CS310: Advanced Data Structures and Algorithms

Spring 2014 Qualifying Assignment

Due: Thursday, Jan. 30 2014, in class

We have designed this assignment so that you and we can find out whether you are properly prepared for CS310. **It is due at the second class meeting.** That is a firm deadline, so that we can grade your paper and return it to you (or notify you by phone or email) in time for you to switch to another course should that be necessary. If you feel that you had trouble with the assignment but still think you belong in the course please contact your instructor immediately. If you did not attend the first class (and so did not receive this assignment in time), make sure you discuss a due date with your instructor.

Your company has a server and there are 500 PC stations whose filesystem resides on it, numbered 1 through 500 inclusive. Every 10 minutes the system appends to a log file the PC numbers and user name of the person currently logged on, one PC number and username per line of text. (Not all stations are in use at every moment). A fragment of the log file looks like this:

```
...   ......
9     ALTEREGO
12    ALIMONY
433   HOTTIPS
433   USERMGR
12    BLONDIE
433   HOTTIPS
354   ALIMONY
...   ...
```

This log file shows HOTTIPS was on PC station 433 twice but USERMGR was on that station at an intervening time. It also shows that ALIMONY was on station 12 at one time and station 354 later on. The log does not display the time at which each line was written. Your job is to write a program in Java that meets the following specifications:

1. The program first reads a log file into memory, storing the input data as it encounters it. Assume IO redirection when your program is invoked, so it can read from System.in rather than by actually opening a named text file. User names are ASCII (plain text) strings with no embedded whitespace and a maximum length of 40 characters.

2. After all the data has been read your program should print data about PC station usage: a header line and then one line of output for each PC showing the PC station number, the most common user of that station (in the event of a tie, choose one user), and a count of how many times that user was on that PC station. Here is sample output:

   ```
   Line   Most Common User   Count
   1      OPERATOR           174983
   2      HANNIBAL           432
   3      <NONE>             0
   4      SYSMGR             945
   ...    ...                ...
   ```

3. **Program design:** The attached listing shows class SinglyLinkedList.java, for your use to hold all the information about one PC. Do not change this class, but instead, wrap it in another class LineUsageData (i.e. LineUsageData HAS-A SinglyLinkedList.) Although normally we would use a JDK LinkedList or other JDK container for this, for the qualifier the essential thing is to check that you know how to work with any reasonable Java classes. Design and implement a class **Usage** to hold a (username, count) pair, and a class **TermReport** for the top-level code. Usage can be a nested class of TermReport, or a

top-level class. An object of type LineUsageData holds all the data for one particular PC station, and has methods addObservation(String username) and Usage findMaxUsage(). TermReport.java should be decomposed into methods; including a main(). Each method should be commented (a comment above the method to describe it and comments for internal statements as needed).

Turn in the code for classes Usage, TermReport, and LineUsageData. Also please put a working email address and phone number at the top of the paper. Your program should be complete (except for the provided code) but need not compile. We would rather you wrote it with a text editor and turned in hard copy, but legible(!) handwritten copy is OK for now (but keep in mind that one of the future assignments will require you to compile and run this code). We will be looking for correct, clear logic, organization and style, not for the precise placement of semicolons or other trivial bugs. Be sure to acknowledge any sources for material you incorporate in your program. Tell us if you used a textbook, a project from another course, code off the web, etc.

Note: Since the purpose of this exercise is to determine whether **you** are prepared for this course you should not discuss the material with anyone else. (Later in the term we will encourage these discussions.)

SinglyLinkedList implementation: do not change this code, just use an object of this type as a field in LineUsageData. This file is available on the class website at www.cs.umb.edu/cs310.

```
/*
 * SinglyLinkedList.java
 */

public class SinglyLinkedList<T>
{
  private Node<T> head; // the head dummy node
  private int size;  // list size (# elements)

  public SinglyLinkedList()
  {
      head = new Node<T>(null, null); // dummy header node!
  }

  // helper method
  // insert a new node with given element after a given node in list
  private void insertAfter(Node<T> node, T element)
  {
      // create the new node
      Node<T> n = new Node<T>(element, node.next);
      // insert the node in the list
      node.next = n;
      // update the list size
      size++;
  }

  // add o to end of list
  public boolean add(T o)
  {
      // we move to the last node
      Node<T> current = head;
      while (current.next != null)
        current = current.next;
      insertAfter(current, o);
      current = null;
      return true;
  }

  // get the ith element, or throw if beyond end of list
  public T get(int index) {
      // we move to the ith node, if possible
```

```
        Node<T> current = head.next; // start after dummy
        int i = 0;
        while (current.next != null && i < index) {
          current = current.next;
          i++;
        }
        if (i == index)
          return current.data;
        else
          throw new IndexOutOfBoundsException();
    }

    public int size()
    {
        return size;
    }

    public boolean isEmpty()
    {
        return size == 0;
    }

    // quick test of this code: list of Strings, list of Integers
    //
    public static void main(String[] args) {
        SinglyLinkedList<String> list = new SinglyLinkedList<String>();
        list.add("foo");
        list.add("bar");
        list.add("bar");
        System.out.println("list size = " + list.size());
        System.out.println("#0 element = " + list.get(0));
        System.out.println("#1 element = " + list.get(1));
        System.out.println("#2 element = " + list.get(2));
        SinglyLinkedList<Integer> list1 = new SinglyLinkedList<Integer>();
        list1.add(6);
        System.out.println("list1 size = " + list1.size());
        System.out.println("#0 element = " + list1.get(0));
        // this will throw an IndexOutOfBoundsException--
        System.out.println("#1 element = " + list1.get(1));
     }
}

// this is our node class, a singly linked node
class Node<T>
{
    T data;
    Node<T> next;

    Node(T data, Node<T> next)
    {
      this.data = data;
      this.next = next;
    }
}
```