CS410J Project 2: Storing An Airline in a Text File (9 points[1])

In this project you will write classes that store your airlines (and its flights) in a text file using a format of your own design.

Goals: Learn how to use Java's I/O facilities and throw exceptions.

For this project you will have to implement the following three classes[2] in your `edu.pdx.cs410J.`*your-login-id* package:

- Class `TextDumper` that implements the `edu.pdx.cs410J.AirlineDumper` interface. A `TextDumper` dumps the contents of an airline[3] (including flights) to a text file.
- Class `TextParser` that implements the `edu.pdx.cs410J.AirlineParser` interface. A `TextParser` reads the contents of a text file and from it creates an airline with its associated flights.
- Class `Project2` that contains a `main` method that optionally reads an `Airline` from the contents of a text file, creates a `Flight` as specified on the command line, adds the `Flight` to the `Airline`, and then optionally writes the `Airline` back to the text file. `Project2` should have the following command line interface:

```
usage: java edu.pdx.cs410J.<login-id>.Project2 [options] <args>
  args are (in this order):
    name                  The name of the airline
    flightNumber          The flight number
    src                   Three-letter code of departure airport
    departTime            Departure date and time (24-hour time)
    dest                  Three-letter code of arrival airport
    arriveTime            Arrival date and time (24-hour time)
  options are (options may appear in any order):
    -textFile file        Where to read/write the airline info
    -print                Prints a description of the new flight
    -README               Prints a README for this project and exits
  Dates and times should be in the format: mm/dd/yyyy hh:mm
```

If the text file does not exist, then the program should create an empty `Airline`. It should **not** issue an error. Dates and times are specified using the same format as in the previous assignment.

[1] 8 for code, 1 for POA

[2] Make sure to submit **all** of your classes, even if you haven't modified them since the previous project.

[3] Note that the `dump` method dumps an `AbstractAirline`. You cannot change this fact. This is okay because your `Airline` **is an** `AbstractAirline`.

Error handling: Your program should exit “gracefully” with a user-friendly error message under all reasonable error conditions. Examples of such conditions include

- Something is missing from the command line or there are extraneous command line arguments
- The format of the day or time is incorrect
- The text file is malformatted
- The airline name given on the command line is different than the one found in the text file.