GUI Documentation

I started out having a bunch of ideas how to go about this project, but realised i should keep it simple ad those idea shrank rather fast.

The program uses the JSoup library to extract data from a website and displays it in the GUI. The information is from a website call coinmarketcap.com, and the gui displays the top 10 cryptocurrencies on the website, their name, price and market cap, and is sorted by market capitalization. I spent a long time learning about different libraries and ended up with JSoup because being able to extract and manipulate data from websites interests me.

This project had a couple of different forms and I used GitHub to keep track of all the test samples of code and different ideas I had through the process of learning JSoup and developing my program idea and code. I uploaded the Full program to Git as well for your viewing pleasure. I had trouble using Git in the last project so I thought i'd incorporate it into this project more for my own learning. The way I used it for this project is a little bit different to how I imagine it's meant to be used, but I wanted to document all of the test code and ideas as separate ideas in the master branch, so you could see my progress.

I learnt the bulk of how JSoup works through these websites.

- https://stackoverflow.com/questions/30154023/web-scraping-with-jsoup
- http://www.htmlgoodies.com/html5/other/web-page-scraping-with-jsoup.html
- https://aboullaite.me/jsoup-html-parser-tutorial-examples/
- https://stackoverflow.com/questions/14904776/parse-javascript-with-jsoup

The rest I just picked up from trial and error or specific questions on stack overflow.

Test Tables

The full code for all the test performed below can be found at can be found www.github.com/ChkRaizDaBTN/CoinMarket.

(Test file = Test code sample (1))

Question #1 - How do I set up a connection to a website using JSoup lib

Trigger -
```
Document doc = Jsoup.connect("https://coinmarketcap.com/").get();
String title = doc.title();
System.out.println("Title: " + title);
```

Area of concern - Connecting to a website

Reason for test - To figure out how to connect to a website

Expected Outcome - A connection will be made and the console will output the title of the website.

Actual Outcome - Ran as expected.

(Test file = Test code sample (1))

Question #2 - How to retrieve a pieces of data within the above website.

Trigger -
```
Elements name = doc.select("td[class=no-wrap currency-name]");
Elements price = doc.select("a[class=price]");

for (Element names : name) {
System.out.println(names.text());
}

for (Element prices : price) {
System.out.println(prices.text());
}
```

Area of concern - For my program to work i'm going to need to be able to scrape a website for certain pieces of information. This will be the first in a series of tests to find out how to do this function.

Reason for test - Learning/Basic functions for the program.

Expected Outcome - It should retrieve the information and i'm hoping it displays them individually.

Actual Outcome - It did retrieve the required information, unfortunately, it displayed them all as two separate strings. This will not work for the program.

(Test code sample (2))

Question #3 - How can I retrieve the names and prices of the coins from the website, and display those as individual elements so that they can be placed into a GUI for display. Using two coins we will test ways to find a solution

Trigger -

```
Elements name1 = doc.select("a[href=/currencies/bitcoin/]");
Elements name2 = doc.select("a[href=/currencies/ethereum/]");

Element price1 =
doc.select("a[href=/currencies/bitcoin/#markets]").first();
Element price2 =
doc.select("a[href=/currencies/ethereum/#markets]").first();

String str = name1.text() + " - " + price1.text();
String str1 = name2.text() + " - " + price2.text();

system.out.println(str + "\n" + str1);
```

Area of concern - Data retrieval, GUI Display.

Reason for test - We need to be able to split up the data we retrieve into separate lines or elements so we can display it vertically in the GUI and allocate its correct price next to it.

Expected Outcome - The elements should print the name with the price next to it, then then the next element beneath it with its price.

Actual Outcome - Ran as expected

(Test code sample (2))

Question #4 - Can we print the above results to a GUI using a JTextArea

Trigger -

```
JTextArea area = new JTextArea(str +"\n" + str1);
```

Area of concern - GUI Display

Reason for test - The gui is vital for the project, we have to be able to display the information retrieved, this will show us if we are on the right track with the development of the code.

Expected Outcome - The gui should display the above results

Actual Outcome - Ran as expected

(Test code sample (3) Using Strings)

Question #5	-	Will a large concatenation of Strings work in creating the entire app
Trigger	-	Code can be found in the github under - Test code sample (3) Using Strings (code is too large to display here)
Area of concern	-	GUI display and overall program creation
Reason for test	-	To see if this method will work in displaying the content for the gui.
Expected Outcome	-	Having tested this on a small scale previous, this should run as above but on a full scale.
Actual Outcome	-	Ran as expected

(Test code sample (4) Hashmap test)

Question #6	-	Can we turn the “Elements” we scrape from the website into strings to be able to use them for a Hashmap solution to the project?
Trigger	-	`String btcName = doc.select("a[href=/currencies/bitcoin/]").text();` `System.out.println(btcName);`
Area of concern	-	GUI Display
Reason for test	-	To add these Elements to a hashmap, turning them into strings is needed.
Expected Outcome	-	Print the name of the coin.
Actual Outcome	-	ran as expected

(Test code sample (4) Hashmap test)

Question #7 - Can we use a Hashmap to plot the content we retrieve. The above method used for displaying works and could be submitted for marking and would pass, but… it seems clunky and inefficient. (If this project was using Agile software development, the above would be the first released version, and this could be the second iteration) Also we would like to incorporate a refresh button you can click to update prices while in app. If prices were to changes and a different coin became more valuable the above list wouldn't change the order in which the coins are displaying, therefore generating false information. A hashmap could be the solution to this problem. Using the market capitalization as the key to the values could be a way to order the top ten coins.

Skipping over the more simple test of incorporating a hashmap for this project, the first test below is how do we display the hashmap on the gui. (The full code for the following tests can be found at the gihub and in the Hashmap test file)

Trigger -

```
for (Map.Entry<String, String> coin : map.entrySet()) {
        String key = coin.getKey();
        String value = coin.getValue();

JTextArea area = new JTextArea(key + " " + value);
}
```

Area of concern - GUI display

Reason for test - To find a way to print the retrieve content that has been placed in a

hashmap to the gui

Expected Outcome - Print the content to the gui through the JTextArea component, first the market cap, followed by the corresponding coin and price.

Actual Outcome - Failed to print all of the content, only printed the last coin, cap and price.

(Test code sample (4) Hashmap test)

Question #8 - A retest of the above question but this time using a StringBuilder to save all of the content.

Trigger -

```
StringBuilder sbCoins = new StringBuilder();

for (Map.Entry<String, String> coin : map.entrySet()) {
        String key = coin.getKey();
        String value = coin.getValue();

sbCoins.append(value + key + "\n");
}

JTextArea area = new JTextArea(sbCoins.toString());
```

Area of concern - GUI Display

Reason for test - A solution is needed to print all the content stored inside the hashmap, the above method did not deliver the desired results.

Expected Outcome - Print the content to the gui through the JTextArea component, first the market cap, followed by the corresponding coin and price.

Actual Outcome - Ran as expected

(Test code sample (4) Hashmap test)

Question #9 - How can we sort the hashmap results in order of market Capitalization.

Trigger -

```
public int compare(Integer o1, Integer o2) {
        int str1 = o1;
        int str2 = o2;

        if(str1 < str2) {
                return 1;
        }
        if(str1 > str2) {
                Return -1;
        }
        Return 0;
-------------------------------
Collections.sort(map);
```

Area of concern - GUI Display, sorting content for future implementation ofa refreshing function.

Reason for test - Creating a top 10 list requires the list to be in order, this is to order the list

Expected Outcome - Pre test I expected this to sort the map the key string lengths, but of course...

Actual Outcome - Did not run as expected, it did not compile, you can use this function for lists, but not hashmaps, meaning I could convert the keys to an array list, then sort them, the iterate through them. But that is a lot of process for a simple program, but of course, this helped develop the next step. Using a List for the entire project.

(Test code sample (5) Two classes using lists)

Question #10 - Using the string building and for loops above, we were able to create the entire project using an ArrayList instead of the hasmap structure. The full testing for this can be found at “Test code sample (5) Two classes using lists)”.

The code still didn't solve the sorting problem, it failed to refresh on request and still seemed clunky and very bulky with large concatenations of strings to get the GUI to display it correctly.

Full test tables of this version would result in all previous test tables combined with the difference of it being a list, therefore was deemed unnecessary to add. All code can be found at the above location.

Test code sample (6) optimising List

Question #11 - The next pieces of code looked to directly solve what Question 10 left us with, Solve the sort, trim the code down, less concatenations, more streamline. Once again if this were build using Agile this would be the next release. The full testing for this can be found at
"Test code sample (6) optimising lists)".

Still directly taking each each coin's name and price from the website, this code slimed down the lines by directly adding these elements as Strings to the list, we previously had done this in the Hashmap testing. This cut the code in half, made it seem less bulky and less clunky, a little more streamlined. But still no answer to how to sort the coins in order, we had abandoned the Market cap idea from the hashmap, but perhaps the market cap could be used as a way to sort the list, but just not added to it to sort it, rather how did the website sort the coins and could we capitalize (no pun intended) on that? More JSoup learning was required at this point.

(Main Program)

Question #12 - Can we find a way to retrieve the content and have it sorted correctly and if updated will change order according to the data?

Trigger -

```
Elements coins =
website.select("table[id=currencies]").select("td[class=no-wrap
currency-name]");

for (org.jsoup.nodes.Element coin : coins) {
        list.add(coin.text() + "\t");
}
```

Area of concern - GUI Display, optimisation

Reason for test - We need the list to change order according to the updated information, previous we had scraped the site and taken information from the object directly. We turned them into strings, added them to the list and iterated over them. This test will show us if we can capture the entire list of names for the coins and add them as elements to a list. This is the first a few steps in an attempt to get the program to refresh, sort and change order if need be.

Expected Outcome - The elements to be added to the list, using the .text() should turn them into strings, I should be able to then go and retrieve any element or print out the entire list and it should change according to market cap as this is how the website orders the coins and i am now taking the entire table rather than separate coins.

Actual Outcome - Ran as expected

(Main Program)

Question #13 - Can we add a refresh function that will refresh the app to update its prices to what the website is displaying, while not having to close and reopen the app.

Trigger -

```
button.addActionListener(new Refresh());

class Refresh implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                try {
                area.setText(site.getWebData());
                area.revalidate();
                area.repaint();
                } catch (IOException e1) {

System.out.println("Unable to connect to coinmarketcap.com");
        }
}
}
```

Area of concern - GUI Display, updating information

Reason for test - Does this refresh the program to the websites version, and does it run stable or crash the program.

Expected Outcome - When the website updates the information, this should also update when the button is clicked

Actual Outcome - Ran as expected. Since the website only updates every 5 minutes, the button appears to not work at times. But it does work, the website just needs to update in order for the button to work, the website isn't constantly tracking the prices.

Question #14 - When there is no internet connection, does the gui launch, prompting you there is no connection, and does the Retry button launch the full program once a connection has been established?

Trigger -

```
Gui gui = new Gui();
try {
        gui.gui();;
} catch (IOException e) {
        gui.guiNoInternet();

}

-------------------------------------
button.addActionListener(new Reconnect());

class Reconnect implements ActionListener {
public void actionPerformed(ActionEvent e) {
try {
        area.setText(site.getWebData());
        area.revalidate();
        area.repaint();
} catch (IOException e1) {
        area.setText("Unable to detect an internet connection");
        area.revalidate();
        area.repaint();
        }
}
}
```

Area of concern - Displaying the GUI

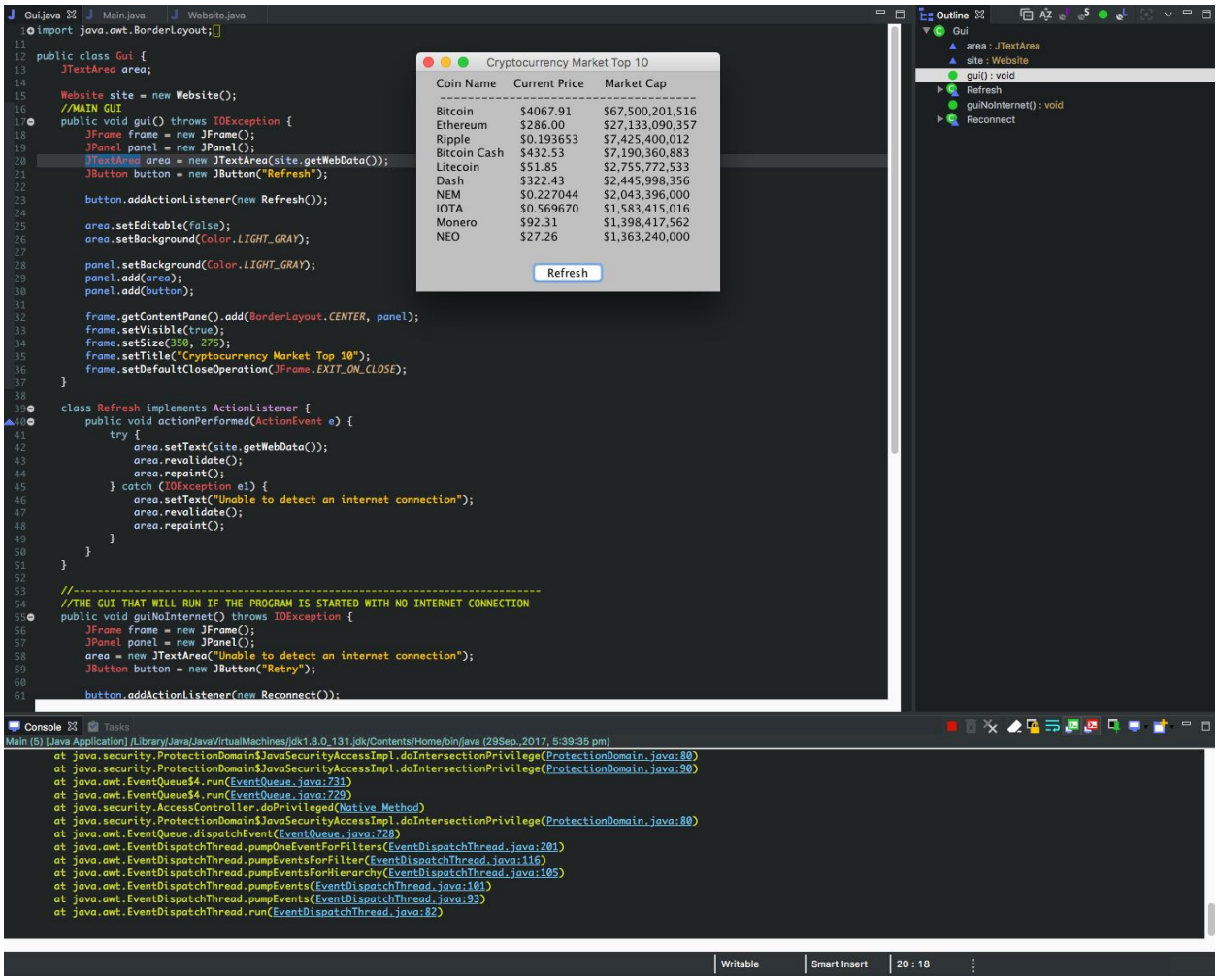
Reason for test - I want the GUI to still launch if there is no internet connection, but i want it to show that the issue is on the users end, so that they are alerted to this and can fix the issue.

Expected Outcome - The program should open, with the message dictated above, and when a connection is established and the button pressed the main program should launch

Actual Outcome - ran as expected

Debug

Debugging was a short process, the only true problem I had all project long was the fresh button. Everytime i clicked refresh the below imagine would happen.



I found out quick enough that this was due to the JTextArea being named outside of the Gui method and inside the Gui method at the same time. Once i removed the JTextArea from JTextArea area = new JTextArea(site.getWebData()); from inside the Gui method, the program ran as expected.