Responding to User Actions

This assignment builds on the assignment 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! We provide starter code which implements the functionality of the program through module 4. If you would prefer to build on your own code you are welcome to copy over your code from module 4, **but please be careful to notice the changes.** We recommend you copy in individual methods. **Do not overwrite whole files.**

Learning outcomes

- Explain execution flow in an event-driven program
- Build event handlers to respond to graphical user input
- Perform code refactoring to add functionality to code while maintaining good design

Introduction

Your map is about to become interactive! You are going to add functionality to your map so that additional information is displayed when the user hovers over or clicks on any marker with her/ his mouse. When she hovers over a city marker, your map will display a box with the city's name, country, and population. When she hovers over an earthquake marker, your map will display the title of the earthquake (including its magnitude and region). Clicking on a marker gives even more information: A click on a city marker will lead to only that city and earthquakes which affect it being displayed on the map. Clicking once again on that marker will bring the rest of the map's markers back. Similarly, after clicking on an earthquake marker, only cities potentially affected by that earthquake will be displayed. You'll use event-driven programming to make this happen.

To accomplish this you will need to override two methods, mouseClicked() and mouseMoved(). In both of the these methods you will need to make use of the isInside() method of the SimplePointMarker class, as well as the PApplet fields, mouseX and mouseY.

Resources to have open

We expect that you are getting good at working with the documentation by now, so you'll again need the two documentation pages handy:

- http://unfoldingmaps.org/javadoc/
- https://www.processing.org/reference/

IF YOU ARE WORKING OFFLINE, you can download these documentation pages when you have an internet connection.

What you will do

1. Find and open the starter code: The starter code for this part of the project is in the module5 package. You will find the same classes you worked with for the previous part of the project: EarthquakeCityMap.java, EarthquakeMarker.java, CityMarker.java, LandQuakeMarker.java, and

OceanQuakeMarker.java. You will also find a new class: CommonMarker.java. The starter code is similar to what you implemented for the last part of the project, but not quite… see the next steps.

IF YOU ARE WORKING OFFLINE: Remember to set the value of the variable offline to true.

2. Modify the class header in CityMarker so that this class extends CommonMarker instead of SimplePointMarker. It will cause an error. That's OK.

3. Draw the UML diagram for the classes:

- SimplePointMarker
- CommonMarker
- EarthquakeMarker
- CityMarker
- LandQuakeMarker
- OceanQuakeMarker

Notice the difference between the class hierarchy now and how it looked in module 4's assignment.

4. Read the compile error in the starter code for the class CityMarker. Explain what caused it, then fix it by changing the name of one of the methods in class. (Hint: which method that we "promised" to define is missing? Look at how draw() is implemented in CommonMarker). Make a note of this because we'll ask you about it on the self-assessment quiz.

Steps 3 and 4 highlighted the fact that we have changed the class hierarchy structure for this part of the project: we have introduced the class CommonMarker as the parent class of both EarthquakeMarker and CityMarker. Now CommonMarker is the class that overrides the draw() method, and it calls drawMarker() which each subclass will implement. We introduced CommonMarker because in this assignment there will be some drawing functionality that is common to all markers on our map, so we didn't want to have to duplicate code between EarthquakeMarker and CityMarker. This process of restructuring our code is known as refactoring and it is very common in software engineering.

5. Implement the selectIfHover helper method in EarthquakeCityMap. This method is called in mouseMoved() method in EarthquakeCityMap, and mouseMoved() is called by the event handler when the user moves the mouse. selectIfHover should set the instance variable selected for the first Marker it finds that mouseX and mouseY is inside of.

Hints for step 5

- You can use the isInside(UnfoldingMap m, float x, float y) method defined in the AbstractMarker abstract class to check if a location is inside a marker.
- PApplet Instance variables mouseX and mouseY store the values of the current mouse position.
- Note that we clear lastSelected in mouseMoved() before we call selectIfHover, so you can set that variable to indicate that you have found a selected marker. No other markers should then be selected.
- The variable selected is a private instance variable in the marker classes. You will use the getter method isSelected() and setter method setSelected().

6. Implement the showTitle(PGraphics pg, float x, float y) in both EarthquakeMarker and CityMarker so that it displays the relevant information about the city or earthquake. For earthquake markers display the title of earthquake. For a city, display its name, country, and population. Notice how showTitle is used in the CommonMarker's draw method. (We'll ask you about this in the quiz)

Then use this implementation to test your selectIfHover method. You should see information about markers appear when you hover over them!

Hints for step 6:

- Don't worry if your labels get overlapped by city and earthquake markers. There's no easy way to insist that they always stay on top. But if you want to force the labels to always be on top, here are a couple of links with information to explore:

  http://processingjs.nihongoresources.com/framework/framework.php

  http://stackoverflow.com/questions/23804651/understanding-void-draw-in-processing
- You might want to draw a rectangle underneath the text so the text is more visible, but you don't have to.

7. Implement the mouseClicked() method in EarthquakeCityMap. This method tests whether the lastClick variable refers to a null object or not; if lastClick is not null, this click "de-selects" whichever marker was clicked on last and so all markers should be displayed. Otherwise, the method must determine which marker is being selected (if any) and hide / display other markers so that:

When an earthquake's marker is selected, all cities within the threat circle of this earthquake are displayed on the map and all other cities and earthquakes are hidden. You are given an implementation of the threatCircle() method in the EarthquakeMarker class.

When a city's marker is selected, all earthquakes which contain that city in their threat circle are displayed on the map and all other cities and earthquakes are hidden. You are given an implementation of the threatCircle() method in the EarthquakeMarker class.

Hints and information for step 7:

- You will use the setHidden() and getHidden() methods to access the private "hidden" flag. You may also use the provided private helper method unhideMarkers().
- From the USGS website, http://earthquake.usgs.gov/research/dyfi/#pda , there is a model for predicted intensity levels of earthquakes at some distance away from their epicenter, as a function of the magnitude of the earthquake. For simplicity, you will approximate the distance away at which the earthquake is felt at an intensity of more than II by the formula: 20*(1.8**(2*Magnitude - 5)). DISCLAIMER: this formula is for illustration purposes only and is not intended to be used for safety-critical or predictive applications. The actual distance at which earthquakes are felt with some intensity depends on the geography of the region and many other factors as well. This method is included in the EarthquakeMarker class.

8. (Optional) Add to the implementation of the drawEarthquake() method in the OceanQuakeMarkerclass so that if this marker has been clicked on, lines are drawn between this marker and all cities within its threat circle. If this marker has not been clicked, these lines should disappear.

Hint for step 8:

- You can find the x and y coordinates of a marker on a canvas by using the class ScreenPosition.
- To make lines disappear, you can re-draw them as transparent, e.g. using the noStroke() command.

Congratulations! You have finished making your map interactive. It's time to take the quiz!