LEBANESE AMERICAN UNIVERSITY
Electrical and Computer Engineering Dept

COE 312
Data Structures

Fall 2013

W. FAWAZ

Project I

I. Objective

In this project, you will develop a software tool that uses a provider of metrological content to display a **map showing a 5-day forecast for the major cities around the world**. In particular, the cities that will be targeted by this "**Capital Weather Forecast Tool**" are, as the name of the tool suggests, the country capital cities from all over the world.

This somewhat complicated task will be decomposed into **three smaller manageable subtasks** to make the process of developing this software tool possible.

The proposed **implementation strategy** is built upon two main pillars, namely the use of a "divide and conquer"-like approach to solve the problem and the gradual integration of features into the "Capital Weather Forecast Tool".

The details pertaining to the aforementioned **three tasks** are provided in what follows.

II. First task

Your first task would be to use the **Google Geocoding API** to map the names of all of the capital cities that exist in the world into geographic coordinates (like latitude 37.423021 and longitude-122.083739). The resulting coordinates will serve as an input for the second task, where you will use the coordinates to obtain the 5-day weather forecast for each capital city.

Translating an address to a pair of geographic coordinates involves using the Google Geocoding API that provides a means for accessing a geocoder as explained at:

https://developers.google.com/maps/documentation/geocoding/#XML

As illustrated in this webpage, an XML response containing the geographic coordinates can be fetched from the geocoding API. Although it is possible to get a JSON version of the coordinates; for this task, an **XML version of the coordinates should be requested from the API**. Once the **XML-formatted coordinates** have been received your next step would be to extract the latitude and longitude coordinates contained inside the received XML response. Armed with these two pieces of information, you will be able to attack the second task of this project.

To accomplish this first task, you are required to use the **JDOM parser** to extract the desired information from the retrieved feed. It goes without saying that this can be achieved only after you study carefully the structure of the XML feed with the purpose of identifying **the tags** enclosing the location-related information for each city. Once you have successfully extracted the information, you would be ready to tackle the **second task** whose guidelines are delineated in the following section.

The process explained above should be repeated for each of the capital cities.

III. Second task

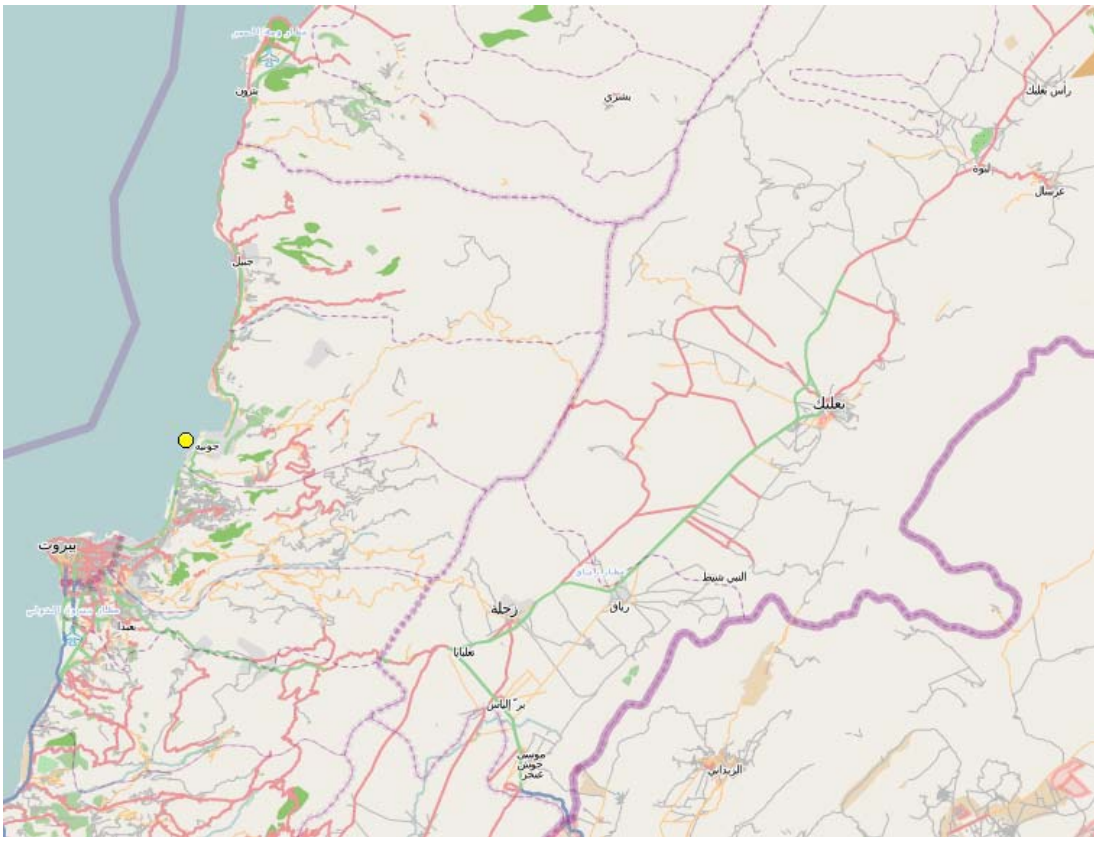
For the second task, your job is **to find a JSON feed** allowing for the retrieval of meteorological data for cities across the globe and more importantly one supplying forecast weather data for these cities. Once you have located a content provider that meets these requirements, you can proceed to connecting to the identified JSON feed and parse it. The main purpose of the parsing process is to extract the weather forecast data in terms for instance temperature values, wind speed, humidity and rainfall levels (if possible) for each country capital. The content provider should be able to generate the JSON feed based on the GPS coordinates that resulted from the first task. That is, you are required to submit to the content provider the longitude and latitude information retrieved earlier to get the weather forecast data pertaining to the capital city that corresponds to these longitude and latitude pieces of data.

What makes this task very challenging is the fact that you are required to: a) do **some research** on your own with a view to finding an appropriate content provider; b) then, use the information generated by the first task to **query the content provider for JSON-formatted weather forecast data**; c) and finally, extract the forecast data of interest, i.e. the ones relating to a time window that is 5 days long and particularly targeting the **next 5 days**.

IV. Third task

At this stage, you are assumed to have **completed the first two tasks correctly**. This is particularly true since the third task will make heavy use of the **latitude and longitude information** generated by the first task and of **the weather forecast data** produced by the second task. Your job at this point is to integrate into your Java application a map with **a set of markers** showing the locations associated with **the latitude and longitude coordinates** resulting from the first task. For illustration purposes, I am enclosing on the next page a map view displaying a yellow marker that points to the city of Jounieh, Lebanon.

You have to produce a similar view but with several markers pinpointing the positions of the capital cities. In addition to the markers, you are required to couple each capital city with a **summary list** presenting the **meteorological data** characterizing the weather condition of that city over the next 5 days. Specifically, for each city you should supply a summary list reporting information about the expected temperature values, wind speed, humidity and rainfall levels for the next 5 days. **Image icons** capturing the reported weather condition should also be added to the map. In other words, an icon showing the sun should be placed next to a set of meteorological data suggesting a hot weather and one showing a cloud should be inserted next to a set indicating a cold weather condition, and so on and so forth.



To add a map to your java application, you can use the **JMapViewer** java component. The following URL: http://wiki.openstreetmap.org/wiki/JMapViewer is a good source of information in this regard. The **external library** (.jar file) that contains the JMapViewer java class can be downloaded from:

http://svn.openstreetmap.org/applications/viewer/jmapviewer/releases/2011-02-19/JMapViewer.zip

Moreover, the **HTML documentation** for the JMapViewer class and other relevant classes can be found at: http://josm.openstreetmap.de/doc/. This URL is very important as it will allow you to gain a deeper understanding of both the JMapViewer class and the other classes that might be needed to accomplish the third task.

What to turn in?

This project is due at the beginning of class on the due date. You have to turn in the following material in both **hard** and **soft** copies.

Criteria	Percentage
Documentation of your solution including explanations and illustrations in one or two pages along with short write-up of questions and/or problems that you encountered while doing this assignment.	2 pts (10%)
Source code that contains an appropriate amount of comments. Well-organized and correct code receives 16 pts, messy yet working code receives 10 pts, code with bugs receives 2 pts, and incomplete code receives 1 pt.	16 pts (80 %)
Execution output such as a snapshot of the contents of standard output. A correct output receives 2 pts, the one with minor errors receives 1 pt, and an incomplete output receives 0 pts.	2 pts (10%)
Total	20 pts (100%)

Good Luck!