CSE 223 Programming Assignment #2

April 18, 2019

1 Introduction

In this assignment, you're going to create a word index utility in Java. This utility will do the following:

- read text from stdin (ignoring anything that's not whitespace or a letter) and break it into tokens (called "words" even if they're not really words in the usual sense), and convert each word to lowercase;
- keep count of the location (starting from 1) of each word in the input stream;
- build an index of each word, identifying all the locations in which it occurs; and
- print the index, with the words alphabetically-sorted.

1.1 Sample Behavior

For example, if the input text consists of the following:

```
Hello, this is a test.
This is FUN!!!
Test (test) hello
Good_bye
```

then the words and their locations are:

```
hello 1
this 2
is 3
a 4
test 5
this 6
is 7
```

```
fun 8
test 9
test 10
hello 11
goodbye 12
```

Collecting this, we have the following index information:

```
hello 1,11
this 2,6
is 3,7
a 4
test 5,9,10
fun 8
goodbye 12
```

so the sorted output would be:

```
a 4
fun 8
goodbye 12
hello 1,11
is 3,7
test 5,9,10
this 2,6
```

2 Implementation Details

Internally, your index is going to be a linked list of linked lists. The main list will contain the words, which will always be sorted. Each node will (as usual) contain a piece of data (the word) and a pointer to the next node; but additionally, each node will also contain a pointer to a linked list of locations (integers). We'll draw some pictures in class...

2.1 Required Classes

You're required to implement the following three classes.

LocationList

This is a simple linked list of integers. Each word will be associated with a LocationList, which will be a list of all the locations where this word occurs.

WordList

This is a separate linked list of words (Strings). Each node will have three pieces:

1. the word;
2. a LocationList, containing all the locations where this word occurs in the input; and
3. a pointer to the next node in the WordList.

IndexUtility
This is the class your main program would construct and use. It has a single method: buildIndex() which reads from stdin and returns a WordList containing the built index.

2.2 Class Details

2.2.1 LocationList

constructor: takes no arguments, does whatever initialization you need
methods:

- void addToEnd(int data); add a new node (containing the given data) to the end of the LocationList

2.2.2 WordList

constructor: takes no arguments, does whatever initialization you need
methods:

- void addWord(String word, int location); either adds a new node (in sorted order), or appends a new location to the LocationList associated with the word
- void print(); traverses the list, and prints the (sorted) output: one word per line, followed by a space-separated list of locations (in increasing order)

2.2.3 IndexUtility

constructor: takes no arguments
methods:

- buildIndex() reads stdin, builds a WordList, and returns that WordList

3 Sample Usage

Here's how one might use these classes to index stdin:

```
class PA2Main
{
  public static void main(String[] args)
  {
    IndexUtility myIU=new IndexUtility(); // construct the IU
    WordList myList=myIU.buildIndex(); // index stdin, build a WordList, save as myList
    myList.print(); // show the index
  }
}
```

4 Additional Notes

- you must implement the linked list classes yourself. Java has linked list classes already available, but you aren't allowed to use those: you need to make your own.
- you do not need sentinel nodes, but may include them if you choose
- you'll want to define separate classes for the *nodes* of the two linked lists
- try to keep an object-oriented mindset when designing your code
- make your code modular as much as possible; develop it in pieces and build on what you've already done
- make sure you understand every line of your code. If you're not sure what something does, please ask about it
- I'll be grading your code by running my own main program, and by looking at your code itself
- this can be a really fun assignment! If you proceed logically, start small and build on your successes, it's actually very satisfying as you complete each piece. The individual steps can each very small, if you approach them in the right way.

5 Suggested Approach

I recommend the following staggered approach to completing PA2. You're welcome to develop this however you choose though.

1. Write a method to read individual "words" (tokens) from stdin using Scanner, hasNext() and next(). Read one word at a time and print it out.

2. Add a counter (beginning at 1) to count the location of each word; print this location followed by each word.

3. The words returned by next() will contain numbers, punctuation, and mixed cases. Write a method which takes such a string and returns a cleaned-up string (letters only, all lowercase). Print out the cleaned-up words along with their location in the input. Note that a string such as "12345" would convert to an empty string "" That's okay for now.

4. Now design a class (WordList) that implements a linked list whose nodes have two data fields: word (a String) and location (an int). For a first version, just add to the beginning of the list. Write a method for traversing the list and printing the location/word combination. Repeat step 3, but add words to the list, and then print the list at the end of your program. You should see the same output as in step 3, but in reverse order.

5. Repeat step 4, but now add the words to the *end* of the list instead of the beginning.

6. Next, change the data part of each node to hold 3 things: a word (String); an *array* of ints; and an int showing how many things are stored in that array. Repeat step 5, but instead of just adding each word to the end of the list, search the list for the word. If you find it, add the current location to the end of the array. If you get to the end of the list and haven't found the word, add it to the end of the list and note its location.

7. Design a new class (LocationList) that implements a simple linked list of integers. Include an addToEnd(int) method for adding an integer to the end of the list

8. Finally, repeat step 6, but change the node's data part to hold a word (String) and a list of locations (LocationList) instead of saving locations in an array.

You now have most of PA2 finished - congratulations! Modify the code for your WordList to add words in alphabetical order. Bundle your traversal code in a print() method, clean things up, and arrange the pieces as described in Section 2 (**Implementation Details**) and do more final testing. Make sure your code is well commented, has a comment block at the beginning of IndexUtility.java, and submit your tarfile to Canvas.

6 Submission

You should develop this project from the command line (on linux.engrcs.com); put all source files into PA2.tar (without subdirectories or packages); and upload PA2.tar to Canvas by the due date.