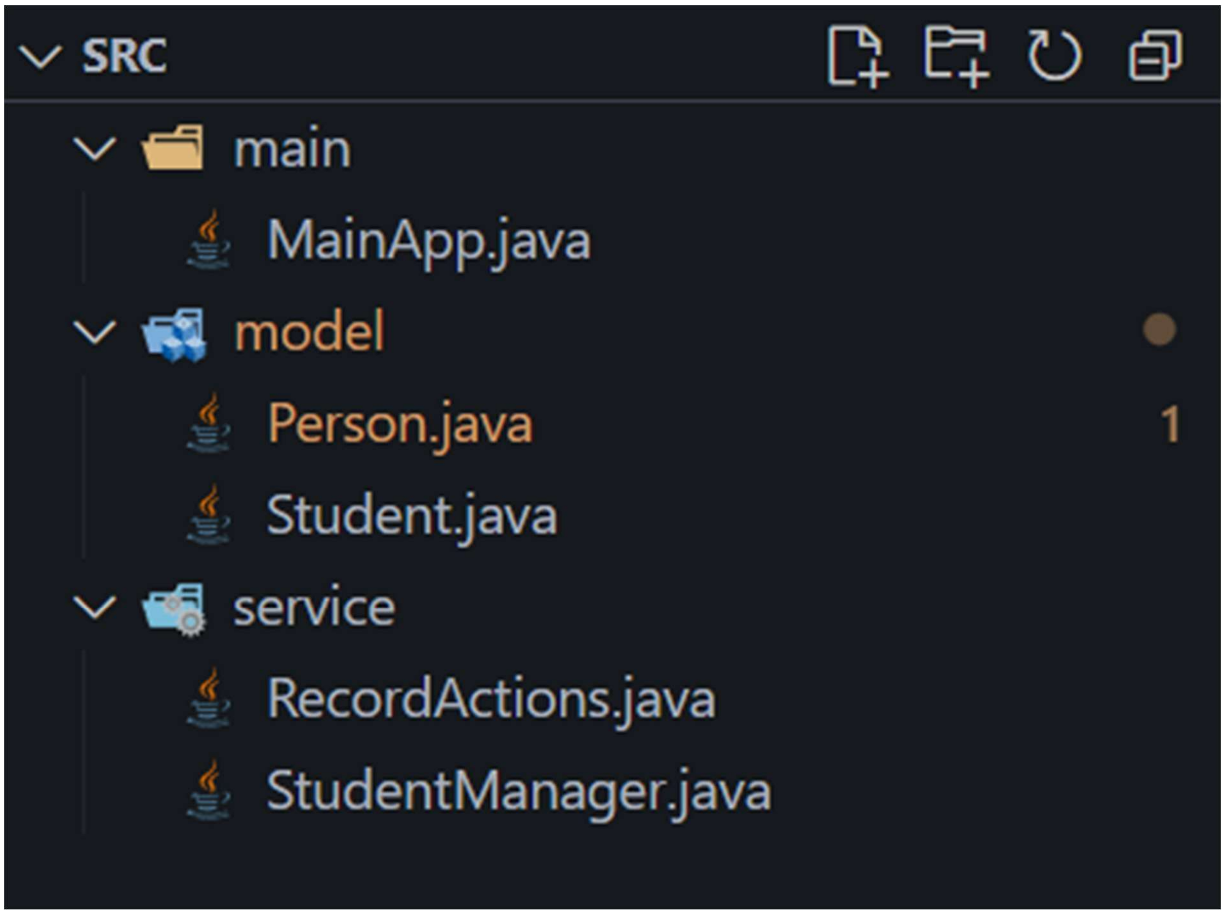
Name: RAKESH G

Roll No: 2401201064

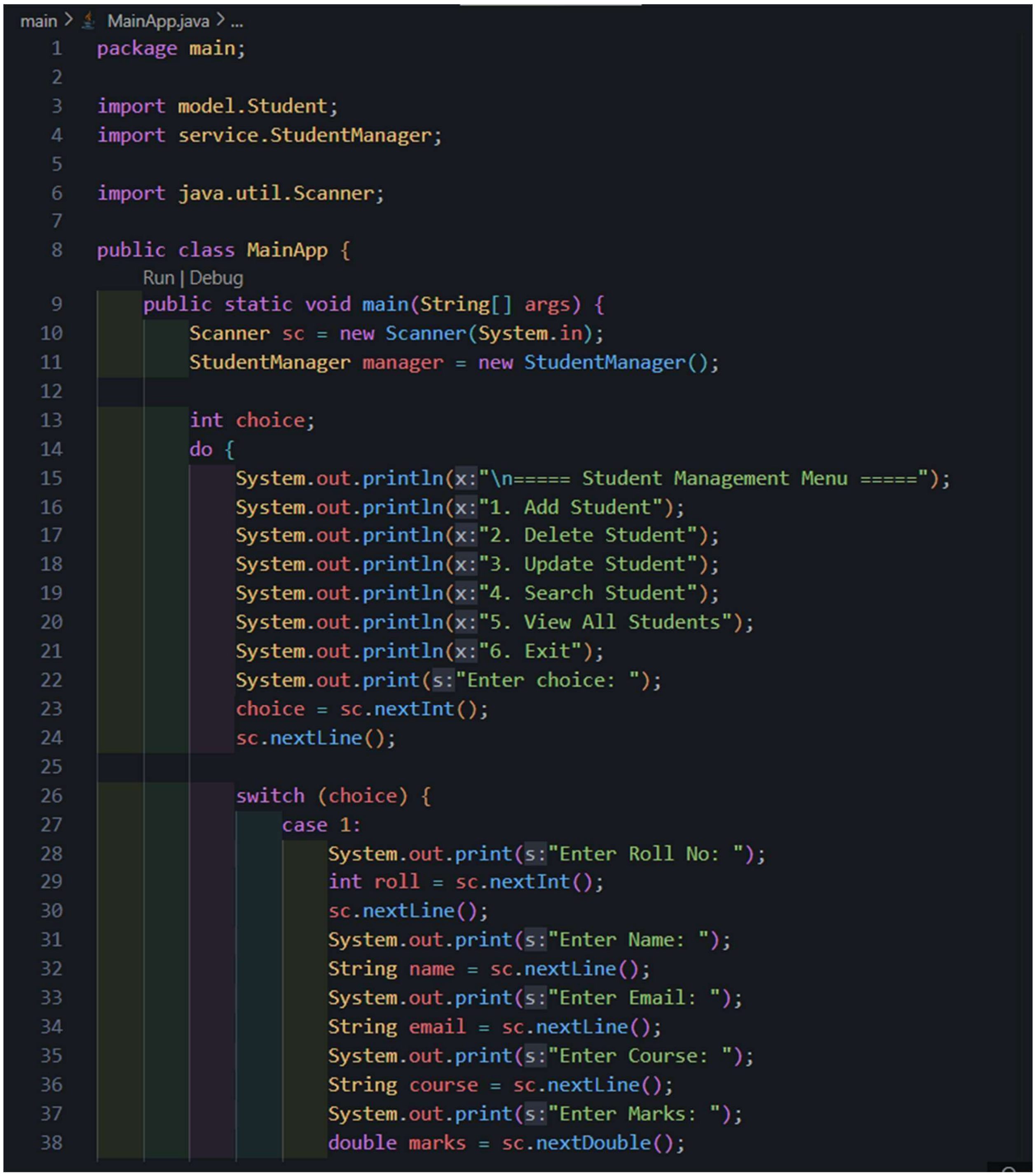
Course: BCA (AI & DS)

Section: B

Structure of the files:

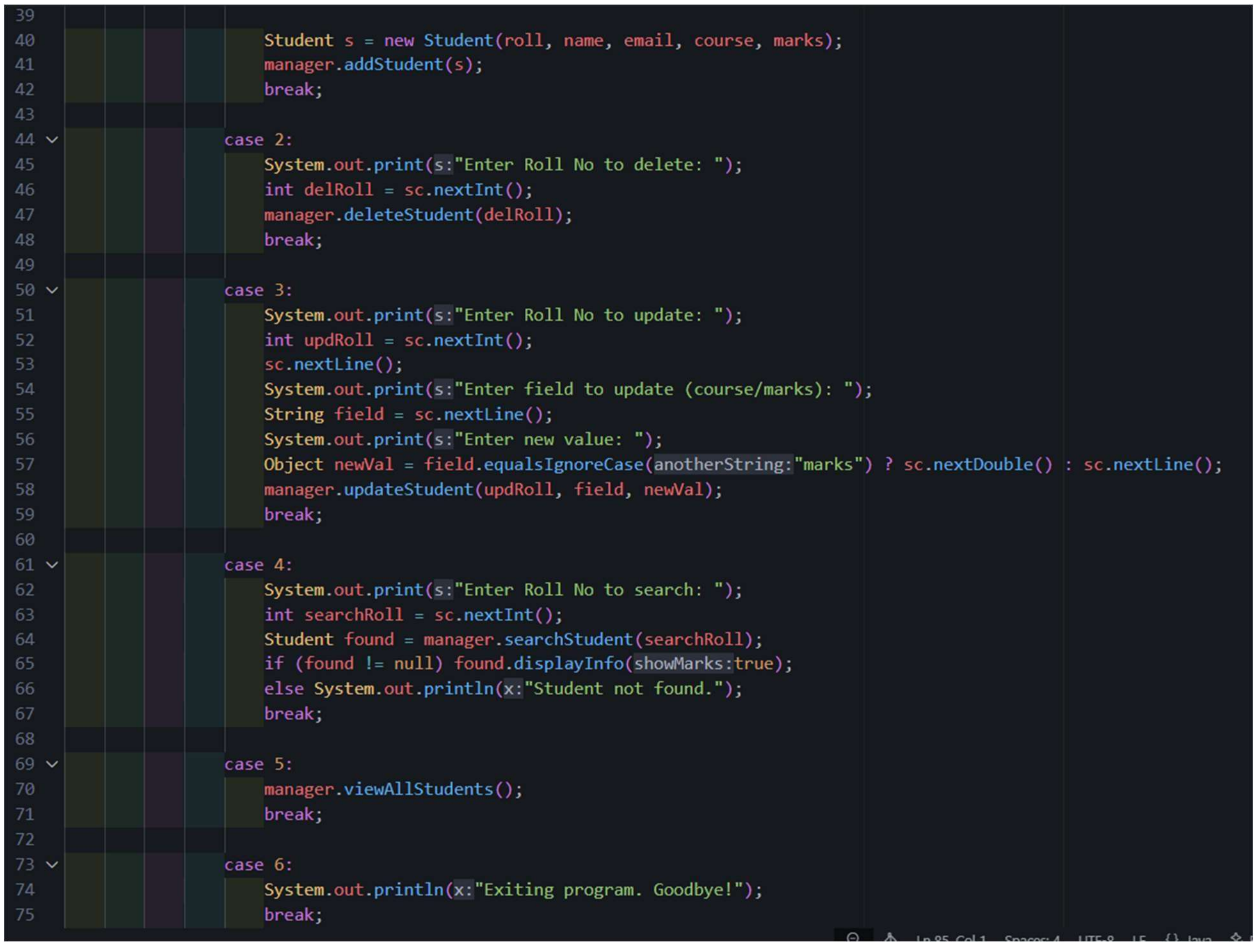

```
SRC
  main
    MainApp.java
  model
    Person.java        1
    Student.java
  service
    RecordActions.java
    StudentManager.java
```

SRC/MainApp.java:

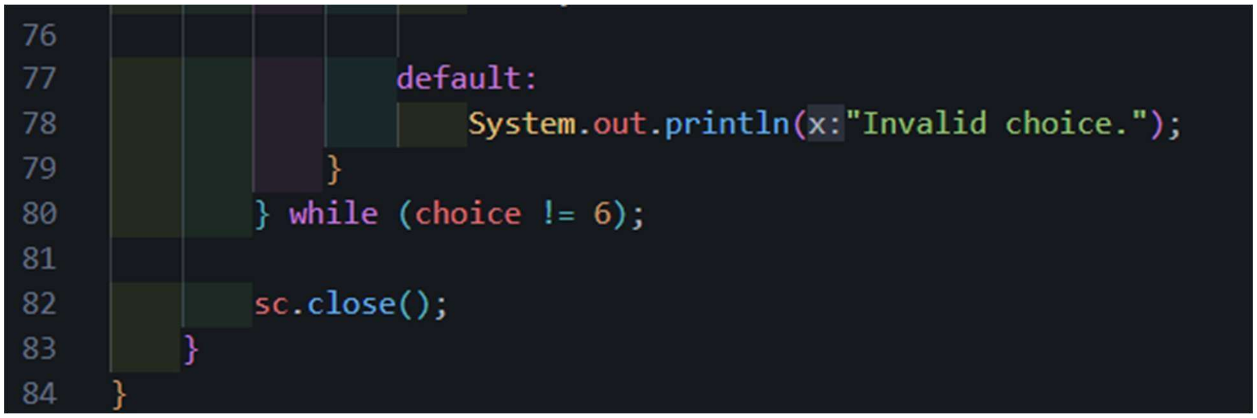


main > MainApp.java > ...

```java
package main;

import model.Student;
import service.StudentManager;

import java.util.Scanner;

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

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

            switch (choice) {
                case 1:
                    System.out.print(s:"Enter Roll No: ");
                    int roll = sc.nextInt();
                    sc.nextLine();
                    System.out.print(s:"Enter Name: ");
                    String name = sc.nextLine();
                    System.out.print(s:"Enter Email: ");
                    String email = sc.nextLine();
                    System.out.print(s:"Enter Course: ");
                    String course = sc.nextLine();
                    System.out.print(s:"Enter Marks: ");
                    double marks = sc.nextDouble();
```



```java

                    Student s = new Student(roll, name, email, course, marks);
                    manager.addStudent(s);
                    break;

                case 2:
                    System.out.print(s:"Enter Roll No to delete: ");
                    int delRoll = sc.nextInt();
                    manager.deleteStudent(delRoll);
                    break;

                case 3:
                    System.out.print(s:"Enter Roll No to update: ");
                    int updRoll = sc.nextInt();
                    sc.nextLine();
                    System.out.print(s:"Enter field to update (course/marks): ");
                    String field = sc.nextLine();
                    System.out.print(s:"Enter new value: ");
                    Object newVal = field.equalsIgnoreCase(anotherString:"marks") ? sc.nextDouble() : sc.nextLine();
                    manager.updateStudent(updRoll, field, newVal);
                    break;

                case 4:
                    System.out.print(s:"Enter Roll No to search: ");
                    int searchRoll = sc.nextInt();
                    Student found = manager.searchStudent(searchRoll);
                    if (found != null) found.displayInfo(showMarks:true);
                    else System.out.println(x:"Student not found.");
                    break;

                case 5:
                    manager.viewAllStudents();
                    break;

                case 6:
                    System.out.println(x:"Exiting program. Goodbye!");
                    break;
```



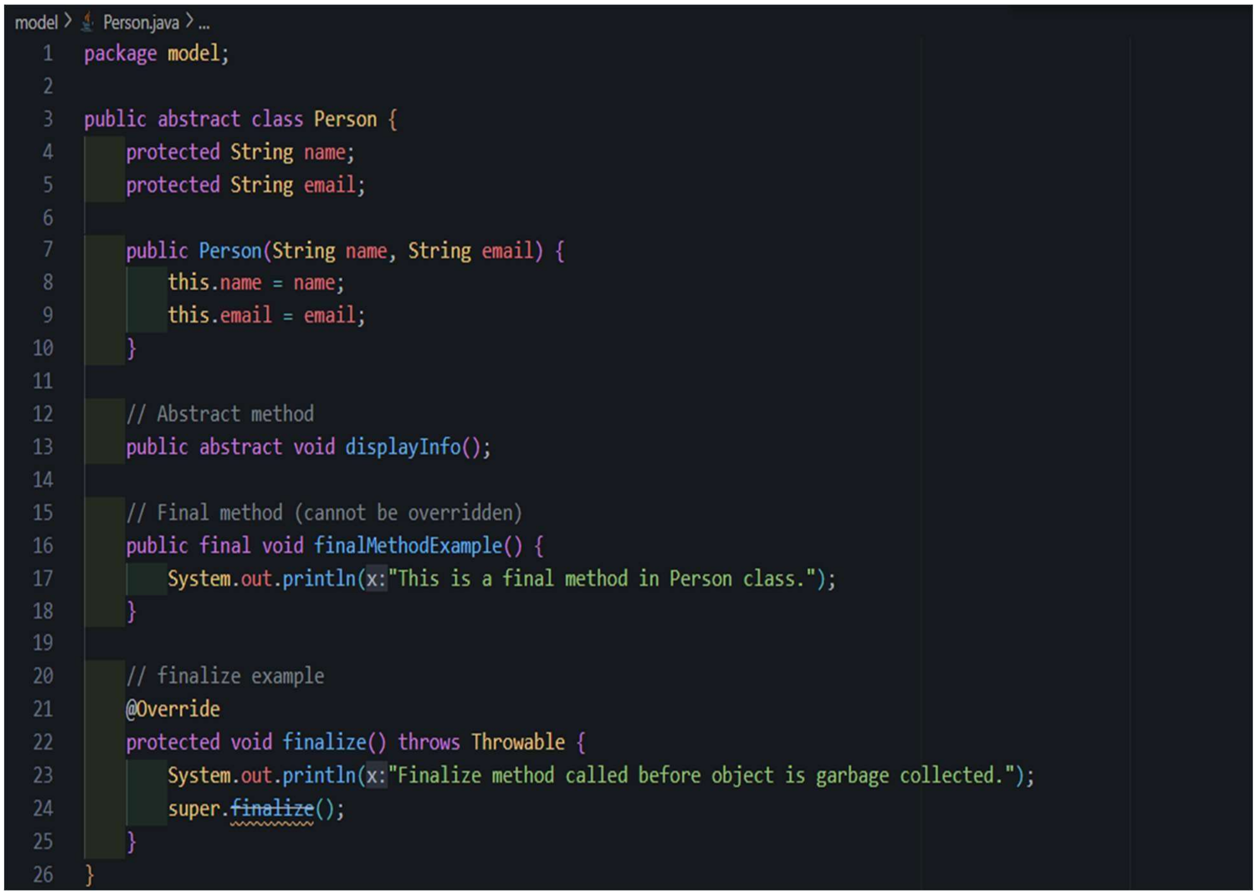
```java

                default:
                    System.out.println(x:"Invalid choice.");
            }
        } while (choice != 6);

        sc.close();
    }
}
```

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:



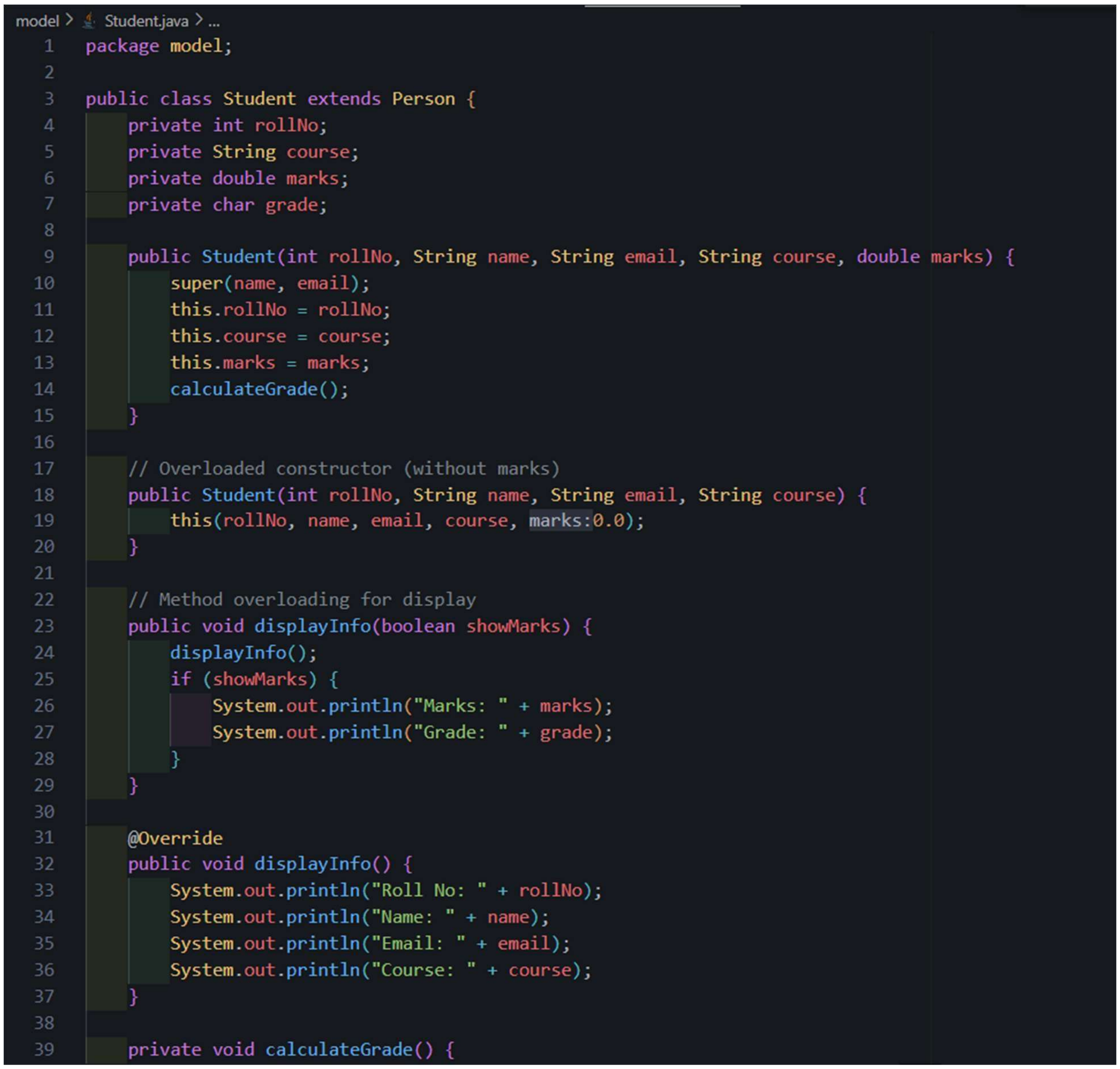
model > Person.java > ...

```java
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(x:"This is a final method in Person class.");
    }

    // finalize example
    @Override
    protected void finalize() throws Throwable {
        System.out.println(x:"Finalize method called before object is garbage collected.");
        super.~~finalize~~();
    }
}
```

Code:

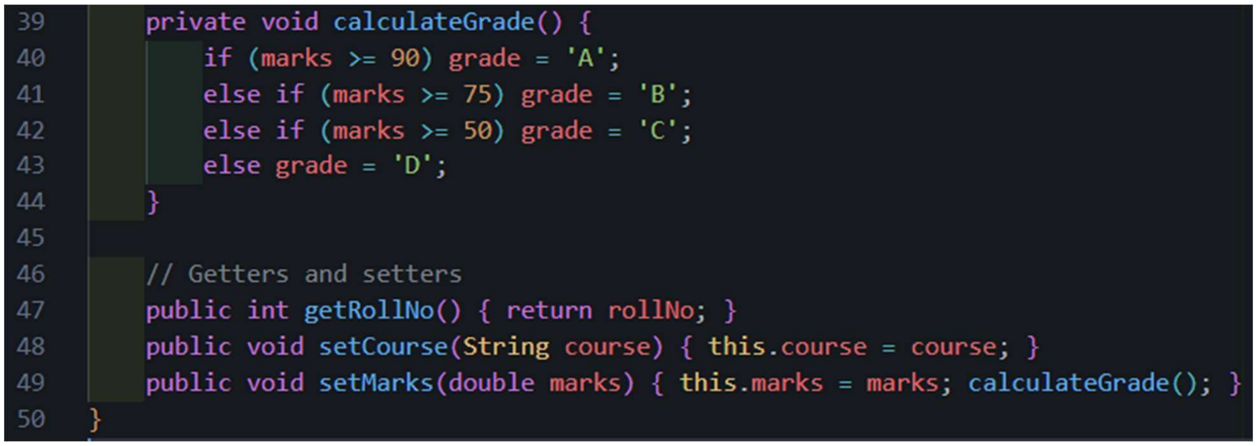
```
package model;


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 > ...

```java
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, marks: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() {
```

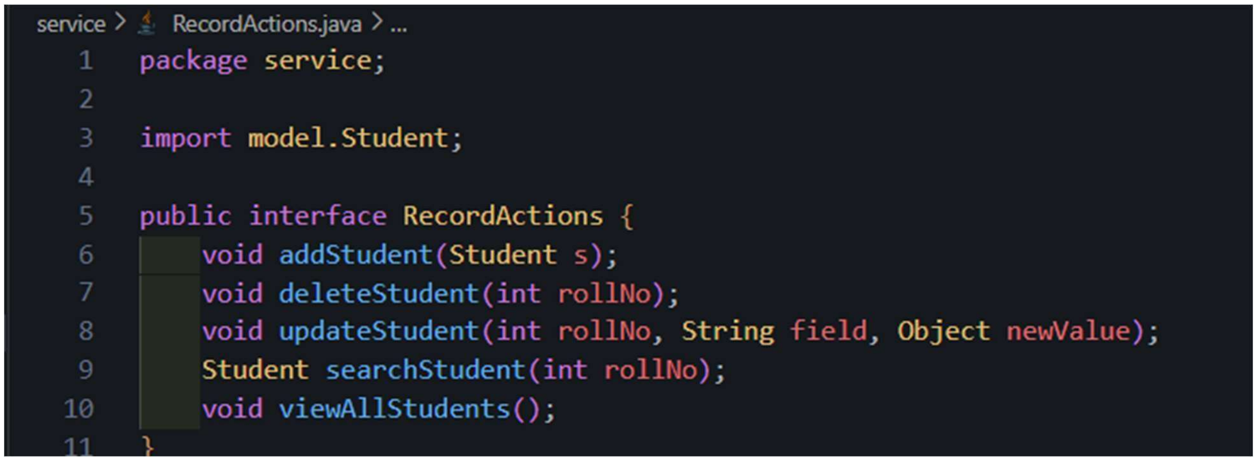


```java
    private void calculateGrade() {
        if (marks >= 90) grade = 'A';
        else if (marks >= 75) grade = 'B';
        else if (marks >= 50) grade = 'C';
        else grade = 'D';
    }

    // 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(); }
}
```

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 > ...

```java
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();
}
```

Code:

package service;

import model.Student;

public interface RecordActions {   void addStudent(Student s);

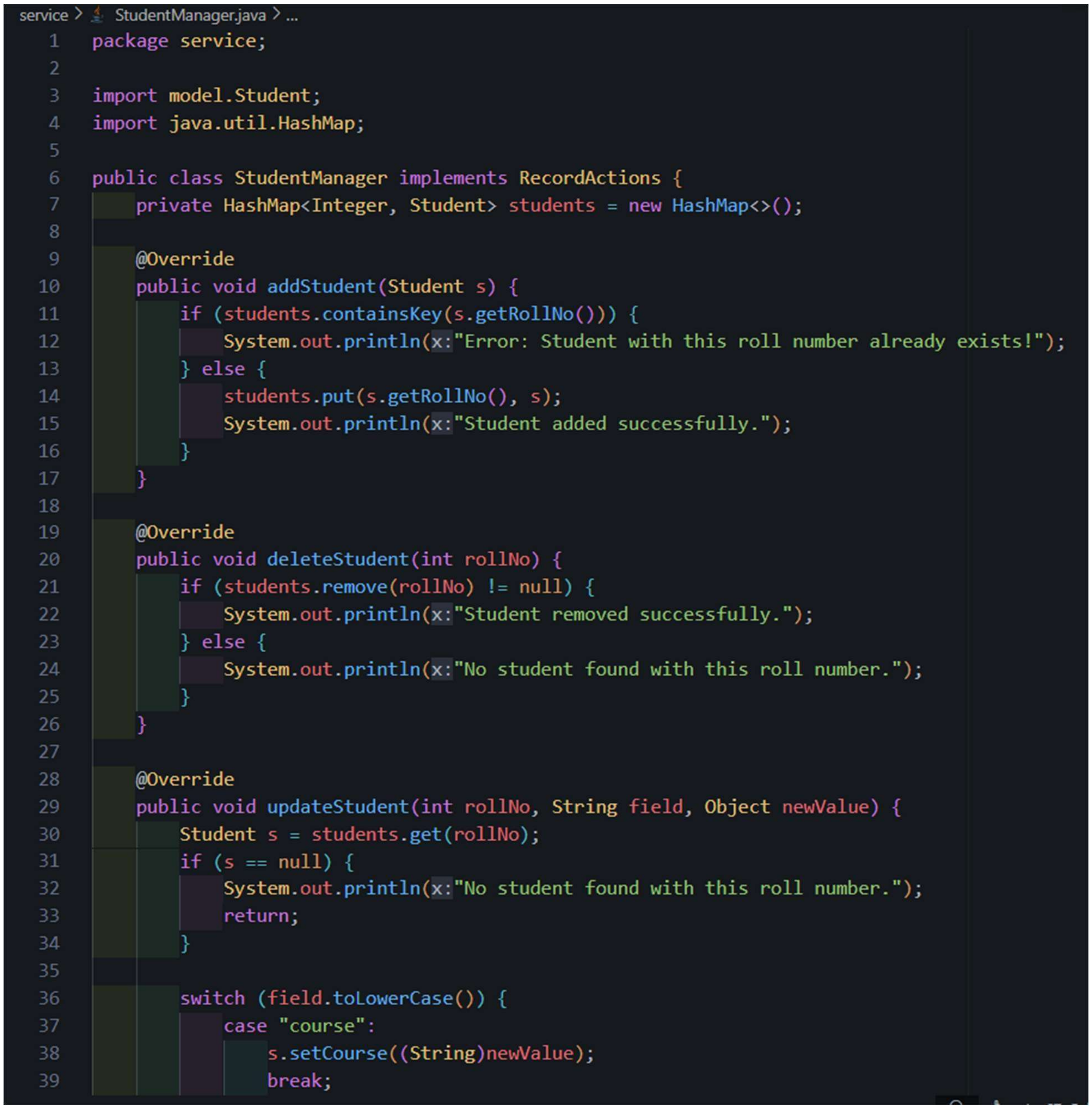
void deleteStudent(int rollNo);   void updateStudent(int rollNo,

String field, Object newValue);   Student searchStudent(int

rollNo);   void viewAllStudents();

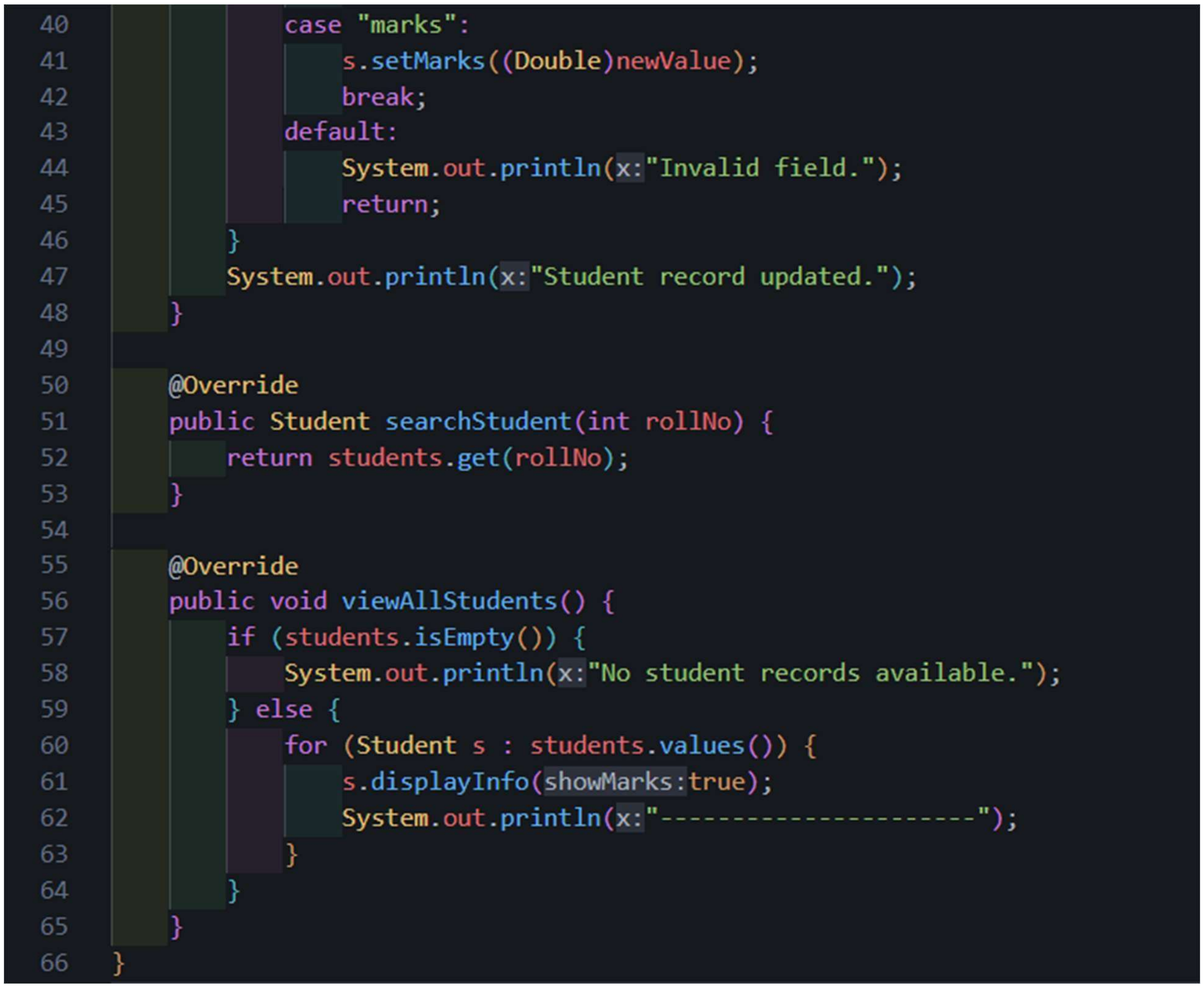
}

SRC/service/StudentManager.java:



service > StudentManager.java > ...

```java
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(x:"Error: Student with this roll number already exists!");
        } else {
            students.put(s.getRollNo(), s);
            System.out.println(x:"Student added successfully.");
        }
    }

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

        switch (field.toLowerCase()) {
            case "course":
                s.setCourse((String)newValue);
                break;
```



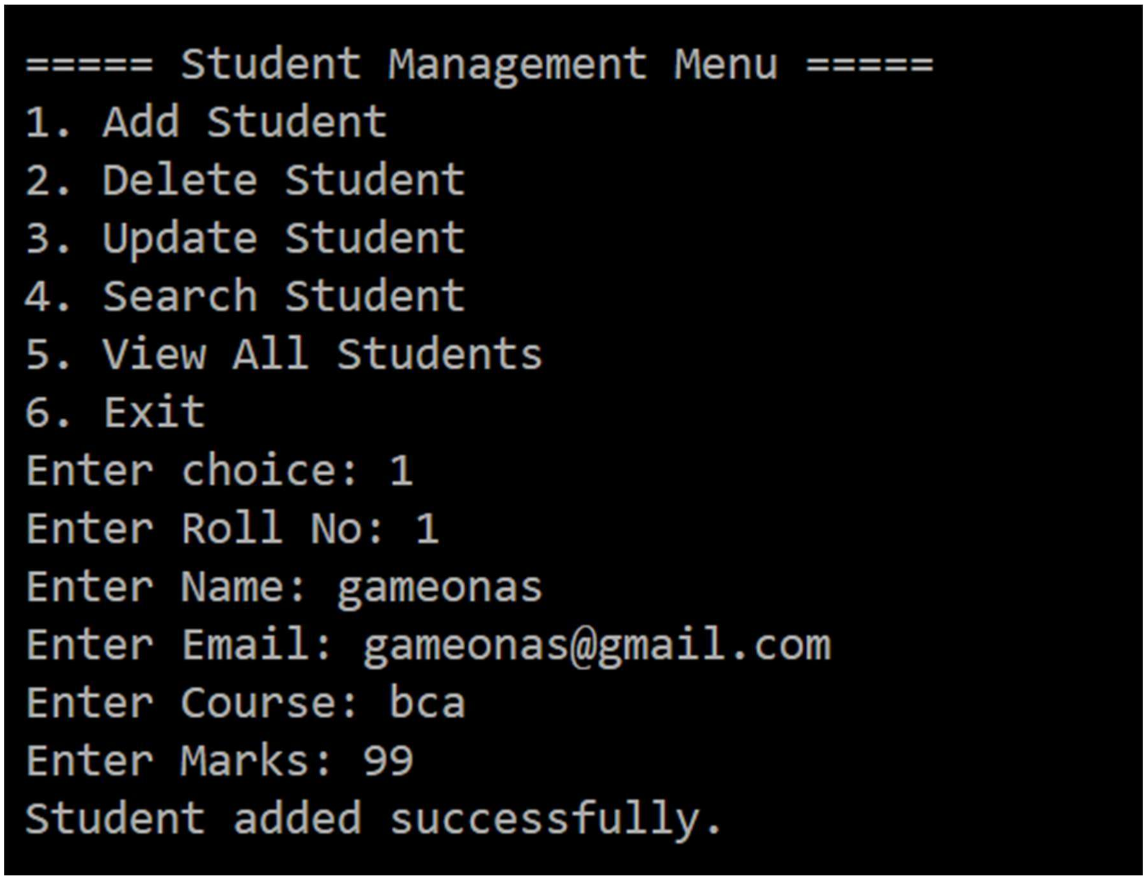
```
            case "marks":
                s.setMarks((Double)newValue);
                break;
            default:
                System.out.println(x:"Invalid field.");
                return;
        }
        System.out.println(x:"Student record updated.");
    }

    @Override
    public Student searchStudent(int rollNo) {
        return students.get(rollNo);
    }

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

Code:

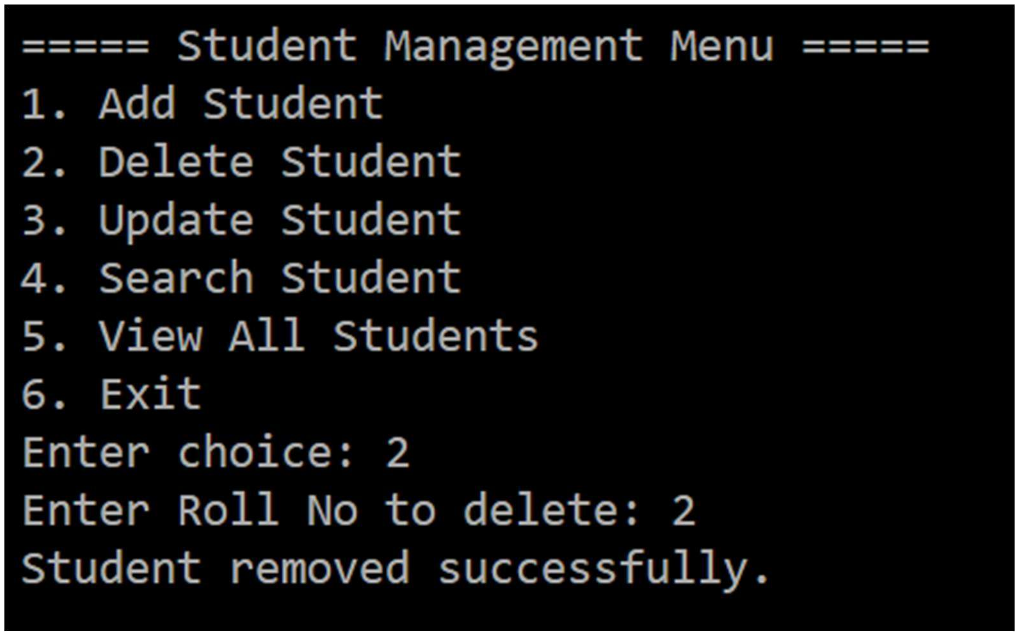
```
package service;

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("----------------------");
           }
       }
   }
}
```

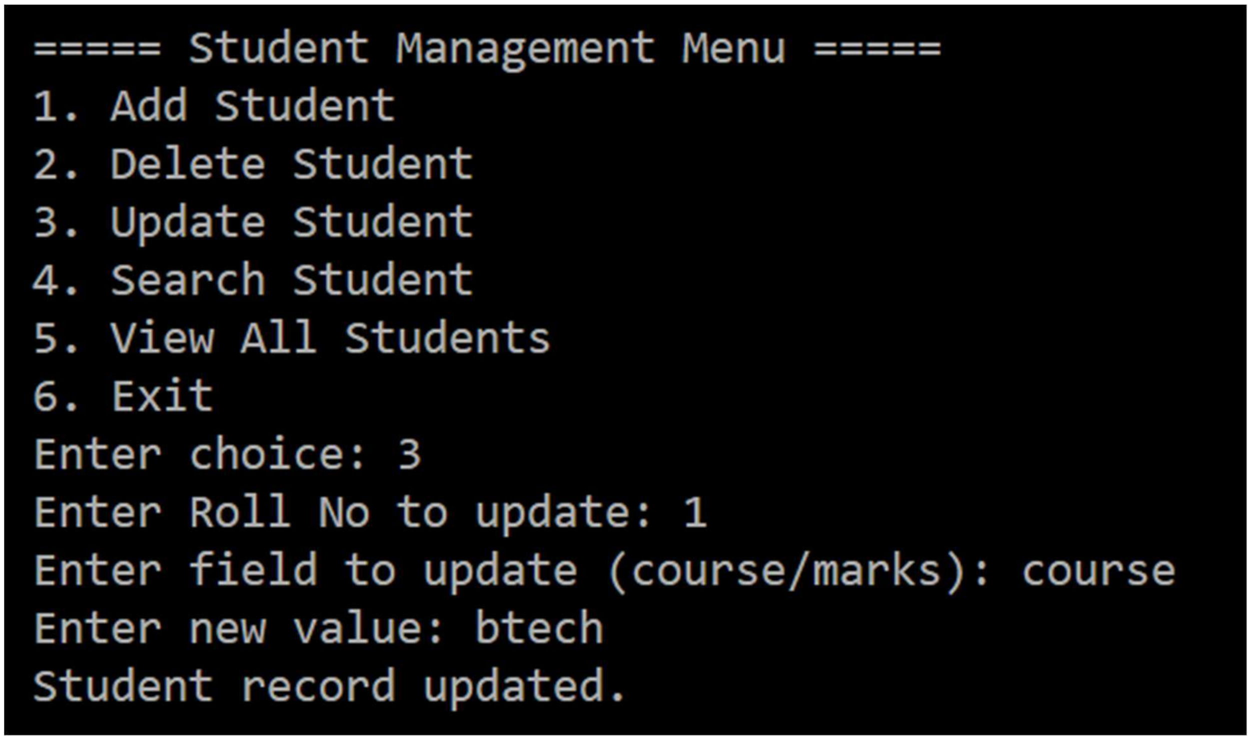
Output:



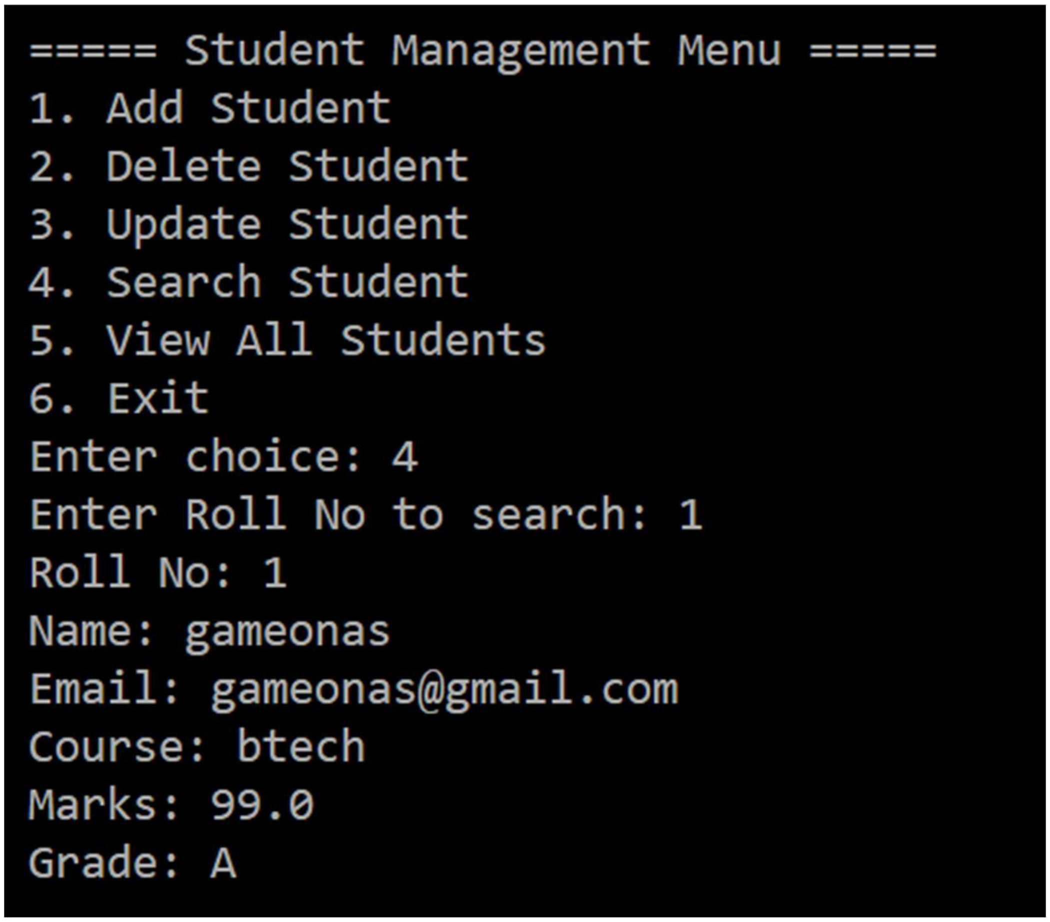
```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 1
Enter Roll No: 1
Enter Name: gameonas
Enter Email: gameonas@gmail.com
Enter Course: bca
Enter Marks: 99
Student added successfully.
```



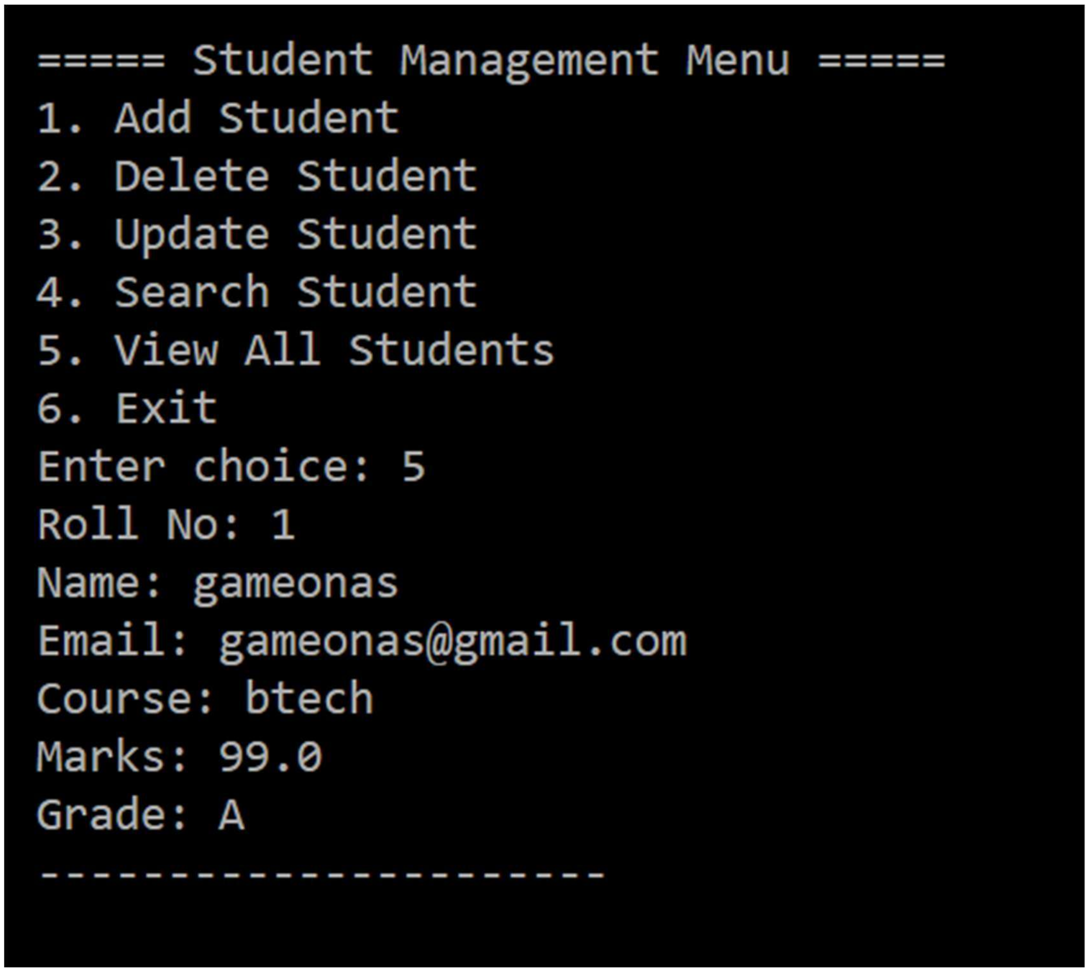
```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 2
Enter Roll No to delete: 2
Student removed successfully.
```



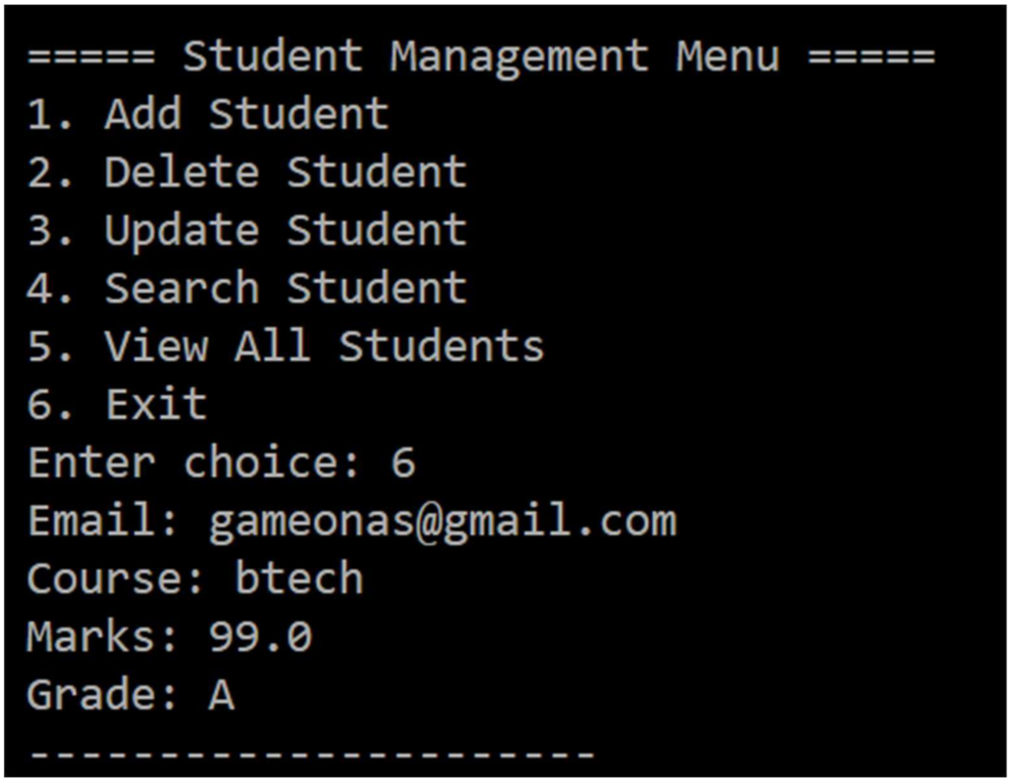
```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 3
Enter Roll No to update: 1
Enter field to update (course/marks): course
Enter new value: btech
Student record updated.
```



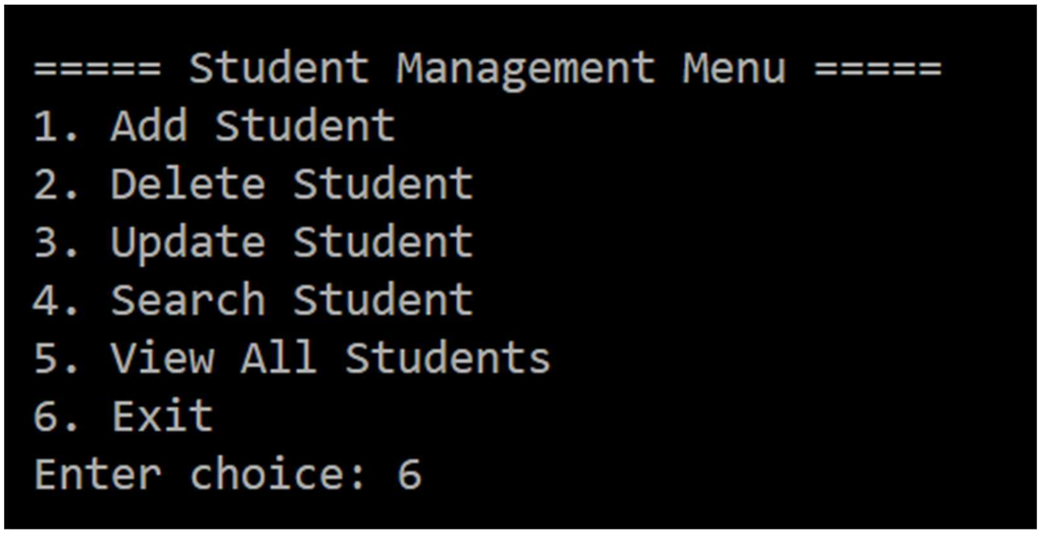
```
===== Student Management Menu =====
1. Add Student
2. Delete Student
3. Update Student
4. Search Student
5. View All Students
6. Exit
Enter choice: 4
Enter Roll No to search: 1
Roll No: 1
Name: gameonas
Email: gameonas@gmail.com
Course: btech
Marks: 99.0
Grade: A
```



```
===== 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.