K.R. Mangalam University

School of Engineering & Technology

Fundamentals Of Java Programming Lab (ENCA203) Assignment

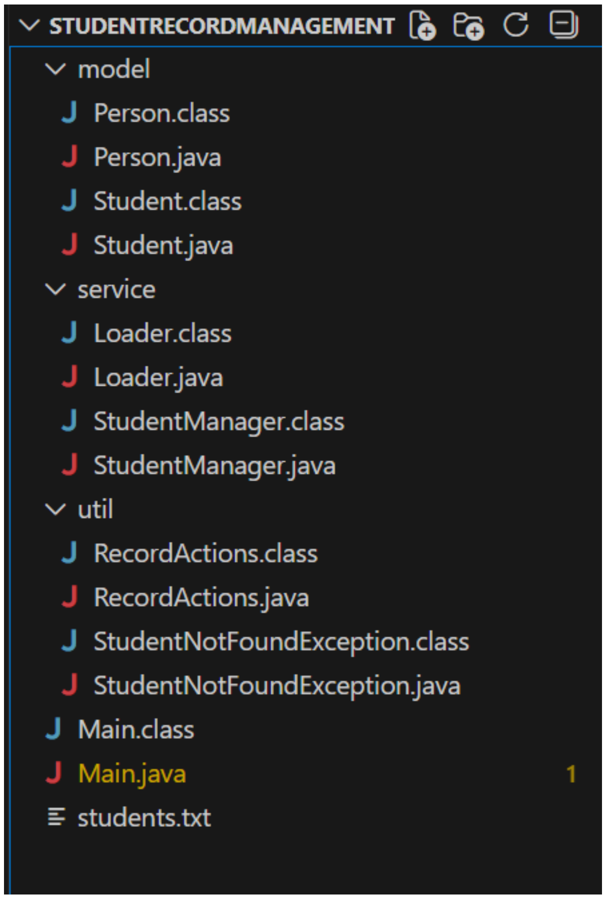
Submitted by:

Name: Bhavya Rattan

Roll No: 2401201004

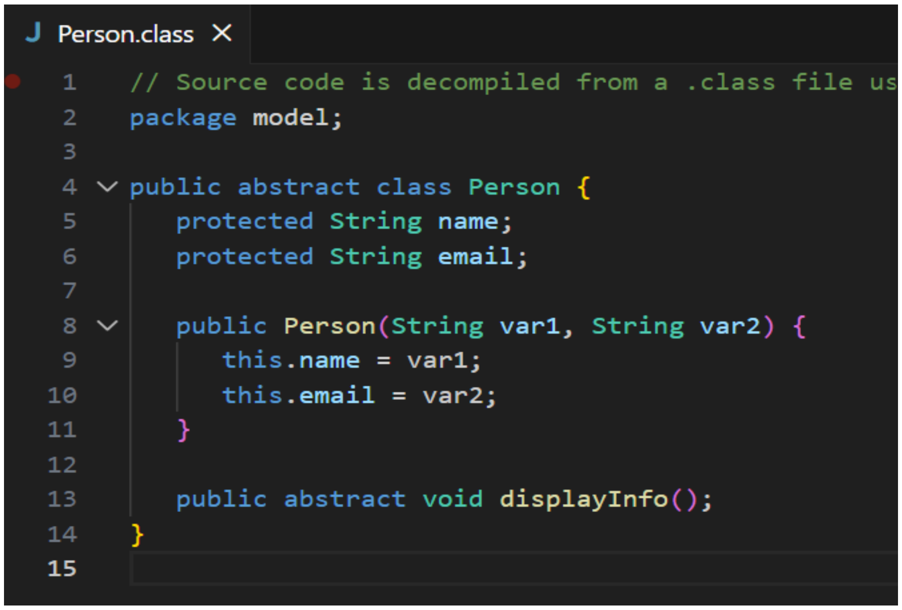
Student Record Management System:

Structure:



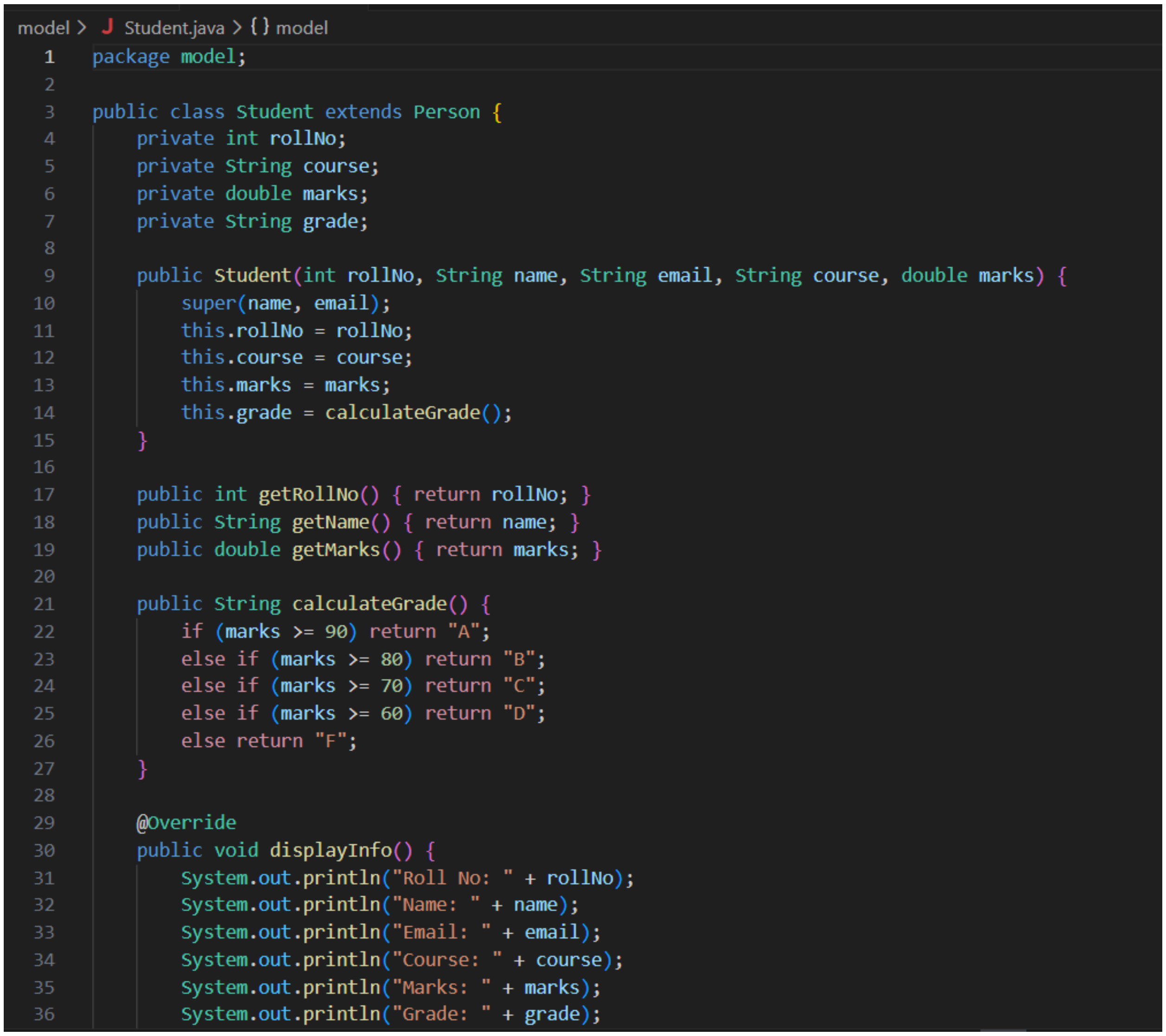
```
STUDENTRECORDMANAGEMENT
  model
    Person.class
    Person.java
    Student.class
    Student.java
  service
    Loader.class
    Loader.java
    StudentManager.class
    StudentManager.java
  util
    RecordActions.class
    RecordActions.java
    StudentNotFoundException.class
    StudentNotFoundException.java
  Main.class
  Main.java                          1
  students.txt
```

Model/Person.java

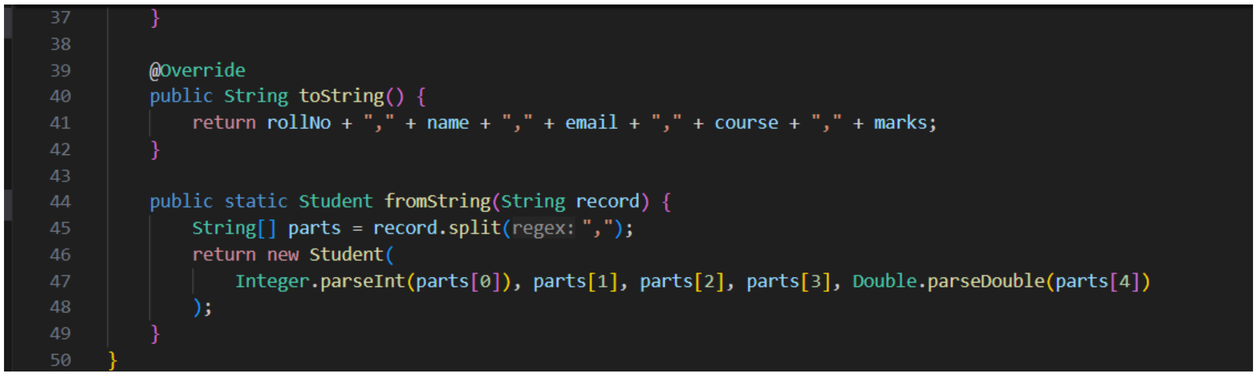

Person.class

```java
// Source code is decompiled from a .class file us
package model;

public abstract class Person {
    protected String name;
    protected String email;

    public Person(String var1, String var2) {
        this.name = var1;
        this.email = var2;
    }

    public abstract void displayInfo();
}

```

Model/Student.java

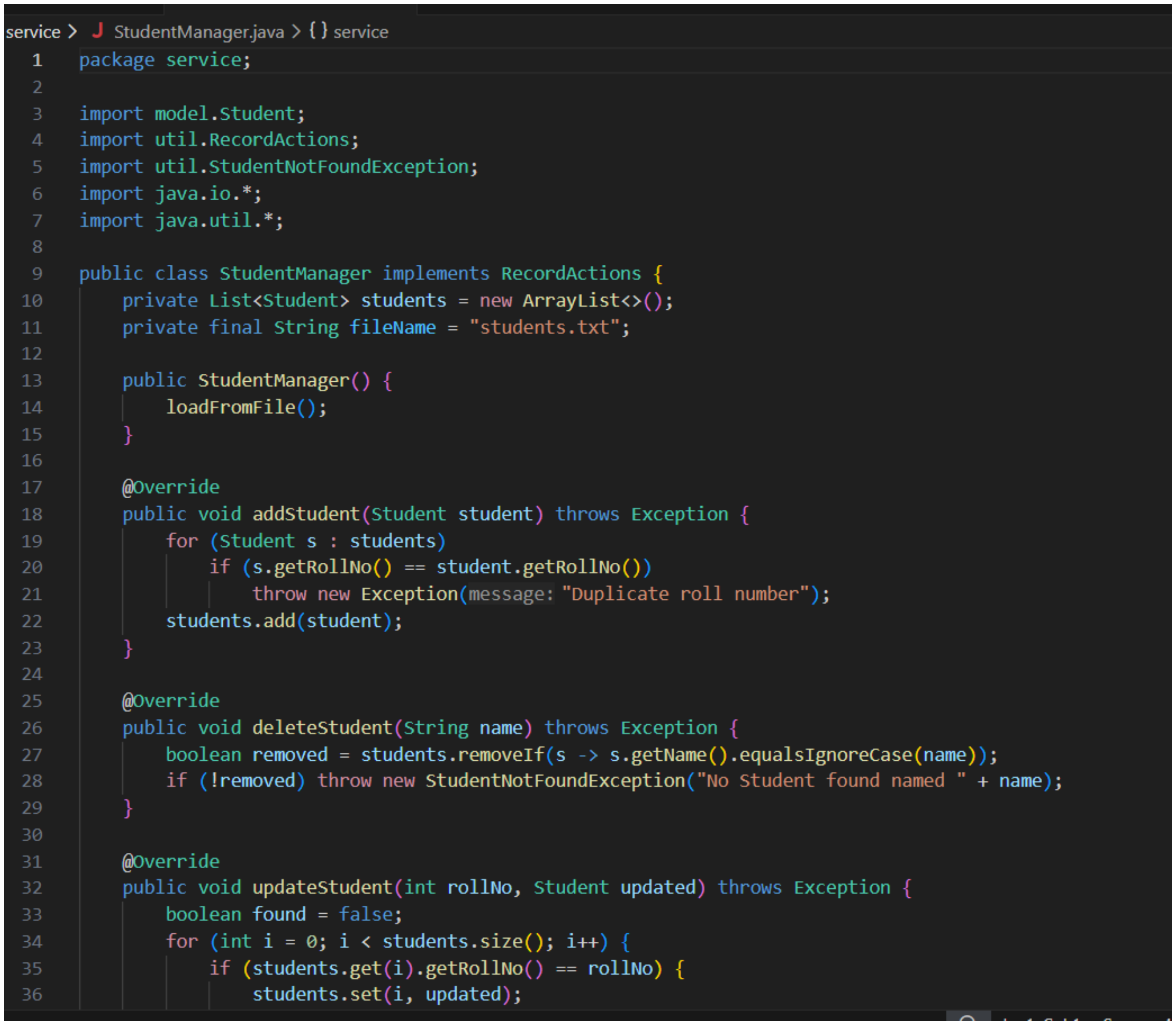


```java
package model;

public class Student extends Person {
    private int rollNo;
    private String course;
    private double marks;
    private String 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;
        this.grade = calculateGrade();
    }

    public int getRollNo() { return rollNo; }
    public String getName() { return name; }
    public double getMarks() { return marks; }

    public String calculateGrade() {
        if (marks >= 90) return "A";
        else if (marks >= 80) return "B";
        else if (marks >= 70) return "C";
        else if (marks >= 60) return "D";
        else return "F";
    }

    @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);
        System.out.println("Marks: " + marks);
        System.out.println("Grade: " + grade);
```



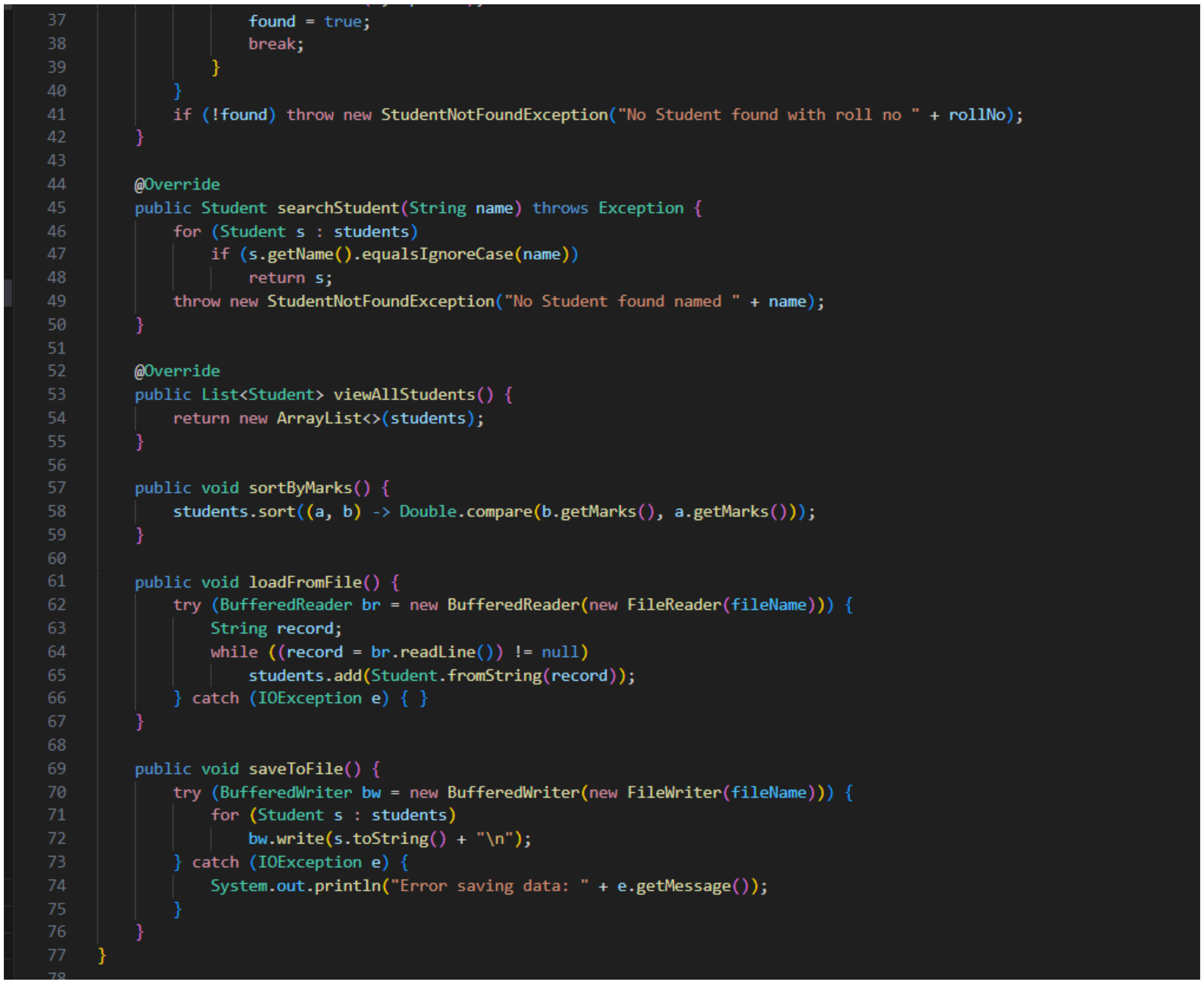
```java
    }

    @Override
    public String toString() {
        return rollNo + "," + name + "," + email + "," + course + "," + marks;
    }

    public static Student fromString(String record) {
        String[] parts = record.split(regex: ",");
        return new Student(
            Integer.parseInt(parts[0]), parts[1], parts[2], parts[3], Double.parseDouble(parts[4])
        );
    }
}
```

Service/StudentManager.java



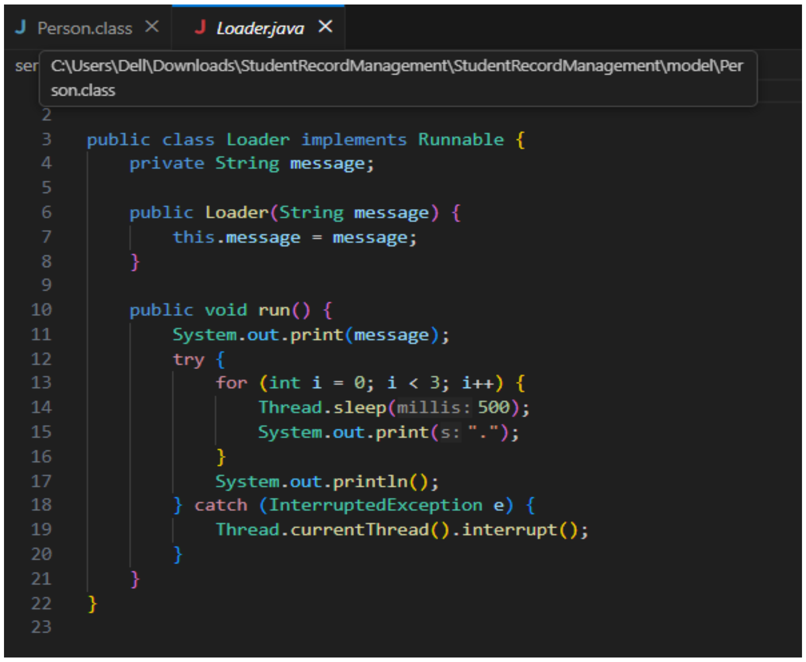
service > StudentManager.java > { } service

```java
package service;

import model.Student;
import util.RecordActions;
import util.StudentNotFoundException;
import java.io.*;
import java.util.*;

public class StudentManager implements RecordActions {
    private List<Student> students = new ArrayList<>();
    private final String fileName = "students.txt";

    public StudentManager() {
        loadFromFile();
    }

    @Override
    public void addStudent(Student student) throws Exception {
        for (Student s : students)
            if (s.getRollNo() == student.getRollNo())
                throw new Exception(message: "Duplicate roll number");
        students.add(student);
    }

    @Override
    public void deleteStudent(String name) throws Exception {
        boolean removed = students.removeIf(s -> s.getName().equalsIgnoreCase(name));
        if (!removed) throw new StudentNotFoundException("No Student found named " + name);
    }

    @Override
    public void updateStudent(int rollNo, Student updated) throws Exception {
        boolean found = false;
        for (int i = 0; i < students.size(); i++) {
            if (students.get(i).getRollNo() == rollNo) {
                students.set(i, updated);
```



```java
                found = true;
                break;
            }
        }
        if (!found) throw new StudentNotFoundException("No Student found with roll no " + rollNo);
    }

    @Override
    public Student searchStudent(String name) throws Exception {
        for (Student s : students)
            if (s.getName().equalsIgnoreCase(name))
                return s;
        throw new StudentNotFoundException("No Student found named " + name);
    }

    @Override
    public List<Student> viewAllStudents() {
        return new ArrayList<>(students);
    }

    public void sortByMarks() {
        students.sort((a, b) -> Double.compare(b.getMarks(), a.getMarks()));
    }

    public void loadFromFile() {
        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String record;
            while ((record = br.readLine()) != null)
                students.add(Student.fromString(record));
        } catch (IOException e) { }
    }

    public void saveToFile() {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileName))) {
            for (Student s : students)
                bw.write(s.toString() + "\n");
        } catch (IOException e) {
            System.out.println("Error saving data: " + e.getMessage());
        }
    }
}
```

service/Loader.java

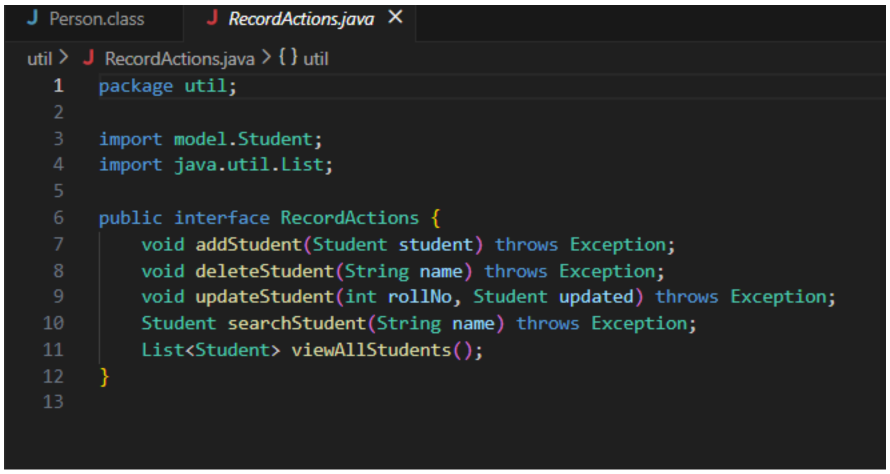


Person.class × Loader.java ×

ser C:\Users\Dell\Downloads\StudentRecordManagement\StudentRecordManagement\model\Person.class

```java

public class Loader implements Runnable {
    private String message;

    public Loader(String message) {
        this.message = message;
    }

    public void run() {
        System.out.print(message);
        try {
            for (int i = 0; i < 3; i++) {
                Thread.sleep(millis: 500);
                System.out.print(s: ".");
            }
            System.out.println();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

```

util/RecordActions.java

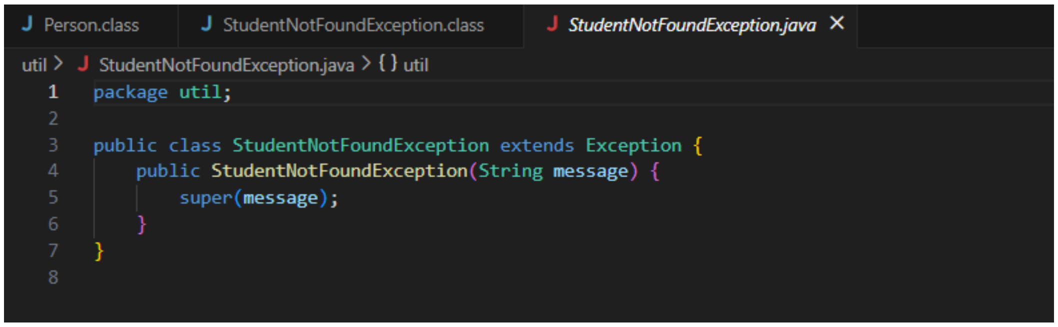


Person.class RecordActions.java ×

util > RecordActions.java > { } util

```java
package util;

import model.Student;
import java.util.List;

public interface RecordActions {
    void addStudent(Student student) throws Exception;
    void deleteStudent(String name) throws Exception;
    void updateStudent(int rollNo, Student updated) throws Exception;
    Student searchStudent(String name) throws Exception;
    List<Student> viewAllStudents();
}

```

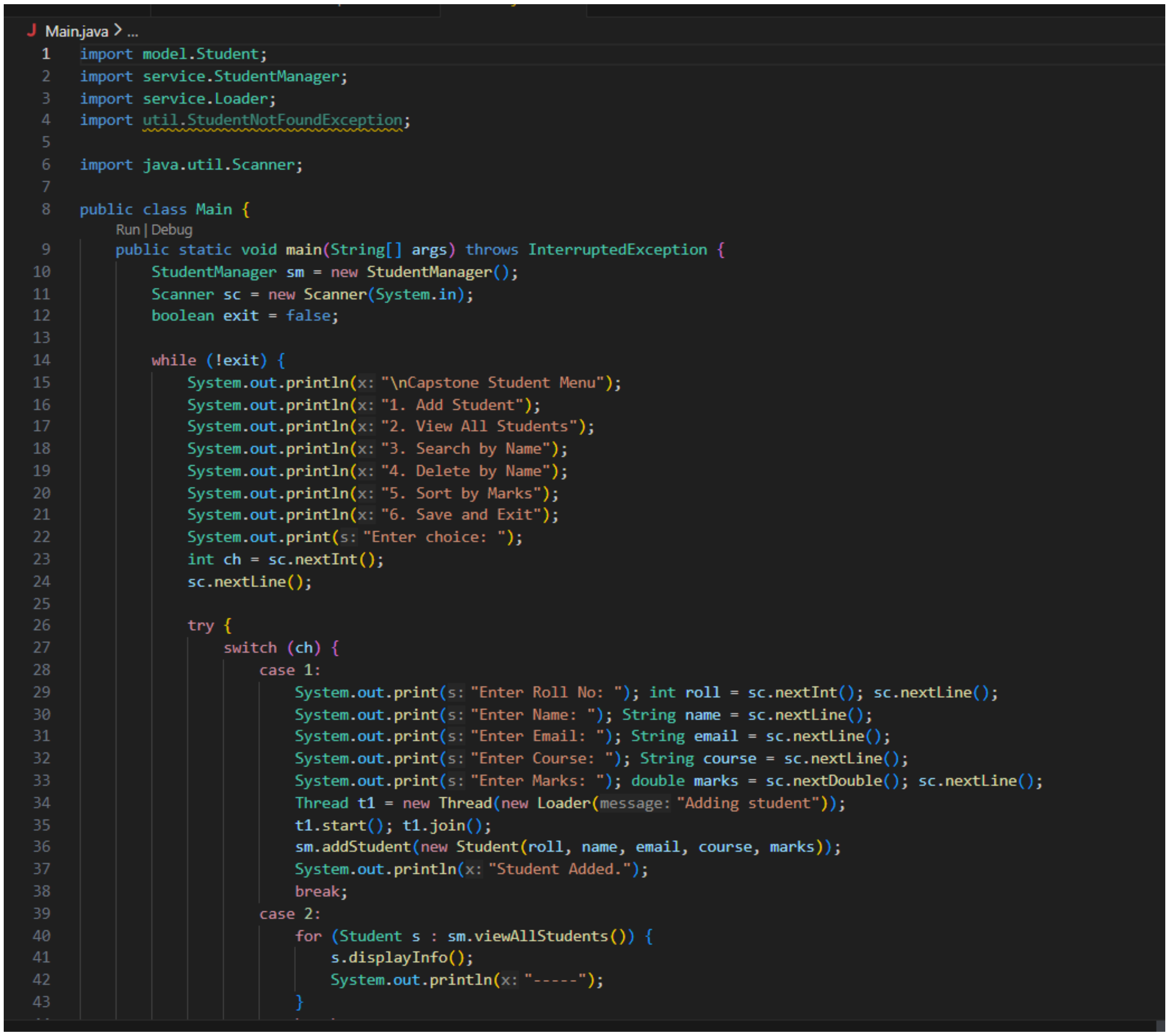
util/StudentNotFoundException.java


Person.class | StudentNotFoundException.class | StudentNotFoundException.java

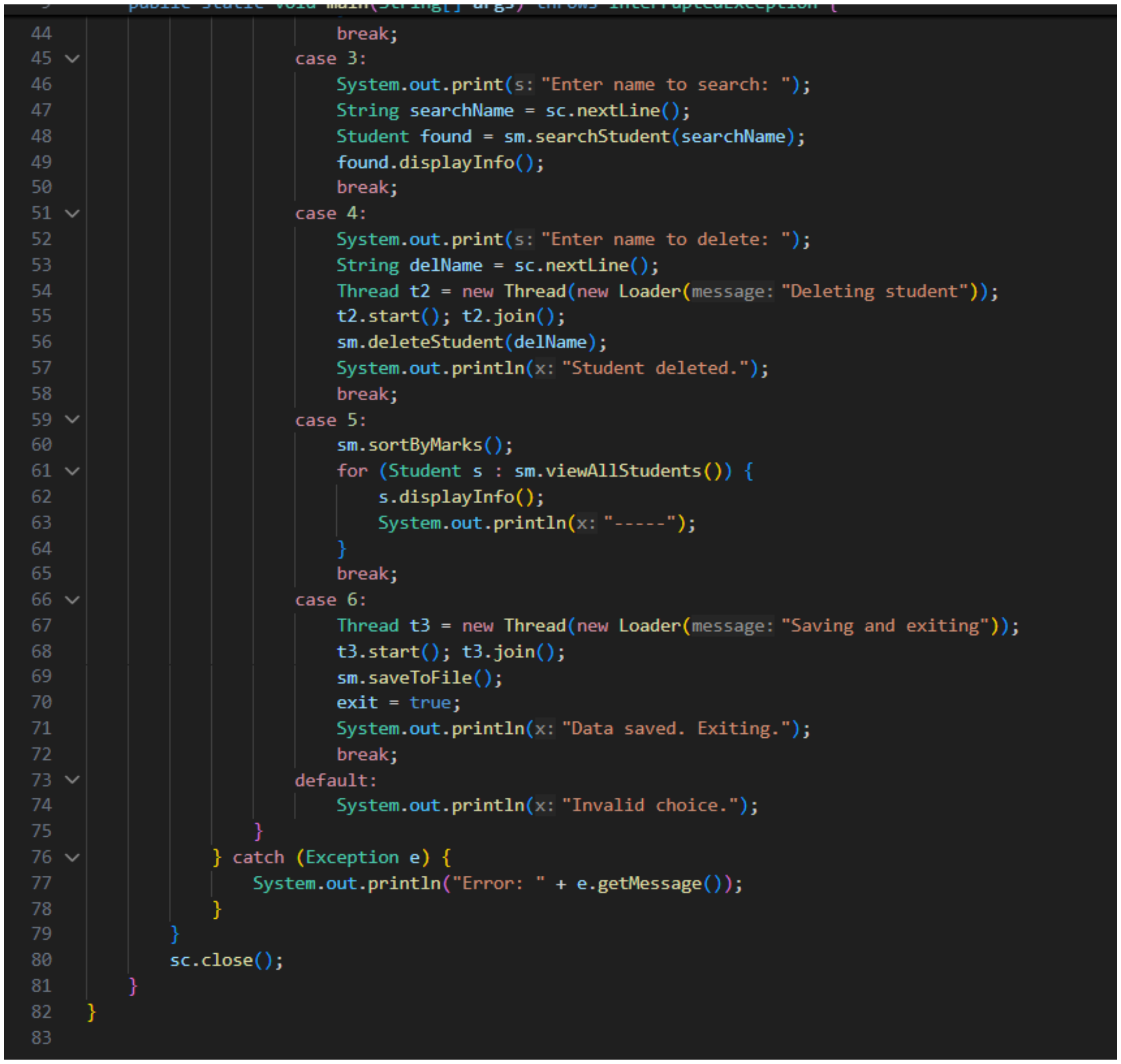
util > StudentNotFoundException.java > { } util

```java
package util;

public class StudentNotFoundException extends Exception {
    public StudentNotFoundException(String message) {
        super(message);
    }
}

```

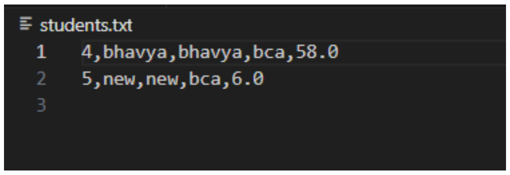
Main.java


Main.java > ...

```java
import model.Student;
import service.StudentManager;
import service.Loader;
import util.StudentNotFoundException;

import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) throws InterruptedException {
        StudentManager sm = new StudentManager();
        Scanner sc = new Scanner(System.in);
        boolean exit = false;

        while (!exit) {
            System.out.println(x: "\nCapstone Student Menu");
            System.out.println(x: "1. Add Student");
            System.out.println(x: "2. View All Students");
            System.out.println(x: "3. Search by Name");
            System.out.println(x: "4. Delete by Name");
            System.out.println(x: "5. Sort by Marks");
            System.out.println(x: "6. Save and Exit");
            System.out.print(s: "Enter choice: ");
            int ch = sc.nextInt();
            sc.nextLine();

            try {
                switch (ch) {
                    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(); sc.nextLine();
                        Thread t1 = new Thread(new Loader(message: "Adding student"));
                        t1.start(); t1.join();
                        sm.addStudent(new Student(roll, name, email, course, marks));
                        System.out.println(x: "Student Added.");
                        break;
                    case 2:
                        for (Student s : sm.viewAllStudents()) {
                            s.displayInfo();
                            System.out.println(x: "-----");
                        }
```

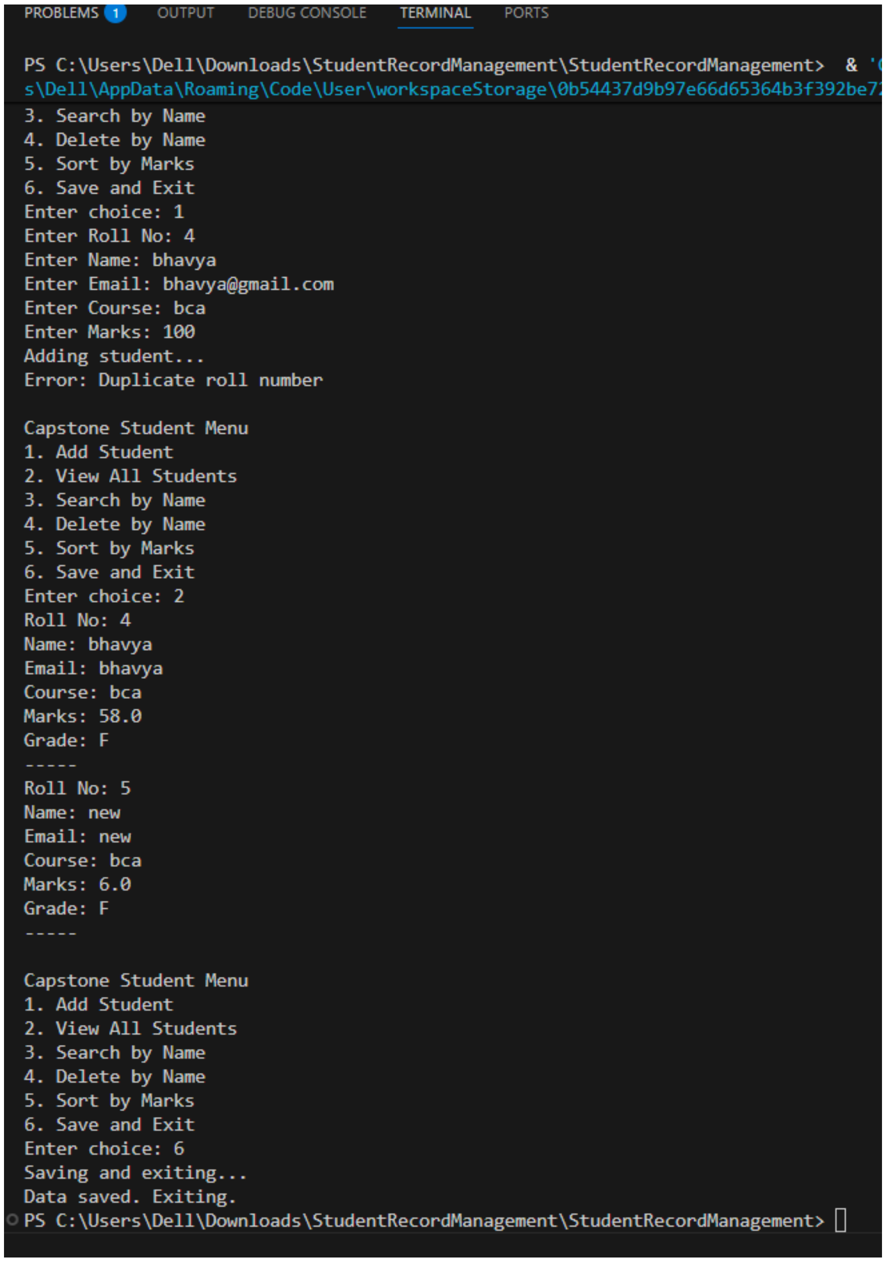

```java
                        break;
                    case 3:
                        System.out.print(s: "Enter name to search: ");
                        String searchName = sc.nextLine();
                        Student found = sm.searchStudent(searchName);
                        found.displayInfo();
                        break;
                    case 4:
                        System.out.print(s: "Enter name to delete: ");
                        String delName = sc.nextLine();
                        Thread t2 = new Thread(new Loader(message: "Deleting student"));
                        t2.start(); t2.join();
                        sm.deleteStudent(delName);
                        System.out.println(x: "Student deleted.");
                        break;
                    case 5:
                        sm.sortByMarks();
                        for (Student s : sm.viewAllStudents()) {
                            s.displayInfo();
                            System.out.println(x: "-----");
                        }
                        break;
                    case 6:
                        Thread t3 = new Thread(new Loader(message: "Saving and exiting"));
                        t3.start(); t3.join();
                        sm.saveToFile();
                        exit = true;
                        System.out.println(x: "Data saved. Exiting.");
                        break;
                    default:
                        System.out.println(x: "Invalid choice.");
                }
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
        sc.close();
    }
}
```

Students.txt:


students.txt

```
4,bhavya,bhavya,bca,58.0
5,new,new,bca,6.0

```

Output:-



```
PS C:\Users\Dell\Downloads\StudentRecordManagement\StudentRecordManagement>  & '
s\Dell\AppData\Roaming\Code\User\workspaceStorage\0b54437d9b97e66d65364b3f392be7
3. Search by Name
4. Delete by Name
5. Sort by Marks
6. Save and Exit
Enter choice: 1
Enter Roll No: 4
Enter Name: bhavya
Enter Email: bhavya@gmail.com
Enter Course: bca
Enter Marks: 100
Adding student...
Error: Duplicate roll number

Capstone Student Menu
1. Add Student
2. View All Students
3. Search by Name
4. Delete by Name
5. Sort by Marks
6. Save and Exit
Enter choice: 2
Roll No: 4
Name: bhavya
Email: bhavya
Course: bca
Marks: 58.0
Grade: F
-----
Roll No: 5
Name: new
Email: new
Course: bca
Marks: 6.0
Grade: F
-----

Capstone Student Menu
1. Add Student
2. View All Students
3. Search by Name
4. Delete by Name
5. Sort by Marks
6. Save and Exit
Enter choice: 6
Saving and exiting...
Data saved. Exiting.
PS C:\Users\Dell\Downloads\StudentRecordManagement\StudentRecordManagement>
```

Explanation:

Objective

The objective of this project was to build a Student Record Management System in Java that is modular, object-oriented, and demonstrates core Java programming principles. The system allows users to add, view, search, update, delete, and sort student records. All records are stored persistently in a text file so data is automatically loaded at startup and saved upon exit.

Key Features

- **Object-Oriented Architecture:**
  Implemented using an abstract class (Person), a subclass (Student), an interface (RecordActions), and a custom exception class (StudentNotFoundException).
- **Full CRUD Functionality:**
  Users can easily add, view, search, update, and delete student entries via a console-driven menu.
- **Validation & Exception Handling:**
  Duplicate roll numbers, invalid inputs, or missing records trigger meaningful error messages. Custom exceptions handle scenarios such as searching for non-existent students.
- **Persistent Storage:**
  Student data is stored in a students.txt file using BufferedReader and BufferedWriter. The system loads existing records on startup and saves updated data on exit.
- **Use of Java Collections:**
  An ArrayList is used to maintain student objects. Sorting functionality is implemented using a Comparator based on student marks.
- **Multithreading:**
  A Loader class (implementing Runnable) simulates time-consuming tasks like saving or loading data, providing a more interactive experience using Java threads.
- **User-Friendly Console Interface:**
  The main menu provides clear options for all features, making navigation simple and intuitive.

Project Structure

- **model/** – Contains data classes such as Person (abstract) and Student.
- **util/** – Includes the RecordActions interface and custom exception StudentNotFoundException.

- **service/** – Houses core business logic (StudentManager) and the threaded loader (Loader).
- **students.txt** – Text file used for persistent student data storage.
- **Main.java** – Entry point containing the interactive console menu.

How to Run

1. Compile all files from the project root.
2. Run the Main class.
3. Use the console menu to add, view, search, update, delete, sort, or save student records.
4. Restart the program to verify that data is correctly reloaded from students.txt.