Daniel Reecy

5/23/2022

CEN 3024C

Module 2 Assignment – 2nd attempt

Requirements

1. Write a program that reads a file and output statistics about that file.
2. The output should include the word frequencies of all the words in the file.
3. The frequency output should be sorted by most frequent to least.
4. The output should be formatted in a set of pairs, containing each word and how many times it occurred in the file.

Planning

My initial plan was to break down the work into segments. Once I had one segment working I would move onto the next, using a loose waterfall methodology. This did not work throughout. I started by focusing on the file reader and printing out the contents of the file. I used a bufferedReader, and looped through each line of the file, assigning each string to an arrayList. With a regular expression I filtered out all non-word characters and whitespace. This removed the punctuation that would affect the word count.

After I had the arrayList which contained the file contents, I worked on tracking word frequencies in the contents. To do this, I initialized a hashMap which would contain each word in the file and their frequencies. I populated the hashMap and tracked the frequencies of each of these words in the same for ea loop. After this was successful I printed out the contents of the hashMap to confirm each word and its frequency.

Next, I tried to figure out how to organize the hashMap. I tried using a tree map but that organized by key and not value. I tried converting the map to an arrayList and I was able to organize that but the arrayList does not keep the keys, so I could not print out the words *and* their frequencies. ~~So far, I have been unable to organize the text in order of frequency and printing out both the word and their frequency.~~

After some emails with Professor Al Kafaf I was able to implement solutions for my problems. Instead of using an arrayList of values, I created an arrayList of Map.Entry<> objects. This allowed me to preserve the key/value pairs while also being able to use the .Sort() method. It required a custom comparator method, comparing the value of a given entry with the previous entry in the list. This allowed me to print out the contents of the list from greatest frequency to smallest.