

Melissa Melaugh, Ciaran O’Boyle, Michelle Loughran

B00743475, B00793232, B00797038

Professional Software Development II

COM 809 – 84365

Rosaleen Hegarty

14 December 2020

COVER PAGE

Computing, Engineering and

the Built Environment

Ciaran O’Boyle, Michelle Loughran, Melissa Melaugh

COM 809 – 84365

Contents

[Abstract 4](#_Toc58717818)

[Introduction 5](#_Toc58717819)

[Development Methodology 6](#_Toc58717820)

[Analysis of requirements 8](#_Toc58717821)

[Design 9](#_Toc58717822)

[UML DESIGN 10](#_Toc58717823)

[Implementation 14](#_Toc58717824)

[TicketBookingSystem 14](#_Toc58717825)

[Method Summary 14](#_Toc58717826)

[Ticket Package for Passengers 15](#_Toc58717827)

[Method Summary Passenger 15](#_Toc58717828)

[Method Summary Adult Class - extends Passenger 15](#_Toc58717829)

[Method Summary Senior Class - extends Passenger 16](#_Toc58717830)

[Testing Strategy 17](#_Toc58717831)

[Appendices 18](#_Toc58717832)

[Appendix 1 – Burndown Chart 18](#_Toc58717833)

[Appendix 2 – Checklist 19](#_Toc58717834)

[Appendix 3 – Sprint Discussion 20](#_Toc58717835)

[Appendix 4 – Design Interface Ideas 21](#_Toc58717836)

[Appendix 5 – Java Docs 23](#_Toc58717837)

[User Guide 24](#_Toc58717838)

[Appendix 4 – Trello Boards 25](#_Toc58717839)

[Bibliography 27](#_Toc58717840)

[Figure 1 Team task development 5](#_Toc58717841)

[Figure 2 Overview of How SCRUM works 6](#_Toc58717842)

[Figure 3 – Comparison between the three frameworks 7](#_Toc58717843)

[Figure 4 – Overall Design of the Package 9](#_Toc58717844)

[Figure 5 : 10](#_Toc58717845)

[Figure 6 11](#_Toc58717846)

[Figure 7 A UML showing the Plane Package 12](#_Toc58717847)

[Figure 8 UML Overview 13](#_Toc58717848)

[Figure 9 The Projected Burn Down Chart 18](#_Toc58717849)

[Figure 10 21](#_Toc58717850)

[Figure 11 Initial design for the booking ticket system 22](#_Toc58717851)

[Figure 12 25](#_Toc58717852)

[Figure 13 25](#_Toc58717853)

[Table 1 - Methods Ticket Booking System 14](#_Toc58717854)

[Table 2 - Method Summary Passenger 15](#_Toc58717855)

[Table 3 - Method Summary Adult Class 15](#_Toc58717856)

[Table 4 - Method Summary Child Class 16](#_Toc58717857)

[Table 5- Method Summary Senior Class 16](#_Toc58717858)

# Abstract

The purpose of this project is to create a booking system for air travel. The system will check the flight status and assign seating tickets accordingly. Passengers will be sent the prices for flights dependant on age, Adult, Child, senior citizen after the passenger has entered their details, selected seats and entered the booking date. The Program will return to the passenger their booking information, assigned seating, the flight details, date of booking and complete costing.

# Introduction

The project aims to provide a booking system for a small airline start-up.

The members designing, implementing and testing the project are, Melissa Melaugh, Ciaran O Boyle, and Michelle Loughran. Each team member has a range of duties to ensure equality and effectiveness at each stage of the design, implementation and testing process.

The project's success will depend on teamwork, which will be achieved and regularly assessed through regular group meetings on Blackboard and Zoom. This ensures all members are on task, can discuss issues problems with the tasks assigned, and as a support network to ensure all members are being heard and can discuss concerns and grievances about their tasks. Discuss whether they need additional support from the other team members. The assignment of tasks using a SCRUM framework was agreed at the beginning of the project:

|  |  |  |
| --- | --- | --- |
| **Team Leader** | Melissa Melaugh | Create a Plane Class  Create Booking/  Scheduling Class  Create Plane Test  Create Schedule Test |
| **Team Member** | Ciaran O Boyle | Create Abstract Passenger class  Create Passenger Test  Create an Adult class  Create Adult Test |
| **Team Member** | Michelle Loughran | Create a Child Class  Create Child Test  Create Senior class  Create Senior Test  Create UML Documents  Create write-up |
| **Scrum Development**  **Team** | MM, COB, ML  Melissa Melaugh  Michelle Loughran  Ciaran O Boyle | Agree Project title and Outline  Test overall Project  Create Presentation  Agree write-up and role responsibility  Use Trello Boards  Use Github  Use Blackboard  Use Zoom  Use IntelliJ |

Figure Team task development

As the project's overall Outline was agreed, we proceeded to continue to work on the tasks assigned, ensuring that any queries could be followed up on Blackboard or Zoom and via a WhatsApp group and Trello boards. Melissa Melaugh was assigned Team Leader responsibilities and directed the overall direction of the Program. Melissa assigned time frames and classes after breaking the Program down into sections. Melissa undertook the Plane, Scheduling/Booking classes, whilst Ciaran undertook the Passenger and Adult classes, and Michelle undertook the Child and senior classes. As information was shared across each class, it was essential to work closely to ensure each class worked as part of the overall Program. Quite a few issues were writing, reviewing and implementing code to ensure the Program would run when assembled.

Creating such a program primarily allowed all team members to experience what it was like to work as a team. The chance to learn from each other and ensure there was peer learning throughout the group.

# Development Methodology

We decided to use an agile SCRUM management framework to complete and develop the Travel Booking Project using Trello. Scrum is one of the most popular agile methodologies used with software development teams and would enable the team to manage the project over a thirty-day sprint. This would empower the team to focus on a common goal of accumulating the project in full and on-time.

Diagram

Description automatically generated

Figure Overview of How SCRUM works

Scrum would help the team manage tasks, track issues, and time using sprints, review questions about inefficiencies, prioritise and rank the tasks, issues raised, and time as required. It means that as a team, we would work together and commit to the project. Using a Scrum framework would support the structure of roles, meetings, rules and tools. As a team, we should create and adapt the processes involved using the Scrum framework.

Scrum would help the team conquer the complications caused by using a waterfall development process and enable the team to identify conflicts early and quickly to make suitable modifications.

Graphical user interface

Description automatically generated

Figure – Comparison between the three frameworks

We decided to use Trello boards to enable the process across the team. The team moved to a spreadsheet to track progress in the program. It was also used to illustrate a burndown chart (Appendix, Burndown ChartsAppendix 2 – Checklist), main development points (Appendix, Checklist, Figure 3), meeting notes (Appendix, Sprint Discussion, Figure 4) and future directions of the program (Future Directions).

At each meeting, we would review the tasks assigned, completed to date and the backlog. This supported effective task completion, sharing responsibility, open discussion and peer learning to ensure the Program would be completed on time. Problems were shared and resolved on WhatsApp, which was especially useful during the program's implementation and testing stage.

# Analysis of requirements

Recreating a booking system was because many businesses, from entertainment to finance or education, use a booking system. It could be for booking a cinema ticket, a hotel, a webinar, a flight, and a conference room amongst many other things. It would give us the chance to replicate a program that required such a booking system, and we agreed the most common booking system used was probably a flight booking system, and by implementing a program for a small and upcoming flight company we could recreate a program that would cover many of the demands required.

The Program would enable the flight company to manage the planes within the company, manage the time schedules, enable passengers to select flight destinations, dates, book seats on the flight, and if the passenger was a child or senior citizen avail of discounts may apply. The Program would consider overall flight prices throughout the year, taking account of peak and quieter times where passengers could benefit from lower purchase prices for flights and seat allocation. Passengers will be able to purchase and print their tickets on completion.

Additional requirements were to show our programming skills; therefore we created a checklist of key points we wished to show off. This checklist is in the Appendix, Appendix 2 – Checklist under the Code heading.

# Design

Design of the Program began with a discussion as to what exactly was required. Who would be using the Program aside from passengers, how it was going to be linked? UML class diagrams were drawn up initially deciding what the creation of each class required. Aside from passengers booking the ticket, we had to think how this would link to the airport and what had to be deduced from a management perspective in terms of scheduling, organising flights and the booking of seats on a plane.

Figure – Overall Design of the Package

The Ticket Booking System was the main class that was to be used throughout the Program. It will be responsible for reading in passenger information, booking them onto a flight, and printing out the passenger's ticket. It will also let you know how much money you owe for your plane ticket or tickets (if you book for your whole family).

For airport employees, additional options are to check the information about a flight if you give the day and month. It will also be responsible for calling the "update schedule" method to ensure that there are planes for both the current month and the following month ready to be booked.

The Plane class would hold all the information about a particular flight. It includes a price breakdown for a given week in any month adjusted for the 'mark-up' depending on the day.

This Schedule class is to help take care of the scheduling of planes. It was originally part of the Plane class; however, it was split up to make it easier to understand.

The Passenger class is an abstract class holding information that all passengers need. This is primarily to show that we can use abstract classes but allow us to make changes depending on if someone is an adult, Child, or senior.

The adult class is an extension of the Passenger class, and it includes a count of all the adults that have been through the airport as well as an array list of Children that the Adult is responsible for (in the event that they are not flying with children this will be an ArrayList of size zero). This class is to hold all information about an adult passenger.

The child class is to hold information about a child that is flying. It also puts on their ticket if they are flying alone or if they have a responsible adult. If there is a responsible adult, this responsible Adult will be linked to them in case of a 'missing child' emergency.

The senior class is very similar to the adult class; however, the senior gets a lower price. They still have the option of bringing children along on their flight.

# UML DESIGN

UMLs were created as part of the design process with the help of a tutorial (Medium, n.d.)

Diagram

Description automatically generated

Figure :

The Top of the UML showing the Application package and the connections to the Passenger Package and Plane Package

# Diagram Description automatically generated

Figure

A UML showing the Passenger Package.

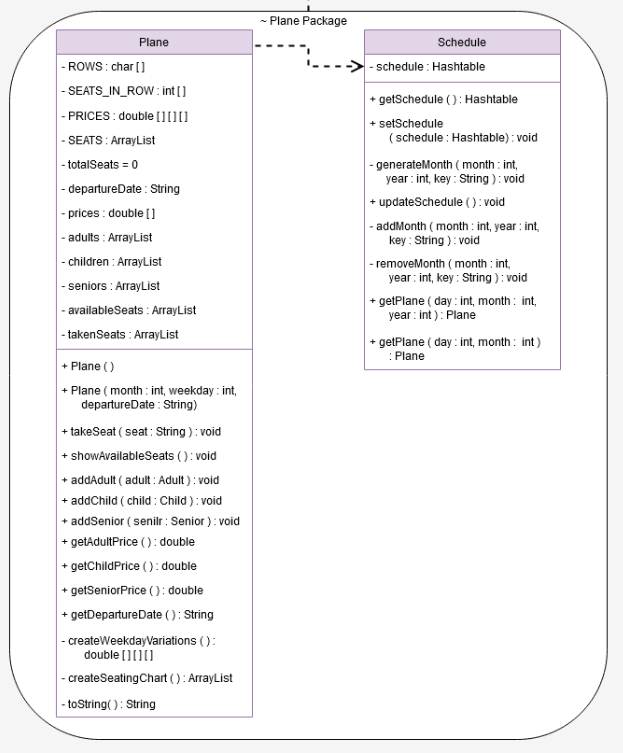


Figure A UML showing the Plane Package

Diagram

Description automatically generated

Figure UML Overview

# Implementation

The implementation of the program used both predefined methods and user defined methods additional information can be found Appendix 5 – Java Docs..

## TicketBookingSystem

The public class TicketBookingSystem extends the java.lang.Object. This is the main class which runs the ticket booking system.

The below table summarises some of the methods used in the TicketBooking system and can been explored in more detail in the attached appendix Appendix 5 – Java Docs.

## Method Summary

|  |  |  |
| --- | --- | --- |
| Modifier and Type | Method | Description |
| private static void | [**buyTickets**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#buyTickets())**()** | This method is responsible for letting the user buy tickets. |
| private static int | [**getNextInt**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#getNextInt(java.lang.String))**​(java.lang.String request)** | This method is responsible for getting the next integer from the user |
| private static java.lang.String | [**getNextString**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#getNextString(java.lang.String))**​(java.lang.String request)** | This method is responsible for getting the next string from the user |
| private static java.lang.String[] | [**getPassengerInfo**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#getPassengerInfo(TicketPackage.PlanePackage.Plane))**​(**[**Plane**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PlanePackage/Plane.html)**plane)** | This method is responsible for getting the basic passenger information and returning it |
| private static [Plane](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PlanePackage/Plane.html) | [**getTravelDate**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#getTravelDate())**()** | This method is responsible for getting the month and day from the user and return the plane for that day/month |
| static void | [**main**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#main(java.lang.String%5B%5D))**​(java.lang.String[] args)** | This method runs the entire Airport Check In Program. |
| private static void | [**shutDown**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#shutDown())**()** | This method is responsible for saving the state of the program when it shuts down |
| private static void | [**startUp**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#startUp())**()** | This method is responsible for loading in any objects that were saved. |
| private static void | [**viewPlaneInformation**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/ApplicationPackage/TicketBookingSystem.html#viewPlaneInformation())**()** | This method asks the user for a month and day, and displays information about the plane requested |

Table - Methods Ticket Booking System

## Ticket Package for Passengers

This package includes the Passenger class from which the ticket type for Adult, Child, and Senior is derived. Adult, child and senior classes all extend from the passenger class.

## Method Summary Passenger

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| protected static void | [**addPassenger**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#addPassenger())() |  |
| java.lang.String | [**getDepartureDate**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#getDepartureDate())() |  |
| java.lang.String | [**getFirstName**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#getFirstName())() |  |
| java.lang.String | [**getLastname**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#getLastname())() |  |
| static int | [**getNumberOfPassengers**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#getNumberOfPassengers())() |  |
| java.lang.String | [**getSeat**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#getSeat())() | This method gets the passengers seat number |
| int | [**getTicketNumber**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#getTicketNumber())() |  |
| void | [**setDepartureDate**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#setDepartureDate(java.lang.String))​(java.lang.String passengerDepartureDate) |  |
| void | [**setFirstname**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#setFirstname(java.lang.String))​(java.lang.String passengerFirstName) | This method sets the Passenger's name to a new value |
| void | [**setLastname**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#setLastname(java.lang.String))​(java.lang.String passengerLastname) |  |
| static void | [**setNumberOfPassengers**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#setNumberOfPassengers(int))​(int numberOfPassengers) |  |
| void | [**setSeat**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#setSeat(java.lang.String))​(java.lang.String passengerSeat) | This method sets the passenger's seat |
| void | [**setTicketNumber**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#setTicketNumber(int))​(int passengerTicketNumber) |  |
| void | [**showAllDetails**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html#showAllDetails())() |  |

Table - Method Summary Passenger

Appendix 5 – Java Docs show the full details methods constructors and fields.

## Method Summary Adult Class - extends [Passenger](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html)

|  |  |
| --- | --- |
| void | [**addChild**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#addChild(TicketPackage.PassengerPackage.Child))**​(**[**Child**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html)**acChild)** |
| int | [**getNumberOfAccompanyingChildren**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#getNumberOfAccompanyingChildren())**()** |
| static int | [**getNumberOfAdults**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#getNumberOfAdults())**()** |
| void | [**printAccompanyingChildren**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#printAccompanyingChildren())**()** |
| void | [**removeChild**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#removeChild(TicketPackage.PassengerPackage.Child))**​(**[**Child**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html)**abandonedKid)** |
| static void | [**setNumberOfAdults**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#setNumberOfAdults(int))**​(int numberOfAdults)** |
| java.lang.String | [**toString**](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Adult.html#toString())**()** |

Table - Method Summary Adult Class

Method Summary Child Class - extends [Passenger](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html)

|  |  |
| --- | --- |
| static int | [getNumberOfChildren](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html#getNumberOfChildren())() |
| java.lang.String | [getResponsibleAdult](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html#getResponsibleAdult())() |
| static void | [setNumberOfChildren](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html#setNumberOfChildren(int))​(int numberOfChildren) |
| void | [setResponsibleAdult](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html#setResponsibleAdult(java.lang.String))​(java.lang.String responsibleAdult) |
| java.lang.String | [toString](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html#toString())() |
|  |  |

Table - Method Summary Child Class

# 

## Method Summary Senior Class - extends [Passenger](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Passenger.html)

|  |  |
| --- | --- |
| void | [addChild](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#addChild(TicketPackage.PassengerPackage.Child))​([Child](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html) acChild) |
| int | [getNumberOfAccompanyingChildren](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#getNumberOfAccompanyingChildren())() |
| static int | [getNumberOfSeniors](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#getNumberOfSeniors())() |
| void | [printAccompanyingChildren](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#printAccompanyingChildren())() |
| void | [removeChild](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#removeChild(TicketPackage.PassengerPackage.Child))​([Child](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Child.html) abandonedKid) |
| static void | [setNumberOfSeniors](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#setNumberOfSeniors(int))​(int numberOfSeniors) |
| java.lang.String | [toString](file:///Users/michelleloughran/Desktop/Group%20Project/JavaDocs/TicketPackage/PassengerPackage/Senior.html#toString())() |

Table - Method Summary Senior Class

Appendix 5 – Java Docs show the full details methods constructors and fields.

# Testing Strategy

Throughout the design of the Program, testing was essential to ensure each part of the Program worked. The best way to do this was by testing each section as it was developed. Therefore, after discussion, the team agreed we would each design a class, and as the superclasses were completed, they would then be forwarded as soon as possible to the other team member to ensure that all the classes were working and that they could be tested.

When we encountered difficulties as a team, we agreed to meet online or via WhatsApp to try and resolve problems and come up with solutions using the SCRUM framework.

This agile approach meant that problems could be dealt with swiftly, problems could be resolved, and we could progress due to feedback from the team leader and adapt to changes as they were required.

# Appendices

# Appendix 1 – Burndown Chart



Figure  The Projected Burn Down Chart

(left/top) along side the actual Burn Down Chart (right/bottom). The black line between the 20th and 23rd of November indicates where Melissa joined the team a week late. Between the Projected Burndown and the final creation of the Actual Burndown chart, a new file has been added to take care of the plane scheduling.

# Appendix 2 – Checklist



# Appendix 3 – Sprint Discussion



# Appendix 4 – Design Interface Ideas

From:

To:

Departure Date:

Return Date:

Passengers:

Search Flights:

MCM Flights:

One Way

Return

Figure

From: Dublin

To: Aberdeen

16 DEC 2020

19 DEC 2020

Search Flights:

MCM Flights:

One Way

Return

3 Passengers:

Figure Initial design for the booking ticket system

# Appendix 5 – Java Docs

JavaDocs are available under the JavaDocs folder in the zipped upload.

# User Guide

Start up the program. It will tell you if you if it’s started fresh or if it’s been able to load a previous save. The program will have a schedule made for the current month and the next month in order to keep at least 28 days of planes.

You will then be presented with options. The first option is to buy a plane ticket or plane tickets, and the program will prompt you for details, and print out your tickets and the price that you owe. The second option is to view flight information, giving a detailed report after getting the date of the flight you want to look at. The third and final option is to shut down. The shutdown sequence includes the saving of the program state so it is important that you let it go through the proper shutdown sequence.

# Appendix 4 – Trello Boards

A picture containing text, computer, indoor, screenshot

Description automatically generated

Figure

A picture containing text, indoor, screenshot, display

Description automatically generated

Figure

# Bibliography

Medium, n.d. *UML Class DIgrams Tutorial.* [Online]   
Available at: https://medium.com/@smagid\_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b  
[Accessed 30 November 2020].