Week 13 Lab – **A Linked List of Integers**

Maximum Points = 10

File *IntList.java* contains definitions for a linked list of integers. The class contains an inner class *IntNode* that holds information for a single node in the list (a node has a value and a reference to the next node) and the following *IntList* methods:

- public IntList()—constructor; creates an empty list of integers
- public void addToFront(int val)—takes an integer and puts it on the front of the list
- public void addToEnd(int val)—takes an integer and puts it on the end of the list
- public void removeFirst()—removes the first value from the list
- public void print()—prints the elements in the list from first to last

File *IntListTest.java* contains a driver that allows you to experiment with these methods. Save both of these files to your directory, compile and run IntListTest, and play around with it to see how it works. Then add the following methods to the IntList class. [use the print method as a model for 1,2, &4; use the addToEnd method as a model for #3.] **For each, add an option to the driver to test it.**

1. **public int length()—returns the number of elements in the list**
2. **public String toString()—returns a String containing the print value of the list.**

IF YOU HAVE TIME:

3. public void removeLast()—removes the last element of the list. If the list is empty, does nothing.
4. public void replace(int oldVal, int newVal)—replaces all occurrences of oldVal in the list with newVal. Note that you can still use the old nodes; just replace the values stored in those nodes.

```
// ************************************************************
// FILE:  IntList.java
//
// Purpose: Defines a class that represents a list of integers
//
// ************************************************************
public class IntList
{
    private IntNode front;      //first node in list

    //----------------------------------------
    //  Constructor.  Initially list is empty.
    //----------------------------------------
    public IntList()
    {
      front = null;
    }

    //----------------------------------------
    //   Adds given integer to front of list.
    //----------------------------------------
    public void addToFront(int val)
    {
      front = new IntNode(val,front);
    }

    //---------------------------------------
    //   Adds given integer to end of list.
    //---------------------------------------
    public void addToEnd(int val)
    {
      IntNode newnode = new IntNode(val,null);

      //if list is empty, this will be the only node in it
```

```
      if (front == null)
          front = newnode;
      else
          {
            //make temp point to last thing in list
            IntNode temp = front;
            while (temp.next != null)
                temp = temp.next;
            //link new node into list
            temp.next = newnode;
          }
    }

    //-----------------------------------------
    //   Removes the first node from the list.
    //   If the list is empty, does nothing.
    //-----------------------------------------
    public void removeFirst()
    {
      if (front != null)
          front = front.next;
    }

    //-----------------------------------------------
    //   Prints the list elements from first to last.
    //-----------------------------------------------
    public void print()
    {
      System.out.println("---------------------");
      System.out.print("List elements: ");

      IntNode temp = front;

      while (temp != null)
          {
            System.out.print(temp.val + " ");
            temp = temp.next;
          }
      System.out.println("\n---------------------\n");
    }

    //*************************************************************
    // An inner class that represents a node in the integer list.
    // The public variables are accessed by the IntList class.
    //*************************************************************
    private class IntNode
    {
      public int val;         //value stored in node
      public IntNode next;    //link to next node in list

    //-----------------------------------------------------------------
    // Constructor; sets up the node given a value and IntNode reference
    //-----------------------------------------------------------------
      public IntNode(int val, IntNode next)
      {
          this.val = val;
          this.next = next;
      }
    }
}
```

```
// ************************************************************
//   IntListTest.java
//
//   Driver to test IntList methods.
// ************************************************************

import java.util.Scanner;

public class IntListTest
{
    private static Scanner scan;
    private static IntList list = new IntList();

    //-----------------------------------------------------------------
    // Creates a list, then repeatedly prints the menu and does what
    // the user asks until they quit.
    //-----------------------------------------------------------------
    public static void main(String[] args)
    {
      scan = new Scanner(System.in);
      printMenu();
      int choice = scan.nextInt();
      while (choice != 0)
          {
            dispatch(choice);
            printMenu();
            choice = scan.nextInt();

          }
    }

    //---------------------------------------
    //  Does what the menu item calls for.
    //---------------------------------------
    public static void dispatch(int choice)
    {
      int newVal;
      switch(choice)
          {
          case 0:
            System.out.println("Bye!");
            break;

          case 1:    //add to front
            System.out.println("Enter integer to add to front");
            newVal = scan.nextInt();
            list.addToFront(newVal);
            break;

          case 2:   //add to end
            System.out.println("Enter integer to add to end");
            newVal = scan.nextInt();
            list.addToEnd(newVal);
            break;

          case 3:  //remove first element
            list.removeFirst();
            break;

          case 4:  //print
```

```
            list.print();
            break;
          default:
            System.out.println("Sorry, invalid choice");
          }
    }

    //----------------------------------------
    //  Prints the user's choices
    //----------------------------------------
    public static void printMenu()
    {
      System.out.println("\n   Menu   ");
      System.out.println("   ====");
      System.out.println("0: Quit");
      System.out.println("1: Add an integer to the front of the list");
      System.out.println("2: Add an integer to the end of the list");
      System.out.println("3: Remove an integer from the front of the list");
      System.out.println("4: Print the list");

      System.out.print("\nEnter your choice: ");
    }

}
```

(Due before end of the day on Friday, April 14, 2011) Submit your .java files containing your program to the dropbox in WebCT.

Grades are determined using the following scale:

- Runs correctly……:___/3
- Correct output……..……:___/2
- Design of output……:___/1
- Design of logic……:___/2
- Standards……:___/1
- Documentation……...:___/1