LEBANESE AMERICAN UNIVERSITY
Electrical and Computer Engineering Dept

COE 593
COE Application

Fall 2013

W. FAWAZ

Project I

I. Objective

In this project, you are tasked with developing a software tool that uses a weather feed to display a **map showing the cities that are currently experiencing severe weather conditions**.

This somewhat complicated task will be decomposed into three smaller manageable subtasks to make the process of developing this "**Extreme Weather Viewer 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 "extreme weather viewer" software tool.

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

II. First task

For the first task, your job is **to find an XML feed** providing meteorological and climate-related data for cities around the world. Once you have located a proper content provider to this end, you can proceed to connecting to the identified XML feed and parse it. The main purpose of the parsing process is to extract information revealing the location of the cities that are suffering from an **extreme weather condition**.

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 made available by that content provider to categorize cities into two groups, namely cities with normal weather and ones with severe weather conditions. This classification can be done based on a variety of factors, including for instance the temperature level, wind speed, humidity and rainfall levels; c) and finally, extract information about the location of the cities found to be plagued with severe weather.

To accomplish this first task, you are required to use the **JDOM parser** to fetch 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.

III. Second task

Given that the information produced by the first task represents an address, your second task would be to use the **Google Geocoding API** to convert that address into geographic coordinates (like latitude 37.423021 and longitude-122.083739). The resulting coordinates will serve as an input for the third task where you will use the coordinates to add a marker, an icon and the collected meteorological data to a map.

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/

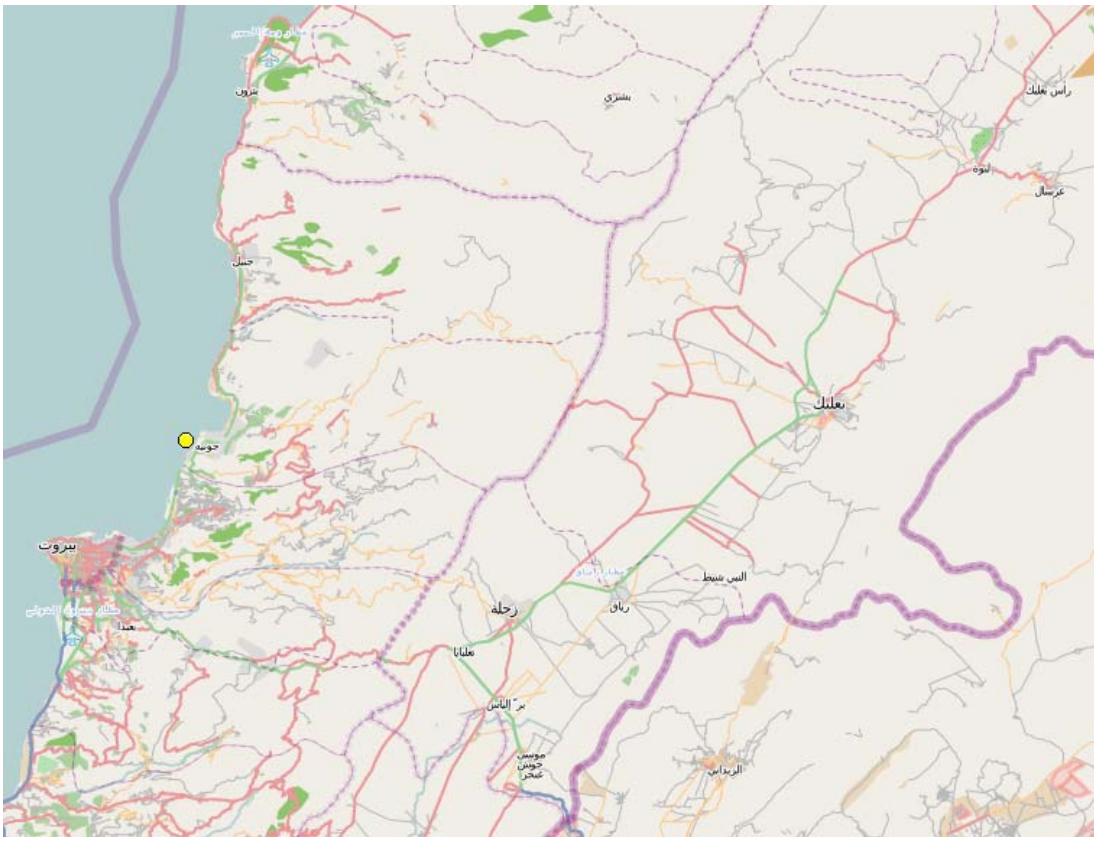
As illustrated in this webpage, a **JSON object** containing the geographic coordinates can be requested from the geocoding API. Although it is possible to get an XML version of the coordinates; for this task, a JSON version of the coordinates should be requested from the API. Once the **JSON-formatted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

Given that the first task will result in **multiple address values**, the process detailed above should be repeated for each of these addresses.

IV. Third task

At this stage, you are assumed to have completed the second task correctly. This is particularly true since the third task will make heavy use of the latitude and longitude information generated by the second task. Your task 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 second 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 cities with severe weather conditions on the map. In addition to the markers, you are required to add **image icons** capturing the reported weather condition. For example, an icon showing the sun should be placed next to a city experiencing extremely hot weather conditions, one with clouds and rain should be used to label a city suffering from an extremely cold rainy weather condition, and so on and so forth. Both the marker and the icon should be supplemented with a **summary list** presenting the **meteorological data** characterizing the weather condition of each one of the cities identified earlier. More specifically, for each city the summary list should report information relating to the temperature level, wind speed, humidity and rainfall levels.



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.

Finally, to complete the third task, add an event handler to your application to allow for a periodic **refreshing** of the extreme weather view. In this way, the map can be updated at regular time intervals with meteorological information pertaining to the newly spotted extreme-weather cities, i.e., the ones that were not identified by the last update of the map view. As you might expect, this final step requires that a **Timer** object be created and incorporated into your application.

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!