EEM 480 Homework 2

This homework is designed to gain the student the notion of working with double-linked lists. Your task is to write the necessary codes which are defined in the interface HW2Interface. This interface is designed to create a LinkedList class that uses a node class DoubleLinkNode as given below:

```
public class DoubleLinkNode {
    public int Element;
    public DoubleLinkNode left;
    public DoubleLinkNode right;
}
```

You may NOT use Java's built-in data structure libraries, like java.util.LinkedList, in this homework. All data structure implementations should be your own or those taken from lectures. **You cannot use stack or queue** in your implementations.

The HW2Interface is :

```
public interface HW2Interface {
    public void Insert(int newElement, int pos) throws Exception;
    public int Delete(int pos) throws Exception;
    public void LinkReverse();
    public void SacuraL();
    public void OpacuraL();
    public void Output();
    public void ReverseOutput();
    @Override
    public String toString();
    public Exception LinkedListException();
}
```

The task of methods are given as:

```
public void Insert(int newElement, int pos) throws Exception // inserts an
int element defined in newElement after the position of pos into the
LinkedList. It checks whether pos is valid or not. If pos is not a valid
position for the LinkedList, it throws an exception.

public int Delete(int pos) throws Exception; // deletes the element stored in
position pos in the linked list and returns the value of the element. If
there is no element in pos, it throws exception.

public void LinkReverse(); // the link will be reversed after this method has
been initiated.

For example, if the link list is in the form [2 3 2 2 1 2 2 2 2 3 3 3 3 2 1]
the reverse method changes it to [1 2 3 3 3 3 2 2 2 2 1 2 2 3 2]

public void SacuraL(); // SacuraL takes the linked list and traces the
contiguous blocks in the list. After initiating the method, it changes the
linked list into the list such that there are tuples: the element and number
of occurrences in the list.
```

For example, if the list is [1 2 3 3 3 3 2 2 2 2 1 2 2 3 2], after initiating the method the link list converts to: [1 1 2 1 3 4 2 4 1 1 2 2 3 1 2 1]

public void OpacuraL(); //OpacuraL is the reverse of the SacuraL function.

For example, if the list is [1 1 2 1 3 4 2 4 1 1 2 2 3 1 2 1], after the method, the link list will be: [1 2 3 3 3 3 2 2 2 2 1 2 2 3 2]

public void Output(); // Put the elements of the link list to console starting from the beginning.

public void ReverseOutput(); // Reverse Output. Put the elements of the link list to the console starting from the end to the beginning. The linked list structure will not be affected.

public String toString(); // No need to explain

public Exception LinkedListException(); // The Insert and Delete methods throw this exception if something wrong.

I will check your codes like the program given below. Check your own codes with this code.

```
public class HW2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        LinkedList myList = new LinkedList();
        try {
            myList.Insert(1, 0);
            myList.Insert(2, 0);
            myList.Insert(3, 0);
            myList.Insert(3, 0);
            myList.Insert(3, 0);
            myList.Insert(3, 0);
            myList.Insert(2, 0);
            myList.Insert(2, 0);
            myList.Insert(2, 0);
            myList.Insert(2, 0);
            myList.Insert(9, 5);
            myList.Insert(9, 6);
            myList.Insert(1, 0);
            myList.Insert(2, 0);
            myList.Insert(2, 0);
            myList.Insert(3, 0);
            myList.Insert(2, 0);
            myList.Insert(5, 54);

        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
        myList.Output();
        myList.LinkReverse();
        myList.Output();
        myList.SacuraL();
```

```
        myList.Output();
        myList.OpacuraL();
        myList.Output();
        myList.ReverseOutput();
        System.out.println(myList);
    }
}
```

The output should look like:

```
run:
hw2.HW2Exception: Not supported yet.
The Elements in the list are : 2 3 2 2 1 2 2 2 2 3 9 9 3 3 3 2 1
The Elements in the list are : 1 2 3 3 3 9 9 3 2 2 2 2 1 2 2 3 2
The Elements in the list are : 1 1 2 1 3 3 9 2 3 1 2 4 1 1 2 2 3 1 2 1
The Elements in the list are : 1 2 3 3 3 9 9 3 2 2 2 2 1 2 2 3 2
The Reverse Elements in the list are : 2 3 2 2 1 2 2 2 2 3 9 9 3 3 3 2 1
1 2 3 3 3 9 9 3 2 2 2 2 1 2 2 3 2

BUILD SUCCESSFUL (total time: 0 seconds)
```

The following files are attached :

DoubleLinkNode.java // Describes LNodeClass

HW2Interface.java // interface which will be used in LinkedList class

LinkedListJava // You have to fill in the codes

HW2.java // test program for your codes.

Rules for HW Submission

. You have to write your HW in NetBeans environment.
. You have to write a report with name "Report_HW2.pdf" explaining your HW (purpose, how did you solve it, algorithm etc.) and the environment you used (NetBeans, for example). The person who read your report can easily use the class you have written.
. Submission should be in the form of a zip. When extracted, the result should be a single folder with the name "HW1", "HW2", or "HW3"
. Don't forget to put your report into the zip file.
. The name of your project will be "Name_Surame_HW2. e.g. *Lutfullah_Arici_HW2.*
. If you do not obey the project sending rules 10% of your grade will be dismissed.
. You have to bundle your whole project folder into your HW2.zip file.
. If I extract your project file, then import to my environment and if it doesn't work, you will be graded on 50 not 100. (Double check. It saves life)
. Do HW by yourself.
. Be honest.