Project 5: Word Comparison

Due date: Friday August 10, 11:00 PM EST. **NO LATE SUBMISSIONS**

### You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own . Any sharing or copying of assignments will be considered cheating (this includes posting of partial or complete solutions on any public forum). If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own). You are responsible for every line in your program: you need to know what it does and why. You should not use any data structures and features of Java that have not been covered in class (or the prerequisite class). If you have doubts whether or not you are allowed to use certain structures, just ask your instructor.

In this project you will analyze the text from various classical books to identify common words between them and determine if they have an appropriate vocabulary to help high school students improve their reading skills. The output for a book will be the number of words in the book that are:

1. In common with another book entered by the user
2. On a provided target vocabulary list

The goal of the exercise to use hashmaps to store the necessary information to do the above analysis.

# Objectives

The goal of this programming project is for you to master (or at least get practice on) the following tasks:

* Segmenting string data for analysis
* Use of Hashmaps to store key, value pairs
* Use the Merge-Sort algorithm
* Debugging and checking results
* Analyzing a problem to determine the best approach

**Start early!** This project may not seem like much coding, but debugging always takes time. Due to the end of the term – all assignment submissions must be on time. **NO LATE submissions will be accepted**, this is because final course grades must be provided soon after assignment submission.

## Inputs

## Your solution should input two files. The first file will be named books.txt and will contain n records. The first record will be the focus book of the analysis. The remaining records will be all the books for comparison to the focus book. For example, if books.txt contains:

## JaneEyre.txt

## War\_peace.txt

## Gone\_with\_wind.txt

## David\_copperfield.txt

## This input would indicate that we be comparing Jane Eyre words against War and Peace, then against Gone with the Wind, and finally comparing Jane Eyre again David Copperfield.

## The second file will be named words.txt and will contain a set of words for comparison against the focus book only. In the above example the analysis would find all words in both Jane Eyre and the list of words in the words.txt file.

## Word Search Analysis

The book will be analyzed to find all words. Here ae some rules about words:

* At a minimum a word is all letters between two blanks.
* Ignore punctuation at the beginning of the word or end of the word – keep punctuation in the middle though.
* Eliminate all words that are only punctuation or numbers
* Identify proper nouns – words that begin with a capital letter, that are not the first word of a sentence
* Keep on the stem word - run vs. running, jump vs jumped or jumping. If a word ends in “ing” or “ed” and its root word exists, only keep the root word. For example - At the start line, the boy heard the whistle and started running.- this sentence would only store “start”, and not “started”.

After the word search analysis, your solution should have a list of unique words in the book. Each word should be labeled regular or proper noun. Your solution can now compare the focus book word list against each other book in the books.txt. Also your solution can now compare the focus book word list versus the list of vocabulary words in words.txt

## Book Comparison

## During the comparison of word sets by book, or for just printing the output word sets in sorted order, you will need to do a sort. Any sort performed should use the Merge-Sort algorithm.

## Expected Results:

## The output will reflect the comparison of the focus word list and the other books in the books.txt file. The output would look like the following (based on the example files above):

## Focus book JaneEyre Total Words: 2933 Proper Nouns: 43 Comparison book: War\_peace Total Words: 2633 Proper Nouns: 65

## Words in both: 3332 Words in JaneEyre only: 323 Words in War\_peace only: 832

## WordList JaneEyre only: alpha, baker, charlie

## WordList war\_peace only: alpha, baker, charlie

## (Repeat above 4 lines for Gone\_with\_wind.txt David\_copperfield.txt)

## Words in focus book: JaneEyre also in vocabulary list: apple, banana, cherry,…..

## Words in vocabulary list not in focus book: aaa, bbb, ccc

## Notes:

## Word lists should be sorted

## Do not print list of words that book have in common

## Include proper nouns in total words count – so 43 or 2933 words in Jane Eyre were proper nouns

## Running the program

The zipped java project file, which contains all your source code in your Eclipse project. This code should include all the code your wrote and two files – books.txt and words.txt. If I need to reconstruct your project because you only provided source files, you will lose 20% of the project points. If this is not clear to you , see me before or after class.

**Grading Criteria – 20 points**

1. (5) Able to accurately determine the words in a book file
2. (3) Processed all books in the books.text file and used hashmaps
3. (4) Comparison between books is correct
4. (4) Comparison between focus book and word list is correct
5. (4) Output is as described above and Merge sort was used, at a minimum, to sort the words in the ouput.