Mapping Earthquakes From USGS GeoRSS Feed Using Java

Spring Semester 19, GEO 876, Project Report, Ursina Christina Boos, 15-725-708. E-Mail: ursinachristina.boos@uzh.ch

Introduction

Around 12.000 – 14.000 earthquakes are recorded worldwide each year, whereas earthquakes with lower magnitudes (*M*) are located more often. For example, tremors with M 2 or smaller are recorded several hundred times a day. Earthquakes with a more destroying character (M 7 or higher) occur more than once per month and devastating earthquakes (> M 8) occur once a year (IRIS, 2011).

This paper describes a possible solution to map all the earthquakes registered worldwide using an object-oriented approach. The quakes are mapped using Java, making use of the two libraries *Unfolding* and *Processing*. These libraries enable parsing a GeoRSS feed and mapping the tremors with relatively little code (Baer, Das & Fu, 2019). For this work, the USGS' earthquake feed has been (USGS, 2019).

Results

My solution consists of three Classes, namely the `Main`, `Map` and `Parsing` class. All three classes are subclasses of the superclass `Object`, which is superclass to all classes implemented in Java. The connection between these classes are illustrated in Figure 1.

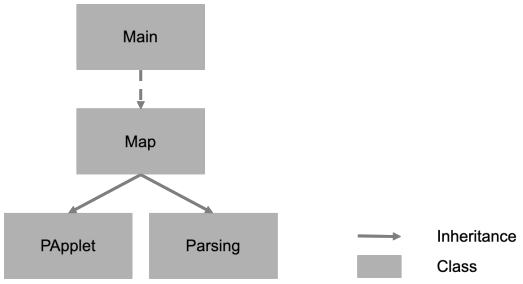

Figure 1: Diagram of the classes (grey boxes) used in the earthquake mapping programme including connections between classes (arrows). The Main, Map and Parsing classes are implemented in this solution, the PApplet class is imported from the Processing library.

The `Parsing` class processes the live GeoRSS feed, which is in an XML format, such that the `Map` class can make use of these data. This class consists of four methods, `analyseEarthquake()`, `getLocation()`, `getStringValue()` and `getFloatValue()`.

The first method goes through the XML file and saves all earthquake events with a location to an array list. In a next step, the method infers the event's properties (title, magnitude and depth) from the title given in the XML file. Finally, the features saved in the array list are returned.

The second method, `getLocation()`, gets and returns the location saved in the `<georss:point>` tag in the XML file. Since the location is saved in the format (latitude longitude), the string needs to be split and saved into two new variables, one each for latitude and longitude, respectively.

The methods `getStringValue()` and `getFloatValue()` are helper methods. The method `analyseEarthquake()` makes use of these two methods when accessing the title string and the value of the event's depth. The `getStringValue()` method returns the content accessed in the XML file in a string format, the `getFloatValue()` method parses and returns the casted float value of the current accessed XML item.

The second class, `Map`, created for this project maps all the processed earthquake features. This class is a subclass of the `PApplet` class, which can be found in the *Processing* library. This second class consists of the methods described as follows.

The `getEarthquakeMarker()` method returns a marker for each earthquake location. This returned marker is then used in the `getPropertyMarker()` method, which sets a colour and a radius for each event according to the earthquake's magnitude and depth.

The radiuses for the markers depend on the depth of the earthquakes and is calculated in the `getRadius()` method. The radius is calculated with respect to the reciprocal logarithm of the depth. The result is multiplied by fifty such that the sizes of the markers are large enough to see on the map.
The earthquakes' magnitudes are classified according to the Richter magnitude scale and hence, to the extent of damage caused by the earthquake (IRIS, 2011). A gradient colour palette has been chosen such that the lighter the colour of the marker, the smaller the magnitude. The colours are in RGB colour model and are initialised at the beginning of the code. These colours are then assigned as marker fill and stroke in the methods `setColourMarkerX()` and `setStrokeColourX()`. These two methods are available for each colour used. The markers with the assigned properties are matched to the corresponding point feature and are then saved in an array list, initialised in the `getEarthquakeMarkersList()` method.
The *Map* class also includes methods to count the number of currently displayed earthquakes to include this information in the map's legend. Moreover, the methods `getDepth()` and `getMagnitude()` are implemented in this class. Both methods iterate through all markers saved in the `getEarthquakeMarkersList()` method. The locations of these markers are then compared to the locations of the point features in the list of all parsed point features, initialised at the beginning of the code. If the location matches, the marker is inside the map and the position of the mouse matches, the magnitude and depth are accessed of the relevant feature. These methods are then used in the `popupMenu()` method, which is based on the *Processing* library and displays a box with some text about the feature's properties. However, this popup is only displayed if the mouse is pressed.
The `setup()` method includes all the initial setups and is called only once when the programme is started. Initial setups include the size of the applet window, the basemap, the markers and the zoom level. On the other hand, the `draw()` method draws the map and other elements such as the markers and legend and runs continuously. Furthermore, the above mentioned behaviour for displaying a popup is implemented in this method.
Both the popup's and legend's properties are set in the methods `popupMenu()` and `addLegend()`. The size of the boxes, the fill colour and all features displayed (e.g. points to represent the markers) in the legend and popup are defined in these methods.
The `Main` class is necessary for all Java programmes, such that Java knows where to start executing. It consists only of the `main()` method, where the executable statement to call the `PApplet` and `Map` are contained.

Concluding Discussion

All in all, the programme runs without compiling errors. However, there are a few details to be improved in the future. First, the basemap has a black gap in the pacific ocean. It would be desirable to have a continuous basemap without such a gap. Furthermore, the popups are not only displayed when clicking on a marker but also when clicking elsewhere on the map. This should be improved in future work. Lastly, the `Map` class includes many methods concerning the colouring of the markers. It would be appropriate to encapsulate these methods in a separate colour class. Also, the earthquake's properties could be saved in another class, for example an Earthquake class to abstract the solution even more.
Though there are a few functionalities to improve, the output mostly works as expected. The solution is object oriented because different functionalities are encapsulated in different classes. The map does not only display the earthquakes' locations but also has some additional information provided by advanced interaction methods.

References

IRIS (2011): How Often Do Earthquakes Occur?. Education and Outreach Series, 3,Washington DC, USA.

Baer, M., Das, R. & Fu, C.: Project: Earthquake mapping, lecture notes distributed in GEO 876 at University of Zurich UZH, on 04. March 2019.

USGS (2019): Earthquakes feed. Available from: https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_day.atom [16.03.2019].