CIT 591 – Homework 5

Due – Nov 22, 2016 at 12.00pm

Part 1 – Theory (20 points)

Please do the following REVIEW EXERCISES from the class textbook:

1. R11.5, R11.7, R11.15, R11.17 (5 points each)

Part 2 – 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. **Quotes:** Many books have lots of quotations. E.g., from Alice: 'and what is the use of a book,' thought Alice 'without pictures or conversations?'; from Jeeves: You go up to them and say: "When's the next train for Melonsquashville, Tennessee?". Consider all quotations. For the given book, print out the top-10 shortest and longest quotations based on length.
5. **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.

Hint: Some of these might require the use of Regular Expressions. In general, you have three options: Use methods from the `String` class; Provide a delimiter to the `Scanner`; use the `Pattern` and

`Matcher` classes. I would recommend using the first option as far as possible – it's much easier to use and has an easier learning curve than the others.

Part 2 – Extra Credit (20 points)

For the EC, you need to create two versions of your program in Java, say Version A and Version B. Version A should be implemented using arrays (`ArrayList` or fixed-size arrays). Version B should be implemented with a `HashMap`. You will instrument these programs to tell you how long it takes (in seconds, milliseconds, etc.) for the following operations in Java for the two versions:

1. Creating the new object
2. Adding a new element
3. Updating an existing element
4. Checking if an element exists
5. Getting the top-10 for the frequency from above

Try out your instrumented programs on the sample text files that are included (you can download additional files, if you want). For each text file, focus on the last metric (#5) as it will be aggregate of the first four. Report this aggregate value in the `ec.txt` using the table below:

	Arrays (Version A)			**HashMap (Version B)**		
	Best Case	Average Case	Worst Case	Best Case	Average Case	Worst Case
Letter frequency						
Word frequency						
Word frequency with stop list						
Quotes						
Wild Card						

Finally, write a 1-2 paragraph evaluation of the results using metrics #1-#4 – Are the numbers as expected? If so, why? If not, why not? What are the bottlenecks? Are there ways to make it faster? Does your instrumented code add any overhead? …

As before, for the EC part, you cannot have any help from the TAs/instructor.

Hint: Take a look at the System class in Java for methods that might be helpful here.
Hint 2: In an ideal design, the two versions should have a set of common Java files, which they both use and (typically) just one Java file that is specific to that version.

Part 2 – Grading Criteria

5% for compilation – If your code compiles, you get full credit. If not, you get a 0.
65% for functionality – Does the code work as required? Does it crash while running? Are there bugs? …
15% for design – Is your code well designed? Does it handle errors well? Are exceptions handled well? …
15% for style – Do you have good comments in the code? Are your variables named appropriately? ...

Challenge Work (not graded)

If you're interested in doing a harder version of this homework, various options are provided below.

1. **Bigrams:** "Bigrams" are groups of two written letters (or sometimes two syllables, or two words). The word "the" has two bigrams – "th" and "he". For the given book, count the number of occurrences of each bigram. Print out the top-10 most frequent bigrams along with the frequency in the book. (Random Aside: Bigrams are very commonly used in most of the successful language models for speech recognition.)
2. **Synonyms:** Many words in the English language have the same meaning. E.g., "cars" means the same as "automobiles". Modify your program for "Word Frequency" and "Word Frequency with Stop List" to handle synonyms.
3. **Bi-words and Stop Lists:** "Bi-words" are groups of two words. E.g. the phrase "Alice was beginning" has two bi-words: "alice was" and "was beginning". Using the stop list provided, count the number of occurrences of each bi-word *excluding words in the stop list*. Print out the top-20 most frequent bi-words along with the frequency in the book.
4. (Advanced) **Stemming:** Many words in the English language have the same root word. E.g. computer, computing, computation, compute, computes. Modify your program for "Word Frequency" and "Word Frequency with Stop List" to handle stemming.

Programming – General Comments

Here are some guidelines with respect to programming style.

Please use Javadoc-style comments.

For things like naming conventions, please see Appendix I (Page A-79) of the Horstmann book. You can also install the Checkstyle plugin (http://eclipse-cs.sourceforge.net/) in Eclipse, which will automatically warn you about style violations.

Submission Instructions

We recommend submitting the theory part electronically also. However, you can turn in a physical copy at the start of class, if you prefer. Please **do not** print out the Java source.

In addition to the theory writeup, you should also submit a text file titled readme.txt. That is, write in plain English and instructions for using your software. You should also include your design decisions for the code. The readme.txt file is also an opportunity for you to get partial credit when certain requirements of the assignment are not met. Think of the readme as a combination of instructions for the user and a chance for you to get partial credit.

Please create a folder called YOUR_PENNKEY. Places all your files inside this – theory writeup, the Java files, the readme.txt file, the ec.txt file. Zip up this folder. It will thus be called YOUR_PENNKEY.zip. So, e.g., my homework submission would be swapneel.zip. Please submit this zip file via canvas.