## Brandon Cartwright

## April 4, 2023

CS 3331 – Advanced Object-Oriented Programming – Spring 2023

## Dr. Mejia

## PA4

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.

# **Program Explanation**

In this programming assignment a few new features were implemented into Miner Airlines. First is a search functionality for customers to search by airports rather than just flight ID. Now the customer can search for a flight by origin and destination airport codes. Once the user enters the airport codes the program loops through the flights checking and displaying all airports that match the criteria. The program also gives the option for the user to enter a departure date making the criteria more specific. For the next feature, employees now have the option to inquire about an airport. They do this by entering the airport code which is used to look up airports that are stored in a HashMap. The HashMap has the airport code as the key and the airport objects as the value. When employees inquire about the airport the program prints out all the information regarding the airport. Another new feature that has been implemented is the auto purchaser. We were given excel spreadsheets of customers with flight purchase attributes. The program reads through the csv file the same way as the flights in previous PAs. After getting a row of information from the csv the program sends the information to a purchase method that goes through the same steps as when customers purchase a ticket. This means the auto purchaser follows the same constraints such as the customer must have enough money and the flight must have enough seats. The last new feature added is the electronic ticket summary. The employee has the option to write an electronic ticket summary for three customers. This is done using a for loop for each ticket the customer has and prints out all information on the ticket.

# **What did I learn?**

In this programming assignment I have gotten more comfortable with dealing with objects inside of objects. For example, each flight has two airport objects stored inside of them, customers have ticket objects inside, and ticket objects have flights stored inside. Doing this makes getting connected information a lot easier since everything relates to each other. I gained experience in implementing new functionality based on user feedback. I learned how to incorporate customer and employee requests for new features into the existing system design, and how to test and iterate on these features to ensure they function properly. Adding search functionality to the system helped me to better understand how to organize and search through large amounts of data. The addition of the auto purchaser and electronic ticket summary features helped me to develop my understanding of how to automate certain aspects of the system and make it more user-friendly for customers. Overall, this update provided me with a valuable learning experience and helped me to further develop my coding and problem-solving skills. Because is used a lot of HashMaps. I gained a better understanding of how they work, as well as how to utilize them in other programming projects. The use of HashMaps in my flight purchasing system helped me to develop my programming skills and further understand data management.

# **Solution Design**

# I was required to add new functionalities based on customer and employee feedback. To fulfill this requirement, I first identified the key features that needed to be added to the system, which included a search function for customers to find flights based on origin and destination airports, as well as an airport information inquiry for employees. To implement these functionalities, I used different data structures such as hashmaps and arrays. In my system, I used hashmaps to efficiently store and retrieve customer and flight information. I also utilized arrays to hold airport information and flight schedules. During the development of PA4, I made certain assumptions about the system's functionality, such as assuming that the airport information and flight schedules would be accurately reflected in the csv file that the program read from. I also assumed that the search function would return accurate results based on the input provided by the customer. Overall, my approach to solving the problem for PA4 involved meeting the requirement of adding new functionalities to the system based on user feedback, utilizing appropriate data structures to optimize storage and retrieval of data, and making certain assumptions about the system's functionality to facilitate development and testing.

# **Testing**

# I tested my flight purchasing system using a combination of white-box and black-box testing techniques. To perform white-box testing, I analyzed the internal structure of the program, focusing on specific functions and methods to ensure that they were functioning properly. I created test cases to check that the hashmaps and arrays were correctly storing and retrieving data, and that the new functionalities were integrated into the existing code correctly. For black-box testing, I evaluated the system from the perspective of a user, creating test cases to focus on the external behavior of the program. For example, I tested the search function by entering different airport combinations and verifying that the correct flights were returned. I also tested the airport information inquiry to make sure that all relevant information was displayed in the output. By using a combination of white-box and black-box testing, I was able to thoroughly test my program for PA4, ensuring that it met the requirements for the new functionalities and that it was functioning correctly.

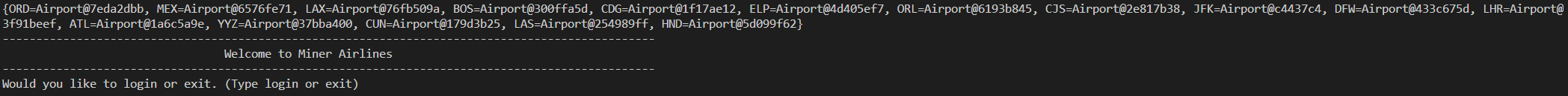
# **Test result**

In the first screenshot I was having issues with the electronic ticket summary so I wanted to make sure that the ticket summary class was actually created properly and the method was working. The three test words you see means that it wrote three separate txt files.

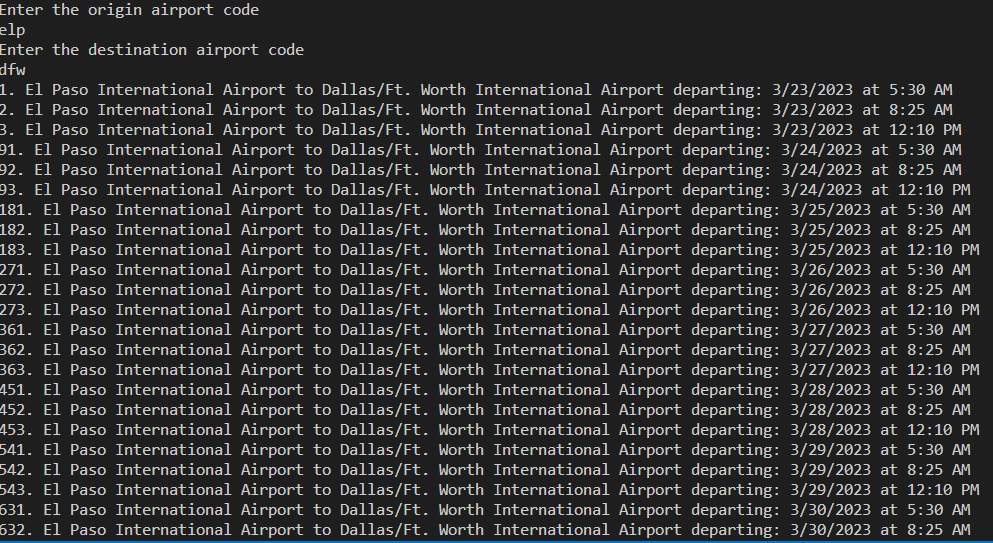
Text

Description automatically generated

For the screenshot below I wanted to see all the airports that were created because there were not that many airports to go through. From the hashmap that was printed I was able to see that all airports were created.



In the next screenshot I tested to see if the search functionality worked properly when customers searched for flights by airports. You can see that all possible flights that were going from El Paso to Dallas were printed and shown.



# **Code Review**

To review I first made sure that every main function and objective of this lab was programmed and worked properly. In this case it was the customers’ ability to purchase a ticket. I then checked that every object that was created was stored in the proper location and the correct data structure. Next I did black box testing to check for different ways that the user could break the program. This helps me implement the proper try-catch blocks so the program would not break. Throughout these tests I went through the code review checklist to make it easier to detect errors in the program or if anything could be improved upon. Another main objective was to make sure most functions were not hard coded, and the program was dynamic as possible. This is the case for storing all flight and customer objects as well the dynamic date and time changes when the departure date and time were created. When it comes to time complexity the main segment, I wanted to watch out for was the creation of the flight and customer objects. They both create and store the objects in O(n2) time. This is due to the nested for loop that was created that reads by column and rows.