CS410J Project 4: A REST-ful Airline Web Service (13 points[1])

In this project you will extend your airline application to support an airline server that provides REST-ful web services to an airline client.

Goals:

- Write a web application in Java
- Work with HTTP-based network communication

For this project you will implement the following classes in the `edu.pdx.cs410J.`*loginid* package:

- An `AirlineServlet` that provides REST access to an `Airline`. The servlet should be deployed in a web application named `airline` and should support the following URLs:
  - `http://host:port/airline/flights?name=airline`
    - `GET` returns all flights in the airline formatted using the `PrettyPrinter`
    - `POST` creates a new flight from the HTTP request parameters `name`, `flightNumber`, `src`, `departTime`, `dest`, and `arriveTime`. If the airline does not exist, a new one should be created.
  - `http://host:port/airline/flights?name=airline&src=airport&dest=airport`
    - `GET` returns all of given airline's flights that originate at the `src` airport and terminate at the `dest` airport.
- Class `Project4` is a client program that sends HTTP requests to the server. Dates and times should be specified using the same format as previous project (AM/PM, not 24-hour).

  If the `-search` option is provided, only the `name`, `src` and `dest` are required. The client should pretty print to standard out all of the direct flights that originate at the `src` airport and terminate at the `dest` airport.

```
usage: java edu.pdx.cs410J.<login-id>.Project4 [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/time
    dest                  Three-letter code of arrival airport
    arriveTime            Arrival date/time
  options are:
    -host hostname        Host computer on which the server runs
    -port port            Port on which the server is listening
```

[1] 12 for code, 1 for POA

```
-search               Search for flights
-print                Prints a description of the new flight
-README               Prints a README for this project and exits
```

It is an error to specify a host without a port and vice versa.

The client can perform several functions:

- Add a flight to the server:

  ```
  $ java edu.---.Project4 -host cs.pdx.edu -port 12345 "AirDave" 123 \
    PDX 07/19/2017 1:02 pm ORD 07/19/2017 6:22 pm
  ```

- Search for a flight between two airports. The below command line should pretty-print all direct flights that originate at PDX and terminate at LAS. A message should be printed if there is no direct flight between the specified airports.

  ```
  $ java edu.---.Project4 -host cs.pdx.edu -port 12345 -search \
    "AirDave" PDX LAS
  ```

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

- The syntax of the command line is invalid
- The format of the day or time is incorrect
- A connection to the server cannot be established

To get you started working with web applications, I put together a Maven archetype that creates a skeleton project.

```
mvn archetype:generate \
  -DarchetypeGroupId=edu.pdx.cs410J \
  -DarchetypeArtifactId=airline-web-archetype \
  -DgroupId=edu.pdx.cs410J.<login> \
  -DartifactId=airline -Dversion=Summer2017
```

(Note that the `artifactId` is `airline` which is probably the same as your Project 1 Maven project. Since the name of the artifact is the name of the web application (which is part of the URL), it really needs to be `airline`. Rename the directory that you created for Project 1 before creating the Maven project for this assignment.)

The archetype project contains an `AirlineServlet` and a `Project4` class. You can run the servlet with Jetty:

```
$ mvn jetty:run
```

You can run the main class as an "executable" jar:

```
$ java -jar target/airline.jar
```

The archetype also creates an integration test that drives the `Project4` main class. Since it requires that the server is running, it is run in the "integration test" phase of the Maven build. If you run this command, it will build and start the web container, run all unit tests, and shut the server down again.

```
$ mvn intergation-test verify
```

More notes:

- Take a look at the `edu.pdx.cs410J.servlets.FamilyTreeServlet` class for ideas on how you could implement the REST functionality.
- The client uses the `edu.pdx.cs410J.web.HttpRequestHelper` class in the `examples.jar` to make requests on the server.
- You only need to submit your source code (`.java` files) for this assignment. I will generate a Maven project containing the appropriate `pom.xml` and `web.xml` into which I will copy your source.