Jeddah University
Faculty of Computing and Information Technology
Computer Science Department
CCCS311, Spring 2019

Program 1: Word Frequency Counter

Assigned: Thursday, Feb 07th, 2019
Due: Thursday, March 7th, 2019

Purpose

1. Learn to implement linked list for a real world problem.
2. Lean how to implement a recursive method for a real world problem.
3. Review file I/O (input/output).

Read Carefully:
This program is worth 4% of your final grade.

WARNING:
This is an individual project; you must solve it by yourself. Any form of cheating will result in receiving zero in the assignment. The deadline for this project is, March 7 2019 by 11:59 PM. Note: Once the clock becomes 11:59PM, the submission will be closed! Therefore, in reality, you must submit by 11:58 and 59 second.

Blackboard Submission:
This project must be submitted online via blackboard.
The source file(s) of your program should be zipped up. You must name the zip file using the following naming convention:
SectionNumber_StudentID_ProgramNumber.zip
Example: EA_1110348_P2.zip

Objective
The primary objective of this program is to implement a **linked list and to practice recursion**. The secondary objective is to practice with File I/O.

Program Description
Write a program to compute the occurrence frequency of each word in a given document and report different statistics. Occurrence frequency is the number of times a word is found in a document. Your program will take two text files as input.
1. The first file (document.txt) is the document from which the statistics will be computed.
2. The second file (commands.txt) contains a series of commands, one per line, which will tell you what to compute.

The basic functionality of computing word frequencies will be implemented using a linked list. Each node in the list contains a word, its frequency and a pointer to the next node.

The basic algorithm is as follows:

Read the text document
For each word in the document
    If the word is NOT found in the list
    Add this word to the list
    Set its frequency to 1
Else
    Increment the word's frequency by 1 //As this word already exists in the list

Hints

- To read the document use the Scanner() method
- To iterate over each word in the document use Scanner.next() method
- To add the word to the list, create a new node with the word and set its frequency to 1
- To increase the frequency, increment the frequency of the word by 1

Once the list is created, read the commands from the commands file to compute various statistics such as: totalWords, , findFrequency, etc.

Implementation

For this program you will create the following classes:

- **Word.java**: This class will be used to create objects of type word. Each word object will store the word and its frequency in the document.
- **DocumentWords.java**: This class will be used to create a linked list with nodes of type Word. All the methods will be implemented in this class.
- **FrequencyCounter.java**: This is the class that will contain main.

Input File Specifications

You will need to read two text files. This should be automated. Do not ask the user to input these files.
Document.txt: This is the text file from which different statistics will be computed.
Commands.txt: This file contains the commands that will tell you what to compute.

Example:
Let's assume that the Document.txt consists of the following paragraph.

> Call me Ishmael. Some years ago, never mind how long precisely, having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world.

The Commands to be implemented are as follows:

1. **TOTALWORDS**

This command will print the total number of words in the document. In this Document it is 43. There are no parameters for this command.

```
Example Input:
TOTALWORDS
Example Output:
Total number of words in the document = 43
```

2. **WORDFREQUENCY w**

   This command will be followed by a word *w*. The command will print the frequency of that word in the Document. If the word is not found, the command will print 0. If the document is empty the command will print "There are no words in this document"

```
Example Input:
WORDFREQUENCY years
WORDFREQUENCY me
WORDFREQUENCY Java
Example Output:
The frequency of the word “years” in the document = 1
The frequency of the word “me” in the document = 2
The frequency of the word “Java” in the document = 0
```

3. **REMOVEWORD w**

This command will be followed by a word *w*. To implement this command, you have to iterate over the linked list of words you created and delete the word (node). If the word is not found, the command will print 0. Then print the total number of words in the document after removing the word *w*.

```
Example Input:
REMOVEWORD years
REMOVEWORD  java
Example Output:
Total number of words in the document after removing the word “years“ = 42
The word "java"  was not found in the document.
 The Total number of words in the document remains the same= 42
```

4. **MOSTFREQUENT**

This command will print the most frequent word in the document.

Example Input:
MOSTFREQUENT
Example Output:
The most frequent word in the document is "me" with a frequency of 2

5. **PRINTLIST**

This command will print all the words of the document along with their frequency.

Example Input:
PRINTLIST
Example Output:
1 - Call : 1
2 - me : 2
3 - Ishmael. : 1
4 - Some : 1
5 - ago, : 1
6 - never : 1
7 - mind : 1
8 - how : 1.
.

6. **REVERSEPRINTLIST**

This command will print all the words of the document along with their frequency in reverse order you should implement this command using **recursion**.

Example Input:
REVERSEPRINTLIST
Example Output:
37 - world. : 1
36 - of : 1
35 - part : 1
34 - watery : 1
33 - the : 2
32 - see : 1
31 - a : 1
30 - about : 1.
.

Sample Input & Output File

You must print your output to an output file called **output.txt**. We have provided you a sample input with matching output.

WARNING

Your program MUST adhere to the EXACT format shown in the sample output file (spacing capitalization, use of dollar signs, periods, punctuation, etc). The graders will use very large input files, resulting in very large output files. As such, the graders will use text comparison programs to compare your output to the correct output. If, for example, you have two spaces between in the output when there should be only one space, this will show up as an error even through you may have the program correct. You WILL get points off if this is the case, which is why this is being explained in detail. Minimum deduction will be 10% of the grade, as the graders will be forced to go to text editing of your program in order to give you an accurate grade. Again, your output MUST ADHERE EXACTLY to the sample output.

Grading Details

Your program will be graded upon the following criteria:

1) Adhering to the implementation specifications listed on this write-up.
2) Your algorithmic design.
3) Correctness.
4) Use of the three classes, as specified. If your program is missing these elements, you will get a zero. Period.
5) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (We're not kidding here. If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
6) Compatibility to the newest version of NetBeans. (If your program does not compile in NetBeans, you will get a large deduction from your grade.)
7) Your program should include a header comment with the following information: your name, email, account number, section number, assignment title, and date.
8) Your output MUST adhere to the EXACT output format shown in the sample output file.

Deliverables

You should submit a zip file with four files inside:

1. Word.java
2. DocumentWords.java
3. FrequencyCounter.java (this is your main program)

***These three files should all be INSIDE the same package called wordcounter. If they are not in this specific package, you will lose points.

NOTE: your name, ID, section number AND EMAIL should be included as comments in all files!

UML Diagrams:

For this program, you will create two Classes (UML diagram shown below) and a third class for the main: