**Module Title:** Software Development Fundamentals (semester 2)

**Assignment Type:** Group Application development project

**Group Size :** max group size of two

**Project Title:** Airline

**Project Date:** 22/10/2019

**Assignment Compiler:** Mark Morrissey

**Weighting:** worth 50%

**Due Date:** 22/11/2019 23:59

**Method of Submission:** Moodle Submission

**Feedback Method:** Moodle feedback

**Module Learning Outcome:**

1. understand and employ fundamental concepts and principles of programming such as variables, Boolean expressions, control flow structures, methods, arrays, etc.
2. demonstrate a structured approach to algorithmic design and problem solving and exhibit professional development best practices in designing and developing robust, maintainable software
3. illustrate and relate object-oriented concepts (encapsulation, inheritance, polymorphism) and employ them to solve practical, real-world problems

Project Brief:

Design an airline schedule to contain as **a minimum** the following.

Provide a user menu to :

Display all flights on the system

Allow a user to view a particular Flight

Allow a user to view all available aircraft

Allow a user to view a particular aircraft

Allow a user to view all Pilots

Allow a user to view a particular Pilot

Allow a user to create a flight(s)

System Development Requirements

Design and implement a class called **AirPlane** that will store data about a commercial aircraft.

Data for an airplane includes (but not limited to):

* make (type of plane, such as, “Boeing”)
* model (model number, for example, 707)
* capacity (seating capacity of the plane – a whole number up to 400) - pilot (name of pilot assigned to the aircraft)

Your implementation should include:

* 1. A constructor that accepts parameters **in the order given above** and initializes the corresponding instance variables;
  2. Getter methods as given below:

getMake() – return a string with the make of the airplane getModel() – return a number being the airplane’s model getPilot() – return (at a minimum) the pilot’s name capacity() – return a number being the capacity of the airplane.

* 1. A method called **assignPilot** which accepts the name of a pilot and assigns the pilot to the airplane ( this may also be created as an object to include the rating of the Pilot, ie. What type of planes they are qualified to fly).
  2. A **toString()** method which should return a string with the format: Airplane Information:

Aircraft : <<insert make>> <<insert model>>

Capacity: \_\_\_ seats

Pilot: \_\_\_\_\_

Design and implement a class called **Flight** that will store data about a commercial flight. Data includes:

* origin (e.g. “Dublin”)
* destination (e.g. “New York”)
* departure time (e.g. “10:10”) \*\*\*\*
* arrival time (e.g “13:30”) \*\*\*\*
* date of flight (e.g. “05/12/2015”)
* aircraft assigned (this should be an object);

Your Flight class implementation should include:

1. A constructor which accepts data (**except for departure time and arrival time**), in the order listed above and initializes each relevant instance variable accordingly.
2. A **toString()** method which should return a string with the format:

Flight Information:

Date: <<insert date of flight>>

From: <<insert origin>> to <<insert destination>>

Flight time: <<insert departure time>> to <<insert arrival time>> Plane Information:

Aircraft : <<insert make>> <<insert model>>

Capacity: \_\_\_ seats

Pilot: \_\_\_\_\_

In the Flight class, define the following methods(overload) that will set the scheduled time for a flight: void schedule(String arrivalTime) void schedule(String arrivalTime, String departureTime)

Implement a driver class (main method), called **CCTAir** that will do the following: **a.** Declare and initialize a number of flights. (create a minimum of 10 flights)

1. Allow a user to add up to 5 flights of their choice.
2. Use the second version of the schedule method to set the time schedule for a flight.
3. Use the first version of the schedule method to update the arrival time for a flight in the list.
4. display all flights.

**Marking Scheme Summary**

|  |  |
| --- | --- |
| **Object Oriented design** | 0 – 20 marks |
| **Use of object oriented programming principles / constructs** | 0 – 20 marks |
| **Program function / fit for purpose** | 0 – 40 marks |
| **Requirements gathering / analysis report** | 0 – 10 marks |
| **API Documentation / commented code** | 0 – 10 marks |

Total 100% weighted 50%

This project will be developed with a formative development methodology, developing the required classes / methods and eventual working program over a period of weeks.

Notes : (**program must run, no debugging will be attempted)**

Along with your reports / documentation you must only submit a **netbeans project folder**; all submitted materials should be compressed into a folder named after a student number from one of your group: eg 2016111.zip. Only one submission per group (be sure to include all group members details).

**Documentation:**

**API documentation** can be generated from within Netbeans (uses comments on your code)

**Requirements gathering and analysis:** This needs to be a detailed account of the functions of the program (step by step walkthrough with a typical user) and the reasoning behind your design choices (classes, inheritance, data structures… etc.). Documention must illustrate a deep understanding of the development process and clearly show the students understanding of the logical approach to algorithmic /program design. Your group members must submit an individual report detailing their role within the project; this document should be saved as:

Your name\_your student number.docx or .pdf.