Platform Computing – Project 1

Due – September 27, 2019 11:59pm

Part 1 – Programming (80 points)

Textual Analysis of books

You are given a large file of text (e.g., books from Project Gutenberg). You need to read in the file and perform the following text analyses for the book:

1. **Letter Frequency:** Consider all letters 'a' – 'z'. For the given book, count the number of occurrences of each letter. Print out the top-10 most frequent letters along with the frequency in the book. (Random Aside: Frequency analysis is commonly used in breaking classical cipher encryption schemes. https://en.wikipedia.org/wiki/Frequency_analysis))
2. **Word Frequency:** Consider all words. For the given book, count the number of occurrences of each word. Print out the top-10 most frequent words along with the frequency in the book.
3. **Word Frequency with Stop List:** A "stop list" is provided (stop-list.txt). This contains the 573 most common words in the English language. This includes words like "the" and "and", which usually don't add any semantic meaning to the text. Consider all words in the book *that are not included in the stop list*. For the given book, count the number of occurrences of each word. Print out the top-10 most frequent words along with the frequency in the book *excluding words in the stop list*. (Random Aside: Word frequency with a stop list can give you a surprisingly good summary of the book.)
4. **Wild Card:** Come up with an interesting question. List the question and find the answer to it.

9 books from Project Gutenberg are provided (alice-in-wonderland.txt, christmas-carol.txt, huck-finn.txt, les-mis.txt, metamorphosis.txt, my-man-jeeves.txt, pride-prejudice.txt, tale-of-two-cities.txt, tom-sawyer.txt). Run your text analysis on these books. Feel free to download other books if you like.

Reading in the text file requires the use of Regular Expressions, which can be tricky. If you are not familiar with regular expressions, you can use the code I provide you below. The code assumes that you've read in each line of text from the file and stores each line as a String in an ArrayList. If you do that, you can use the code below to get each word from the String:

```
for(String line: fileLines) {
    scan = new Scanner(line);
    while(scan.hasNext()) {

    //this will read in the file and separate out each
word
```

```
        scan.useDelimiter("[^a-zA-Z']");
        String word = scan.next();
        word=word.toLowerCase();
        //replace all leading apostrophes
        word = word.replaceAll("^'+", "");
        //replace all trailing apostrophes
        word = word.replaceAll("'+$", "");
        //don't put empty string in map
        //you should implement that yourself
        //a simple if statement can check for that
        /* now you have a word */
}
```

You are free to design this program however you'd like. However, the program must start with a class named Project1.java. Project1.java will be the only class with a main method and will be the starting point of your project. You can use as many support classes as you'd like to design your solution.

This is a good opportunity to practice designing a modular piece of software. Think about breaking up your code into different classes. What methods should be included in each class.? What are the main tasks of the project? – reading a file, creating a HashMap with the contents of the file, analyzing the results and finally reporting the results.

Make sure to use Javadoc style comments for each method in your code.

Part 1 – Reflection (20 points)

Describe how you designed your program. What classes did you use? Did each class have a separate job or purpose? If you could refactor your code now that you've completed the project, what improvements would you make?

Part 2 – Grading Criteria

5% for compilation – If your code compiles, you get full credit. If not, you get a 0.
75% for functionality – Does the code work as required? Does it crash while running? Are there bugs? …
10% for design – Is your code well designed? Does it handle errors well? Are exceptions handled well? …
10% for style – Did you use Javadoc style comments in your code? Are your variables named appropriately? ...

Submission Instructions

Submit Project1.java and any other java files you used in your solution. Do not put your code in a package. If your project requires something other than putting all files in a java project in Eclipse to run, then please submit a readme file as well, describing how to run your program. Submit your reflection as a pdf file.