Cosc 1p03 Assignment 2

(Due date Oct. 21th 16:00 est, Late date Oct 24th 16:00 est)

Objective

Practice link list mechanics

Background

A cross referencer is a program which will scan source code and then produce a table of identifiers and the line numbers in which they appear on. Consider the below code fragment.

```
1. l=(int) sum;
2. for (int i=0;i<10;i++){
3.    remainder = l%2;
4.    l=l/2;
5.    t = t + remainder;
6. }
```

Applying a cross reference to the above code would produce a table as follows.

i	2
l	1 3 4
remainder	3 5
sum	1
t	5

Notice, that the table lists all identifiers in alphabetical order. For each identifier in the table, the line number in which it appears is listed. Notice once again that even though an identifier will appear multiple times on a single line, it is only listed once. As well, reserved words which make up the Java language are not listed. Identifiers are any user defined variable or type (unless it is a reserved word). Thus defined objects such as **String** would be considered an identifier (type identifier) and thus appear in the table. Syntactic components: (; + etc are not entered into the table, neither are literals e.g. 10. All identifiers are case sensitive. For the purpose of this assignment there are no comments.

The Assignment

You are to write a program which will produce a table similar to the one above. Create a linked list of java reserved words. The linked list will be sorted. The data file will need to be read and a sorted insert used to build this list. Each node in the list will contain 2 fields, (String word, and Node next).

You are to read in the source code file and parse it identifying all words, which will be either a reserved word or an identifier. To determine if a word is reserved it must be compared to the reserved word list. Once identified as a valid identifier as defined above, it is to be entered into a second linked list. Each node of this list will contain 3 fields (String word, int lineNum, Node2 next). Using the above code example the linked list would look as follows.

(i,2) (l,1)(l,3)(l,4) (remainder,3) (remainder,5) (sum, 1) (t,5)

As identifiers are entered, they are entered in alphabetical order, as well, as the identifier is encountered in subsequent lines, it is entered once again sub-sorted based on the line number. To create the table, this linked list is printed.

How to approach this assignment

1) The first major hurtle will be to tokenize the input. That is, when reading in the reserved words or parsing the source code file, it is desirable to retrieve only 1 word at a time. Create a method, getWord which will return exactly 1 word as a string from the desired input. Consider using String Tokenizer to help filter the input. Test your getWord method until it works flawlessly. Believe me when I say, this is a very important step. Your tutorial session will introduce you to using String Tokenizer.

2) Create a Node class to define the nodes to be used in the linked lists. The difference between the 2 lists is the nodes used. The fact is, you can define 1 node (word, line#, next) and use it for both. Just ignore the line# field in the reserved word list.

3) Create the reserved word list. Write a test method to verify that the reserved word list is in-order and complete.

4) Write the routines to insert a word into the cross reference list. You may need to draw many diagrams to help you follow your code, and verify that it actually does what you think it does. Write a small print routine to print the nodes and contents to System.out to help you verify that it is working.

5) Write the print routine to create the xref table.

6) This is likely your first time working with dynamic data structures. Small steps are advised. As you will find, Null Pointer Exceptions will be many and frequent. It will take time for you to wrap your head around the code, concepts and assignment. This means to start early and work on it over several days. DO NOT start the day it is due.

Submission

Include in your submission, the IntelliJ project directory, zipped. Be sure your name and student number are included in your source file. Also, provide good commenting as per javaDoc standards (see class example code).

Assignment submission is via Sakai.