Sorting Data + Your Own Extension!

This assignment completes the project for this course. It builds on the assignments you did in the previous modules. As usual, you will not submit your code for grading, but you should do it because you'll learn and it's fun! This time our starter code is simply our solutions to module 5 (minus the optional content) so if you would like to simply copy your module 5 code over into the module6 files that we provide, please feel free. As before, we still we recommend you copy in individual methods just to be safe, rather than replacing whole files.

As before, you'll want the UnfoldingMaps javadoc and the Processing reference page open. **IF YOU ARE WORKING OFFLINE**, make sure you've got this downloaded.

Learning outcomes

- Use sorting to answer questions about data sets
- Implement the Comparable interface
- Complete a medium-sized project, going beyond defined requirements

Introduction and What You Will Do...

In the final piece of your project, you will organize the earthquake data and compute statistics on it. You will then extend your project and share your work with other learners in this class.

To complete this project, follow these steps:

1. Find and open the starter code: You will add functionality to the classes you developed in the previous part of the project: EarthquakeCityMap.java , EarthquakeMarker.java , CityMarker.java , CommonMarker.java, LandQuakeMarker.java , OceanQuakeMarker.java. You will find these files (with our implementations) in the module6 package. You will also find three new files: AirportMap.java, AirportMarker.java and LifeExpectancy.java. These are files you might use in the extension (step 5, below). **IF YOU ARE WORKING OFFLINE**, you should once again make sure the variable offline is set to true.

2. Compare our implementation to yours. Look at the code and notice any differences between the way you implemented the functionality in module 5 and the way we did. Find at least one difference between your code and ours (could be helper method organization, variable names, the way we implemented a method, etc) and be ready to reflect on it for the quiz.

3. Implement the Comparable interface in EarthquakeMarker:

- add the keywords “implements Comparable<EarthquakeMarker>” to the class definition. After you change the header, but before you do step b, **make a note of what error(s) occur in your code, where they occurred and why**. We’ll ask you about this on the quiz.
- implement the compareTo(EarthquakeMarker marker) method in the EarthquakeMarker class so that it sorts earthquakes in reverse order of magnitude.

4. Add and Implement the private method void sortAndPrint(int numToPrint) in EarthquakeCityMap. This method will create a new array from the list of earthquake markers (**hint**: there is a method in the List interface named toArray() which returns the elements in the List as an array of Objects). Then it will sort the array of earthquake markers in reverse order of their magnitude (highest to lowest) and then print out the top numToPrint earthquakes. If numToPrint is larger than the number of markers in quakeMarkers, it should print out all of the earthquakes and stop, but it should not crash.Call this method from setUp() to test it. An example input and output files are provided in the data folder: use test2.atom as the input file, and sortandPrint.test2.out.txt is the expected output for a couple different calls to sortAndPrint.

5. Add your own extension! This is the fun part and the main part of this final programming assignment. Add any extension you choose to the earthquake program as it stands. You will submit this extension for peer grading, as described in the Peer Review Assignment at the end of this module, following the Programming Assignment Quiz.

The only conditions for this piece is that your extension be (1) something you are interested in doing and (2) something that was not required in any of the previous programming assignments in this course. Also, you will be asked to submit a Peer Review assignment talking about your extension, so it would be a good idea to read over the requirements for that assignment before you get started programming. (It's the last assignment in this module).

We encourage you to get creative, but if you need a little inspiration, here are some ideas you are welcome to use along two different lines: direct extensions to the earthquake application, and working with other data sets. Again, these are just ideas. You are not required to use any of them.

Ideas for extending the functionality of the earthquake application:

- When clicking on a city, display a popup menu either on or off the map (off will be easier) which displays a count for the number of nearby earthquakes (within threatCircle) the average magnitude, and most recent earthquake. Get creative with the data you want to display.
- Style your earthquake or city markers to make them more aesthetically pleasing. You could use an image for the marker, or play around with PGraphics and PApplet drawing methods to make them how you want.
- Ensure that the popup info boxes about the earthquakes and cities are always drawn on top of the other graphical information (markers). There are some links that will be helpful with this in the module 5 programming assignment.
- Support clicking on markers which are still displayed while another is clicked rather than always resetting on the second click.
- Use keyboard events to change which earthquakes are displayed by age or only display cities which are above a certain latitude. etc.

Ideas for incorporating new data sources:

These extensions will be a little more difficult than just working within the earthquake application because you will have to work with new data and understand how it is represented. Depending on what you want to do, you might even write your own data parsing functions, or modify ours. However, even though these extensions are challenging, we think they will be a lot of fun.

New data source idea 1: Work with the airport and route data: There are two data files in the data directory: airports.dat and routes.dat which contain information about all of the airports in the world and routes between these airports, respectively. There are also two methods in ParseFeed.java for loading the data in these files, and we provide a starter file AirportMap.java that shows how to use the parser to read in the data from these files. Once you've read in the data, what you do with it is up to you. You might:

- Display all the airports in the world as features, and then display additional information about them when the user hovers over them.
- Display only a subset of the airports in the world, based on some criteria.
- Display airports, and when the user clicks on one, display the routes out of that airport.
- Display all the airports that you have been to.
- … the list of possibilities is endless!

New data source idea 2: Work with data from the World Bank: The World Bank maintains a large number of datasets with some really interesting international data. You saw Mia work with one such dataset: life expectancy. We have included this data in your eclipse package as well as a method in ParseFeed to load its data. We also have a class called LifeExpectancy.java that implements the example Mia showed in module 3. You cannot use this as your extension, but you can modify it to do something different, and this would be a fine extension.

If you want to use other data from the World Bank, that's great too! You can find other data sources here: http://data.worldbank.org/ You will probably need to add a method to the ParseFeed class to read in that data you choose. You can do this following the examples that are already there.

Finishing up

Once you've completed your extension, you are done with this programming assignment and almost done with this course! Go on to take the quiz, write up and submit your peer review assignment, and do some peer reviews, and that's it. Congratulations!