Programming Homework 1

Data Structures and Algorithms in JAVA (3137)
Shlomo Hershkop
Department of Computer Science
Columbia University
Fall 2013

Out: Tue Sept 17, 2013
Due: Sunday, Oct 6 (11pm).

Programming Assignment 1 (100 points)

We will programming a modified linked lists implimentation. You will need to code your own linked list class either based on what we discussed in class, or based on the code from the book (available online) or made up from scratch.

In general, you will be held responsible to understand what any code you submit is doing and to document any outside code used (when appropriate). Please seek help from TA/Prof sooner rather than later. Your code needs to run everywhere, including the TAs laptop/desktop/etc

In addition to (hopefully) code, you need to create and include a text file called "readme.txt" which states your name, uni, and for each java class you create, briefly outlines what it does. Each java class file requires your uni on the top in the comments section. Failure will result in no credit for the specific part of the assignment.

Please backup work when you can, erasing all your work 2 hours before dealine will be excused for the first two people who use this excuse, everyone else is expected to have backups.....if you have questions on how/where/when/why backups, come to office hours. I would advise against naming your files "bankofamericasecrets.zip" and seeding it to the p2p networks.

Comments are important! Sprinkle them liberally throughout the code and before methods. NOTE: Comments will form a percentage of your grade, so working code without comments will affect your grade.

Step 1:

Create a class called SuperDuperLinkedLists.java which will implement a basic linked list class with a head and tail. Your class needs to support Insert, Delete, and Find operations on your list. In addition later on you will be adding Print, Reverse methods.

We will be keeping a linked list of String objects but feel free to use generics if you like. Feel free to create as many classes as you need (Example LinkedItem class). In addition create at least one exception class called SDLException (super duper linked exception) for use in your code.

Step 2:

- Adopt the delete method to throw an exception if the item is not in the current list.
- Implement a print method to print out the list in a nice fashion.

Step 3:

Now adopt the insert method and the class code so that a counter is kept with each item in the list. When insert happens, and the item is **already** present you should increment the counter instead of inserting it again in the list. So inserting “b”,”A”,”dd”,”b”,”b” will only have a list of unique items and b’s count will be 3. Change print so it also prints out counting information. Printing should take an argument to either print specific counts or each item or percentage (of all inserts) of each item. For example in the above example, print will either print “b:3” or “ b 60%” (since there are 5 inserts in total 3/5).

Step 4:

Implement a reverse method which flips the order of the list. Again feel free to adopt the class code for this. So if your original list is “a”,”b”,”x” calling reverse will make the list to be: “x”,”a”,”b”

Step 5:

Create a main in the HWTest.java class. Write code to insert 1000 random numbers (between 0-30) as strings. Adopt a print method so that you can print the list, and output each item along with a frequency counts:

```
2 14.29%
1 28.57%
3 14.29%
```

Step 6:

Add a method PrintN which will print out the top n items…for example if n=3 it prints the top 3 inserted elements.

In your readme, describe/analyze the running times for each of the methods you have implemented in your class.

Step 7:

Adding timing information to the code to print out the time at the beginning of the code, and time how long 1000 inserts takes, and how long reverse takes on the class.

Step 8:

Change Main: Take an arguement which will be the name of a text file. make sure it exists and insert each line of the file into your list. When done print out the entire list and its statistics. Extra Credit for sorted printout (highest occuring first).