Faculty of Engineering and Information Technology
Computer Science Department
COMP242 (Project No. 3)

Project Title: Airport Check-in and Boarding System with Undo/Redo and Queue Management

Overview:

In this project, you will develop an **Airport Check-in and Boarding System** that manages flights and passengers. The system should support operations such as **check-in, boarding, cancellation**, as well as the ability to **undo and redo these actions**. It will **track boarded passengers** using a **LinkedList** and **passengers who canceled their flights using another LinkedList**. Additionally, the system will use **queues for regular and VIP passengers**, and maintain **undo/redo stacks for each flight,** with each stack storing **a history of operations**.

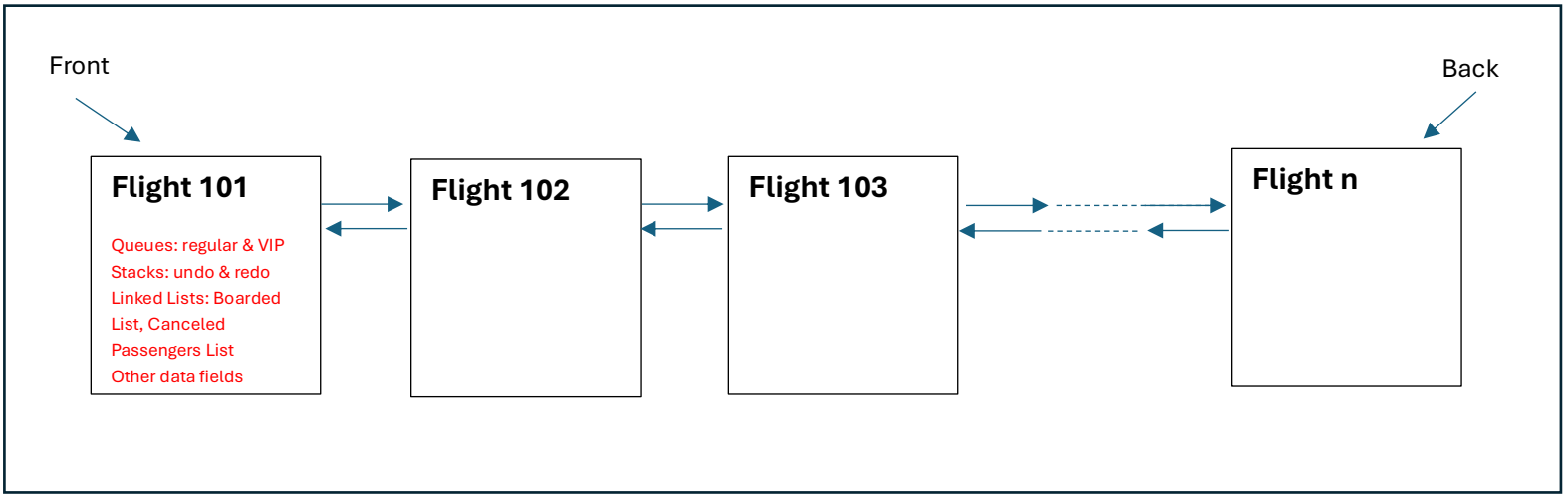

Figure: Abstract overview of how to look at the system

System Features:

1. **Passenger Management Menu**:
    - **Add Passenger**: Add a new passenger with a **unique ID**, name, flight ID, and status (e.g., Regular, VIP).
    - **Update Passenger**: Update a passenger's information (e.g., change status, name, etc).
    - **Remove Passenger**: Remove a passenger by **Passenger ID**.
    - **Search Passenger**: Search for a passenger by **Passenger ID**.
    - **Print All Passengers**: Display details of all passengers.
    - **Print Specific Passenger Info**: Display details of a specific passenger by **Passenger ID**.
2. **Flight Management Menu**:
    - **Add Flight**: Add a new flight with a **Flight ID**, destination, and status (Active/Inactive).
    - **Update Flight**: Update flight details (e.g., destination, etc).
    - **Remove Flight**: Remove a flight by **Flight ID**.
    - **Search Flight**: Search for flights by **Flight ID** or **destination**.
    - **Print All Flights**: Display details of all flights.
    - **Print Specific Flight Info**: Display details of a specific flight.
    - **Display all active flights and their destinations.**
    - **Display all inactive flights**.
    - **Integrating the Navigation Options in the JavaFX Application**: You can place the Next Flight and Previous Flight buttons anywhere you prefer. These will be used to present TableView displaying passengers for the selected flight.
3. **Operation Menu**:
    - **Check-in Passenger**: Check-in a passenger (either **Regular or VIP**) and add them to the appropriate queue.
    - **Board Passenger**: Move a passenger from the queue to the **boarded passengers list** (LinkedList).

- **Cancel Passenger**: Cancel a passenger's booking and remove them from the queue (store a reference to them in the Canceled Passengers List).
- **Undo**: Undo the last operation performed for a specific flight.
- **Redo**: Redo an undone operation for a specific flight.

4. **Log File Menu:**

• **Print Log File**: Add a menu option to import and display the contents of the log.txt file using a suitable GUI. This operation will allow users to view the log of all performed actions (check-in, boarding, cancellation, undo, redo, etc.) directly within the application.

• **Export Log File**: Add an option to export the log file, for example, saving the log to a text file (e.g., log_export.txt).

5. **Statistical Menu:**

- Display the total number of canceled VIP passengers.
- Display the total number of canceled regular passengers.
- Display the total number of VIP passengers currently in the queue.
- Display the total number of regular passengers currently in the queue.
- Display the total number of VIP passengers who have boarded.
- Display the total number of regular passengers who have boarded.

Flight and Passenger Management Structure:

1. **Flight Node:**
   - **FlightID**: Unique identifier for the flight.
   - **Destination**: Flight destination (e.g., "New York", "London").
   - **Status**: Active/Inactive.
   - **Queues**: Regular and VIP queues for passengers.
   - **Undo Stack**: A stack to track all operations for undoing them in **LIFO order**.
   - **Redo Stack**: A stack to track all undone operations, allowing for redo.
   - **Boarded Passengers List**: A **LinkedList** of passengers who have successfully boarded the flight (VIP passengers should be stored at the beginning of the list, while regular passengers should be stored at the end of the list)

Canceled Passengers List: A LinkedList of passengers who have canceled their flight

2. **Queues**:
    - **Regular Queue**: For standard passengers.
    - **VIP Queue**: For priority passengers.
3. **Undo/Redo Stack**:
    - **Undo Stack**: Stores all operations (e.g., check-in, boarding, cancel) in **LIFO** order for a specific flight.
    - **Redo Stack**: Stores reversed operations to allow for redoing them.
4. **LinkedList for Boarded Passengers**:
    - Use a **LinkedList** to track **all boarded passengers**.

File Formats:

Flight Data (flights.txt): This is just an example; please expand this table

FlightID, Destination, Status
101,New York,Active
102,London,Active

Passenger Data (passengers.txt): This is just an example; please expand this table

PassengerID,Name,FlightID,Status
1, Alice,101,Regular
2, Bob, 102,VIP

Log File (log.txt): (This is a global log file, not related to a specific flight. It contains all operations.

2024-11-08 12:00:00 | Check-in | Alice | 101 | Checked-in Alice for Flight 101
2024-11-08 12:10:00 | Check-in | Bob | 102 | Checked-in Bob for Flight 102
2024-11-08 12:20:00 | Boarding | Alice | 101 | Boarded Alice on Flight 101
2024-11-08 12:30:00 | Undo | Boarding | Alice | 101 | Undo boarding of Alice from Flight 101
2024-11-08 12:40:00 | Redo | Boarding | Alice | 101 | Redo boarding of Alice to Flight 101
2024-11-08 12:50:00 | Cancel | Alice | 101 | Canceled Alice from Flight 101
2024-11-08 13:00:00 | Undo | Cancel | Alice | 101 | Undo cancel of Alice from Flight 101

Example of Operations for Flight 101 (One Flight Example):

Here's an example of the operations performed for **Flight 101**, including the **queues, undo stack, redo stack, boarded passengers list, and the list of passengers who have canceled their flight**. (Keep in mind that you have different flights. e.g., Flight 101, Flight 102, Flight 103, etc)

Example Table of different Actions:

Step	Action	Regular Queue	VIP Queue	Undo Stack (Flight 101)	Redo Stack (Flight 101)	Boarded Passengers (LinkedList)	Canceled Passengers
1	Initial State	---	---	---	---	---	---
2	Check-in Alice (Regular, Flight 101)	Alice (Regular)	---	Check-in Alice (Flight 101)	---	---	---
3	Check-in Bob (VIP, Flight 101)	Alice (Regular)	Bob (VIP)	Check-in Bob (Flight 101), Check-in Alice (Flight 101)	---	---	---
4	Board Alice (Flight 101)	---	Bob (VIP)	Board Alice (Flight 101), Check-in Bob (Flight 101), Check-in Alice (Flight 101)	---	Alice (Boarded)	---
5	Undo Board Alice (Flight 101)	Alice (Regular)	Bob (VIP)	Check-in Bob (Flight 101), Check-in Alice (Flight 101)	Board Alice (Flight 101)	---	---
6	Redo Board Alice (Flight 101)	---	Bob (VIP)	Board Alice (Flight 101), Check-in Bob (Flight 101), Check-in Alice (Flight 101)	---	Alice (Boarded)	---
7	Cancel Alice (Flight 101)	---	Bob (VIP)	Cancel Alice (Flight 101), Board Alice (Flight 101), Check-in Bob (Flight 101), Check-in Alice (Flight 101)	---	---	Cancel Alice (Flight 101)
8	Undo Cancel Alice (Flight 101)	---	Bob (VIP)	Redo Board Alice (Flight 101), Check-in Bob (Flight 101), Check-in Alice (Flight 101)	Cancel Alice (Flight 101)	Alice (Boarded)	---

Implementation Guidelines:

1. **Flight Node (LinkedList)**:
    - Each flight is represented by a node in a **LinkedList** with fields for flight ID, destination, status, two **queues** (regular and VIP), **undo stack**, **redo stack**, a **LinkedList of boarded passengers, and a linked list of passengers who have canceled their flight**.

2. **Queues**:
    - Implement **two queues per flight**: Regular and VIP queues for handling passengers based on their status.

3. **Undo/Redo Stack**:
    - Implement an **undo stack** for each flight to track **all operations** in **LIFO** order.
    - Implement a **redo stack** for each flight to **store reversed operations, allowing for redo.**

4. **LinkedList for Boarded Passengers**:
    - Use a **LinkedList** to track all boarded passengers, updating the list when passengers board (VIP passengers should be inserted at the front of this list)

5. **A LinkedList of passengers who canceled their flights**
    - Use a L**inkedList** to track passengers who have canceled their flight, updating the list whenever a passenger cancels their booking

6. **JavaFX UI**:
    - Implement **JavaFX** interfaces for managing passengers and flights, with appropriate buttons and controls for actions such as check-in, boarding, cancellation, undoing actions, and any other actions you may need.
    - **For the date and time**, you need to use the Date and Time Picker from JavaFX.
    - Use the **file chooser** to import and export files.
    - **Do not use Scene Builder.**

Please note the Followings:

I. Your application should have all functionalities working properly.

II. There must be adequate documentation and comments in the code (e.g., functions, loops, etc.).

III. Your code should follow coding conventions (e.g., spacing, indentation, etc.) and guidelines (**Remember COMP2311**).

IV. This is an individual project. Disciplinary action will be taken against those who cheat. Additionally, **the use of AI tools for generating solutions or copying from websites is strictly prohibited**. Students found in violation of these policies will face severe consequences. It is crucial to ensure that all the work submitted is your own and adheres to the guidelines provided for this project.

V. Please submit your Java files (java) and corresponding test text files (txt) via the ITC by Monday, 2/ 12/2024, at 11:00 PM. **Late submissions will not be accepted under any circumstances.**

VI. **Please note that any updates made after submitting the code will receive a zero mark**

All the Best 😊