```
                   _________________________________
 ___             /| ##### ##### #####   #   # ####  |
 \  \     _ _   / | #     #       #    # #  # #   # |
  \**\ ___\/ \<   | #     #       #   ##### # ####  |
X*#####*+^^\_\  \ | #     #       #   #   # # #  #  |
  o/\  \         \| ##### #####   #   #   # # #   # |
     \__\          ----------------------------------
```

COLLEGE OF COMPUTING TECHNOLOGY - DUBLIN
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

OBJECT ORIENTED CONSTRUCTS

Assessment 1 – Design of an airline schedule

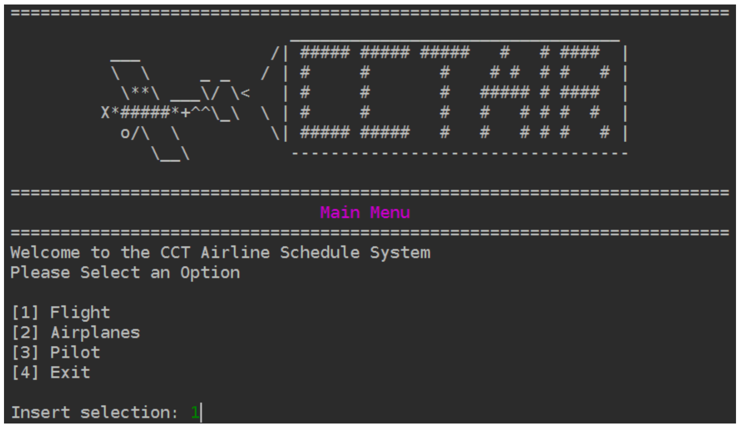
Adelo Vieira
Student Number: 2017279
Lecturer: Mr. Mark Morrissey
April 1, 2018

Contents

1 Purpose of the class

My main participation in the project was in the programming of the *class Menu*.

This class arises from the need to provide an easy way of interaction between the user and the system. We decided that the easier way of implement it was to create a selection Menu in the NetBeans Console. This Menu will display an Option Tree that will allows the user to enter the different options in a logical way and organized (See Figure 1.1).



```
=========================================================================
                            _________________________________
          ___             /| ##### ##### #####   #   # ####  |
          \  \     _ _   / | #     #       #    # #  # #   # |
           \**\ ___\/ \<   | #     #       #   ##### # ####  |
         X*######*+^^\_\ \ | #     #       #   #   # # #  #  |
           o/\  \         \| ##### #####   #   #   # # #   # |
              \__\          ---------------------------------

=========================================================================
                               Main Menu
=========================================================================
Welcome to the CCT Airline Schedule System
Please Select an Option

[1] Flight
[2] Airplanes
[3] Pilot
[4] Exit

Insert selection: 1
```

Figure 1.1: Main Menu output in the NetBeans console.

In Figure 1.2 we show the *InterfaceMenu* and ***class Menu***.

Figure 1.2: *InterfaceMenu* and ***class Menu***

2 Description of the class and reasoning behind the design

2.1 Constructor

```
public Menu(ArrayList<Pilot> p, ArrayList<AirPlane> a, ArrayList<Flight> f)
```

- In this class we need to initialize the data we will use. That is why the Constructor takes the list of Pilots, Airplanes and Flights.

2.2 Methods

The first thing we did was to create a method for the Main Menu of the system. This should print a Welcome message in the console along with the mains options (See Figure 1.1). Depending on the selected option, we implemented a set of method that manage the different Sub-Menus.

As show in Figure 1.1 the Main menu displays the options:

[1] **Flight**

[2] **Airplanes**

[3] **Pilot**

[4] **Exit**

We decided the best way to manage the option was entering a number associated whit each option.

It is very important to notice that our Menu will not have the function of creating flights, airplanes or Pilots. The menu will only be in charge of displaying the previously created data and updating Flights. Therefore, we decided:

[2] **Airplanes** - will only have the function of displaying the list of all Airplanes.

- The printing of the flights achieved through ***menuAirplanes()***. In this method we just print each AirPlane description thanks to the toString method implemented by Asmer in the *AirPlane* class.

[3] **Pilot** - will only display the list of Pilots.

- This action is achieved through the ***menuPilots()*** method. Because we wanted to print just certain information from Pilots, we asked Asmer (who was responsible of the Pilot class) to provide an appropriate method for this purpose: ***showPilotsSimplified()***.

[1] **Flight** - This will provides the different options responsible of updating Flight:

1 **Display All Flights:**

  ∗ This action is achieved through the ***subDisplayAllFlights()***. This method prints each Flight description thanks to the toString method created by Miguelantonio in the *Flight* class.

2 **Update Flight** - This option displays a menu through the ***updateFlight()*** method.

  1 **Update Pilot** - This operation is performed through the ***updateFlightsPilot()*** method.

    · We decided this option first had to provide the list of Flight. So the user can select the pertinent Flight. In order to do so, we wanted to show just a simplified list of Flight. That is why ***showFlightsSimplified()*** was programmed. This method was provided by Miguelantonio, who was responsible for the *Flight class.* After the user select a valid option from the list of Flights, the Menu then displays the list of Pilots. The user has to enter a valid option from that list of Pilots. At this point we had to make sure that the selected Pilots has the appropriate license to fly this airplane. Only after both selections are correct, the system will perform the update operation of the Flight.

  2 **Update Copilot** - Also performed by ***updateFlightsPilot()***.

  3 **Update Schedule** (Departure and Arrival times) - performed by ***updateSchedule()***

    · Here we used the ***updateTime()*** and ***validateTime()*** methods to make sure the user enters a time in the right format (hh:mm).

  4 **Update Arrival time**

cabecera() and **deleteMenu()**

These Methods and the drawing of the plane (See Figure 1.1) was provided by Miguelantonio.

The ***cabecera()*** method prints a header for every menu and sub-menu in our system. We considered the logo of the system should be displayed every time the user enter a Sub-menu. So this method is called by every Sub-menu method.

We also decided that it would be positive to implement a way of cleaning the console every time the user enters a submenu. That is way the method ***deleteMenu()*** was developed. This method provides elegance to our menu and helps the user to have a better visual experience.