Phase 2

Flight Reservation System

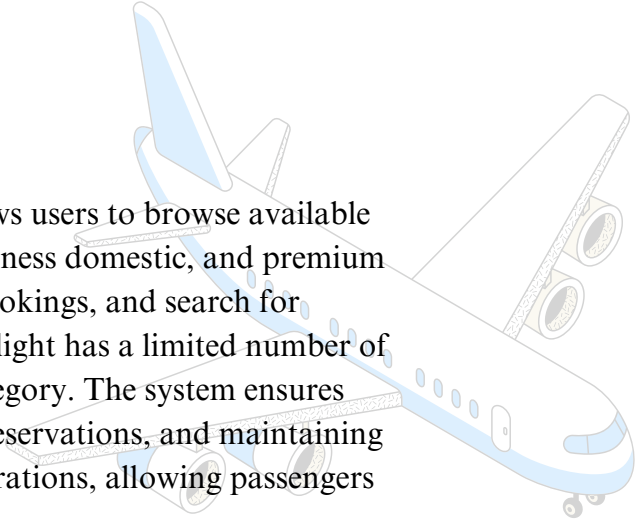
Group members:

Lama Almubarak (Leader)
Tala Alqahatni
Aljoharh Aldaej

The Division of work:

- **Lama**: Created First & Admin Frame (GUI) , fixed runtime/compilation errors in main and other classes
- **Tala**: Created user-defined exception class, Node class, and edited old methods in Airline and main classes to suit changes
- **Aljoharh**: Created Customer frame (GUI)

Introduction



The program is an airline flight reservation system that allows users to browse available flights based on type, including domestic, international, business domestic, and premium international flights. Users can make reservations, cancel bookings, and search for reservations using a passenger ID and flight number. Each flight has a limited number of available seats, and pricing may vary based on the flight category. The system ensures smooth booking operations by managing flights, handling reservations, and maintaining availability. It provides a structured approach to airline operations, allowing passengers to check in, book seats, and modify reservations efficiently.

Changes Made:

1. Custom Mobile Number Validation:
A user-defined exception class invalidMobileNum was introduced to ensure mobile numbers are exactly 10 digits long and start with '05'. This exception is triggered and handled within the Customer Frame, providing immediate feedback via a GUI prompt when invalid input is detected.

2. Robust Exception Handling:
NumberFormatException is caught and handled in both Admin Frame and Customer Frame, inside the actionPerformed() methods. This prevents the program from crashing when users enter non-numeric data in fields like flight number or seat count.
IOException and ClassNotFoundException are handled in the Airline class, during file input/output operations. These exceptions ensure stability when saving or loading data from the file system.

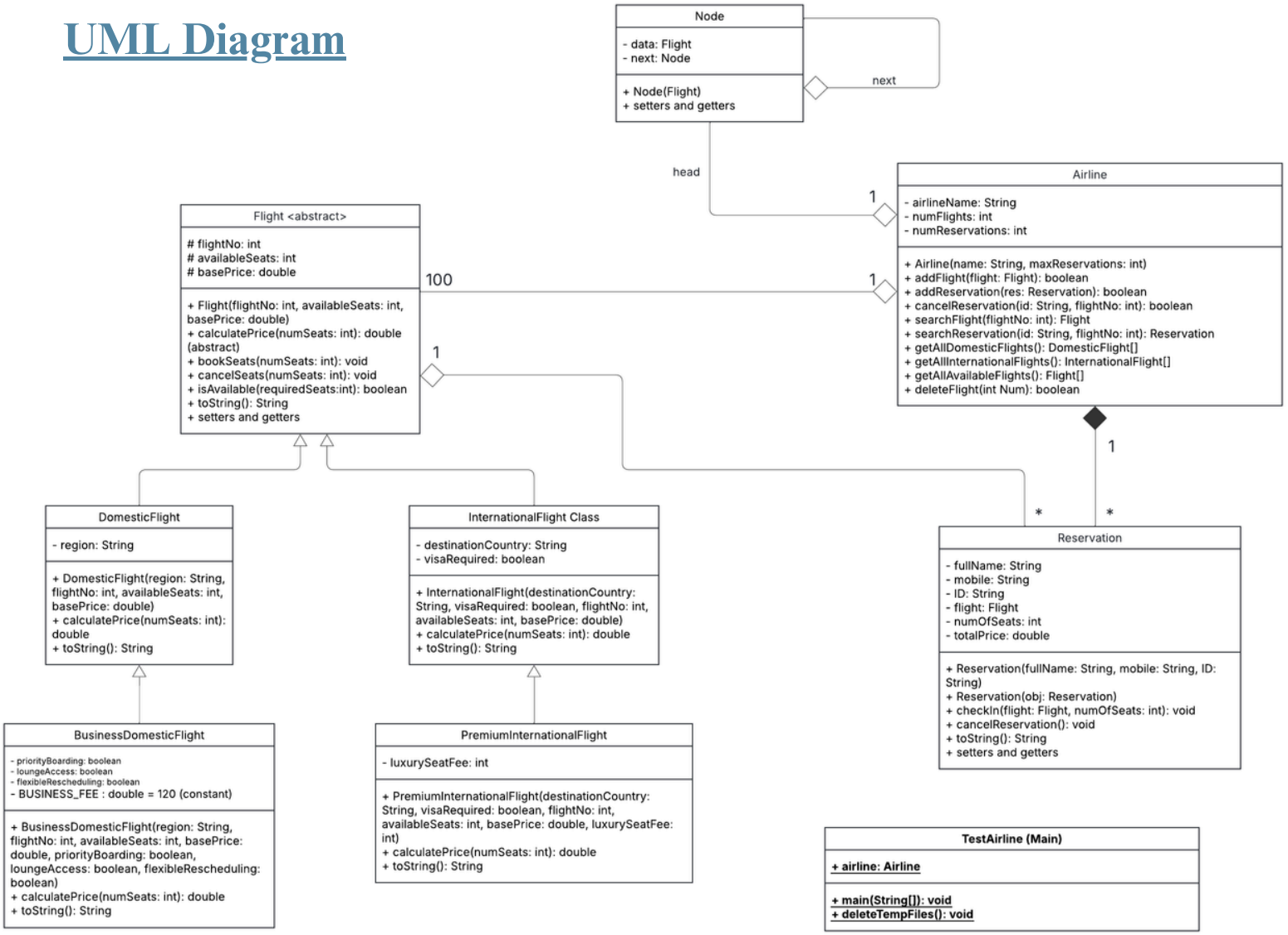
3. Linked List Data Structure:
Instead of using arrays, the program now uses a custom Node class to manage flights via a linked list. This allows more flexible memory management and simplifies the addition or removal of flights during runtime.

4. File Management:
Flight data is serialized and saved to a file named Flights.dat. The Airline class is responsible for reading and writing this data using ObjectOutputStream and ObjectInputStream. Proper exception handling ensures that missing files, data corruption, or class mismatches do not crash the program.

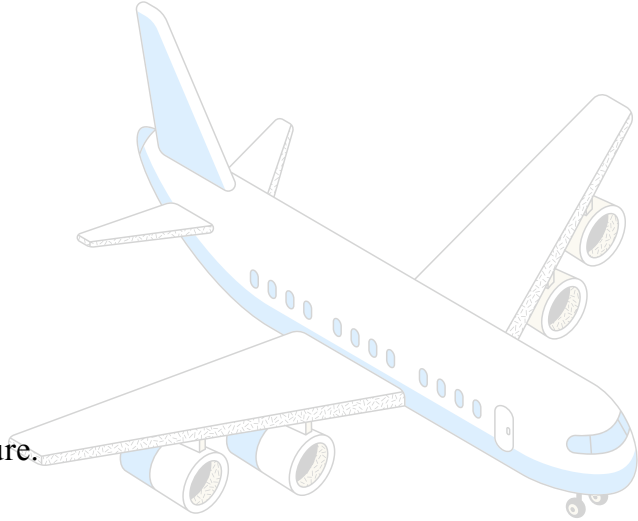
GUI Frames:

The program features three main frames for smooth navigation and role separation. The **First** (Main Menu) Frame appears first and allows users to choose between customer or admin access. The **Customer** Frame handles flight booking, cancellation, and reservation searches. The **Admin** Frame provides options to add, remove, and view flights, as well as manage saved data.

UML Diagram



Description of added classes :



Class Node (Implements Serializable):

Attributes:

• data: a Flight object that holds the data stored in this node.
• next: a Node reference pointing to the next node in the structure.

Methods:

• Node(obj: Flight): Constructor that initializes the node with a Flight object and sets the next reference to null.
• setNext(nextPtr: Node): Sets the reference to the next node.
• getNext(): Node: Returns the reference to the next node.
• setData(obj: Flight): Sets the Flight object stored in the node.
• getData(): Flight: Returns the Flight object stored in the node.

Class invalidMobileNum (Extends Exception):

Attributes:

Inherits attributes from the Exception class.

Methods:

• invalidMobileNum(s: String): Constructor that calls the superclass constructor with a custom error message describing the reason for the exception.

Interface InputOutputInterface:

Attributes:

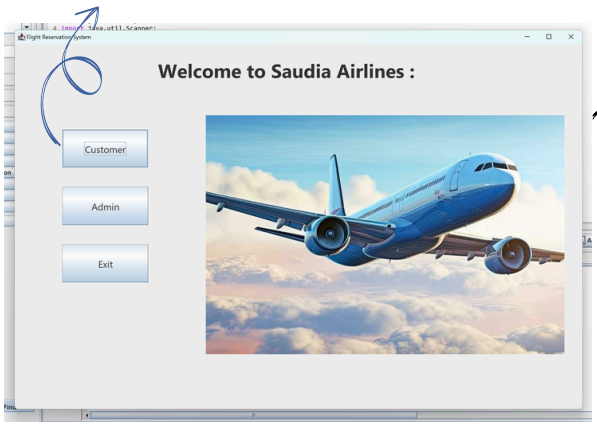
• fileOutput: a String constant that represents the filename where flight data is stored ("Flights.dat").

Methods:

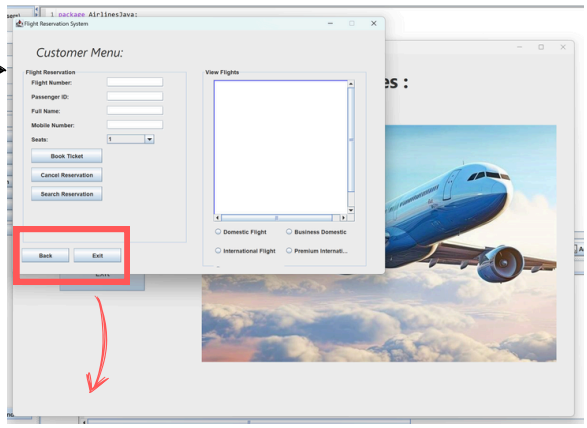
• saveAllInformation(): Abstract method for saving all necessary information, likely to the file specified by fileOutput.
• readAllData(): Abstract method for reading data from the file and restoring it into the program.

Screen shot of sample Run :

1. Main menu:
If you choose the Customer button



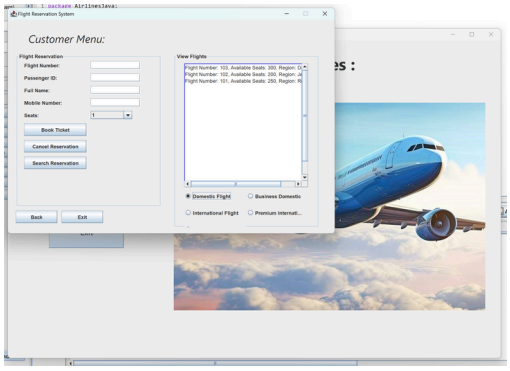
2. Customer screen:
A panel appears that allows you to: View available flights, Book a ticket, Cancel a reservation, Search for a reservation

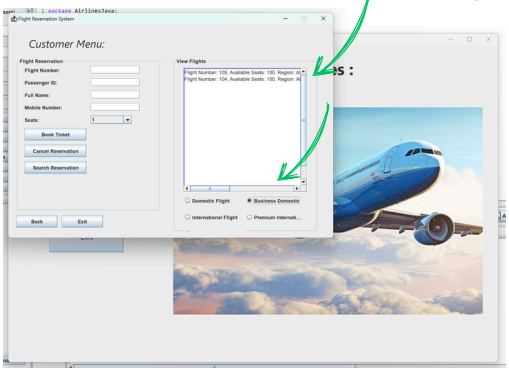


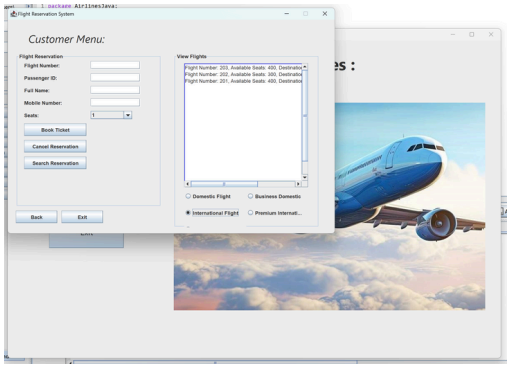
Additionally, you can choose to return to the previous screen or exit the program.

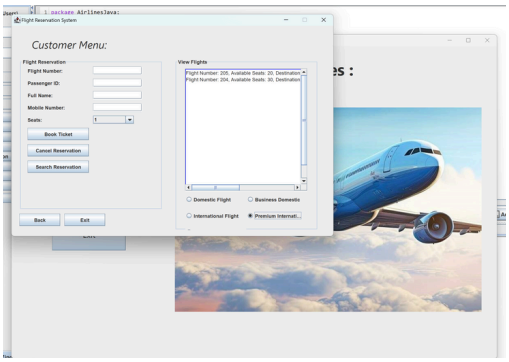
View Flights Options
depends on the flight type chosen

Scrollable panel that allows easy navigation in all directions: up, down, left, and right



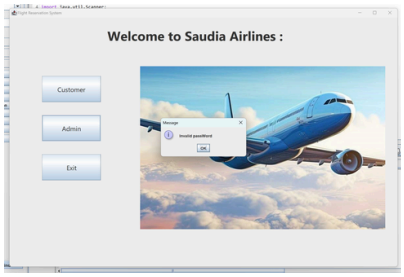






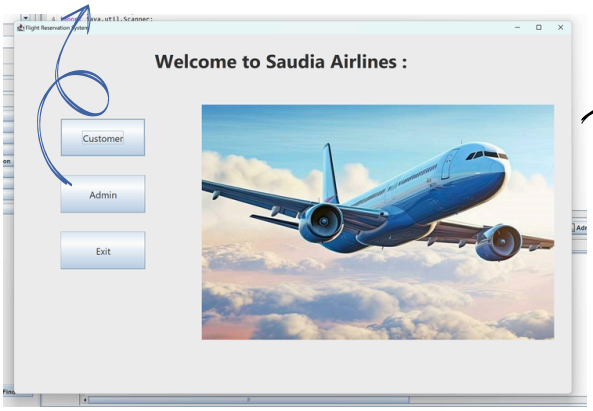
Screen shot of sample Run :

If the password is incorrect

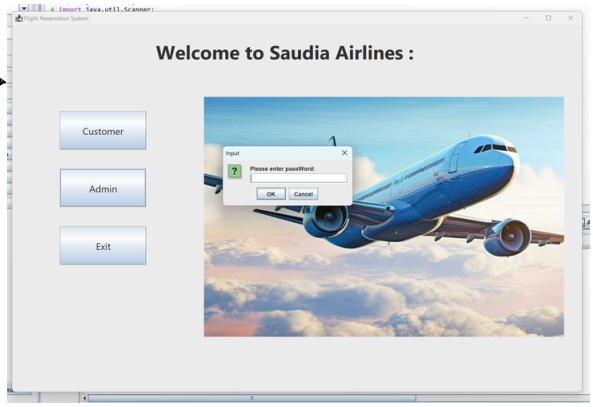


1. Main menu:

If you choose the Admin button

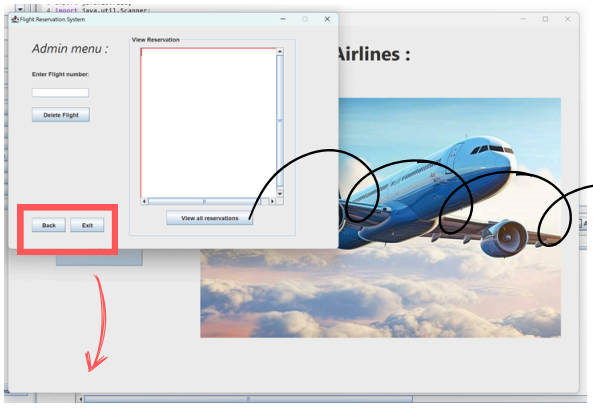


2. A dialog message will appear, asking you to enter a password to access the Admin menu



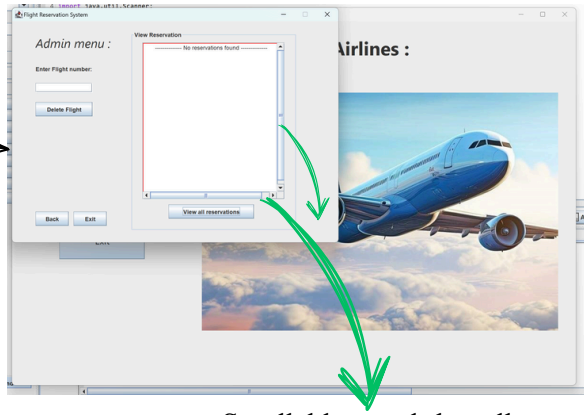
3. Admin screen:

A panel appears that allows you to: View available reservations and Delete a flight



Additionally, you can choose to return to the previous screen or exit the program.

If you choose “View all reservations”, it will display the list of reservations with its details, but if there are no reservations, a message will appear saying “No reservations found”

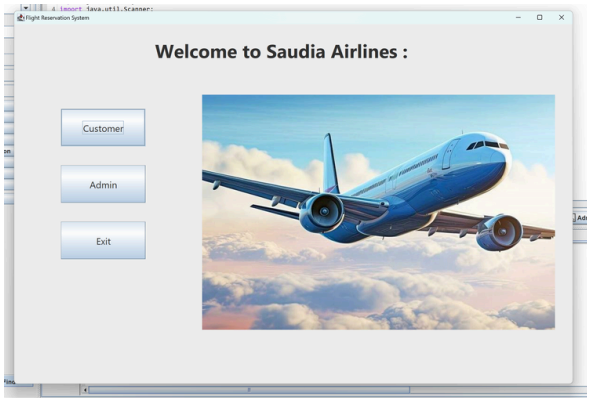


Scrollable panel that allows easy navigation in all directions: up, down, left, and right

Screen shot of sample Run :

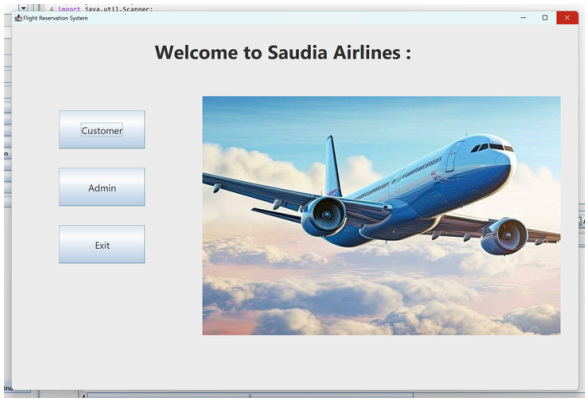
1. Main menu:

If you choose the Exit button

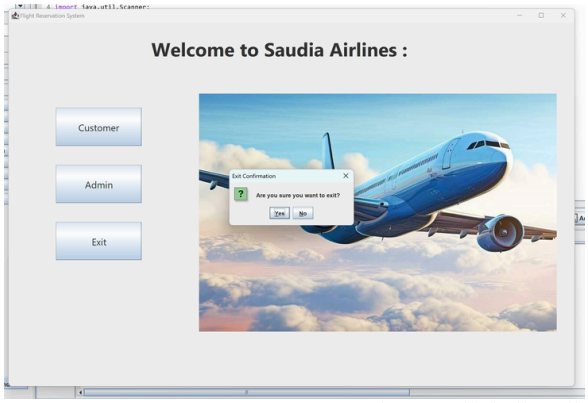


1. Main menu:

If you choose the X button



2. A confirmation dialog will appear when you try to exit, asking if you're sure you want to exit the program



If you click “yes”

The message “We hope you enjoyed the experience. Goodbye!” will show, and the program will close