Java Lab Assignment 3

A system for managing different types of vehicles in a rental service

Problem Statement

Enhance the **Student Management System** by implementing **exception handling** and **multithreading** to ensure safe execution and responsiveness. The system should handle invalid input (such as marks outside the valid range or empty fields) using **try-catch-finally** blocks and **custom exceptions** like StudentNotFoundException. Additionally, the system should simulate a loading process when adding or saving student data by using **multithreading**. The program should utilize **wrapper classes** (such as Integer, Double) for data conversion and **autoboxing** where applicable, providing a robust and responsive user interface for managing student records.

Objective:
Handle runtime exceptions and implement threading and wrapper classes for effective student data management.

Learning Outcomes

Upon completion of this assignment, the student will be able to:

1. Implement **try-catch-finally** blocks for handling exceptions.
2. Use **multithreading** to simulate delays and ensure responsive UI.
3. Work with **wrapper classes** (Integer, Double) for data conversions.

Class Hierarchy & Data Types

Class Hierarchy:

1. **StudentManager**: Implements RecordActions
2. **Loader**: Implements Runnable for simulating loading in multithreading.
3. **Custom Exception**: StudentNotFoundException

Data Types:

- Integer, Double: For handling numeric data and autoboxing.
- Thread: For multithreading to simulate loading.

Dr. Manish Kumar

Detailed Instructions

1. **Exception Handling**: Add validation for invalid input (marks, course, etc.) and missing fields.
2. **Multithreading**: Use Thread class for simulating a loading process during data operations.
3. **Wrapper Classes**: Use **autoboxing** to convert primitive types to wrapper types (e.g., int to Integer).

Expected Output

```
Enter Roll No (Integer): 102
Enter Name: Karan
Enter Email: karan@mail.com
Enter Course: BCA
Enter Marks: 77.5
Loading.....
Roll No: 102
Name: Karan
Email: karan@mail.com
Course: BCA
Marks: 77.5
Grade: B
----------------------
Program execution completed.
```

Guidelines to Students

1. **Use of Multithreading**: Implement a basic thread simulation for loading data.
2. **Wrapper Classes**: Ensure data conversion from primitive types to wrapper types.

Improvements/Adjustments

1. **Enhance Threading**: Implement additional tasks like database queries during multithreading.
2. **Advanced Exception Handling**: Handle more complex errors like NullPointerException.

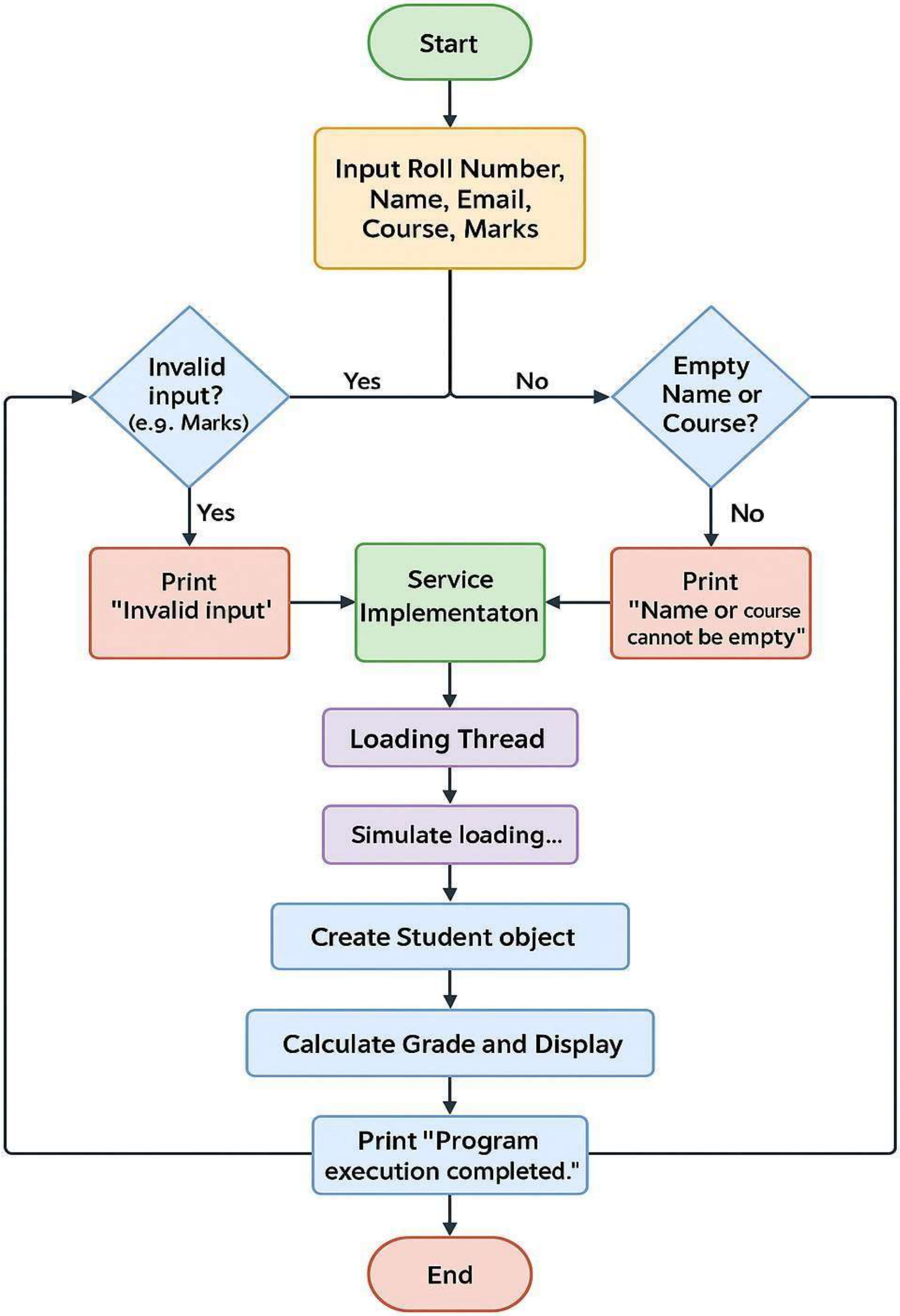
Submission Guidelines

1. Submit **Java source files** with all necessary exception handling.
2. Make sure your program validates inputs effectively.

Performance Metrics (Out of 10 Marks)

Criteria	**Marks**
Exception Handling Implementation	3
Multithreading and Responsiveness	2
Wrapper Classes and Data Validation	2
Code Quality and Structure	2
Documentation and Testing	1

Flow Chart:


Start
Input Roll Number, Name, Email, Course, Marks
Invalid input? (e.g. Marks)
Yes
No
Empty Name or Course?
Yes
No
Print "Invalid input'
Service Implementaton
Print "Name or course cannot be empty"
Loading Thread
Simulate loading...
Create Student object
Calculate Grade and Display
Print "Program execution completed."
End