Denisse Loya Villalobos

March 25, 2023
CS 3331 – Advanced Object-Oriented Programming – Spring 2023
Instructor: Dr. Daniel Mejia
Programming Assignment 3

This work was done individually and completely on my own. I did not share, reproduce, or alter any part of this assignment for any purpose. I did not share code, upload this assignment online in any form, or view/received/modified code written from anyone else. All deliverables were produced entirely on my own. This assignment is part of an academic course at The University of Texas at El Paso and a grade will be assigned for the work I produced.

1. Program Explanation

The assignment was to create an airline company called Miner Air we were given a CSV file with all the flights, their respective IDs, and schedules, the difference this time it was shuffled. The task was to read the file without using hardcode, create a flight class, read thefile, make an object, store them in the appropriate data structure, and display a menu, so the user can interact with the Airline.
My approach was to break down the problem, starting with the new attributes for the csv file, and add them to the constructors, getters, and setters. After I used a HashMap to find the index for the flight attribute. Then used the factory to create objects for each type of flight. Next, I designed a menu using the new attributes, an letting the employees purchase a flight.

2. What did I learn?

From this assignment, I got to challenge myself to think about a creative idea to read the file without hardcoding the row index, I had a couple of ideas, and implemented the one that felt like the best approach. My solution could be improved with debugging. I would have to spend more days cleaning the code and maybe finding an alternative that did not use as much memory space.

3. Solution Design

First, I created the flight class with the new information from the file header after this using another design pattern called “factory”, to check if the type of flight was International or Domestic. After this, I imported the HashMap library, and use one to store the headers as keys, then with a loop I found the index of the row and put it as the value then used the scanner again to create all the objects. Then, focused on editing all the transaction class, to add the employee discount and the fees and taxes. At the end all I did was to test the code and fix the errors I had from previous assignments.

4. Testing

I tested my program using the white box method because I already knew how the methods worked, so I designed some test cases, according to the code. After, I asked a friend to test my code using the black box method, in order to improve the user experience.

Test Case 1:
Username: belledisney, Customer
ID: 1
Quantity: 1, Main Class
Transaction: No discount, security fee 5.60, minerairlines fee 9.15, and tax 8.25%

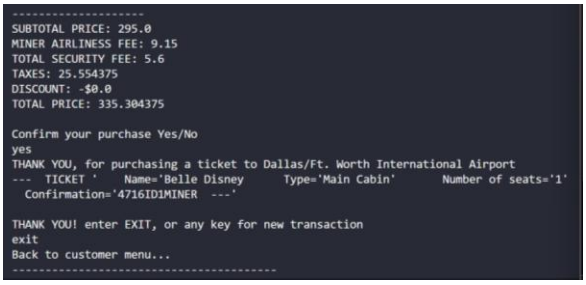
Test Case 2:
Username: enriquerico, Customer
ID: 1
Cancel flight

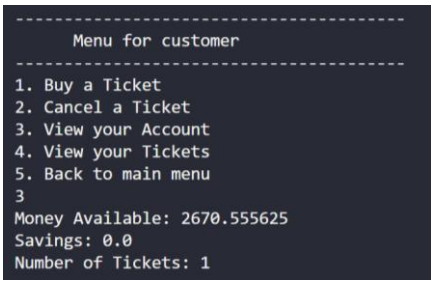
Test Case 3:
Username: belledisney, Customer
Check ticket was cancelled, and total refund

The first time testing my program could not handle the error of introducing a non-existing ID, so I used that to improve it.

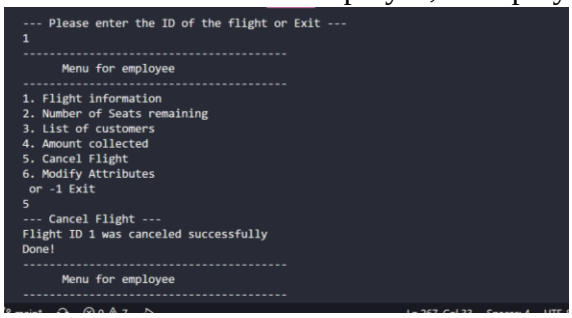
5. Test results

1. Test case to see if taxes were applied and if a customer has no membership then no discount

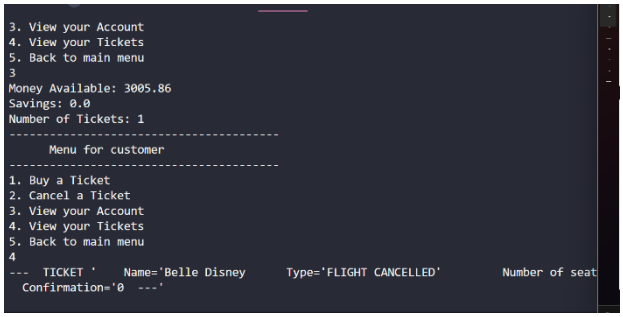




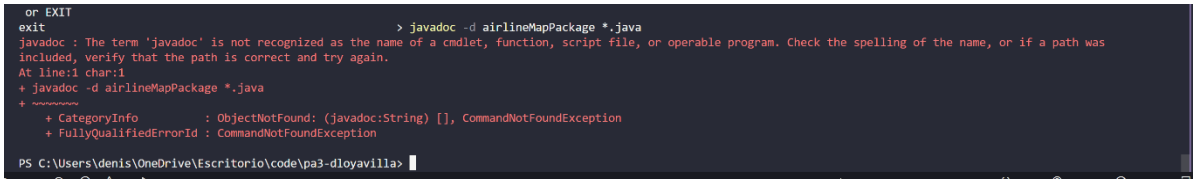
2. Test case for menu employee, if employee cancels flight then complete refund.



3. Test case for total refund of ticket and change of ticket status, number of tickets remains the same but status and accessibility has changed to cancelled.



Note: My JavaDoc couldn’t be created.



6. Code Review

Implementation

- Does this code do what it is supposed to? Yes, the code is designed to read the file without hardcode
- Can it be simplified? My guess is yes, but at the moment I can't think of any ideas.
- Is the code dynamic or hardcoded? Dynamic
- Is the code maintainable? Yes, it only needs to be cleaned.

Logic

- Cases where code does not behave as expected/intended? I couldn't find a case where the code didn't behave as expected.
- Test cases where it may fail? It wouldn't fail but some print statements can be improved to be more user friendly

Readability/Style

- Easy to read/understand? Yes
- What parts can be modified or adjusted? I think the code readability is fine
- Is the structure appropriate? Yes
- Does it follow the appropriate language style? Yes
- Is the code well documented? Yes

Performance

- What is the code complexity? O(n)
- How does the complexity change with various inputs? If the list of airports increases the time will increase