CS4092: Project 3 Outline (10%) - Flight Manager

Overview

You are required to develop a Java application to provide the functionality outlined herein to the end user. Input to your application should be via the command line (i.e. command-line arguments). You are expected to (i) validate user input and (ii) make appropriate use of methods. In addition, your code should not undertake any unnecessary comparisons and files should be maintained in the order specified here.

You are required to document your source code such that the input to, the processing performed by and the output from each programmer-defined method is detailed.

State clearly any assumptions you make.

Functionality

Your application will allow the end user manage flight details which should be stored in two data files. Airports.txt will hold the name of each airport and its airport code for example,

```
Aberdeen,ABZ
Belfast City,BHD
Dublin,DUB
New York,JFK
Shannon,SNN
Venice,VCE
```

This file should be maintained in ascending sequence on airport name.

Flights.txt will hold the details of flights and the structure of this file is as follows:

Flight Number,Source Airport Code,Destination Airport Code,Departure Time,Arrival Time,Start Date,End Date

e.g.

```
EI3240,DUB,ABZ,1310,1445,-TWTFS-,25/10/2017,24/12/2017
```

i.e. between the dates of 25/10/2017and 24/12/2017, flight EI3240 is scheduled to leave Dublin Airport for Aberdeen Airport at 13h10 and arrive at 14h45 on each Tuesday, Wednesday, Thursday, Friday and Saturday.

This file should be maintained in ascending sequence on Flight number.

The end user should be able to

1. add a new airport to Airports.txt
2. edit the name of an airport in Airports.txt
3. delete an airport from Airports.txt and its associated flights in Flights.txt
4. edit travel days and travel dates of a flight
5. delete a flight
6. search for flights by entering a source airport and a destination airport
7. search for flights by entering a source airport, a destination airport, a departure date

Your team should begin by deciding how the user will interact with your software. A suggestion is presented here for the first three functions.

Function #	Sample Command-Line Instruction
1.	java FlightManager AA Lisbon LIS *- add a new airport with the airport name of Lisbon and the airport code of LIS (if an airport with this code does not already exist)*
2.	java FlightManager EA BHD Belfast *- edit the name of an airport if it exists - in this case, change the name of the airport with the code BHD to Belfast*
3.	java FlightManager DA SNN *- delete entry in the Airports.txt file if it exists and also delete any entries in Flights.txt if they exist*
4.	java FlightManager EF EI3240 -TWTF-- 1/5/2017 31/10/2017 *- edit flight details for EI3240 changing the days and the times to those specified here as command-line arguments*
5.	java FlightManager DF EI3240 *- delete flight details for flight number EI3240*
6.	java FlightManager SF Dublin Shannon *- display details of all flights departing from Dublin and arriving in Shannon*
7.	java FlightManager SD Dublin Shannon 1/12/2017 *- display details of all flights departing from Dublin on 1/12/2017 and arriving in Shannon*

Deadline for Submission

All material must be submitted by 13h00 on Tuesday of Week 10. Please refer to the document titled CS4092: Project Submission Documentation for specific deliverable instructions.

Notes

- Assumption - there is only one record in Flights.txt for a Flight number
- Overall if we have less than 2 command-line arguments and greater than 5 command-line arguments then display an error message.
- You should first identify what is involved in the processing of each instruction and identify those methods that are common to one or more functions. It is best to write these methods first.

Suggested steps involved in Function 1 i.e. java FlightManager AA Lisbon LIS

```
if (Airport Code is not three characters in length)
   display error
else if (Airports.txt does not exist)
{
   create it
   add Airport name merged with a comma and merged with Airport code
   close file
}
else // Airports.txt already exists
{
   create an empty array list
   set searchText = "," + Airport Code
   set airportFound to false
   open Airports.txt
   while (data in file && !airportFound)
   {
        read a line from file
        if (lineFromFile.indexOf(searchText) != - 1)  // Airport Code already exists
           set airportFound = true
   }
   close file
   if (airportFound)
     display message to state airport already exists
   else
   {
      add Aiport Name merged with a comma and merged with Airport Code to array list
      order array list on ascending sequence
      open Airports.txt file
      sequentially scan the array list and write each line to the file (i.e. overwrite
                                                    contents of existing file)
      close the file
   }
}
```

I suggest you write steps similar to these for each function prior to writing any actual code.

A suggestion when validating days in Function 4.

```
set days = "MTWTFSS";
if (command-line argument is 7 characters in length)
{
    boolean validInput = true;
    String aCharacter;
    String input = value of particular command-line argument
    input = input.toUpperCase();
    for (int i = 0; i < days && validInput; i++)
    {
      aCharacter = input.substring(i, i+1);
      if (aCharacter is not a - or a character is not equivalent to the corresponding
                              characters in the string days)
          validInput = false;
    }
    if (!validInput)
       report error
}
```