

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	2
How we use:.....	2
Detailed Features of Web App Methods:	3
Extra Features:	11
References:	11

Screenshot of Word Cloud Results



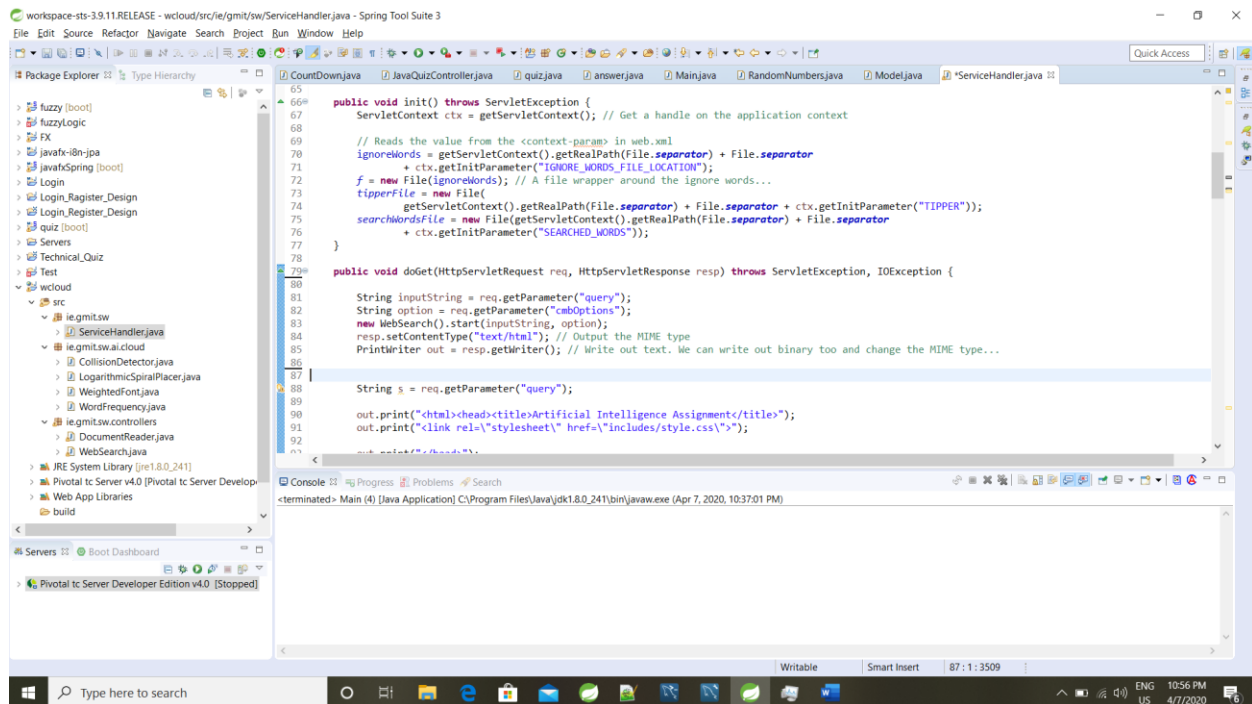
How we use:

- 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:

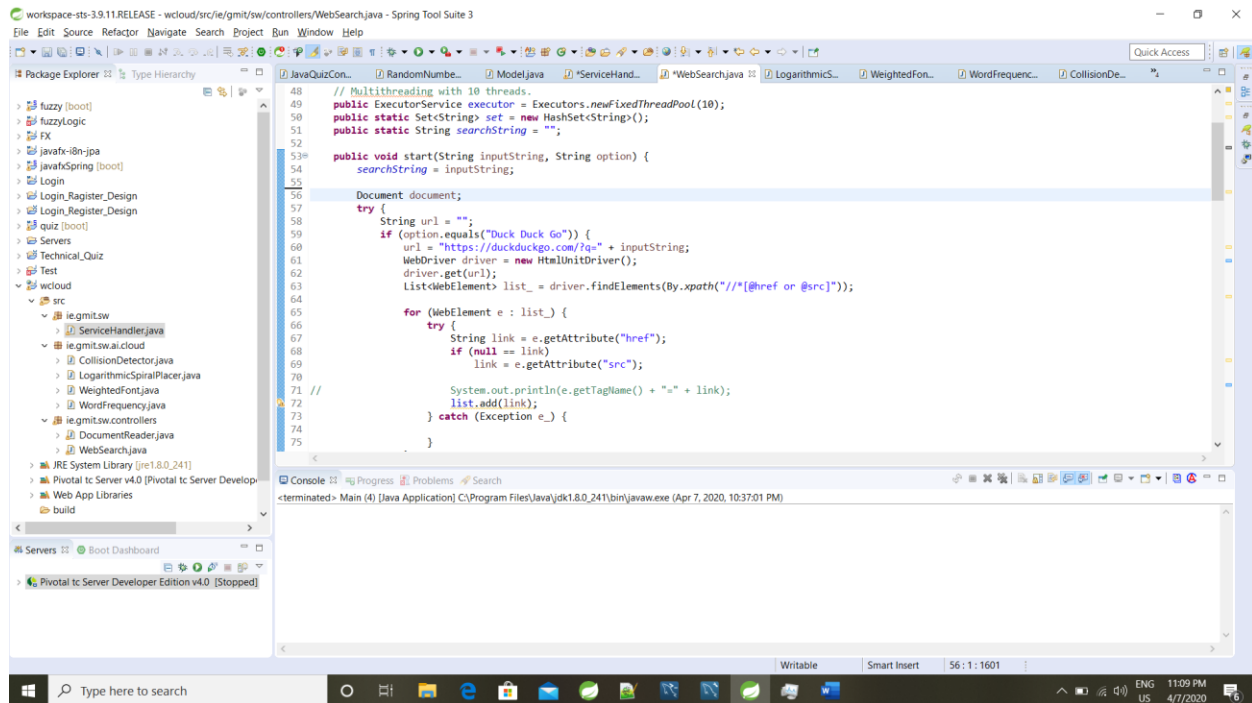
When we run our program index.html gets called and the first page is displayed. After that when we search our item and browser and click on search we get redirected to this controller.

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.xml.



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 two 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), **executor 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.

The screenshot displays an IDE window with the following components:

- Top Bar:** workspace-sts-3.9.11.RELEASE - wcloud/src/ie/gmit/sw/controllers/WebSearch.java - Spring Tool Suite 3
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard IDE icons for file operations, editing, and running.
- Package Explorer:** Shows a project structure with folders like 'fuzzy', 'Login_Register_Design', 'Servers', 'Test', and 'wcloud'. The 'wcloud' folder is expanded, showing 'src' and 'ie.gmit.sw' subfolders.
- Editor:** Displays the code for 'WebSearch.java'. The code includes a 'test()' method that uses an 'ExecutorService' to execute a 'DocumentReader' and write data to a CSV file. The code is as follows:

```
114 }
115
116 }
117
118 }
119
120 public void test() {
121     try {
122
123         for (Object object : list) {
124             String link = object.toString();
125             executor.execute(new DocumentReader(link));
126         }
127         executor.shutdown();
128         try {
129             executor.awaitTermination(Long.MAX_VALUE, TimeUnit.MILLISECONDS);
130         } catch (InterruptedException e) {
131             e.printStackTrace();
132         }
133         Map<String, Integer> m = DocumentReader.mmp;
134
135         FileWriter writer = new FileWriter("fcl\\read.csv");
136         //
137         //
138         //
139         //
140         //
141         //
142         //
143         //
144         //
145         //
146         //
147         //
148         //
149         //
150         //
151         //
152         //
153         //
154         //
155         //
156         //
157         //
158         //
159         //
160         //
161         //
162         //
163         //
164         //
165         //
166         //
167         //
168         //
169         //
170         //
171         //
172         //
173         //
174         //
175         //
176         //
177         //
178         //
179         //
180         //
181         //
182         //
183         //
184         //
185         //
186         //
187         //
188         //
189         //
190         //
191         //
192         //
193         //
194         //
195         //
196         //
197         //
198         //
199         //
200         //
201         //
202         //
203         //
204         //
205         //
206         //
207         //
208         //
209         //
210         //
211         //
212         //
213         //
214         //
215         //
216         //
217         //
218         //
219         //
220         //
221         //
222         //
223         //
224         //
225         //
226         //
227         //
228         //
229         //
230         //
231         //
232         //
233         //
234         //
235         //
236         //
237         //
238         //
239         //
240         //
241         //
242         //
243         //
244         //
245         //
246         //
247         //
248         //
249         //
250         //
251         //
252         //
253         //
254         //
255         //
256         //
257         //
258         //
259         //
260         //
261         //
262         //
263         //
264         //
265         //
266         //
267         //
268         //
269         //
270         //
271         //
272         //
273         //
274         //
275         //
276         //
277         //
278         //
279         //
280         //
281         //
282         //
283         //
284         //
285         //
286         //
287         //
288         //
289         //
290         //
291         //
292         //
293         //
294         //
295         //
296         //
297         //
298         //
299         //
300         //
301         //
302         //
303         //
304         //
305         //
306         //
307         //
308         //
309         //
310         //
311         //
312         //
313         //
314         //
315         //
316         //
317         //
318         //
319         //
320         //
321         //
322         //
323         //
324         //
325         //
326         //
327         //
328         //
329         //
330         //
331         //
332         //
333         //
334         //
335         //
336         //
337         //
338         //
339         //
340         //
341         //
342         //
343         //
344         //
345         //
346         //
347         //
348         //
349         //
350         //
351         //
352         //
353         //
354         //
355         //
356         //
357         //
358         //
359         //
360         //
361         //
362         //
363         //
364         //
365         //
366         //
367         //
368         //
369         //
370         //
371         //
372         //
373         //
374         //
375         //
376         //
377         //
378         //
379         //
380         //
381         //
382         //
383         //
384         //
385         //
386         //
387         //
388         //
389         //
390         //
391         //
392         //
393         //
394         //
395         //
396         //
397         //
398         //
399         //
400         //
401         //
402         //
403         //
404         //
405         //
406         //
407         //
408         //
409         //
410         //
411         //
412         //
413         //
414         //
415         //
416         //
417         //
418         //
419         //
420         //
421         //
422         //
423         //
424         //
425         //
426         //
427         //
428         //
429         //
430         //
431         //
432         //
433         //
434         //
435         //
436         //
437         //
438         //
439         //
440         //
441         //
442         //
443         //
444         //
445         //
446         //
447         //
448         //
449         //
450         //
451         //
452         //
453         //
454         //
455         //
456         //
457         //
458         //
459         //
460         //
461         //
462         //
463         //
464         //
465         //
466         //
467         //
468         //
469         //
470         //
471         //
472         //
473         //
474         //
475         //
476         //
477         //
478         //
479         //
480         //
481         //
482         //
483         //
484         //
485         //
486         //
487         //
488         //
489         //
490         //
491         //
492         //
493         //
494         //
495         //
496         //
497         //
498         //
499         //
500         //
501         //
502         //
503         //
504         //
505         //
506         //
507         //
508         //
509         //
510         //
511         //
512         //
513         //
514         //
515         //
516         //
517         //
518         //
519         //
520         //
521         //
522         //
523         //
524         //
525         //
526         //
527         //
528         //
529         //
530         //
531         //
532         //
533         //
534         //
535         //
536         //
537         //
538         //
539         //
540         //
541         //
542         //
543         //
544         //
545         //
546         //
547         //
548         //
549         //
550         //
551         //
552         //
553         //
554         //
555         //
556         //
557         //
558         //
559         //
560         //
561         //
562         //
563         //
564         //
565         //
566         //
567         //
568         //
569         //
570         //
571         //
572         //
573         //
574         //
575         //
576         //
577         //
578         //
579         //
580         //
581         //
582         //
583         //
584         //
585         //
586         //
587         //
588         //
589         //
590         //
591         //
592         //
593         //
594         //
595         //
596         //
597         //
598         //
599         //
600         //
601         //
602         //
603         //
604         //
605         //
606         //
607         //
608         //
609         //
610         //
611         //
612         //
613         //
614         //
615         //
616         //
617         //
618         //
619         //
620         //
621         //
622         //
623         //
624         //
625         //
626         //
627         //
628         //
629         //
630         //
631         //
632         //
633         //
634         //
635         //
636         //
637         //
638         //
639         //
640         //
641         //
642         //
643         //
644         //
645         //
646         //
647         //
648         //
649         //
650         //
651         //
652         //
653         //
654         //
655         //
656         //
657         //
658         //
659         //
660         //
661         //
662         //
663         //
664         //
665         //
666         //
667         //
668         //
669         //
670         //
671         //
672         //
673         //
674         //
675         //
676         //
677         //
678         //
679         //
680         //
681         //
682         //
683         //
684         //
685         //
686         //
687         //
688         //
689         //
690         //
691         //
692         //
693         //
694         //
695         //
696         //
697         //
698         //
699         //
700         //
701         //
702         //
703         //
704         //
705         //
706         //
707         //
708         //
709         //
710         //
711         //
712         //
713         //
714         //
715         //
716         //
717         //
718         //
719         //
720         //
721         //
722         //
723         //
724         //
725         //
726         //
727         //
728         //
729         //
730         //
731         //
732         //
733         //
734         //
735         //
736         //
737         //
738         //
739         //
740         //
741         //
742         //
743
```

The screenshot shows an IDE with a Java project. The left sidebar displays the Project Explorer with a tree structure of folders and files. The main editor window shows the code for `WebSearch.java`. The code defines a `WebSearch` class with a `start` method that takes an `inputString` and an `option`, and a `findElements` method that uses Selenium WebDriver to find elements on a web page.

```

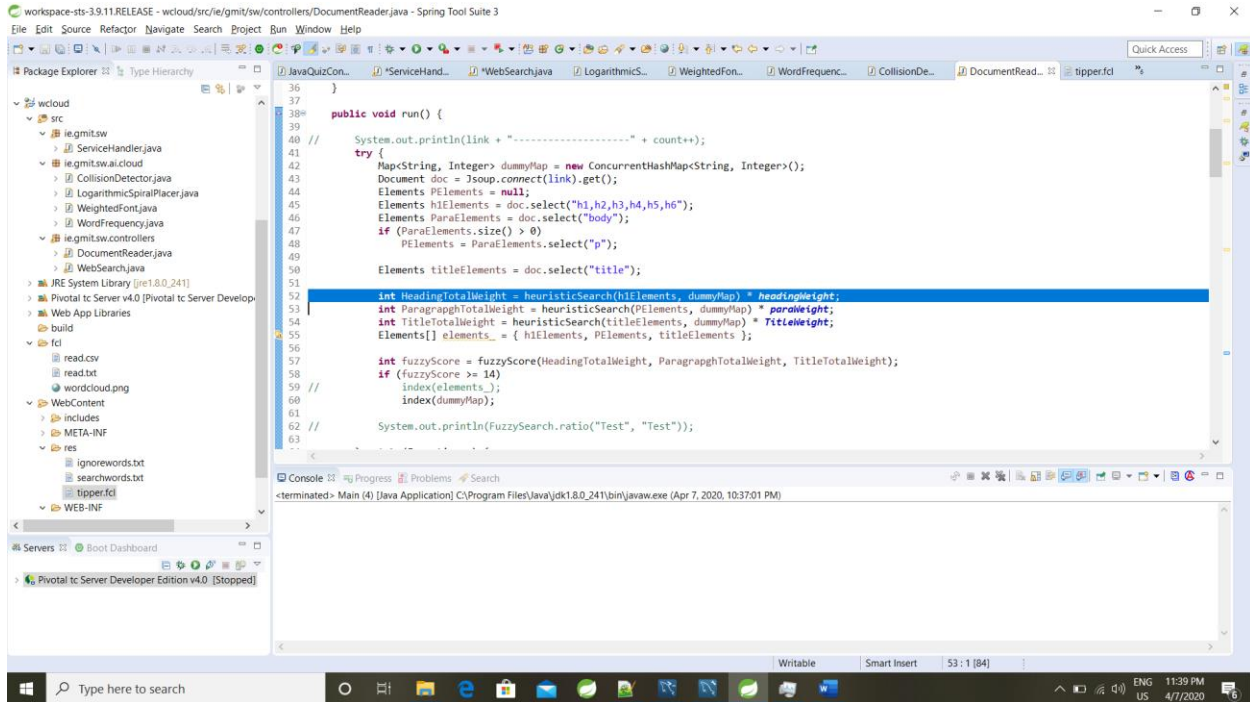
1 package ie.gmit.sw.controllers;
2
3 import java.net.*;
4
5 public class WebSearch {
6
7     // public Queue queue = new PriorityQueue<Comparator.comparing(classname:method)>;
8     public List list = new ArrayList<>();
9
10    // Multithreading with 10 threads.
11    public ExecutorService executorService = Executors.newFixedThreadPool(10);
12    public static Set<String> set = new HashSet<String>();
13    public static String searchString = "";
14
15    public void start(String inputString, String option) {
16        searchString = inputString;
17
18        Document document;
19        try {
20            String url = "";
21            if (option.equals("Duck Duck Go")) {
22                url = "https://duckduckgo.com/?q=" + inputString;
23                WebDriver driver = new HtmlUnitDriver();
24                driver.get(url);
25                List<WebElement> list = driver.findElements(By.xpath("//*[@href or @src]"));
26
27                for (WebElement e : list) {
28
29                }
30            }
31        } catch (Exception e) {
32            e.printStackTrace();
33        }
34    }
35
36    // ...
37
38    // ...
39
40    // ...
41
42    // ...
43
44    // ...
45
46    // ...
47
48    // ...
49
50    // ...
51
52    // ...
53
54    // ...
55
56    // ...
57
58    // ...
59
60    // ...
61
62    // ...
63
64    // ...
65
66    // ...
67
68    // ...
69
70    // ...
71
72    // ...
73
74    // ...
75
76    // ...
77
78    // ...
79
80    // ...
81
82    // ...
83
84    // ...
85
86    // ...
87
88    // ...
89
90    // ...
91
92    // ...
93
94    // ...
95
96    // ...
97
98    // ...
99
100   // ...
101
102   // ...
103
104   // ...
105
106   // ...
107
108   // ...
109
110   // ...
111
112   // ...
113
114   // ...
115
116   // ...
117
118   // ...
119
120   // ...
121
122   // ...
123
124   // ...
125
126   // ...
127
128   // ...
129
130   // ...
131
132   // ...
133
134   // ...
135
136   // ...
137
138   // ...
139
140   // ...
141
142   // ...
143
144   // ...
145
146   // ...
147
148   // ...
149
150   // ...
151
152   // ...
153
154   // ...
155
156   // ...
157
158   // ...
159
160   // ...
161
162   // ...
163
164   // ...
165
166   // ...
167
168   // ...
169
170   // ...
171
172   // ...
173
174   // ...
175
176   // ...
177
178   // ...
179
180   // ...
181
182   // ...
183
184   // ...
185
186   // ...
187
188   // ...
189
190   // ...
191
192   // ...
193
194   // ...
195
196   // ...
197
198   // ...
199
200   // ...
201
202   // ...
203
204   // ...
205
206   // ...
207
208   // ...
209
210   // ...
211
212   // ...
213
214   // ...
215
216   // ...
217
218   // ...
219
220   // ...
221
222   // ...
223
224   // ...
225
226   // ...
227
228   // ...
229
230   // ...
231
232   // ...
233
234   // ...
235
236   // ...
237
238   // ...
239
240   // ...
241
242   // ...
243
244   // ...
245
246   // ...
247
248   // ...
249
250   // ...
251
252   // ...
253
254   // ...
255
256   // ...
257
258   // ...
259
260   // ...
261
262   // ...
263
264   // ...
265
266   // ...
267
268   // ...
269
270   // ...
271
272   // ...
273
274   // ...
275
276   // ...
277
278   // ...
279
280   // ...
281
282   // ...
283
284   // ...
285
286   // ...
287
288   // ...
289
290   // ...
291
292   // ...
293
294   // ...
295
296   // ...
297
298   // ...
299
300   // ...
301
302   // ...
303
304   // ...
305
306   // ...
307
308   // ...
309
310   // ...
311
312   // ...
313
314   // ...
315
316   // ...
317
318   // ...
319
320   // ...
321
322   // ...
323
324   // ...
325
326   // ...
327
328   // ...
329
330   // ...
331
332   // ...
333
334   // ...
335
336   // ...
337
338   // ...
339
340   // ...
341
342   // ...
343
344   // ...
345
346   // ...
347
348   // ...
349
350   // ...
351
352   // ...
353
354   // ...
355
356   // ...
357
358   // ...
359
360   // ...
361
362   // ...
363
364   // ...
365
366   // ...
367
368   // ...
369
370   // ...
371
372   // ...
373
374   // ...
375
376   // ...
377
378   // ...
379
380   // ...
381
382   // ...
383
384   // ...
385
386   // ...
387
388   // ...
389
390   // ...
391
392   // ...
393
394   // ...
395
396   // ...
397
398   // ...
399
400   // ...
401
402   // ...
403
404   // ...
405
406   // ...
407
408   // ...
409
410   // ...
411
412   // ...
413
414   // ...
415
416   // ...
417
418   // ...
419
420   // ...
421
422   // ...
423
424   // ...
425
426   // ...
427
428   // ...
429
430   // ...
431
432   // ...
433
434   // ...
435
436   // ...
437
438   // ...
439
440   // ...
441
442   // ...
443
444   // ...
445
446   // ...
447
448   // ...
449
450   // ...
451
452   // ...
453
454   // ...
455
456   // ...
457
458   // ...
459
460   // ...
461
462   // ...
463
464   // ...
465
466   // ...
467
468   // ...
469
470   // ...
471
472   // ...
473
474   // ...
475
476   // ...
477
478   // ...
479
480   // ...
481
482   // ...
483
484   // ...
485
486   // ...
487
488   // ...
489
490   // ...
491
492   // ...
493
494   // ...
495
496   // ...
497
498   // ...
499
500   // ...
501
502   // ...
503
504   // ...
505
506   // ...
507
508   // ...
509
510   // ...
511
512   // ...
513
514   // ...
515
516   // ...
517
518   // ...
519
520   // ...
521
522   // ...
523
524   // ...
525
526   // ...
527
528   // ...
529
530   // ...
531
532   // ...
533
534   // ...
535
536   // ...
537
538   // ...
539
540   // ...
541
542   // ...
543
544   // ...
545
546   // ...
547
548   // ...
549
550   // ...
551
552   // ...
553
554   // ...
555
556   // ...
557
558   // ...
559
560   // ...
561
562   // ...
563
564   // ...
565
566   // ...
567
568   // ...
569
570   // ...
571
572   // ...
573
574   // ...
575
576   // ...
577
578   // ...
579
580   // ...
581
582   // ...
583
584   // ...
585
586   // ...
587
588   // ...
589
590   // ...
591
592   // ...
593
594   // ...
595
596   // ...
597
598   // ...
599
600   // ...
601
602   // ...
603
604   // ...
605
606   // ...
607
608   // ...
609
610   // ...
611
612   // ...
613
614   // ...
615
616   // ...
617
618   // ...
619
620   // ...
621
622   // ...
623
624   // ...
625
626   // ...
627
628   // ...
629
630   // ...
631
632   // ...
633
634   // ...
635
636   // ...
637
638   // ...
639
640   // ...
641
642   // ...
643
644   // ...
645
646   // ...
647
648   // ...
649
650   // ...
651
652   // ...
653
654   // ...
655
656   // ...
657
658   // ...
659
660   // ...
661
662   // ...
663
664   // ...
665
666   // ...
667
668   // ...
669
670   // ...
671
672   // ...
673
674   // ...
675
676   // ...
677
678   // ...
679
680   // ...
681
682   // ...
683
684   // ...
685
686   // ...
687
6
```

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`.

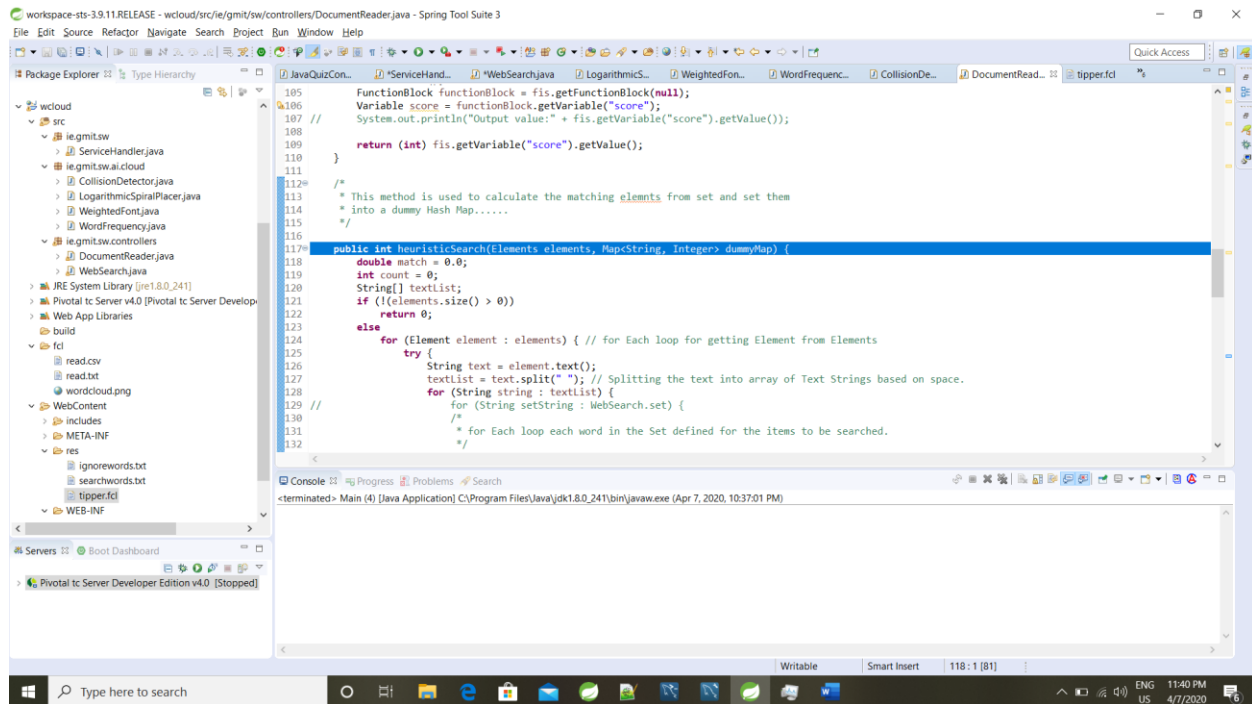
The screenshot displays an IDE with the following components:

- Package Explorer (Left):** Shows the project structure. The 'controllers' package is expanded, showing 'DocumentReader.java'.
- Code Editor (Center):** Displays the source code of 'DocumentReader.java'. The code implements the 'Runnable' interface and uses 'ConcurrentHashMap' for thread-safe map operations. The line numbers 19 through 46 are visible.
- Console (Bottom):** Shows the output of the application, indicating it has terminated successfully.

We call heuristic search method here .



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



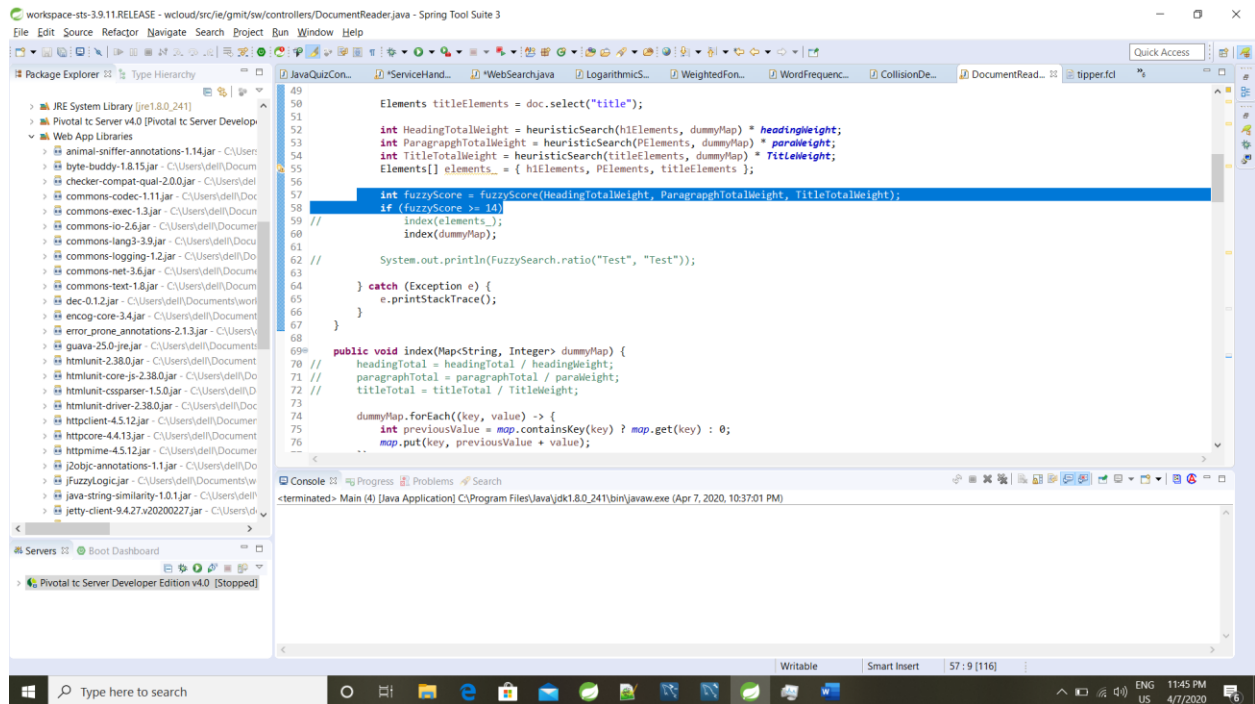
The screenshot shows an IDE window with the following components:

- Package Explorer:** Displays the project structure with folders like 'src', 'ie.gmit.sw', 'ie.gmit.sw.ai.cloud', 'ie.gmit.sw.controllers', and 'ie.gmit.sw.web'. The 'ie.gmit.sw.controllers' folder is expanded, showing 'DocumentReader.java'.
- Editor:** Displays the code for the 'HeuristicSearch' method in 'DocumentReader.java'. The code is as follows:

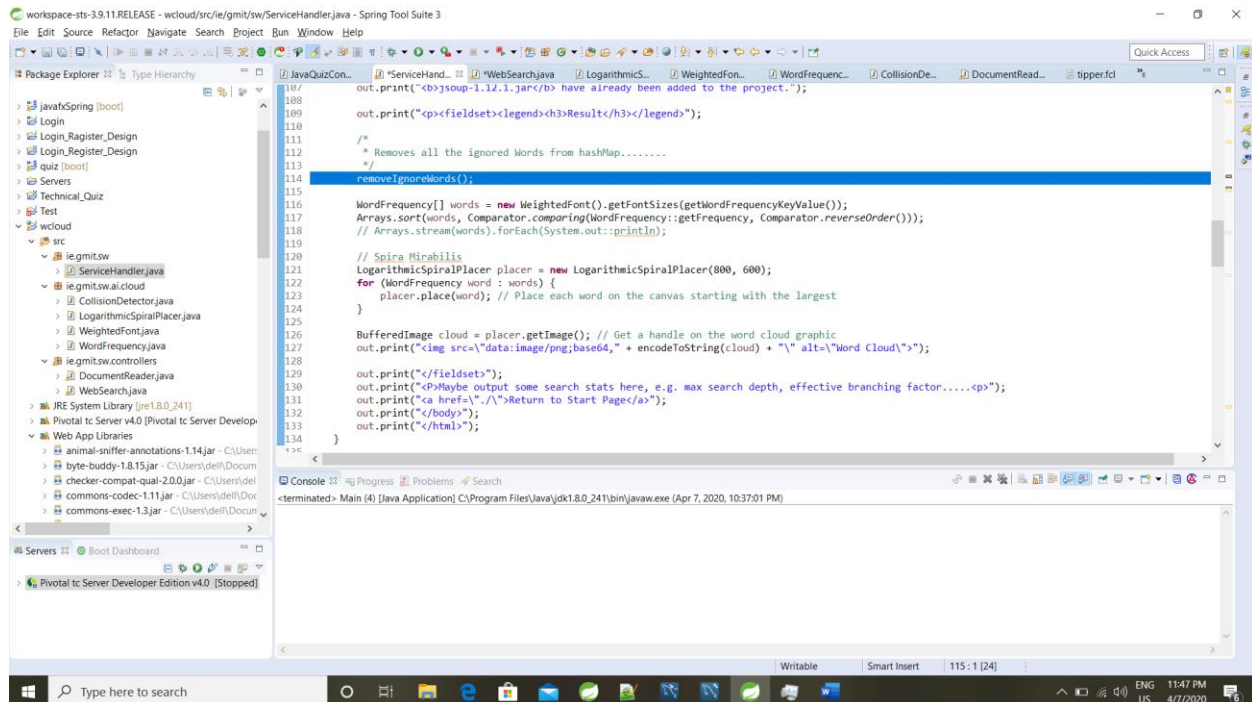
```
1120  /*  
1121   * This method is used to calculate the matching elements from set and set them  
1122   * into a dummy Hash Map.....  
1123   */  
1124  public int heuristicSearch(Element element, Map<String, Integer> dummyMap) {  
1125      double match = 0.0;  
1126      int count = 0;  
1127      String[] textList;  
1128      if (!element.isEmpty()) {  
1129          return 0;  
1130      }  
1131      for (Element element : elements) { // for Each loop for getting Element from Elements  
1132          try {  
1133              String text = element.getText();  
1134              textList = text.split(" "); // Splitting the text into array of Text Strings based on space.  
1135              for (String string : textList) {  
1136                  for (String setString : WebSearch.set) {  
1137                      /*  
1138                       * for Each loop each word in the Set defined for the items to be searched.  
1139                       */  
1140                  }  
1141              }  
1142          }  
1143      }  
1144  }
```
- Console:** Shows the message '<terminated> Main (4) [Java Application] C:\Program Files\Java\jdk1.8.0_241\bin\javaw.exe (Apr 7, 2020, 10:37:01 PM)'.
- Taskbar:** Shows the Windows taskbar with the search bar and various application icons.

Method to calculate fuzzy score and compare with greater than 14... for being relevant 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>