Java based Flight Reservation System Project

Assignment 2

Regis University

MSSE670 – Java Software Development

Spring 2020 Week 2

03/22/2020

Brenda Palmer

Table of Contents

[Design Rationale 4](#_Toc35794553)

[Class Diagram 5](#_Toc35794554)

[Structure the System into Objects 6](#_Toc35794555)

[<<entity>> 6](#_Toc35794556)

[CustomerAccount: Retains customer information for the database 6](#_Toc35794557)

[Login: Authenticates customer to allow access to the system 6](#_Toc35794558)

[SearchFlightInformation: Search flight inventory 6](#_Toc35794559)

[ListAvailableItineraryOptions: List available itineraries with there options 6](#_Toc35794560)

[ReserveFlight: Reserves a flight from the list 6](#_Toc35794561)

[BookItinerary: Books a flight once it has been reserved 6](#_Toc35794562)

[Serializable Interface: Used to store and retrieve objects 6](#_Toc35794563)

[<<coordinator>> 6](#_Toc35794564)

[CustomerCoordinator: Receives customer requests and forwards them to the appropriate class 6](#_Toc35794565)

[<<service>> 6](#_Toc35794566)

[FlightReservationService: Processes reservation records when called ValidateCardService: Transmits info to credit card company database 6](#_Toc35794567)

[Sequence Diagram 7](#_Toc35794568)

[Captured Images from Eclipse 7](#_Toc35794569)

[BookItinerary class image 7](#_Toc35794570)

[CustomerAccount class image 8](#_Toc35794571)

[ListAvailableFlightOptions class image 8](#_Toc35794572)

[Login class image 9](#_Toc35794573)

[ReserveItinerary class image 10](#_Toc35794574)

[SearchFlightInformation class 10](#_Toc35794575)

[Captured JUint Tests images from Eclipse 11](#_Toc35794576)

[BookItineraryTest image 11](#_Toc35794577)

[CustomerAccountTest image 12](#_Toc35794578)

[ListAvailableFlightOptionsTest image 12](#_Toc35794579)

[LoginTest image 13](#_Toc35794580)

[ReserveItineraryTest image 13](#_Toc35794581)

[SearchFlightInformationTest image 14](#_Toc35794582)

# Design Rationale

Assignment 2 was to create java classes and Junit tests that correspond with the java classes to implement test driven development. The first task was to select a Use Case that we consider to be highest priority, then create a class diagram to know what the classes are along with their attributes and operations and then to create a sequence diagram to see the dynamic movement of the objects in the use case. Once this was accomplished, I created the java project to implement the domain layer which holds the business logic and rules. In the src folder for the domain layer, there are the following classes: CustomerAccount, Login, SearchFlightInformation, ListAvailableFlightOptions, ReserveItinerary and BookItinerary. Each class includes the required attributes and operations necessary for the Use Case “Book Itinerary” to be implemented. During the process it was necessary to create Junit tests to do test-driven development. Each class has its corresponding Junit test class.

After entering the instance variables of the class, it was discovered how easy it was to generate the constructor, setters and getters, hashcode and equals methods and the toString method. Once completed, it was made sure that the object was being printed to the screen by use of the main method. Then the Junit tests were created to test the validate and equals method for each class. Creating the validate method was pretty simple, it was creating a criterion what should be false and if it isn’t false then it must be valid. I referred to the validate and equals methods during my testing when anything passed or failed. When a test failed, I just looked at what logical I applied on what should be valid or not. If it was valid then I knew I had an issue with my code, if not I may have an issue with the validate method not being coded correctly.

During my Junit testing I did run into a logical error that Junit caught and then after some problem solving, I made the correction and modified the code. It dealt with the logic of the ListAvailableFlightOptions class field departureTimeOffsetOnce. This was finally corrected which later showed the Junit test as passing.

Once extra step I did was to add my code to GitHub, so I never lose my code and preparing the code to be turned in I don’t accidently lose anything during the process.

# Class Diagram

A screenshot of a cell phone

Description automatically generated

# Structure the System into Objects

## <<entity>>

### CustomerAccount: Retains customer information for the database

### Login: Authenticates customer to allow access to the system

### SearchFlightInformation: Search flight inventory

### ListAvailableItineraryOptions: List available itineraries with there options

### ReserveFlight: Reserves a flight from the list

### BookItinerary: Books a flight once it has been reserved

### Serializable Interface: Used to store and retrieve objects

## <<coordinator>>

### CustomerCoordinator: Receives customer requests and forwards them to the appropriate class

## <<service>>

### FlightReservationService: Processes reservation records when called ValidateCardService: Transmits info to credit card company database

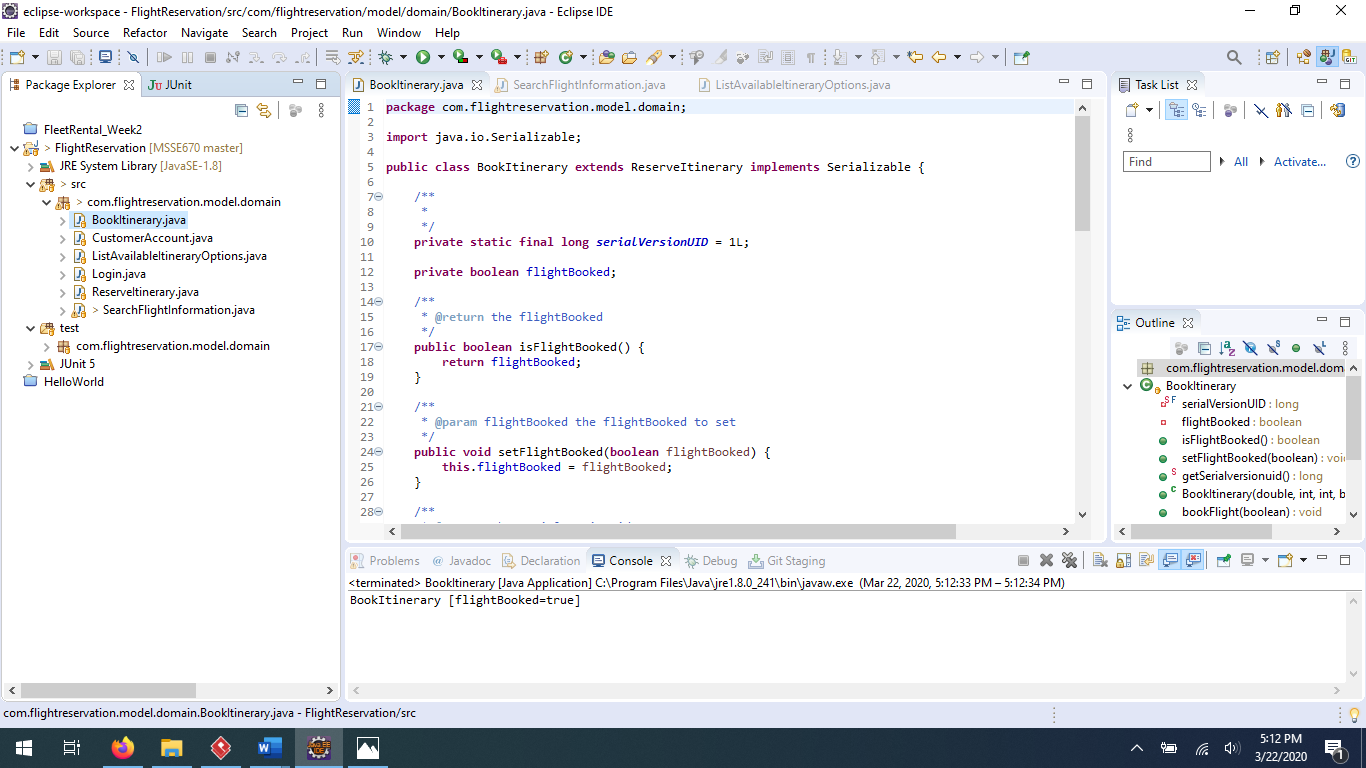
# Sequence Diagram

A screenshot of a social media post

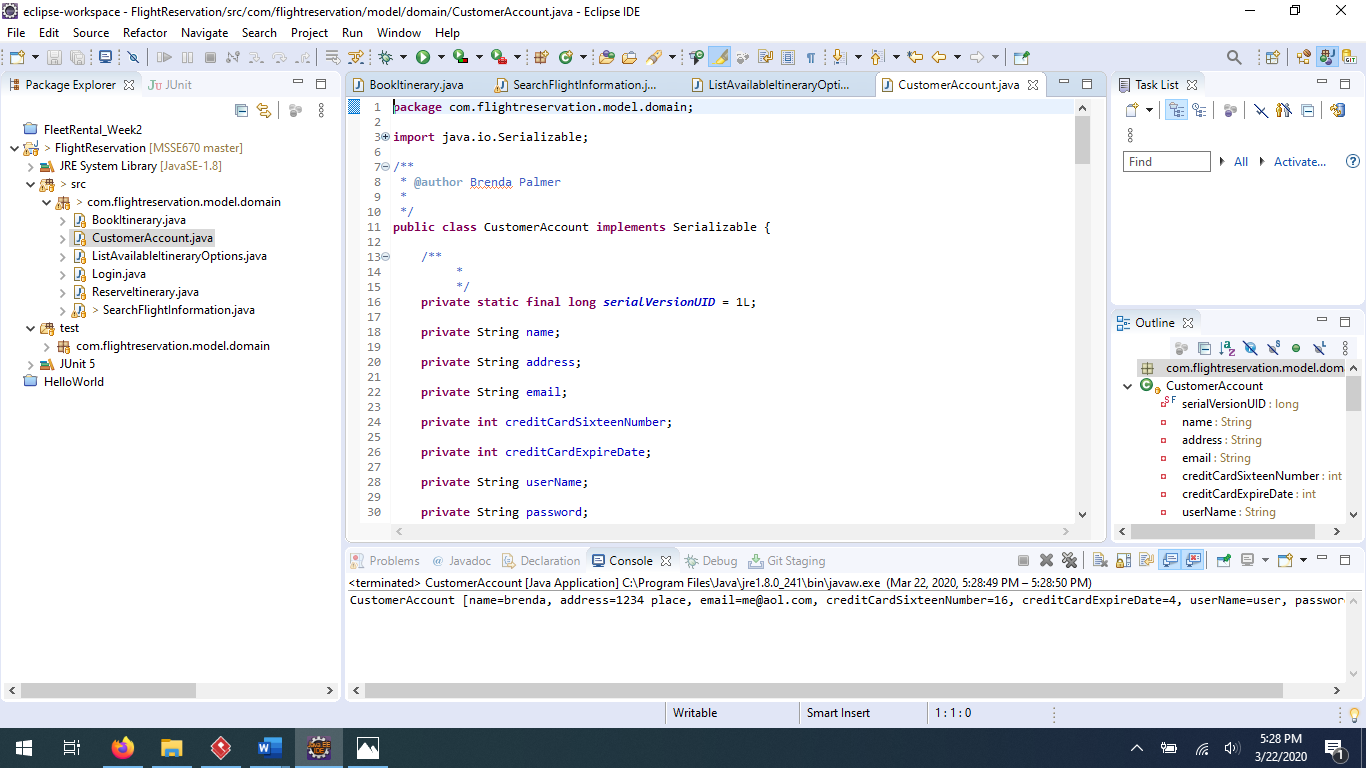
Description automatically generated

# Captured Images from Eclipse

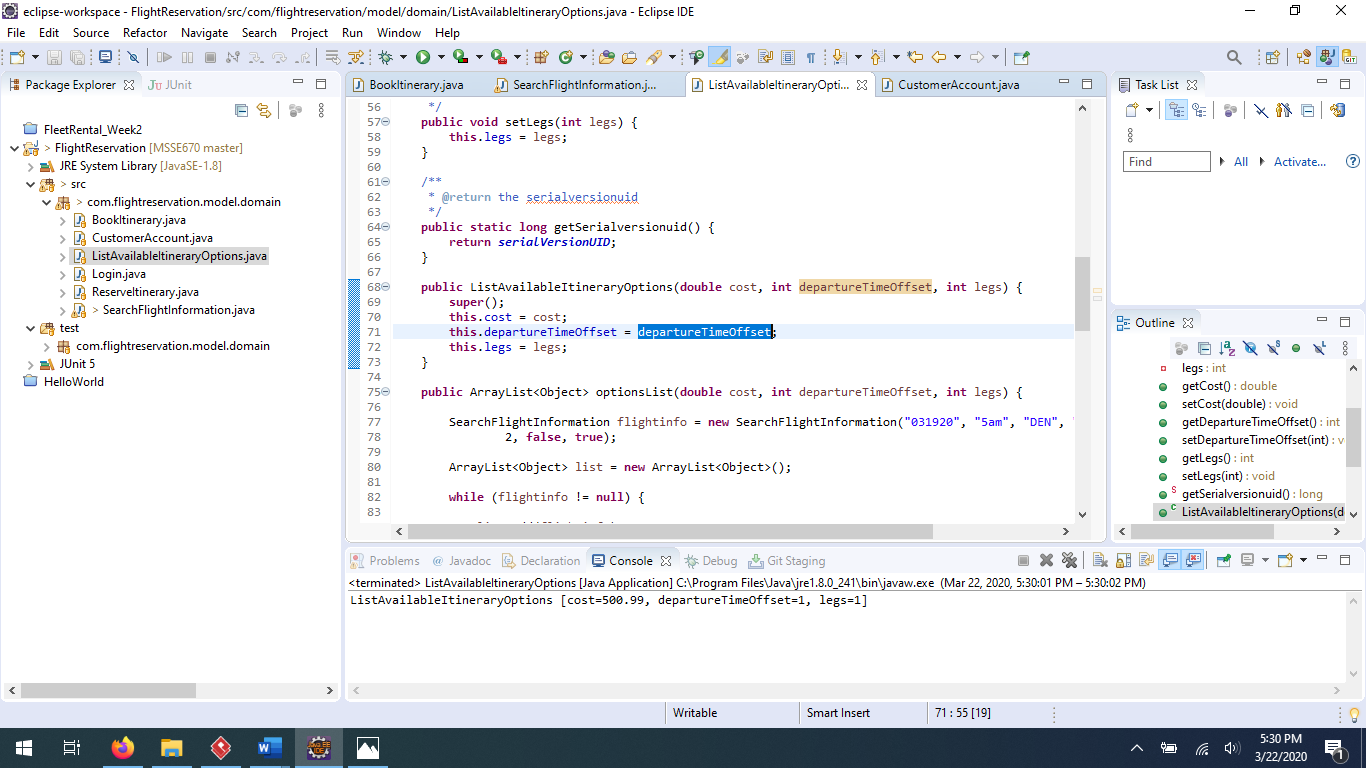
## BookItinerary class image



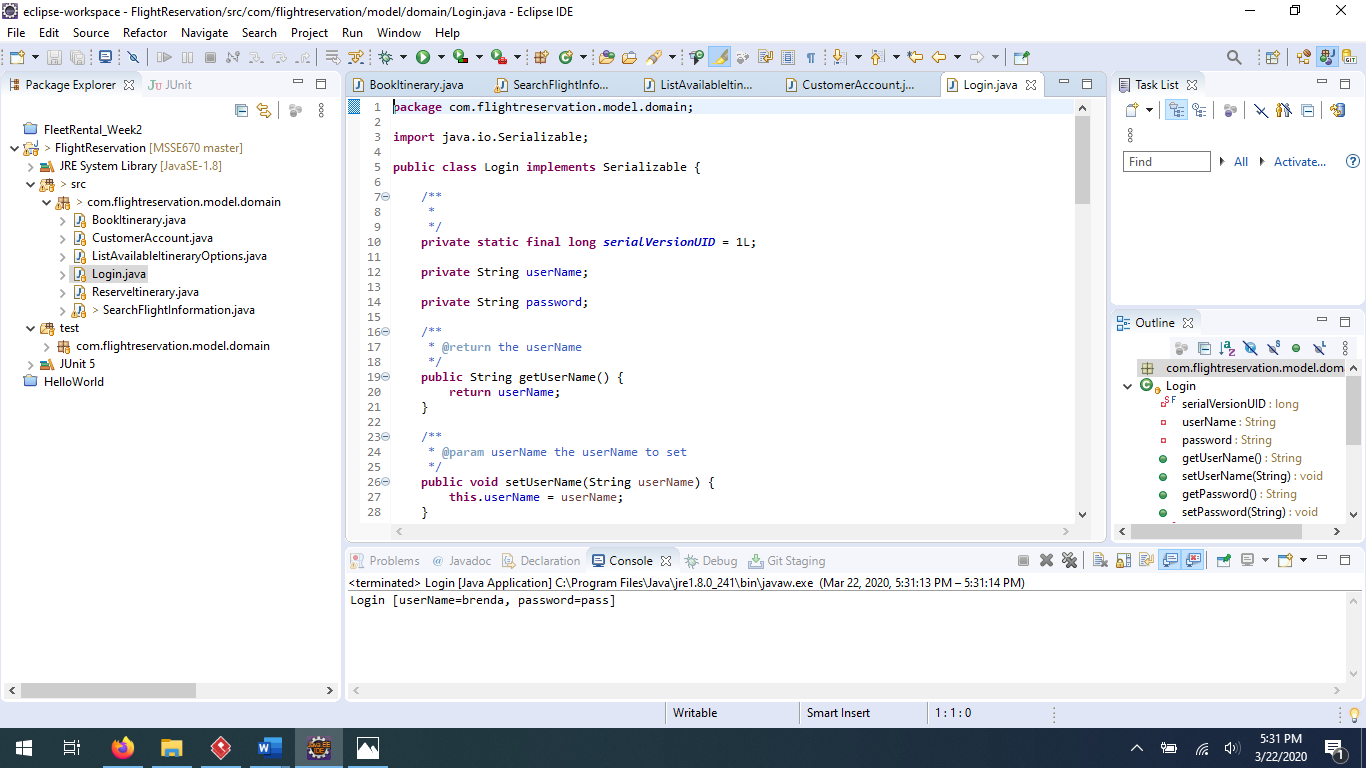
## CustomerAccount class image



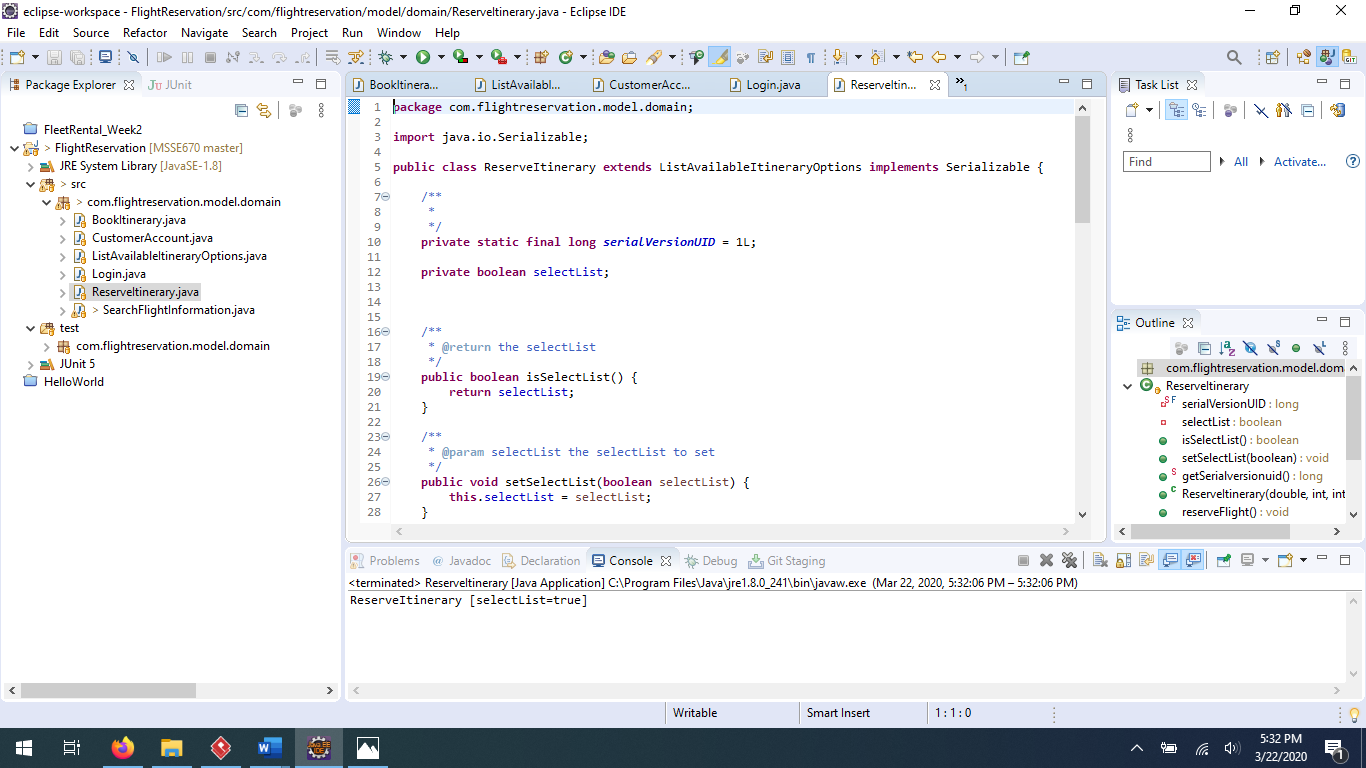
## ListAvailableFlightOptions class image



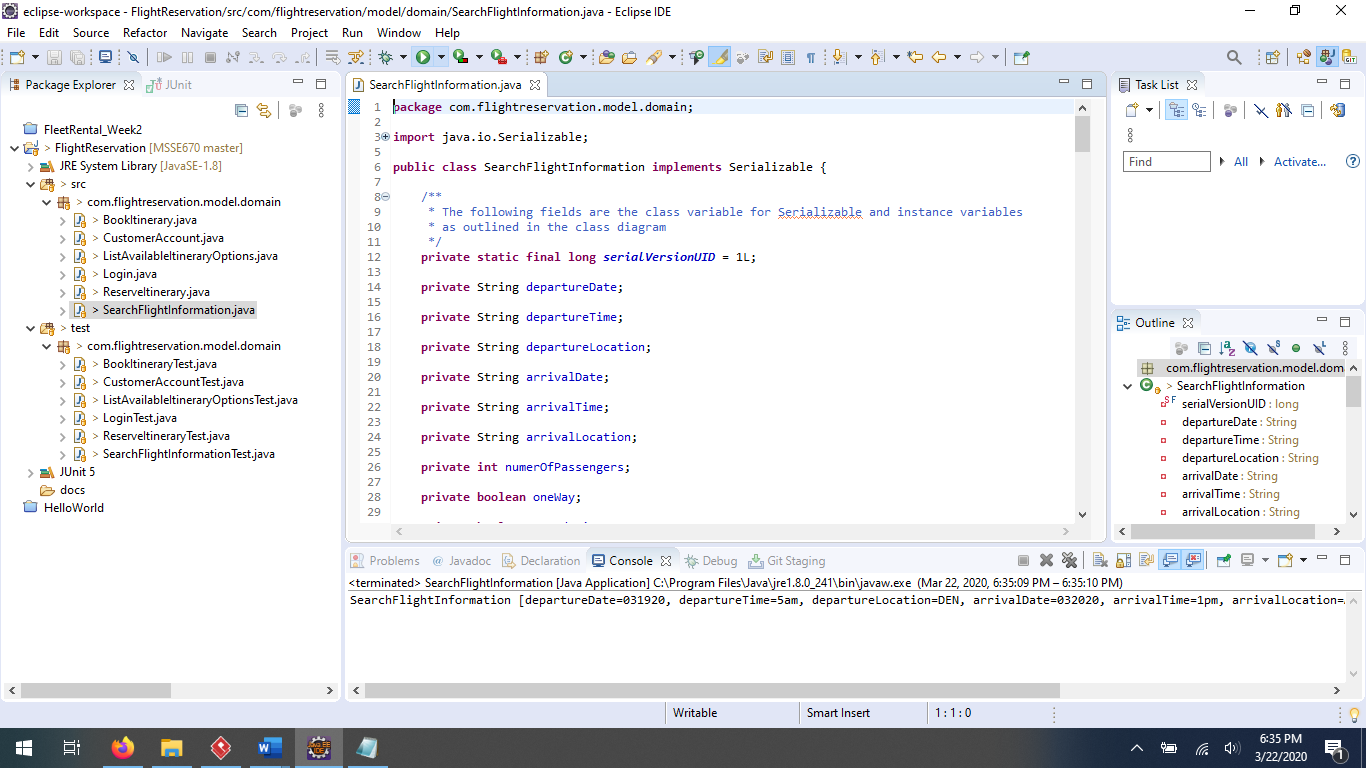
## Login class image



## ReserveItinerary class image

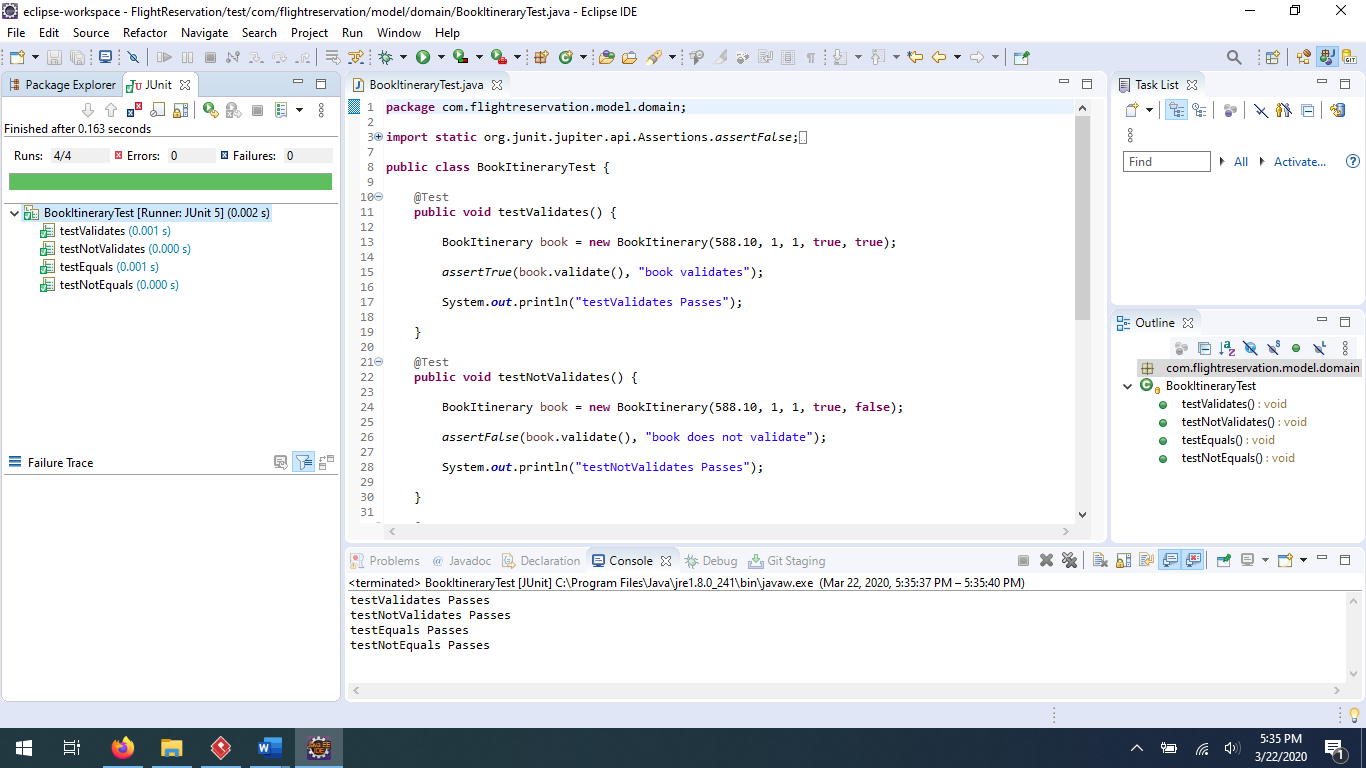


## SearchFlightInformation class

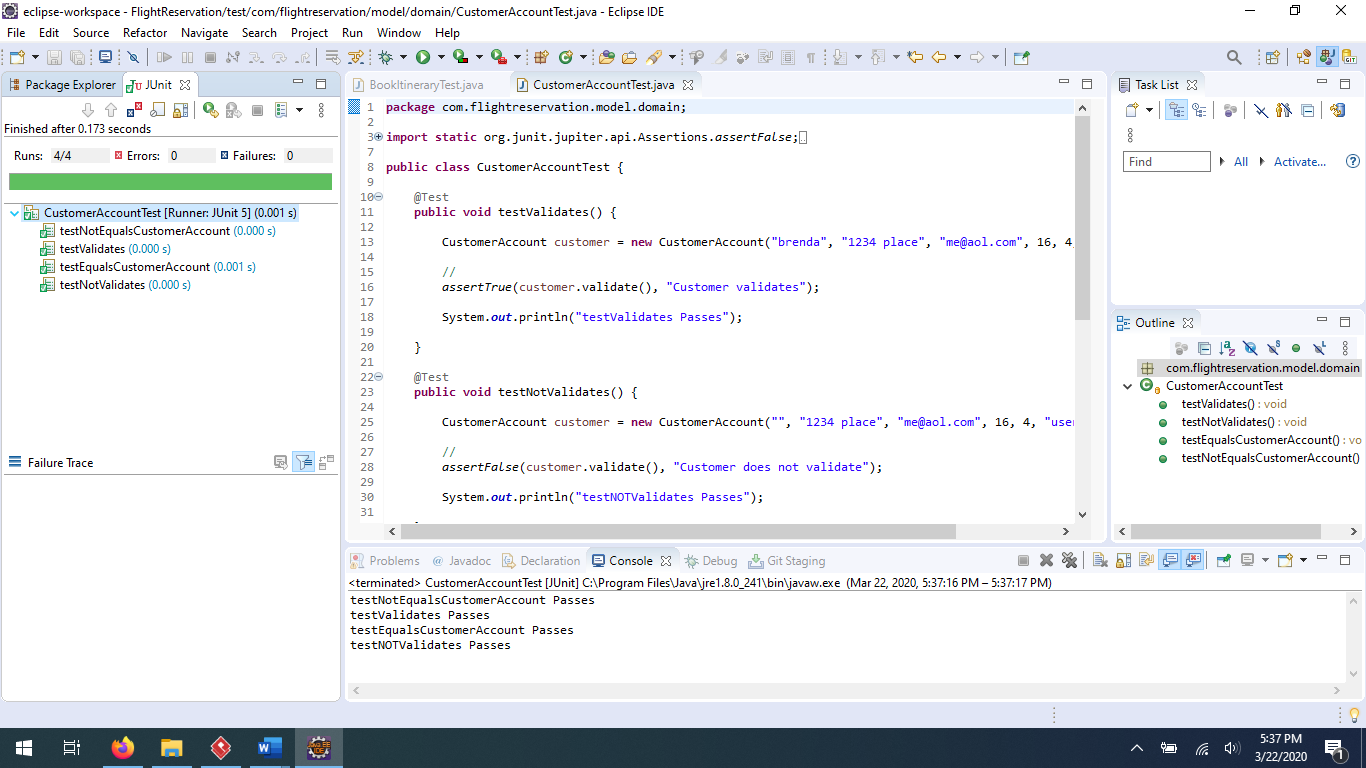


# Captured JUint Tests images from Eclipse

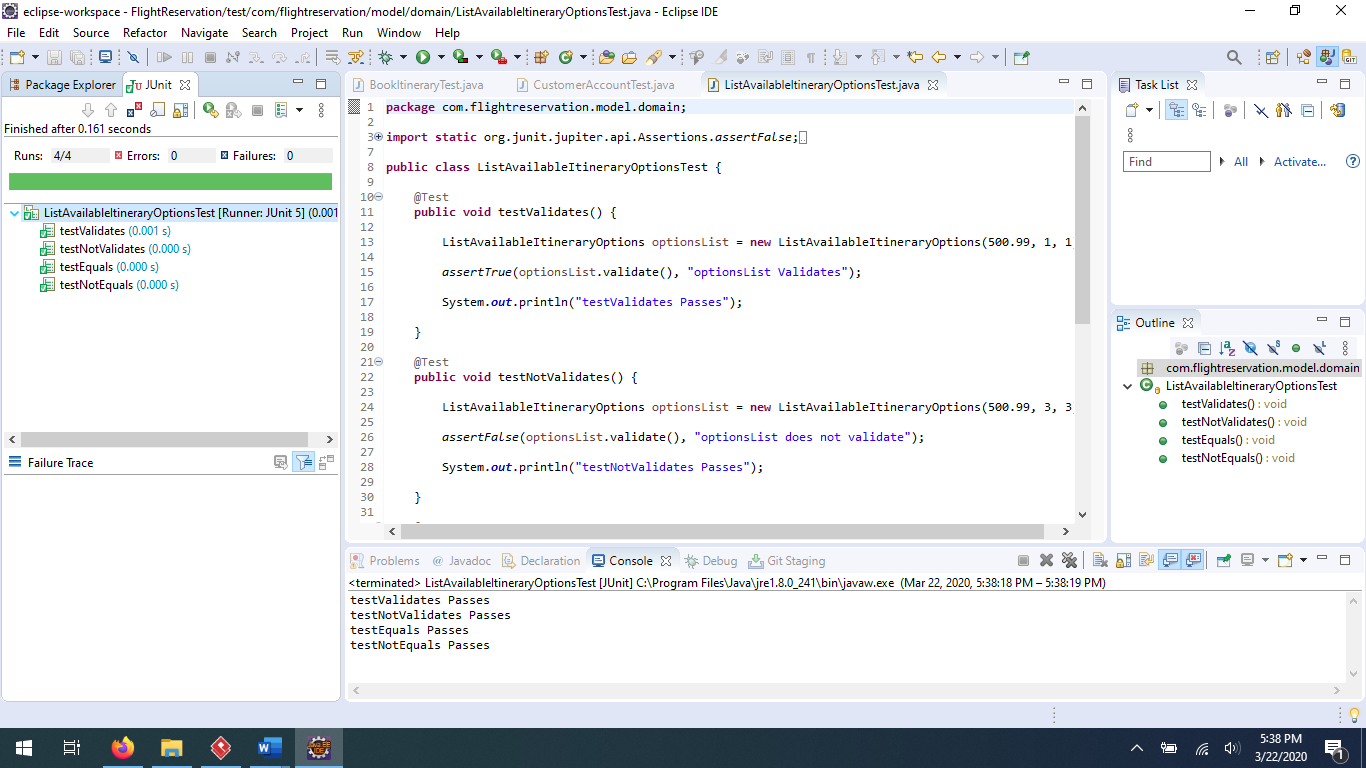
## BookItineraryTest image



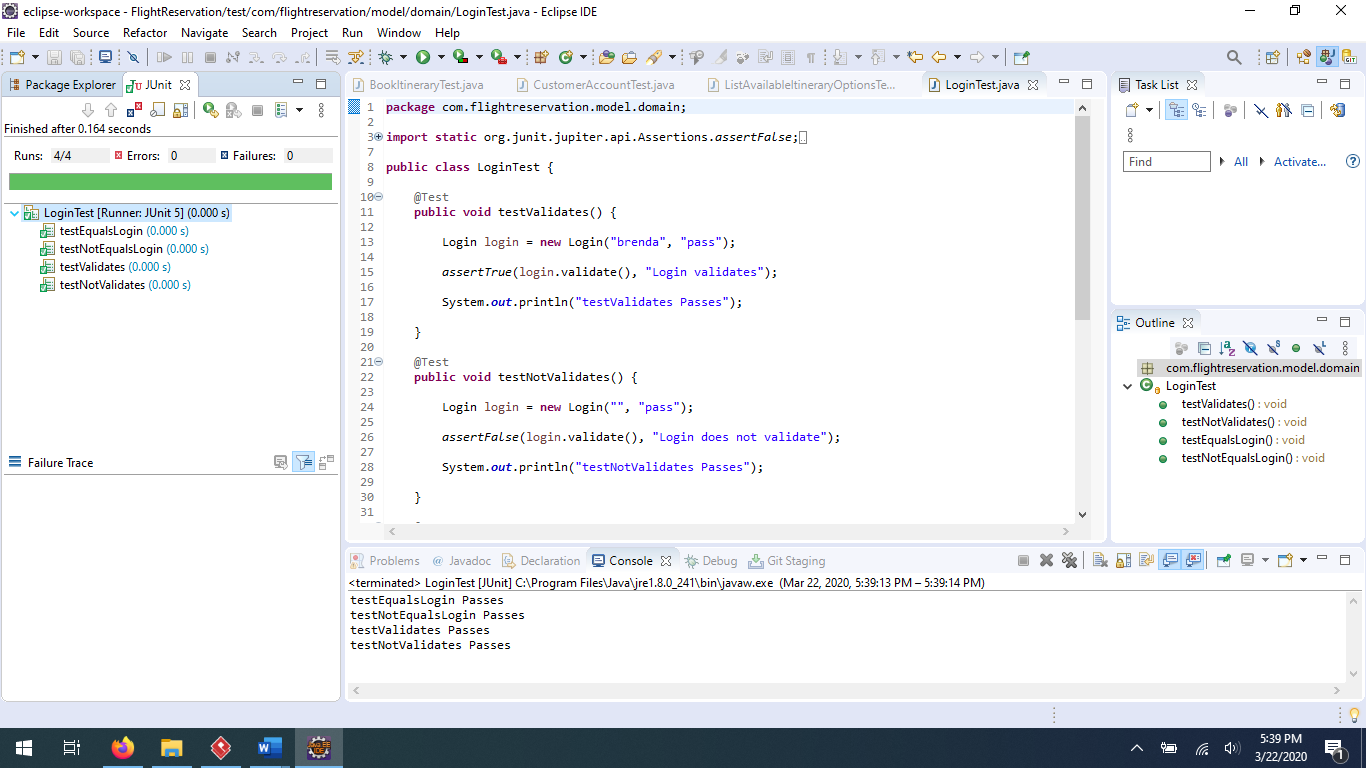
## CustomerAccountTest image



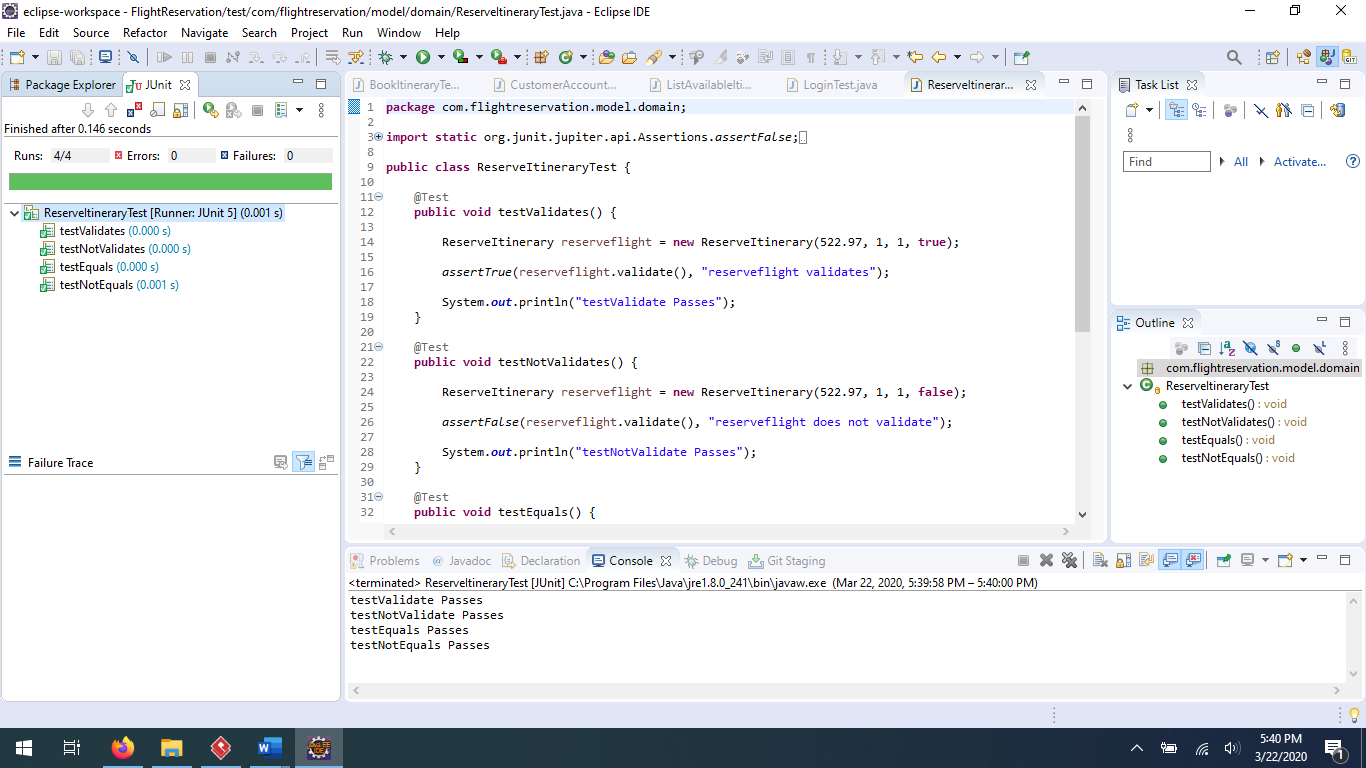
## ListAvailableFlightOptionsTest image



## LoginTest image



## ReserveItineraryTest image



## SearchFlightInformationTest image

