Object Orientated Software Development

Project Assignment

Peter Leonard, D16124538

Flight Travel Planner

**Functional Requirements:**

The aim of this application was to create an object orientated software program designed to dynamically provide travel & flight information to the user. By inputting an aircraft’s or airport’s identification, the program will return the stored information of that object to the user. The user may also plan a route using the application. This is carried out by inputting the airport code for each destination as well as the aircraft code for the plane carrying out the journey. The program will then simulate the aircrafts journey to each airport and account for its range and fuel capabilities. This functionality is carried out by two different menu functions. The user has the option to find the shortest flight route for a given number of destinations. Separately, the user can then input a flight plan into the system. The program will then return the total distance traveled and the total cost in euro of the journey the user has input in the order given. All of this is achieved through a command-line interface accessed through the UserInterface.py file.

**Design:**

A major focus of the design for this application was encapsulation, ensuring that relevant data and necessary functions are tied together separately from the interference of other elements in the program. To this end, each major class; Airport.py, Aircraft.py and Currency.py are stored in their own file. One file named FileReader.py was used to instantiate all objects in the program, from which all other files could import necessary object information.  
Due to the specificity of this program, use of inheritance was restricted as each object created came from an entirely unique class. Had the project included alternate travel methods, such as boats or trains, a broader class would have been useful to create inheritance for different objects, however as this project only contains travel by plane to and from airports the use of inheritance for these classes is unnecessary.

Assumptions:

* All files for the project are stored within the same folder and all CSV files are stored within a sub-folder of the main.
* All program functionality is accessed through the UserInterface.py file.
* Interaction with the program is carried out directly by single user input.
* The program doesn’t accommodate journeys beyond the aircraft’s maximum flight range and will instead inform the user that the journey is beyond the aircraft’s capabilities.

Input Data:

Aircraft.csv, Airport.csv, NationalCurrencies.csv, CurrencyExchangeRates.csv

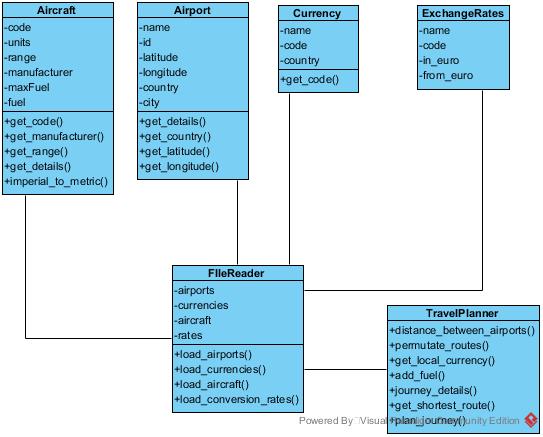
Output Data:

The returned data is output onto the command-line interface the user is interacting with and is not stored when the program is closed.

Testing:

To ensure the integrity of the program, a test case was written for calculating the distance between two airports(TestDistance.py). This was done by comparing the values of determined test cases (Airport Code Inputs) with the distance that these values should return as had been identified through external research of the given values.  
Continuous testing of the program was also carried out using print statements of the functions and methods being called to guarantee the expected data were being returned.

Class Diagram:

****

Classes:

All classes are built by the file-reader class. The file reader then communicates with the TravelPlanner, which stores the majority of the programs functions. The user is connected to the program through the UserInterface.py file. All of the programs functionality is split between these three files. They allow the user to access information stored on aircraft and airports in the program. They also allow the user to create a list of flights tied to an aircraft, with this information they can determine the flight cost and distance of a particular root or find the shortest for the given aircraft to visit each airport.

**Program Flow - Planning a Route:**

Yes

Yes

NO

NO

Add Fuel

Calculate Local Fuel Cost

Add to Total Cost

Input Aircraft

Sufficient Fuel Level?

Return Total Cost and Distance

Sufficient Maximum Range?

Return Error Message

Run distance calculation for airport set

Store User Input

Input the No. of desired Airports