**Program Documentation: Jai-Kishan Timmapatruni**

**- EE36052  
  
Summary:**

Here we have created a Maven project, where the main purpose is to take the following 503 HTML files as input and produce the output of tokens for each HTML, total no of tokens sorted by frequency, total no of tokens sorted by tokens, the first fifty and last fifty tokens in tokens sorted by tokens and the first fifty and last fifty tokens in tokens sorted by frequency.

**The Approach for Tokenization:**

To use different libraries of Java I had to create a Maven project which helped utilize the Jsoup library to parse in the HTML documents and get the content from them. After getting the content we tried to check if the head and body content in the HTML files are not null then we can try to combine the following data into a string by making use of the functions we have like document. body().text() and document. head().text() (Note: Here .text() has helped us convert the following content to string.). If the body is null then we only try to take the content of the head tag in the HTML file. I have created the output files with the name of the respective input files so that we can understand that the following tokens belong to the following input html file.

After getting the content in the format of string by making use of the StringTokenizer() we can go through each token present in the string try to convert it into lower case and then try to remove all the unnecessary symbols, and numbers if they are included by using a regular expression [^a-zA-Z].

**At first, I tried a different approach for the regular expression [\\[\\].@,<>|!?:,;'[()]['']] but it didn't make sense as many unnecessary symbols were remaining. So what I did was by using this [^a-zA-Z] approach I was able to remove all the irrelevant content in the string but it was also not able to correctly tokenize some of the strings like résumé where the output was rsum where my regular expression failed to tokenize.**

To calculate the frequencies I had to check if the following token was present in the map if present I pulled the value of the token and replaced it with the updated value where I incremented it with +1.

And to sort the token by token I just used tree map for it as it sorts the following keys automatically. Whereas for sorting tokens by frequency as I needed to sort by value I had to use the comparator library to sort in descending order. To get the first 50 and last 50 tokens in each of the files I have used a counter so that it only writes the first 50 and keeps counting till map.size()-49 and stops writing at map.size() where it covers the last 50 tokens.

The following files are in the Attached Folder:

Input Directory: It contains a list of all the 503 html files.

Output Directory: Has 503 token output files for each input file, fiftysortbytoken.txt

Fiftytokenbyfreq.txt, totalsortbytoken.txt, and Totaltokenbyfreq.txt.

Fiftysortbytoken.txt: Contains the first 50 and last 50 tokens sorted alphabetically.

Totalsortbytoken.txt: Contains all tokens sorted alphabetically.

Totaltokenbyfreq.txt: Contains all tokens sorted in descending order by frequency value.

Fiftytokenbyfreq.txt: Contains first 50 and last 50 tokens sorted in descending order by frequency value.

**Execution:**

I have tried to execute the following program by giving the input and output directory paths internally which will help take the input from the HTML documents and create the following output files in the right output directory which has been successfully executed and the outputs were generated.

I have also tried another approach through the terminal but the following Maven project has some issues in finding the maven-plugin which needs to be updated in the pom.xml where I have found difficulty in executing using this approach.

| No of Documents | And its Elapsed Time |
| --- | --- |
| 10 | 150.92 milliseconds |
| 50 | 754.60 milliseconds |
| 100 | 1509.20 milliseconds |
| 500 | 7594 milliseconds |

**Conclusion:**

In the end, the code successfully tokenizes the text content of the input files removing non-alphabetic characters and converting words to lowercase. It counts the frequency of each token using TreeMap, parses HTML content using jsoup, and does file handling by iterating through each input file and processing them. It creates the output files for token frequencies, ensuring the output directory structure is maintained.