CS 202 Lab/Programming Assignment 1

This Lab/Assignment serves as a review of (reference based) linked lists. You need to do the following:

1. Download **StudentRec.java, SortedStudentList.java, Lab1.java** and **StudentDataFile.txt** from Kodiak.

   Note that I have removed the iterator method from the SortedStudentList class. Therefore to step through a student linked list, you may use code like this:

   ```
   p = head;
   while (p != null)
   {
         ...   // access the node
         ...

         p = p.next;
   }
   ```

   I also add a print method in the class which allows us to display contents of the linked list.

2. Add the following methods to SortedStudentList class:

   (1) **int csMajors()**

   This method returns the number of CS majors in the linked list.

   Hint: Initial a count variable to 0. Step through the linked list – see code above. If a node contains a CS major, increment the count.

   (2) **float highestGPA()**

   This method finds and returns the highest GPA of all students in the list. If the list is empty, throw an exception.

   Hint: Initial a max variable to something (what??). Step through the linked list – see code above. If a node contains a GPA that is higher than max, set max to this GPA value.

(3) **StudentRec findFirst()**

This method returns the first element of the list (the student data stored in the first node). If the list is empty, throw an exception.

(4) **StudentRec findSecond()**

This method returns the second element of the list (the student data stored in the second node). If the list does not have a second element, throw an exception.

(5) **StudentRec findLast()**

This method returns the last element of the list (the student data stored in the last node). If the list is empty, throw an exception.

Hint:

```
if the list is empty  // does not have a last node
     ...
else
{
     p = head;
     while ( ... )    // How to determine p is not
                      // the last node?
     {
          ...  // move to next node
     }
}
```

(6) **StudentRec findSecondLast()**

This method returns the second element from the end of the list (the student data stored in the second last node). If the list does not have a second last element, throw an exception.

Hint:

```
if the list contains less than two nodes
     ...
```

```
else
{
     p = head;
     while ( ... )    // How to determine p is not
                      // the second last node?
     {
          ...   // move to next node
     }
}
```

(7) **void deleteFirst()**

This method deletes the first node of the list. If the list is empty before the deletion, throw an exception.

(8) **void deleteSecond()**

This method deletes the second node of the list. If the list does not have a second node before the deletion, throw an exception.

(9) **void deleteLast()**

This method deletes the last node of the list. If the list is empty before the deletion, throw an exception.

Hint: First, you need to find the last node. What should be done to delete that node?
Beware the case that the last node is also the head of the linked list.

(10) **void deleteSecondLast()**

This method deletes the second last node of the list. If the list does not have a second last node before the deletion, throw an exception.

Hint: First, you need to find the second last node. What should be done to delete that node?
Beware the case that the second last node is also the head of the linked list.

(11) **void removeDismissed()**

This method deletes all nodes that contain GPA less than 2.0.

```
Hint:     if the head of linked list has GPA < 2.0
          {
               ...
               // Beware: what if next node(s) also has (have)
               // GPA < 2.0?
          }

          go through the remaining linked list (if there is more
          node) and remove those nodes with GPS < 2.0  (what do
          you need to do to remove a node?)
```

(For efficiency, methods 7 – 11 should not call the existing delete method.)

3. Complete **equals** and **clone** functions in SortedStudentList class.

Hints:

```
public boolean equals(Object anotherList)
{
     SortedStudentList  list
                = (SortedStudentList) anotherList;

     if (numItems != list.numItems)
          return false;
     else
     {
          use a while loop to step through both lists    (How?)
          {
               if at any time, the corresponding elements of 2 lists
               are different
                    return false;
          }

          return true;     (Why?)
     }
}
```

```
public Object clone
{
     create a new SortedStudentList object;
     if this original object is empty then you are done
     otherwise
     {
          copy over the number of items;
          create the head of the new object so it contains the same
               data as the head of this original object;
          use a while loop to step through the rest of this
          original list, and for each node
          {
               create a new Node object that contains the same
                    data;
               attach it to the new linked list;
          }
     }
     return the list object that you have created and copied;
}
```

(For efficiency, the clone method should not call the existing insert method.)

4. Compile StudentRec.java and your completed SortedStudentList.java.

   Compile and run the Lab1.java program which tests your completed SortedStudentList.java class. Use **StudentDataFile.txt** as your input data file.

5. Submit your completed **SortedStudentList.java** and the **output** file (a text file) to Kodiak. Be sure to comment your code and put **your name** and a **brief description** at the beginning of your program. In the output file, be sure to put **your name**, and "**Output from Lab1**" at the beginning of the file.

Here is a copy of Lab1.java. It is used to test your completed SortedStudentList class.

```
//  Lab1.java

import java.io.*;
import java.util.*;

/**
 *  This program tests your completed SortedStudentList class.
 */

public class Lab1
{

     /**
      *   Program execution starts from this main program
      */

     public static void main(String[] args)
     {
          Scanner             keyboard = new Scanner(System.in);
          String              fileName;
          SortedStudentList   class1 = new SortedStudentList();
          SortedStudentList   class2 = new SortedStudentList();
          SortedStudentList   class3;
          StudentRec          aStudent;
          int                 n;
          float               max;

          if (args.length == 1)
               fileName = args[0];
          else
          {
               System.out.print("Please enter the data file name:  ");
               fileName = keyboard.next();
          }

          readData(fileName, class1, class2);

          if (class1.equals(class2))
               System.out.println("\n\nClass lists 1 and 2 have "
                                  + "the same contents");
          else
               System.out.println("Class lists 1 and 2 do NOT "
                                  + "have the same contents");
```

```
if (class1 == class2)
     System.out.println("Class lists 1 and 2 have "
                              + "the same address");
else
     System.out.println("Class lists 1 and 2 do NOT "
                              + "have the same address\n\n");

class3 = (SortedStudentList) class1.clone();

if (class1.equals(class3))
     System.out.println("Class lists 1 and 3 have "
                              + "the same contents");
else
     System.out.println("Class lists 1 and 3 do NOT "
                              + "have the same contents");

if (class1 == class3)
     System.out.println("Class lists 1 and 3 have "
                              + "the same address");
else
     System.out.println("Class lists 1 and 3 do NOT "
                              + "have the same address\n\n");

System.out.println("This is the original class list 1:\n");
class1.print();
System.out.println("\n\nThis is the class list 3:\n");
class3.print();

n = class1.csMajors();
System.out.println("\n\nThere are " + n
          + " CS majors in the class list 1.\n");

try
{
     max = class1.highestGPA();
     System.out.println("\n\nThe highest GPA in list 1 is "
               + max + "\n");
}
catch (Exception e)
{
     System.out.println("\n\nThe list is empty, "
               + "cannot find highest GPA.\n");
}
```

```
        try
        {
             aStudent = class1.findFirst();
             System.out.println("\n\nThe first student in list 1 is "
                         + aStudent + "\n");
        }
        catch (Exception e)
        {
             System.out.println("\n\nThe list is empty, "
                         + "it does not have the first element.\n");
        }

        try
        {
             aStudent = class1.findSecond();
             System.out.println("\n\nThe second student in list 1 is "
                         + aStudent + "\n");
        }
        catch (Exception e)
        {
             System.out.println("\n\nThe list does not have "
                         + "second element.\n");
        }

        try
        {
             aStudent = class1.findLast();
             System.out.println("\n\nThe last student in list 1 is "
                         + aStudent + "\n");
        }
        catch (Exception e)
        {
             System.out.println("\n\nThe list is empty, "
                         + "it does not have the last element.\n");
        }

        try
        {    aStudent = class1.findSecondLast();
             System.out.println("\n\nThe second last student in list "
                         + "1 is " + aStudent + "\n");
        }
        catch (Exception e)
        {
             System.out.println("\n\nThe list does not have second "
                         + "last element.\n");
        }
```

```
          try
          {
               class1.deleteFirst();
               class1.deleteSecond();
               class1.deleteLast();
               class1.deleteSecondLast();
          }
          catch (Exception e)
          {
               System.out.println("\n\nError occurred during the "
                         + "deletion.\n");
          }

          System.out.println("This is the list 1 after deletions:\n");
          class1.print();

          class3.removeDismissed();
          System.out.println("This is the list 3 after all students "
                    + " with GPA < 2.0 deleted\n");
          class3.print();

     }



     /**
      *   This method reads data from data file, and store the
      *   records in two SortedStudentList objects.
      *   @param fileName The input data file.
      *   @param list1 Sorted linked list read form the data file
      *   @param list2 Another sorted list read from the same file
      */

     static void readData(String fileName, SortedStudentList list1,
                                          SortedStudentList list2)
     {
          ...  // same as Sort List handout
     }

}
```