K.R. Mangalam University

School of Engineering & Technology

Fundamentals Of Java Programming Lab

(ENCA203) Assignment 3

Student Management System

Submitted by:

Name: Khushi

Roll No: 2401201102

Course: BCA (AI & DS)

Section: B

Code:

ResultManager.java > ResultManager

```java
import java.util.*;

// Custom Exception for missing students
class StudentNotFoundException extends Exception {
    public StudentNotFoundException (String message) {
        super(message);
    }
}

// Interface for defining record actions
interface RecordActions {
    void addStudent();
    void displayStudents();
}

// Loader class to simulate loading (Multithreading)
class Loader implements Runnable {
    private String task;

    Loader (String task) {
        this.task = task;
    }

    @Override
    public void run() {
        System.out.print(task);
        try {
            for (int i = 0; i < 5; i++) {
                System.out.print(s: ".");
                Thread.sleep(millis: 100); // Reduced sleep time for quicker demo
            }
        } catch (InterruptedException e) {
```

```
            System.out.println(x: "\nLoading interrupted!");
        }
        System.out.println(x: "\nLoading completed!\n");
    }
}

// Student class using wrapper classes (Integer, Double)
class Student {
    private Integer rollNo;
    private String name;
    private String email;
    private String course;
    private Double marks;

    public Student (Integer rollNo, String name, String email, String course, Double marks) {
        this.rollNo = rollNo;
        this.name = name;
        this.email = email;
        this.course = course;
        this.marks = marks;
    }

    // Getter for rollNo to check for duplicates
    public Integer getRollNo() {
        return rollNo;
    }

    public String calculateGrade() {
        if (marks >= 90) return "A";
        else if (marks >= 75) return "B";
```

```
        else if (marks >= 60) return "C";
        else if (marks >= 40) return "D";
        else if (marks >= 40) return "D";
        else return "F";
    }

    public void display() {
        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: " + calculateGrade());
    }
}

// Manager class implementing interface
class StudentManager implements RecordActions {
    private List<Student> students = new ArrayList<>();
    private Scanner sc = new Scanner(System.in);

    // Helper method to check for duplicate roll numbers
    private boolean isRollNoDuplicate(Integer rollNo) {
        for (Student s : students) {
            if (s.getRollNo().equals(rollNo)) {
                return true;
            }
        }
        return false;
    }
```

```
93
94      @Override
95      public void addStudent() {
96          try {
97              System.out.print(s: "Enter Roll No (Integer): ");
98              // Input validation for rollNo
99              Integer rollNo = Integer.parseInt(sc.nextLine());
100
101             // Check for duplicate roll number
102             if (isRollNoDuplicate(rollNo)) {
103                 throw new IllegalArgumentException("Student with Roll No " + rollNo + " already exists!");
104             }
105
106             System.out.print(s: "Enter Name: ");
107             String name = sc.nextLine().trim();
108             if (name.isEmpty()) throw new IllegalArgumentException(s: "Name cannot be empty!");
109
110             System.out.print(s: "Enter Email: ");
111             String email = sc.nextLine().trim();
112             if (email.isEmpty()) throw new IllegalArgumentException(s: "Email cannot be empty!");
113
114             System.out.print(s: "Enter Course: ");
115             String course = sc.nextLine().trim();
116             if (course.isEmpty()) throw new IllegalArgumentException(s: "Course cannot be empty!");
117
118             System.out.print(s: "Enter Marks: ");
119             // Input validation for marks
120             Double marks = Double.parseDouble(sc.nextLine());
121
122             if (marks < 0 || marks > 100)
123                 throw new IllegalArgumentException(s: "Marks must be between 0 and 100!");
124
125             // Simulate loading with thread
126             Thread loaderThread = new Thread(new Loader(task: "Saving student record"));
127             loaderThread.start();
128             loaderThread.join(); // Wait for the loader thread to finish
129
130             students.add(new Student(rollNo, name, email, course, marks));
131             System.out.println(x: "Student added successfully!\n");
132         } catch (NumberFormatException e) {
133             System.out.println(x: "Invalid input format. Please enter correct numeric values for Roll No/Marks.");
134         } catch (IllegalArgumentException e) {
135             System.out.println("Error: " + e.getMessage());
136         } catch (InterruptedException e) {
137             System.out.println("Thread interrupted during loading: " + e.getMessage());
138         } finally {
139             System.out.println(x: "Data input process completed.\n");
140         }
141     }
142
143     @Override
144     public void displayStudents() {
145         try {
146             // Simulate loading before displaying records
147             Thread loaderThread = new Thread(new Loader(task: "Fetching student records"));
148             loaderThread.start();
149             loaderThread.join(); // Wait for the loader thread to finish
150
151             if (students.isEmpty())
```

```
152                 throw new StudentNotFoundException(message: "No student records found!");
153
154             System.out.println(x: "----- Student Records ----");
155             for (Student s : students) {
156                 s.display();
157                 System.out.println(x: "-------------------------");
158             }
159         } catch (StudentNotFoundException e) {
160             System.out.println(e.getMessage());
161         } catch (InterruptedException e) {
162             System.out.println("Display interrupted: " + e.getMessage());
163         }
164     }
165 }
166
167 // Main class matches file name
168 public class ResultManager {
        Run | Debug
169     public static void main(String[] args) {
170         System.out.println(x: "================================================");
171         System.out.println(x: "STUDENT MANAGEMENT SYSTEM RESULT MANAGER");
172         // --- MODIFIED LINE ---
173         System.out.println(x: "Developed by: Khushi");
174         // --- MODIFIED LINE ---
175         System.out.println(x: "Email: khushi@gmail.com");
176         System.out.println(x: "K.R. Mangalam University");
177         System.out.println(x: "================================================\n");
178
179         StudentManager manager = new StudentManager();
```

```
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println(x: "===== MAIN MENU =====");
            System.out.println(x: "1. Add Student");
            System.out.println(x: "2. Display Students");
            System.out.println(x: "3. Exit");
            System.out.print(s: "Enter your choice: ");

            int choice;
            try {
                // Read the whole line to avoid issues with nextInt/nextLine
                String input = sc.nextLine().trim();
                if (input.isEmpty()) {
                    System.out.println(x: "Please enter a choice.\n");
                    continue;
                }
                choice = Integer.parseInt(input);
            } catch (NumberFormatException e) {
                System.out.println(x: "Please enter a valid number (1, 2, or 3)!\n");
                continue;
            }

            switch (choice) {
                case 1:
                    manager.addStudent();
                    break;
                case 2:
                    manager.displayStudents();
                    break;
                case 3:
                    System.out.println(x: "Exiting system... Goodbye!");
                    sc.close(); // Close the main Scanner before exiting
                    return;
                default:
                    System.out.println(x: "Invalid choice! Please try again.\n");
            }
        }
    }
}
```

Output:

```
================================================
STUDENT MANAGEMENT SYSTEM RESULT MANAGER
Developed by: Khushi
Email: khushi@gmail.com
K.R. Mangalam University
================================================

===== MAIN MENU =====
1. Add Student
2. Display Students
3. Exit
Enter your choice:
```

```
===== MAIN MENU =====
1. Add Student
2. Display Students
3. Exit
Enter your choice: 1
Enter Roll No (Integer): 1111
Enter Name: khushi
Enter Email: khushi@gmail.com
Enter Course: bca
Enter Marks: 99
Saving student record.....
Loading completed!

Student added successfully!

Data input process completed.

===== MAIN MENU =====
1. Add Student
2. Display Students
3. Exit
Enter your choice: 2
Fetching student records.....
Loading completed!

----- Student Records ----
Roll No: 1111
Name: khushi
Email: khushi@gmail.com
Course: bca
Marks: 99.0
```

```
Grade: A
--------------------------
===== MAIN MENU =====
1. Add Student
2. Display Students
3. Exit
Enter your choice: 3
Exiting system... Goodbye!
PS C:\Users\Hp\.java>
```