Student Name : Raja Naseer Ahmed Khan (G00351263)

Project Name : Word Cloud Generator

Module : Aritifical Inteligence, B.Sc. (Hons) 4th Year.

Lecturer : Dr. John Healy

Contents

Screenshot of Word Cloud Results

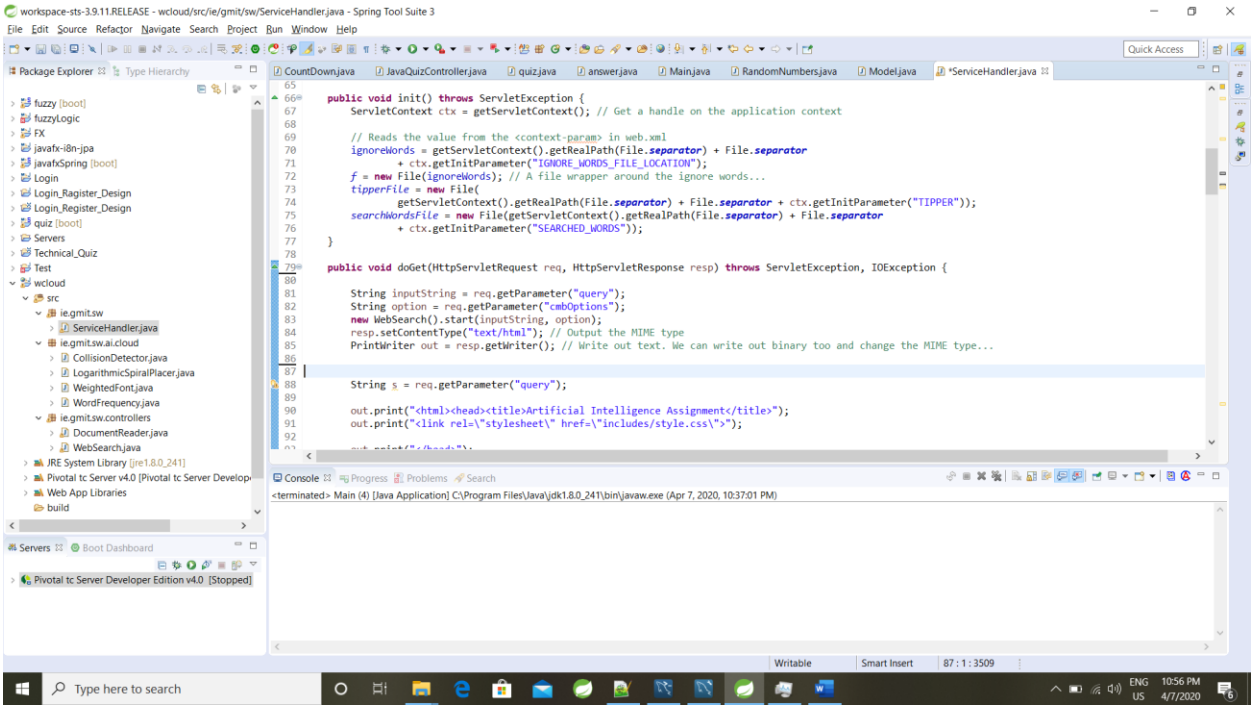
How we use:

- deploy the war file into tomcat directory (I used tomcat 9) .
- start the server by running startup command, then you can navigate into localhost:8080/your_war_file_name(in my case wcloud) and you will see the app running.
- The first drop down menu from the top is different web search providers, select anyone (in my case used Bing, as google and duck duck go keep blocking me).
- Enter the search term you want to search in the text box (I have searched the corona, America, Europe, England).
- After a while word cloud is generated (be patient might take time to heuristic search and scrap down the elements, making weights in fcl file getting result back as frequency then generating word cloud.

Detailed Features of Web App Methods:

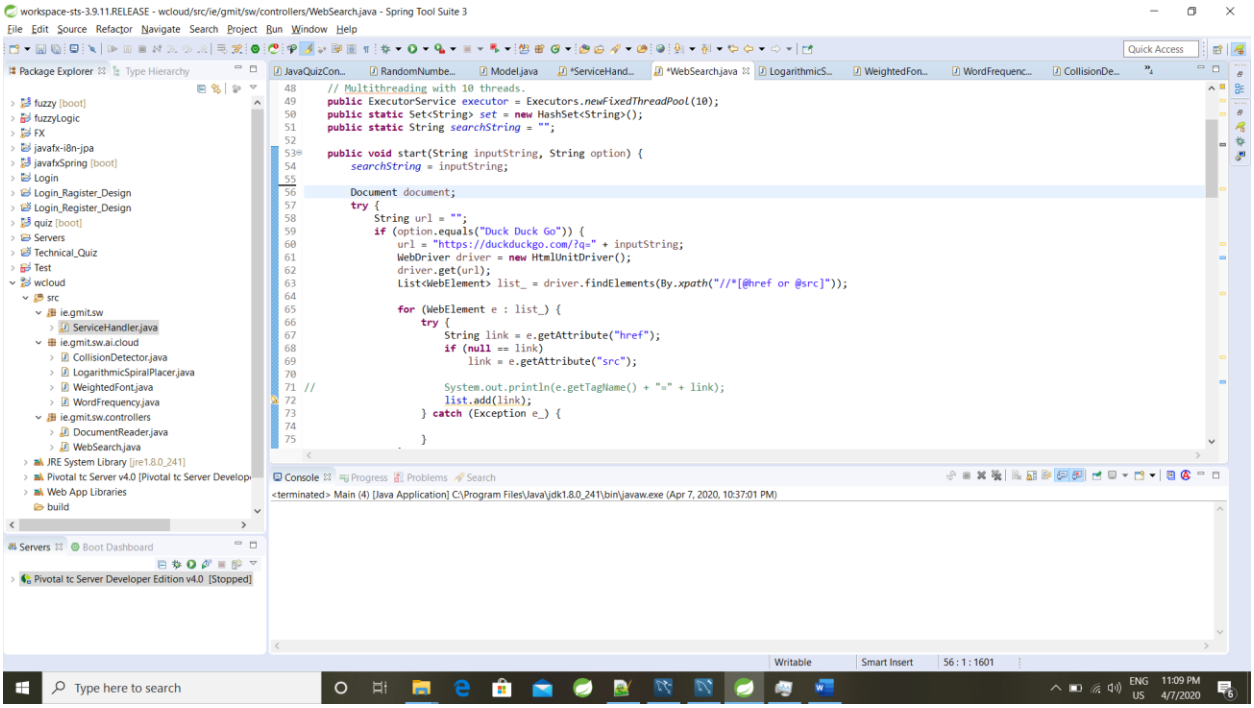
When we run our program index.html gets called and the first page is displayed. After that when we search our item and broweser and click on search we get redirected to this controller.

The program starts from init(). Here we are getting the path of ignore words file which is kept at WebContent ->res->ignorewords.txt. We find the path of it by passing the argument "`IGNORE_WORDS_FILE_LOCATION`" and the value of it is saved in **web.xml,** which is located at WebContent ->res ->WEB-INF -> web.xl.



Now doGet() is called for Servlet lifecycle. This method has two parameters, HttpRequest and HttpResponse. So whatever we typed in our search area and the browser type, we can get from the object of HttpResponse and save the search string in **inputString** and the browser value in **option** ..

After this we called the method start which is in Class **WebSearch** and pass two parameters inputString and option.

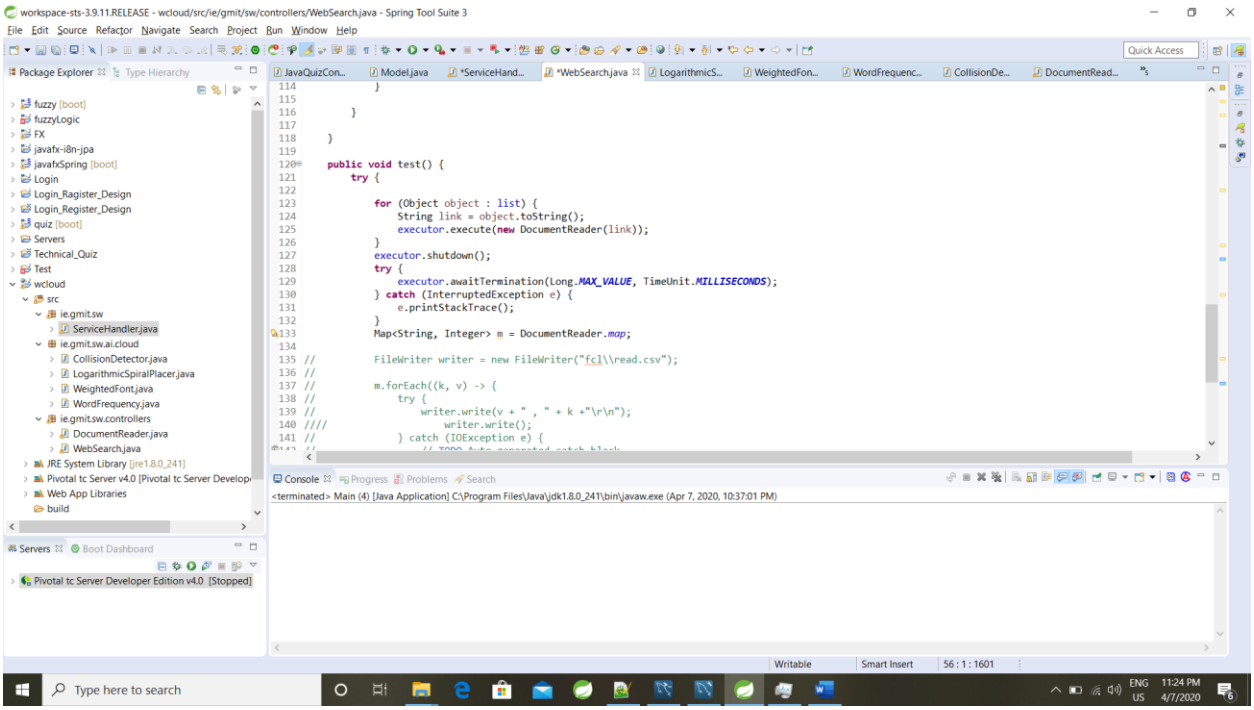


In this **start()** method, we get the twpo parameters passed. Now, we have if else condition to check what kind of browser we are using, if it is Duck Duck Go, we are using headerless Selenium (Selenium can get the values of elements which are dynamically loaded using javascript). Since Duck Duck Go data is loaded dynamically so we cannot use Jsoup because jsoup only fetches Static Content. So whatever **links(Href elements)** we get we put them in **ArrayList** named "list".

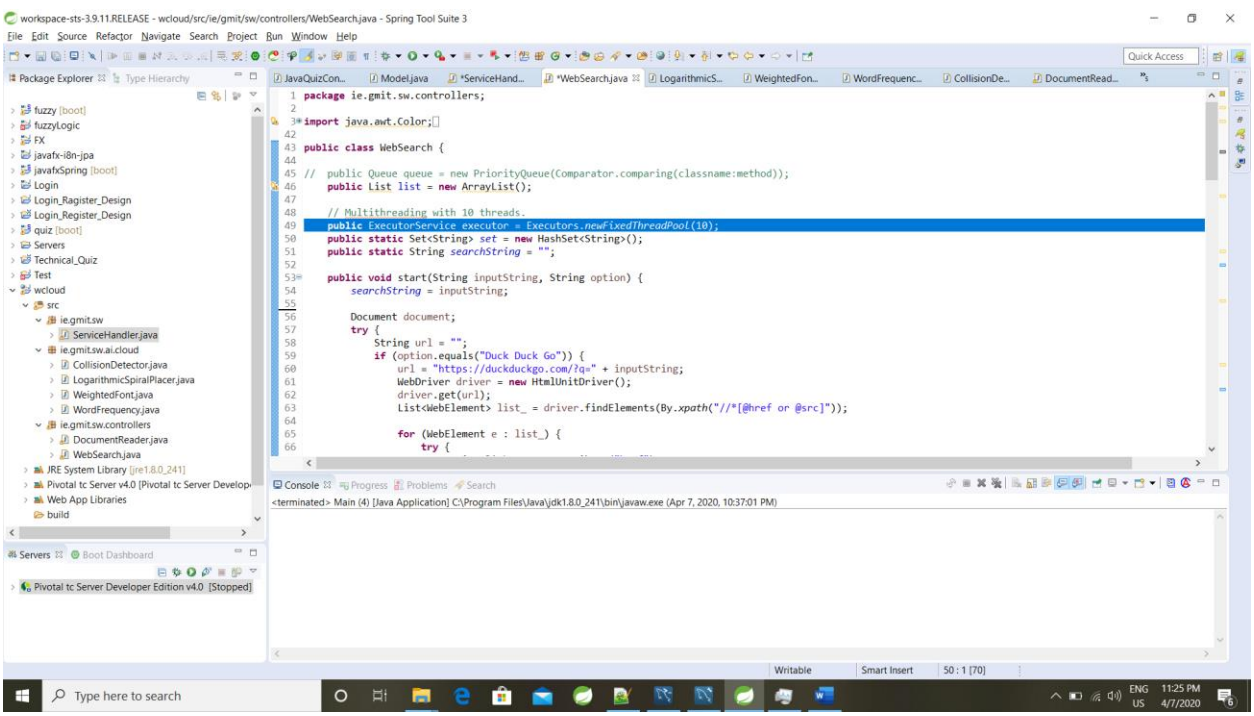
Similarly for other browsers (Google and Bing) use Jsoup to get the href elements and put them in "list".

Now when we have elements in "list", we call method **test().** In this method we have a for Each loop , i.e each url in list is fetched and using this line "`executor.execute(new DocumentReader(link));`" we call the run method in Document Reader. Executor.execute will call the run method and when we say new DocumentReader(link), this calls the constructor of DocumentReader(String link), **<u>executor</u> is an instance of ExecutorService ,** in this we pass the number of threads. This creates a thread pool of 10 threads and internally uses Blocking queue to assign the tasks to the threads, All the time 10 threads are active.

Test()



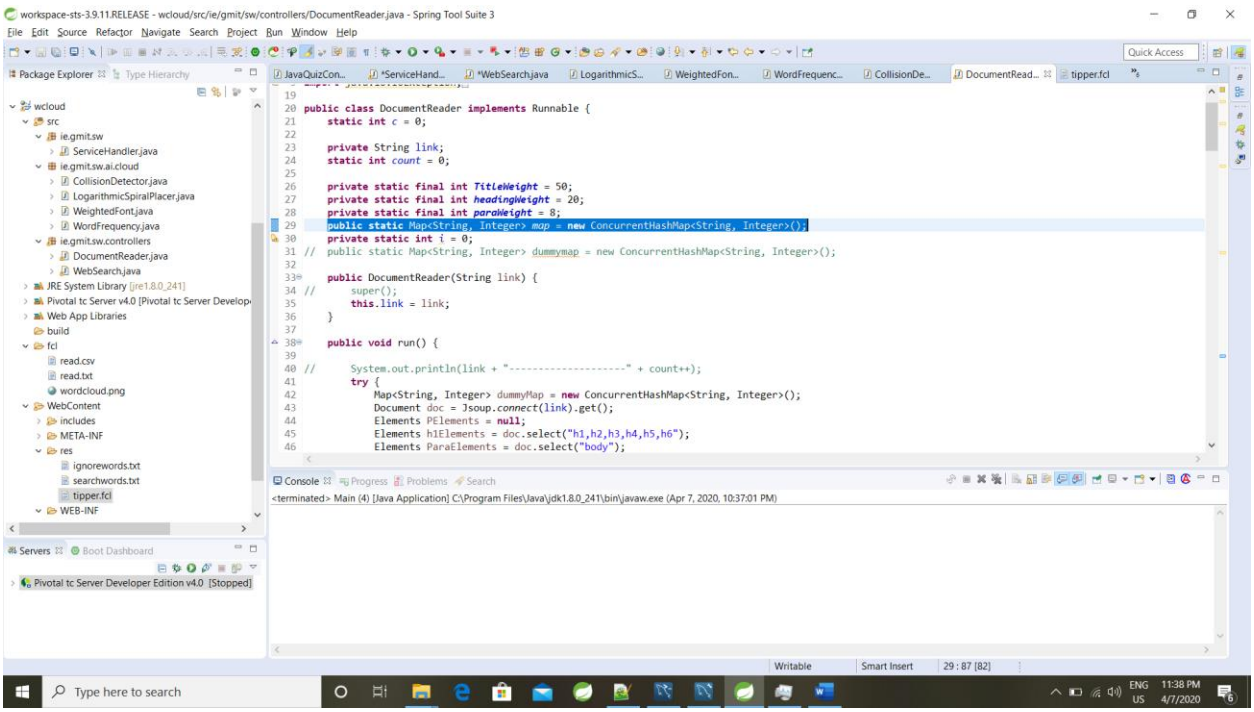
ExecutorsService taking argument as 10…



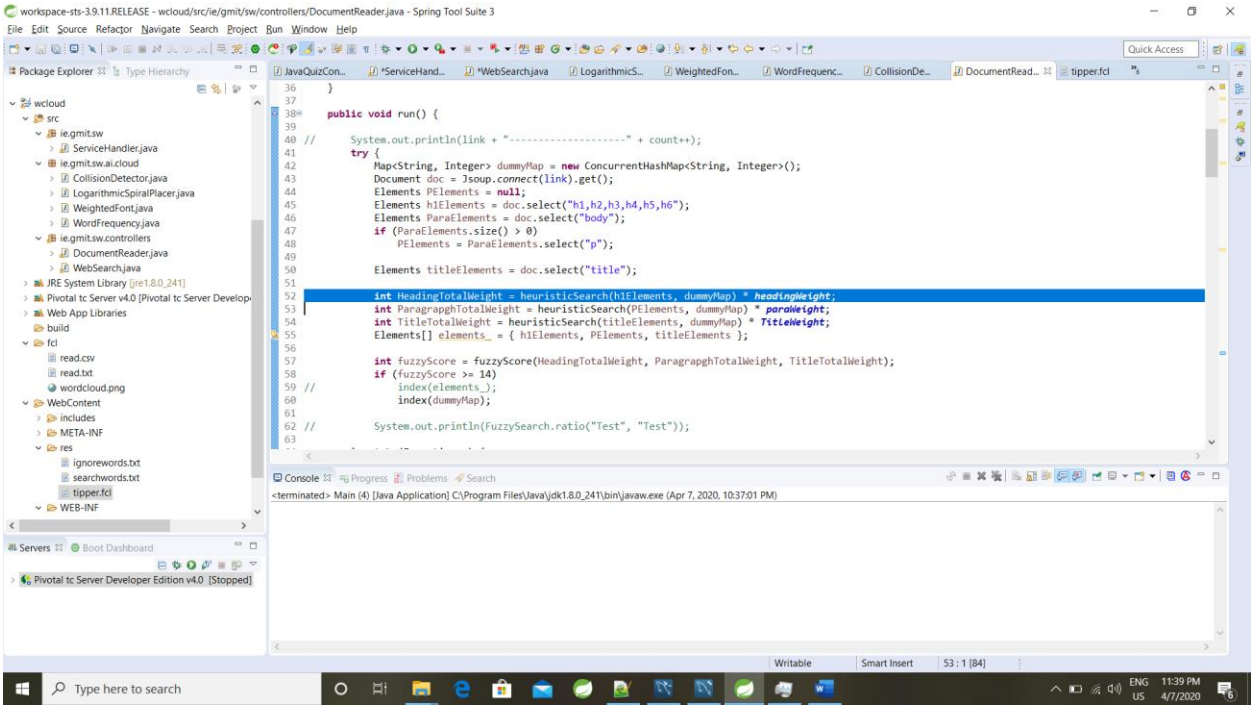
In run() we get Body elements, heading and title. For each of them we call a method heauristicSearch(), For example lets take body elements,In this method we compare each word in Body with searched word. If the word matches more than 0.75or 75 percent then we accept that word and increase the count in Body . Also we have a dummyMap object of type HashMap which has all the words in the Body element put into it with their count. Similarly these things happen with Title and Heading.

After this we multiple each of them with their respective weight and send them to a method `fuzzyScore() which takes 3 arguments , weights of heading,body and title. Here we calculate the fuzzy score, based on whether it is whether it is significant, relevant or insignificant. If score is greater than 15 then we accept the fuzzy score and we put the dummyMap data to “map” which is a type of ConcurrentHashMap.`

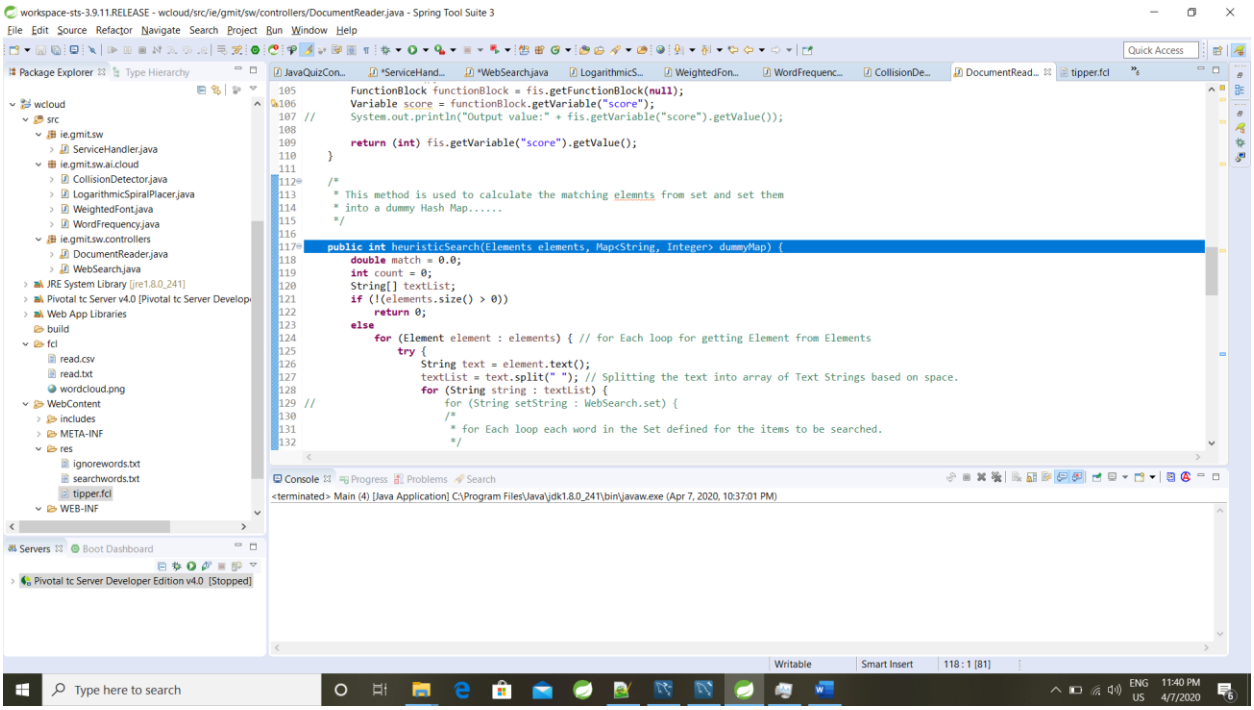
`“map” object.`



We call heuristic search method here .

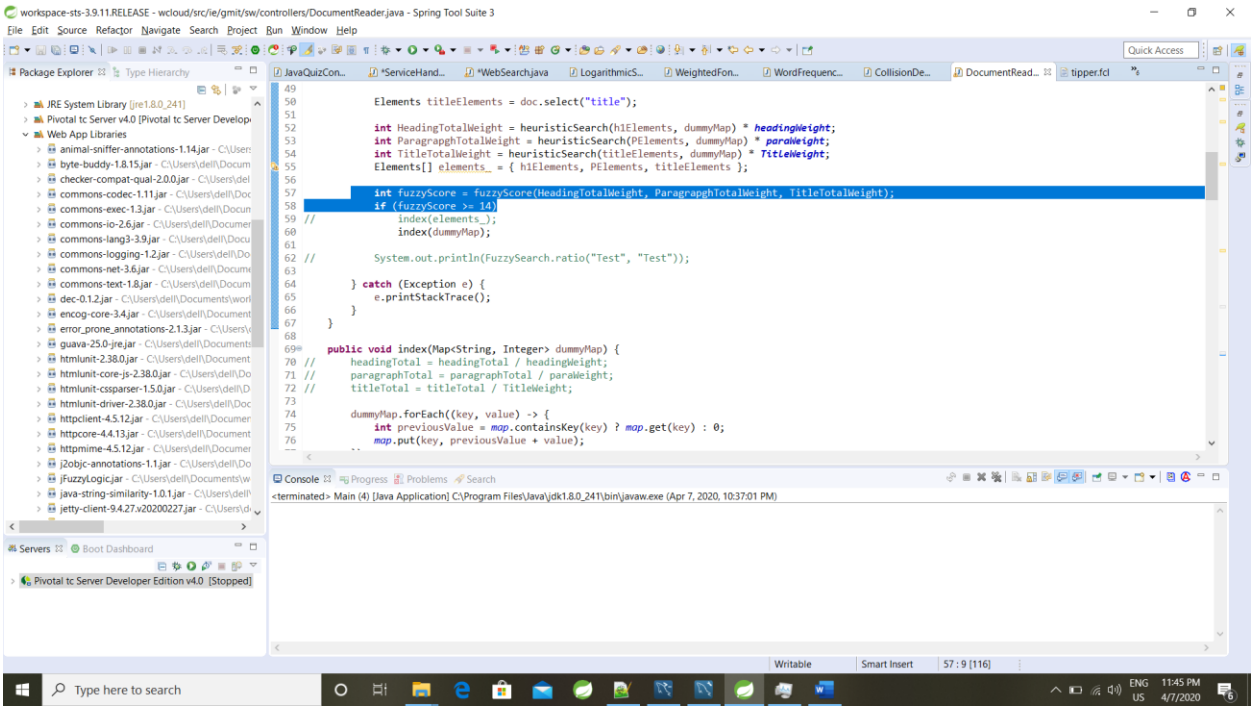


HeauristicSearch(). Takes two arguments, String in the element(bodyor title or heading) and dummymap. We use **Jaccard library** to get the matching percentage between word.

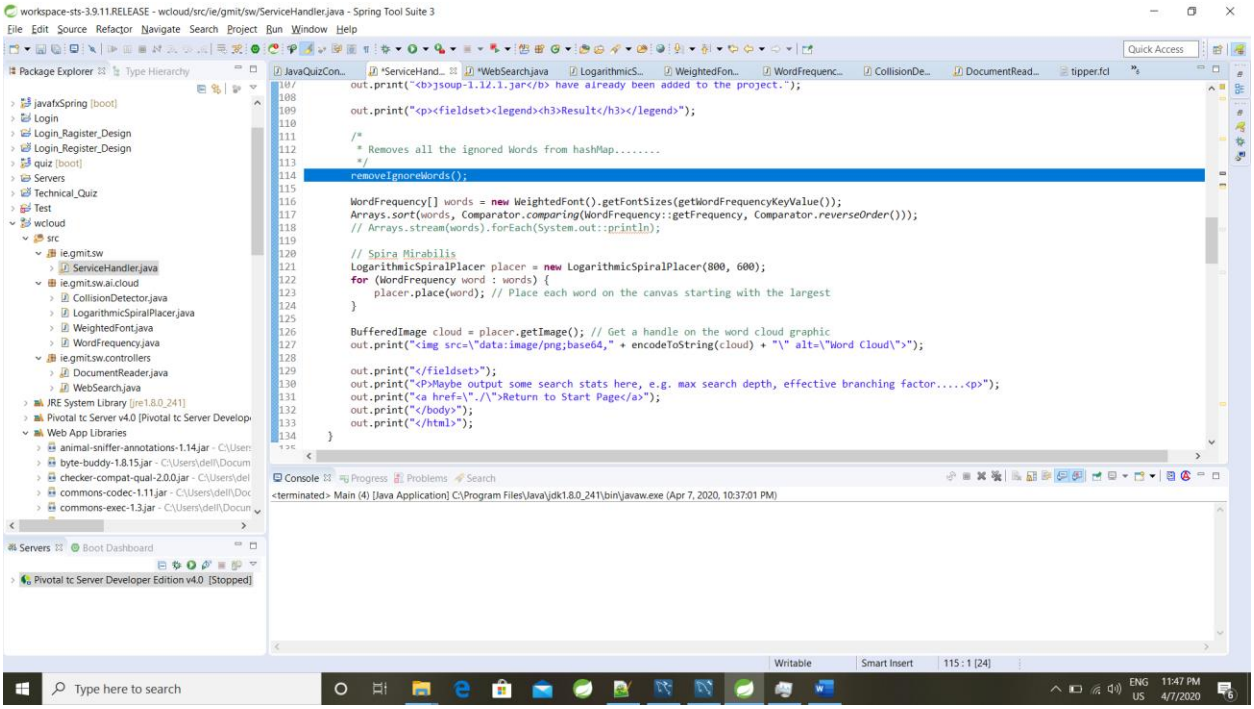


Method to calcuate fuzzy score and compare with greater than 14… for being relavant and significant….\

We call index () and put the values of dummyhashmap to map object.



Again we come back to Service Handler class and we call removeIgnorewords() to remove ignore words from the “map” and then we generate the word cloud..



Extra Features:

1. **Selenium**
   selenium is used to scrap the website which loads content dynamically.
   since duck duck go loads contents dynamically so jsoup cannot be used to retrieve the elements because jsoup only fetches static data. To get the data from duck duck go, we imported the library of selenium which get the dynamic contents. It scraps the data when the browser is loaded. The browser does not open in this case that is why no chrome or Mozilla can be seen.

   Link to Selenium Tutorial:
   https://www.guru99.com/selenium-with-htmlunit-driver-phantomjs.html

2. **Jaccard**
   Jaccard us used to compare two strings. It compares the strings and returns the percentage similarity between words. If the words are similar by more than 75% then we have considered it as matching word.

   Link to Jaccard:
   https://www.javadoc.io/doc/info.debatty/java-string-similarity/1.0.0/info/debatty/java/stringsimilarity/Jaccard.html

References:

- https://www.guru99.com/selenium-with-htmlunit-driver-phantomjs.html
- https://saucelabs.com/resources/articles/getting-started-with-webdriver-selenium-for-java-in-eclipse
- https://www.javadoc.io/doc/info.debatty/java-string-similarity/1.0.0/info/debatty/java/stringsimilarity/Jaccard.html
- https://www.youtube.com/watch?v=WzuJANOPLyQ
- https://www.youtube.com/watch?v=oo8hakhidQM
- https://stackoverflow.com/questions/43327446/how-to-run-fuzzy-control-language
- https://www.researchgate.net/publication/261576997_jFuzzyLogic_A_Java_Library_to_Design_Fuzzy_Logic_Controllers_According_to_the_Standard_for_Fuzzy_Control_Programming/figures?lo=1
- https://www.programcreek.com/java-api-examples/?class=net.sourceforge.jFuzzyLogic.FIS&method=load