Object-Oriented-Programming Final Project: Airline Reservation System

Lecturer:
Jude Joseph Lamug Martinez

Arranged by:
Edelyne Keisha - L2BC

Student ID:
2602169850

Object Oriented Programming
Computer Science Faculty
Binus International University 2022/2026

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per
lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which
automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating, and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity, and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Edelyne Keisha

Table of Contents

1.1. Introduction

As the semester comes into a close, students of Binus University International are expected to showcase everything that they've learned in the semester by making something of their own, using all the lessons and materials given to them. For this final project, students are encouraged to go beyond what was taught in class and apply everything that's taught in Object Oriented Programming.

The project of this report is a simulation of an Airline Reservation Management System, where two kinds of users can login and use it; A passenger, who in this case will use it to book flights and see all available flights, and an admin, who can oversee all the flights and the passengers in it. This project makes use of Java as an object oriented programming language as a whole, and as such it utilizes concepts such as Inheritance between classes, Abstract Classes, and Interfaces.

1.2. Inspirations

When I got the news for the final project, I searched long and hard for ideas I wanted to do. I didn't want it to be too simple, as it is a final semester project, but I didn't want it to be too difficult either, as I fear I do not have the time to finish it. Either way, my ideas for this project was something that had to strike a balance between complexity and simplicity.

After thinking through many ideas, I finally came up with this idea. I knew I wanted to make some sort of a management system. However I didn't know which one yet, but in the end, I made an airline system because I was reminded of the recent time where I went on a trip overseas, and I thought about how airports kept track of all the schedules and their passengers.

However, because I didn't have time to implement a GUI for the project, it ended up being on a command line. And so I used ASCII art to make it look better and less empty.

2. Project Specifications

2.1. Program Flow

The program allows two kinds of users to login and use it: Admin and Passenger.

For Passenger: The Passenger needs to first register an account. After that, once they do, their data will be saved in a txt file called PassengerData.txt. After they've logged in, they are given options on what they want to do, such as booking a flight, updating their data like their name and address, deleting their Passenger account, displaying all the available flight schedules, canceling a flight that they've booked, and display all the flights that they've booked.

For Admin: The Admin will also need to register their account. Once they're done and they have logged in, they also have their own options on what to do. They can add a Passenger by themselves, and that data would also be saved in the txt file. They can also search for a Passenger by looking for their ID. They can manually update a Passenger's data, delete a Passenger, display all the registered Passengers, display all the flights registered by a single passenger, display all the Passengers in a flight, or even delete a flight from the list.

2.2. Program Libraries

java.io: this library is used to add all the Passenger data into a txt file. It makes a new txt file, and then inserts all the data inputted by the user into that file.
java.time: this library is used for all the flight times. It's used in the flight schedule where it formats the time.
java.util: this library is used for standard things. It provides utility, such as data structures like a List or Arraylist, or a function like Random, which in this program is used to randomize the flight schedules

2.3. Program Files

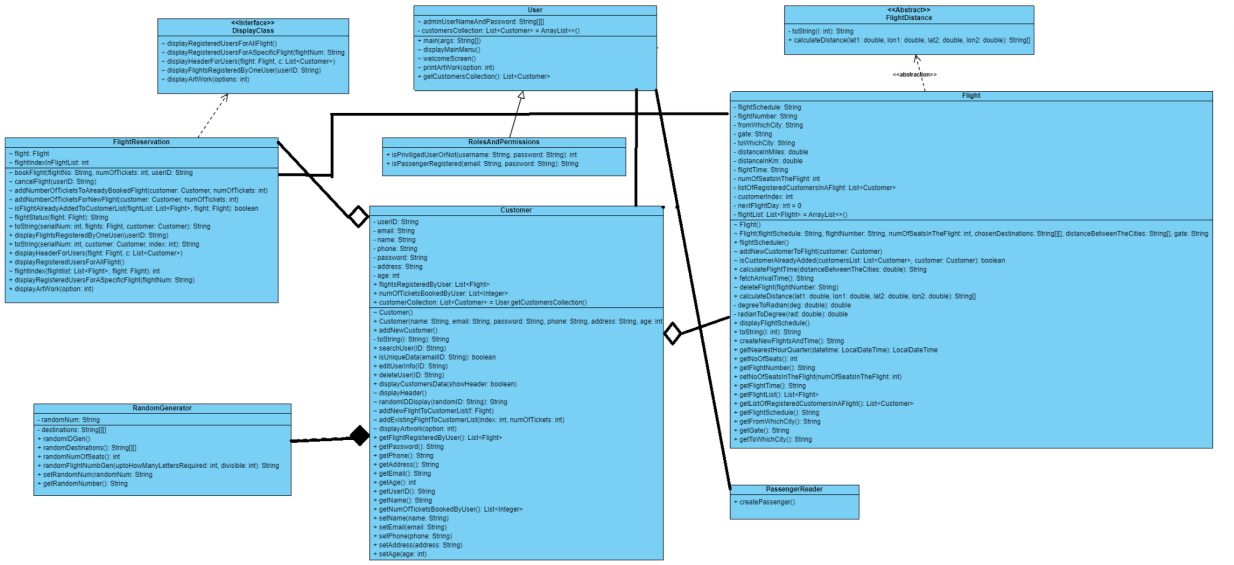
Here are all the classes in the Program:

User: A class that runs all the methods. Essentially it's the class that can be run.

Customer: A class with all the methods for the Passengers, including adding, searching, deleting, checking their data, editing, and displaying all Passengers.
RolesAndPermissions: A class to check whether or not the Admin and/or Passenger is registered in the data.
RandomGenerator: A class with all the methods used to randomize things. It can randomize a Passenger ID, number of seats in a flight, random flight number, and random destinations between two cities.
Flight: A class used to manage all the flight data, such checking the schedule and calculating the time needed.
FlightDistance: An abstract class containing the abstract method toString and the one to calculate distance between cities.
FlightReservation: A class that allows the user (whether Admin or Passenger) to book, cancel, and check flight status.
DisplayClass: An interface containing empty methods to display things, such as displaying registered users in a flight, or displaying the ASCII art in the command line.
PassengerReader: A class used to add Passenger data to a txt file.

2.4. Class Diagram

Here are all the hierarchy between the classes:



Flight and FlightDistance: The relationship between these two classes is Inheritance, in which Flight extends the abstract class FlightDistance. It is used so that it can reuse the toString method and add new functionality for the calculateDistance method that's defined in the FlightDistance class.

FlightReservation and DisplayClass: FlightReservation implements the DisplayClass interface, which is responsible for all the methods used to display items. The DisplayClass interface is sort of like a blueprint for the FlightReservation class.

RolesAndPermissions and User: The relationship between these two classes is Inheritance, in which RolesAndPermissions extends the class User. It does this so that the User class can reuse the methods in the RolesAndPermissions class to check whether or not a user has been registered or not.

Flight and Customer: The relationship between these two classes is Aggregation, this means that there is an instance of the Flight class in the Customer class. However, the Flight class can function on its own.

Flight and FlightReservation: The relationship between these two classes is Association, there is an instance of the Flight class in the FlightReservation class.

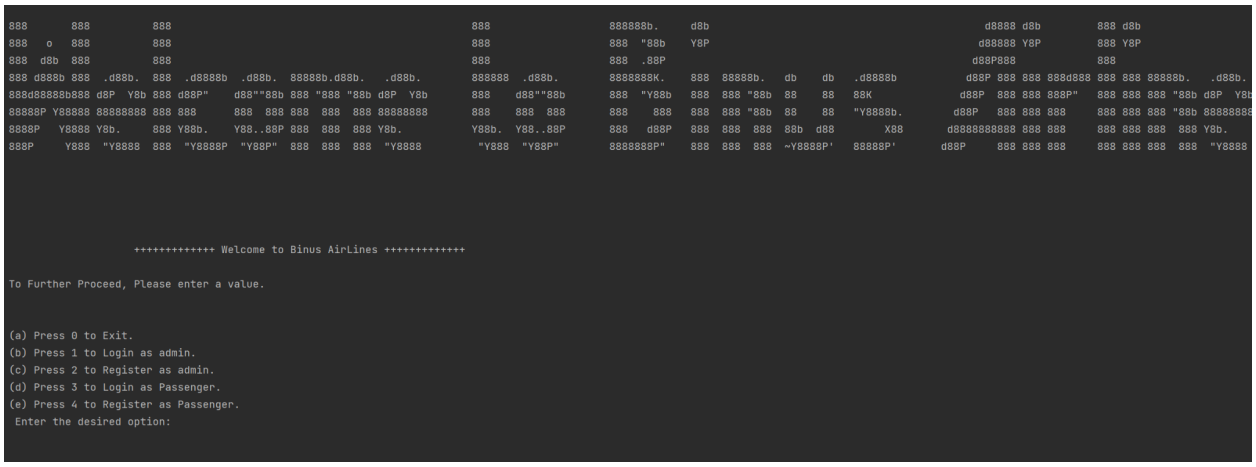
Customer and RandomGenerator: The relationship between these two classes is Composition, in which the Customer class wouldn't be able to run without the RandomGenerator class because that class is what gives the Customers their ID which is generated randomly from the RandomGenerator class.

Customer and User: The relationship between these two classes is Association, in which there is a connection between the two classes, because the Customer can also be a User, and vice versa.

FlightReservation and Customer: The relationship between these two classes is Aggregation, there is an instance of FlightReservation in the Customer class.

3. Program Flow

Beginning Screen



Passenger register

```
 .o88b. db    db .d8888. d888888b  .d88b.  .88b  d88. d88888b d8888b.      .d8888. d888888b  d888b  d8b   db db    db d8888b.
d8P  Y8 88    88 88'  YP `~~88~~' .8P  Y8. 88'YbdP`88 88'     88  `8D      88'  YP   `88'   88' Y8b 888o  88 88    88 88  `8D
8P      88    88 `8bo.      88    88    88 88  88  88 88ooooo 88oobY'      `8bo.      88    88      88V8o 88 88    88 88oodD'
8b      88    88   `Y8b.    88    88    88 88  88  88 88~~~~~ 88`8b          `Y8b.    88    88  ooo 88 V8o88 88    88 88~~~
Y8b  d8 88b  d88 db   8D    88    `8b  d8' 88  88  88 88.     88 `88.      db   8D   .88.   88. ~8~ 88  V888 88b  d88 88
 `Y88P' ~Y8888P' `8888Y'    YP     `Y88P'  YP  YP  YP Y88888P 88   YD      `8888Y' Y888888P  Y888P  VP   V8P ~Y8888P' 88




                                                                 +++++++++++++++ Welcome to the Customer Registration Portal +++++++++++++++
Enter your name : edelyne
Enter your email address : email
Enter your Password : pass
Enter your Phone number : 2434545
Enter your address : senayan
Enter your age : 24
```

Passenger Login

```
 .o88b. db    db .d8888. d888888b  .d88b.  .88b  d88. d88888b d8888b.      db       .d88b.   d888b  d888888b d8b   db
d8P  Y8 88    88 88'  YP `~~88~~' .8P  Y8. 88'YbdP`88 88'     88  `8D      88      .8P  Y8. 88' Y8b   `88'   888o  88
8P      88    88 `8bo.      88    88    88 88  88  88 88ooooo 88oobY'      88      88    88 88         88    88V8o 88
8b      88    88   `Y8b.    88    88    88 88  88  88 88~~~~~ 88`8b        88      88    88 88  ooo    88    88 V8o88
Y8b  d8 88b  d88 db   8D    88    `8b  d8' 88  88  88 88.     88 `88.      88booo. `8b  d8' 88. ~8~   .88.   88  V888
 `Y88P' ~Y8888P' `8888Y'    YP     `Y88P'  YP  YP  YP Y88888P 88   YD     Y88888P  `Y88P'   Y888P  Y888888P VP   V8P




Enter the Email to Login :  email
Enter the Password :  pass


                  Logged in Successfully as "email"..... For further Proceedings, enter a value from below....

                                                  +++++++++ 3rd Layer Menu +++++++++                                                  Logged in as "email"
                                    (a) Enter 1 to Book a flight....
                                    (b) Enter 2 to update your Data....
                                    (c) Enter 3 to delete your account....
                                    (d) Enter 4 to Display Flight Schedule....
                                    (e) Enter 5 to Cancel a Flight....
                                    (f) Enter 6 to Display all flights registered by "email"....
                                    (g) Enter 0 to Go back to the Main Menu/Logout....
Enter the desired Choice :
```

Passenger: Book a Flight

```
d8888b.  .d88b.   .d88b.  db   dD      d88888b db      d888888b  d888b  db   db d888888b
88  `8D .8P  Y8. .8P  Y8. 88 ,8P'      88'     88        `88'   88' Y8b 88   88 `~~88~~'
88oooY' 88    88 88    88 88,8P        88ooo   88         88    88      88ooo88    88
88~~~b. 88    88 88    88 88`8b        88~~~   88         88    88  ooo 88~~~88    88
88   8D `8b  d8' `8b  d8' 88 `88.      88      88booo.   .88.   88. ~8~ 88   88    88
Y8888P'  `Y88P'   `Y88P'  YP   YD      YP      Y88888P Y888888P  Y888P  YP   YP    YP
```

Num	FLIGHT SCHEDULE	FLIGHT NO	No. of Seats	FROM =====>	====>> TO	ARRIVAL TIME	FLIGHT TIME	GATE	DISTANCE(MILES/KM
1	Thursday, 22 June 2023, 03:45 AM	GW-409	135	Gilgit Baltistan	Beijing	Thu, 22-06-2023 08:30 AM	04:45 Hrs	J-9	1995.69 / 3698.46
2	Sunday, 25 June 2023, 06:45 AM	DL-348	415	Pune	Baghdad	Sun, 25-06-2023 11:35 AM	04:50 Hrs	W-6	1818.09 / 3369.33
3	Thursday, 29 June 2023, 10:45 AM	KV-401	278	Sydney	Johannesburg	Fri, 30-06-2023 04:35 AM	17:50 Hrs	P-7	7846.91 / 14542.1
4	Tuesday, 04 July 2023, 15:45 PM	YK-200	372	Kuwait	Beijing	Tue, 04-07-2023 23:35 PM	07:50 Hrs	C-7	3366.42 / 6238.74
5	Tuesday, 04 July 2023, 16:00 PM	JA-334	346	Beijing	Berlin	Wed, 05-07-2023 01:20 AM	09:20 Hrs	T-3	3971.38 / 7359.89
6	Tuesday, 04 July 2023, 16:00 PM	PS-151	173	Riyadh	New Taipei City	Wed, 05-07-2023 01:30 AM	09:30 Hrs	F-12	4010.2 / 7431.82
7	Wednesday, 05 July 2023, 17:00 PM	IC-383	101	Istanbul	Rio de Janeiro	Thu, 06-07-2023 06:50 AM	13:50 Hrs	S-12	6055.44 / 11222.1
8	Thursday, 06 July 2023, 18:00 PM	KM-259	269	Sydney	Madrid	Fri, 07-07-2023 15:00 PM	21:00 Hrs	Z-10	9535.8 / 17672.0
9	Thursday, 06 July 2023, 18:15 PM	IK-254	112	Tokyo	Beijing	Thu, 06-07-2023 21:05 PM	02:50 Hrs	K-7	1127.64 / 2089.78
10	Tuesday, 11 July 2023, 22:30 PM	GJ-235	191	Lagos	Islamabad	Wed, 12-07-2023 02:40 AM	04:10 Hrs	D-14	1837.04 / 3404.46
11	Saturday, 15 July 2023, 02:00 AM	MS-280	465	Gilgit Baltistan	Santiago	Sat, 15-07-2023 22:00 PM	20:00 Hrs	O-13	9080.79 / 16828.7
12	Saturday, 15 July 2023, 01:30 AM	TJ-135	397	Stockholm	New Taipei City	Sat, 15-07-2023 11:30 AM	10:00 Hrs	P-10	4503.51 / 8346.04
13	Friday, 21 July 2023, 07:45 AM	YL-326	359	New Taipei City	Shenzhen	Fri, 21-07-2023 09:25 AM	01:40 Hrs	N-8	449.12 / 832.33
14	Thursday, 27 July 2023, 14:00 PM	HQ-393	344	Dhaka	Bangkok	Thu, 27-07-2023 16:25 PM	02:25 Hrs	X-11	829.53 / 1537.32
15	Monday, 31 July 2023, 18:00 PM	YT-346	222	Pune	Melbourne	Tue, 01-08-2023 05:50 AM	11:50 Hrs	F-4	5226.73 / 9686.33

```
Enter the desired flight number to book :  yt-346
Enter the Number of tickets for yt-346 flight :   6

                                        You've booked 6 tickets for Flight "YT-346"...
```

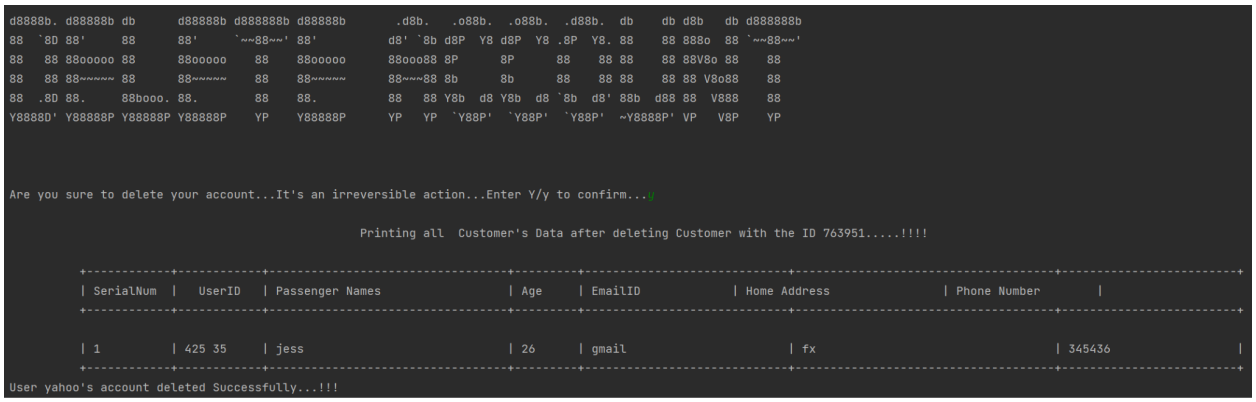
Passenger: Update your Data

```
d88888b d8888b. d888888b d888888b      d888888b d8b   db d88888b  .d88b.
88'     88  `8D   `88'   `~~88~~'        `88'   888o  88 88'     .8P  Y8.
88ooooo 88   88    88       88            88    88V8o 88 88ooo   88    88
88~~~~~ 88   88    88       88            88    88 V8o88 88~~~   88    88
88.     88  .8D   .88.      88           .88.   88  V888 88      `8b  d8'
Y88888P Y8888D' Y888888P    YP         Y888888P VP   V8P YP       `Y88P'
```

```
Enter the new name of the Passenger: jess
Enter the new email address of Passenger jess: gmail
Enter the new Phone number of Passenger jess: 345436
Enter the new address of Passenger jess: fx
Enter the new age of Passenger jess: 26
```

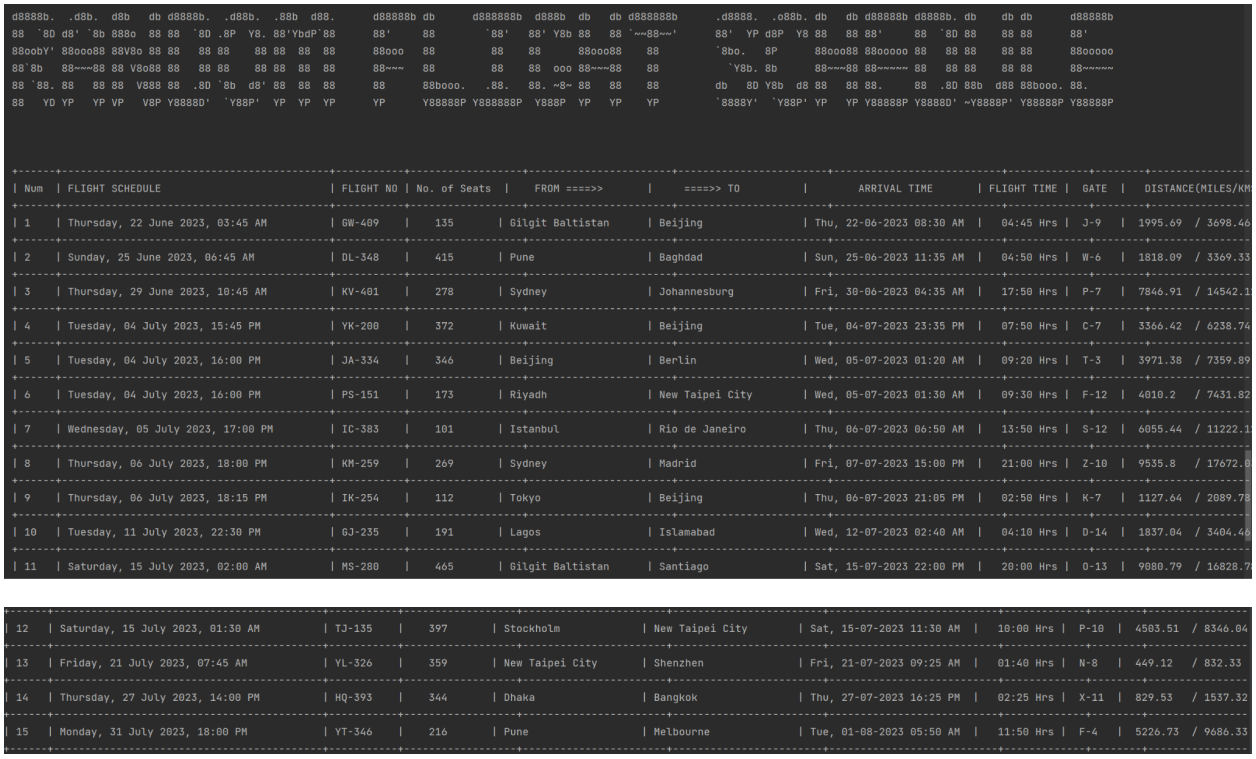
SerialNum	UserID	Passenger Names	Age	EmailID	Home Address	Phone Number
1	425 35	jess	26	gmail	fx	345436

Passenger: Delete Account



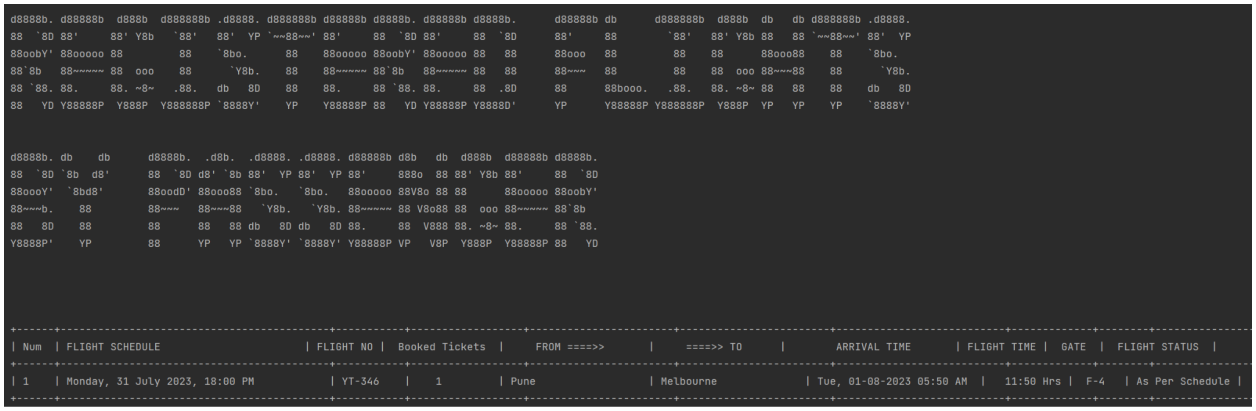
(this screenshot was taken after I made another Passenger account with the name “yahoo” as an example.)

Passenger: Display Flight Schedule

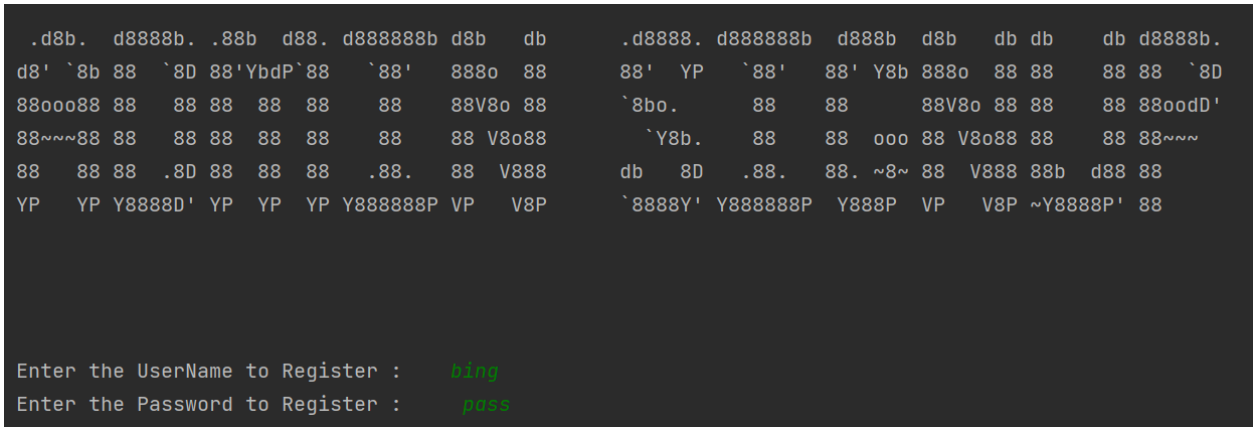


Passenger: Cancel a Flight

Passenger: Display a Flight Registered by User



Admin Register:



Admin Login:

```
 .d8b.  d8888b. .88b  d88. d888888b d8b   db      db       .d88b.   d8888b  d888888b d8b   db
d8' `8b 88  `8D 88'YbdP`88   `88'   888o  88      88      .8P  Y8. 88' Y8b   `88'   888o  88
88ooo88 88   88 88  88  88    88    88V8o 88      88      88    88 88         88    88V8o 88
88~~~88 88   88 88  88  88    88    88 V8o88      88      88    88 88  ooo    88    88 V8o88
88   88 88  .8D 88  88  88   .88.   88  V888      88booo. `8b  d8' 88. ~8~   .88.   88  V888
YP   YP Y8888D' YP  YP  YP Y888888P VP   V8P      Y88888P  `Y88P'   Y888P  Y888888P VP   V8P




Enter the Username to login to the Management System :     bing
Enter the Password to login to the Management System :    pass

                  Logged in Successfully as "bing"..... For further Proceedings, enter a value from below....

                                                 +++++++++ 2nd Layer Menu +++++++++                                                  Logged in as "bing"
                           (a) Enter 1 to add new Passenger....
                           (b) Enter 2 to search a Passenger....
                           (c) Enter 3 to update the Data of the Passenger....
                           (d) Enter 4 to delete a Passenger....
                           (e) Enter 5 to Display all Passengers....
                           (f) Enter 6 to Display all flights registered by a Passenger...
                           (g) Enter 7 to Display all registered Passengers in a Flight....
                           (h) Enter 8 to Delete a Flight....
                           (i) Enter 0 to Go back to the Main Menu/Logout....
Enter the desired Choice :
```

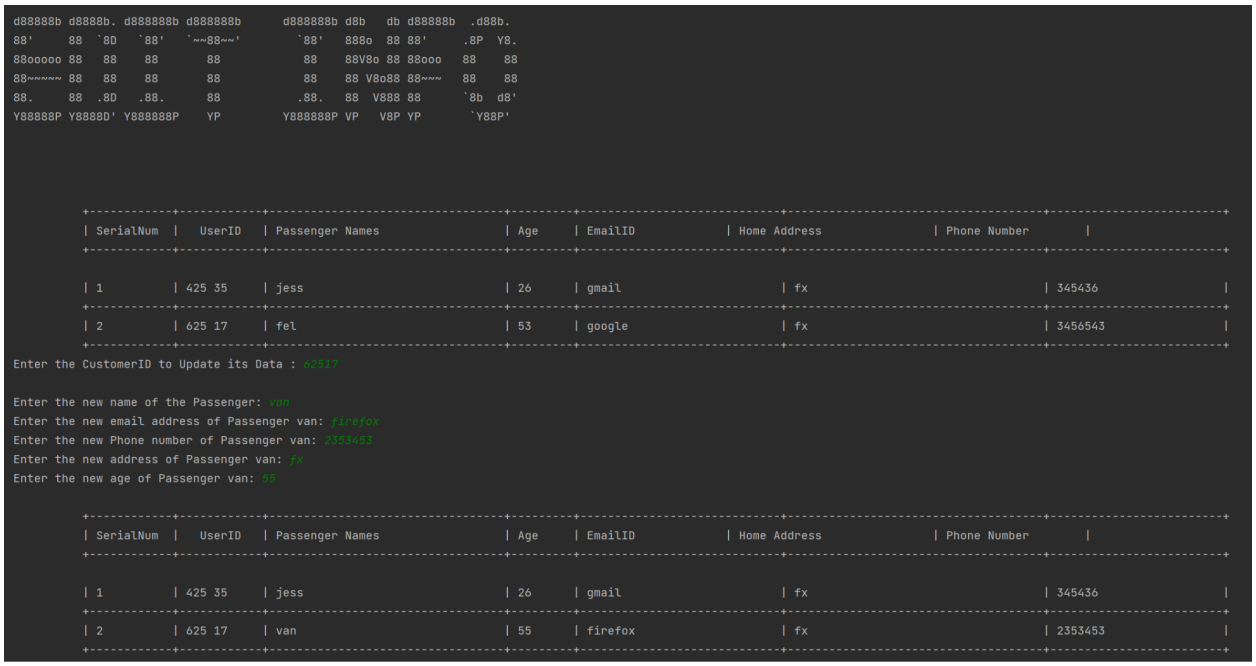
Admin: Add New Passenger

```
d8b   db d88888b db   d8b   db       .o88b. db    db .d8888. d888888b  .d88b.  .88b  d88. d88888b d8888b.
888o  88 88'     88   I8I   88      d8P  Y8 88    88 88'  YP `~~88~~' .8P  Y8. 88'YbdP`88 88'     88  `8D
88V8o 88 88ooooo 88   I8I   88      8P      88    88 `8bo.      88    88    88 88  88  88 88ooooo 88oobY'
88 V8o88 88~~~~~ Y8   I8I   88      8b      88    88   `Y8b.    88    88    88 88  88  88 88~~~~~ 88`8b
88  V888 88.     `8b d8'8b d8'      Y8b  d8 88b  d88 db   8D    88    `8b  d8' 88  88  88 88.     88 `88.
VP   V8P Y88888P  `8b8' `8d8'        `Y88P' ~Y8888P' `8888Y'    YP     `Y88P'  YP  YP  YP Y88888P 88   YD




                                                                 +++++++++++++++ Welcome to the Customer Registration Portal +++++++++++++++
Enter your name : fel
Enter your email address : google
Enter your Password : pass
Enter your Phone number : 3456543
Enter your address : fx
Enter your age : 53
```

Admin: Search a Passenger

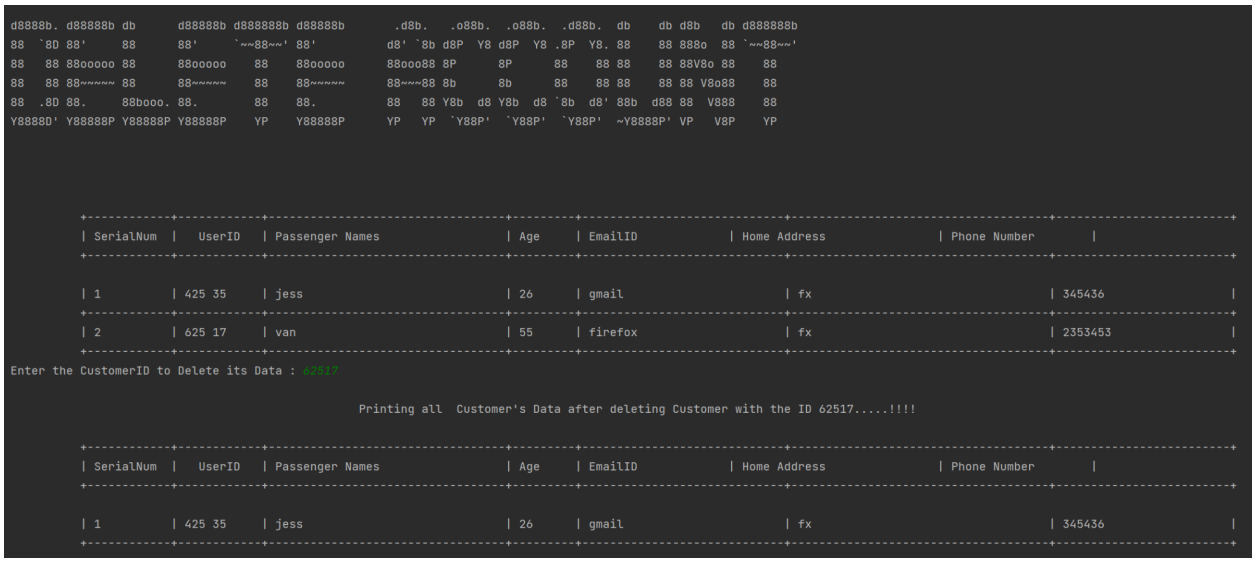
```
.d8888. d88888b  .d8b.  d8888b.  .o88b. db   db       .o88b. db    db .d8888. d888888b  .d88b.  .88b  d88. d88888b d8888b.
88'  YP 88'     d8' `8b 88  `8D d8P  Y8 88   88      d8P  Y8 88    88 88'  YP `~~88~~' .8P  Y8. 88'YbdP`88 88'     88  `8D
`8bo.   88ooooo 88ooo88 88oobY' 8P      88ooo88      8P      88    88 `8bo.      88    88    88 88  88  88 88ooooo 88oobY'
  `Y8b. 88~~~~~ 88~~~88 88`8b   8b      88~~~88      8b      88    88   `Y8b.    88    88    88 88  88  88 88~~~~~ 88`8b
db   8D 88.     88   88 88 `88. Y8b  d8 88   88      Y8b  d8 88b  d88 db   8D    88    `8b  d8' 88  88  88 88.     88 `88.
`8888Y' Y88888P YP   YP 88   YD  `Y88P' YP   YP       `Y88P' ~Y8888P' `8888Y'    YP     `Y88P'  YP  YP  YP Y88888P 88   YD


          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | SerialNum  |   UserID   | Passenger Names                  | Age     | EmailID                     | Home Address                      | Phone Number            |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+

          | 1          | 425 35     | jess                             | 26      | gmail                       | fx                                | 345436                  |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | 2          | 625 17     | fel                              | 53      | google                      | fx                                | 3456543                 |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
Enter the CustomerID to Search : 62517

                                          Customer Found...!!!Here is the Full Record...!!!


          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | SerialNum  |   UserID   | Passenger Names                  | Age     | EmailID                     | Home Address                      | Phone Number            |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+

          | 1          | 625 17     | fel                              | 53      | google                      | fx                                | 3456543                 |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
```

Admin: Update Data of Passenger

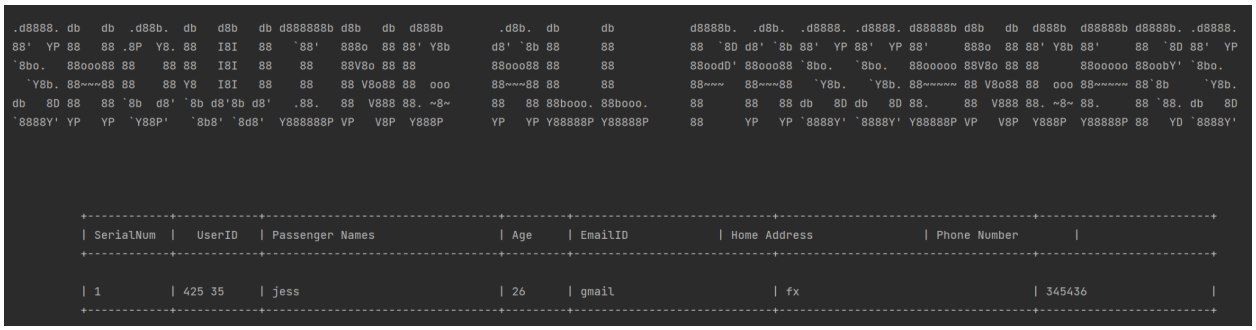


```
d88888b d8888b. d888888b d888888b      d888888b d8b   db d88888b  .d88b.
88'     88  `8D   `88'   `~~88~~'        `88'   888o  88 88'     .8P  Y8.
88ooooo 88   88    88       88            88    88V8o 88 88ooo   88    88
88~~~~~ 88   88    88       88            88    88 V8o88 88~~~   88    88
88.     88  .8D   .88.      88           .88.   88  V888 88      `8b  d8'
Y88888P Y8888D' Y888888P    YP         Y888888P VP   V8P YP       `Y88P'


          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | SerialNum  |   UserID   | Passenger Names                  | Age     | EmailID                     | Home Address                      | Phone Number            |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+

          | 1          | 425 35     | jess                             | 26      | gmail                       | fx                                | 345436                  |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | 2          | 625 17     | fel                              | 53      | google                      | fx                                | 3456543                 |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
Enter the CustomerID to Update its Data : 62517

Enter the new name of the Passenger: van
Enter the new email address of Passenger van: firefox
Enter the new Phone number of Passenger van: 2353453
Enter the new address of Passenger van: fx
Enter the new age of Passenger van: 55

          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | SerialNum  |   UserID   | Passenger Names                  | Age     | EmailID                     | Home Address                      | Phone Number            |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+

          | 1          | 425 35     | jess                             | 26      | gmail                       | fx                                | 345436                  |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
          | 2          | 625 17     | van                              | 55      | firefox                     | fx                                | 2353453                 |
          +------------+------------+----------------------------------+---------+-----------------------------+-----------------------------------+-------------------------+
```

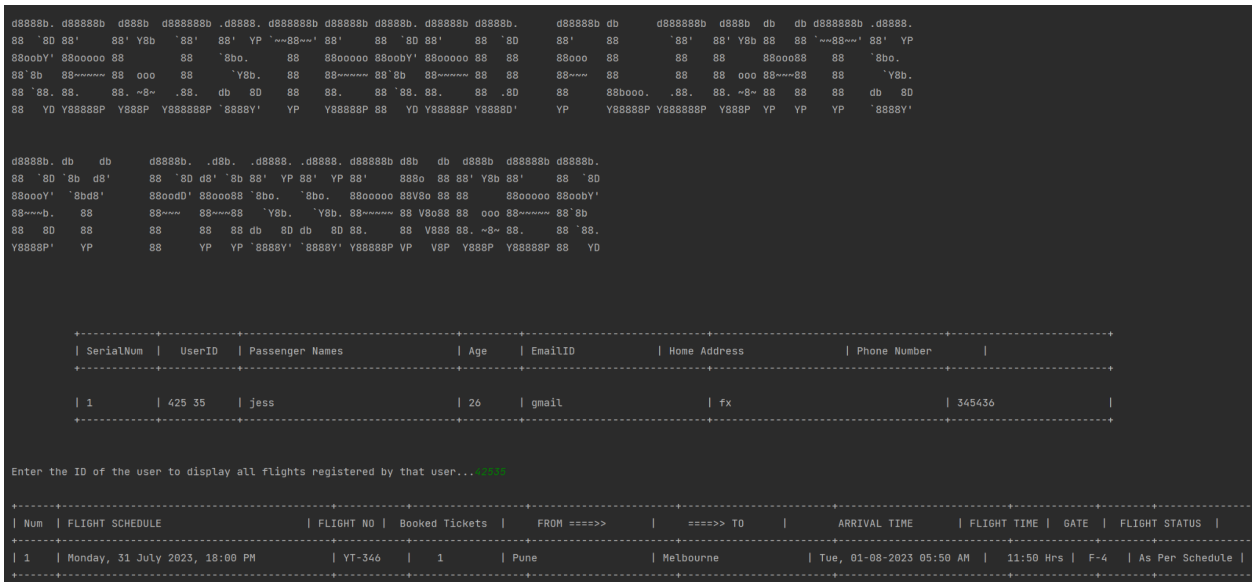
Admin: Delete Passenger



Admin: Display all Passengers

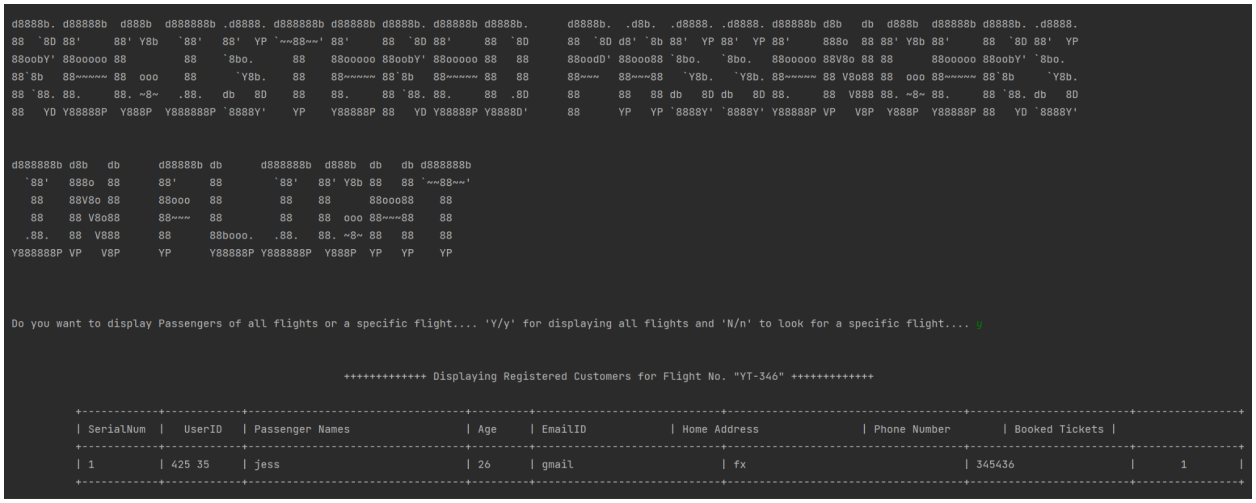


Admin: Display All Flights Registered by One Passenger



Admin: Display All Passengers in a Flight

(if you press yes.)



(if you press no.)

```
d8888b. d88888b  d888b  d888888b .d8888. d888888b d88888b d8888b. d88888b d8888b.      d8888b.  .d8b.  .d8888. d88888b d8b   db  d888b  d88888b d8888b. .d8888.
88  `8D 88'     88' Y8b   `88'   88'  YP `~~88~~' 88'     88  `8D 88'     88  `8D      88  `8D d8' `8b 88'  YP 88'     888o  88 88' Y8b 88'     88  `8D 88'  YP
88oobY' 88ooooo 88         88    `8bo.      88    88ooooo 88oobY' 88ooooo 88   88      88   88 88ooo88 `8bo.   88ooooo 88V8o 88 88      88ooooo 88oobY' `8bo.
88`8b   88~~~~~ 88  ooo    88      `Y8b.    88    88~~~~~ 88`8b   88~~~~~ 88   88      88   88 88~~~88   `Y8b. 88~~~~~ 88 V8o88 88  ooo 88~~~~~ 88`8b     `Y8b.
88 `88. 88.     88. ~8~   .88.   db   8D    88    88.     88 `88. 88.     88  .8D      88  .8D YP   88 db   8D db   8D 88. ~8~ 88.     88 V888 88. ~8~ 88.     88 `88. db   8D
88   YD Y88888P  Y888P  Y888888P `8888Y'    YP    Y88888P 88   YD Y88888P Y8888D'      Y8888D' YP   YP `8888Y' `8888Y' Y88888P VP   V8P  Y888P  Y88888P 88   YD `8888Y'


d888888b d8b   db      d88888b db      d888888b  d888b  db   db d888888b
  `88'   888o  88      88'     88        `88'   88' Y8b 88   88 `~~88~~'
   88    88V8o 88      88ooo   88         88    88      88ooo88    88
   88    88 V8o88      88~~~   88         88    88  ooo 88~~~88    88
  .88.   88  V888      88      88booo.   .88.   88. ~8~ 88   88    88
Y888888P VP   V8P      YP      Y88888P Y888888P  Y888P  YP   YP    YP



Do you want to display Passengers of all flights or a specific flight.... 'Y/y' for displaying all flights and 'N/n' to look for a specific flight.... n

+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| Num  | FLIGHT SCHEDULE                         | FLIGHT NO | No. of Seats  |    FROM =====>       |     =====>> TO         |        ARRIVAL TIME       | FLIGHT TIME |  GATE  |   DISTANCE(MILES/KM
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 1    | Thursday, 22 June 2023, 03:45 AM        | GW-409    |    135       | Gilgit Baltistan      | Beijing               | Thu, 22-06-2023 08:30 AM  |   04:45 Hrs |  J-9   |  1995.69  / 3698.46
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 2    | Sunday, 25 June 2023, 06:45 AM          | DL-348    |    415       | Pune                  | Baghdad               | Sun, 25-06-2023 11:35 AM  |   04:50 Hrs |  W-6   |  1818.09  / 3369.33
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 3    | Thursday, 29 June 2023, 10:45 AM        | KV-401    |    278       | Sydney                | Johannesburg          | Fri, 30-06-2023 04:35 AM  |   17:50 Hrs |  P-7   |  7846.91  / 14542.1
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 4    | Tuesday, 04 July 2023, 15:45 PM         | YK-200    |    372       | Kuwait                | Beijing               | Tue, 04-07-2023 23:35 PM  |   07:50 Hrs |  C-7   |  3366.42  / 6238.74
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 5    | Tuesday, 04 July 2023, 16:00 PM         | JA-334    |    346       | Beijing               | Berlin                | Wed, 05-07-2023 01:20 AM  |   09:20 Hrs |  T-3   |  3971.38  / 7359.89
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 6    | Tuesday, 04 July 2023, 16:00 PM         | PS-151    |    173       | Riyadh                | New Taipei City       | Wed, 05-07-2023 01:30 AM  |   09:30 Hrs |  F-12  |  4010.2   / 7431.82
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 7    | Wednesday, 05 July 2023, 17:00 PM       | IC-383    |    101       | Istanbul              | Rio de Janeiro        | Thu, 06-07-2023 06:50 AM  |   13:50 Hrs |  S-12  |  6055.44  / 11222.1
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 8    | Thursday, 06 July 2023, 18:00 PM        | KM-259    |    269       | Sydney                | Madrid                | Fri, 07-07-2023 15:00 PM  |   21:00 Hrs |  Z-10  |  9535.8   / 17672.0
```

```
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 9    | Thursday, 06 July 2023, 18:15 PM        | IK-254    |    112       | Tokyo                 | Beijing               | Thu, 06-07-2023 21:05 PM  |   02:50 Hrs |  K-7   |  1127.64  / 2089.78
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 10   | Tuesday, 11 July 2023, 22:30 PM         | GJ-235    |    191       | Lagos                 | Islamabad             | Wed, 12-07-2023 02:40 AM  |   04:10 Hrs |  D-14  |  1837.04  / 3404.46
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 11   | Saturday, 15 July 2023, 02:00 AM        | MS-280    |    465       | Gilgit Baltistan      | Santiago              | Sat, 15-07-2023 22:00 PM  |   20:00 Hrs |  O-13  |  9080.79  / 16828.7
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 12   | Saturday, 15 July 2023, 01:30 AM        | TJ-135    |    397       | Stockholm             | New Taipei City       | Sat, 15-07-2023 11:30 AM  |   10:00 Hrs |  P-10  |  4503.51  / 8346.04
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 13   | Friday, 21 July 2023, 07:45 AM          | YL-326    |    359       | New Taipei City       | Shenzhen              | Fri, 21-07-2023 09:25 AM  |   01:40 Hrs |  N-8   |  449.12   / 832.33
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 14   | Thursday, 27 July 2023, 14:00 PM        | HQ-393    |    344       | Dhaka                 | Bangkok               | Thu, 27-07-2023 16:25 PM  |   02:25 Hrs |  X-11  |  829.53   / 1537.32
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
| 15   | Monday, 31 July 2023, 18:00 PM          | YT-346    |    221       | Pune                  | Melbourne             | Tue, 01-08-2023 05:50 AM  |   11:50 Hrs |  F-4   |  5226.73  / 9686.33
+------+-----------------------------------------+-----------+-----------------+-----------------------+-----------------------+---------------------------+-------------+--------+-----------------
Enter the Flight Number to display the list of passengers registered in that flight... yt-346


                                    ++++++++++++ Displaying Registered Customers for Flight No. "YT-346" ++++++++++++

        +------------+------------+-------------------------------+--------+---------------------------+-------------------------------+--------------------------+----------------+
        | SerialNum  |   UserID   | Passenger Names               | Age    | EmailID                   | Home Address                  | Phone Number             | Booked Tickets |
        +------------+------------+-------------------------------+--------+---------------------------+-------------------------------+--------------------------+----------------+
        | 1          | 425 35     | jess                          | 26     | gmail                     | fx                            | 345436                   |       1        |
        +------------+------------+-------------------------------+--------+---------------------------+-------------------------------+--------------------------+----------------+
```

Admin: Delete a Flight

```
d8888b. d88888b db      d88888b d888888b d88888b      d88888b db      d888888b  d888b  db   db d888888b
88  `8D 88'     88      88'     `~~88~~' 88'          88'     88        `88'   88' Y8b 88   88 `~~88~~'
88   88 88ooooo 88      88ooooo    88    88ooooo      88ooo   88         88    88      88ooo88    88
88   88 88~~~~~ 88      88~~~~~    88    88~~~~~      88~~~   88         88    88  ooo 88~~~88    88
88  .8D 88.     88booo. 88.        88    88.          88      88booo.   .88.   88. ~8~ 88   88    88
Y8888D' Y88888P Y88888P Y88888P    YP    Y88888P      YP      Y88888P Y888888P  Y888P  YP   YP    YP
```

Num	FLIGHT SCHEDULE	FLIGHT NO	No. of Seats	FROM =====>>	=====>> TO	ARRIVAL TIME	FLIGHT TIME	GATE	DISTANCE(MILES/KM
1	Thursday, 22 June 2023, 03:45 AM	GW-409	135	Gilgit Baltistan	Beijing	Thu, 22-06-2023 08:30 AM	04:45 Hrs	J-9	1995.69 / 3698.46
2	Sunday, 25 June 2023, 06:45 AM	DL-348	415	Pune	Baghdad	Sun, 25-06-2023 11:35 AM	04:50 Hrs	W-6	1818.09 / 3369.33
3	Thursday, 29 June 2023, 10:45 AM	KV-401	278	Sydney	Johannesburg	Fri, 30-06-2023 04:35 AM	17:50 Hrs	P-7	7846.91 / 14542.1
4	Tuesday, 04 July 2023, 15:45 PM	YK-200	372	Kuwait	Beijing	Tue, 04-07-2023 23:35 PM	07:50 Hrs	C-7	3366.42 / 6238.74
5	Tuesday, 04 July 2023, 16:00 PM	JA-334	346	Beijing	Berlin	Wed, 05-07-2023 01:20 AM	09:20 Hrs	T-3	3971.38 / 7359.89
6	Tuesday, 04 July 2023, 16:00 PM	PS-151	173	Riyadh	New Taipei City	Wed, 05-07-2023 01:30 AM	09:30 Hrs	F-12	4010.2 / 7431.82
7	Wednesday, 05 July 2023, 17:00 PM	IC-383	101	Istanbul	Rio de Janeiro	Thu, 06-07-2023 06:50 AM	13:50 Hrs	S-12	6055.44 / 11222.1
8	Thursday, 06 July 2023, 18:00 PM	KM-259	269	Sydney	Madrid	Fri, 07-07-2023 15:00 PM	21:00 Hrs	Z-10	9535.8 / 17672.0
9	Thursday, 06 July 2023, 18:15 PM	IK-254	112	Tokyo	Beijing	Thu, 06-07-2023 21:05 PM	02:50 Hrs	K-7	1127.64 / 2089.78
10	Tuesday, 11 July 2023, 22:30 PM	GJ-235	191	Lagos	Islamabad	Wed, 12-07-2023 02:40 AM	04:10 Hrs	D-14	1837.04 / 3404.46
11	Saturday, 15 July 2023, 02:00 AM	MS-280	465	Gilgit Baltistan	Santiago	Sat, 15-07-2023 22:00 PM	20:00 Hrs	O-13	9080.79 / 16828.7
12	Saturday, 15 July 2023, 01:30 AM	TJ-135	397	Stockholm	New Taipei City	Sat, 15-07-2023 11:30 AM	10:00 Hrs	P-10	4503.51 / 8346.04
13	Friday, 21 July 2023, 07:45 AM	YL-326	359	New Taipei City	Shenzhen	Fri, 21-07-2023 09:25 AM	01:40 Hrs	N-8	449.12 / 832.33
14	Thursday, 27 July 2023, 14:00 PM	HQ-393	344	Dhaka	Bangkok	Thu, 27-07-2023 16:25 PM	02:25 Hrs	X-11	829.53 / 1537.32
15	Monday, 31 July 2023, 18:00 PM	YT-346	221	Pune	Melbourne	Tue, 01-08-2023 05:50 AM	11:50 Hrs	F-4	5226.73 / 9686.33

```
Enter the Flight Number to delete the flight : hq-393
```


Num	FLIGHT SCHEDULE	FLIGHT NO	No. of Seats	FROM =====>>	=====>> TO	ARRIVAL TIME	FLIGHT TIME	GATE	DISTANCE(MILES/KM
1	Thursday, 22 June 2023, 03:45 AM	GW-409	135	Gilgit Baltistan	Beijing	Thu, 22-06-2023 08:30 AM	04:45 Hrs	J-9	1995.69 / 3698.46
2	Sunday, 25 June 2023, 06:45 AM	DL-348	415	Pune	Baghdad	Sun, 25-06-2023 11:35 AM	04:50 Hrs	W-6	1818.09 / 3369.33
3	Thursday, 29 June 2023, 10:45 AM	KV-401	278	Sydney	Johannesburg	Fri, 30-06-2023 04:35 AM	17:50 Hrs	P-7	7846.91 / 14542.1
4	Tuesday, 04 July 2023, 15:45 PM	YK-200	372	Kuwait	Beijing	Tue, 04-07-2023 23:35 PM	07:50 Hrs	C-7	3366.42 / 6238.74
5	Tuesday, 04 July 2023, 16:00 PM	JA-334	346	Beijing	Berlin	Wed, 05-07-2023 01:20 AM	09:20 Hrs	T-3	3971.38 / 7359.89
6	Tuesday, 04 July 2023, 16:00 PM	PS-151	173	Riyadh	New Taipei City	Wed, 05-07-2023 01:30 AM	09:30 Hrs	F-12	4010.2 / 7431.82
7	Wednesday, 05 July 2023, 17:00 PM	IC-383	101	Istanbul	Rio de Janeiro	Thu, 06-07-2023 06:50 AM	13:50 Hrs	S-12	6055.44 / 11222.1
8	Thursday, 06 July 2023, 18:00 PM	KM-259	269	Sydney	Madrid	Fri, 07-07-2023 15:00 PM	21:00 Hrs	Z-10	9535.8 / 17672.0
9	Thursday, 06 July 2023, 18:15 PM	IK-254	112	Tokyo	Beijing	Thu, 06-07-2023 21:05 PM	02:50 Hrs	K-7	1127.64 / 2089.78
10	Tuesday, 11 July 2023, 22:30 PM	GJ-235	191	Lagos	Islamabad	Wed, 12-07-2023 02:40 AM	04:10 Hrs	D-14	1837.04 / 3404.46
11	Saturday, 15 July 2023, 02:00 AM	MS-280	465	Gilgit Baltistan	Santiago	Sat, 15-07-2023 22:00 PM	20:00 Hrs	O-13	9080.79 / 16828.7
12	Saturday, 15 July 2023, 01:30 AM	TJ-135	397	Stockholm	New Taipei City	Sat, 15-07-2023 11:30 AM	10:00 Hrs	P-10	4503.51 / 8346.04
13	Friday, 21 July 2023, 07:45 AM	YL-326	359	New Taipei City	Shenzhen	Fri, 21-07-2023 09:25 AM	01:40 Hrs	N-8	449.12 / 832.33
14	Monday, 31 July 2023, 18:00 PM	YT-346	221	Pune	Melbourne	Tue, 01-08-2023 05:50 AM	11:50 Hrs	F-4	5226.73 / 9686.33

Logging out:

```
db       .d88b.   d888b   d888b  d88888b d8888b.       .d88b.  db    db d888888b
88      .8P  Y8. 88' Y8b 88' Y8b 88'     88  `8D      .8P  Y8. 88    88 `~~88~~'
88      88    88 88      88      88ooooo 88   88      88    88 88    88    88
88      88    88 88  ooo 88  ooo 88~~~~~ 88   88      88    88 88    88    88
88booo. `8b  d8' 88. ~8~ 88. ~8~ 88.     88  .8D      `8b  d8' 88b  d88    88
Y88888P  `Y88P'   Y888P   Y888P  Y88888P Y8888D'       `Y88P'  ~Y8888P'    YP



Thanks for Using Binus Airlines Ticketing System...!!!


(a) Press 0 to Exit.
(b) Press 1 to Login as admin.
(c) Press 2 to Register as admin.
(d) Press 3 to Login as Passenger.
(e) Press 4 to Register as Passenger.
 Enter the desired option:    0
Thank you for visiting Binus Airlines !

Process finished with exit code 0
```

4. How the Program Works

User.java

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

All the imported libraries used in this class

```java
public class User {

    // array to store admin information
    5 usages
    static String[][] adminUserNameAndPassword = new String[10][2]; // max number of admins is 10
    3 usages
    private static List<Customer> customersCollection = new ArrayList<>();
```

Initialize the variables.

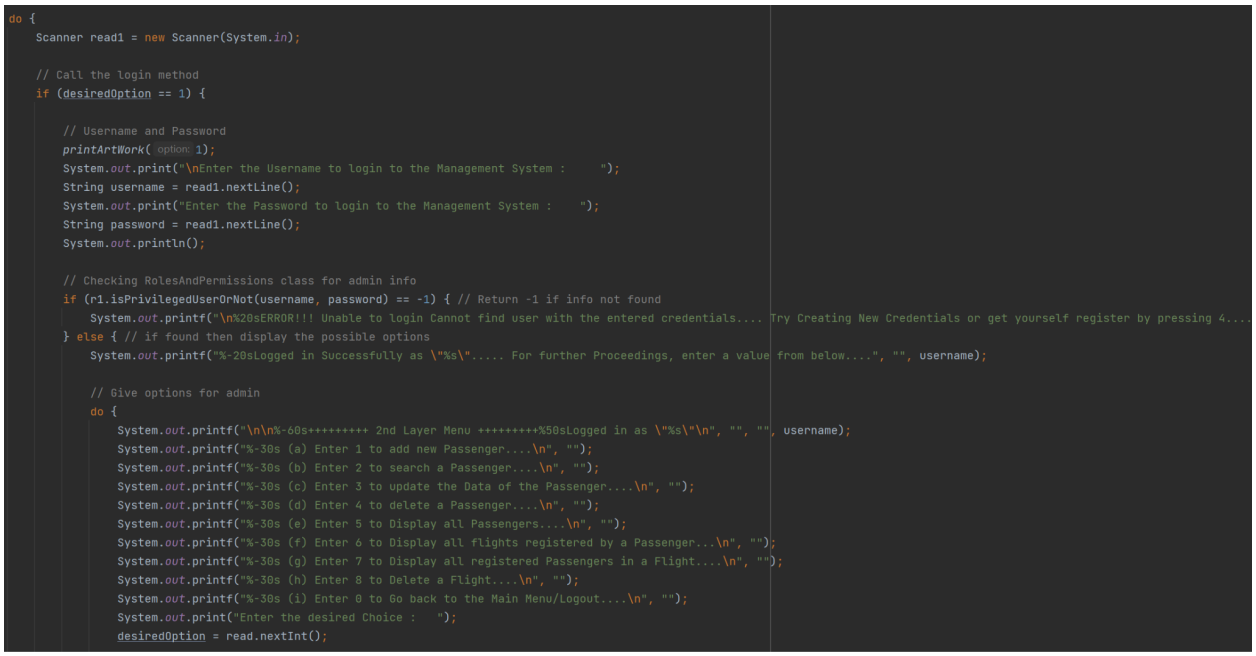
adminUserNameAndPassword is a variable that consists of two lists taking in strings. In that list, only 10 can be inserted, which means the number of admins can only be 10 at a time.

Another list called customersCollection is an ArrayList that will contain all the Customers (Passengers).

```java
public static void main(String[] args) throws IOException {
    int countNumOfUsers = 1;
    RolesAndPermissions r1 = new RolesAndPermissions();
    Flight f1 = new Flight();
    FlightReservation bookingAndReserving = new FlightReservation();
    Customer c1 = new Customer();
    f1.flightScheduler();
    Scanner read = new Scanner(System.in);
    PassengerReader.createPassenger(); // read the txt file


    welcomeScreen( option: 1);
    System.out.println("\n\t\t\t\t\t\t++++++++++++++ Welcome to Binus AirLines ++++++++++++++\n");
    System.out.println("To Further Proceed, Please enter a value.");
    displayMainMenu();
    int desiredOption = read.nextInt();
    while (desiredOption < 0 || desiredOption > 8) {
        System.out.print("ERROR!! Please enter value between 0 - 4. Enter the value again : ");
        desiredOption = read.nextInt();
    }
```

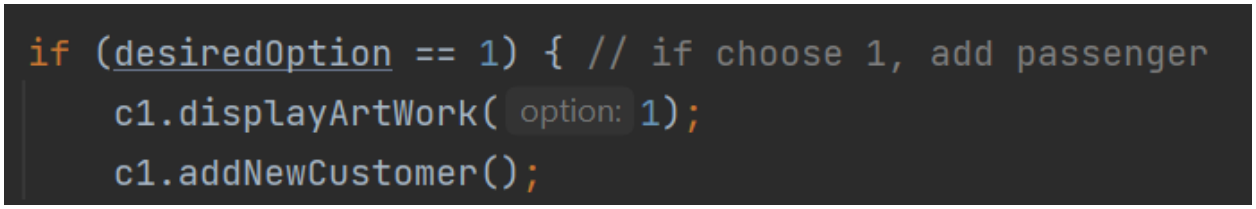
The main(String[] args) method is the entry point of the program. In this method, there are variables initialized for later use. And it starts the program by running welcomeScreen which will pop out an ASCII art with the writing WELCOME TO BINUS AIRLINES, where the user will be prompted to choose an option, whether to register/login as Passenger or Admin. If the

input the user inserted is less than 0 or more than 8, the program will keep looping until the user enters the correct number or exits the program.



```
do {
    Scanner read1 = new Scanner(System.in);

    // Call the login method
    if (desiredOption == 1) {

        // Username and Password
        printArtWork( option: 1);
        System.out.print("\nEnter the Username to login to the Management System :     ");
        String username = read1.nextLine();
        System.out.print("Enter the Password to login to the Management System :    ");
        String password = read1.nextLine();
        System.out.println();

        // Checking RolesAndPermissions class for admin info
        if (r1.isPrivilegedUserOrNot(username, password) == -1) { // Return -1 if info not found
            System.out.printf("\n%20sERROR!!! Unable to login Cannot find user with the entered credentials.... Try Creating New Credentials or get yourself register by pressing 4....
        } else { // if found then display the possible options
            System.out.printf("%-20sLogged in Successfully as \"%s\"..... For further Proceedings, enter a value from below....", "", username);

            // Give options for admin
            do {
                System.out.printf("\n\n%-60s++++++++ 2nd Layer Menu ++++++++%50sLogged in as \"%s\"\n", "", "", username);
                System.out.printf("%-30s (a) Enter 1 to add new Passenger....\n", "");
                System.out.printf("%-30s (b) Enter 2 to search a Passenger....\n", "");
                System.out.printf("%-30s (c) Enter 3 to update the Data of the Passenger....\n", "");
                System.out.printf("%-30s (d) Enter 4 to delete a Passenger....\n", "");
                System.out.printf("%-30s (e) Enter 5 to Display all Passengers....\n", "");
                System.out.printf("%-30s (f) Enter 6 to Display all flights registered by a Passenger...\n", "");
                System.out.printf("%-30s (g) Enter 7 to Display all registered Passengers in a Flight....\n", "");
                System.out.printf("%-30s (h) Enter 8 to Delete a Flight....\n", "");
                System.out.printf("%-30s (i) Enter 0 to Go back to the Main Menu/Logout....\n", "");
                System.out.print("Enter the desired Choice :   ");
                desiredOption = read.nextInt();
```

Choosing number 1 will prompt the user to login as an admin, it will ask the user to enter the Admin Username and Password, it will also check whether or not the data has been entered in the system or not. If it hasn't, it will print out an error message, but if it has, then the admin will be given options on what they want to do.



```
if (desiredOption == 1) { // if choose 1, add passenger
    c1.displayArtWork( option: 1);
    c1.addNewCustomer();
```

Pressing 1 will allow them to add a Passenger, and it will call the addNewCustomer method from the Customer class.

```
} else if (desiredOption == 2) { // if choose 2, call search method from Customer class
    c1.displayArtWork( option: 2);
    c1.displayCustomersData( showHeader: false);
    System.out.print("Enter the CustomerID to Search : ");
    String customerID = read1.nextLine();
    System.out.println();
    c1.searchUser(customerID);
```

Pressing 2 will allow them to search for a Passenger, it will ask the Admin for the Passenger's ID, and it will call the searchUser method from the Customer class using said ID.

```
} else if (desiredOption == 3) { // if choose 3, call update method from Customer class
    bookingAndReserving.displayArtWork( option: 2);
    c1.displayCustomersData( showHeader: false);
    System.out.print("Enter the CustomerID to Update its Data : ");
    String customerID = read1.nextLine();
    if (customersCollection.size() > 0) {
        c1.editUserInfo(customerID);
    } else {
        System.out.printf("%-50sNo Customer with the ID %s Found...!!!\n", " ", customerID);
    }
```

Pressing 3 will allow them to update a Passenger data. It will ask the Admin for the Passenger's ID again, and it will call the editUserInfo method from the Customer class. However, if the customersCollection list is empty (no Passengers are registered), then it will print out an error message.

```
} else if (desiredOption == 4) { // if choose 4, delete customer by taking their id
    bookingAndReserving.displayArtWork( option: 3);
    c1.displayCustomersData( showHeader: false);
    System.out.print("Enter the CustomerID to Delete its Data : ");
    String customerID = read1.nextLine();
    if (customersCollection.size() > 0) {
        c1.deleteUser(customerID);
    } else {
        System.out.printf("%-50sNo Customer with the ID %s Found...!!!\n", " ", customerID);
    }
```

Pressing 4 will allow them to delete a Passenger. It will ask the Admin for the Passenger's ID again, and it will call the deleteUser method from the Customer class. However, if the

customersCollection list is empty (no Passengers are registered), then it will print out an error message.

```
} else if (desiredOption == 5) { // if choose 5, Display availability of Passengers
    c1.displayArtWork( option: 3);
    c1.displayCustomersData( showHeader: false);
```

Pressing 5 will allow them to display all the Passengers available.

```
} else if (desiredOption == 6) { // if choose 6, display flights registered by Passengers
    bookingAndReserving.displayArtWork( option: 6);
    c1.displayCustomersData( showHeader: false);
    System.out.print("\n\nEnter the ID of the user to display all flights registered by that user...");
    String id = read1.nextLine();
    bookingAndReserving.displayFlightsRegisteredByOneUser(id);
```

Pressing 6 will allow them to display flights registered by Passengers. It will ask the Admin for the Passenger's ID again, and it will call the displayFlightsRegisteredByOneUser method from the FlightReservation class.

```
} else if (desiredOption == 7) { // if choose 7, display Passengers in a flight
    c1.displayArtWork( option: 4);
    System.out.print("Do you want to display Passengers of all flights or a specific flight.... 'Y/y' for displaying all flights and 'N/n' to look for a" +
            " specific flight.... ");
    char choice = read1.nextLine().charAt(0);
    if ('y' == choice || 'Y' == choice) {
        bookingAndReserving.displayRegisteredUsersForAllFlight();
    } else if ('n' == choice || 'N' == choice) {
        f1.displayFlightSchedule();
        System.out.print("Enter the Flight Number to display the list of passengers registered in that flight... ");
        String flightNum = read1.nextLine();
        bookingAndReserving.displayRegisteredUsersForASpecificFlight(flightNum);
    } else {
        System.out.println("Invalid Choice...No Response...!");
    }
```

Pressing 7 will allow them to display all the Passengers in a certain flight. It will ask for a choice on whether the Admin wants to display Passengers for all the flights or just display Passengers for a specific flight. Pressing Y will display all the Passengers in a flight and Pressing N will prompt the Admin to insert a flight number where they want to see the Passengers in and it will display the Passengers in that flight only. However, if the Admin entered a choice that isn't "Y" or "N", the system will output an error message.

```
else if (desiredOption == 8) { // if choose 8, delete a flight
  c1.displayArtWork( option: 5);
  f1.displayFlightSchedule();
  System.out.print("Enter the Flight Number to delete the flight : ");
  String flightNum = read1.nextLine();
  f1.deleteFlight(flightNum);
```

Pressing 8 will allow them to delete a flight. It will print out the randomized flight schedule and ask the Admin for the flight number they want to delete. Once obtained, the system will call the deleteFlight method from the Flight class to delete the flight from the list.

```
} else if (desiredOption == 0) { // if choose 0, go back to main menu
    bookingAndReserving.displayArtWork( option: 22);
    System.out.println("Thanks for Using Binus Airlines Ticketing System...!!!");
```

Pressing 0 will return the Admin back to the main menu.

```
} else {
    System.out.println("Invalid Choice. Try Logging in again");
    bookingAndReserving.displayArtWork( option: 22);
    desiredOption = 0;
}
```

Pressing any number that's not 0-8 will prompt out an error message.

```
    } while (desiredOption != 0);

}
```

The loop will continue as long as the option isn't 0 (exit).

```
} else if (desiredOption == 2) { // if option 2, register admin
    printArtWork( option: 2);
    System.out.print("\nEnter the UserName to Register :    ");
    String username = read1.nextLine();
    System.out.print("Enter the Password to Register :     ");
    String password = read1.nextLine();
    while (r1.isPrivilegedUserOrNot(username, password) != -1) { // if admin info is found, ask to change username
        System.out.print("ERROR!!! Admin with same UserName already exist. Enter new UserName:   ");
        username = read1.nextLine();
        System.out.print("Enter the Password Again:   ");
        password = read1.nextLine();
    }

    // insert admin username and password into the array
    adminUserNameAndPassword[countNumOfUsers][0] = username;
    adminUserNameAndPassword[countNumOfUsers][1] = password;

    // increment number of users
    countNumOfUsers++;
```

Pressing 2 will allow the user to register as an Admin. it will ask the user to enter the Admin Username and Password, it will also check whether there is a similar admin info or not, if there is, it will ask the user to change their Username or Password. Once done, it will add the Admin's Username and the Password to the adminUserNameAndPassword ArrayList.

```
} else if (desiredOption == 3) { // if option 3, Passenger login
    printArtWork( option: 3);
    System.out.print("\n\nEnter the Email to Login :  ");
    String userName = read1.nextLine();
    System.out.print("Enter the Password :  ");
    String password = read1.nextLine();
    String[] result = r1.isPassengerRegistered(userName, password).split( regex: "-");
```

Pressing 3 will allow the user to login as a Passenger. It will take the Passenger's email and password, and put it in a list, separated by a "-".

```
if (Integer.parseInt(result[0]) == 1) {
    int desiredChoice;
    System.out.printf("\n\n%-20sLogged in Successfully as \"%s\"..... For further Proceedings, enter a value from below....", "", userName);
    do { // options for the Passenger
        System.out.printf("\n\n%-60s+++++++++ 3rd Layer Menu +++++++++%50sLogged in as \"%s\"\n", "", "", userName);
        System.out.printf("%-40s (a) Enter 1 to Book a flight....\n", "");
        System.out.printf("%-40s (b) Enter 2 to update your Data....\n", "");
        System.out.printf("%-40s (c) Enter 3 to delete your account....\n", "");
        System.out.printf("%-40s (d) Enter 4 to Display Flight Schedule....\n", "");
        System.out.printf("%-40s (e) Enter 5 to Cancel a Flight....\n", "");
        System.out.printf("%-40s (f) Enter 6 to Display all flights registered by \"%s\"....\n", "", userName);
        System.out.printf("%-40s (g) Enter 0 to Go back to the Main Menu/Logout....\n", "");
        System.out.print("Enter the desired Choice :   ");
        desiredChoice = read.nextInt();
```

```
} else {
    System.out.printf("\n%20sERROR!!! Unable to login Cannot find user with the entered credentials.... Try Creating New Credentials or get yourself register by pressing 4....\n", "");
}
```

If the system can find the email and password that's been entered, it will run the program and give options for the Passenger, if there isn't it will run an error message.

```
if (desiredChoice == 1) { // if choose 1, display all schedules and book a flight
    bookingAndReserving.displayArtWork( option: 1);
    f1.displayFlightSchedule();
    System.out.print("\nEnter the desired flight number to book :  ");
    String flightToBeBooked = read1.nextLine();
    System.out.print("Enter the Number of tickets for " + flightToBeBooked + " flight :   ");
    int numOfTickets = read.nextInt();
    while (numOfTickets > 10) { // limit the number of tickets ordered by 10
        System.out.print("ERROR!! You can't book more than 10 tickets at a time for single flight....Enter number of tickets again : ");
        numOfTickets = read.nextInt();
    }
    bookingAndReserving.bookFlight(flightToBeBooked, numOfTickets, result[1]);
```

Pressing 1 for the Passenger will allow them to book a flight. It starts by displaying all the 15 flight schedules, and asks the Passenger for the flight number that they want to book, it will also ask for the amount of tickets they want. If they order more than 10, it will display an error message. However, if everything goes well, it will run all the data through the bookFlight method.

```
} else if (desiredChoice == 2) { // if choose 2, call the method to edit passenger info
    bookingAndReserving.displayArtWork( option: 2);
    c1.editUserInfo(result[1]);
```

Pressing 2 will call the method to edit the Passenger's info, it will ask the Passenger for their ID, and it will call the searchUser method from the Customer class using said ID.

```
} else if (desiredChoice == 3) { // if choose 3, delete a passenger
    bookingAndReserving.displayArtWork( option: 3);
    System.out.print("Are you sure to delete your account...It's an irreversible action...Enter Y/y to confirm...");
    char confirmationChar = read1.nextLine().charAt(0);
    if (confirmationChar == 'Y' || confirmationChar == 'y') {
        c1.deleteUser(result[1]); // call method to delete user
        System.out.printf("User %s's account deleted Successfully...!!!", userName);
        desiredChoice = 0; // go back to main menu
    } else {
        System.out.println("Action has been cancelled...");
    }
```

Pressing 3 will ask the Passenger if they want to delete their account. If they confirm it, the account will be deleted using the deleteUser method from the Customer class and it will go back to the main menu. However, if they cancel it, it will prompt a message.

```
} else if (desiredChoice == 4) { // if choose 4, call method to display the randomized flight schedule
    bookingAndReserving.displayArtWork( option: 4);
    f1.displayFlightSchedule();
```

Pressing 4 will show all the flight schedules by calling the displayFlightSchedule method from the Flight class.

```
} else if (desiredChoice == 5) { // if choose 5, call the cancel flight method
    bookingAndReserving.displayArtWork( option: 5);
    bookingAndReserving.cancelFlight(result[1]);
```

Pressing 5 will call the cancelFlight method from the FlightReservation class, allowing the Passenger to remove tickets from a flight they've registered in.

```
} else if (desiredChoice == 6) { // if choose 6, call the method to display flights registered by passenger
    bookingAndReserving.displayArtWork( option: 6);
    bookingAndReserving.displayFlightsRegisteredByOneUser(result[1]);
```

Pressing 6 will display flights registered by a single Passenger.

```
        if (desiredChoice != 0) {
            System.out.println("Invalid Choice. Try logging in again...");
        }
        desiredChoice = 0;
    }
} while (desiredChoice != 0);
```

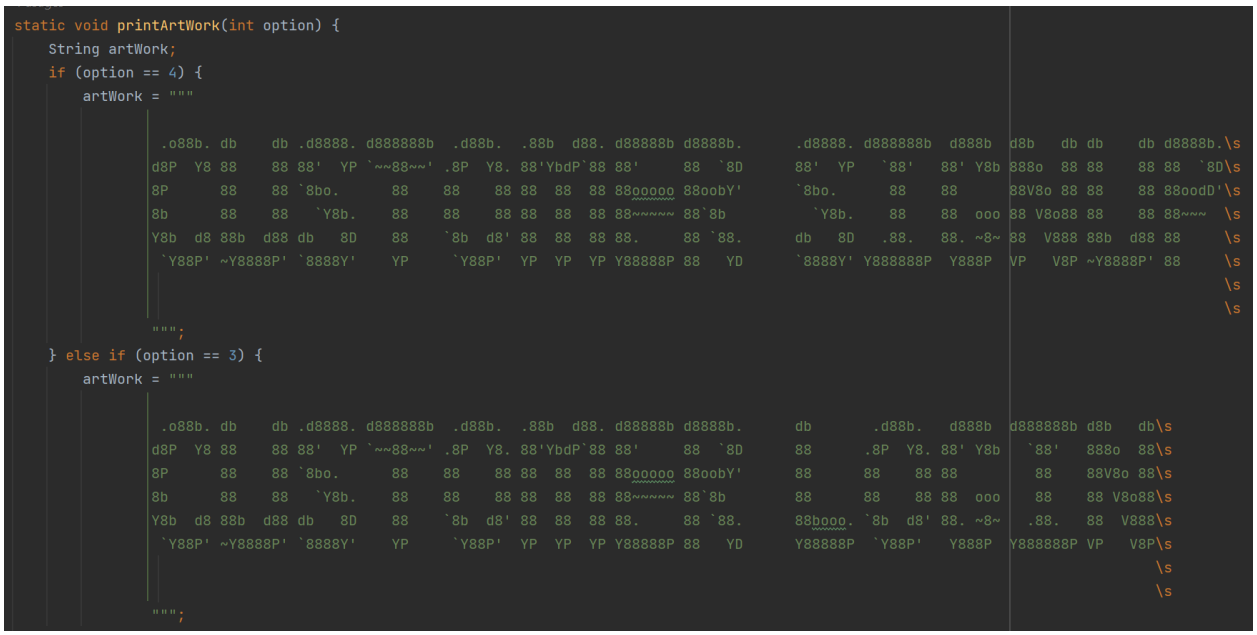
As long as the Passenger doesn't press 0, the program will keep running.

```
static void displayMainMenu() { // main menu
    System.out.println("\n\n(a) Press 0 to Exit.");
    System.out.println("(b) Press 1 to Login as admin.");
    System.out.println("(c) Press 2 to Register as admin.");
    System.out.println("(d) Press 3 to Login as Passenger.");
    System.out.println("(e) Press 4 to Register as Passenger.");
    System.out.print(" Enter the desired option:    ");
}
```

Method to display the main menu.

```
static void welcomeScreen(int option) {
    String artWork;

    if (option == 1) {
        artWork = """
                888       888          888                                          888                 888888b.     d8b                                                  d8888 d8b
                888   o   888          888                                          888                 888  "88b    Y8P                                                 d88888 Y8P
                888  d8b  888          888                                          888                 888  .88P                                                     d88P888
                888 d888b 888  .d88b.  888  .d8888b  .d88b.  88888b.d88b.   .d88b.       888888  .d88b.       8888888K.    888  88888b.   db    db   .d8888b           d88P 888 888 888d88
                888d88888b888 d8P  Y8b 888 d88P"    d88""88b 888 "888 "88b d8P  Y8b      888    d88""88b      888  "Y88b   888  888 "88b  88    88   88K               d88P  888 888 888P"
                88888P Y88888 88888888 888 888      888  888 888  888  888 888  888      888    888  888      888    888   888  888  888  88    88   "Y8888b.          d88P   888 888 888
                8888P   Y8888 Y8b.     888 Y88b.    Y88..88P 888  888  888 Y8b.          Y88b.  Y88..88P      888   d88P   888  888  888  88b  d88        X88         d8888888888 888 888
                888P     Y888  "Y8888  888  "Y8888P  "Y88P"  888  888  888  "Y8888        "Y888  "Y88P"       8888888P"    888  888  888  ~Y8888P'   88888P'       d88P      888 888 888
                """;
    } else {
        artWork = "Thank you for visiting Binus Airlines !";
    }
    System.out.println(artWork);
}
```

Method to draw the ASCII art for the welcome screen.



```
static void printArtWork(int option) {
    String artWork;
    if (option == 4) {
        artWork = """
                 .o88b. db    db .d8888. d888888b  .d88b.  .88b  d88. d88888b d8888b.       .d8888. d888888b  d888b  d8b   db db    db d8888b.\s
                d8P  Y8 88    88 88'  YP `~~88~~' .8P  Y8. 88'YbdP`88 88'     88  `8D      88'  YP   `88'   88' Y8b 888o  88 88    88 88  `8D\s
                8P      88    88 `8bo.      88    88    88 88  88  88 88ooooo 88oobY'      `8bo.      88    88      88V8o 88 88    88 88oodD'\s
                8b      88    88   `Y8b.    88    88    88 88  88  88 88~~~~~ 88`8b          `Y8b.    88    88  ooo 88 V8o88 88    88 88~~~  \s
                Y8b  d8 88b  d88 db   8D    88    `8b  d8' 88  88  88 88.     88 `88.      db   8D   .88.   88. ~8~ 88  V888 88b  d88 88     \s
                 `Y88P' ~Y8888P' `8888Y'    YP     `Y88P'  YP  YP  YP Y88888P 88   YD      `8888Y' Y888888P  Y888P  VP   V8P ~Y8888P' 88     \s
                                                                                                                                      \s
                                                                                                                                      \s
                """;
    } else if (option == 3) {
        artWork = """
                 .o88b. db    db .d8888. d888888b  .d88b.  .88b  d88. d88888b d8888b.       db       .d88b.   d888b  d888888b d8b   db\s
                d8P  Y8 88    88 88'  YP `~~88~~' .8P  Y8. 88'YbdP`88 88'     88  `8D      88      .8P  Y8. 88' Y8b   `88'   888o  88\s
                8P      88    88 `8bo.      88    88    88 88  88  88 88ooooo 88oobY'      88      88    88 88         88    88V8o 88\s
                8b      88    88   `Y8b.    88    88    88 88  88  88 88~~~~~ 88`8b        88      88    88 88  ooo    88    88 V8o88\s
                Y8b  d8 88b  d88 db   8D    88    `8b  d8' 88  88  88 88.     88 `88.      88booo. `8b  d8' 88. ~8~  .88.   88  V888\s
                 `Y88P' ~Y8888P' `8888Y'    YP     `Y88P'  YP  YP  YP Y88888P 88   YD      Y88888P  `Y88P'   Y888P  Y888888P VP   V8P\s
                                                                                                                                  \s
                                                                                                                                  \s
                """;
```

```
} else if (option == 2) {
    artWork = """
              .d8b.  d8888b. .88b  d88. d888888b d8b   db      .d8888. d888888b  d888b  d8b   db db    db d8888b.\s
             d8' `8b 88  `8D 88'YbdP`88   `88'   888o  88      88'  YP   `88'   88' Y8b 888o  88 88    88 88  `8D\s
             88ooo88 88   88 88  88  88    88    88V8o 88      `8bo.      88    88      88V8o 88 88    88 88oodD'\s
             88~~~88 88   88 88  88  88    88    88 V8o88        `Y8b.    88    88  ooo 88 V8o88 88    88 88~~~  \s
             88   88 88  .8D 88  88  88   .88.   88  V888      db   8D   .88.   88. ~8~ 88  V888 88b  d88 88     \s
             YP   YP Y8888D' YP  YP  YP Y888888P VP   V8P      `8888Y' Y888888P  Y888P  VP   V8P ~Y8888P' 88     \s
                                                                                                               \s
                                                                                                               \s
                \s""";
} else {
    artWork = """
              .d8b.  d8888b. .88b  d88. d888888b d8b   db      db       .d88b.   d888b  d888888b d8b   db\s
             d8' `8b 88  `8D 88'YbdP`88   `88'   888o  88      88      .8P  Y8. 88' Y8b   `88'   888o  88\s
             88ooo88 88   88 88  88  88    88    88V8o 88      88      88    88 88         88    88V8o 88\s
             88~~~88 88   88 88  88  88    88    88 V8o88      88      88    88 88  ooo    88    88 V8o88\s
             88   88 88  .8D 88  88  88   .88.   88  V888      88booo. `8b  d8' 88. ~8~   .88.   88  V888\s
             YP   YP Y8888D' YP  YP  YP Y888888P VP   V8P      Y88888P  `Y88P'   Y888P  Y888888P VP   V8P\s
                                                                                                        \s
                                                                                                        \s
            """;
}

System.out.println(artWork);
```

Method for the ASCII art. Pressing 4 will show the Passenger register art, pressing 3 will show the Passenger login art, pressing 2 will show the Admin register art, and pressing 1 will show the Admin login art.

```
public static List<Customer> getCustomersCollection() { return customersCollection; }
```

Getter for the customersCollection list.

RandomGenerator.java

```
private String randomNum;
// array of locations, city at index 0, latitude at index 1, longitude at index 2
9 usages
private static final String[][] destinations = {
        {"Karachi", "24.871940", "66.988060"}, {"Bangkok", "13.921430", "100.595337"}, {"Jakarta", "-6.174760", "106.827072"},
        {"Islamabad", "33.607587", "73.100316"}, {"New York City", "40.642422", "-73.781749"}, {"Lahore", "31.521139", "74.406519"},
        {"Gilgit Baltistan", "35.919108", "74.332838"}, {"Jeddah", "21.683647", "39.152862"}, {"Riyadh", "24.977080", "46.688942"}, {"New Delhi", "28.555764", "77.096520"},
        {"Hong Kong", "22.285005", "114.158339"}, {"Beijing", "40.052121", "116.609609"}, {"Tokyo", "35.550899", "139.780683"}, {"Kuala Lumpur", "2.749914", "101.707160"},
        {"Sydney", "-33.942028", "151.174304"}, {"Melbourne", "-37.671812", "144.846079"}, {"Cape Town", "-33.968879", "18.596982"}, {"Madrid", "40.476938", "-3.569428"},
        {"Dublin", "53.424077", "-6.256792"}, {"Johannesburg", "25.936834", "27.925890"}, {"London", "51.504473", "0.052271"}, {"Los Angeles", "33.942912", "-118.406829"},
        {"Brisbane", "-27.388925", "153.116751"}, {"Amsterdam", "52.308100", "4.764170"}, {"Stockholm", "59.651236", "17.924793"}, {"Frankfurt", "50.050085", "8.571911"},
        {"New Taipei City", "25.066471", "121.551638"}, {"Rio de Janeiro", "-22.812160", "-43.248636"}, {"Seoul", "37.558773", "126.802822"}, {"Yokohama", "35.462819", "139.637008"},
        {"Ankara", "39.951898", "32.688792"}, {"Casablanca", "33.368202", "-7.580998"}, {"Shenzhen", "22.633977", "113.809360"}, {"Baghdad", "33.264824", "44.232014"},
        {"Alexandria", "40.232302", "-85.637150"}, {"Pune", "18.579019", "73.908572"}, {"Shanghai", "31.145326", "121.804512"}, {"Istanbul", "41.289143", "41.261401", "28.742376"},
        {"Bhutan", "22.648322", "88.443152"}, {"Dhaka", "23.847177", "90.404133"}, {"Munich", "48.354327", "11.788688"}, {"Perth", "56.435749", "-3.371675"},
        {"Mexico", "21.038103", "-86.875259"}, {"California", "32.733089", "-117.194514"}, {"Kabul", "34.564296", "69.211574"}, {"Yangon", "47.604505", "-122.330604"},
        {"Lagos", "17.981829", "102.565684"}, {"Santiago", "-33.394795", "-70.790183"}, {"Kuwait", "29.239250", "47.971575"}, {"Nairobi", "39.958361", "41.174310"},
        {"Tehran", "35.696000", "51.401000"}, {"Saint Petersburg", "60.013492", "29.722189"}, {"Hanoi", "21.219185", "105.803967"}, {"Sialkot", "32.328361", "74.215310"},
        {"Berlin", "52.554316", "13.291213"}, {"Paris", "48.999560", "2.539274"}, {"Dubai", "25.249869", "55.366483"}
};
```

Initialize a variable called randomNum.

Another variable called destinations is a list that takes a String. In it, it contains an array of locations. City is at index 0, latitude at index 1, and longitude at index 2.

```
public void randomIDGen() {
    Random rand = new Random();
    String randomID = Integer.toString(rand.nextInt( bound: 1000000)); // make random ID, limit the number by 1,000,000

    while (Integer.parseInt(randomID) < 20000) {
        randomID = Integer.toString(rand.nextInt( bound: 1000000));
    }
    setRandomNum(randomID);
}
```

Method used to generate random IDs for Passengers. It makes a random ID of numbers up to 1 million. If the generated number is less than 20.000, it will generate another one again.

```java
public String[][] randomDestinations() {
    Random rand = new Random();
    int randomCity1 = rand.nextInt(destinations.length); // pick two random cities
    int randomCity2 = rand.nextInt(destinations.length);
    String fromWhichCity = destinations[randomCity1][0]; // city name
    String fromWhichCityLat = destinations[randomCity1][1]; // latitude
    String fromWhichCityLong = destinations[randomCity1][2]; // longitude
    while (randomCity2 == randomCity1) { // if both cities the same, randomize again
        randomCity2 = rand.nextInt(destinations.length);
    }
    String toWhichCity = destinations[randomCity2][0];
    String toWhichCityLat = destinations[randomCity2][1];
    String toWhichCityLong = destinations[randomCity2][2];
    String[][] chosenDestinations = new String[2][3];
    chosenDestinations[0][0] = fromWhichCity;
    chosenDestinations[0][1] = fromWhichCityLat;
    chosenDestinations[0][2] = fromWhichCityLong;
    chosenDestinations[1][0] = toWhichCity;
    chosenDestinations[1][1] = toWhichCityLat;
    chosenDestinations[1][2] = toWhichCityLong;
    return chosenDestinations;
}
```

Method used to generate the random destinations for the flight schedule. It will take two random cities from the destinations list, and it will put all the data like the city name, latitude, and longitude into different variables, it also checks whether or not the two cities are the same, if it is, it will randomize again. In the end, it will put the two cities' data into another list called chosenDestinations.

```java
public int randomNumOfSeats() {
    Random random = new Random();
    int numOfSeats = random.nextInt( bound: 500); // randomize a number, limit is 500
    while (numOfSeats < 75) { // if number of seats is less than 75, randomize again
        numOfSeats = random.nextInt( bound: 500);
    }
    return numOfSeats;
}
```

Method to generate a random number of seats in a flight. The maximum number of seats is 500, and if it produces a number lower than 75 it will generate it again.

```java
public String randomFlightNumbGen(int uptoHowManyLettersRequired, int divisible) {
    Random random = new Random();
    StringBuilder randomAlphabets = new StringBuilder();
    for (int i = 0; i < uptoHowManyLettersRequired; i++) {
        // char represents the random alphabets, randomize limited by 26
        randomAlphabets.append((char) (random.nextInt( bound: 26) + 'a'));
    }
    // after generating random letters, add a - to it followed by a random number based on the amount of seats divided by a number
    randomAlphabets.append("-").append(randomNumOfSeats() / divisible);
    return randomAlphabets.toString(); // return the random alphabets in String
}
```

Method to generate a random flight number for a flight. The for loop iterates according to uptoHowManyLettersRequired, and a random letter is generated. The end product is a random sort of alphabet added to the randomAlphabets variable. After generating random letters, add a "-" to it based on the randomized amount of seats divided by a number.

DisplayClass.java

```java
import java.util.List;

1 usage  1 implementation
public interface DisplayClass {

    1 usage  1 implementation
    void displayRegisteredUsersForAllFlight();

    1 usage  1 implementation
    void displayRegisteredUsersForASpecificFlight(String flightNum);

    2 usages  1 implementation
    void displayHeaderForUsers(Flight flight, List<Customer> c);

    3 usages  1 implementation
    void displayFlightsRegisteredByOneUser(String userID);

    12 usages  1 implementation
    void displayArtWork(int options);
}
```

It contains all the methods that are going to be used in the Flight Reservation class.

FlightDistance.java

```java
public abstract class FlightDistance {
    1 usage  1 implementation
    public abstract String toString(int i);

    1 usage  1 implementation
    public abstract String[] calculateDistance(double lat1, double lon1, double lat2, double lon2);
    /*
     * lat1 origin city/airport latitude
     * lon1 origin city/airport longitude
     * lat2 destination city/airport latitude
     * lon2 destination city/airport longitude
     */

}
```

It contains all the methods that's going to be used in the Flight class.

Customer.java

```java
public void addNewCustomer() {
    System.out.printf("\n\n\n%60s ++++++++++++++ Welcome to the Customer Registration Portal ++++++++++++++", "");
    Scanner read = new Scanner(System.in);
    System.out.print("\nEnter your name : ");
    String name = read.nextLine();
    System.out.print("Enter your email address : ");
    String email = read.nextLine();
    while (isUniqueData(email)) { // checks email, if already exists, prompts Passenger to enter new email
        System.out.println("ERROR!!! User with the same email already exists... Use new email or login using the previous credentials....");
        System.out.print("Enter your email address : ");
        email = read.nextLine();
    }
    System.out.print("Enter your Password : ");
    String password = read.nextLine();
    System.out.print("Enter your Phone number : ");
    String phone = read.nextLine();
    System.out.print("Enter your address : ");
    String address = read.nextLine();
    System.out.print("Enter your age : ");
    int age = read.nextInt();
    customerCollection.add(new Customer(name, email, password, phone, address, age));

    // adds all the input into a txt file
    File file = new File( pathname: "C:/Users/edely/Downloads/AirLineReservationSystem/AirLineReservationSystem/src/PassengerData.txt");
    Writer fileWrite = null;
    try {
        fileWrite = new FileWriter(file,  append: true);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    try {
        fileWrite.write( str: "Passenger:\n");
        fileWrite.write( str: name + "\n");
        fileWrite.write( str: email + "\n");
        fileWrite.write( str: password + "\n");
        fileWrite.write( str: phone + "\n");
        fileWrite.write( str: address + "\n");
        fileWrite.write( str: age + " \n");
        fileWrite.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
```

This method is used to add a new Passenger. It will ask for all the details like their name, email, password, etc, and it will check whether or not the email has been registered or not, if it has, it will ask for a new email. After it's all added, the data will be added in the PassengerData.txt file.

```java
public void searchUser(String ID) {
    boolean isFound = false;
    Customer customerWithTheID = customerCollection.get(0);
    for (Customer c : customerCollection) {
        if (ID.equals(c.getUserID())) { // if ID equals the ID in the customerCollection list
            System.out.printf("%-50sCustomer Found...!!!Here is the Full Record...!!!\n\n\n", " ");
            displayHeader();
            isFound = true;
            customerWithTheID = c;
            break;
        }
    }
    if (isFound) {
        System.out.println(customerWithTheID.toString( i: 1));
        System.out.printf("%10s+------------+------------+---------------------------------+------
    } else {
        System.out.printf("%-50sNo Customer with the ID %s Found...!!!\n", " ", ID);
    }
}
```

This method is used to search for a user. It uses a for loop to iterate through the customerCollection list. If it finds the ID in the list, then it will show the user data, if it doesn't, then it will print an error message.

```java
public boolean isUniqueData(String emailID) {
    boolean isUnique = false;
    for (Customer c : customerCollection) { // searches through each Customer
        if (emailID.equals(c.getEmail())) {
            isUnique = true; // returns true if emailID inputted is the same as email in list
            break;
        }
    }
    return isUnique;
}
```

This method is used to check whether or not the email inputted is the same in the list. It searches through the customerCollection list.

```java
public void editUserInfo(String ID) {
    boolean isFound = false;
    Scanner read = new Scanner(System.in);
    for (Customer c : customerCollection) { // search every Customer for their ID
        if (ID.equals(c.getUserID())) {
            isFound = true;
            System.out.print("\nEnter the new name of the Passenger: ");
            String name = read.nextLine();
            c.setName(name); // set new name
            System.out.print("Enter the new email address of Passenger " + name + ": ");
            c.setEmail(read.nextLine()); // set new email
            System.out.print("Enter the new Phone number of Passenger " + name + ": ");
            c.setPhone(read.nextLine()); // set new phone number
            System.out.print("Enter the new address of Passenger " + name + ": ");
            c.setAddress(read.nextLine()); // set new address
            System.out.print("Enter the new age of Passenger " + name + ": ");
            c.setAge(read.nextInt()); // set new age
            displayCustomersData( showHeader: false);
            break;
        }
    }
    if (!isFound) { // if found then will print out message
        System.out.printf("%-50sNo Customer with the ID %s Found...!!!\n", " ", ID);
    }
}
```

This method is used to edit a Passenger's data. It loops through the customersCollection list to find the ID inputted, and the Passenger will be prompted to input their new data.

```java
public void deleteUser(String ID) {
    boolean isFound = false;
    Iterator<Customer> iterator = customerCollection.iterator(); // iterator to modify the whole list
    while (iterator.hasNext()) { // as long as there's an element to iterate to next
        Customer customer = iterator.next(); // makes a Customer object using the next iteration
        if (ID.equals(customer.getUserID())) {
            isFound = true; // if Customer found, set to true
            break;
        }
    }
    if (isFound) { // if isFound is true
        iterator.remove(); // remove an element from the customerCollection
        System.out.printf("\n%-50sPrinting all  Customer's Data after deleting Customer with the ID %s.....!!!!\n", "", ID);
        displayCustomersData( showHeader: false);
    } else { // if false, print out message
        System.out.printf("%-50sNo Customer with the ID %s Found...!!!\n", " ", ID);
    }
}
```

This method is used to delete a Passenger. It uses the Iterator library to modify the whole list. If the ID is found, then it will remove all of the Customer data from the customersCollection list.

```
String randomIDDisplay(String randomID) {
    StringBuilder newString = new StringBuilder(); // stringbuilder to manipulate strings
    for (int i = 0; i <= randomID.length(); i++) { // loop through the length of the id
        if (i == 3) { // add an empty space to the fourth character in randomID
            newString.append(" ").append(randomID.charAt(i));
        } else if (i < randomID.length()) { // if it isn't the 3rd index yet
            newString.append(randomID.charAt(i));
        }
    }
    return newString.toString(); // return a string id with the added space
}
```

This method is used to add a space between the generated userID to make it easier to read.

```
void addExistingFlightToCustomerList(int index, int numOfTickets) {
    // index represents position of flight in the list
    // numOfTickets represents number of tickets to be added
    int newNumOfTickets = numOfTicketsBookedByUser.get(index) + numOfTickets; // add number of new tickets to the current list
    this.numOfTicketsBookedByUser.set(index, newNumOfTickets); // updates the numOfTicketsBookedByUser list
}
```

This method is used to add tickets to an already booked flight.

RolesAndPermissions.java

```
public int isPrivilegedUserOrNot(String username, String password) {
    int isFound = -1; // return -1 if admin not found
    for (int i = 0; i < adminUserNameAndPassword.length; i++) { // loop through each letter of username and password
        if (username.equals(adminUserNameAndPassword[i][0])) {
            if (password.equals(adminUserNameAndPassword[i][1])) {
                isFound = i;
                break;
            }
        }
    }
    return isFound;
}
```

```java
public String isPassengerRegistered(String email, String password) {
    String isFound = "0"; // return 1 with user ID if passenger is registered
    for (Customer c : Customer.customerCollection) { // loop to check each customer in customerCollection
        if (email.equals(c.getEmail())) {
            if (password.equals(c.getPassword())) {
                isFound = "1-" + c.getUserID();
                break;
            }
        }
    }
    return isFound;
}
```

Both these methods are used to check whether or not an Admin or Passenger is registered. For Admin, it checks through the adminUserNameAndPassword ArrayList, while for Passenger, it looks through the customerCollection list.

PassengerReader.java

```java
public class PassengerReader {
    1 usage
    public static void createPassenger() throws IOException {
        File file = new File( pathname: "C:/Users/edely/Downloads/AirLineReservationSystem/AirLineReservationSystem/src/PassengerData.txt");
        BufferedReader buffer = new BufferedReader(new FileReader(file));
        String txt;
        ArrayList<String> txtArray = new ArrayList<>();
        while((txt = buffer.readLine()) != null){
            txtArray.add(txt);
        }

        for(int i = 0; i < txtArray.size(); i += 6){
            String name = txtArray.get(1 + i);
            String email = txtArray.get(2 + i);
            String password = txtArray.get(3 + i);
            String phone = txtArray.get(4 + i);
            String address = txtArray.get(5 + i);
            int age = Integer.parseInt(txtArray.get(6 + i));
            i++;
            Customer.customerCollection.add(new Customer(name, email, password, phone, address, age));
        }
    }
}
```

The method createPassenger is used to add the data into the txt file. It initializes a string txt, and an ArrayList called txtArray. If the string isn't empty, then it will add whatever is in the string into the ArrayList.

The for loop is used to determine what data is in each line and add it to the customersCollection list. Since "Passenger:" is at index 0 (added for readability), index 1 will be the name, 2 will be

the email, etc. And the final results will be added to the list. It skips every 6 indexes so that it always starts from the beginning. (see below)

PassengerData.txt

```
Passenger:
edelyne
email
pass
2434545
senayan
24
```

Flight.java

```
public void flightScheduler() {
    int numOfFlights = 15; // decides how many unique flights to be included/display in scheduler
    RandomGenerator r1 = new RandomGenerator();
    for (int i = 0; i < numOfFlights; i++) { // loop for 15 times
        String[][] chosenDestinations = r1.randomDestinations();
        // calculate distance using the latitude and longitude values of the chosen destinations
        String[] distanceBetweenTheCities = calculateDistance(Double.parseDouble(chosenDestinations[0][1]), Double.parseDouble(chosenDestinations[0][2]), Double.parseDouble(chosenDestinations[1]
        String flightSchedule = createNewFlightsAndTime(); // generates new flight and the time
        String flightNumber = r1.randomFlightNumbGen( uptoHowManyLettersRequired: 2,  divisible: 1).toUpperCase();
        int numOfSeatsInTheFlight = r1.randomNumOfSeats(); // generate random number of seats
        String gate = r1.randomFlightNumbGen( uptoHowManyLettersRequired: 1,  divisible: 30); // generates gate with parameters being the length of gate number and maximum gate number is 30
        // a flight object is created with all those details, gate is converted to uppercase
        flightList.add(new Flight(flightSchedule, flightNumber, numOfSeatsInTheFlight, chosenDestinations, distanceBetweenTheCities, gate.toUpperCase()));
    }
}
```

This method generates a specified number of unique flights by randomly selecting destinations, calculating distances, generating flight schedules, flight numbers, number of seats, and gate numbers. It creates new Flight objects and adds them to a flightList.

```
boolean isCustomerAlreadyAdded(List<Customer> customersList, Customer customer) { // customerList of flight, specified customer to be checked
    boolean isAdded = false;
    for (Customer customer1 : customersList) {
        if (customer1.getUserID().equals(customer.getUserID())) {
            isAdded = true; // return true if registered
            customerIndex = customersList.indexOf(customer1);
            break;
        }
    }
    return isAdded;
}
```

This method checks whether a Passenger is already present in a given list of customers.

```java
public String calculateFlightTime(double distanceBetweenTheCities) { // distance between cities in miles
    double groundSpeed = 450;
    double time = (distanceBetweenTheCities / groundSpeed);
    String timeInString = String.format("%.4s", time); // formats the time as a string with four characters
    String[] timeArray = timeInString.replace( oldChar: '.',  newChar: ':').split( regex: ":"); // replace decimal point with a colon, and the colon is used to divide the string
    int hours = Integer.parseInt(timeArray[0]); // array[0] contains the hours
    int minutes = Integer.parseInt(timeArray[1]); // array[1] contains the minutes
    int modulus = minutes % 5;
    // Changing flight time to make minutes near/divisible to 5.
    if (modulus < 3) {
        minutes -= modulus;
    } else {
        minutes += 5 - modulus;
    }
    if (minutes >= 60) { // if minutes exceed 60, then reduce it by 60 and add the hour by 1
        minutes -= 60;
        hours++;
    }
    if (hours <= 9 && Integer.toString(minutes).length() == 1) { // return formatted flight time
        return String.format("0%s:%s0", hours, minutes); // if hour less than 9 and minutes is single digit, time is formatted as 0H:0M
    } else if (hours <= 9 && Integer.toString(minutes).length() > 1) {
        return String.format("0%s:%s", hours, minutes); // time is formatted as 0H:MM
    } else if (hours > 9 && Integer.toString(minutes).length() == 1) {
        return String.format("%s:%s0", hours, minutes); // time is formatted as HH:0M
    } else {
        return String.format("%s:%s", hours, minutes); // time is formatted as HH:MM
    }
}
```

This method calculates the estimated flight time based on the distance between two cities, it then also formats the time as hours and minutes.

```java
public String fetchArrivalTime() {
    // format the time as for example: Wednesday, 12 July 2023, 08:45 PM
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE, dd MMMM yyyy, HH:mm a ");
    // convert the String of flightSchedule to LocalDateTime and add the arrivalTime to it
    LocalDateTime departureDateTime = LocalDateTime.parse(flightSchedule, formatter);

    // Getting the Flight Time, plane was in air
    String[] flightTime = getFlightTime().split( regex: ":");
    int hours = Integer.parseInt(flightTime[0]);
    int minutes = Integer.parseInt(flightTime[1]);


    LocalDateTime arrivalTime;

    // add departure time by the hours it takes for the flight for the arrival time
    arrivalTime = departureDateTime.plusHours(hours).plusMinutes(minutes);
    DateTimeFormatter formatter1 = DateTimeFormatter.ofPattern("EE, dd-MM-yyyy HH:mm a");
    return arrivalTime.format(formatter1); // return flight arrival time

}
```

This method calculates the arrival time of a flight based on the departure time and flight duration. It also adds the departure time with the hours it takes on the flights to get the arrival time.

```
void deleteFlight(String flightNumber) {
    boolean isFound = false;
    Iterator<Flight> list = flightList.iterator();
    while (list.hasNext()) { // make object flight for every item in list
        Flight flight = list.next();
        if (flight.getFlightNumber().equalsIgnoreCase(flightNumber)) { // if flight id inputted is in flight list, return true
            isFound = true;
            break;
        }
    }
    if (isFound) { // remove the flight from the list
        list.remove();
    } else {
        System.out.println("Flight with given Number not found...");
    }
    displayFlightSchedule();
}
```

This method removes a flight from a list of flights based on the flight number. It loops through the list to find the number (which ignores case sensitivity).

```
@Override
public String[] calculateDistance(double lat1, double lon1, double lat2, double lon2) {
    double theta = lon1 - lon2;
    // equation to calculate the distance
    double distance = Math.sin(degreeToRadian(lat1)) * Math.sin(degreeToRadian(lat2)) + Math.cos(degreeToRadian(lat1)) * Math.cos(degreeToRadian(lat2)) * Math.cos(degreeToRadian(theta));
    distance = Math.acos(distance);
    distance = radianToDegree(distance);
    distance = distance * 60 * 1.1515;
    String[] distanceString = new String[3];
    distanceString[0] = String.format("%.2f", distance * 0.8684); // distance in miles
    distanceString[1] = String.format("%.2f", distance * 1.609344); // distance in km
    distanceString[2] = Double.toString( d: Math.round(distance * 100.0) / 100.0); // distance in knots
    return distanceString; // return distance both in miles and km between the cities/airports
}
```

This method calculates the distance between two cities using the haversine formula. The end result is an array containing the distance in miles, km and knots.

```java
public String createNewFlightsAndTime() {

    Calendar c = Calendar.getInstance();
    // incrementing nextFlightDay, so that next scheduled flight would be in the future, not in the present
    nextFlightDay += Math.random() * 7;
    c.add(Calendar.DATE, nextFlightDay);
    c.add(Calendar.HOUR, nextFlightDay);
    c.set(Calendar.MINUTE, ((c.get(Calendar.MINUTE) * 3) - (int) (Math.random() * 45)));
    Date myDateObj = c.getTime();
    LocalDateTime date = Instant.ofEpochMilli(myDateObj.getTime()).atZone(ZoneId.systemDefault()).toLocalDateTime();
    date = getNearestHourQuarter(date);
    return date.format(DateTimeFormatter.ofPattern("EEEE, dd MMMM yyyy, HH:mm a "));
}
```

This method generates a new flight schedule by calculating a future date and time for the flight.

```java
public LocalDateTime getNearestHourQuarter(LocalDateTime datetime) {
    int minutes = datetime.getMinute();
    int mod = minutes % 15;
    LocalDateTime newDatetime;
    if (mod < 8) {
        newDatetime = datetime.minusMinutes(mod); // subtract the datetime with the mod value to round it down to the prev quarter
    } else {
        newDatetime = datetime.plusMinutes(15 - mod); // add datetime with the result of 15 - mod to round it up to the next quarter
    }
    newDatetime = newDatetime.truncatedTo(ChronoUnit.MINUTES); // only hour and minute components remain
    return newDatetime; // returns formatted LocalDateTime with minutes close to the nearest hour quarter
}
```

This method formats the flight schedule so that the minutes would be to the nearest quarter, like 15, 30, 45, etc.

FlightReservation.java

```
void bookFlight(String flightNo, int numOfTickets, String userID) { // flightNo = the flight id. numOfTickets = number of tickets to be booked. userID = user's id
    boolean isFound = false;
    for (Flight f1 : flight.getFlightList()) { // find the Flight list
        if (flightNo.equalsIgnoreCase(f1.getFlightNumber())) { // get the flight number
            for (Customer customer : Customer.customerCollection) {
                if (userID.equals(customer.getUserID())) {
                    isFound = true;
                    f1.setNoOfSeatsInTheFlight(f1.getNoOfSeats() - numOfTickets); // reduce number of seats by tickets bought
                    if (!f1.isCustomerAlreadyAdded(f1.getListOfRegisteredCustomersInAFlight(), customer)) { // if new Passenger registers for flight
                        f1.addNewCustomerToFlight(customer); // adds Passenger to that flight
                    }
                    if (isFlightAlreadyAddedToCustomerList(customer.flightsRegisteredByUser, f1)) { // if Passenger is already added to flight
                        addNumberOfTicketsToAlreadyBookedFlight(customer, numOfTickets);
                        if (flightIndex(flight.getFlightList(), flight) != -1) { // if flight index isn't -1
                            customer.addExistingFlightToCustomerList(flightIndex(flight.getFlightList(), flight), numOfTickets); // updates numOfTickets
                        }
                    } else {
                        customer.addNewFlightToCustomerList(f1); // add to flights registered by user
                        addNumberOfTicketsForNewFlight(customer, numOfTickets);
                    }
                break;
                }
            }
        }
    }
    if (!isFound) { // if id is not found
        System.out.println("Invalid Flight Number...! No flight with the  ID \"" + flightNo + "\" was found...");
    } else {
        System.out.printf("\n %50s You've booked %d tickets for Flight \"%5s\"...", "", numOfTickets, flightNo.toUpperCase());
    }
}
```

This method is used to book a number of tickets for a user. It finds the flight based on the flight number given, and then adds the Passenger to said flight. It also updates the number of available seats in the flight based on how many tickets the Passenger bought.

```
void cancelFlight(String userID) {
    String flightNum = ""; // flight number of cancelled flight
    Scanner read = new Scanner(System.in);
    int index = 0, ticketsToBeReturned;
    boolean isFound = false;
    for (Customer customer : Customer.customerCollection) {
        if (userID.equals(customer.getUserID())) { // if user id inputted is found in the list of passenger user id
            if (customer.getFlightsRegisteredByUser().size() != 0) { // as long as the customer flights is not 0
                System.out.printf("%50s %s Here is the list of all the Flights registered by you %s", " ", "+++++++++++++++", "+++++++++++++++");
                displayFlightsRegisteredByOneUser(userID);
                System.out.print("Enter the Flight Number of the Flight you want to cancel : ");
                flightNum = read.nextLine();
                System.out.print("Enter the number of tickets to cancel : ");
                int numOfTickets = read.nextInt();
                Iterator<Flight> flightIterator = customer.getFlightsRegisteredByUser().iterator(); // look through the flights registered by Customer
                while (flightIterator.hasNext()) {
                    Flight flight = flightIterator.next();
                    if (flightNum.equalsIgnoreCase(flight.getFlightNumber())) { // if flight number inputted is the same as the flight number already in list
                        isFound = true;
                        int numOfTicketsForFlight = customer.getNumOfTicketsBookedByUser().get(index);
                        while (numOfTickets > numOfTicketsForFlight) { // if tickets that the Passenger wants to delete is more than the number of tickets that is available
                            System.out.print("ERROR!!! Number of tickets cannot be greater than " + numOfTicketsForFlight + " for this flight. Please enter the number of tickets again : ");
                            numOfTickets = read.nextInt();
                        }
                        if (numOfTicketsForFlight == numOfTickets) { // if the same
                            ticketsToBeReturned = flight.getNoOfSeats() + numOfTicketsForFlight; // add the returned tickets to the number of seats
                            customer.numOfTicketsBookedByUser.remove(index); // remove the number of tickets ordered
                            flightIterator.remove();
                        } else {
                            ticketsToBeReturned = numOfTickets + flight.getNoOfSeats(); // if different
                            customer.numOfTicketsBookedByUser.set(index, (numOfTicketsForFlight - numOfTickets)); // update the amount of tickets booked by the user
                        }
                        flight.setNoOfSeatsInTheFlight(ticketsToBeReturned); // return the available number of tickets back to normal
                        break;
                    }
                    index++;
                }
```

```
            }else{
                System.out.println("No Flight Has been Registered by you with ID \"\" + flightNum.toUpperCase() +"\"\".....");
            }
            if (!isFound) {
                System.out.println("ERROR!!! Couldn't find Flight with ID \"" + flightNum.toUpperCase() + "\".....");
            }
        }
    }
}
```

This method is used to cancel a flight and returns the number of tickets back to the main flight schedule. First, it searches for the customer in the customersCollection list, and then looks for the flights that they've registered, and then asks for an input for what flight they want to remove. If the flight ID is found, then it will ask how many tickets they want to return, and it will update the flight schedule accordingly.

```
void addNumberOfTicketsToAlreadyBookedFlight(Customer customer, int numOfTickets) {
    int newNumOfTickets = customer.numOfTicketsBookedByUser.get(flightIndexInFlightList) + numOfTickets;
    customer.numOfTicketsBookedByUser.set(flightIndexInFlightList, newNumOfTickets); // set the new number of tickets to the flight that got new tickets
}
```

This method is used to add the number of tickets to an already booked flight for a Passenger.

```
void addNumberOfTicketsForNewFlight(Customer customer, int numOfTickets) {
    customer.numOfTicketsBookedByUser.add(numOfTickets);
}
```

This method is used to add the number of tickets to a brand new flight.

```
boolean isFlightAlreadyAddedToCustomerList(List<Flight> flightList, Flight flight) {
    boolean addedOrNot = false;
    for (Flight flight1 : flightList) { // loops through the flightList
        if (flight1.getFlightNumber().equalsIgnoreCase(flight.getFlightNumber())) { // if flight number same as flightNumber is registered flight
            this.flightIndexInFlightList = flightList.indexOf(flight1); // get the index of flight1 in flightList
            addedOrNot = true;
            break;
        }
    }
    return addedOrNot;
}
```

This method is used to check if a flight is already added to a Passenger's list. It loops through the flightList to check if the flight number given is the same as the flight in the list, if it is, then it returns true, if the flight is not found, it returns false.

```java
String flightStatus(Flight flight) {
    boolean isFlightAvailable = false;
    for (Flight list : flight.getFlightList()) {
        if (list.getFlightNumber().equalsIgnoreCase(flight.getFlightNumber())) { // see if flight number is in the list
            isFlightAvailable = true;
            break;
        }
    }
    if (isFlightAvailable) {
        return "As Per Schedule";
    } else {
        return "   Cancelled   ";
    }
}
```

This method is used to determine the status of a flight. It checks if the flight is in the flightList, if it's not found, then it's canceled, but if it is, then it's as per schedule.

```java
@Override
public void displayFlightsRegisteredByOneUser(String userID) { // show all flights registered by user
    System.out.println();
    System.out.print("+------+-----------------------------------------+----------+----------------+-----------------------+-----------------------+-----------------------+----------
    System.out.printf("| Num  | FLIGHT SCHEDULE\t\t\t\t\t\t   | FLIGHT NO |  Booked Tickets  | \tFROM ====>>       | \t====>> TO\t   | \t    ARRIVAL TIME       | FLIGHT TIME |  GATE  |  FLIGHT S
    System.out.print("+------+-----------------------------------------+----------+----------------+-----------------------+-----------------------+-----------------------+----------
    for (Customer customer : Customer.customerCollection) {
        List<Flight> f = customer.getFlightsRegisteredByUser(); // make a list that contains all the flight registered by a user
        int size = customer.getFlightsRegisteredByUser().size();
        if (userID.equals(customer.getUserID())) { // if inputted id is the same as the available customer ID
            for (int i = 0; i < size; i++) {
                System.out.println(toString((i + 1), f.get(i), customer));
                System.out.print("+------+-----------------------------------------+----------+----------------+-----------------------+-----------------------+-----------------------+----------
            }
        }
    }
}
```

This method prints out all the flights registered by a specific Passenger. It loops through the customersCollection list to get the flights registered by a specific Passenger. If the inputted ID is the same as the ID in the list, then it will print out the flight information and add a line to separate each row.

```java
@Override
public void displayRegisteredUsersForAllFlight() {
    System.out.println();
    for (Flight flight : flight.getFlightList()) {
        List<Customer> c = flight.getListOfRegisteredCustomersInAFlight();
        int size = flight.getListOfRegisteredCustomersInAFlight().size();
        if (size != 0) {
            displayHeaderForUsers(flight, c);
        }
    }
}
```

This method is used to display the registered Passengers for all flights. It loops through each flight in flightList, and then gets the list of registered Passengers for the current flight. If the list isn't empty, then it will display the information of all the Passengers on that flight.

```java
int flightIndex(List<Flight> flightList, Flight flight) {
    int i = -1; // -1 so index 2 will give the 2nd element, not the third one
    for (Flight flight1 : flightList) {
        if (flight1.equals(flight)) {
            i = flightList.indexOf(flight1);
        }
    }
    return i;
}
```

This method is used to find the index of a specific flight within a list of flights. It returns the index if the flight is found in the list, or -1 if it is not found.

```java
@Override
public void displayRegisteredUsersForASpecificFlight(String flightNum) { // display all the users that have booked a specific flight
    System.out.println();
    for (Flight flight : flight.getFlightList()) { // loops through the flight list
        List<Customer> c = flight.getListOfRegisteredCustomersInAFlight(); // make a list filled with all the customers in a flight
        if (flight.getFlightNumber().equalsIgnoreCase(flightNum)) { // find the flight number
            displayHeaderForUsers(flight, c);
        }
    }
}
```

This method is used to display all the Passengers that have booked a specific flight. It loops through the flightList, finds the specific flight based on the flight number, gets the list of registered customers for that flight, and shows the header and customer information using the displayHeaderForUsers method.

5. Lesson Learned

As my first major project in Java, I had a lot of trouble making it along the way. I had a hard time because my skills at the beginning were still new and I didn't know a lot about its functions, and I encountered a lot of errors along the way. For example, I couldn't figure out how to put the data into a txt file at first, as I wasn't sure if the feature existed in the first place. Another problem I encountered was with the flight time. I wasn't sure if I wanted to use it as it was something out of my comfort zone, but with the help of my friends, I managed to push through it and make it work.

Overall, after finishing it, I felt satisfied with myself because I was able to make a project that I was proud of, and it taught me a lot about Object Oriented Programming, and Java as a language in general. When I have time in the future, I want to perfect this project by adding GUI and other error handlings and really push my knowledge of the language.