**EARTHQUAKE MAP -DATA VISUALIZATION**

Report submittedin partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Computer science & Engineering

By

**SONALI SINGH**

(Enrollment number-07815002715)

To



**MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY**

Affiliated to Guru Gobind Singh Indraprastha University

Janakpuri, New Delhi-58

August ,2017

**CERTIFICATE**

I, Sonali Singh, student of maharaja surajmal institute of technology, enrolment number-07815002715, declare that project report “**Earthquake Map -Data Visualization”** is my original work.

Dated: 2017. Sign of Candidate

**ACKNOWLEDGEMENT**

I extend my sincerest gratitude to my respected teacher Mr. Sumeet Malik under whose supervision this project has been undertaken. He is the guiding force and prime inspiration to lift me from the initialization state to the successful completion of the project. His friendly guidance and discussions over the complexities of a real-time project have invoked a deep thought in me.

I would also like to thank my college management and faculty for providing this opportunity of making this project and learning from it.

Last but not the least, undertaking a project such as this, places an equal if not greater pressure on friends and family members closest to me. Hence, I express my gratitude to my parents for being a constant support throughout this project and helping me in every possible way they could. Thanks to all those who helped me in one-way or other, and to all, whose names go unmentioned.

**CONTENT**

|  |  |
| --- | --- |
| TOPICS | PAGE NUMBER |
| ABSTRACT | 6 |
| CHAPTER 1: INTRODUCTION |  |
| 1.1 Earthquake | 7 |
| 1.2 Need | 8 |
| 1.3 Overview | 9 |
| CHAPTER 2: DATA VISUALIZATION |  |
| 2.1 Introduction | 11 |
| 2.2 Importance | 11 |
| 2.3 Advantages | 11 |
| CHAPTER 3: TECHNOLOGIES USED |  |
| 3.1 unfolding maps library | 13 |
| 3.2 Processing Library | 13 |
| 3.3 JAVA | 14 |
| 3.4Algorithm | 15 |
| 3.5Data Structurs | 16 |
| CHAPTER 4: PApplet |  |
| 4.1 Introduction to PApplet | 17 |
| 4.2 Difference between Applet and Application | 17 |
| 4.3 How is an applet executed | 17 |
| 4.4 Advantages | 18 |
| 4.5 Life Cycle of PApplet | 18 |
| CHAPTER 5: DESCRIPTION |  |
| 5.1 Setting up the map | 20 |
| 5.2Earthquake data from USGS API | 21 |
| 5.3 Class hierarchy | 22 |
| 5.4 Key | 23 |
| CHAPTER 6: WORKING |  |
| 6.1Zoom and pan | 24 |
| 6.2 Hover | 26 |
| 6.3Click | 28 |
| 6.4 Offline map | 31 |
| CHAPTER-7: CLASS HIERARCHY | 32 |
| CONCLUSION | 35 |
| BIBLIOGRAPHY | 36 |
| APPENDIX-1 | 37 |

**ABSTRACT**

This project entitled ‘EARTHQUAKE MAP-DATA VISUALIZATION’, as the name suggests visualizes earthquake data on map which helps in easy understanding of the problem at hand as it helps to identify a pattern, know the problem areas and decide upon an action plan.

The main objective of this project is to resolve the issue of visualizing earthquake data obtained from the USGS API on a map provided by google, Microsoft or any other map provider to study the previous earthquake data, identify a pattern and come up with optimal solutions to predict or identify the earthquake prone zones, take preventive measures on time and reduce the impact of destruction caused by earthquakes.

Chapter-1 gives brief information about earthquakes, their causes, effects, preventive measures. It also gives the reason for this projects existence i.e. the need for it and an overview of the project. Chapter-2 explains the core concept of this project-data visualization, its importance as well as it’s advantages. Chapter-3 describes in brief the technologies used in this project which include unfolding map, processing library, java, algorithm and data structure. Chapter-4 is an extension of previous chapter and explains PApplet, which is a component of processing library, in detail along with it’s life cycle. Chapter-5 provides description of the project in terms of setting the map, key, usgs data etc. Chapter-6 deals with the working functionality of the project like zooming, hover, click and offline map. Chapter-7 deals with class hierarchy of the project along with UML diagram.

**CHAPTER-1**

**INTRODUCTION**

* 1. EARTHQUAKE:

An **earthquake** (or quakes, tremors) is shaking of the surface of earth, caused by sudden movement in the Earth's crust. They can be extremely violent.

Earthquakes are usually quite brief, but may repeat. They are the result of a sudden release of energy in the Earth's crust. This creates seismic waves, which are waves of energy that travel through the Earth. The study of earthquakes is called seismology. Seismology studies the frequency, type and size of earthquakes over a period of time.

There are large earthquakes and small earthquakes. Large earthquakes can take down buildings and cause death and injury. Earthquakes are measured using observations from seismometers. The magnitude of an earthquake, and the intensity of shaking, is usually reported on the Richter scale. On the scale, 3 or less is scarcely noticeable, and magnitude 7 (or more) causes damage over a wide area.

An earthquake under the ocean can cause a tsunami. This can cause just as much death and destruction as the earthquake itself. Landslides can happen, too. Earthquakes are part of the Earth's rock cycle.

CAUSES:

* Movement of Tectonic plates
* Volcanic activity
* Underground explosion

EFFECTS:

* Ground shaking and rupture
* Tsunamis
* Landslides
* Liquefaction
* Fires
* Damage to man-made structures
* Injuries and Death

PREVENTIVE MEASURES:

##### Personal measures:

* Seek shelter under stable tables or under door frames.
* If outside, stay away from buildings, bridges and electricity pylons and move to open areas.
* Avoid areas at risk from secondary processes, such as landslides, rockfall and soil liquefaction.
* After an earthquake, check gas, water and electricity pipes and lines for damage.
* Listen to the radio and follow the instructions issued by the authorities.

##### Technical/biological measures:

* Earthquake-proof planning and design of buildings
* The microzoning of the local geological substratum provides indicators of areas in which tremors will have a particularly strong or attenuated effect.

##### Organisational measures:

* At present, earthquake prediction is insufficiently precise to provide the public with sufficient advance warning. For this reason, adequate preparedness and assistance in catastrophes is extremely important in areas affected by earthquakes. Measures of this nature enable numbers of human lives to be saved.
  1. NEED:

Every year lot of destruction is caused by earthquakes, be it in terms of death toll or damage to infrastructure or other natural calamities like landslides, Tsunamis etc. thereby leading to havoc in society and in the lives of those affected.

Therefore it becomes extremely essential that we be able to identify the earthquake prone zones based on the data of previous earthquakes and to take the preventive measures in time to reduce the impact of destruction on our lives.

At present, earthquake prediction is insufficiently precise to provide the public with sufficient advance warning. For this reason, adequate preparedness and assistance in catastrophes is extremely important in areas affected by earthquakes. Measures of this nature enable numbers of human lives to be saved.

An efficient technique to study the data of previous earthquakes and obtain important conclusion is to visualize the data of previous earthquake on a map.

Data visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software

To meet this requirement of studying previous earthquakes, visualizing data on a map to obtain insightful conclusion and predict a pattern to the occurrence of earthquake, this project came into being.

* 1. OVERVIEW:

In this project, EARTHQUAKE MAP- Data Visualization, earthquake data from USGS API is parsed and the data obtained is plotted on a map. Map is setup by the mapProviders available and supported in Unfolding Maps Library. Various Map providers are available such as GoogleMapProvider, MicrosoftProvider, OpenWeatherProvider etc.

The Unfolding map constructor takes a PApplet as parameter. PApplet is provided by the processing library which includes the setup() method that is executed only once and draw() method which draws the map as many number of times as it is executed. Unfolding map also takes map provider as a parameter and four other parameters which helps to specify the size of the map to be displayed.

Unfolding maps also provide the functionality of zooming and panning through createdefaulteventdispatcher().the map can be zoomed in and out by the ’+’ and ‘-‘ buttons on keyboard and also by scrolling the roller present on mouse. Double clicking on a certain location which centers and zooms the map to that location.

Arrow keys can be used to move across the map. Up, Down, Left, Right arrows on keyboard are used to up, down, left and right on the map.

For offline working, a map tile is provided and a sample earthquake data is used. The zooming is also limited to two levels.

This project is built on extensive use of object oriented concepts of JAVA such as Generics, Inheritance, Polymorphism, Function Overriding, Function Overloading, Interfaces and use of data structures and algorithms.

All major cities around the world are marked on the map by a triangle of maroon colour. The earthquakes that occur on land are marked by sphere while those that occur in ocean are marked by square. The dimension of the sphere and square markings vary with the magnitude of earthquake it represents i.e. the earthquake of greater magnitude is represented by symbol larger in size than the symbol which represents an earthquake of smaller magnitude.

Depending on the depth of earthquake, the marked symbols are assigned different colours. If the earthquake occurred at shallow depth(<70 km), it is marked with yellow colour. If it occurred at intermediate depth(between 70 and 300 km), it is marked with blue colour and if it occurred deep(>300kms), it is marked with red colour.

Along with the map, a key is also displayed to the right of the map depicting the above specifications of earthquake markers. The size of the Appletviewer is set by using size() method in the setup() method.

The project is designed to provide the following functionality:

* When we hover over a city marker, a rectangular white box is visible adjacent to the marker displaying details relate to the city such as city name, country name and population of the city.
* When we hover over an earthquake marker, a rectangular white box is visible adjacent to the earthquake marker displaying the title of that earthquake which includes the magnitude of that earthquake and its location.
* When we click on any earthquake marker whether it is on land or ocean, only that earthquake marker and the city marker within the threat circle of that earthquake are displayed. All other markers are hidden.
* When we click on a city marker, only that city marker and the marker of earthquake that has occurred there are visible.
* All the hidden markers are visible again by clicking anywhere on the map.

Before describing the project in detail and explaining its in-depth working , it is important to take a look at the technologies this project is based on like JAVA, Unfolding Map, Processing library and PApplet as well as understand what is meant by data visualization.

**CHAPTER-2**

**DATA VISUALIZATION**

* 1. INTRODUCTION:

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, we can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data we see and how it’s processed.

### IMPORTANCE:

Because of the way the human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than poring over spreadsheets or reports. Data visualization is a quick, easy way to convey concepts in a universal manner – and you can experiment with different scenarios by making slight adjustments.

Data visualization can also:

* Identify areas that need attention or improvement.
* Clarify which influencing factors.
* Help you understand problem at hand efficiently.
* Predict and enable us to take necessary future action.
  1. ADVANTAGES:

### **Comprehend information quickly**

By using graphical representations of information, it is easier to see large amounts of data in clear, cohesive ways – and draw conclusions from that information. And since it’s significantly faster to analyse information in graphical format (as opposed to analysing information in spreadsheets), problems can be addressed in a more timely manner.

### **Identify patterns**

Even extensive amounts of complicated data start to make sense when presented graphically. Highly correlate data can be recognized to identify a pattern

### **Communicate the story to others**

Using charts, graphs or other visually impactful representations of data is important in this step because it’s engaging and gets the message across

**CHAPTER-3**

**TECHNOLOGIES USED**

* 1. UNFOLDING MAPS LIBRARY:

Unfolding is a library to create interactive maps and geo visualizations in Processing and Java.

FEATURES:

* Interaction Events:

Unfolding enables you to quickly create interactive maps. Basic interactions such as Zoom & Pan are included. More advanced functionality such as Overview + Detail, or multitouch gestures can be easily added.

* Data Visualization:

Simply create geo-positioned markers to display data on a map. The visual style can be adapted freely. The library supports loading and displaying user-defined shapes, such as points, lines, or polygons.

### Styled Maps:

Unfolding is a tile-based map library. Map tiles can have various geographic features, and come in all kind of styles. It comes with various map providers, such as OpenStreetMap or TileMill.

### Clean & Extendable Code:

Unfolding enables beginners to easily create simple maps. Advanced users can quickly sketch out prototypes, or create sophisticated visualizations. And expert users can extend Unfolding's functionality.

* 1. PROCESSING LIBRARY
* Initially created to serve as a software sketchbook and to teach programming fundamentals within a visual context, Processing has also evolved into a development tool for professionals. The Processing software is free and open source, and runs on the Mac, Windows, and GNU/Linux platforms.
* The Processing community has written more than a hundred libraries to facilitate computer vision, data visualization, music composition, networking, 3D file exporting, and programming electronics.
* Processing is geared toward creating visual, interactive media, so the first programs start with drawing.
* Software prototyping and data visualization are two of the most important areas for Processing developers. Research labs inside technology companies like Google and Intel have used Processing for prototyping new interfaces and services. Companies including General Electric, Nokia, and Yahoo! have used Processing to visualize their internal data.
* In this project, PApplet is used which is described in detail in the next chapter.
  1. JAVA:

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform.

FEATURES:

* **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
* **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
* **Distributed** − Java is designed for the distributed environment of the internet.
* **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.
  1. ALGORITHM:

An algorithm (pronounced AL-go-rith-um) is a procedure or formula for solving a problem, based on conducting a sequence of specified actions. A computer [program](http://searchsoftwarequality.techtarget.com/definition/program) can be viewed as an elaborate algorithm. In mathematics and computer science, an algorithm usually means a small procedure that solves a recurrent problem.

In programming, algorithm is the set of well-defined instruction in sequence to solve a program. An algorithm should always have a clear stopping point.

Qualities of a good algorithm:

1. Inputs and outputs should be defined precisely.
2. Each step in algorithm should be clear and unambiguous.
3. Algorithm should be most effective among many different ways to solve a problem.
4. An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages.
   1. DATA STRUCTURES:

Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage.

In simple language, Data Structures are structures programmed to store ordered data, so that various operations can be performed on it easily. It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.

Some example of Data Structure are: Linked List, Arrays, Tree, Graph, Stack, Queue etc.

All these data structures allow us to perform different operations on data.

**CHAPTER-4**

**PApplet**

* 1. Introduction to PApplet:

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side. After a user receives an applet, the applet can produce a graphical user interface. It has limited access to resources so that it can run complex computations without introducing the risk of viruses or breaching data integrity.

An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application. JVM creates an instance of the applet class and invokes **init()** method to initialize an Applet.

A Processing window is a special type of Java program called a PApplet**.** It isBase class for all sketches that use processing.core . As of Processing 3.0, we have removed Applet as the base class for PApplet.

* 1. Difference between Applets and Application in JAVA:

 The basic differences between Java Applications and Java Applets.

Java**applications**must have at least one public class that contains a **public static void main(String [])** method.  The ***main*** method is where the application's execution begins.  If there is more than one public class which contain a main method, then you will also need to specify which class is the application that you wish to run.  The application runs until the end of the main method is reached (or the program crashes).

**Java applets are a little different.**  Applets do not require a main method. If they do not have a main method they can not be run as a stand-alone application. If an applet does have a main method, then it can be run as an application with execution beginning at the main method as described earlier.

## How is an applet executed?

In Eclipse, the process to build and run an applet is very similar to building and running a Java application.  Just click the run button and make sure that it is trying to run your project ***as an applet*** and not as an application.

Applet's (without main methods) are actually run by other programs.  For instance an applet viewer can be launched and the applet started by the applet viewer.

There are two ways to run an applet:

* Executing the Applet within Java-compatible web browser.
* Using an Applet viewer, such as the standard tool, applet viewer. An applet viewer executes your applet in a window
  1. Advantages:
* It works at client side so less response time.
* Secured.
* It can be executed by browsers running under many platforms, including Linux, Windows, Mac OS etc.
  1. Life cycle of a PApplet:

All Java applets must be initialized and started. Once the applet is started, it runs until it is stopped by the user.  In the case of a PApplet, the developer (you) only needs to define the draw() and setup() methods.  Processing handles the rest.  This diagram shows the order that Java's Applet methods are called by PApplet.

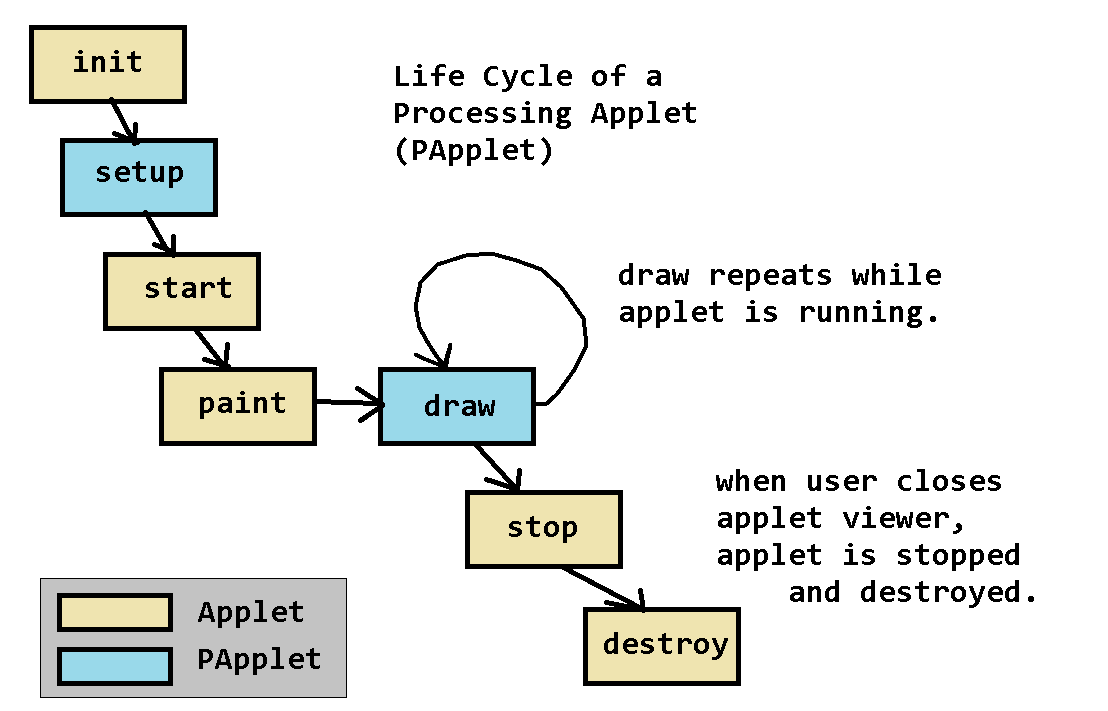
[](http://pages.cs.wisc.edu/~cs302/labs/EclipseTutorial_Processing/images/processing_applet_life_cycle.png)

Fig 4a: life cycle of PApplet

PApplet execute the setup() method and then repeatedly call the draw() method to continuously update the window.  If your draw() method does not change the position, color, size, etc of any of the items displayed, it will appear as if nothing is happening.  To create movement (animation), you must move your objects on each call to the draw() method. The degree of that objects are changed in each new frame will determine how "fast" the objects appear to be moving or otherwise changing.

Call the **frameRate(int)** method to see your objects move slower and faster.

**CHAPTER-5**

**DESCRIPTION**

* 1. SETTING UP THE MAP:

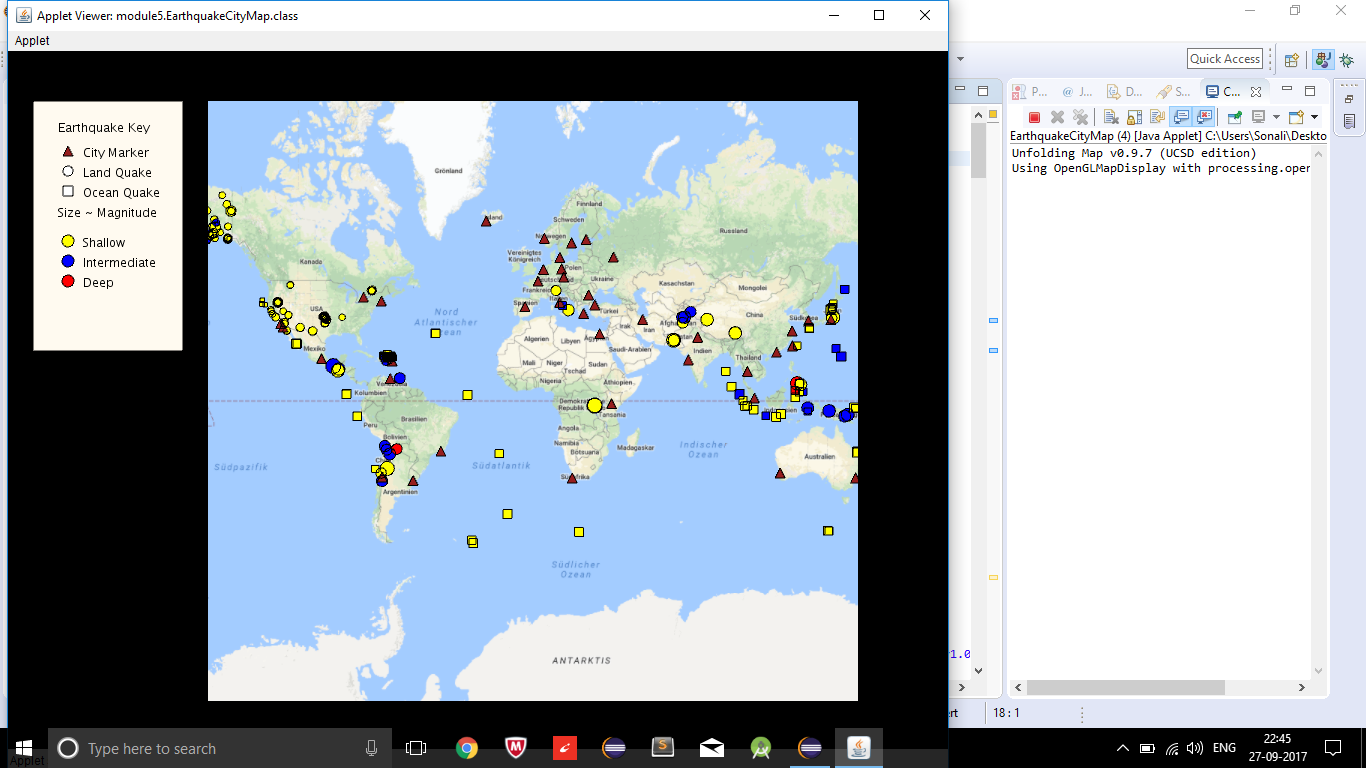


Fig 5a: map setup

Map is setup by the various map Providers available and supported in Unfolding Maps Library. Various Map providers are available such as GoogleMapProvider, MicrosoftProvider, OpenWeatherProvider etc. The complete list of map providers is given in appendix-1.

In the above figure, GoogleMapProvider is used.

Unfolding map takes map provider as a parameter and four other parameters (x-coordinate and y-coordinate of starting point of map and width and height of map respectively) which helps to specify the size of the map to be displayed.

* 1. EARTHQUAKE DATA FROM USGS API:

In this project, EARTHQUAKE MAP- Data Visualization, earthquake data from USGS API is parsed and the data obtained is plotted on a map.

**The USGS API provides the earthquake data so that it can be used by scientists and researchers to provide and apply relevant earthquake science information and knowledge for reducing deaths, injuries, and property damage from earthquakes through understanding of their characteristics and effects and by providing the information and knowledge needed to mitigate these losses.**

The goals of the USGS' Earthquake Hazards Program are:

* Improve earthquake hazard identification and risk assessment methods and their use;
* Maintain and improve comprehensive earthquake monitoring with focus on "real-time" systems in urban areas;
* Improve the understanding of earthquakes occurrence and their effects and consequences.
  1. CLASS HIERARCHY:

This project consists of six classes which are:

* EarthquakeCityMap class
* CommonMarker class
* CityMarker class which
* EarthquakeMarker class
* LandQuakeMarker class
* OceanQuakeMarker class

EarthquakeCityMap class extends from PApplet of Processing Library. CommonMarker class extends from SimplePointMarker class of Unfolding Map which in turn implements Marker interface. CityMarker class and EarthquakeMarker class extends from CommonMarker. LandQuakeMarker class and OceanQuakeMarker class extends from EarthquakeMarker

The class hierarchy is discussed in detail in chapter-7 along with UML diagram.

* 1. KEY:

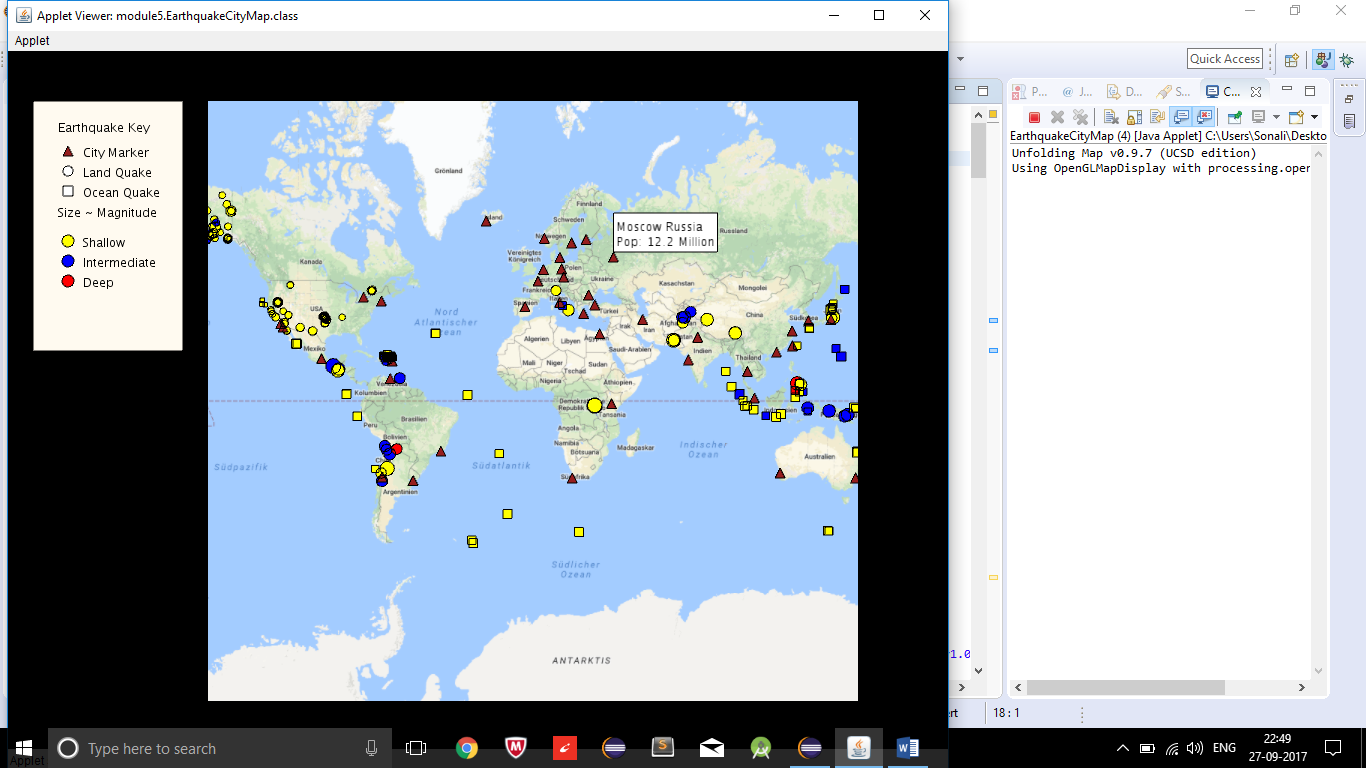


Fig 5b: Key

The key is displayed to the right of the map depicting the below specifications of earthquake markers.

All major cities around the world are marked on the map by a triangle of maroon colour. The earthquakes that occur on land are marked by sphere while those that occur in ocean are marked by square. The dimension of the sphere and square markings vary with the magnitude of earthquake it represents i.e. the earthquake of greater magnitude is represented by symbol larger in size than the symbol which represents an earthquake of smaller magnitude.

Depending on the depth of earthquake, the marked symbols are assigned different colours. If the earthquake occurred at shallow depth(<70 km), it is marked with yellow colour. If it occurred at intermediate depth(between 70 and 300 km), it is marked with blue colour and if it occurred deep(>300kms), it is marked with red colour.

**CHAPTER-6**

**WORKING**

* 1. ZOOM AND PAN :

Unfolding maps also provide the functionality of zooming and panning through createdefaulteventdispatcher().The map can be zoomed in and out by the ’+’ and ‘-‘ buttons on keyboard and also by scrolling the roller present on mouse. Double clicking on a certain location which centers and zooms the map to that location.

Arrow keys can be used to move across the map. Up, Down, Left, Right arrows on keyboard are used to up, down, left and right on the map.

For offline working, zooming is limited to two levels.

The figures 6a and 6b show the zooming effect of earthquake map. Fig 6a shows the zoomed out world map while Fig 6b shows the map zoomed in at Brazil.

In figure 6c and 6d map is moved using arrow keys.

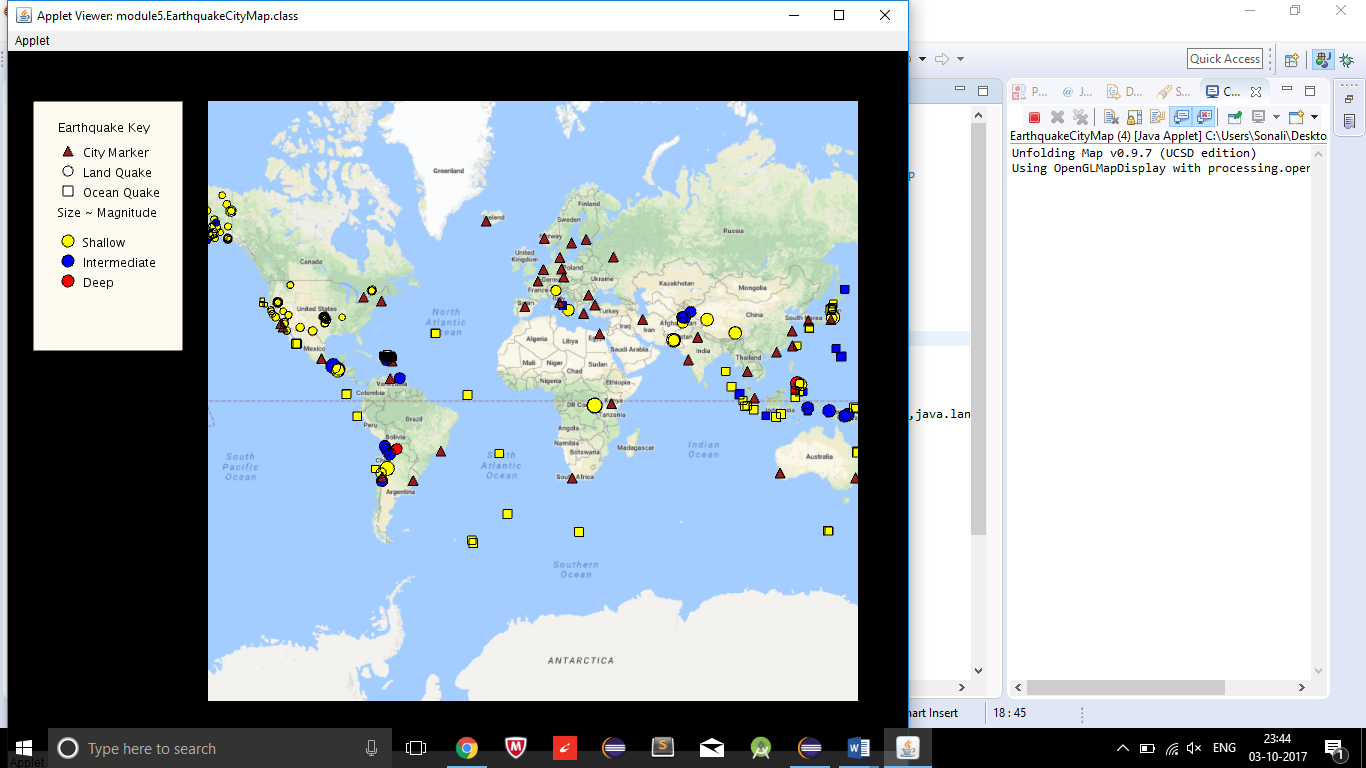


Fig 6a: zoomed out map

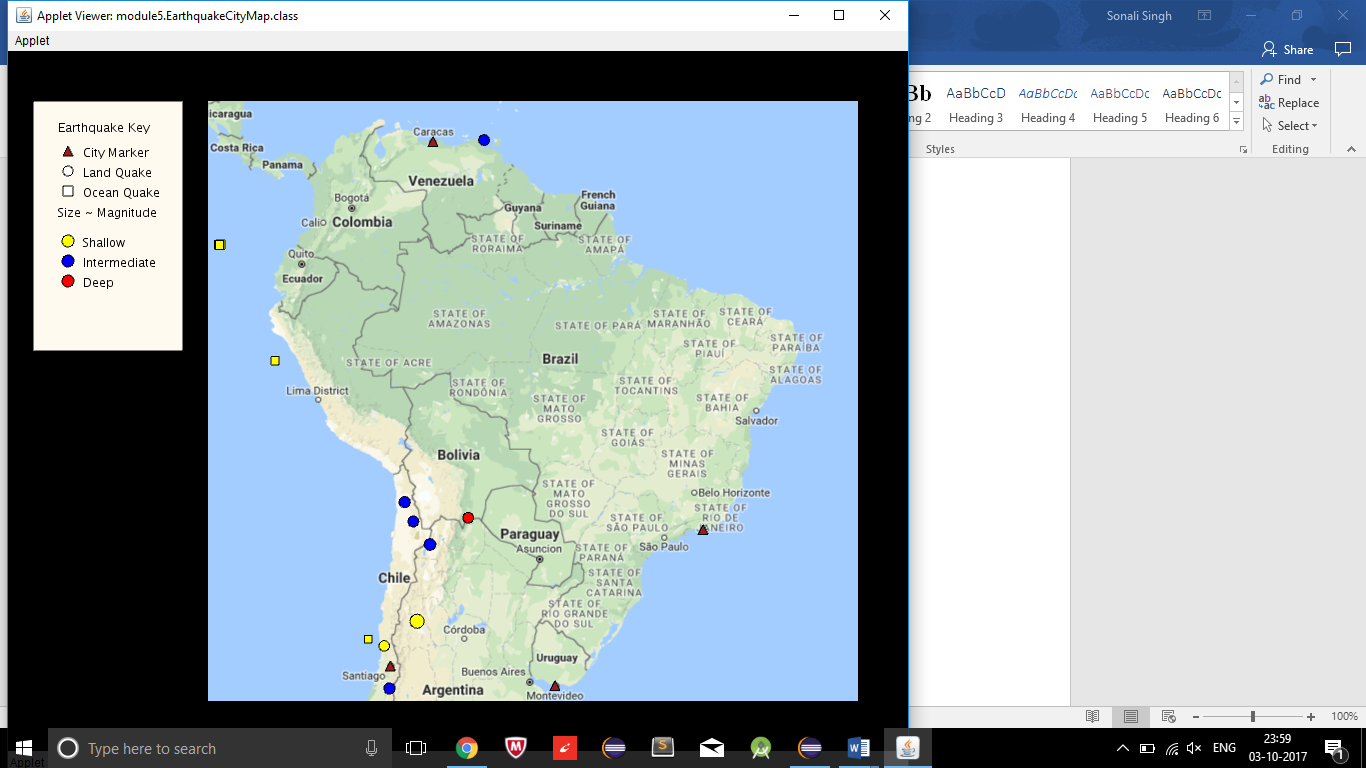


Fig 6b: zoomed in map

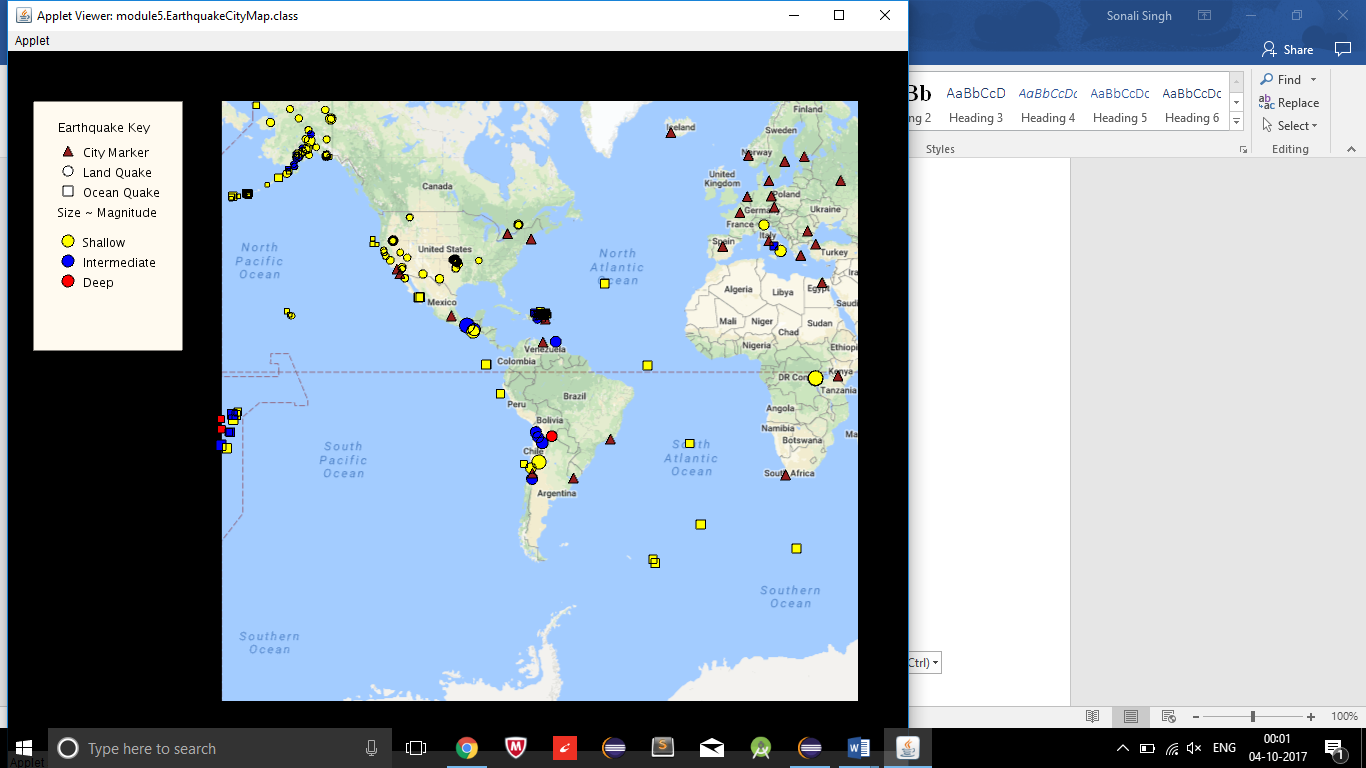


Fig 6c: left half of map visible using left arrow key

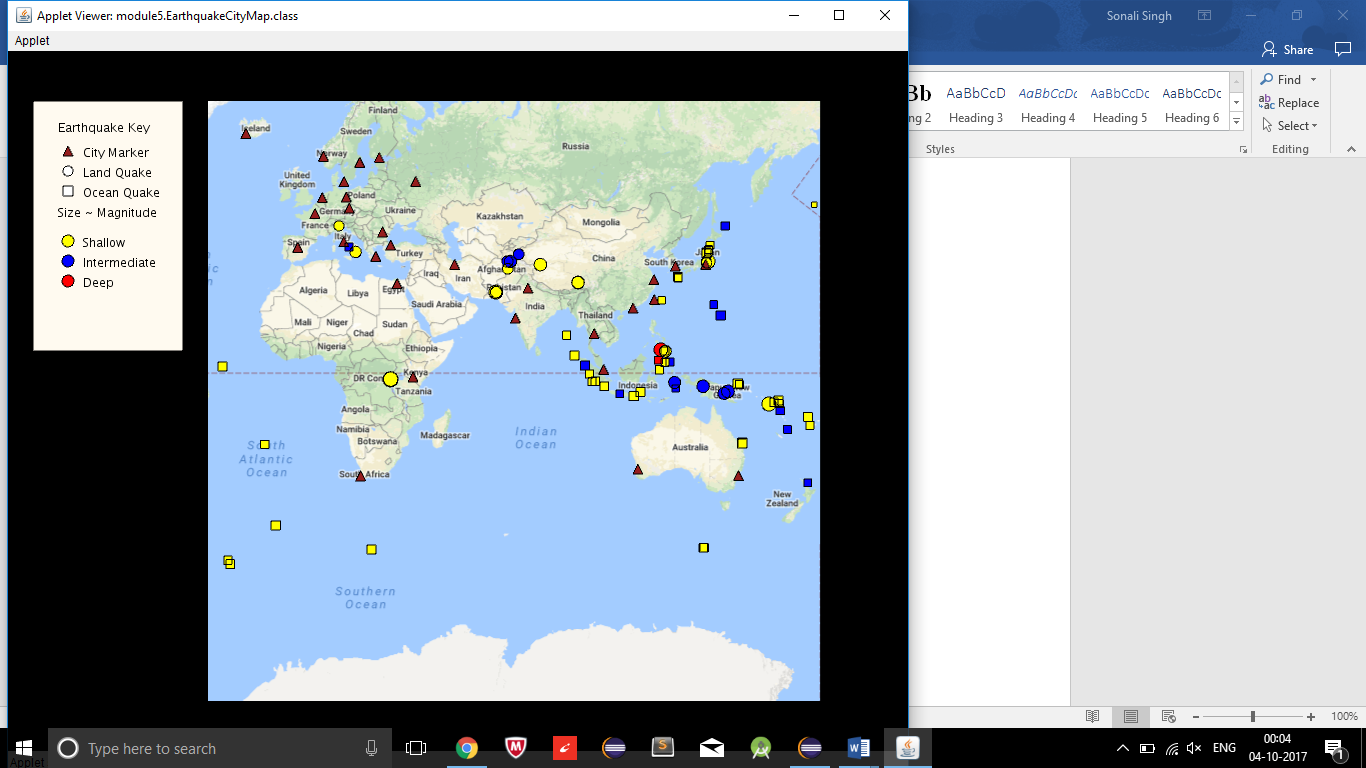


Fig 6d: right half of map visible by using right arrow key

* 1. HOVER:

1. HOVER OVER CITY MARKER:

When we hover over a city marker, a rectangular white box is visible adjacent to the marker displaying details relate to the city such as city name, country name and population of the city.

The figure 6e demonstrates the functionality of hovering over a city marker.

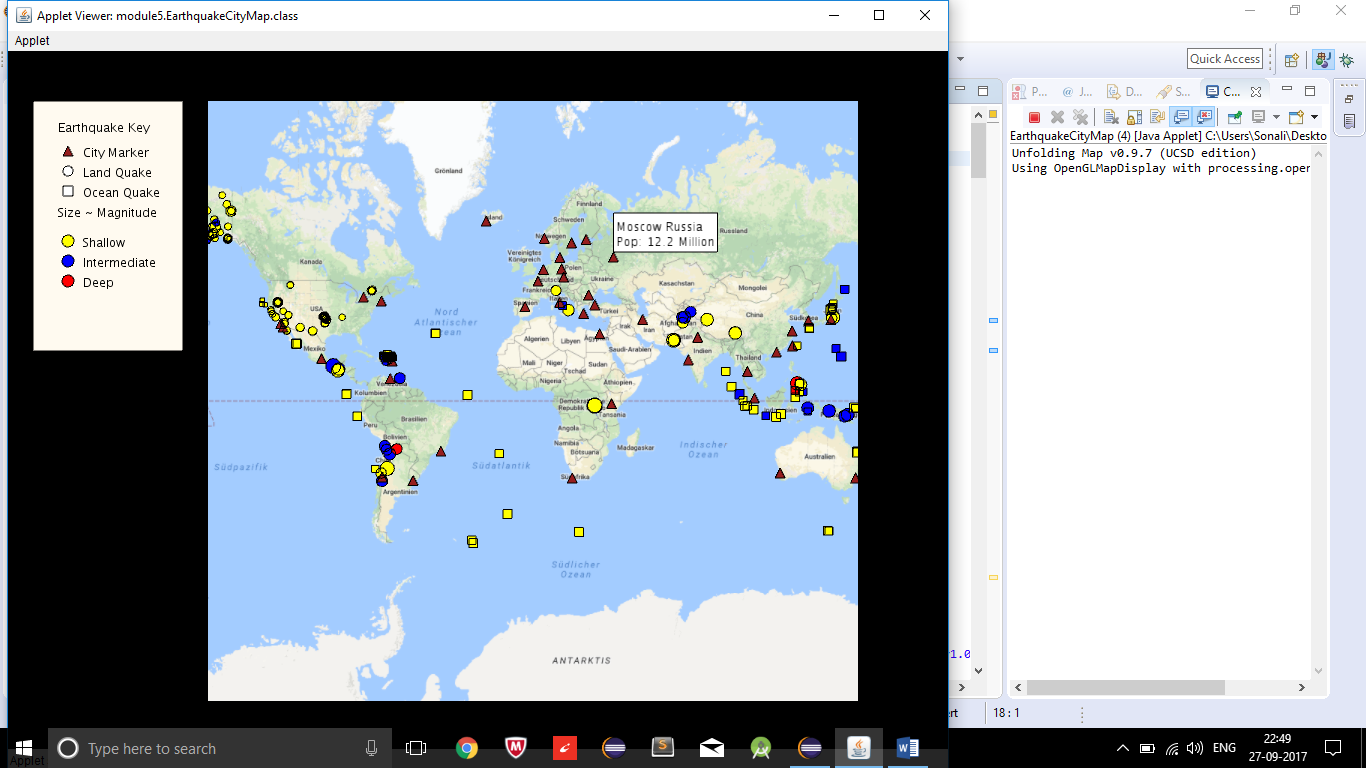


Fig 6e: Hover over city marker

1. HOVER OVER EARTHQUAKE MARKER:

When we hover over an earthquake marker whether it is on land or ocean, a rectangular white box is visible adjacent to the earthquake marker displaying the title of that earthquake which includes the magnitude of that earthquake and its location.

The figure 6f demonstrates the functionality of hovering over an earthquake marker in ocean and figure 6g demonstrates the functionality of hovering over a earthquake marker on land.

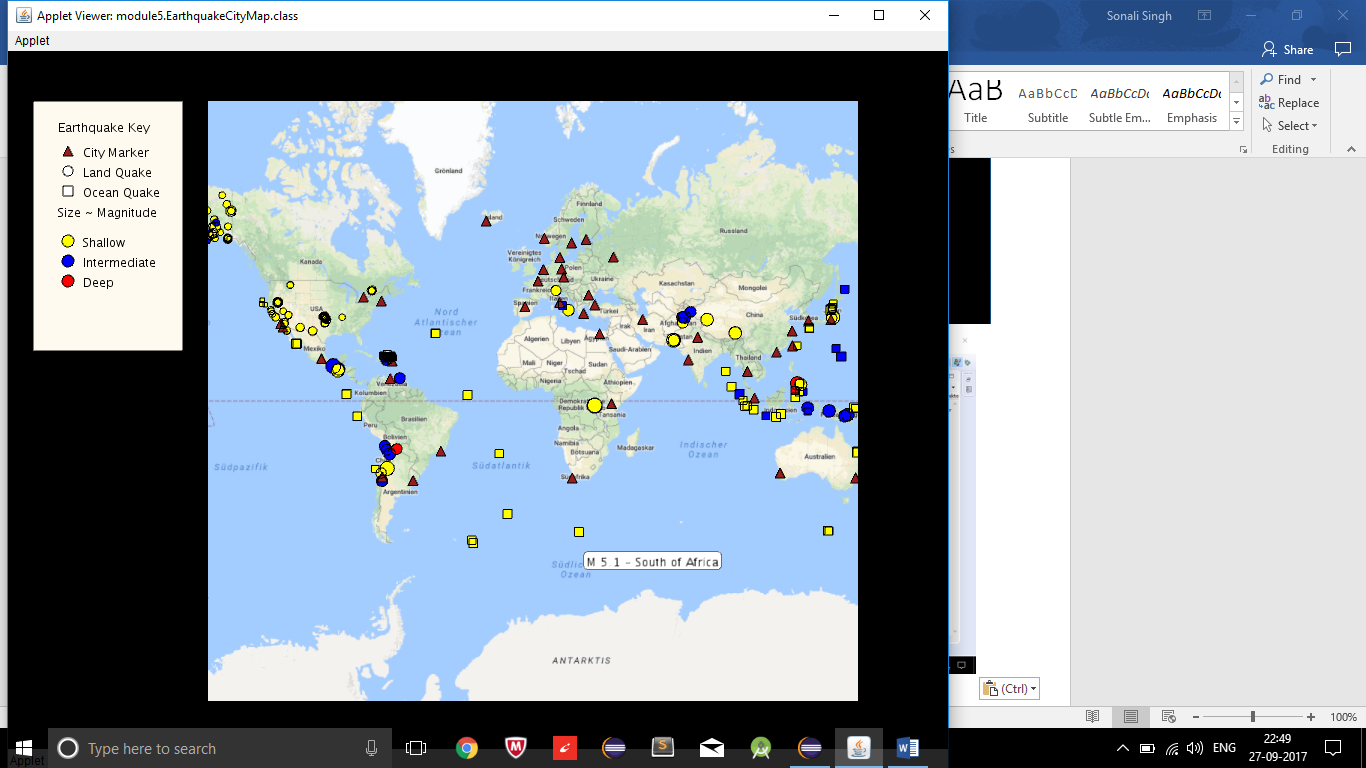


Fig 6f: Hover over earthquake marker in ocean

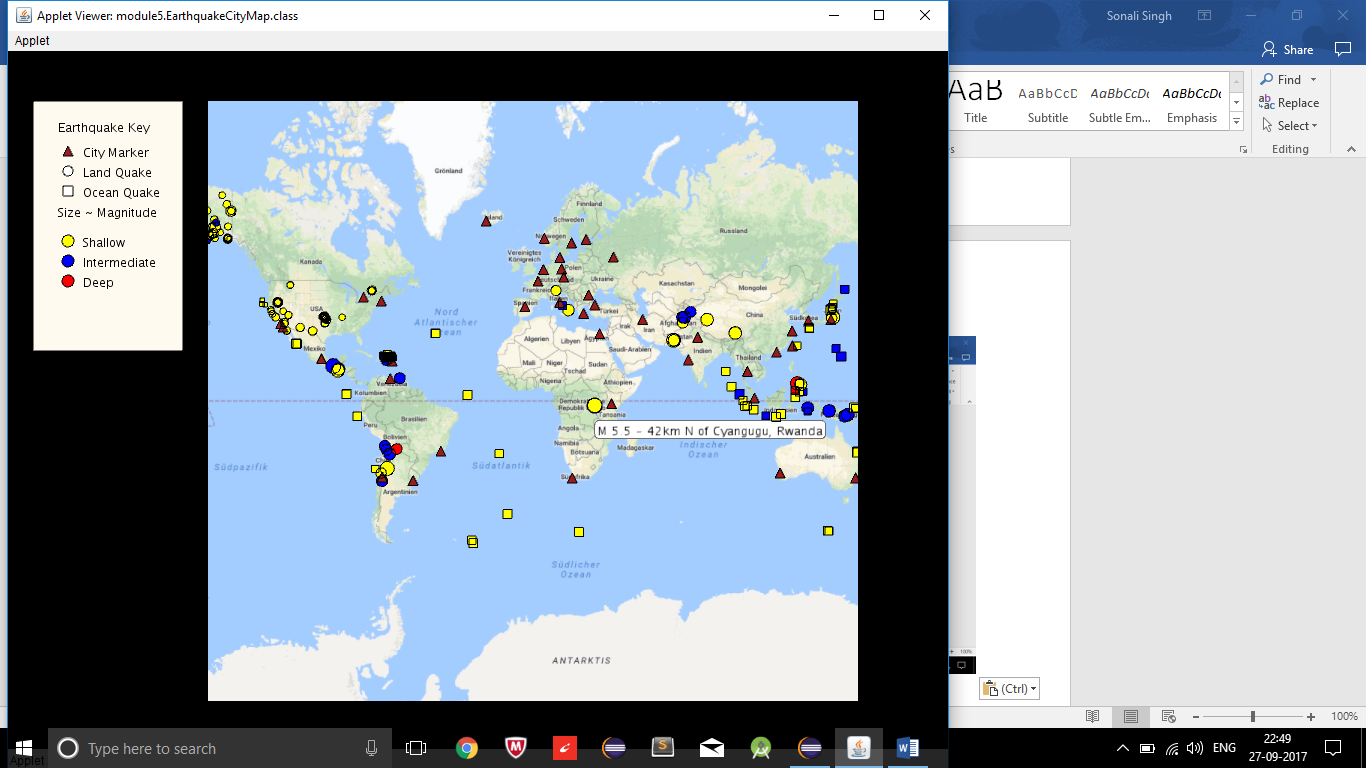


Fig 6g: Hover over earthquake marker on land

* 1. CLICK:

1. CLICK ON CITY MARKER:

When we click on a city marker, only that city marker and the marker of earthquake that has occurred there are visible.

Fig 6h demonstrates the functionality of clicking on city marker.

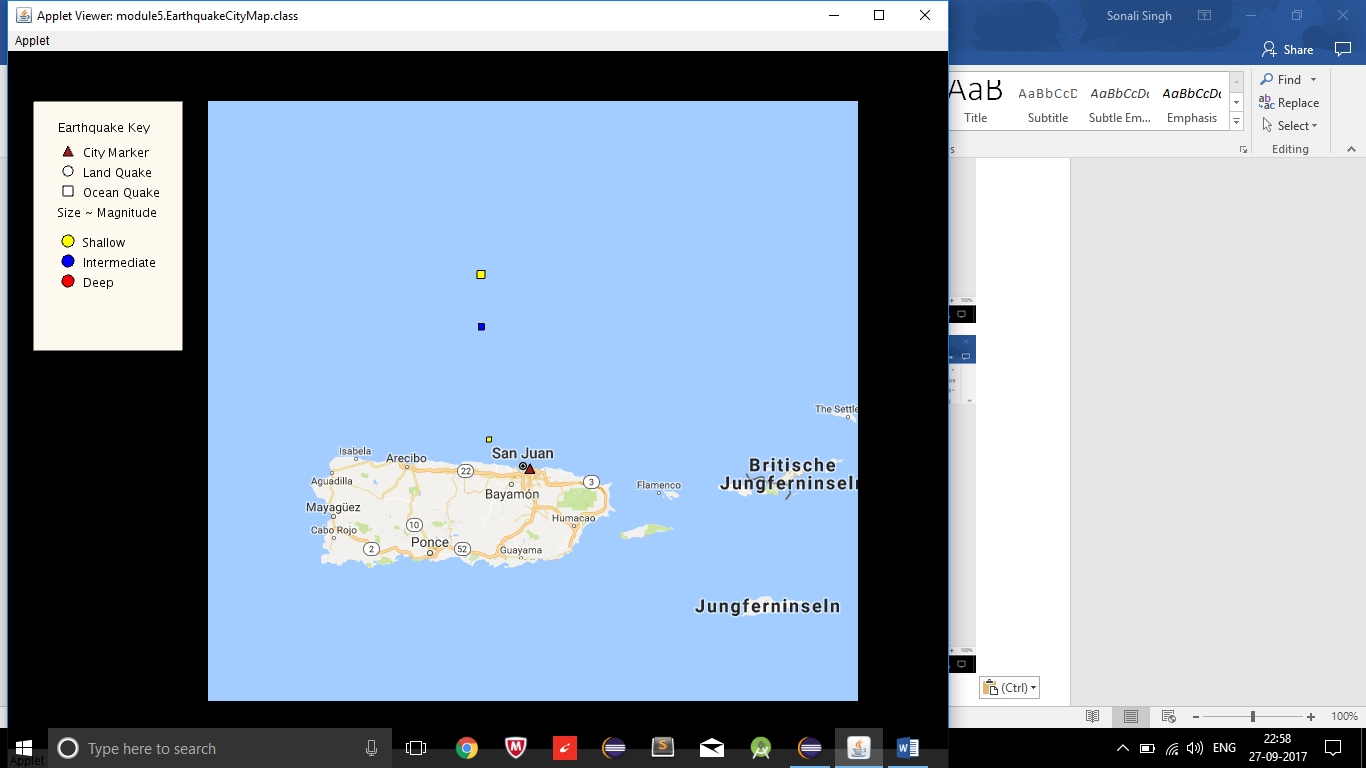


Fig 6h: click on city marker

1. CLICK ON EARTHQUAKE MARKER:

When we click on any earthquake marker whether it is on land or ocean, only that earthquake marker and the city marker within the threat circle of that earthquake are displayed. All other markers are hidden.

The figure 6i demonstrates the functionality of clicking over an earthquake marker in ocean and figure 6j demonstrates the functionality of clicking over an earthquake marker on land.

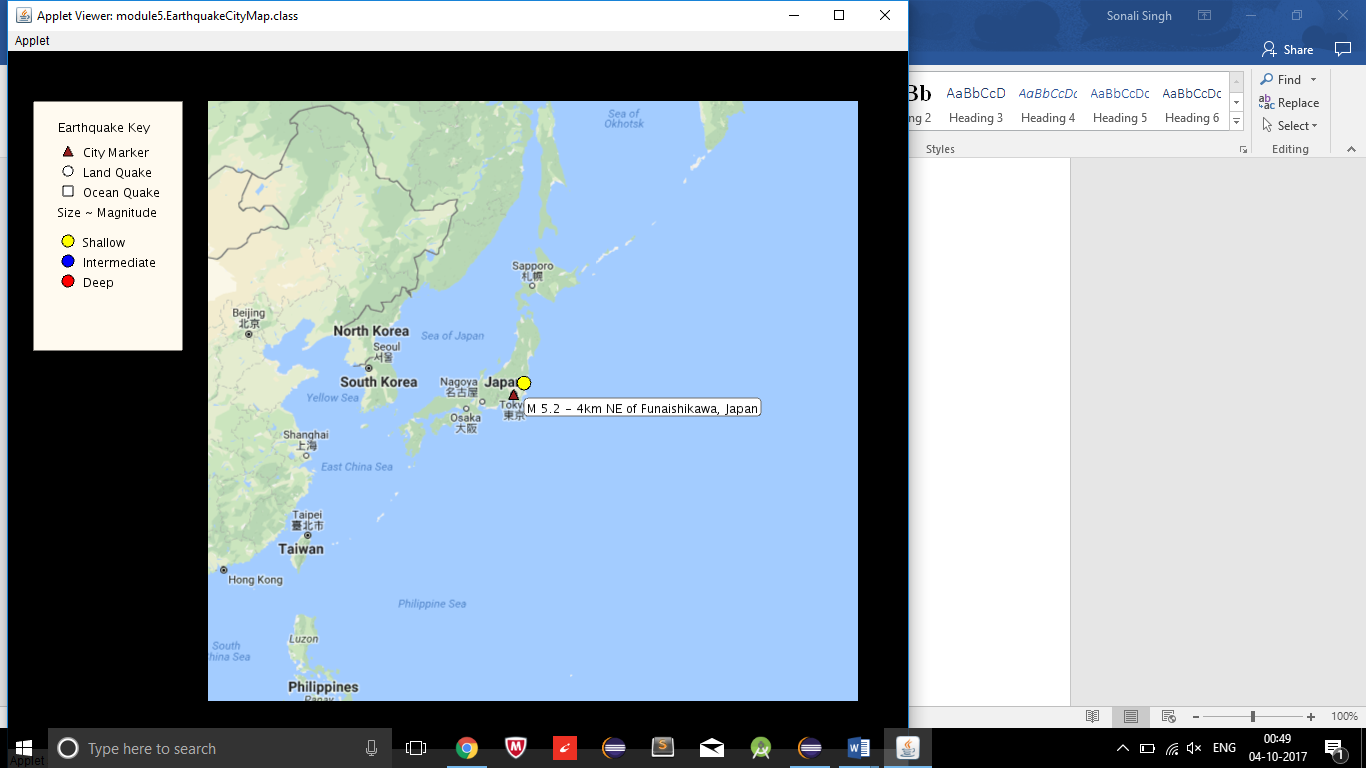


Fig 6i: click on an earthquake marker on land

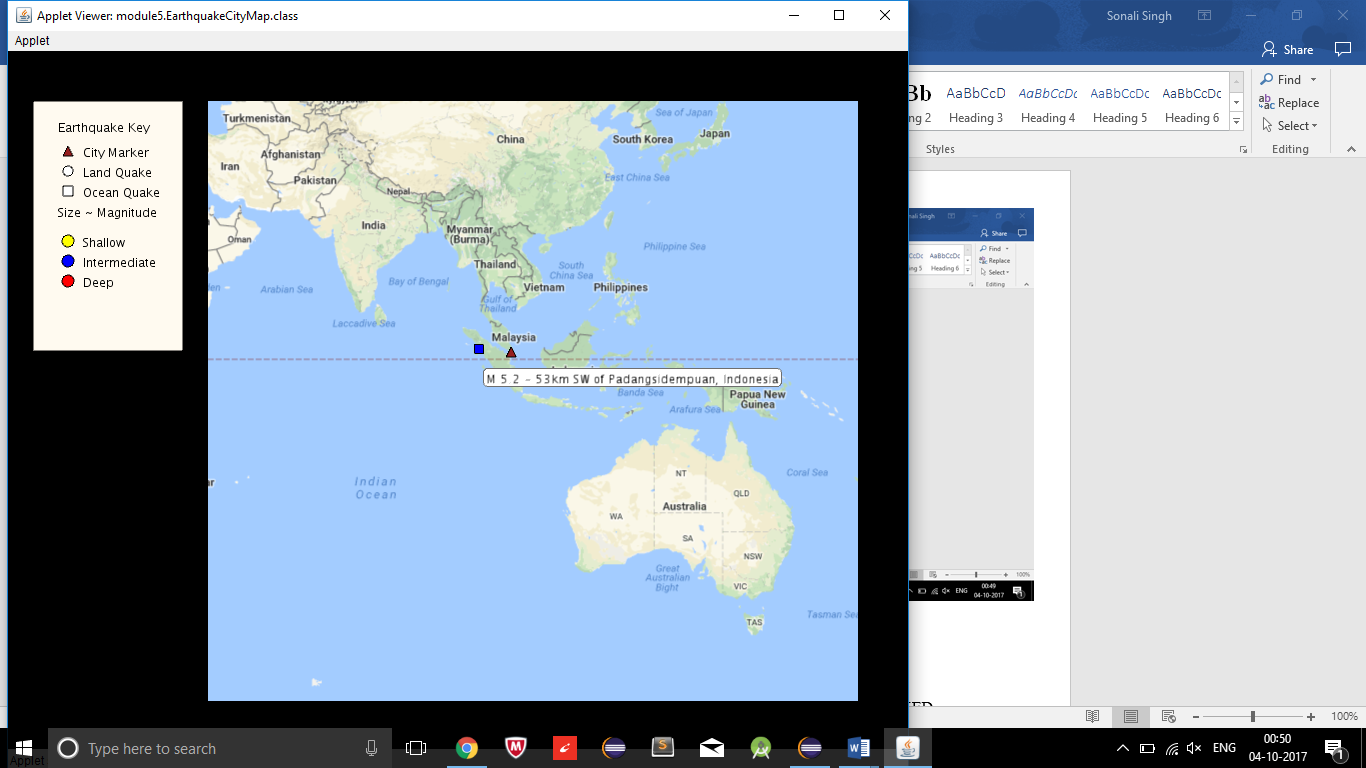


Fig 6j: Click on an earthquake marker in ocean

1. CLICKING ANYWHERE ELSE WHEN A MARKER IS ALREADY CLICKED:

All the hidden markers are visible again by clicking anywhere on the map when a marker is already clicked whether it is a city marker or earthquake marker. Figures 6k and 6l demonstrate this functionality.

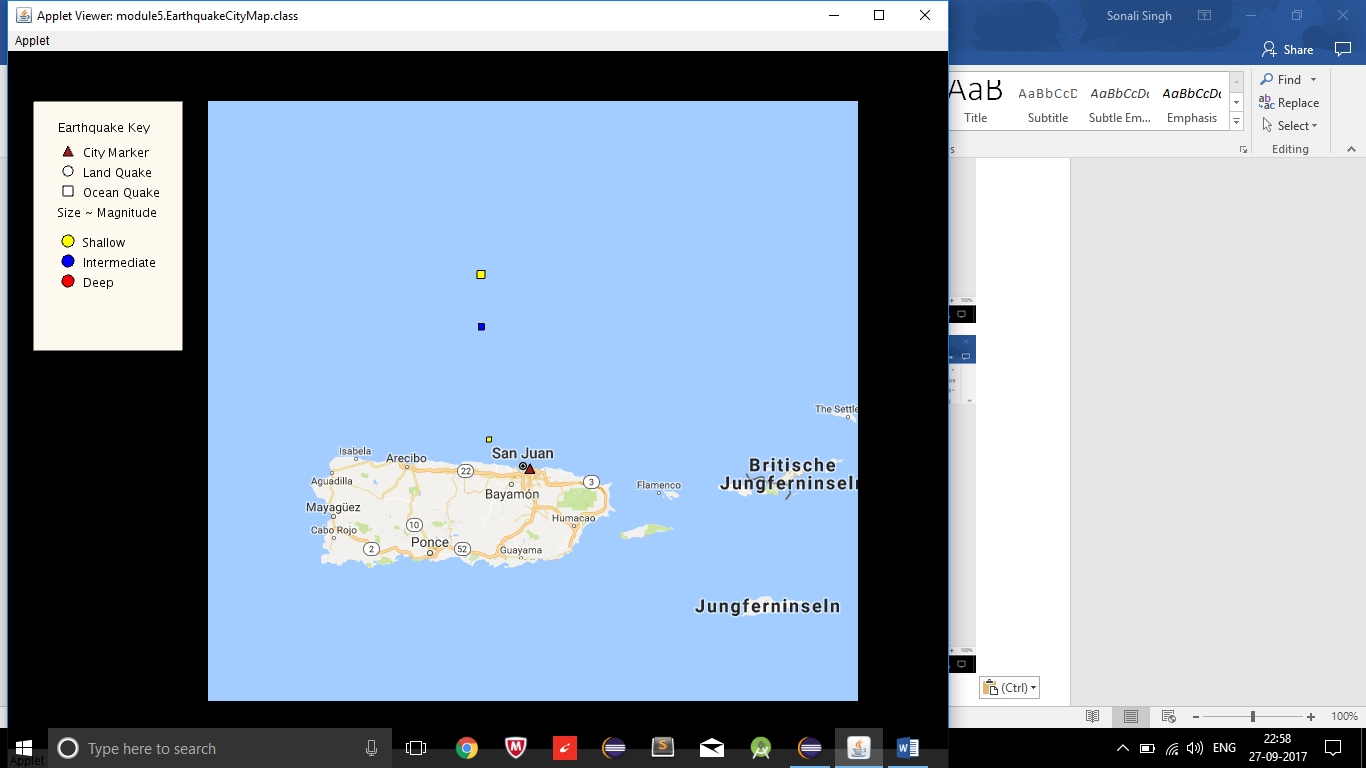


Fig 6k: when a marker is clicked

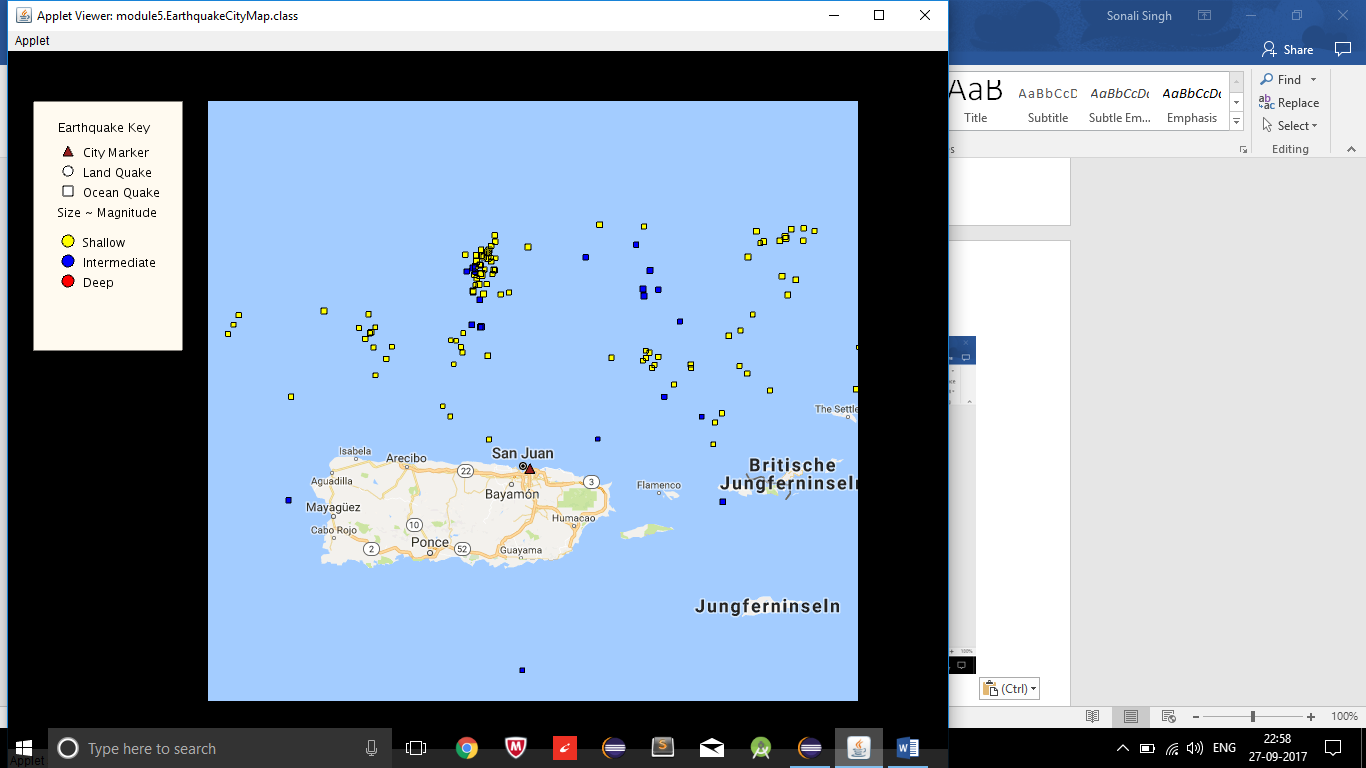


Fig 6l: clicking anywhere else on map when a marker is already clicked

* 1. OFFLINE MAP:

For offline working, a map tile is provided and a sample earthquake data is used. The zooming is also limited to two levels.

Figure 6m shows the offline map.

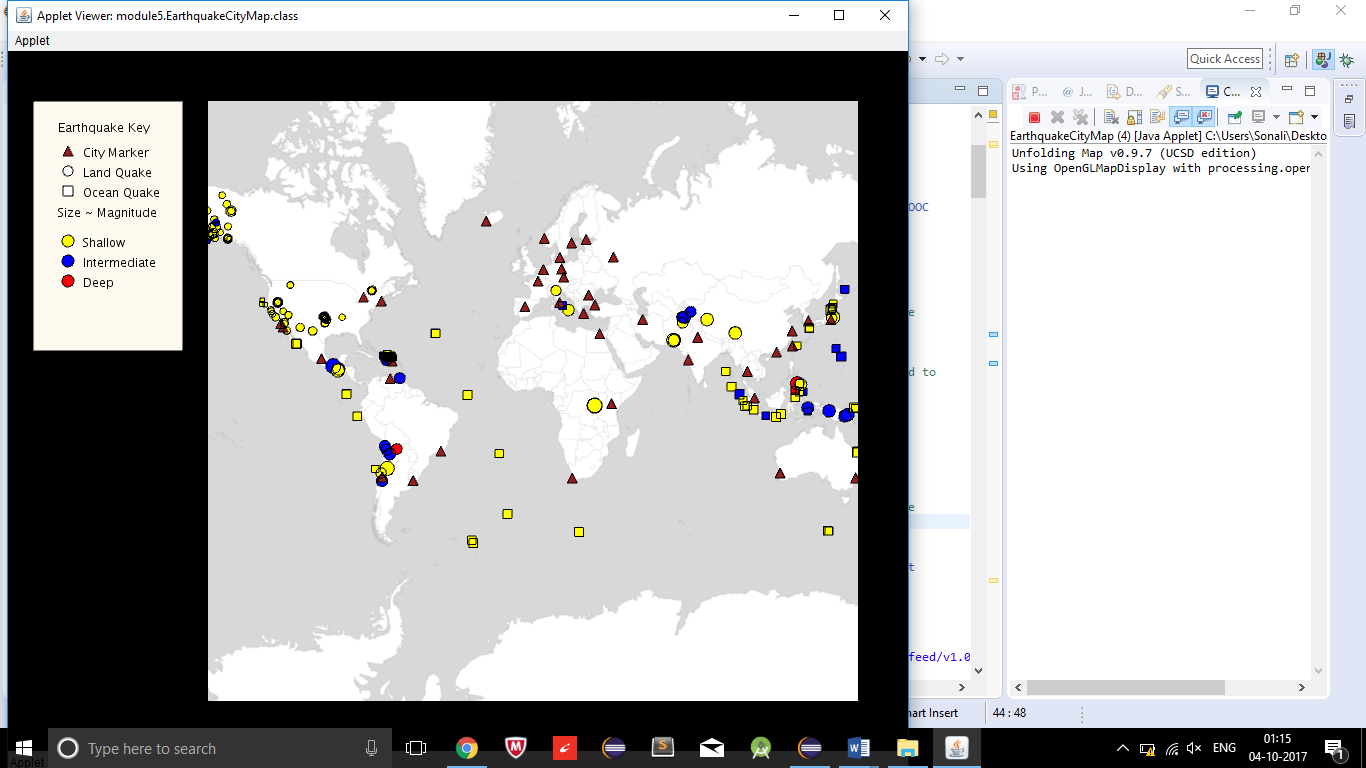


Fig 6m: Offline map

**CHAPTER-7**

**CLASS HIERARCHY**

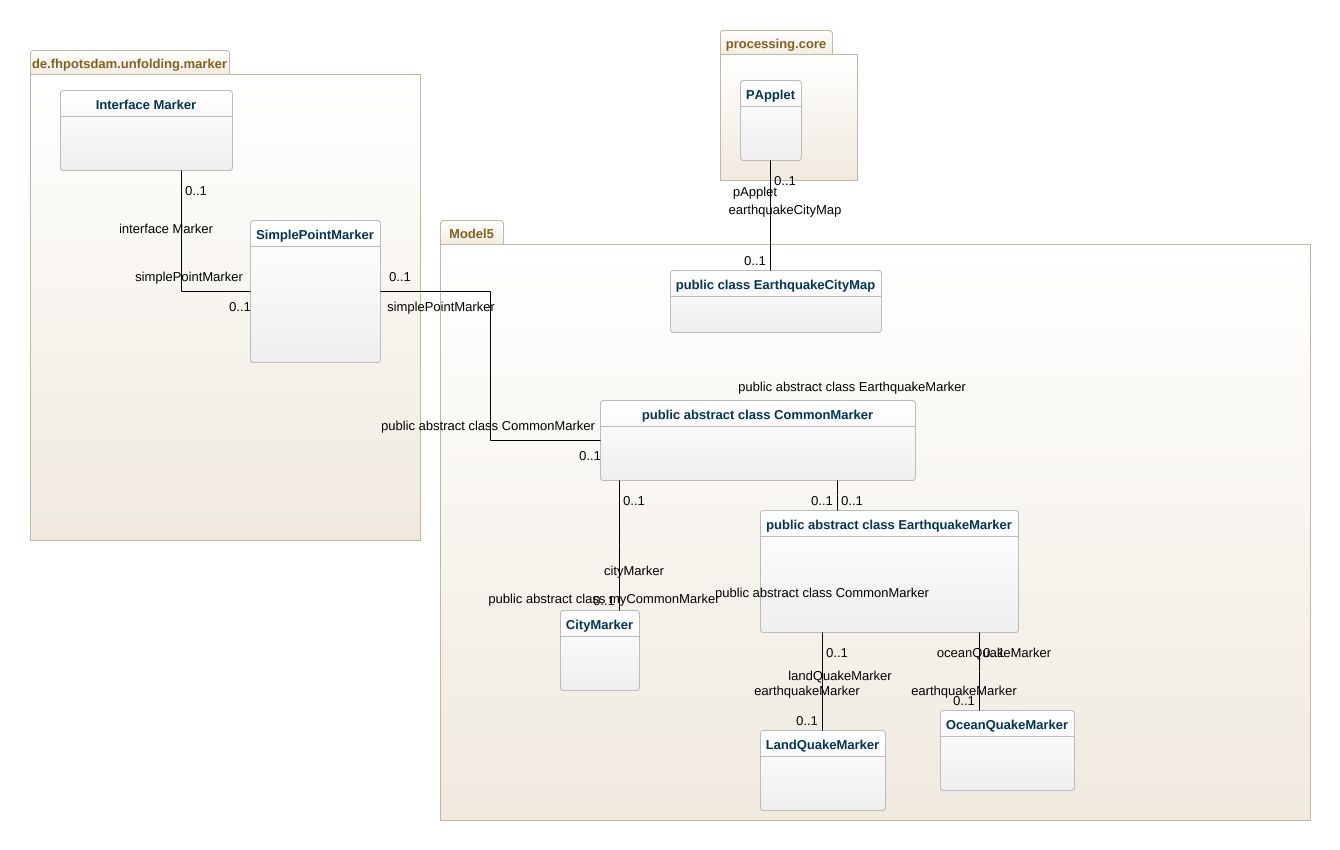


Fig: UML diagram of Earthquake Map Project

1. EarthquakeCityMap class:

This class extends PApplet class of processing library.It contains the setup() method which sets map using constructor of unfolding map which takes map provider as parameter.The createDefaultEventDispatcher() is fired which enables zoom and pan functionality. The country features are loaded and stored in countryMarkers list, city data is read and stored in cityMarkers list and earthquake data is read from RSS feed and stored in earthquakes list. For every earthquake Feature in list, if it is on land then a LandQuakeMarker is added to quakeMarkers list else an OceanQuakeMarker is added to quakeMarkers list. All the quakeMarkers and cityMarkers are added to map and displayed on map. The draw() method draws map as many number of times it is called.

The mouseMoved() is an event handler that gets called automatically when the mouse moves. It clears the last selection and calls selectMarkerHover() for list of quakeMarkers followed by cityMarkers. In selectMarkerIfHover(), if there is a marker under the cursor and lastSelected is null, lastSelected is set to be the first marker found under the cursor.

The mouseClicked() is an event handler for mouse clicks. It will display an earthquake and its threat circle of cities or if a city is clicked, it will display all the earthquakes where the city is in the threat circle. It uses two helper methods checkCitiesForClick() and checkEarthquakesForClick() which checks if a city marker or earthquake marker is clicked and responds accordingly. The unhideMarkers() unhides all the hidden markers. The addKey() draws a key to the left of map.

The isLand() calls the helper method isInCountry() which checks whether this quake occurred on land. If it did, the helper method sets the "country" property of its PointFeature to the country where it occurred and island() returns true. Otherwise it returns false.

1. CommonMarker class:

This class extends from SimplePointMarker class of unfolding maps which in turn implements Marker interface. It is an abstract class and contains two abstract methods drawMarker() and showTitle(). It has two constructors. One constructor takes location as parameter and makes a point marker while the othe constructor takes location as well as list of properties and constructs a point marker for it.

The getClicked() method returns state of protected boolean variable ‘clicked’ while setClicked() is used to set the value of clicked to state passed as parameter. If a marker is not hidden draw() method calls drawMarker() method and if it is selected as well , then the showTitle() is called.

1. CityMarker class:

This class extends CommonMarker class and has two constructors. The first one takes location as parameter and calls the first constructor of parent class while second constructor takes Feature class object as parameter and calls second constructor of parent class with location and properties as parameters for them.

It also Implements the drawMarker(), which makes a triangle marker for cities and showTitle() methods.It contains getter methods to get city, country and population and uses it in showTitle().

1. EarthquakeMarker class:

It is an abstract class which extends from CommonMarker class and contains an abstract function drawEarthquake(). The constructor takes an object of PointFeature class of unfolding map as parameter.The constructor gets the magnitude of earthquake using PointFeature and Feature class through getProperties() method. A new property ‘radius’ is set having value equal to twice the magnitude of earthquake using setProperties() method.

The showTitle() method and drawMarker() method are coded. The drawMarker() calls drawEathquake() and colorDetermine() methods. The colorDetermine() determines the colour of earthquake marker based on their depth. The threatcircle is calculated in km using magnitude of earthquake which is useful for click functionality of earthquake map.

This class also includes various getter methods to get magnitude, title, depth, radius and isOnLand properties.

1. LandQuakeMarker class:

This class extend EarthquakeMarker class. It’s constructors takes PointFeature object as parameter, calls the constructor of parent class and sets the value of isOnLand variable to true. It contains a getter method to get country of earthquake and drawEarthquake() to draw a circle marker for earthquakes that occur on land.

1. OceanQuakeMarker class:

This class extend EarthquakeMarker class. It’s constructors takes PointFeature object as parameter, calls the constructor of parent class and sets the value of isOnLand variable to false. It contains drawEarthquake() to draw a square marker for earthquakes that occur in ocean.

**CONCLUSION**

From this project report, we conclude that visualizing a data helps in easy understanding of the problem at hand as it helps to identify a pattern, know the problem areas and decide upon an action plan.

This project resolves the issue of visualizing earthquake data obtained from the USGS API on a map provided by google, Microsoft or any other map provider to study the previous earthquake data, identify a pattern and come up with optimal solutions to predict or identify the earthquake prone zones, take preventive measures on time and reduce the impact of destruction caused by earthquakes.

This project is an implementation of all the concepts of data structure and algorithms that was taught at Coding Blocks which include Arrays, Generics, Inheritance, Polymorphism, Function Overriding, Function Overloading, Interface, Class, Hash Maps, Strings, Array Lists etc

.

Apart from implementing the acquired knowledge, I also got a chance to work with the documentation of Unfolding Maps Library and Processing Library. Nowadays almost all projects that are developed in software industry are not built from scratch. Instead it is built upon existing Libraries. Thus, this project has equipped me with the confidence to work with documentation of existing libraries and provided a glimpse of how large-scale software projects are developed.

The scope of this project can further be extended by adding more functionality to the existing ones. For example, the Airports and railway stations across countries can be plotted on the map and the real-time traffic monitored so that an efficient escape plan can be determined to evacuate people in times of calamities like Tsunamis caused due to earthquake or to provide help, food and other materials to the affected areas during rescue operation.

**BIBLIOGRAPHY**

* <https://processing.org/overview/>
* <http://unfoldingmaps.org/>
* <https://www.coursera.org/>
* <http://pages.cs.wisc.edu/~cs302/labs/EclipseTutorial_Processing/Step_04.html>
* <https://simple.wikipedia.org/wiki/Earthquake>
* <https://earthquake.usgs.gov/aboutus/mission.php>
* <https://www.sas.com/en_us/insights/big-data/data-visualization.html#dmimportance>

**APPENDIX-1**

LIST OF MAP PROVIDERS

1. AbstractMapProvider:

AbstractMapTileProvider   
[AbstractMapTileUrlProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AbstractMapTileUrlProvider.html) 

1. AcetateProvider:  
   [AcetateProvider.All](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.All.html)   
   [AcetateProvider.Basemap](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.Basemap.html)   
   [AcetateProvider.Foreground](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.Foreground.html)   
   [AcetateProvider.GenericAcetateProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.GenericAcetateProvider.html)   
   [AcetateProvider.Hillshading](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.Hillshading.html)   
   [AcetateProvider.Labels](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.Labels.html)   
   [AcetateProvider.Roads](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.Roads.html)   
   [AcetateProvider.Terrain](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/AcetateProvider.Terrain.html)
2. EsriProvider:  
   [EsriProvider.DeLorme](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.DeLorme.html)   
   [EsriProvider.GenericEsriProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.GenericEsriProvider.html)   
   [EsriProvider.NatGeoWorldMap](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.NatGeoWorldMap.html)   
   [EsriProvider.OceanBasemap](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.OceanBasemap.html)   
   [EsriProvider.WorldGrayCanvas](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.WorldGrayCanvas.html)   
   [EsriProvider.WorldPhysical](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.WorldPhysical.html)   
   [EsriProvider.WorldShadedRelief](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.WorldShadedRelief.html)   
   [EsriProvider.WorldStreetMap](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.WorldStreetMap.html)   
   [EsriProvider.WorldTerrain](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.WorldTerrain.html)   
   [EsriProvider.WorldTopoMap](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/EsriProvider.WorldTopoMap.html)
3. GeoMapApp:  
   [GeoMapApp.GeoMapAppProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/GeoMapApp.GeoMapAppProvider.html)   
   [GeoMapApp.TopologicalGeoMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/GeoMapApp.TopologicalGeoMapProvider.html)
4. Google:  
   [Google.GoogleMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Google.GoogleMapProvider.html)   
   [Google.GoogleProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Google.GoogleProvider.html)   
   [Google.GoogleSimplified2Provider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Google.GoogleSimplified2Provider.html)   
   [Google.GoogleSimplifiedProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Google.GoogleSimplifiedProvider.html)   
   [Google.GoogleTerrainProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Google.GoogleTerrainProvider.html)
5. ImmoScout:

[ImmoScout.HeatMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ImmoScout.HeatMapProvider.html)   
[ImmoScout.ImmoScoutProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ImmoScout.ImmoScoutProvider.html) 

1. MapBox:

[MapBox.BlankProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.BlankProvider.html)   
[MapBox.ControlRoomProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.ControlRoomProvider.html)   
[MapBox.LacquerProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.LacquerProvider.html)   
[MapBox.MapBoxProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.MapBoxProvider.html)   
[MapBox.MuseDarkStyleProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.MuseDarkStyleProvider.html)   
[MapBox.PlainUSAProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.PlainUSAProvider.html)   
[MapBox.WorldLightProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapBox.WorldLightProvider.html) 

1. MapQuestProvider:

[MapQuestProvider.Aerial](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapQuestProvider.Aerial.html)   
[MapQuestProvider.GenericMapQuestProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapQuestProvider.GenericMapQuestProvider.html)   
[MapQuestProvider.OSM](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MapQuestProvider.OSM.html) 

1. Microsoft:

[Microsoft.AerialProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Microsoft.AerialProvider.html)   
[Microsoft.HybridProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Microsoft.HybridProvider.html)   
[Microsoft.MicrosoftProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Microsoft.MicrosoftProvider.html)   
[Microsoft.RoadProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Microsoft.RoadProvider.html)

1. OpenMapSurferProvider:

[OpenMapSurferProvider.GenericOpenMapSurferProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenMapSurferProvider.GenericOpenMapSurferProvider.html)   
[OpenMapSurferProvider.Grayscale](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenMapSurferProvider.Grayscale.html)   
[OpenMapSurferProvider.Roads](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenMapSurferProvider.Roads.html)

1. OpenStreetMap:

[OpenStreetMap.CloudmadeProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenStreetMap.CloudmadeProvider.html)   
[OpenStreetMap.GenericOpenStreetMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenStreetMap.GenericOpenStreetMapProvider.html)   
[OpenStreetMap.OpenStreetMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenStreetMap.OpenStreetMapProvider.html)   
[OpenStreetMap.OSMGrayProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenStreetMap.OSMGrayProvider.html) 

1. OpenWeatherProvider:  
   [OpenWeatherProvider.Clouds](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Clouds.html)   
   [OpenWeatherProvider.CloudsClassic](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.CloudsClassic.html)   
   [OpenWeatherProvider.GenericOpenWeatherMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.GenericOpenWeatherMapProvider.html)   
   [OpenWeatherProvider.Precipitation](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Precipitation.html)   
   [OpenWeatherProvider.PrecipitationClassic](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.PrecipitationClassic.html)   
   [OpenWeatherProvider.Pressure](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Pressure.html)   
   [OpenWeatherProvider.PressureContour](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.PressureContour.html)   
   [OpenWeatherProvider.Rain](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Rain.html)   
   [OpenWeatherProvider.RainClassic](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.RainClassic.html)   
   [OpenWeatherProvider.Snow](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Snow.html)   
   [OpenWeatherProvider.Temperature](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Temperature.html)   
   [OpenWeatherProvider.Wind](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/OpenWeatherProvider.Wind.html)
2. StamenMapProvider:

[StamenMapProvider.Toner](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/StamenMapProvider.Toner.html)   
[StamenMapProvider.TonerBackground](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/StamenMapProvider.TonerBackground.html)   
[StamenMapProvider.TonerLite](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/StamenMapProvider.TonerLite.html)   
[StamenMapProvider.WaterColor](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/StamenMapProvider.WaterColor.html) 

1. ThunderforestProvider:

[ThunderforestProvider.GenericThunderforestProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ThunderforestProvider.GenericThunderforestProvider.html)   
[ThunderforestProvider.Landscape](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ThunderforestProvider.Landscape.html)   
[ThunderforestProvider.OpenCycleMap](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ThunderforestProvider.OpenCycleMap.html)   
[ThunderforestProvider.Outdoors](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ThunderforestProvider.Outdoors.html)   
[ThunderforestProvider.Transport](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ThunderforestProvider.Transport.html)

1. Yahoo:

[Yahoo.AerialProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Yahoo.AerialProvider.html)   
[Yahoo.HybridProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Yahoo.HybridProvider.html)   
[Yahoo.RoadProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Yahoo.RoadProvider.html)   
[Yahoo.YahooProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/Yahoo.YahooProvider.html)

1. Others:  
   [CartoDBProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/CartoDBProvider.html)[ImmoScout](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/ImmoScout.html)    
   [MBTilesMapProvider](http://unfoldingmaps.org/javadoc/de/fhpotsdam/unfolding/providers/MBTilesMapProvider.html)