CIS 232 – Assignment 2

(125 points)

Due: Upload to Canvas by Fri. 10/13 at 11:59p (email late assignments)

In this assignment you will get practice working with a singly-linked list. Implement (in Java) a node/iterator/linked list combo that accomplishes the following. You may want to use the text's classes as a starting point as you must retain (unchanged) all methods of Weiss' `nonstandard LinkedList` classes (the three in 17.2, not 17.5)

Amend the `LinkedList` so that:

1. Your list classes must be generic (see below) and be able to work with any `Comparable` data.
    - Be sure to genericize any static methods, e.g. `printList()` would become:
      `public static <AnyType extends Comparable<? super AnyType>> void printList( YourLinkedList<AnyType> theList )`
2. The list is always maintained in ascending order so as each item is added to (or removed from) the list, it must be put in its proper place in the list just like in Assignment 1. This will be handled by a new method, `add()`, that takes in a lone parameter (the data object to be inserted - you do not pass in an iterator like you do with `insert()`.) Duplicate values are allowed in the list.
3. Change the `insert()` method to `private` visibility.
4. Add a method named `replace()` that takes in two parameters (the data object to replaced, followed by the one to be inserted) that will remove the first occurrence of the item to be replaced and add the second parameter to the list. The method returns a `boolean` - true if the item to be replaced was found, false if not.
    - If the value to be replaced is not present, do not perform the add.
5. Add an instance method named `showList()` (no parameters or return value) that will extract all of the values from the list and display them in order, one per line, followed by another line displaying how many values are currently in the list. If the list is empty, display the message "The list is currently empty" instead.
6. Overload `showList()` with an integer parameter which represents how many items to display on each line. The method needs to newline after the last item no matter how many items were displayed (i.e. don't leave the cursor hanging.)
7. As with Assignment 1, add an instance method named `getMode()` that functions the same as the version in Assignment 1. Reuse the same interface but change the package to the one designated below.
8. Add a method named `author()` that returns a String consisting of your name.
9. Revise and do not subclass the Weiss classes. Do not make your classes part of any package other than the one specified below for this assignment. That package should consist only of the four classes (list, iterator, node, result) used in this assignment.

(continued on back)

Additional notes:

1. You cannot use any of the API collection/iterator classes to implement or provide infrastructure for your three list classes. If there is a particular class you're not sure about, please ask as early as possible.
2. Your node/link type must have its data field be of type `Comparable`. Note that you may have to amend your classes/methods to handle this.
3. Efficiency counts as with Assignment 1 – notable redundancies and other correctable inefficiencies will be penalized. Pay special attention to your inserts and the mode.
4. All incumbent methods of the Weiss remain part of your classes and cannot be changed in any way except where specified above.
5. All other stipulations from Assignment 1 carry over unless explicitly rescinded.

Various Administrative:

1. Keep all list-related class names the same and preface each one (and its source code file) as follows: A2232, the first letter of your first name followed by your last name - e.g. `A2232RScoLinkedList.java`
2. All of your classes must belong to this package: `cis232.a2;` - using another package will result in a noticeable deduction. `Result` is also in this package.
3. Documentation requirements are the same as for Assignment 1. Include the writeup file in your zip.
4. Upload (ONLY) your write up and source code files (as an attachment, send one .zip file) to Canvas by the deadline. Be sure to NOT send the `Result.java` interface or any other project infrastructure.