CS601-01/CS601-02: Lab 4

Using Google Places API to fetch info about tourist attractions

Due Date: October 18th, 11:59pm

In lab4, you will use Google Places API Web Service to fetch information about tourist attractions near the hotels. Specifically, for each hotel in in the provided json files (hotels1.json and hotels4.json), you need to find a list of tourist attractions within a certain radius from it.

For this lab, you may **not** use the `URLConnection` class or any third party libraries except for `JsonSimple`. You need to use `SSLSocket-s` to communicate with the web server.

Before you start:

- Get an API key for Google Places API Web Service (click on the link below and follow the instructions). Note that getting an API key is free, you do **not** need to pay for any service or product.
  `https://developers.google.com/places/web-service/get-api-key`

  Important Note: Google gives you a free API key for this web service, but the number of API calls you can make in a day is limited. It's crucial that you work on this assignment over the course of multiple days - if you delay and try to finish it in the last couple of days, it's highly likely that you will exceed your quota of the # of API calls per day and won't be able to debug your code properly.

- We will be using `The Google Places API Text Search Service`, so you need to click on the link below and read the section called "**Text Search Requests**" at `https://developers.google.com/places/web-service/search`
  It describes the url request in detail. Before you start coding, you should play with the sample requests listed in the document above - I recommend typing these requests in the browser.
- The starter code for the lab is available at https://classroom.github.com/a/3mPHOgw1
  Note that you'd need your files from lab 3 (Hotel, Address, Review, HotelData, ThreadSafeHotelData, HotelDataBuilder, ReentrantReadWriteLock etc) for this lab - copy them from lab 3 part 1 (or part 2, does not matter) into the appropriate folder.

Requirements:

1. Fill in code in `TouristAttraction`. The class should store `attractionId, name, address, and rating` (feel free to store other information as well). It also needs to have a constructor, getters and a toString method.

2. Fill in code in `TouristAttractionFinder`, whose job is to fetch data from Google Places API. `TouristAttractionFinder` should have
    - a constructor that takes `ThreadSafeHotelData` as a parameter,
    - `public void fetchAttractions(int radiusInMiles)` method. This method should do the follwing:
        - open an `SSLSocket` to "talk to" the `maps.googleapis.com` webserver,
        - prepare an HTTP GET request to the Places API Text Search Service. Use version 1.1 of the HTTP protocol. Specify json as the output type. Make sure you pass the radius as a parameter. Note that the API expects radius in **meters**, while the fetchAttractions method takes a radius in **miles**. Here is a sample request:

```
GET /maps/api/place/textsearch/json?query=tourist%20attractions+in+Santa%20Marta&location=-74.225873,11.203319
&radius=20&key=AIzaSyCX_q6ipbMi5Xh2G_9Uylwh_bOkURs4T3I HTTP/1.1
Host: maps.googleapis.com
Connection: close
```

        - send the request to the webserver via the output stream of the socket,
        - receive the response; remove the headers
        - parse the json you got in the response, create `TouristAttraction` instances, and add them to a data structure (or data structures) in `TouristAttractionFinder`. The data structures should allow us to quickly get a list of nearby tourist attractions by the id of the hotel. Alternatively, you can store info about the attractions in ThreadSafeHotelData.
        - close the streams and the socket.

- `printAttractions(Path file)` that outputs information about the attractions to the file. The format of the file should be as following:
  Attractions near hotelName, id
  TouristAttraction

  TouristAttraction

  TouristAttraction

  ++++++++++++++++++++
  Attractions near hotelName, id
  TouristAttraction

  TouristAttraction

  TouristAttraction

  ++++++++++++++++++++

Tests
Use Lab4Test file in the test folder to test your code.

Javadoc Comments

You are required to add javadoc comments to your code (above each class and above each public method).

Submission:

Your lab4 should be submitted to your private lab4-username repo on github. Please keep pushing your code to github as you work on the lab. You are required to have at least 4 meaningful commits before the deadline.

Extra Credit: You might want to make the process of fetching attractions **multithreaded.** Discuss your algorithm with the instructor before you implement it. Finish the required part of the assignment before you start working on the extra credit.