Airline Fuel Manager

Software Development: OOP Project 1

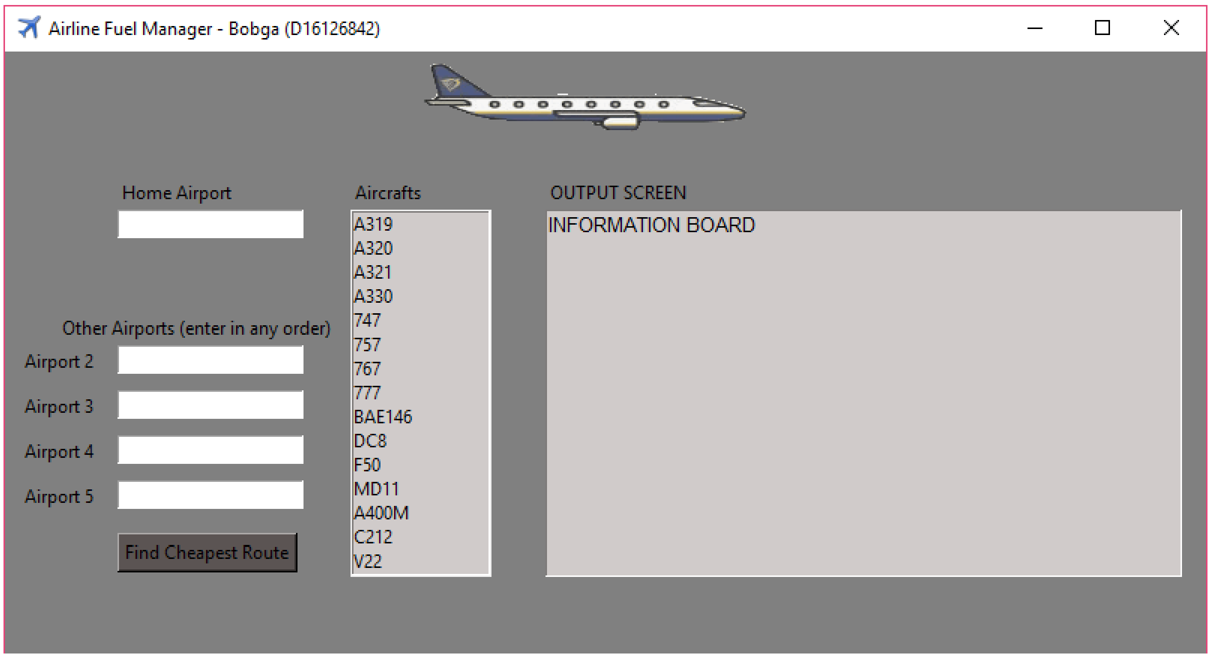
April 2017

Bob Bobga Nti

Student N0.: D16126842

Dublin Institute of Technology

Documentation



A319
A320
A321
A330
747
757
767
777
BAE146
DC8
F50
MD11
A400M
C212
V22

INFORMATION BOARD

Overview

The Airline Fuel Manager is a fuel management software written In Python 3.5

The program loads in the following as csv files

- aircraft.csv
- airport.csv
- countrycurrency.csv
- currencyRates.csv

The program imports csv, os, itertools, tkinter, tkinter.messagebox, defaultdict(from cllections)

The following class files are also imported: AirportAtlas, AircraftCatalog, CurrencyCatalog, CurrencyRatesCatalog

The program also receives input data representing 5 airports intended to visit, in any order, from the user.

The program then computes this information and return the most economic route

Interface

The Graphical Interface (GUI) is in the class called MainGUI.

This is built using the tkinter module. It consists of widgets that allow the user to input data. This data is integrated into the program by the get() method.

The resulting 'cheapest' route is displayed in an Information Board.

The program terminates when the user clicks on the 'X' button at the top-right corner of the program window.

Execution

Step	Action	Input data	Expected System Response
1	User inputs a home airport, any any-case: uppercase, lowercase	homeAirport	
2	User inputs 4 other airports in any order (airport2, airport3, airport4, airport5), In any-case: uppercase, lowercase	Airport2 Airport3 Airport4 Airport5	

3	User clicks on the 'Find Cheapest Route' button		• System reads in all the text box input using the .get() method. • System convert all inputs to uppercase string • System upload all necessary csv files • System checks that the code is in the 'airport.csv' file using the checkCode_airportDict() method of the AirportAtlas class • System displays the cheapest route combination of airports. The estimated cost price of fuel over the trip • The system also displays the route option with the shortest distance, sometimes different from the cheapest route option, indicating that the shortest distance might not necessarily by the cheapest route since the fuel might by costlier due to difference in exchange rates at each airport
	Test Blank fields		
4	Repeat step 1		
5	Leave any of the fields blank	" "	ErrorMessage! : 'You must fill in all boxes'
	Test Invalid data		
6	Repeat step 1		
7	Enter wrong airport code in any of the fields e.g. 'DUBS' for 'DUB'	DUBS	Alert message: "DUBS" is not found
	Test file upload		
8	Change filename/location of the csv file		ErrorMessage!: "File not found"
9	User clicks on 'OK'		System exits and the program is closed

Obtaining the Cheapest Route

The Boolean variables; isValidCode1, isValidCode2, isValidCode3, isValidCode4, isValidCode5 are used to store the output from the airport code verification (atlas.checkCode_airportDict(airportCode), from all the airport code inputs from the user (homeAirport, airport2, airport 3, airport4, airport5)

If all True, the system then takes the list of the 4 other airports, apart from the home airport, and carries out permutation to produce **24 route options** (4!=24). The resulting lists of route options are store in **permList**

permList = list(itertools.permutations(other_airtports_list))

Since the 5-trip weekly flight, without the optional extra stop, must begin and end at the home airport, the home airport is inserted into each route option of the permList:

```
for i in permList:
        i = list(i)
        i.insert(0, homeAirport)
        i.insert(len(i), homeAirport)
```

A "trip" is the distance between each airport. Thus, for the 5 trips we have trip1, trip2 trip3, trip4, trip5. The sum of all the trips of each "I" is the total distance for that "I".

Stages is the list of all the trips

total_distance_per_route = trip1 + trip2 + trip3 + trip4 + trip5

The "I" with the smallest distance is the shortest route option

A "trip_cost" follows the same argument as a "trip". But instead of representing the distance, they represent the cost of fuel bought at airport(a) to cover the distance to airport(b), in each "I".

total_cost_per_route = float("%.2f" % (trip1_cost + trip2_cost + trip3_cost + trip4_cost + trip5_cost))

A dictionary of **myDict_cost**[**total_cost_per_route**] **= i**

costs is a list of myDict_cost.keys()

smallest_cost = min(costs)

the **Cheapest Route** therefore is myDict_cost[smallest_cost]

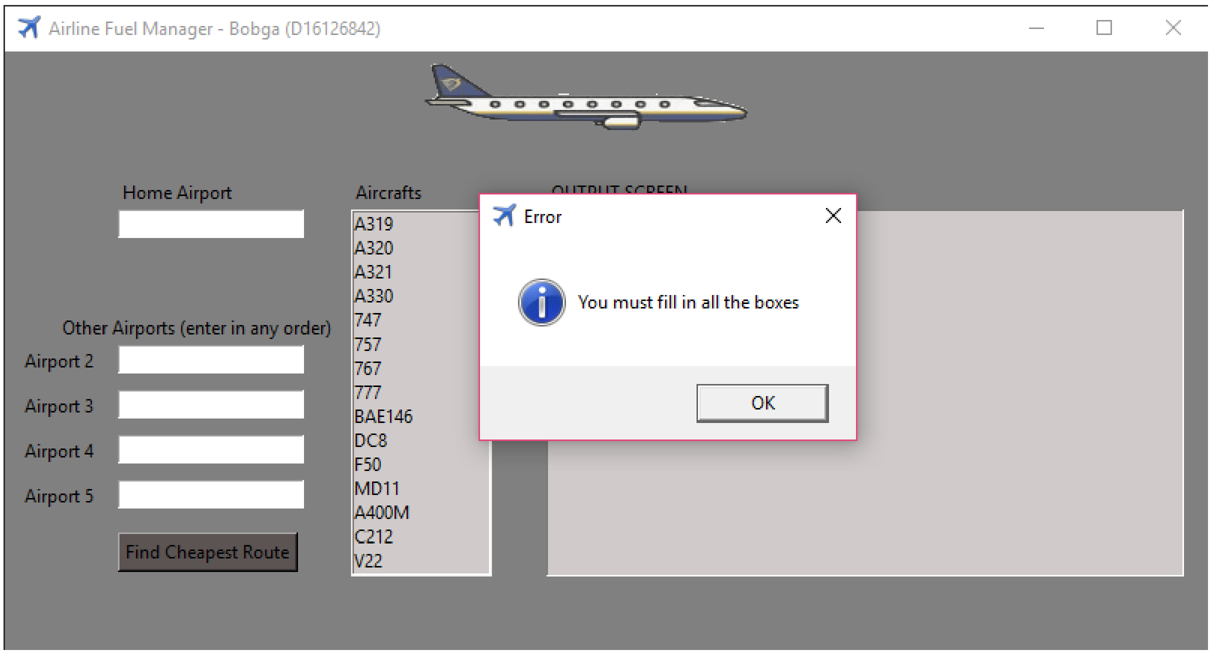
summary:

```
myDict_cost[total_cost_per_route] = i
costs = list(myDict_cost.keys())
smallest_cost = min(costs)
largest_cost = max(costs)
cheapest_route = myDict_cost[smallest_cost]
```

longestStage is the longest trip in myDict_cost[smallest_cost]

For any for any aircraft to embark on this journey, its range >= longestStage

Blank fields: If no data is input in any of the text boxes, the following message box is shown:


Airline Fuel Manager - Bobga (D16126842)

Home Airport

Aircrafts

OUTPUT SCREEN

Other Airports (enter in any order)

Airport 2

Airport 3

Airport 4

Airport 5

Find Cheapest Route

A319
A320
A321
A330
747
757
767
777
BAE146
DC8
F50
MD11
A400M
C212
V22

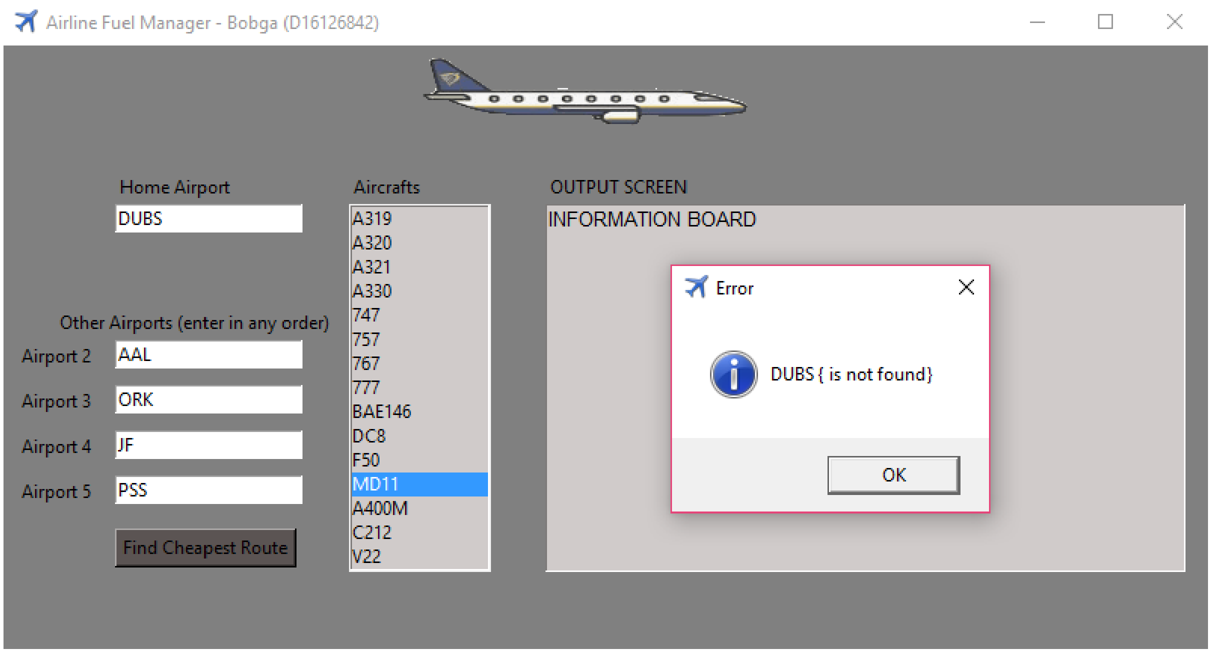
Error

You must fill in all the boxes

OK

When the user clicks ‘OK’, the file checks the next box, and so on until all the boxes are checked for valid data.

Invalid data: If any of the fields contain an invalid information the following error message is shown:


Airline Fuel Manager - Bobga (D16126842)

Home Airport: DUBS

Aircrafts

OUTPUT SCREEN

INFORMATION BOARD

Other Airports (enter in any order)

Airport 2: AAL

Airport 3: ORK

Airport 4: JF

Airport 5: PSS

Find Cheapest Route

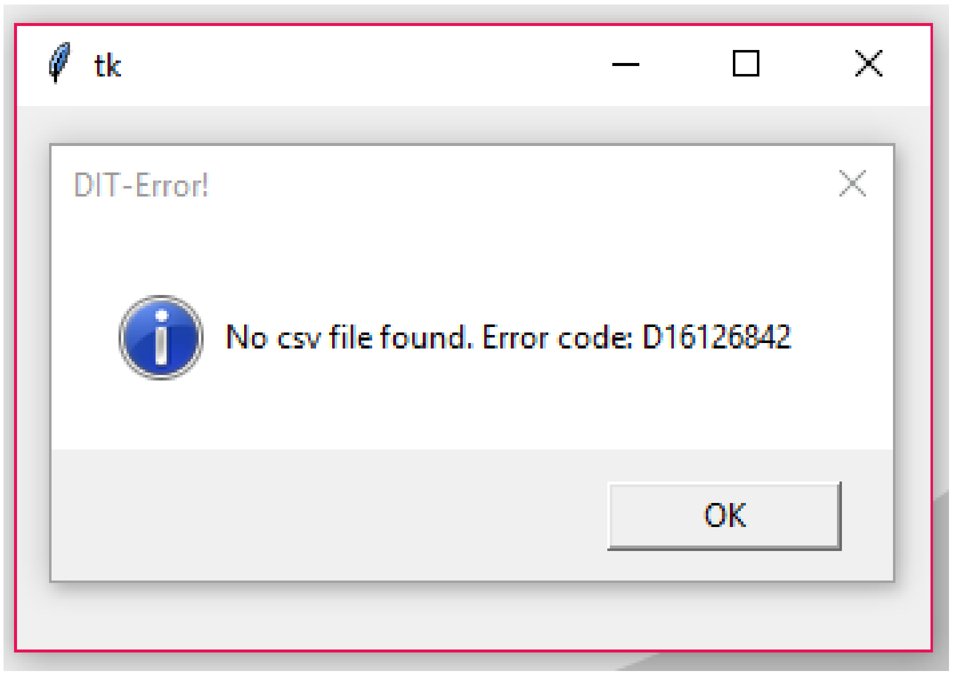
A319
A320
A321
A330
747
757
767
777
BAE146
DC8
F50
MD11
A400M
C212
V22

Error

DUBS { is not found}

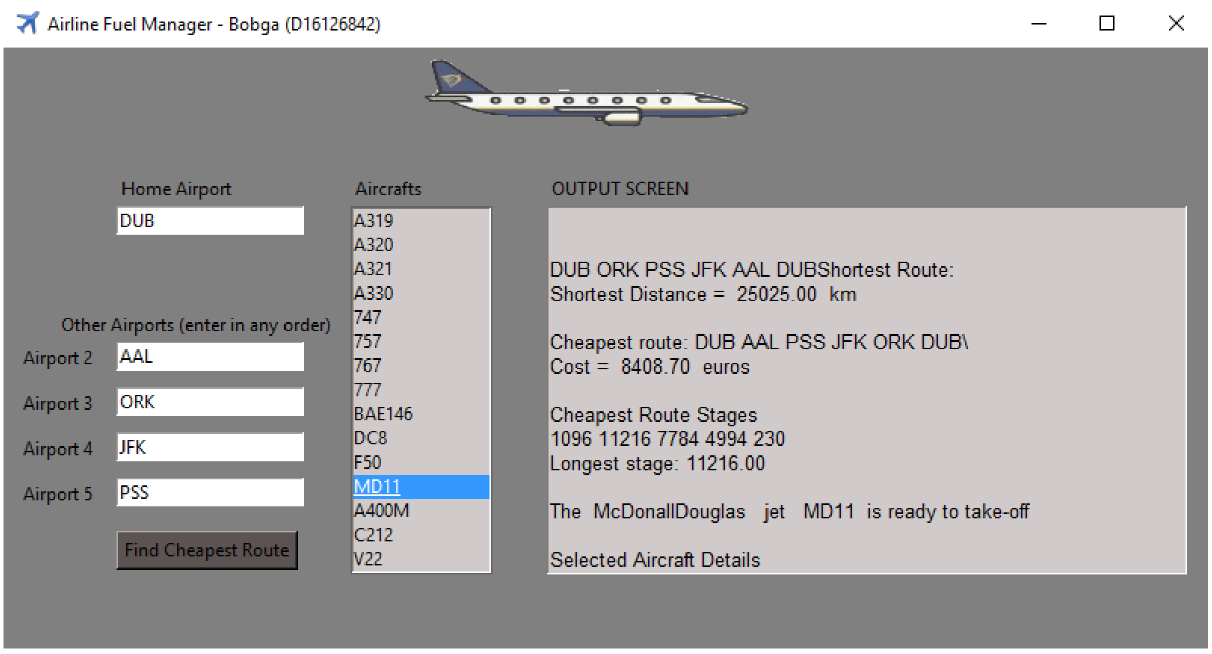
OK

File No Found Error: If the system doesn't find the required csv file, the following message box is shown:



No csv file found. Error code: D16126842

Information Board: When all the inputs are validated for correct data, the system then displays the result in the Information Board as follows:



Home Airport: DUB

Other Airports (enter in any order)

Airport 2	AAL
Airport 3	ORK
Airport 4	JFK
Airport 5	PSS

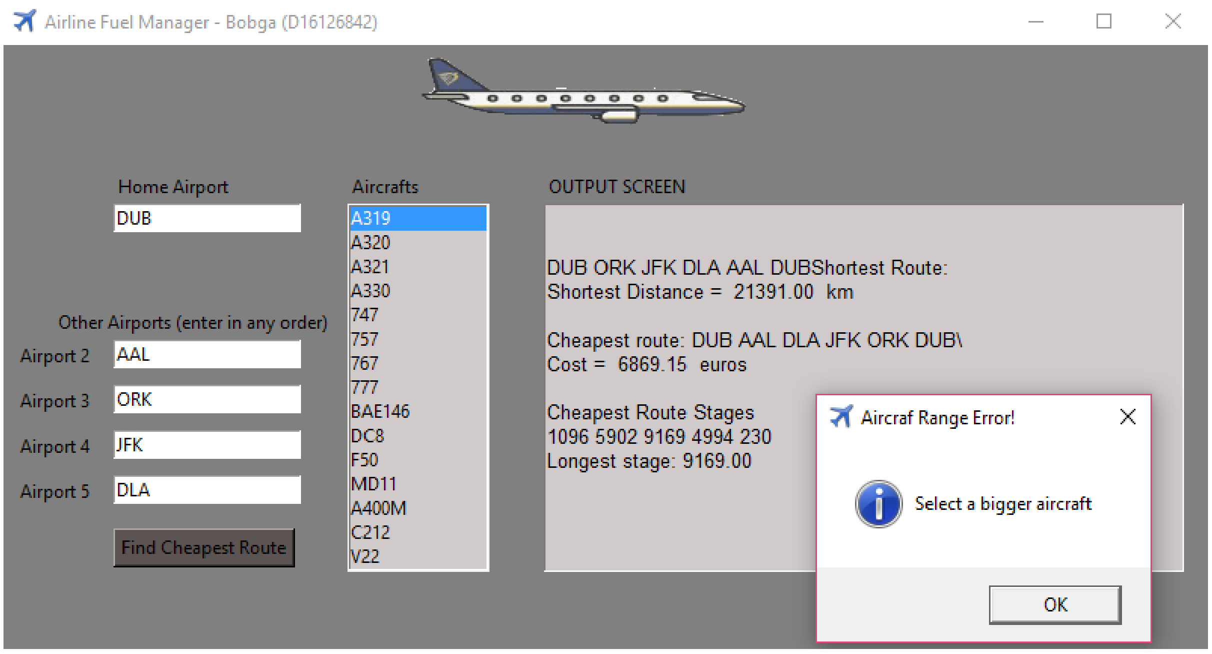
Aircrafts: A319, A320, A321, A330, 747, 757, 767, 777, BAE146, DC8, F50, MD11, A400M, C212, V22

OUTPUT SCREEN

```
DUB ORK PSS JFK AAL DUBShortest Route:
Shortest Distance =  25025.00  km

Cheapest route: DUB AAL PSS JFK ORK DUB\
Cost =  8408.70  euros

Cheapest Route Stages
1096 11216 7784 4994 230
Longest stage: 11216.00

The  McDonallDouglas   jet   MD11  is ready to take-off

Selected Aircraft Details
```

Spinning the mouse wheel will scroll up and down the board to expose more information.

Aircraft has insufficient range: If the range of the selected aircraft is less than the longest stage of the cheapest route, an error will promt the user to choose a bigger plane. See the figure below.



```
DUB ORK JFK DLA AAL DUBShortest Route:
Shortest Distance =  21391.00  km

Cheapest route: DUB AAL DLA JFK ORK DUB\
Cost =  6869.15  euros

Cheapest Route Stages
1096 5902 9169 4994 230
Longest stage: 9169.00
```

Select a bigger aircraft

Tests-Drive-Programs: Test drive programs was developed for each of the major components of the program to test the vital functionality.

1. test_Aircraft: This tests that the aircraft properties are displayed as desired
2. test_AircraftCatalog: This test ensures that the aircraft.csv file upload function is without fault.
3. test_Airport: This tests that the airport properties are displayed as desired
4. test_AirportAtlas: This test ensures that the airport.csv file upload function is without fault.
5. test_AirportAtlasGUI: This test the get_dist_between_airports() function and display result in a message box
6. test_CurrencyCatalog: This test ensures that the countrycurrency.csv file upload functions without fault

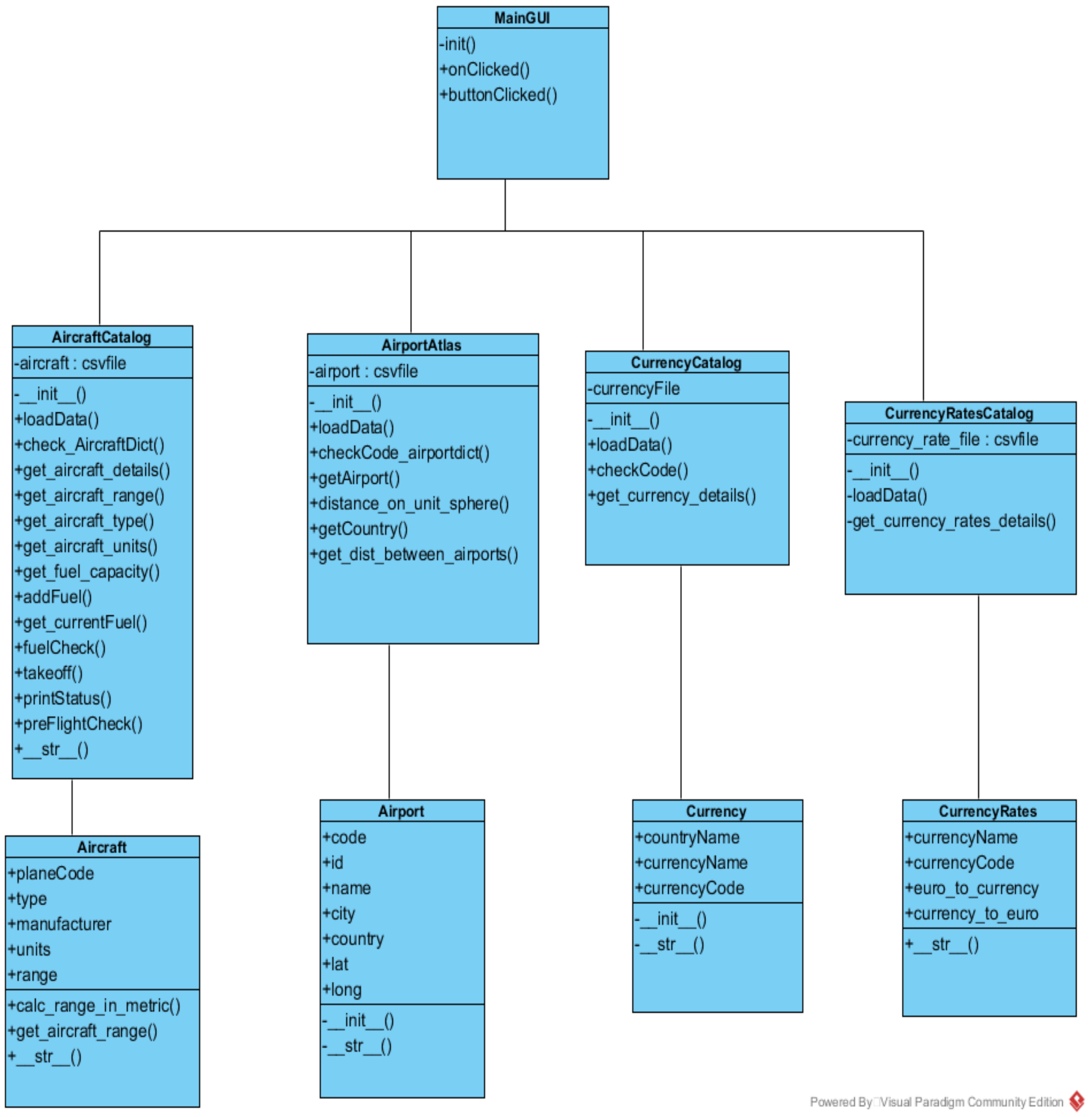
7. ttest_currencyRatesCatalog: This test ensures that the urrencyrates.csv file upload functions without fault

Unittest: Two unittest cases were written

1. unittest_AirportAtlas: This is a test for distance between known values.
2. Unittest_CurrencyRates: This tests the test_get_currency_rates_details() method

Program Structure

Class Diagram



Appedix

#	Class name	Description
1	MainGUI	This is the interface or the gateway into the program. The MainGUI inherits the Frame Class and presents a panel for sticking text boxes and buttons
2	AirportAtlas	This holds the properties and other information about the airport in a dictionary, uploaded from the airport.csv file **Properties:** airport.csv
3	Airport	This class defines how the properties of the Airport class are printed by calling the __str__() method **Properties:** code, id, name, city, country, lat, long
4	AircraftCatalog	This holds the properties and other information about aircrafts in a dictionary, uploaded from the aircraft.csv file **Properties:** aircraft.csv
5	Aircraft	This class defines how the properties of the Aircraft class are printed by calling the __str__() method **Properties**: planeCode, type, manufacturer, units, range
6	CurrencyCatalog	This holds the properties and other information about Country currencies in a dictionary, uploaded from the countrycurrency.csv file **Properties:** countrycurrency.csv
7	Currency	This class defines how the properties of the Currency class are printed by calling the __str__() method **Properties:** countryName, currencyName, currencyCode
8	CurrencyRatesCatalog	

		This holds the properties and other information about Country rates in a dictionary, uploaded from the currencyrates.csv file **Properties:** currencyrates.csv
9	currencyRates	This class defines how the properties of the CurrencyRates class are printed by calling the __str__() method **Properties:** currencyName, currencyCode, euro_to_currency, currency_to_euro