Airline Reservation System

Proposal

Rashed Al Fuhed
October 29th, 2021

Purpose

This project is intended to demonstrate mastery of concepts learned in Computer Science, specifically, Object-Oriented Programming, use of data structures, and use of abstract data types.

Overview

Airline Reservation system is a system for making flight tickets for customers online! There will be two users: Admin and Customer. The admin is the one who takes care of the tickets and customers record, while the customer is the one who books a ticket for a flight.

Airline Classes

The Flight class will be an interface, describing elements common to all classes. There will be 3 classes, and probably 1-2 of them are implementations of the interface.

The User class will hold the records of the customer (e.g., name, ID, etc.). There will be two different menus, one for the admin, and one for the customer. The admin will have more features than the customer does, and the customer will have features that are not available to the admin (e.g., booking).

Reservation class is where the program will be mostly working on. It will take the User class for the records and the menus of each user, and then will be implemented inside a while loop and if-statements. I will also use Array List for the users and for their records so it can make the implementation easier for me and for the program.

Anticipated Use Case Analysis

The final version will have two use cases: Admin will use the editing system to manage and list each customer's record. The customer will use the system to book, edit, and cancel a ticket.

Data Design

The project Is ideally suited to Object-Oriented Programming. It features 2-3 classes and an interface (for now). It illustrates mostly all the major features of OOP:

Encapsulation – Most of the data members are hidden and accessible only though methods.

Polymorphism – Some methods and attributes in the User class are accessible (e.g., isAdmin), and they will be implemented in inherited classes (Reservation class).

Inheritance – There will be 1 class that inherit from the interface.

Abstraction – By definition, an interface is abstract, and the AirlineSystem class represents an interface.

Composition – Reservation class holds one instance from User and one from Flight

Array – Both User class and Flight class instance is an ArrayList inside the Reservation class.

Serialization – The ticket will be serializable and will be implemented in Reservation class

File I/O – The ticket is going to print out when the user is done in a text file

Algorithm

The system will be written in Java. Some features in Java are easier and less time consuming than C++. The program will be implemented in Tesla command-line, with a makefile to run the program.

Goal: Implement a program that simulate Airline reservation system

Input: Menu, menu's options, and log in

Output: Ticket, Menu, and Log in

Steps:

1) Begin creating the flight class, which simple will be the info about the flight such as airline, source, etc.
2) Make 3 private strings and 1 int
3) Build a constructor that hold all of them
4) Now make setters and getters for each
5) Create another class called User
6) Make 5 strings and 1 boolean. Note that Boolean should be accessible through other classes
7) Build a constructor that holds all of the data members and read them
8) Now make setter and getter for each one
9) Create a string that holds the user menu
10) Make it have a total of 5 choices and set an appropriate input
11) Same goes with admin menu but different options
12) Now make an interface that holds the ticket actions such as printing, editing, saving, loading, etc.
13) Finally, make the reservation class which will be the main class
14) Set an ArrayList for both User and Flight instances
15) Now build a constructor, and add to the user's array a random user's or admin's info
16) Output the Log in gate
17) Initialize a while loop, and inside it, make a for loop for all users, and put a condition if the the inputted user name and password is the same of anything up, then the condition is true
18) Also, put a condition if the user is admin, then display the admin's menu
19) If user chose #1, then print out the first and last name of each user
20) If user chose #2, then list all of the users' info
21) If user chose #3, then ask for the wanted info to add
22) If user chose #4, then ask for the user's username required to be removed
23) If user chose #5, then show the admin's profile
24) If user chose #6, then stop the loop
25) If the user's a customer, then display the user's menu
26) If user chose #1, then ask for the source, departure, and airline
27) When finished, generate a random ticket number by using the Random from the Util package
28) Set it to 6 digits and print it out
29) Add the info to the flight array
30) Lastly, print and save ticket in a text file
31) If user chose #2, Ask for the ticket number required to be canceled
32) If true, use the iterator and iterate the flight arrayand if matched, remove it
33) Else, tell the user
34) If user chose #3, Show the profile
35) If user chose #4, then use the method from the interface, editTicket() and loadTicket

36) The goal of this method is to edit the ticket, if found
37) Input is the info the ticket, output is the questions of the ticket info
38) Generate a new random number after these are true and add to the array
39) If user chose #5, stop the loop
40) Now with the interface methods, printTicket is where obviously printing the ticket ina text file.
41) Use the FileWriter and printWriter, and print all of the user's tickets details, then close the file

UI Design

There will be a log in gate. If the user is an admin, the system will know by the entered username and password. Same goes for the customer. If the user is unknown, the system will display an error message.

Admin menu

1. List all customers (first and last name only)
2. Show customers' records.
3. Add a new customer.
4. Delete a customer's account.
5. Log out.

Customer menu

1. Book a flight ticket.
2. Cancel a flight ticket.
3. Show profile.
4. Edit ticket.
5. Log out.

UML Diagram


User
-firstName: string
-lastName: string
-ID: string
+User(String,String,String,String,String, bool)
+getFirstName() : string
+setFirstName(string) : void
+getLastName() : string
+setLastName(string) : void
+getID() : string
+setID(string) : void
+userMenu() : void
+adminMenu() : void
Flight
-airline: string
-source: string
-departure: string
-ticketNum: int
+Flight()
+getTicketNum() : int
+getAirline() : string
+setAirline(string) : void
+getDeparture() : string
+setDeparture(string) : void
+getSource() : string
+setSource(string) : void
Reservation
+Reservation()
+main(string[] args)
+printTicket() : void
+editTicket() : void
+saveTicket() : void
+loadTicket() : void
+serial() : void
AirlineSystem
(interface)