Assigned: Friday 23 April 2010
Due: Tuesday 4 May 2010
Topics: I/O, Collections, Algorithms, Classes

Collaboration: This homework assignment may be completed individually or in pairs.

Submission: Follow the instructions for submitting programs via P-Web and handing in a printed copy. Be sure to generate unit tests and a statement justifying their sufficiency. One only one submission (paper and digital) per group is necessary.

Concordance

Recall from Assignment 5 that a concordance usually features the immediate context of each query word, along with where it appears. We did not have the tools to implement this efficiently at that time, but you do now. In constructing a interactive concordance for an arbitrary work, you will now be including a fixed, arbitrary amount of context on either side of a query word.

How shall we do this? We can still use a map from key words to a collection of entries. However, since the context is presumably different we really do want repeats of the same word on each line to be represented. Is this still a set? It depends on how we consider the equality of the set members. Although the line numbers will be the same, there are other things to distinguish them. Thus, we will still map entries to a collection that is a set.

We will want our program to be efficient, making only a single pass over each line of the text. Thus, we have to store the context for each word. You only need to display a fixed amount of context, so we now have just the right data structure for storing a fixed number of items, getting rid of the oldest as new items are added the fixed-length queue.

Assignment

You will write a program that takes a text file, builds an internal representation of the words it contains, and a set of all the appearances of each word (irrespective of case) with its associated context. As before, your program should be interactive and allow someone to query by word, being given the number of times it appears and then a sorted list of all the appearances. For instance (using the Project Gutenberg text of War of the Worlds):

```
$ java Concordance wells-wow.txt 3
Welcome to the Concordance. There are 6722 words in the index.
Enter a word: war
Word: WAR
12 contexts

[76] EXISTENCE IN A WAR OF EXTERMINATION WAGED
[86] THE STAR OF WAR BUT FAILED TO
[1081] THE FEVER OF WAR THAT WOULD PRESENTLY
[1368] VERY LIKE THE WAR FEVER THAT OCCASIONALLY
[4306] ACCOUNT OF THE WAR THE ARTIST HAD
[4411] EARTH DURING THE WAR AND IT WAS
[5200] NOTHING ELSE THIS WAR HAS TAUGHT US
[5328] ISN T A WAR SAID THE ARTILLERYMAN
[5328] NEVER WAS A WAR ANY MORE THAN
[5329] THAN THERE S WAR BETWEEN MAN AND
[5938] IN THEIR OVERTURNED WAR MACHINES SOME IN
[6229] EXAMINED AFTER THE WAR NO BACTERIA EXCEPT
```
Enter a word: envious
Word: ENVIous
1 contexts

[21] THIS EARTH WITH ENVIOUS EYES AND SLOWLY

Enter a word: disillusionment
Word: DISILLUSIONMENT
1 contexts

[22] CAME THE GREAT DISILLUSIONMENT THE PLANET MARS
Enter a word: “C

Note that there are two appearances of the first word on the same line. These should be given in the same order they appear in the text. Carefully note the line numbers of the last two examples.

Your program should take an arbitrary “window” size for the amount of context on each side of the word that appears in the concordance. You should choose data structure(s) that support fast access and appropriate representations for the the types of queries and processing that will be performed. When you submit your work, include a transcript of your working program (for any input), similar to the one shown above.

Storing Entries
You must print all the entries for a word in the order they appear in the text. While there are a variety of ways to do this, we will focus on one in particular. You are to write your own companion class, called Entry, to represent a single appearance of a word so that objects of this class can be stored in a set.

Noting the output shown above and the hints below, you should carefully consider

• what data should be stored in the object,
• how the supporting set implementation will store/process the instances of the object,
• what methods you may need to override in order to facilitate those set operations, and
• any other general methods you may need to implement or override

Helpful Hints
A basic algorithm for building your concordance might be organized as follows.

• Keep a collection of the (fixed-size) window of text that occurs before the “current” word (which should be the query entry).
• Keep a collection of the (fixed-size) window of text (possibly including the query entry) that occurs after the “current” word.
• Keep a corresponding collection of the line numbers the words in the “after” window are actually found
• Add an entry to your concordance consisting of the “before” and “after” text, with the appropriate line number
• As you process the file, add a word to the “after” window, placing the removed word from the “after” window in the “before” window.
• Fill your “before” window with empty strings before you start adding entries to your concordance
• Don’t forget to index the last few words of the file (when the “after” context is shrinking to empty)
• Class files and Javadoc for an appropriate FixedLengthQueue implementation are in
  /home/weinman/courses/CSC207/code/flqueue