COSC 201 – Lab #4

LinkedList vs ArrayList

Purpose: Practice using the Linked List class from the API

Tasks:

1.) Open Eclipse and start a new Java Project called LinkedList Project

2.) Add a new class called LLPractice.

3.) Your task: for this lab, you will be comparing the insertion and removal speeds of ArrayList and LinkedList.

   a. First, create two lists: an ArrayList of Doubles called **testlist1** and a LinkedList of Doubles called **testlist2**. Note, if you aren't type specifying your lists correctly, there will be a deduction assessed.
   b. Second, in main, get the first command-line argument and store it as an integer **n.** Randomly generate **n** doubles and insert them at random indices in **testlist1**. The easiest way to do this is to generate a random int from 0 to testlist.size(), inclusive, and use the add(index, element) method from LinkedList.
   c. Third, remove 1/4 of those elements, again picking random indices.
   d. Fourth, do b. and c. with **testlist2**.
   e. Add in timing code for both sets of code. To help you out, here's the timing code, with helpful comments on where to insert your code from b, c, and d. Test with command line arguments of 100, 1000, 10000, and 100000.

```
public static void main(String[] args){

      //declaration and instantiation of lists
      //process command-line argument

      long starttime = System.nanoTime();
      //testlist 1 code
      long totaltime = System.nanoTime() - starttime;
      System.out.println("ArrayList insertion and removal
      time: " + totaltime + " ns");

      starttime = System.nanoTime();
      //testlist 2 code
      totaltime = System.nanoTime() - starttime;
      System.out.println("LinkedList insertion and removal
      time: " + totaltime + " ns");

}
```

4.) In the comments of your submission (not in your code!) describe your observations in regards to the timing results that you see when you run your code.

5.) Turn in your properly commented code via Blackboard by 11:59 Monday. You may work on this lab in pairs.

Grading expectations: this is a two part lab – the implementation, but also the reflection. Possible deductions:

* Missing one of the two data structures.
* Missing timing code.
* Not using command line arguments.
* Lack of comments – comment block at the top of the file
* Lack of comment reflection as noted in #4.
* Not doing the correct type specification for the lists.