ITEC 2610-A *Assignment Five*

Due: December 7th

1. **Singly-linked Lists** Remember that a singly linked list defines a collection of nodes that are arranged in a linear order. The order in a linked list is determined by a reference in each node. The first and the last node of a linked list are called the head and tail of the list, respectively. Linked list operations usually include:

   1) ***insertFirst(e)*** *Insert element e to be the head of the linked list;*
   2) ***removeFirst()*** *Remove and return the first element from the linked list and an error would occur if the list is empty;*
   3) ***size()*** *Return the number of elements in the linked list;*
   4) ***isEmpty()*** *Return TRUE if the linked list is empty, FALSE otherwise;*
   5) ***search(e)*** *search for the first appearance of the element e in the list.*

   For each node in the list, the following methods are provided.

   1) ***getElement()*** *Return the element of the current node;*
   2) ***getNextNode()*** *Return the next node of the current node;*
   3) ***setElement(e)*** *Set the value of the element of the current node;*
   4) ***setNextNode(n)*** *Set the next node to node n.*

   A generic framework Java implementation of singly-linked lists is provided for this project. The class *LinkedList* including several methods to be implemented by you. The *main*() method is defined in the *Driver.java.* You are required to modify the code within this method to carry out a variety of different operations. As an example, the following code defines a linked list A and then do the operations: *insertFirst*(7) and *remove*(). The status after each operation is also displayed.

```
Object o;
LinkedList<Integer> A = new LinkedList<Integer>();
A.status("Define new Integer linked List A", null);
A.insertFirst(7);
A.status("A.insertFirst(7)", null);
o = A.removeFirst();
A.status("A.removeFirst()", o);
```

   One should obtain results shown below.

```
Operation: Define new Integer linked List A, Returns: null
Linked List Status: size = 0, isEmpty = true, Linked List Content: [ ]

Operation: A.insertFirst(7), Returns: null
Linked List Status: size = 1, isEmpty = false, Linked List Content: [ 7 ]

Operation: A.removeFirst(), Returns: 7
Linked List Status: size = 0, isEmpty = true, Linked List Content: [ ]
```

   You need to

   - Download the starter code from the course web site. Read the starter code and make sure you understand how it works before attempting to modify it.
   - Design modifications to the starter code for the methods that described in the Introduction section.
   - Update the starter code with your modifications.
   - Run the code and your simulation.

- Submit a printout of every file that you modified or added to the starter code.
- Submit a printout of the output from your test runs.

Test your implementation with the following code added in *Driver.java*, where a linked list of *Integer* should be defined and the following operations are to be performed with the linked list.

```
insertFirst(1);
insertFirst(2);
insertFirst(2);
insertFirst(3);
search(1);
search(2);
search(3);
search(5);
removeFirst();
removeFirst();
removeFirst();
removeFirst();
removeFirst();
```

2. **Multithread/Queue** Implement a `Queue` class whose add and remove methods are synchronized. Supply one thread, called the producer, which keeps inserting strings into the queue as long as there are fewer than ten elements in it. When the queue gets too full, the thread waits. As sample strings, simply use time stamps new `Date().toString()`. Supply a second thread, called the consumer, that keeps removing and printing strings from the queue as long as the queue is not empty. When the queue is empty, the thread waits. Both the consumer and producer threads should run for 100 iterations.