Jonathan Simonin

Professor Wei

CSE 2100

10 November 2016

Assignment 7 Document - Shakespeare

Programming Design:

The overall design of this program isn't too complex or in depth. The abstract idea behind it is essentially to gather all words found in the Shakespeare plays and find the frequency of each. In order to do the following, these steps were taken:

a. Import all the things required from the book and an Array List, buffer reader and a java exception handler
b. Read in the document, given the correct file path and read it line by line
c. Use a limiter to read each line word by word and store each unique word into a hashmap
d. If the word was already seen in the hashmap then increase the integer value it has by one, this will keep track of the frequency of that word
e. Create an array list that will store every value of the hashmap and sort it (sort goes from lowest to highest)
f. Print out all the buckets that have collisions, as well as the size of the bucket.
g. Print out each word frequency with its corresponding key value started from the highest value until the 1000^{th} most frequent word

Tradeoffs:

Considering how straight forward and to the point this assignment was, there are not many, if any, tradeoffs I would have considered or used for this assignment. I saw a few ways of doing this assignment, but the overall result in the end would have been the same and it seems to be efficient either way. Due to some of the draw backs of not having a perfectly documented file to work with there are some errors with this program that can't truly be fixed unless the file is fixed.

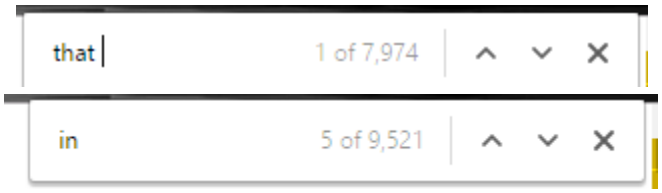
Extensions

If I were to have an extension for this program, I'd probably add some sort of error checking method that looks for words that should be counted correctly but aren't due to either punctuation or the space errors in the file.

Test Cases:

The only test cases I could come up required looking at the document and comparing it to my results in the output. Screenshots below will display and explain what I mean more appropriately – as mentioned before, the numbers may be off due to either Java not reading the file correctly or the way the hashmap is mapping all of the values and what is actually in the text file. The following images are taken from the output and the text file viewed from a browser and hitting control+f to view how many times it finds that word.

```
1. the was found: 26856 times.
2. and was found: 24116 times.
3. i was found: 22412 times.
4. to was found: 19225 times.
5. of was found: 16018 times.
6. you was found: 14097 times.
7. a was found: 13986 times.
8. my was found: 12283 times.
9. that was found: 11171 times.
10. in was found: 10640 times.
```



Considering the standard usage of these words, these results seem to be correct, however; due to errors in the document it is hard to pinpoint where exactly these inaccuracies lie in – since they could be in the way I'm searching for the words in the chrome ctrl+F window, the document itself, or the way the java code parsed the document and mapped it to a hash map. Moreover, the incorrect number counts could be due to collisions with the hashcode.