CSC 207    Assignment 5: Using Collections    Fall 2010

Assigned: Tuesday 2 March 2009
Due: Tuesday 9 March 2009

Topics: I/O, Collections

Collaboration: This homework assignment must be completed individually.

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.

You now have some experience using the Java Collections API, which features many of the data structures we’ll be taking a close look at later in the semester. For now, we’ll practice using them.

Concordance

Traditionally, a concordance is an alphabetical list of words used in a book (or some other body of text), typically with a list of places the word appears and its immediate context. Such an effort is often useful for literary analysis. For instance, here is an entry from a concordance of Shakespeare’s works for the word *kindest* (http://www.opensourceshakespeare.com):

<table>
<thead>
<tr>
<th>Context</th>
<th>Line</th>
<th>Character and Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>We do request your kindest ears</td>
<td>1279</td>
<td>First Senator (Coriolanus II.2)</td>
</tr>
<tr>
<td>The dearest friend to me, the kindest man</td>
<td>1669</td>
<td>Bassano (Merchant of Venice III.2)</td>
</tr>
<tr>
<td>O, the kindest Kate!</td>
<td>1153</td>
<td>Petruchio (Taming of the Shrew II.1)</td>
</tr>
</tbody>
</table>

Concordances were tremendously difficult to create manually, but they are relatively easy to construct with the aid of a computer and a digital text. We’ll make some simplifications to the richness of the result, but you will be tasked with constructing a interactive concordance for an arbitrary work.

Assignment

You will write a program that takes a text file builds an internal representation of the words it contains, and a list of all the lines the word appears (irrespective of case).

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 lines the word appears on (not including any repeats). For instance (using the Project Gutenberg text of *War of the Worlds*):

Welcome to the Concordance. There are 6723 words in the index. Enter a word: war
Word: WAR
11 contexts
75 86 1081 1368 4306 4411 5200 5328 5329 5938 6229
Enter a word: world
Word: WORLD
25 contexts
4 26 28 65 66 112 896 1080 1628 2746 3142 3642 3965 4407 4675 5026 5136 5218 5304 5318 5461 5599 5971 6052 6064
Enter a word: WORLDS
Word: WORLDS
1 contexts
13
Enter a word: multiverse
Word: MULTIVERSE
0 contexts
Enter a word: °C
Additionally, as part of your program, you are to implement the following interface:

```java
public interface Concordance {
    /** Initializes a concordance from input */
    public void initialize (BufferedReader in) throws IOException;

    /** Returns a collection of lines where the word is found */
    public Collection<Integer> getLines (String word);

    /** Returns an iterator over all the words */
    public Iterator<String> getWordIterator();
}
```

In your program, you should choose data structure(s) that support fast access and appropriate representations for the types of queries that will be performed. When you submit your work, include a transcript of your working program, similar to the one shown above.

**Helpful Hints**

You can split a string into words (which we’ll consider things that only contain letters or numbers) using the `String` method `split`. This takes a regular expression (a method for string matching) to identify the delimiters. You may use the string “[\^A-Za-z0-9]”, which says to match anything that is not a letter (either case) or a number.

You may find the text of *War of the Worlds* on the MathLAN at `/home/weinman/courses/CSC207/misc/wells-wow.txt`. Other texts are available from Project Gutenberg [http://www.gutenberg.org](http://www.gutenberg.org).