Beteab Gebru

Compilers I – Project 1 : Iteration 2 **9/27/2018**

Software Design Document

The Word Counter

Overview

The program will initially be designed with command line interface to take in name of a document from a user which must be placed in the same directory as the program.

The main () function will take the name of the text document as an argument, which in turn is presented by the user to the system.

The program intends to use some libraries such as the hash maps, scanner from java library to read words. The functions will be modified to use several symbols and spaces as delimiters for the words. The collections library is used to achieve project aims.

Design

The Source files – there are three files containing all the program code.

`MyFrame.java`

`TempWindow.java`

`Words.java`

We call Libraries from the collections as well as the GUI component libraries needed to accomplish the task.

Functions Defined

//used to get the right type of words[no numbers, remove leading and trailing symbols...]

- *public static String formatWords(String Word)*

  //creates hashmap pairing of words and num of apps.

- *public static HashMap createHashMapofWords(File FileName)*

  //using sorted arraylist to write them and their corresponding counts in sorted order

- *public static void writeToTextFile(HashMap wordMap, ArrayList listOfWords)*

  //Prints contents of hashmap to console.out

- *static void - printer(HashMap Words, ArrayList listOfWords)*

The program Launches a GUI when it is run on IDE or cmd prompt. IN the GUI there is an option to place filename into the text input area or use the file chooser (labeled browse). Once the item is selected the name of the file is passed into the main(filename) as a parameter. The main function invokes some functions which were built for this word-scan-analyse tool.

The program builds a hash map of words and appearances in the text. The first word is stored and assigned count of 1. for every subsequent word that is grabbed by the scanner we check if the search returns true for a key match/ if false it goes to add a new word with count of 1. If a word is found to have been on the record already the count is incremented.

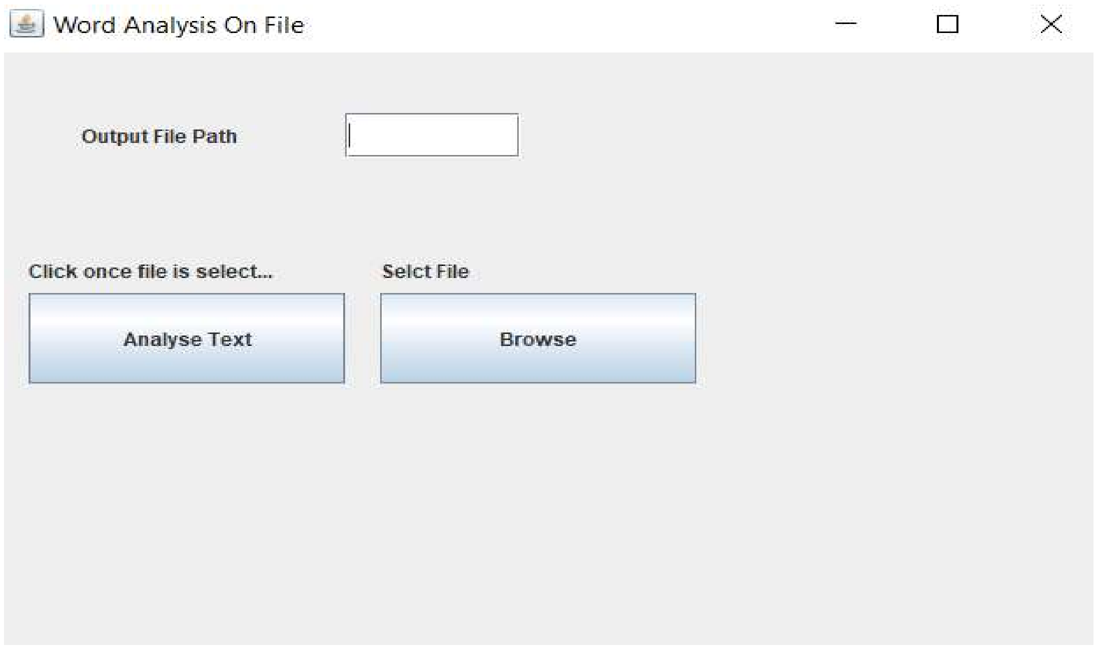
The GUI

-The GUI consists of a file chooser button, action button which results in main() being invoked. Main can call the various functions to achieve the results of words counts using some collections data structures.

-Describe the various functions

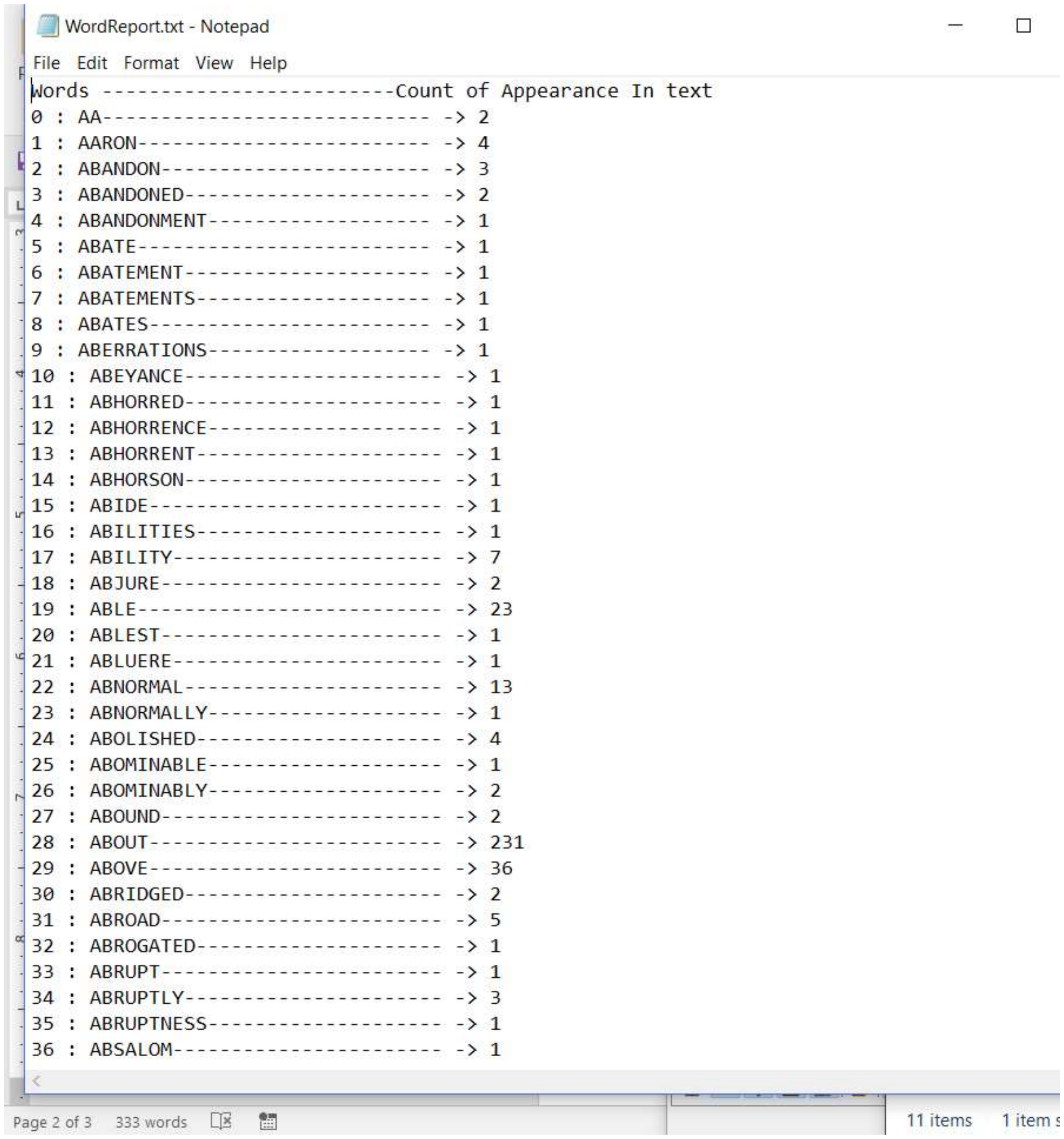
By the end of second iteration the program will be able to read a document and report some data and indicate where to find the results of the words analysis.

GUI used to select input file from directory

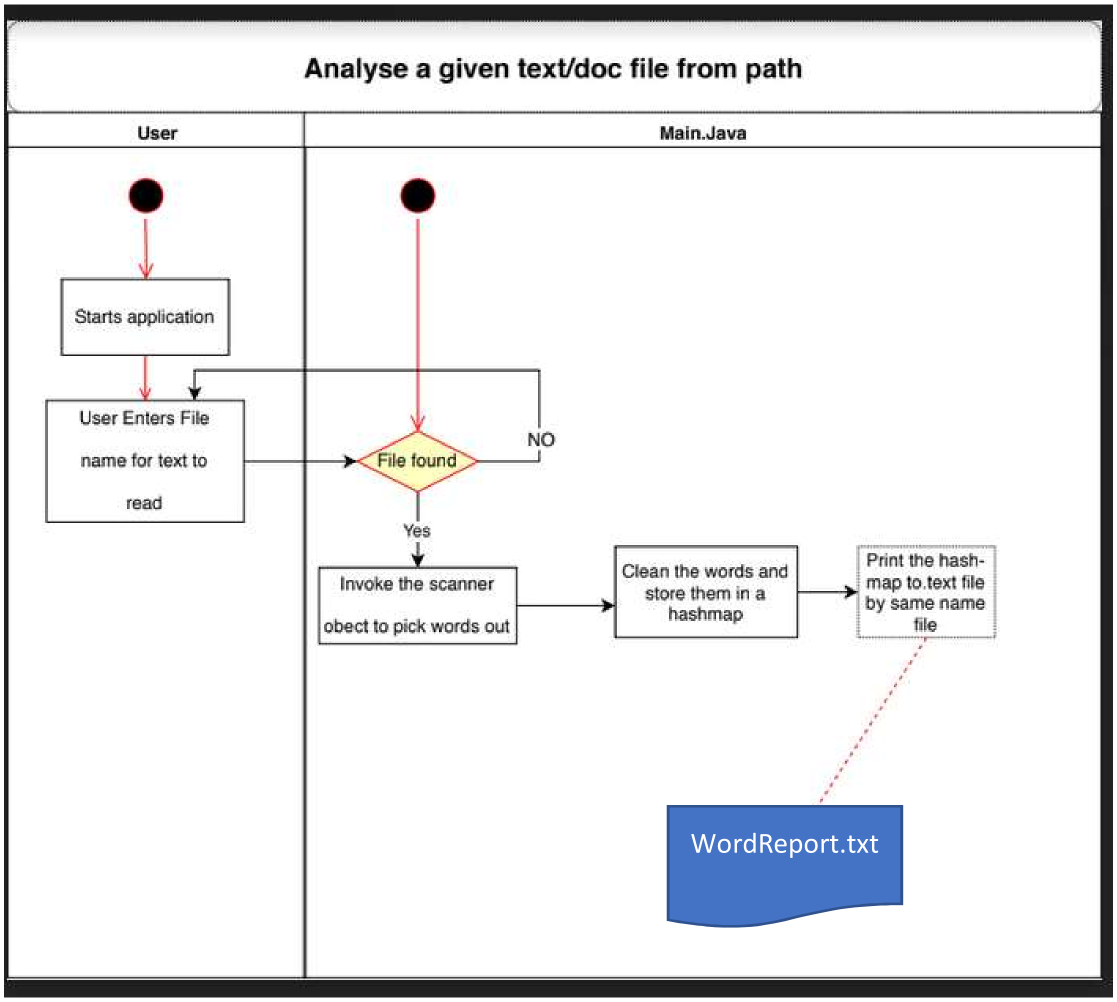


When the Analyze Button is clicked it produces a file output.txt with word report showing words and the number of times they appear in the text file selected.

An sample shot of report of the text analysis on Hamlet By Shakespeare is shown below.



```
Words -------------------------Count of Appearance In text
0 : AA---------------------------- -> 2
1 : AARON------------------------- -> 4
2 : ABANDON----------------------- -> 3
3 : ABANDONED--------------------- -> 2
4 : ABANDONMENT------------------- -> 1
5 : ABATE------------------------- -> 1
6 : ABATEMENT--------------------- -> 1
7 : ABATEMENTS-------------------- -> 1
8 : ABATES------------------------ -> 1
9 : ABERRATIONS------------------- -> 1
10 : ABEYANCE---------------------- -> 1
11 : ABHORRED---------------------- -> 1
12 : ABHORRENCE-------------------- -> 1
13 : ABHORRENT--------------------- -> 1
14 : ABHORSON---------------------- -> 1
15 : ABIDE------------------------- -> 1
16 : ABILITIES--------------------- -> 1
17 : ABILITY----------------------- -> 7
18 : ABJURE------------------------ -> 2
19 : ABLE-------------------------- -> 23
20 : ABLEST------------------------ -> 1
21 : ABLUERE----------------------- -> 1
22 : ABNORMAL---------------------- -> 13
23 : ABNORMALLY-------------------- -> 1
24 : ABOLISHED--------------------- -> 4
25 : ABOMINABLE-------------------- -> 1
26 : ABOMINABLY-------------------- -> 2
27 : ABOUND------------------------ -> 2
28 : ABOUT------------------------- -> 231
29 : ABOVE------------------------- -> 36
30 : ABRIDGED---------------------- -> 2
31 : ABROAD------------------------ -> 5
32 : ABROGATED--------------------- -> 1
33 : ABRUPT------------------------ -> 1
34 : ABRUPTLY---------------------- -> 3
35 : ABRUPTNESS-------------------- -> 1
36 : ABSALOM----------------------- -> 1
```


Analyse a given text/doc file from path
User
Main.Java
Starts application
User Enters File name for text to read
File found
NO
Yes
Invoke the scanner obect to pick words out
Clean the words and store them in a hashmap
Print the hash-map to.text file by same name file
WordReport.txt