Ashley Magallanes
March 31, 2023
CS 3331 – Advanced Object-Oriented Programming – Spring 2023
Professor Daniel Mejia
Programming Assignment #4

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

Programming Assignment number four is the fourth version of the *Miner Airlines* System that we have been developing since the beginning of this spring semester. In this new version updated functionalities are needed for both customers and employees. Customers are required to have a search functionality. This will allow them to search for flight information through ID or through airport codes. I also included the extra option of also searching flights by departure date. Then the customer can choose to make a purchase. The employee needs several new functionalities for this programming assignment. These include acquiring airport information, processing an automatic purchasing file and creating electronic ticket summaries for customers.

Before starting part B of the assignment, I looked back into my UML Diagrams to see how these new requirements could be incorporated through the program. Then I created the UML State Diagram for the process of purchasing flight tickets as an individual customer. Since this functionality was developed since past assignments, I worked backwards and used my code to guide the state diagram. Through this process I reviewed that the purchasing process I had developed was complete and had a consistent understandable flow. The diagram begins at the state “Pick Flight” which is the state where the customer choses the ID of the flight wanted to be purchased. The next state handles the amount of seat ticket wanted, then there is a composite state to showcase the prices of the seat options, which are Main Cabin, Business Class, and First Class. This is followed by the customer choosing the option wanted, checking the availability of these seats, computing the total, and depending on if the customer has enough money available then the transaction composite is passed before the state diagram finalizes. If the customer doesn’t have this requirement, then a state that prints a message letting the customer know the transaction can’t be made is printed before finalizing the process. I also modified the UML Class Diagram to include two new classes, one that creates AutoPurchase objects and another one that handles the purchasing of tickets. I decided to create this SellFlights class to make my code easier to read, separate these methods into a single space and just refactor the main class. The UML Use Case Diagram stayed almost the same since the process still only requires two actors the customer and the employee, and the use cases are also the ones as before. However, I did add the make electronic ticket summary use case for the employee as well as the make automatic purchases case.

The first step that I took when starting to implement the new required functionalities into the program system, was to add the “buttons” or like the new options needed for the employee and

customer. This includes an option for customers to search for flight, the option for employee to handle auto purchases by only selecting that option and allowing the electronic summary creation. Then I began to work with the customer option of searching for flights. I decided that it was a good idea to have two new buttons one that allows the customer to search flights by flight ID and another one that allows him to search by airport code and even also break it further into the destination date wanted if that's the customers case. If the customer picks the option to search by flight, they are asked what ID they want to search for, and the system looks for that key in the hash map of flight objects. If the flight ID is found, then the flight information is showcased in a neatly manner and then the system asks the customer if he wants to make the purchase, if he answers the option to do so then the normal transaction process is called. If the flight ID is not found then the system lets the customer know, and if the customer answers no to the purchasing question, then the system just goes back to the sign in menu. Now, if the customer choses the new functionality of searching by airport codes, it gets prompted for the origin airport code wanted, then the destination airport code wanted. Then that information is hold while the system asks the customer if he knows the departure date, if he does then it asks for it. This needs to be inputted exactly as it is in the file since it will look through all the departure dates that match the flight objects. Therefore, it is MM/DD/YYYY but only with the needed spaces, so march is not supposed to be expressed as 03 but as just 3. If the date is imputed correctly then only flights with the wanted origin code, destination code, and departure date are showcased. After this the customer is asked if he/she wants to make a purchase and if the yes option is pressed then the normal transaction process takes place. Just as the search by ID option if the customer doesn't want to purchase the ticket, then the system goes to the sign in system. If no flights with the wanted characteristics are found the system outputs "There are no flights with those specific properties." This is the main process for this new customer functionality. After completing it I moved into the employee.

The employee is now able to inquire airport functionalities. To be able to complete this functionality I had to add some attributes to the airport class. The airport now required an attribute that keeps track of the money that the airport has earned and after every transaction in which this airport takes place their airport fee is added to their earnings. I added this extra process into the transaction procedure to collect this amount needed. Then for the functionality I search through the hash map of flights and the first flight that has either the origin airport code or the destination airport code I hold the ID. This helps in the accessing of the airport wanted. Then I use the flights and the airport's getters to acquire the information wanted and showcase it to the employee. The information needed is the code, airport name, city, state, country, fees, lounge status, and the earnings. When the employee choses to open this function, they are prompted for the airport code and all the information is printed immediately. If the airport code inputted by the employee is not corresponding to anything, then the message "Invalid Airport Code!" is showcased. That is, it for how I implemented that functionality.

Then I started on the option for the employee to implement the Ticket Buyer Automation. To be able to process the AutoPurchasing files, I implemented the same procedure in the read class as I had for the other csv files I had to read for flights and people. Therefore, I also had to make objects of each line in the file, that is why I created a AutoPurchase class that has all attributes from the file, so when I make the auto purchase, I have stored all the information required in an object form. Then in the sellFlight class I decided to create another method in which the same purchasing process would take place but without the user interaction, therefore each object would be placed as a parameter for the method, and everything needed to make the transaction

would be there. This way I had two different methods one containing the user interaction and one that would go straight to the point, all computation of subtotals and totals stayed the same and the available money of the customer was first checked before the transaction, as well as making sure there were enough seats available. To be able to process both employees and customers in my program I had to create an exact copy of the auto purchasing method but using employees in the other one. This is necessary in my system because I have two hash maps storing either employees or customers. To process all the transactions for each loop was used to transverse the auto purchases linked hash map and make a call to the method handling the auto purchasing process. I used a linked hash map instead of a regular hash map to preserve the order of the file when placing the objects into the structure. In that method the first name and last name of the person are searched for in the customer list and if it is not found then the employee method is called, this happens because if the person is not a customer, then it is an employee and the other method specifically for employees must be called. This other method has the same functionality, but it uses the employee hash map as the customer list instead of the customer hash map. If the transaction does happen the method prints a ticket, and it also sets the ticket to the customer list, not only that but everything is set, the seats, the discounts, the revenues, the savings, and the airport money earned. If the transaction breaks requirements, then its just not made and the next object is passed through the method until the auto purchase hash map is null. Therefore, whenever the employee logs in and clicks the auto generate purchases option for the tickets are shown as the transactions are made immediately.

Then I continued to the electronic ticket summary which I implemented in the employee class since it makes sense for the employee to be able to access this function. The employee is asked for the ID of the person for which the summary would be generated, and right after that an if statement is used to check which method to call either the one that has the customer hashmap as a parameter or the one with the employee. For this I had to add the ID and the role attribute to the person class. If the person for which the ID, the employee wants to generate the ticket summary is a role of customer the customer method is called else the employee is called. In that method a file is used to write all the attributes and the customer or employee ticket hashmap is transverse to have the information of all their tickets. If the person has no tickets, then a message saying that appears. After all the tickets are transverse and the information is written in the file the file is closed and flushed. Therefore, the ticket summary is generated, and to have multiple files per person, I named the text file by their last name and first name so that the file doesn't get overwritten by other people's ticket summaries. These electronic ticket summaries include the confirmation number, the origin code, origin airport name, destination code, destination airport name, departure date, departure time, arrival date, arrival time, ticket type, ticket quantity, and the total cost for that ticket transaction.

2. What did I learn?

This programming assignment showed me the importance in not only having user interaction but also not having it. Being able to process the auto purchasing file and just with a click of a button being able to make all those transactions nonstop until up to 40k purchase requests were placed is just so satisfying and efficient. I also learned that although having some attributes in a class may not fully make sense, they are necessary. For example, all the previous programming assignments the people class only had the first name, last name, and date of birth, but for this one I had to add the ID and role of the person as well. In addition, I also have even more knowledge

on making and writing on files, which was something necessary for the electronic ticket summary procedure. I believe that next time I could combine both my customer and my employee hash map together into a type people one because it would save me having some of the functions duplicated. It seemed like a good implementation idea at the time but now it is just a little messy. Another way I could solve the auto purchasing process functionality is by breaking down the original function that makes ticket purchases into smaller bits and just call each function after the user interaction, then I could use those same functions for the auto purchasing process. I think this is a really good way of refactoring, but it can also mess up the process if done improperly, therefore I didn't want to take the risk and just use the computation process into a different method. This lab took me less to implement that the others however I did require to plan my process wisely before doing the auto purchasing, since in my opinion that requirement was the hardest or more complex one.

3. Solution Design

In this programming assignment I decided to go requirement by requirement when programming. Keeping the pace would help me not get overwhelmed with the whole thing all at once. My approach was to plan before coding as I mentioned earlier filling the UML diagrams again with the new material and attributes that would be needed to fulfill the new update functionality. In this programming assignment I added functionality to the customer and to the employee. Then to solve the problem was to reorganize my previous work and to make sure I knew where the new functions would be located. Then I began by working on the main class in which the log-in functionality takes place. I added all the options available for either customers or employees then used print statements as space holders for the new functions so that I could know what I needed to implement. I focused on updating the customer role functionality first since the modifications where minor to those in the employee role functionality. Then I moved on to the employee functionality. To allow the automatic purchases, I created the new class that would hold all the attributes from the csv file and therefore have all the information needed to complete or nor complete transactions. I used a linked hash maps to read the auto purchases file and be able to preserve the order of the transactions. A linked hash map will connect the entries and therefore preserve the order, which is something that a normal hashmap won't do. Then I used a for each loop to traverse this linked hash map and process transactions. I also used all the other data structures used on the previous programming assignments like array list, arrays, and normal attributes. Here are some assumptions that I took into consideration:

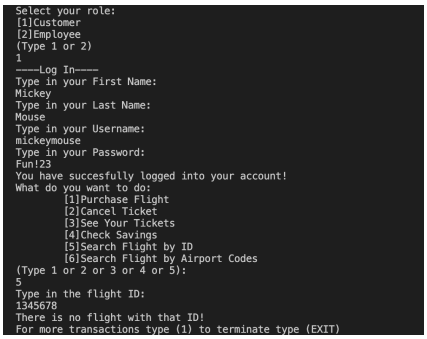
*Assumptions
-User will follow directions syntax wisely. Specially in spelling their name.
-Although I tried to handle most erroneous user input, I am assuming that the dates, times, and overall attributes will be inputted following the FlightSchedule.csv syntax if the manager menu is show cased
-I also focused on keeping the date and time as accurate as possible however the only assumption that I made regarding this is that the months have the number of days corresponding to this year 2023! And that the user should only change dates to existent ones in this year, for example there is no 2/29/23!!!!
-I am assuming that the customer will input dates as in the csv file when search for flights with them

-I am assuming that it is allowed for the csv file of auto generated purchases to be done in any order since a hash map is not ordered.
-I am assuming that even if all three Auto Purchasing files work keeping the 400k for the system is alright, therefore the file my program reads is the "AutoPurchaser400K.csv" but I tested that the other files worked too!!!

4. Testing

Throughout the programming process I tested that the instances of the objects were created and stored in the corresponding HashMap by using a for each loop to print the contents of them and check if it matched the csv files information. I would run the program continuously during the programming process to make sure updates worked properly or see if I had to fix something specific. I tested that my login process was working properly by trying different customers and employees with correct and erroneous data to catch the flaws. I then tested that the new functions for both employees and customers worked properly. The customer now has two more buttons one that lets search for the flight by ID and for this one I just inputted different IDs and made sure the corresponding information for it was given. For the other option of searching flights by airport codes I made sure that when inputting specific airport codes their information also corresponded, meaning that the flights showcased matched with the information the customer searched for. Then I tested that the employee option to acquire information about airports worked using the same logic as before and making sure that the information printed after the employee inputs the airport code wanted showcased the correct output information. The auto purchasing process was harder to test than all the other ones since there are up to forty cases and there is no "solution" wanted for the process. I just made sure that the tickets for customers where added, that money was subtracted from their accounts, that seats where adjusted and that when getting savings or airport information there was a significant amount of change. Using a liked hash map secures the requirement of having the transactions done in the order they were inserted, and since the file is read up to down then then elements are placed into the linked hash map in order. I also made sure that the ticket summary was done properly by logging in as a customer making some purchases and then login in as an employee and generating their electronic ticket summary and making sure those purchases are reflected. I did test my solution multiple times to make sure everything was running properly specially all the new functionalities. I used both white-box and black-box testing because I had access to all my code and knew what the program is supposed to do, but I don't know exactly the solutions on "money" based like the output for the schedule csv and the output for the customer csv.

5. Test results



Here I am testing the search by ID functionality for customers, and in this case an invalid ID was searched for, resulting in the message above.

Here I am testing the search by ID functionality for customers, and in this case ID 1 was searched for, then the customer gets asked if he/she wants to make purchase. Here the option chosen was no, so it returns to the login menu. If the option was yes, then the normal transaction process would take place.

```
For more transactions type (1) to terminate type (EXIT)
1
Select your role:
[1]Customer
[2]Employee
(Type 1 or 2)
1
----Log In----
Type in your First Name:
Mickey
Type in your Last Name:
Mouse
Type in your Username:
mickeymouse
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Purchase Flight
        [2]Cancel Ticket
        [3]See Your Tickets
        [4]Check Savings
        [5]Search Flight by ID
        [6]Search Flight by Airport Codes
(Type 1 or 2 or 3 or 4 or 5):
6
Type in the origin airport code:
ELP
Type in the destination airport code:
DFW
Do you know the departure date? [1]YES [2]NO
2
```

In this image the customer option to search flight by airport codes is pressed. The customer types in the origin code and then the destination code. Then gets asked if he knows the date for the program to also take that into consideration.

```
Origin Airport City: El Paso
Origin Airport State: Texas
Origin Airport Country: United States
Origin Airport Fee: 3.5
Origin Airport Lounge: false
Destination Airport City: Dallas
Destination Airport State: Texas
Destination Airport Country: United States
Destination Airport Fee: 3.0
Destination Airport Lounge: true

Flight ID: 1263
********** Flight Information **********
Flight Number: 2331263
Origin Airport: El Paso International Airport
Origin Code: ELP
Destination Airport: Dallas/Ft. Worth International Airport
Destination Code: DFW
Departure Date: 4/6/23
Departure Time: 12:10 PM
Arrival Date: 4/6/23
Arrival Time: 2:44 PM
Duration: 94
Distance: 551
Time Zone Difference: 1
Flight Type: Domestic
Surcharge: 0
Food Served: false
Route Cost: 11686
Miner Points: 1102
First Class Price: 1849
Business Class Price: 699
Main Cabin Price: 517
First Class Seats: 8
Business Class Seats: 60
Main Cabin Seats: 101
Total Seats: 169
Origin Airport City: El Paso
Origin Airport State: Texas
Origin Airport Country: United States
Origin Airport Fee: 3.5
Origin Airport Lounge: false
Destination Airport City: Dallas
Destination Airport State: Texas
Destination Airport Country: United States
Destination Airport Fee: 3.0
Destination Airport Lounge: true

Do you want to purchase a flight? [1]YES [2]NO
```

If the customer doesn't specify the date, then all the flights with the corresponding origin and destination codes get showcased. Here we can see that the origin code actually is ELP and the destination is DFW. This is the case for all of the flights printed.

```
        [4]Check Savings
        [5]Search Flight by ID
        [6]Search Flight by Airport Codes
(Type 1 or 2 or 3 or 4 or 5):
6
Type in the origin airport code:
DFW
Type in the destination airport code:
ELP
Do you know the departure date? [1]YES [2]NO
1
Type the departure date (MM/DD/YY):
3/23/23
Flight ID: 4
********** Flight Information **********
Flight Number: 2344
Origin Airport: Dallas/Ft. Worth International Airport
Origin Code: DFW
Destination Airport: El Paso International Airport
Destination Code: ELP
Departure Date: 3/23/23
Departure Time: 5:30 AM
Arrival Date: 3/23/23
Arrival Time: 6:04 AM
Duration: 94
Distance: 551
Time Zone Difference: -1
Flight Type: Domestic
Surcharge: 0
Food Served: false
Route Cost: 19082
Miner Points: 1102
First Class Price: 2189
Business Class Price: 1066
Main Cabin Price: 339
First Class Seats: 14
Business Class Seats: 41
Main Cabin Seats: 90
Total Seats: 145
Origin Airport City: Dallas
Origin Airport State: Texas
Origin Airport Country: United States
Origin Airport Fee: 3.0
Origin Airport Lounge: true
Destination Airport City: El Paso
Destination Airport State: Texas
Destination Airport Country: United States
```

In this image the date was specified as “3/23/23” which corresponds to the format in the csv file. The origin code and the destination code not only also match but the departure date is the corresponding one as well. All the flights with these attributes get showcased for the customer. Then just like the search by ID option the customer gets asked if he/she wants to make a purchase and if the option is yes then the normal purchasing process happens, else it returns to the log in menu.

```
For more transactions type (1) to terminate type (EXIT)
1
Select your role:
[1]Customer
[2]Employee
(Type 1 or 2)
2
----Log In----
Type in your First Name:
Ashley
Type in your Last Name:
Magallanes
Type in your Username:
ashleymagallanes
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Inquire Information
        [2]Cancel Flight
        [3]Purchase Flight
        [4]See Tickets
        [5]Cancel Personal Ticket
        [6]Inquire Airport Info
        [7]Automatic Purchasing
        [8]Customer Electronic Ticket Summary
(Type 1 or 2 or 3):
6
Which airport do you want to inquire information about? (Type the CODE of the airport)
ELP
Airport Code: ELP
Airport Name: El Paso International Airport
Airport City: El Paso
Airport State: Texas
Airport Country: United States
Airport Fee: 3.5
Airport Lounge Status: false
Money Earned: 3.5
For more transactions type (1) to terminate type (EXIT)
```

In this image I am testing the employee functionality of acquiring airport information. As it is shown by imputing the airport code ELP we can see that it has earned $3.5 due to mickey mouse making a purchase earlier.

```
What do you want to do:
        [1]Inquire Information
        [2]Cancel Flight
        [3]Purchase Flight
        [4]See Tickets
        [5]Cancel Personal Ticket
        [6]Inquire Airport Info
        [7]Automatic Purchasing
        [8]Customer Electronic Ticket Summary
(Type 1 or 2 or 3):
6
Which airport do you want to inquire information about? (Type the CODE of the airport)
DFWW
Invalid Airport Code!
For more transactions type (1) to terminate type (EXIT)
```

Here an invalid airport code was chosen so it handles that properly.

```
        [8]Customer Electronic Ticket Summary
(Type 1 or 2 or 3):
6
Which airport do you want to inquire information about? (Type the CODE of the airport)
DFW
Airport Code: DFW
Airport Name: Dallas/Ft. Worth International Airport
Airport City: Dallas
Airport State: Texas
Airport Country: United States
Airport Fee: 3.0
Airport Lounge Status: true
Money Earned: 0.0
For more transactions type (1) to terminate type (EXIT)
```

In this case the airport DFW has no earnings because no ticket has been bought for that airport yet. The purchase mickey did was for ELP and LAS.

```
For more transactions type (1) to terminate type (EXIT)
1
Select your role:
[1]Customer
[2]Employee
(Type 1 or 2)
2
----Log In----
Type in your First Name:
Ashley
Type in your Last Name:
Magallanes
Type in your Username:
ashleymagallanes
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Inquire Information
        [2]Cancel Flight
        [3]Purchase Flight
        [4]See Tickets
        [5]Cancel Personal Ticket
        [6]Inquire Airport Info
        [7]Automatic Purchasing
        [8]Customer Electronic Ticket Summary
(Type 1 or 2 or 3):
7
```

```
STATUS: PAID
QUANTITY: 1
TOTAL PRICE: $331.25
CONFIRMATION NUMBER: 8393
NUMBER OF SEATS: 1
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 2
TOTAL PRICE: $1982.98
CONFIRMATION NUMBER: 6394
NUMBER OF SEATS: 2
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 2
TOTAL PRICE: $581.68
CONFIRMATION NUMBER: 4411
NUMBER OF SEATS: 2
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 2
TOTAL PRICE: $4313.06
CONFIRMATION NUMBER: 1710
NUMBER OF SEATS: 2
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 1
TOTAL PRICE: $62.79
CONFIRMATION NUMBER: 7022
NUMBER OF SEATS: 1
/\/\/\/\/\/\/\/\/\/\/\/\
```

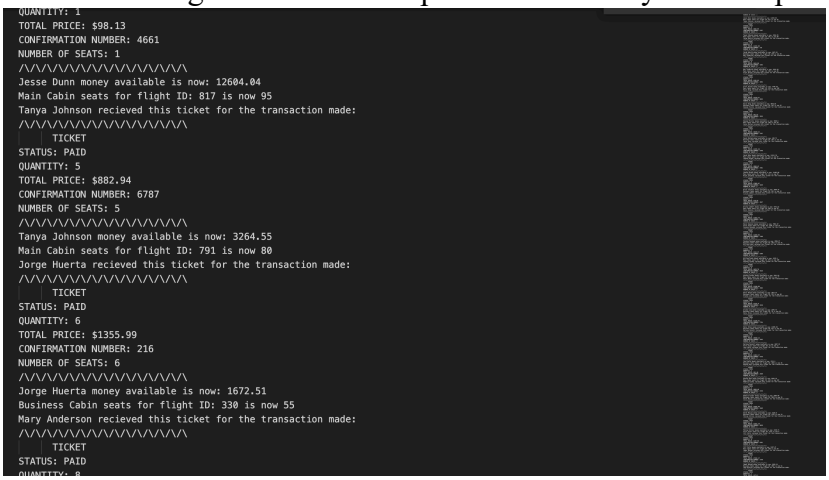
In these images I am demonstrating the automatic purchasing button number seven. All the transactions are made, of course only if they are appropriate regarding money and seats. The system takes less than a minute, but it does run continuously for a couple of seconds. As it shows I left the tickets being printed to visually see all those transactions. After all the transactions take place, the program returns to the main menu again. Therefore, it is only reeling on one button.

```
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Purchase Flight
        [2]Cancel Ticket
        [3]See Your Tickets
        [4]Check Savings
        [5]Search Flight by ID
        [6]Search Flight by Airport Codes
(Type 1 or 2 or 3 or 4 or 5):
3
------------------Tickets------------------
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 3
TOTAL PRICE: $2087.28
CONFIRMATION NUMBER: 5601
NUMBER OF SEATS: 3
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 2
TOTAL PRICE: $555.59
CONFIRMATION NUMBER: 4850
NUMBER OF SEATS: 2
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 4
TOTAL PRICE: $1694.82
CONFIRMATION NUMBER: 1894
NUMBER OF SEATS: 4
/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 3
TOTAL PRICE: $3598.99
CONFIRMATION NUMBER: 3992
NUMBER OF SEATS: 3
/\/\/\/\/\/\/\/\/\/\/\/\
```

When Mickey Mouse logs in to check his tickets there are now more since the auto purchases process handled the rest.

```
Log In
Type in your First Name:
Mickey
Type in your Last Name:
Mouse
Type in your Username:
mickeymouse
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Purchase Flight
        [2]Cancel Ticket
        [3]See Your Tickets
        [4]Check Savings
        [5]Search Flight by ID
        [6]Search Flight by Airport Codes
(Type 1 or 2 or 3 or 4 or 5):
4
Total savings: $557.4
For more transactions type (1) to terminate type (EXIT)
```

Then his savings also reflect the purchases done by the auto purchases procedure.



After the code termination the log ends up to be this really long text file since it showcases all of the transactions made.

updatedFlightSheet.csv

```
stance,Time Zone Difference,Arrival Time,Departing Date,Origin Airport State,Origin Airport Country,Origi
las/Ft. Worth International Airport,60,El Paso,2052,0,3,13,0,3459,3835
las/Ft. Worth International Airport,49,El Paso,1745,3,5,18,5235,3090,9504
las/Ft. Worth International Airport,39,El Paso,1856,1,3,7,1856,2568,3752
38,4,12,El Paso International Airport,41,Dallas,2189,2,0,5,4378,0,1695
29,5,8,El Paso International Airport,49,Dallas,2490,0,0,16,0,0,8544
138,6,10,El Paso International Airport,56,Dallas,2856,2,4,18,5712,2628,4842
00,7,9,Los Angeles International Airport,28,El Paso,1342,0,15,37,0,10335,14134
22,8,10,Los Angeles International Airport,37,El Paso,2005,0,9,12,0,8640,2748
108,9,7,Los Angeles International Airport,37,El Paso,2034,3,11,34,6102,7920,6868
08,10,9,El Paso International Airport,44,Los Angeles,1223,0,6,42,0,4956,6930
,120,11,9,El Paso International Airport,33,Los Angeles,2259,0,11,15,0,12056,4095
,160,12,13,El Paso International Airport,58,Los Angeles,3342,0,0,13,0,0,2860
,14,Chicago International Airport,49,El Paso,1810,0,13,32,0,9191,6848
4,10,Chicago International Airport,46,El Paso,2828,0,12,11,0,7332,2959
5,7,Chicago International Airport,40,El Paso,2303,4,9,19,9212,7947,8531
6,7,El Paso International Airport,37,Chicago,1405,1,15,23,1405,13590,12121
17,9,El Paso International Airport,45,Chicago,2606,0,15,5,0,10725,1625
18,8,El Paso International Airport,53,Chicago,1784,0,7,19,0,4564,10127
,13,Hartsfield-Jackson Atlanta International Airport,44,El Paso,1924,1,14,12,1924,9772,6096
,11,Hartsfield-Jackson Atlanta International Airport,49,El Paso,1724,2,9,26,3448,10206,9776
,9,Hartsfield-Jackson Atlanta International Airport,50,El Paso,1827,6,12,8,10962,10332,3264
,false,10990,110,22,11,El Paso International Airport,29,Atlanta,2396,4,23,28,9584,12765,11088
,false,10990,121,23,9,El Paso International Airport,41,Atlanta,2878,4,6,32,11512,5754,6208
.5,false,10990,142,24,7,El Paso International Airport,35,Atlanta,1699,4,16,0,6796,13040,0
25,11,John F. Kennedy International Airport,48,El Paso,3007,1,0,4,3007,0,1136
26,12,John F. Kennedy International Airport,27,El Paso,2168,1,25,15,2168,26575,4815
,27,8,John F. Kennedy International Airport,43,El Paso,2925,0,18,6,0,19530,2766
13,113,28,11,El Paso International Airport,31,New York,1254,0,27,14,0,22923,6272
2213,124,29,10,El Paso International Airport,23,New York,1961,0,34,2,0,37230,424
2213,122,30,12,El Paso International Airport,32,New York,1528,1,11,9,1528,11627,4689
39,31,14,Boston Logan International Airport,43,El Paso,2866,0,6,10,0,5484,4190
2,32,10,Boston Logan International Airport,24,El Paso,3015,3,19,1,9045,18088,191
159.33.8.Boston Logan International Airport.54.El Paso.2060.2.7.7.4120.8092.2940
```

updatedCustomerSheet.csv

```
Username,ID,Money Available,Role,Password,First Name,DOB,Flights Purchased,MinerAirline
mickeymouse,1,112.05,Customer,Fun!23,Mickey,2/29/36,3,true,Mouse,
donaldduck,2,35.85,Customer,Fun!23,Donald,3/21/17,6,true,Duck,
minniemouse,3,108.42,Customer,Fun!23,Minnie,11/23/42,5,false,Mouse,
goofydisney,4,164.04,Customer,Fun!23,Goofy,6/10/34,7,false,Disney,
plutodisney,5,87.64,Customer,Fun!23,Pluto,8/5/95,2,true,Disney,
chipdisney,6,53.56,Customer,Fun!23,Chip,11/28/41,2,false,Disney,
daledisney,7,98.63,Customer,Fun!23,Dale,10/8/50,6,false,Disney,
cinderelladisney,8,93.82,Customer,Fun!23,Cinderella,4/9/98,4,false,Disney,
arieldisney,9,76.43,Customer,Fun!23,Ariel,3/12/37,3,true,Disney,
jasminedisney,10,41.52,Customer,Fun!23,Jasmine,3/22/02,2,true,Disney,
aladdindisney,11,51.6,Customer,Fun!23,Aladdin,11/3/66,5,true,Disney,
simbadisney,12,26.04,Customer,Fun!23,Simba,3/23/29,7,true,Disney,
mufassadisney,13,44.04,Customer,Fun!23,Mufassa,1/24/00,3,true,Disney,
auroradisney,14,55.03,Customer,Fun!23,Aurora,7/18/62,9,true,Disney,
peterpan,15,80.05,Customer,Fun!23,Peter,12/26/30,7,true,Pan,
woodydisney,16,147.46,Customer,Fun!23,Woody,2/20/91,6,false,Disney,
buzzlightyear,17,194.93,Customer,Fun!23,Buzz,12/2/27,8,true,Lightyear,
jafaraladdin,18,12.66,Customer,Fun!23,Jafar,3/16/28,5,false,Aladdin,
urselamermaid,19,115.85,Customer,Fun!23,Ursela,1/11/94,4,true,Mermaid,
beastdisney,20,71.24,Customer,Fun!23,Beast,1/18/44,5,true,Disney,
```

The updated csv files generated after the program terminates also showcase all the transactions done by the auto generate purchases button employee functionality. The last three columns show the revenues and seats bought which is not zero for most. Then the second image shows the money available for the customers after the program terminated and it decreased significantly due to the transactions.

```
Select your role:
[1]Customer
[2]Employee
(Type 1 or 2)
2
----Log In----
Type in your First Name:
Ashley
Type in your Last Name:
Magallanes
Type in your Username:
ashleymagallanes
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Inquire Information
        [2]Cancel Flight
        [3]Purchase Flight
        [4]See Tickets
        [5]Cancel Personal Ticket
        [6]Inquire Airport Info
        [7]Automatic Purchasing
        [8]Customer Electronic Ticket Summary
(Type 1 or 2 or 3):
8
Customer ID:
60
For more transactions type (1) to terminate type (EXIT)
```

Now I am testing the last new employee functionality to generate the electronic ticket summary for the customer or any person. In this case I restarted the system so no transactions have been done and I generated the summary for myself (id 60).

electronicTicketSummaryMagallanesAshley.txt

```
No Tickets!
```

As shown the text summary is empty so a message saying no tickets is showed, since I haven't purchased any flights.

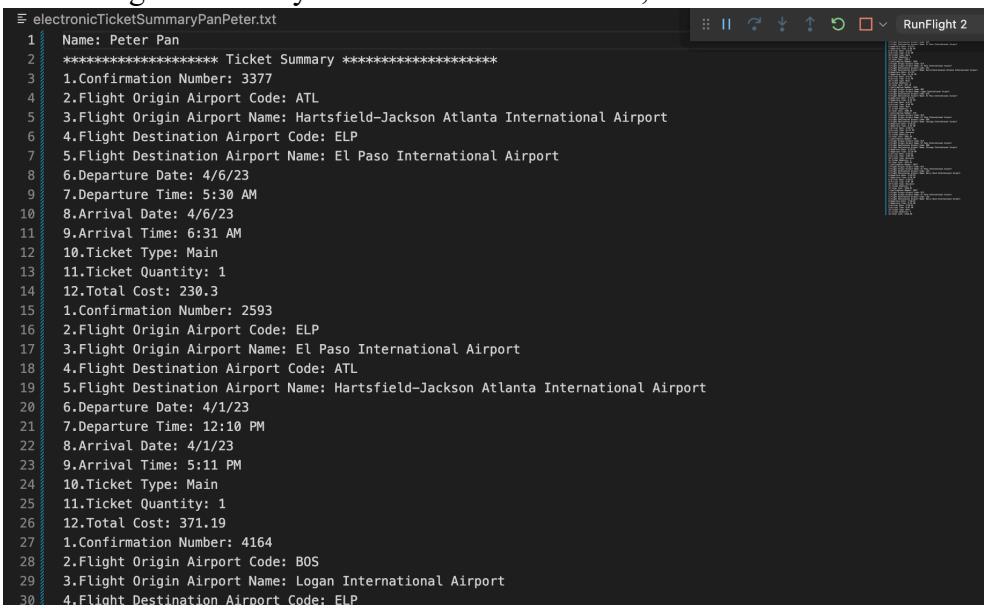
electronicTicketSummaryMagallanesAshley.txt

```
Name: Ashley Magallanes
********** Ticket Summary **********
1.Confirmation Number: 969
2.Flight Origin Airport Code: CDG
3.Flight Origin Airport Name: Paris Charles de Gaulle Airport
4.Flight Destination Airport Code: ELP
5.Flight Destination Airport Name: El Paso International Airport
6.Departure Date: 3/24/23
7.Departure Time: 12:10 PM
8.Arrival Date: 3/24/23
9.Arrival Time: 2:49 PM
10.Ticket Type: Business
11.Ticket Quantity: 5
12.Total Cost: 3661.45
1.Confirmation Number: 557
2.Flight Origin Airport Code: BOS
3.Flight Origin Airport Name: Logan International Airport
4.Flight Destination Airport Code: ELP
5.Flight Destination Airport Name: El Paso International Airport
6.Departure Date: 3/26/23
7.Departure Time: 5:30 AM
8.Arrival Date: 3/26/23
9.Arrival Time: 12:05 PM
10.Ticket Type: Main
11.Ticket Quantity: 2
12.Total Cost: 145.98
```

After generating the automatic purchases and creating the ticket summary for myself again we can see that it is now filled whit the transactions applied.

```
Type in your Password:
Fun!23
You have succesfully logged into your account!
What do you want to do:
        [1]Inquire Information
        [2]Cancel Flight
        [3]Purchase Flight
        [4]See Tickets
        [5]Cancel Personal Ticket
        [6]Inquire Airport Info
        [7]Automatic Purchasing
        [8]Customer Electronic Ticket Summary
(Type 1 or 2 or 3):
4
-----------------Tickets-----------------
/\/\/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 5
TOTAL PRICE: $3661.45
CONFIRMATION NUMBER: 969
NUMBER OF SEATS: 5
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
          TICKET
STATUS: PAID
QUANTITY: 2
TOTAL PRICE: $145.98
CONFIRMATION NUMBER: 557
NUMBER OF SEATS: 2
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
For more transactions type (1) to terminate type (EXIT)
```

When I log in to see my tickets these are reflected, which are the same one as the summary.



This is another electronic ticket summary for my favorite Disney character. Just to showcase that customers are also appropriately working.

*I also made sure that the transactions were done in order by running my system multiple times with the auto purchasing button only and checking that the number of tickets bought for specific people were always the same. I observed that they were, plus the linked hash map should take care of that functionality.

6. Code Review

My code does do what it is supposed to. It follows all the new instructions and functionalities. I do really think that it can be simple since I am using two hash maps instead of only one for the people but in my mind, it was a better way to already have the roles separated. I don't believe that my code is hardcoded since everything is based on user interaction and the file

readers handle the csv order. I made sure that the code behaved as intended throughout all the testing even when writing the upper portion of the report I found some missing data in the new csv file. I fixed everything that was not properly working. I think that my code is not that complex therefore it is easy to understands. Plus, I added multiple comments and annotations throughout the code. As mentioned earlier I could use one HashMap instead of multiple ones. I feel that my program is well structured and follows the class relationships. It does follow java language and java conventions. I did make my code well documented because I like to be able to look back and remember what every part does without tracing that much.

7. **Partner Demo – Christina Valtierra**

- What questions did you have about your partners functionality?
  My partners log just appends the information after every program compilation. Therefore, I was wondering why her file writer cannot be flushed in order to avoid this issue and restart the log file every time. Another question that I have about her transaction process is that her program takes away the transaction total from the customer before the customer accepts or confirms the transaction. Therefore, I was wondering if she sets the new available money before the confirmation statement.

- What concerns do you have about your partners functionality?
  I am concern that her customers get charged with the transactions even if they don't accept the confirmation of the purchase since her purchase process removes the money beforehand. I am also concern that with the new auto purchaser functionality her log file will be filled and since it doesn't flush then after multiple compilations it would be hard to track actions.

- How did you try to break it?
- I attempted to place invalid ID's and invalid username/password codes. I also tried signing in as an employee when the credentials were corresponding to a customer. Lastly, I tried to purchase ticket flights and then cancel them. In addition to this I made sure that the money was returned, and the seats were also returned to place.

- What test cases did you use? I tested Mickey Mouse as the customer and myself as the Employee.

*She noticed the concerns and mentioned to fix these before submission

*I really liked her menu and how it flows really nice with the user interaction!