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

In this project you will extend your phone bill application to support an phone bill server that provides REST-ful web services to a phone bill 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:

- A `PhoneBillServlet` that provides REST access to an `PhoneBill`. The servlet should be deployed in a web application named `phonebill` and should support the following URLs:
  - `http://host:port/phonebill/calls?customer=name`
    - `GET` returns all calls in the phone bill formatted using the `PrettyPrinter`
    - `POST` creates a new call from the HTTP request parameters `customer`, `callerNumber`, `calleeNumber`, `startTime`, and `endTime`. If the phone bill does not exist, a new one should be created.
  - `http://host:port/phonebill/calls?customer=name&startTime=start&endTime=end`
    - `GET` returns all of given phone bill's calls that occurred between the `startTime` and the `endTime`.
  - Note that, unlike previous assignments, the web application must support multiple phone bills.
- 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 `customer`, `startTime` and `endTime` are required. The client should pretty print to standard out all of the phone calls made between those two times.

```
usage: java edu.pdx.cs410J.<login-id>.Project4 [options] <args>
  args are (in this order):
    customer               Person whose phone bill we're modeling
    callerNumber           Phone number of caller
    calleeNumber           Phone number of person who was called
    startTime              Date and time call began
    endTime                Date and time call ended
  options are (options may appear in any order):
    -host hostname         Host computer on which the server runs
    -port port             Port on which the server is listening
    -search                Phone calls should be searched for
    -print                 Prints a description of the new phone call
    -README                Prints a README for this project and exits
```

[1] 12 for code, 1 for POA

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

The client can perform several functions:

- Add a phone call to the server:

```
$ java -jar target/phonebill.jar edu.pdx.cs410J.<login-id>.Project4 \
  -host linux.cs.pdx.edu -port 12345 "Dave" \
  503-245-2345 765-389-1273 02/27/2018 8:56 am 02/27/2018 10:27 am
```

- Search for the calls begun between certain times. The below command line should pretty-print all calls made in the month of March. A message should be printed if no calls during the time interval.

```
$ java -jar target/phonebill.jar edu.pdx.cs410J.<login-id>.Project4 \
  -host linux.cs.pdx.edu -port 12345 -search "Dave" \
  03/01/2018 12:00 am 03/31/2018 11:59 pm
```

- Pretty print all phone calls in a phone bill.

```
$ java -jar target/phonebill.jar edu.pdx.cs410J.<login-id>.Project4 \
  -host linux.cs.pdx.edu -port 12345 "Dave"
```

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 that modles a simple "dictionary" application that contains words and definitions. You will have to make significant changes to this code for your project.

```
mvn archetype:generate \
  -DarchetypeGroupId=edu.pdx.cs410J \
  -DarchetypeArtifactId=phonebill-web-archetype \
  -DgroupId=edu.pdx.cs410J.<login> \
  -DartifactId=phonebill \
  -Dversion=Summer2018
```

(Note that the `artifactId` is `phonebill` 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 `phonebill`. Rename the directory that you created for Project 1 before creating the Maven project for this assignment.)

The archetype project contains an `PhoneBillServlet` 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/phonebill.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 integration-test verify
```

The archetype contains an `index.html` file that lets you experiment with the REST API by submitting an HTML form. Note that this file is for experimentation purposes only, is not part of the grade, and should not (cannot) be submitted with the rest of your project.

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 can run your Jetty server on a port other than 8080 by setting the `jetty.http.port` variable in Maven:
  - `$ mvn jetty:run -Djetty.http.port=8888`
- 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.