COM6516 Object oriented programming and software design

Programming assignment: A weather data viewer

Mark Stevenson

Task

Introduction

The task for this assignment is to design and build a viewer for weather data. The viewer should download weather records from airports. The data should be obtained from the Weather Underground (www.wunderground.com) server and presented to the user via a GUI.

Obtaining Weather Data

Your program should retrieve data from the Weather Underground data server. This data can be obtained using a URL constructed in the following way: http://www.wunderground.com/history/airport/AAAA/YYYY/MM/DD/DailyHistory.html?HideSpecis=1&format=1

Where AAAA is the four letter ICAO code for an airport or airfield, for example EGLL for London Heathrow Airport; see http://en.wikipedia.org/wiki/List_of_airports_by_ICAO_code:_E for a complete list of codes for Northern European airports. YYYY is the year (e.g. 2011), MM the month (e.g. 10), and DD the day (e.g. 01). Thus, the following URL retrieves data for London Heathrow (code EGLL) for 30 November 2010:
http://www.wunderground.com/history/airport/EGLL/2010/11/30/DailyHistory.html?HideSpecis=1&format=1

Handling Weather Data

Your program should be able to handle missing data as well as ambiguities in the individual readings (for example 'calm' is equivalent to a windspeed of 0). Your program should also be able to cope with the situation where no data are available at all (for example if the date requested is in the future or if data is missing) or if the data is obviously incorrect (for example visibility measurements of -9999.0 or precipitation of N/A).

Program GUI

When the program starts, a window should be displayed with drop down boxes for the user to select an airport from a list (you can choose how many locations are in the list but there should be at least two airports), and select a date using drop down boxes for the year, month and day. When the user has selected the date and location, then clicking on a button should retrieve the appropriate weather data.

The data that are retrieved should then be displayed in the following way:

- Temperature should be shown on a graph that indicates changes over the 24 hour period (see http://www.wunderground.com/history/airport/EGLL/2010/11/30/DailyHistory.html

for an example, but you only need to display temperature on these axes, not dew point or anything else), with the average temperature displayed as a number to one side of the graph.
- Atmospheric pressure should be displayed as a graph, with the pressure trend displayed as a number to one side of the graph,
- Wind data should be displayed as a graph, showing both average wind speed and gust speed.
- The total precipitation should be displayed as a number.

These components should be displayed in a new window, i.e. a window that is separate from the window used to select location and date.

Implementation

The main class in your solution should be called WeatherGUI.java. Your solution should run using Java version 1.7. Your code should compile on the command line by executing the command "javac WeatherGUI.java" and run by executing the command "java WeatherGUI".

Hints

- Start by downloading an example weather data from the Weather Underground server. Take a look at this file using a text editor such as jEdit, and make sure that you understand how it is structured. Note that in some cases measurements are missing or clearly incorrect. Your program should deal with data like this in a sensible way.

- When you design your solution you should consider separating the reading and storing of the weather data from displaying it to the user. One approach would be to read the data from the Weather Underground site and store it as a collection of objects. An ArrayList could be an appropriate way to manage these data. You can use the Sheffield package to read data from the file.

- The GUI includes graphs displaying the temperature, pressure and wind data. One possible approach would be to implement a general method that can display a suitable graph using the Graphics2D library which can be applied to any of the three types of data.

Deliverables: what to hand in

You should upload your Java source code (.java files only) to MOLE. Do not upload project files, packages or archives.

You should also upload a short (maximum 5 sides) report in PDF format that includes the following three parts:

1. A UML class diagram (or diagrams) showing the classes in your system and the associations between them. You should use the correct syntax for class diagrams, indicating the associations between classes and their multiplicities, as well as the most important class attributes and methods.
2. A brief (less than two sides) explanation of your design.
3. A brief (less than two sides) description of how you have tested the behaviour of each class in your program.

Deadline: when to hand in

You should upload your first stage work to MOLE by **11.00 on Monday 19 January 2015**. Don't leave your upload to the last minute.

Marking: how your submission will be assessed

A total of 60 marks are available for the assignment. These marks will be awarded as follows:

- Design (15 marks)
- Code functionality (15 marks)
- GUI (15 marks)
- Coding style (15 marks)

You should note that for coding style, readability is of great importance. You should therefore take great care with line breaks, document your code appropriately and use indentation consistently. Before you hand in your code, ask yourself whether the code is understandable by someone marking it.

Your GUI should display the required data clearly. There should be one window that allows the user to select location and date information, and another window that displays data as specified in the requirements.

Unfair means

When you submit your work, you are required to acknowledge that the material you submit (including your code) is your own work. ***This is an individual assignment, and so you are expected to work on your own***. You may wish to discuss your design with other students, but ***you must not work together to develop your solution***. All the reports and code that is submitted will be examined using specialised software that will identify if there is evidence of using code written by another student, or downloaded from online resources.

Mark Stevenson October 2014