K.R. Mangalam University
School of Engineering & Technology

Fundamentals Of Java Programming Lab
(ENCA203) Assignment 3
Student Management System

Submitted by:

Name: RAKESH G

Roll No: 2401201064

Class: BCA (AI & DS)

Submitted to:

Dr. Manish Kumar

GitHub Repository:

https://github.com/rakesh4407/Java_Assignment_Rakesh

CODE:

Rakesh_Assignment.ipynb ● StudentManagementSystem.java 3 ●

JAVA > StudentManagementSystem.java > ...

```java
import java.util.*;

// RAKESH G  2401201064
class StudentNotFoundException extends Exception {

    public StudentNotFoundException(String message) {
        super(message);
    }
}

interface RecordActions {

    void addStudent();

    void displayStudent(int rollNo) throws StudentNotFoundException;
}

class Loader implements Runnable {

    @Override
    public void run() {
        try {
            System.out.print("Loading");
            for (int i = 0; i < 5; i++) {
                Thread.sleep(400);    Thread.sleep called in loop
                System.out.print(".");
            }
            System.out.println();
        } catch (InterruptedException e) {
            System.out.println("Loading interrupted!");
        }
    }
}

class StudentManager implements RecordActions {

    private static class Student {

```

Rakesh_Assignment.ipynb ● StudentManagementSystem.java 3 ●

JAVA > StudentManagementSystem.java > ...

```java
class StudentManager implements RecordActions {
    private static class Student {
        Integer rollNo;
        String name;
        String email;
        String course;
        Double marks;

        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;
        }

        String getGrade() {
            if (marks >= 90) {
                return "A";
            } else if (marks >= 75) {
                return "B";
            } else if (marks >= 60) {
                return "C";
            } else if (marks >= 40) {
                return "D";
            } else {
                return "F";
            }
        }

        @Override
        public String toString() {
            return "Roll No: " + rollNo
                    + "\nName: " + name
                    + "\nEmail: " + email
                    + "\nCourse: " + course
                    + "\nMarks: " + marks
                    + "\nGrade: " + getGrade();
```

Rakesh_Assingment.ipynb ● StudentManagementSystem.java 3 ●

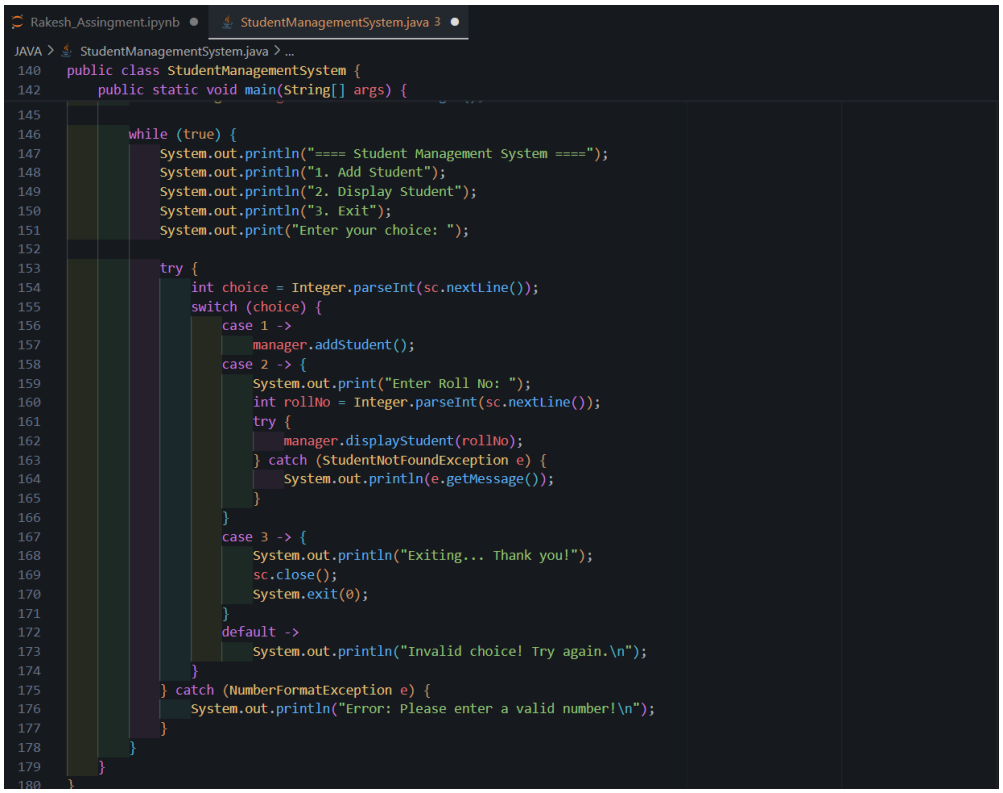
JAVA > StudentManagementSystem.java > ...

```
35  class StudentManager implements RecordActions {
37      private static class Student {
68          public String toString() {
75          }
76      }
77
78      private final Map<Integer, Student> students = new HashMap<>();
79      private final Scanner sc = new Scanner(System.in);
80
81      @Override
82      public void addStudent() {
83          try {
84              System.out.print("Enter Roll No (Integer): ");
85              Integer rollNo = Integer.parseInt(sc.nextLine());    Unnecessary temporary when converting from String
86
87              System.out.print("Enter Name: ");
88              String name = sc.nextLine();
89              if (name.isEmpty()) {
90                  throw new IllegalArgumentException("Name cannot be empty!");
91              }
92
93              System.out.print("Enter Email: ");
94              String email = sc.nextLine();
95              if (email.isEmpty()) {
96                  throw new IllegalArgumentException("Email cannot be empty!");
97              }
98
99              System.out.print("Enter Course: ");
100             String course = sc.nextLine();
101             if (course.isEmpty()) {
102                 throw new IllegalArgumentException("Course cannot be empty!");
103             }
104
105             System.out.print("Enter Marks: ");
106             Double marks = Double.parseDouble(sc.nextLine());    Unnecessary temporary when converting from String
107             if (marks < 0 || marks > 100) {
108                 throw new IllegalArgumentException("Marks must be between 0 and 100!");
109             }
```

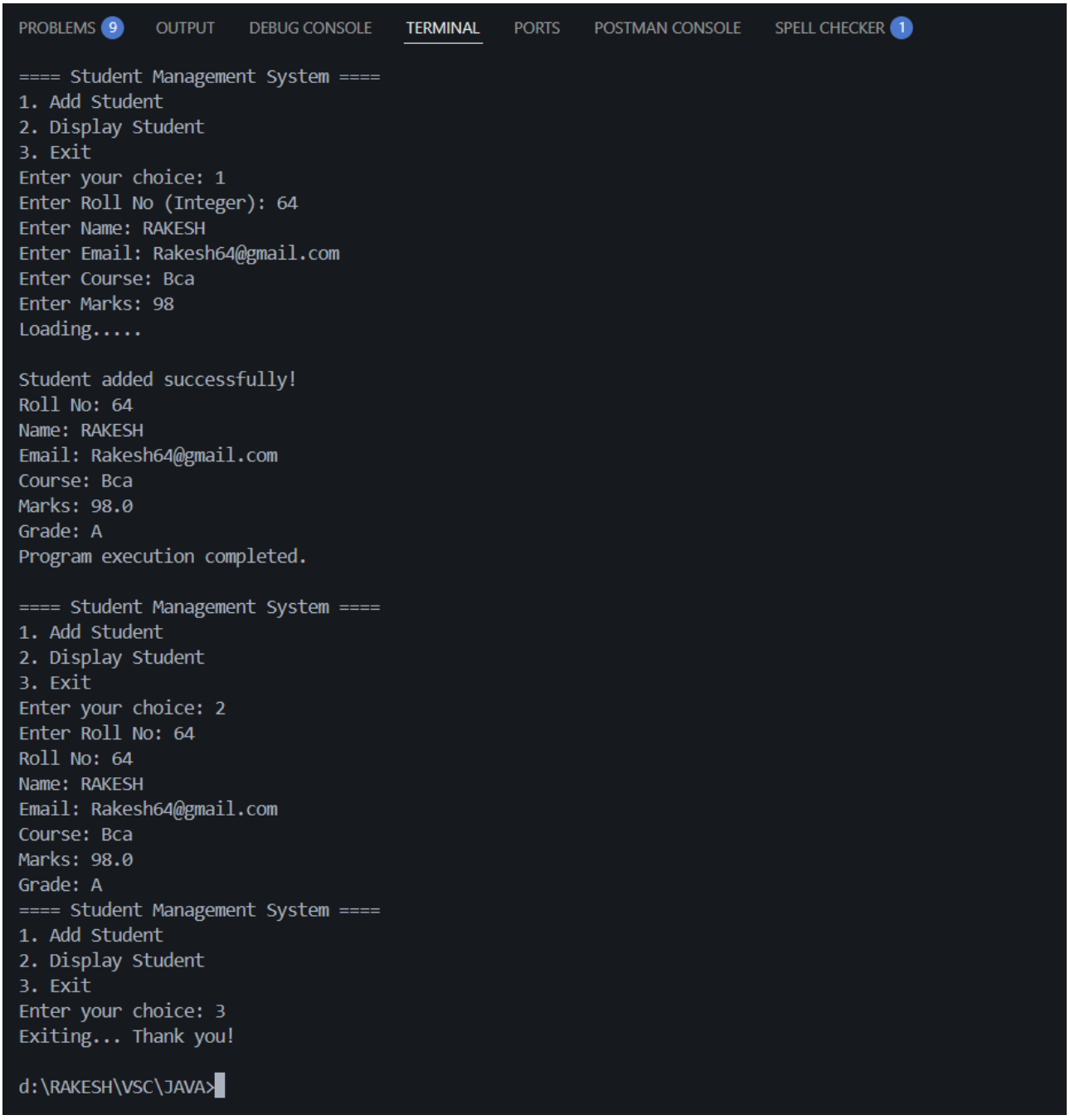
Rakesh_Assingment.ipynb ● StudentManagementSystem.java 3 ●

JAVA > StudentManagementSystem.java > ...

```
35  class StudentManager implements RecordActions {
82      public void addStudent() {
110
111             Thread loaderThread = new Thread(new Loader());
112             loaderThread.start();
113             loaderThread.join();
114
115             students.put(rollNo, new Student(rollNo, name, email, course, marks));
116             System.out.println("\nStudent added successfully!");
117             System.out.println(students.get(rollNo));
118
119         } catch (NumberFormatException e) {
120             System.out.println("Error: Invalid number format!");
121         } catch (IllegalArgumentException e) {
122             System.out.println("Error: " + e.getMessage());
123         } catch (InterruptedException e) {
124             System.out.println("Error: Loading interrupted!");
125         } finally {
126             System.out.println("Program execution completed.\n");
127         }
128     }
129
130     @Override
131     public void displayStudent(int rollNo) throws StudentNotFoundException {
132         Student student = students.get(rollNo);
133         if (student == null) {
134             throw new StudentNotFoundException("Student with Roll No " + rollNo + " not found!");
135         }
136         System.out.println(student);
137     }
138 }
139
140 public class StudentManagementSystem {
141
        Run main | Debug main | Run | Debug
142     public static void main(String[] args) {
143         Scanner sc = new Scanner(System.in);
144         StudentManager manager = new StudentManager();
```


Rakesh_Assingment.ipynb
StudentManagementSystem.java 3
JAVA > StudentManagementSystem.java > ...
public class StudentManagementSystem {
public static void main(String[] args) {
while (true) {
System.out.println("==== Student Management System ====");
System.out.println("1. Add Student");
System.out.println("2. Display Student");
System.out.println("3. Exit");
System.out.print("Enter your choice: ");
try {
int choice = Integer.parseInt(sc.nextLine());
switch (choice) {
case 1 ->
manager.addStudent();
case 2 -> {
System.out.print("Enter Roll No: ");
int rollNo = Integer.parseInt(sc.nextLine());
try {
manager.displayStudent(rollNo);
} catch (StudentNotFoundException e) {
System.out.println(e.getMessage());
}
}
case 3 -> {
System.out.println("Exiting... Thank you!");
sc.close();
System.exit(0);
}
default ->
System.out.println("Invalid choice! Try again.\n");
}
} catch (NumberFormatException e) {
System.out.println("Error: Please enter a valid number!\n");
}
}
}
}

OUTPUT:



```
PROBLEMS 9   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   POSTMAN CONSOLE   SPELL CHECKER 1

==== Student Management System ====
1. Add Student
2. Display Student
3. Exit
Enter your choice: 1
Enter Roll No (Integer): 64
Enter Name: RAKESH
Enter Email: Rakesh64@gmail.com
Enter Course: Bca
Enter Marks: 98
Loading.....

Student added successfully!
Roll No: 64
Name: RAKESH
Email: Rakesh64@gmail.com
Course: Bca
Marks: 98.0
Grade: A
Program execution completed.

==== Student Management System ====
1. Add Student
2. Display Student
3. Exit
Enter your choice: 2
Enter Roll No: 64
Roll No: 64
Name: RAKESH
Email: Rakesh64@gmail.com
Course: Bca
Marks: 98.0
Grade: A
==== Student Management System ====
1. Add Student
2. Display Student
3. Exit
Enter your choice: 3
Exiting... Thank you!

d:\RAKESH\VSC\JAVA>
```

Explanation:

This Java program is a **Student Management System** designed to demonstrate *exception handling, multithreading,* and *wrapper classes*. It allows the user to add and display student details while ensuring that invalid inputs are handled safely. The program consists of several classes that work together to provide a complete, robust solution.

The StudentNotFoundException class is a **custom exception** that is used to handle situations where a student with a given roll number does not exist. The RecordActions interface defines two essential methods, addStudent() and displayStudent(int rollNo), which must be implemented by any class that manages student records. The Loader class implements the Runnable interface and is used to simulate a loading process using **multithreading**. When a student is added, a separate thread runs to print "Loading….." with a delay, making the system appear more interactive and realistic.

The main functionality resides in the StudentManager class, which implements the RecordActions interface. Inside it, there is an inner Student class that holds student details such as roll number, name, email, course, and marks. Here, **wrapper classes** like Integer and Double are used instead of primitive types to demonstrate **autoboxing**. The addStudent() method collects user input and validates it using **try-catch-finally** blocks. If the user enters invalid data, such as marks outside the range of 0–100 or empty fields, the program throws an appropriate exception and displays a clear error message. Once validation succeeds, a loader thread runs before saving the data to simulate a delay. All student data is stored in a HashMap using the roll number as a key for quick access.

The displayStudent() method retrieves a student's details from the map using their roll number. If the student is not found, it throws the custom StudentNotFoundException. The StudentManagementSystem class contains the main method, which presents a **menu-driven interface** that allows users to add students, display records, or exit the program. The menu continuously runs until the user chooses to exit, ensuring easy interaction.