**Software Requirements Specification**

**for**

**Searchquake**

**Prepared by Adam Davis, Kyle Emery, and**

**Jacob Kickbush**

**April 6, 2018**

**1 Introduction**

* 1. **Purpose**

This document describes the requirements for the Searchquake application, which allows users to explore earthquake data.

* 1. **Document Conventions**

The functional requirements for this project are organized by Use Case within the corresponding Primary Actor group.

**2 Overall Description**

**2.1 Use Cases**

**2.1.1 User**

A user is someone who uses Searchquake to explore earthquake data.

UC-U-1 Type “Help” for a list of commands

UC-U-2 Type “Summary” to print all of the data

UC-U-3 Type “Print” to print all earthquake events

UC-U-4 Type “Printby” to print earthquake events sorted by some field

UC-U-5 Type “Search” to search by specific criteria

UC-U-6 Type “Exit” to end the application

**3 Functional Requirements**

**3.1 Use Cases for Users**

**UC-U-1: Help Command**

**Description:** Typing help in the console prints out a description of all commands and how to invoke them appropriately.

Command.Help Using a while loop that runs our user’s commands until they break it, if the user types help, it will print out all of the available commands and what they do.

**UC-U-2: Summary Command**

**Description:** Typing Summary in the console prints out a summary of all the data (# of events,

Time range of the events, etc.)

Command.Summary If the user enters summary, it will print a summary of our EarthquakeCollection’s data

**UC-U-3: Print Command**

**Description:** Typing Print in the console prints out all of the earthquake events

Command.Print If the user enters Print, it will print out the information for every earthquake.

**UC-U-4: PrintBy Command**

**Description:**  Typing PrintBy\_\_\_ print out all of the earthquake events sorted by some field

Command.PrintBy\_\_ If the user enters PrintBy\_\_, use a comparator to print out the earthquake

events in descending order based on the field provided by the user.

**UC-U-5: SearchBy Command**

**Description:** Allows the user to search earthquake events that meet some criteria.

Command.SearchBy\_ If the user enters SearchBy\_\_, we take the string as an argument,

match the field in earthquakes and output a list of matching earthquakes.

First define the arrayLists, then loop through the earthquakes, and if it

matches the argument, add to a list and return it. Input errors are handled if

the user invokes the command incorrectly.

**UC-U-6: Exit Command**

**Description:** Allows the user to exit the application whent hey are finished

Command.Exit If the user types exit, the while loop will end and they will be unable to enter any more commands.

.

Main Class:

Main method: We start a while loop that asks the user for inputs in the console continually until they exit out of it. It looks at what the user types, ignoring capitalization, and executes the correct method. If the command does not exist, it will tell the user that and tell them to look at the help command for a list of all commands.

Earthquake Class:

Earthquake Constructor: Takes 22 variables that describe the earthquake and creates an object with those variables.

Getters for every variable: Can be called to get that value in the correct data type for the earthquake object

toString override: override the basic to string of the object and prints out a custom string that lists all of the variables of that earthquake

Comparators for Location, Date, and Status (strings): These comparators takes in string inputs, normalize them to all uppercase and in the return they are evaluated with a .compareTo() function. The comparator is then called in the main file with the user input, and the array list is sorted and returns it to the console.

Comparators for Depth, Magnitude (doubles): These comparators take numerical inputs and evaluate them with a logical statement. Then, they output either a -1, 1 or 0. The comparator is then called in the main file with the user input, and the array list is sorted and returns it to the console.

Earthquake Collection:

arrayListtoString Method: this takes all of the data from our arraylist of earthquakes, and takes the size, date range, latitude and longitude range, average depth, average magnitude, magnitude types, average number of stations contributing, average gap between stations, amount of earthquakes vs mining blasts, and amount of automatic statuses vs reviewed. It finds all of these and prints it out as a summary of the earthquake arraylist’s data.

Searchbystatus: takes a string argument from the user, looks through every earthquake in our arraylist. It then uses the get status to check if this earthquake’s status equals our argument. If it does then it adds to an arraylist of matching earthquakes and returns it.

Searchbyplace: takes a string argument from the user, looks through every earthquake in our arraylist. It then uses the get place to check if this earthquake’s place equals our argument. If it does then it adds to an arraylist of matching earthquakes and returns it.

Searchbymagtype: takes a string argument from the user, looks through every earthquake in our arraylist. It then uses the get magtype to check if this earthquake’s magtype equals our argument. If it does then it adds to an arraylist of matching earthquakes and returns it.

Searchbymag: Takes two string arguments and parses them into doubles. We then loop through every earthquake and get its magnitude. We check to see if the earthquake’s mag is in between our two arguments, and if it is then it adds it to the arraylist of matching earthquakes and returns it.

Searchbydepth: Takes two string arguments and parses them into doubles. We then loop through every earthquake and get its depth. We check to see if the earthquake’s depth is in between our two arguments, and if it is then it adds it to the arraylist of matching earthquakes and returns it.

Searchbylocation: Takes four string arguments and parses them into doubles. This is the lat 1, long 1, lat 2, long 2. We then loop through every earthquake and get its latitude and longitude from our getters. We check to see if the earthquake’s latitude is in between our two given latitudes, and if it’s two longitudes is in between our two given longitudes. If it is then it adds it to the arraylist of matching earthquakes and returns it.

Searchbydate: Takes two string arguments and creates a simpledateformat that includes the date and time. We parse our string arguments into dates with the simpledateformat and add a time of 00:00:00. We then loop through every earthquake, grab it’s date and time substrings from gettime, and parse them into a date with the simpledateformat. We then use .before and .after on the earthquake’s date to see if it takes place between the two. If it does then we add it to the arraylist of matching earthquakes and return it.

getfileInformation: This method reads our csv file in and splits the data into each individual variable. It then puts each rows data into an earthquake object and adds it to the earthquake arraylist.