Joanna Klukowska
joannakl@cs.nyu.edu

Assignment 9

Due date: May 8, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but **all work for all assignments must be entirely your own**. Any sharing or copying of assignments will be considered cheating.

You should not use any features of Java that have not been covered in class. If you have doubt if you are allowed to use certain structures, just ask your instructor.

Introduction

In this project you will develop a program that given a New York City zip code, displays the data about all the Wi-Fi hotspots available near-by.

The program should work with two input files: one containing the data about locations of Wi-Fi Hotspots, the other containing the information about the "locations" of NYC zip codes. The user of the program should be able to enter a zip code that she/he is interested in and the program should produce the list of all of the closest five wi-fi hotspots. Once this is displayed the user should be given a choice of viewing five additinal wi-fi hotspots, entering a different zip code or terminating the program.

The design of the user interface is up to you. Make sure it is intuitive and that the user has clear directions for what to do.

Your program should validate all the data entered by the user and should not crash if the user enters invalid information.

Data Sets for This Project

NYC Wi-Fi Hotspot Locations

You will be working with open data from NYC Open Data project, https://data.cityofnewyork.us/. Specifically, you will be working with the data set that contains a list of NYC Wi-Fi Hotspots locations
https://data.cityofnewyork.us/City-Government/NYC-Wi-Fi-Hotspot-Locations/yjub-udmw.
Note: you do not need to download your own data. The entire data set **`NYC_Wi-Fi_Hotspot_Locations.csv`** in the format that you should use is located on the course homepage - download it from there.

Zip Code Locations

In order to find Wi-Fi Hotspots closes to a particular zip code, you will need the longitude and lattitude information for each NYC zip code. This data is located in another CSV file provided with this assignment: **`zip_codes_NYC.csv`** [1]

Parsing Input Files in CSV Format

Your program needs to read the CSV files and parse each line into appropriate fields. CSV stands for *comma separated values* - each field is separated by commas and some of the fields (the ones that contain text) may be surrounded by double quotes.

The code below shows you how to split a single line read from a CSV file and parse it into separate entries that are text. Once you have these strings you can convert them into appropriate types for use in your code. You can use this code as is in your program.

```
1  /**
2   * Splits a given line according to commas (commas within entries are ignored)
3   * @param textLine line of text to be parsed
4   * @return an ArrayList object containing all individual entries/tokens
5   * found on the line.
6   */
```

[1] The full version contains all of the zip codes in USA. See: https://www.gaslampmedia.com/download-zip-code-latitude-longitude-city-state-county-csv/.

Joanna Klukowska
joannakl@cs.nyu.edu

```
7   public static ArrayList<String> split (String textLine ) {
8     ArrayList<String> entries = new ArrayList<String>();
9     int lineLength = textLine.length();
10    StringBuffer nextWord = new StringBuffer();
11    char nextChar;
12    boolean insideQuotes = false;
13
14    for(int i = 0; i < lineLength; i++ ) {
15      nextChar = textLine.charAt(i);
16      //add character to the current entry
17      if ( nextChar != ',' && nextChar != '"' ) {
18        nextWord.append( nextChar );
19      }
20      //double quote found, decide if it is opening or closing one
21      else if (nextChar == '"' ) {
22        if ( insideQuotes ) {
23          insideQuotes = false;
24        }
25        else {
26          insideQuotes = true;
27        }
28      }
29      //found comma inside double quotes, just add it to the string
30      else if (nextChar == ',' && insideQuotes) {
31        nextWord.append( nextChar );
32      }
33      //end of the current entry reached, add it to the list of entries
34      //and reset the nextWord to empty string
35      else if (nextChar == ',' && !insideQuotes) {
36        //trim the white space before adding to the list
37        entries.add( nextWord.toString().trim() );
38
39        nextWord = new StringBuffer();
40      }
41
42      else {
43        System.err.println("This should never be printed.\n");
44      }
45    }
46    //add the last word
47    //trim the white space before adding to the list
48    entries.add( nextWord.toString().trim() );
49
50    return entries;
51  }
52
```

Distance Computation

Given a particular zip code entered by the user, the program needs to find the distance between that zip code (its longitude and latitude) and all of the hotspots and then pick five (or more) that are closest.

The formula for computing the distance beween two points on Earth given their longitude and latitude is provided by Haversite formula, see https://en.wikipedia.org/wiki/Haversine_formula for more details.

You can use the following Java method[2] to compute it:

```
1
2 double haversine(double lat1, double lon1, double lat2, double lon2) {
3   final double R = 6372.8; // In kilometers
```

[2] source: http://rosettacode.org/wiki/Haversine_formula

Joanna Klukowska
joannakl@cs.nyu.edu

```
4   double dLat = Math.toRadians(lat2 - lat1);
5   double dLon = Math.toRadians(lon2 - lon1);
6   lat1 = Math.toRadians(lat1);
7   lat2 = Math.toRadians(lat2);
8
9   double a = Math.pow(Math.sin(dLat / 2),2) + Math.pow(Math.sin(dLon / 2),2)
10                * Math.cos(lat1) * Math.cos(lat2);
11  double c = 2 * Math.asin(Math.sqrt(a));
12  return R * c;
13 }
14
```

Class Design

You should implement at least three different classes:

`FindHotspots` - this is the actual program with **`main`** method. This is the class that should read the input from the input files, provide the user interface, perform the computations and print the results. You should consider a subdivinding it into several methods that will accomplish these tasks.

`Hotspot` - this is a class that represents a single wi-fi hotspot. The objects do not need to store all of the information about the hotspots that are listed in the input file, but should use enough to provide useful information to the user (where is it, is it free or not, etc) as well as the longitude and latitude information for computation of distances.

`ZipCode` - this is a class that represents a single zip code. It should store the information about the longitude and latitude so that they can be used for distance computations.

You may implement additional classes if you wish.

What and how to submit?

You should submit **all source code files** compressed into a single **zip** file to NYU Classes. Do not submit all the files that Eclipse creates, just the source code files that have .java extension. **Do not submit the `core.jar` file.**

If you wish to use your (one and only) freebie for this project (one week extension, no questions asked), then complete the form at
http://goo.gl/forms/fpUJrF64b5

before the due date for the assignment. All freebies are due seven days after the original due date and should be submitted to NYU Classes.

Questions

Post any questions you have regarding this assignment to NYU Classes Forums under the Assignments topic.