Terminal Based Virtual Class Documentation.

Problem Statement

Imagine you are developing the backend for an EdTech platform that aims to host virtual classrooms. Your task is to create a terminal-based Virtual Classroom Manager that handles class scheduling, student attendance, and assignment submissions.

User Input

add_classroom classroom_name: to create class | eg: add_classroom Semester4

list_classroom: to list classes | eg: list_classroom

remove_classroom classroom_name: to remove class |eg: remove_classroom Semester4

add_student student_id classroom_name: to add student |eg: add_student 2159 Semester4

list_students classroom_name: to list students |eg: list_students Semester4

submit_assignment student_id classroom_name assignment_description: to submit assignment | eg: submit_assignment 2159 Semester4 quiz1

schedule_assignment classroom_name assignment_description: to add assignment| eg:schedule_assignment 2159 Semester4 quiz1

list_assignments classroom_name: to list assignments | eg: list_assignments Semester4

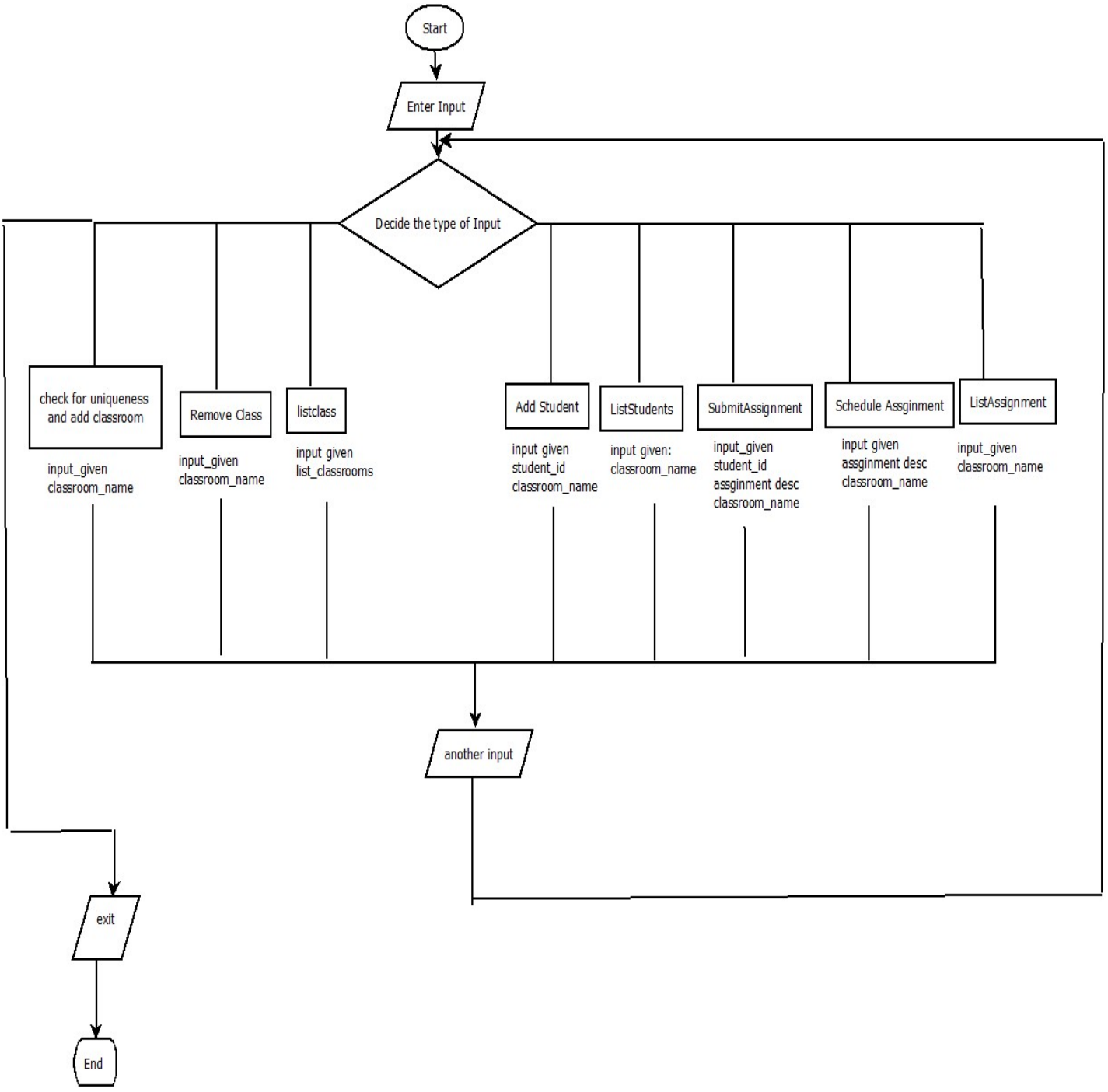
Output

```
------------------< com.mycompany:virtual_class_room >------------------
Building virtual_class_room 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.1.0:exec (default-cli) @ virtual_class_room ---
Enter any one of the below command or press exit to stop

add_classroom classroom_name: to create class
list_classroom: to list classes
remove_classroom classroom_name: to remove class
add_student student_id classroom_name: to add student
list_students classroom_name: to list students
submit_assignment student_id classroom_name: to submit assignment
schedule_assignment classroom_name assignment_description: to add assignment
list_assignments classroom_name: to list assignments


add_classroom cg
Classroom added: cg
Classrooms added so far: [cg]
Another input

Another input
add_student 2159 cg
Student added with ID: 2159 to classroom: cg
Another input
schedule_assignment cg quiz1
added assignment :  quiz1 to classroom: cg
Another input
submit_assignment 2159 cg quiz1
2159 submitted the assignment: quiz1
Another input

add_classroom phy
Classroom added: phy
Classrooms added so far: [cg, phy]
Another input
list_classroom
List of Classrooms:
cg
phy
Another input
remove_classroom phy
Classroom removed: phy
Remaining classrooms: [cg]
Another input
```

Functional Requirements

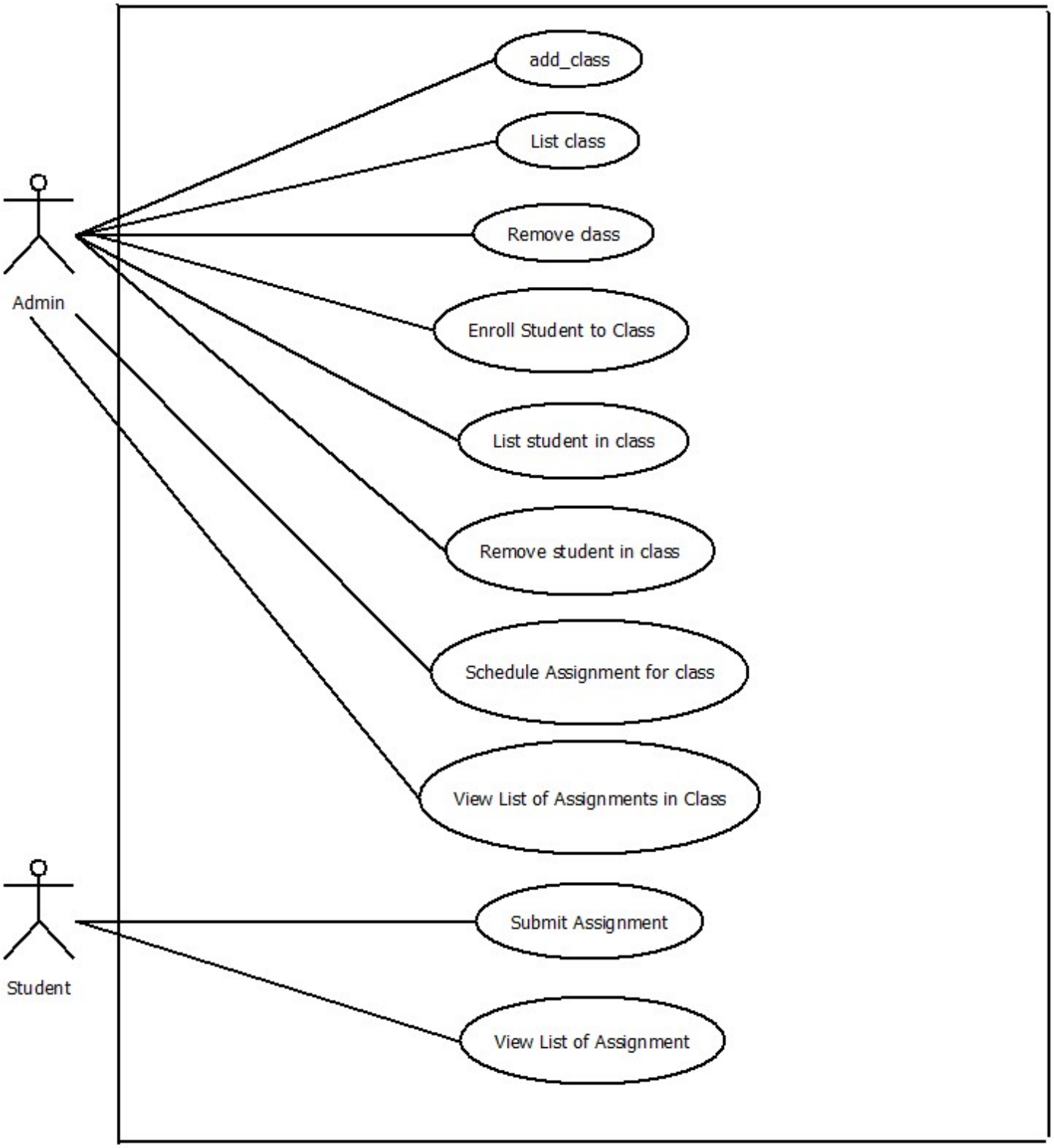
1. The system should be able to add virtual classes
2. The system should be able to list virtual classes
3. The system should be able to remove virtual classes
4. The system should allow the user to enroll the student to specific class
5. The system should be able to list the students in in classes
6. The system should allow the user to schedule assignment for each class
7. The system should list all the assignments scheduled for the class
8. The system should allow the student to submit the assignment
9. The system should display the list of assignments submitted in the class, with student ID as submission ID

Flow Chart Diagram



Flowchart will start with taking input and deciding the type of the class that is needed to be invoked. And then once the user enters exit. The loop will end.

Use Case Diagram



Student and Admin are two actors those, performs different actions at different levels.

Best Principles Followed:

1. **Modularity is achieved through segregation of packages.**
   - The entire project is divided into packages, named Classroom, Student, Assignment.
   - Each handles their own functionality.
   - The project hence achieves modularity and segregation of functions.

2. **The entire Program is built on the SOLID principles, segregating data, and business logic.**
   - In each package, there is one single class that maintains the data. And other classes will operate on data using business logic.
   - Each class has a single function to implement and single reason to change thereby adhering to the Single Responsibility Principles.
   - The business logic classes, that operate on data, are implemented using an Interface created in Classroom package. Since interface is used, each package is open to extension and each class within a package is closed for modification, thereby adhering to the Open/Closed Principles.
   - The interface methods are defined default so that the classes implementing the interface only implement the method that is need and not all, thereby adhering to the Interface Segregation Principles.

3. **Extensive logging mechanism is used for the purpose of user accessibility.**
   - Each action of a user is logged using the print statements and all the possible conditions are evaluated using conditions, thereby logging all the action of user.

4. **Exceptional Handling is used to check for all necessary conditions.**
   - The exceptions in the code, that the user may encounter is handled using try catch block, for example it is used in checking valid inputs.
   - Thereby the project handles most of the exception.