COMP-10205 – Data Structure and Algorithms

Assignment#3
Objects, Hashing and Searching

Who really wants the "Ring"?
7% of course grade

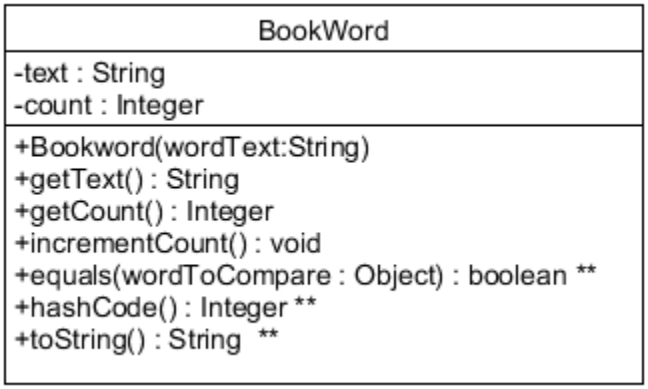
Submission Requirements

Complete the following exercise and submit electronically in the assignments folder on eLearn as an IntelliJ Project – Zip the entire folder not just the source files in MyCanvas. Please refer the course Calendar for the exact date and time of the submission. This assignment is to be completed individually.

Background

You are to write a program that will count the words and occurrences of words in the complete **Lord of the Rings Trilogy by J.R.R Tolkien**. In addition to counting words you must spell check the document and report how many words are not found in the provided dictionary. You will also answer the question of which character is most closely associated with the Ring using proximity searching.

You must use a class called BookWord (provided as UML below) to store each word for the words in the novel and for the words in the provided dictionary. The words must be stored without concern for case sensitivity (all characters must be converted to lowercase). A word is considered to be one or more characters in length separated by white space (space character, tab, new line character) or a comma, period, exclamation point or question mark. Quotes may appear in a word but these must be ignored and not included as part of the word. The apostrophe character may also appear in a word. You are provided with a regex that will filter these characters out during the file reading process. **Do not change the regex as doing so will possibly result in your results not matching the correct results**.



BookWord Class UML

** These methods exist in the Object class and must be overridden in your BookWord class

`BookWord` will be used to hold the characters for each unique word as a string along with a count which will contain the number of occurrences of that word in the novel. The main method will create an `ArrayList` of `BookWord` to hold the Words for the novel. The starting code should be used to read the words from the **LordoftheRings.txt** file. You may decide to put the initialization code into a method.

The **us.txt** file is a dictionary of words. You must use the dictionary to find the number of misspelled words (defined as not present in the provided dictionary) in the novel. You will need to create 2 dictionaries for this lab.

- The first must be an `ArrayList<BookWord>`. Sort the ArrayList using the `Collections.sort` method once all dictionary has been stored in the ArrayList.
- The second must be a `SimpleHashSet<BookWord>`. Do not use the `HashSet` or `HashMap` that are supplied as part of the Java Collection Framework.

Suggested Steps:

- Download the starter java code, SimpleHashSet.java, the novel(LordoftheRings.txt), and the dictionary (US.txt) from MyCanvas.
- Create a project in IntelliJ.
    - Add the US.txt and the LordoftheRings.txt file to the src folder.
    - Run the starter code which should print all the words in the novel to the screen.
    - Once the words have been displayed on screen, it is suggested you remove / comment out the printing of the words.
- Complete the BookWord class based on the UML provided.
  Notes:
    - Override the `toString` method to return a String that includes both the string representation of the word and the count.
    - The java Object class `equals` method must be overridden. The equals method must confirm that the **text** is the same in both `BookWords`. The count can be different.
    - The `hashCode` method must calculate and return a hashCode for each BookWord. The hashCode must be <u>based only on the characters in the word and not the count.</u> It is suggested that you find a good hashing algorithm on the internet and implement it in hashCode (You must code this and not use a built-in hashing method – i.e. do not use String.hashCode() for this assignment).
        - Site your reference in the source code for the hashing algorithm used.
    - Confirm that the selected algorithm produces good hashes. If you have less than 10% empty buckets in your HashSet on the Dictionary Words your algorithm is acceptable. There are public methods in the SimpleHashSet class that will allow you to calculate this value.

Part A – Hashing / Counting and Spelling

Ensure your application prints all of the following information to the standard output.

- Total # of words in the novel.
- Total # of unique words in the file.
- The list of the 10 most frequent words and counts

    - The data must be sorted to do this. Sort using the built-in Collections.sort method and supply a Lambda expression to complete the sort. This needs to be a two-key sort (first by count and then alphabetically).
- The list of words that occur exactly 64 times in the file. These words must be listed in alphabetical order. You should not have to sort the data more than once to achieve this.
- The # of words that are not contained in the dictionary. All three methods below should yield the same number of words. You must implement all three techniques and measure the performance of each. At the top of the code you must include a discussion of the results and why the performance is as measured.
    1. Use the **contains** method of `ArrayList` to find matches.
    2. Use the **binarySearch** method of the Collections class (**Collections.binarySearch**) to search the ArrayList dictionary. Supply a Lambda expression for the binary search of the Dictionary.
    3. Use the SimpleHashSet dictionary and the **contains** method provided

Part B – The Lord of the Rings

There are a number of novel characters in the Lord of the Rings Trilogy. At one time or other many are tempted by the Ring. Your goal in this section is to find the character that wants the ring the most.

One way to determine this is to perform a Proximity Search (https://en.wikipedia.org/wiki/Proximity_search_(text)). One method to calculate a numerical proximity is to determine the proximity distance between one word and another in terms of word position. To do this, each word must be numbered starting from 1 in the order they appear in the text. In the Lord of the Rings text, our goal is to measure the proximity distance of a character to the word "ring" within the text and count up the times it is within a proximity cut-off distance. For example consider the text below:

Frodo has entered the land of Mordor. Frodo must carry the Ring, but Gollum loves the ring.
(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17)

If we use a proximity distance cut-off of 4 word positions, then Frodo (word position 1) is 11 positions (12-1) away from the first ring and 16 (17-1) positions away from the second occurrence of ring. Neither of these two comparisons would result in a closeness count. The second occurrence of Frodo (word position 8) is 4 positions (12-8) away while the second occurrence is 9 (17-8) positions away. The first one is in this case would increase the closeness count. In the case of Gollum, he is only 2 positions (14-12) away from the first ring and 3 (17-14) from the second which would result in a closeness count of 2.

The word Frodo appears 2 times in the text and has a closeness count of 1. This would result is a closeness factor of 1/2 = 0.5 for Frodo. Gollum appears only once but is close to the Ring twice so for Gollum the closeness factor is 2/1 2.0. By this closeness definition Gollum is the Lord of the Rings.

Sample output required for Part b – Display the closeness factor with 4 decimal places

```
[gollum, 1] Close to Ring 2 Closeness Factor 2.0000
[frodo, 2] Close to Ring 1 Closeness Factor 0.5000
```

Who is the real Lord of the Rings

Evaluate the character list below with a proximity distance cut-off of 42 and present a sorted list by closeness factor (highest at the top). The Lord of the "Ring" will be at the top.

frodo", "sam", "bilbo", "gandalf", "boromir", "aragorn", "legolas", "gollum", "pippin", "merry", "gimli", "sauron", "saruman", "faramir", "denethor", "treebeard", "elrond", "galadriel"

Your solution includes a performance grade. The code for this part of the assignment must complete in approximately 50 ms (0.05 s). It is suggested that a class be created to store the positions of each of the words of interest and that this be constructed during the initial reading of the file. There are a number of ways to solve the problem that will give performance values around 50ms which is your target for this assignment. However, one technique should be able to complete the search in under 5ms. Can you find a way?

Marking Scheme

Program structure – Comments, follows best programming practices - 15%
Program Implementation – BookWord class implemented as required and complete and includes hashCode method with a cited reference for the hashing method – 10%
Program stores novel words into an ArrayList of BookWord and stores dictionary into both an ArrayList and SimpleHashSet of BookWord – 15%
Part A) - Output matches requirement – 20%
Part B) – Output is correct with all counts shown and closeness factor calculated correctly. Solution for Part B completes in approximately 50ms. – 20%
Discussion in comment at top of code – Dictionary lookup comparison discussion completed for ArrayList (linear search and binary search) and using the SimpleHashSet data structure – 20%