Submitted to : Prof. Ali El Zaart

Faculty Of Science

Student 1 : Jad EL Mograbi 202303028

Student 2 : Moamen Hamdan 202302491

Fall 2023-2024

Table of Contents

Abstract:

The Airport Management System is a comprehensive project designed to address the intricacies involved in efficiently managing an airport's operations, including administration, flight scheduling, and passenger interactions. The project encompasses a robust data structure implementation, featuring classes for administrators, passengers, airplanes, flights, and priority queues, all meticulously designed and integrated to facilitate seamless airport management. The primary objective of this project is to develop a sophisticated system that not only provides essential functionalities but also prioritizes efficiency and user-friendly interactions. By amalgamating data structures, object-oriented programming, and user interfaces, the Airport Management System strives to be a valuable tool for airport personnel, administrators, and passengers alike.

Introduction:

The Airport Management System represents a significant advancement in airport operations, leveraging modern software engineering principles to create an intuitive and powerful platform. This introduction provides an overview of the project's key components and explains the problem statement that this system seeks to address.

Overview:

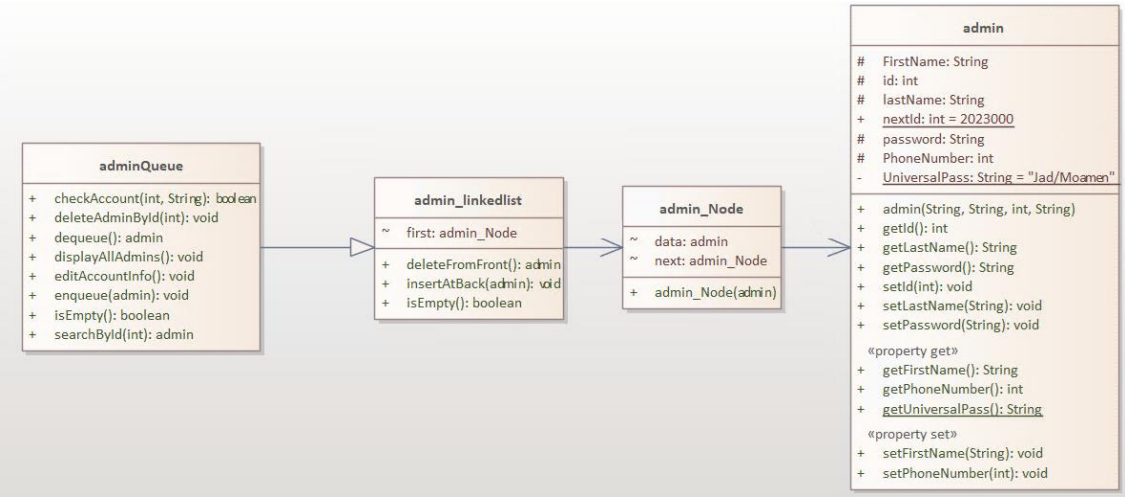
Managing the myriad aspects of an airport requires a multifaceted approach, from handling administrative tasks to ensuring smooth interactions with passengers and efficient flight scheduling. The Airport Management System aims to streamline these processes through an organized and interconnected set of classes and functionalities. With a focus on data structures, the system employs linked lists and priority queues to manage administrators, passengers, and flights effectively. The introduction of a priority queue for flight scheduling enhances the system's ability to prioritize and manage flights based on urgency and importance.

Problem Statement:

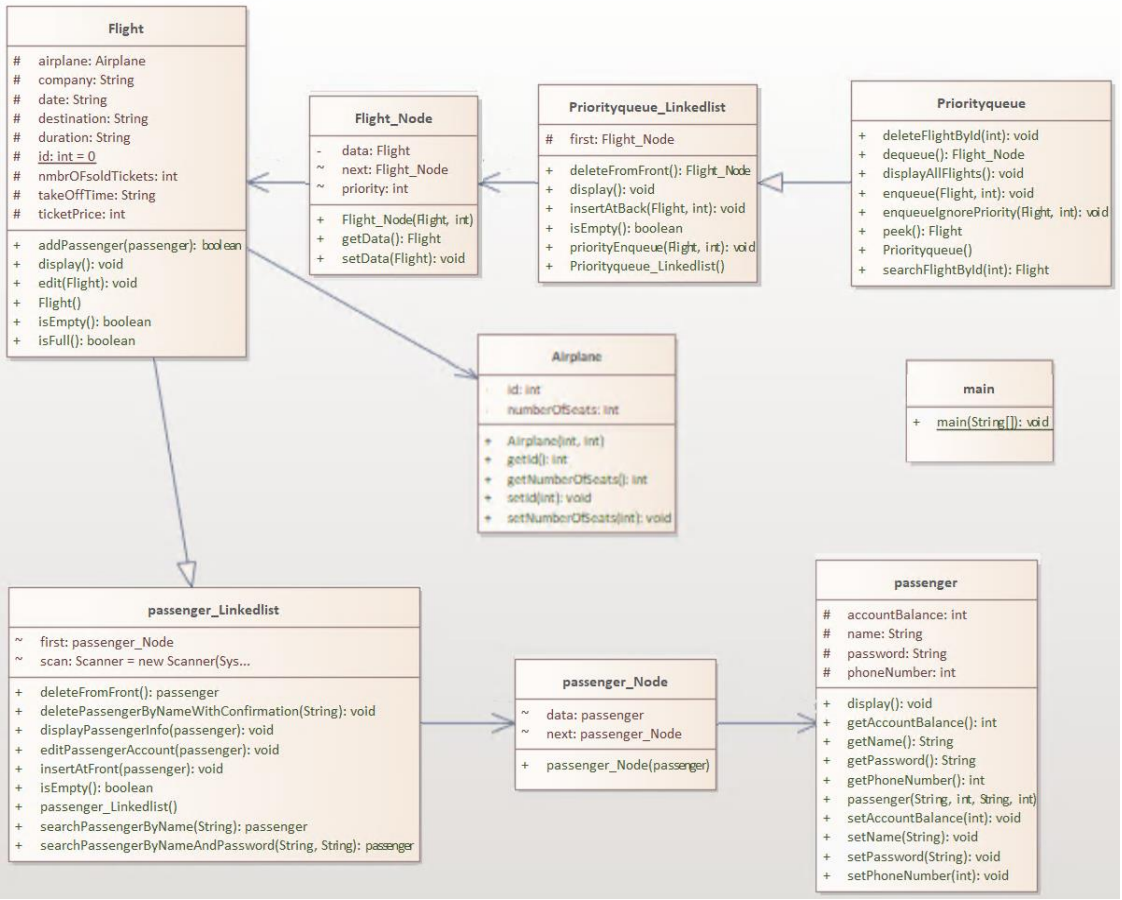
Traditional airport management systems often lack the versatility and efficiency required to handle the dynamic nature of airport operations. Challenges arise in managing passenger data, flight scheduling, and administrative tasks seamlessly. The need for a comprehensive solution led to the development of the Airport Management System. This project addresses the shortcomings of existing systems by integrating advanced data structures, ensuring optimal performance, and providing a user-friendly interface for administrators and passengers. The system aims to solve the complexities associated with airport management, offering a robust and efficient platform for diverse airport-related tasks.

UML :

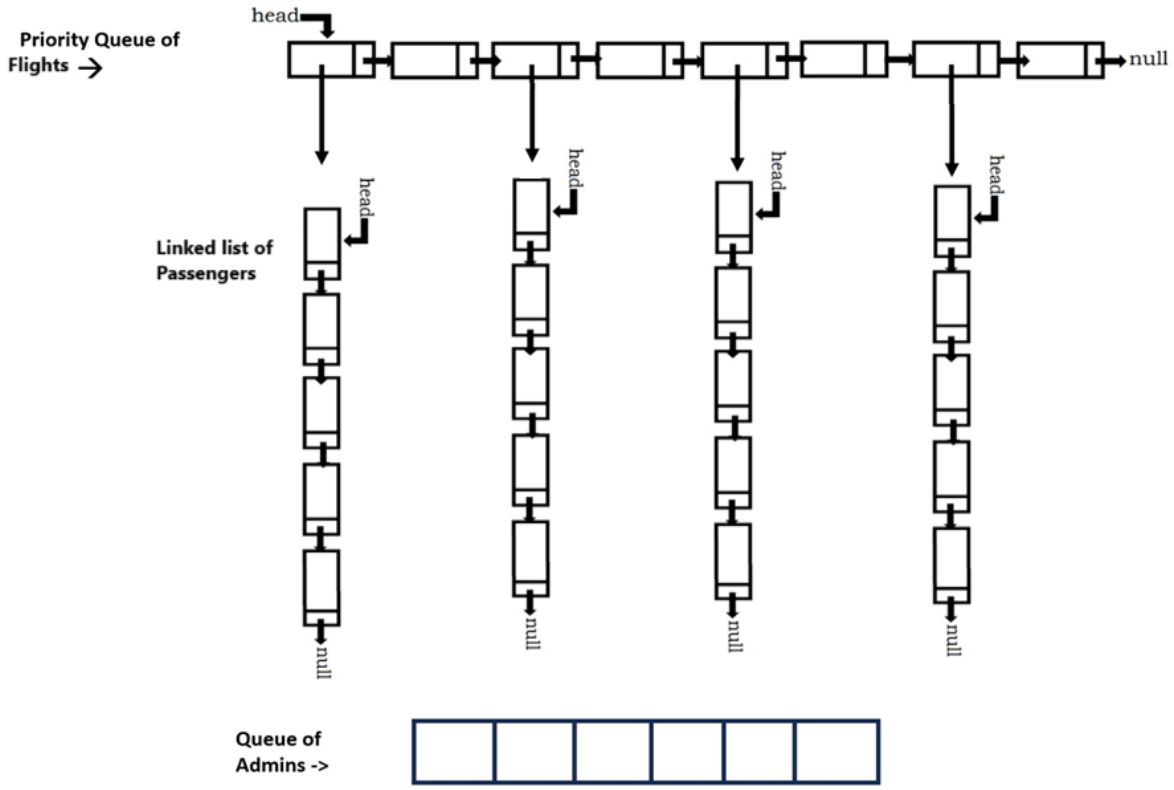
Admin queues



Airport/passengers



Data Strutures used:



Code Part:

Classes

1. admin Class:

The admin class represents an administrator within the airport system. It encapsulates essential attributes and methods for admin management.

Attributes:

id: An integer representing the unique identifier of the admin.

firstName: A string representing the first name of the admin.

lastName: A string representing the last name of the admin.

phoneNumber: An integer representing the phone number of the admin.

password: A string representing the password of the admin.

universalPass (static): A static string representing a universal password for airport access.

Methods:

getId(): Returns the unique identifier of the admin.

getFirstName(): Returns the first name of the admin.

getLastName(): Returns the last name of the admin.

getPhoneNumber(): Returns the phone number of the admin.

getPassword(): Returns the password of the admin.

getUniversalPass() (static): Returns the universal password for airport access.

editAccountInfo(): Allows the admin to edit their account information.

display(): Displays detailed information about the admin.

2. adminQueue Class:

The adminQueue class represents a queue specifically designed for admin management. It facilitates the addition, deletion, and display of admin accounts.

Attributes:

first: A reference to the first node in the linked list.

Methods:

isEmpty(): Checks if the admin queue is empty.

enqueue(admin a): Enqueues an admin into the queue.

dequeue(): Dequeues and returns the admin at the front of the queue.

checkAccount(int id, String password): Checks if an admin account matches the provided ID and password.

deleteAdminById(int adminId): Deletes an admin from the queue based on their ID.

searchById(int adminId): Searches for an admin in the queue based on their ID.

displayAllAdmins(): Displays detailed information about all admins in the queue.

3. passenger_Linkedlist Class:

The passenger_Linkedlist class represents a linked list specifically designed for passenger management. It provides functionalities for adding, searching, editing, and deleting passenger accounts.

Attributes:

first: A reference to the first node in the linked list.

scan: An instance of the Scanner class for user input.

Methods:

isEmpty(): Checks if the passenger linked list is empty.

insertAtFront(passenger data): Inserts a passenger at the front of the linked list.

deleteFromFront(): Deletes and returns the passenger from the front of the linked list.

searchPassengerByNameAndPassword(String name, String password): Searches for a passenger by name and password.

editPassengerAccount(passenger x): Edits the information of a passenger account.

searchPassengerByName(String name): Searches for a passenger by name.

deletePassengerByNameWithConfirmation(String name): Deletes a passenger by name with confirmation.

displayPassengerInfo(passenger x): Displays detailed information about a passenger.

4. passenger_Node Class:

The passenger_Node class is a node in a linked list specifically designed for passenger management. It holds a reference to a passenger object and a reference to the next node.

Attributes:

data: A reference to a passenger object.

next: A reference to the next passenger_Node in the linked list.

Methods:

getData(): Returns the passenger object associated with the node.

setData(passenger data): Sets the passenger object for the node.

5. passenger Class:

The passenger class represents a passenger within the airport system. It encapsulates essential attributes and methods for passenger management.

Attributes:

accountBalance: An integer representing the account balance of the passenger.

name: A string representing the name of the passenger.

password: A string representing the password of the passenger.

phoneNumber: An integer representing the phone number of the passenger.

Methods:

getAccountBalance(): Returns the account balance of the passenger.

getName(): Returns the name of the passenger.

getPassword(): Returns the password of the passenger.

getPhoneNumber(): Returns the phone number of the passenger.

setAccountBalance(int accountBalance): Sets the account balance of the passenger.

setName(String name): Sets the name of the passenger.

setPassword(String password): Sets the password of the passenger.

setPhoneNumber(int phoneNumber): Sets the phone number of the passenger.

display(): Displays detailed information about the passenger.

6. admin_Node Class:

The admin_Node class is a node in a linked list specifically designed for admin management. It holds a reference to an admin object and a reference to the next node.

Attributes:

data: A reference to an admin object.

next: A reference to the next admin_Node in the linked list.

Methods:

getData(): Returns the admin object associated with the node.

setData(admin data): Sets the admin object for the node.

7. admin_Linkedlist Class:

The admin_Linkedlist class represents a linked list specifically designed for admin management. It provides functionalities for adding, searching, editing, and deleting admin accounts.

Attributes:

first: A reference to the first node in the linked list.

Methods:

isEmpty(): Checks if the admin linked list is empty.

insertAtFront(admin data): Inserts an admin at the front of the linked list.

deleteFromFront(): Deletes and returns the admin from the front of the linked list.

searchAdminById(int adminId): Searches for an admin by ID.

editAdminAccount(admin x): Edits the information of an admin account.

deleteAdminById(int adminId): Deletes an admin by ID with confirmation.

displayAllAdmins(): Displays detailed information about all admins in the linked list.

1. Airplane Class:

The Airplane class represents an airplane within the airport system. It encapsulates essential attributes and methods for airplane management.

Attributes:

id: An integer representing the unique identifier of the airplane.

numberOfSeats: An integer indicating the total number of seats in the airplane.

Methods:

getId(): Returns the unique identifier of the airplane.

setId(int id): Sets the unique identifier of the airplane.

getNumberOfSeats(): Returns the total number of seats in the airplane.

setNumberOfSeats(int numberOfSeats): Sets the total number of seats in the airplane.

2. Flight_Node Class:

The Flight_Node class is a node in a linked list specifically designed for flight management. It holds a reference to a Flight object, a priority value, and a reference to the next node.

Attributes:

data: A reference to a Flight object.

priority: An integer representing the priority of the flight.

next: A reference to the next Flight_Node in the linked list.

Methods:

getData(): Returns the Flight object associated with the node.

setData(Flight data): Sets the Flight object for the node.

3. Flight Class (extends passenger_Linkedlist):

The Flight class represents a flight within the airport system. It extends the passenger_Linkedlist class to leverage linked list functionalities.

Attributes:

airplane: An instance of the Airplane class representing the airplane associated with the flight.

ticketPrice: An integer indicating the price of a ticket for the flight.

destination: A string representing the destination of the flight.

company: A string indicating the airline company operating the flight.

date: A string representing the date of the flight.

duration: A string indicating the duration of the flight.

nmbrOFsoldTickets: An integer representing the number of sold tickets for the flight.

takeOffTime: A string indicating the takeoff time of the flight.

Methods:

isEmpty(): Checks if the flight is empty.

isFull(): Checks if the flight is full.

edit(Flight x): Modifies the details of the flight based on another flight object.

addPassenger(passenger x): Adds a passenger to the flight.

display(): Displays detailed information about the flight.

4. Priorityqueue_Linkedlist Class:

The Priorityqueue_Linkedlist class represents a linked list implementation of a priority queue for managing flights.

Attributes:

first: A reference to the first node in the linked list.

Methods:

isEmpty(): Checks if the priority queue is empty.

insertAtBack(Flight v, int p): Inserts a flight with a priority at the end of the linked list.

priorityEnqueue(Flight a, int priority2): Enqueues a flight with priority based on the priority value.

deleteFromFront(): Removes and returns the first node in the linked list.

display(): Displays information about all flights in the priority queue.

5. Priorityqueue Class:

The Priorityqueue class extends the Priorityqueue_Linkedlist class and provides additional methods for enqueueing, dequeuing, searching, and displaying flights.

Methods:

enqueue(Flight flight, int priority): Enqueues a flight with a specified priority.

enqueueIgnorePriority(Flight flight, int priority): Enqueues a flight without considering priority.

peek(): Returns the flight at the front of the priority queue without dequeuing.

dequeue(): Dequeues and returns the flight at the front of the priority queue.

deleteFlightById(int flightId): Deletes a flight from the priority queue based on its ID.

searchFlightById(int flightId): Searches for a flight in the priority queue based on its ID.

displayAllFlights(): Displays detailed information about all flights in the priority queue.

Abstract: (About Main)

The main class is the core of the Airport Management System, responsible for initializing essential data structures, presenting a user-friendly interface, and coordinating interactions between administrators and users. This report comprehensively details the functionalities of the main class, exploring its initialization, the main menu, admin interactions, and user interactions.

Introduction:

The main class encapsulates the system's entry point, orchestrating the entire Airport Management System. It leverages a menu-driven approach, facilitating seamless navigation for administrators and users. The class interacts with critical components like the admin_Linkedlist, Priorityqueue, and various other classes to ensure a cohesive and efficient system.

Detailed Implementation:

Initialization:

The main class commences by initializing key data structures: the admin_Linkedlist and Priorityqueue. An initial admin is added to the system, ensuring a foundational user for authentication purposes.

```
admin_Linkedlist admins = new admin_Linkedlist();
Priorityqueue Flightqueue = new Priorityqueue();
admins.insertAtFront(new admin("Jad", "Elmoghraby", 76751060, "123"));
admins.insertAtFront(new admin("Moamen", "Hamdan", 71117842, "123"));
```

Main Menu:

The main menu is designed to provide users with clear choices: entering as an admin, entering as a user, or quitting the system. It utilizes a loop to continuously present the menu until the user chooses to exit.

```
int choice = 0;

while (choice != -1) {

  // Displaying main menu options

  // User input is then processed to determine the flow of the program

} Admin Menu:
```

If the user opts to enter as an admin, the program checks for an existing account. If the admin is already registered, authentication is required. Otherwise, a new admin account is created. The admin is then presented with an extensive menu of actions such as editing accounts, managing Flight Plans, and modifying the list of administrators.

```
private static void adminMenu(admin_Linkedlist admins, Priorityqueue Flightqueue) {

  // Admin authentication and menu options

  // Extensive switch-case structure for admin functionalities

} private static void adminMenu(admin_Linkedlist admins, Priorityqueue Flightqueue) {

  // Admin authentication and menu options

  // Extensive switch-case structure for admin functionalities

}
```

User Menu:

When the user selects to enter as a regular user, the program prompts for authentication or allows the user to create a new account. Upon successful authentication, the user is presented with a menu offering various actions, including managing their account, viewing available Flight Plans, and booking a flight.

```
private static void userMenu(admin_Linkedlist admins, Priorityqueue Flightqueue) {

  // User authentication and menu options

  // Extensive switch-case structure for user functionalities

}
```

Conclusion:

In conclusion, the main class serves as the linchpin of the Airport Management System. Its detailed implementation ensures a smooth and intuitive experience for administrators and users alike. By employing a modular approach and leveraging well-defined menu structures, the main class provides a robust foundation for the overall functionality of the Airport

Management System. The clarity and comprehensiveness of this implementation make it an essential component in the seamless operation of the system.

Code

Because of the extensive volume of code lines, we are unable to include it in the Word document.

But we found a way to put the code on GitHub so you will be able to see it
Code Link
We have a version with the code included that is around 70 pages long
if you want us to submit it at any given time just send me an email at
jne302@student.bau.edu.lb

Output

Passenger LinkedList

Insert:

```
Flight ID: 0
Number of available seats: 7
Ticket price: 50
Destination: Hawaii
Flight airline: MEA
Date : 12/12/2023
Duration: 6hrs
Number of sold tickets : 0
Takeoff Time : 5PM


Please enter the Flight ID of your desired flight
0
Do you want to buy a ticket (yes/no)
yes
```

Search:

```
1-Edit your account
2-delete your account
3-display your  account
4-display all the flight
5-book a flight
6-unbook a flight
7-Display flights by destination
8-Display flights i can afford
9-Display flights i bought
10-display Transaction
-1 Exit
9
Flight ID: 0
Number of available seats: 6
Ticket price: 50
Destination: Hawaii
Flight airline: MEA
Date : 12/12/2023
Duration: 6hrs
Number of sold tickets : 1
Takeoff Time : 5PM
```

Delete:

```
Please enter the Flight ID of the flight you want to refund
0
Do you want to refund the tiket for jad? (yes/no)
yes
Account refunded successfully.

1-Edit your account
2-delete your account
3-display your  account
4-display all the flight
5-book a flight
6-unbook a flight
7-Display flights by destination
8-Display flights i can afford
9-Display flights i bought
10-display Transaction
-1 Exit
```

Display:

```
1-edit account
2-display all the admins
3-delete admin by his Id
4-add Flight Plan
5-update Flight Plan
6-delete Flight Plan
7-display Flight Plans
8-add new admin
9-Search for an admin by his id
10-Display Passengers in a flight
-1 to exit
10
```

```
Please enter the Flight ID of the flight to display passengers
0
Passenger Information:
Name: moamen
Phone Number: 71117741
Account Balance: 1950
Passenger Information:
Name: jad
Phone Number: 77777777
Account Balance: 1950
```

Flight Queue

Insert:

```
1-edit account
2-display all the admins
3-delete admin by his Id
4-add Flight Plan
5-update Flight Plan
6-delete Flight Plan
7-display Flight Plans
8-add new admin
9-Search for an admin by his id
10-Display Passengers in a flight
-1 to exit
4
Enter the Flight Details
Enter the number of seats:
2
Enter the destiation of the flight:(word)
France
Enter the Airline company:(word)
MEA
Enter the date of the flight:(word)
12/9/2024
Enter the duration of the flight:(word)
4hrs
Enter the takeoff time of the flight:(word)
5pm
Enter the seat price of the flight:
30
```

Search:

//To find the flight we use searchflightbyid

```
Flight ID: 0
Number of available seats: 7
Ticket price: 50
Destination: Hawaii
Flight airline: MEA
Date : 12/12/2023
Duration: 6hrs
Number of sold tickets : 0
Takeoff Time : 5PM


Please enter the Flight ID of your desired flight
0
Do you want to buy a ticket (yes/no)
yes
```

Delete:

```
1-edit account
2-display all the admins
3-delete admin by his Id
4-add Flight Plan
5-update Flight Plan
6-delete Flight Plan
7-display Flight Plans
8-add new admin
9-Search for an admin by his id
10-Display Passengers in a flight
-1 to exit
6
Enter the flight id to delete
WARNING: This action cannot be undone!
0
```

Display:

```
7
Flight ID: 3
Number of available seats: 2
Ticket price: 30
Destination: France
Flight airline: MEA
Date : 12/9/2024
Duration: 4hrs
Number of sold tickets : 0
Takeoff Time : 5pm

Flight ID: 1
Number of available seats: 3
Ticket price: 40
Destination: France
Flight airline: AirFrance
Date : 13/12/2023
Duration: 4hrs
Number of sold tickets : 0
Takeoff Time : 8PM

Flight ID: 2
Number of available seats: 5
Ticket price: 30
Destination: UAE
Flight airline: Emirates
Date : 12/14/2023
Duration: 6hrs
Number of sold tickets : 0
Takeoff Time : 9AM
```