**CS 1181**

**Programming Assignment 2**

**NOTICE: There are NO static variables allowed in this lab. The only class that a static method is allowed is in the TestDriver class. The only static method that you actually really need is main.**

PURPOSE: The purpose of this lab is to give you an opportunity to create your own linked list data structure and implement all the methods specified below to do list operations.

Lab Description: The following specifies the linked list class which you must implement. Only implement the methods listed below. **Do NOT add any additional member variables.**

class OrderedLinkedList

{

private class Node

{

// NOTE: The member variables are public so the methods

// of the OrderedLinkedList class have direct access to them

// NO setters and getters are needed – more efficient access.

public String payload;

public int keyValue;

public Node next;

// Explicit value constructor for the Node class

public Node(String payload, int value) { …. }

}

private Node first; // This is the **only member variable** allowed.

// **You can NOT add any other member variables!!!**

public OrderedLinkedList ( ) { … } // default constructor

public boolean empty( ) { … } // returns true if list is empty

public String getFirst( ) throws Exception { … }

// The first node is removed from the list and its payload is returned.

// An exception with a meaningful error message is thrown if the list is empty.

public String getLast( ) throws Exception { … }

// The last node is removed from the list and its payload is returned.

// An exception with a meaningful error message is thrown if the list is empty.

public void insert(String payload, int key) throws Exception { … }

// The insert method creates a new node, puts the input parameter values

// into the new node, and then inserts the node into the list **in sorted order**

**// based on the *keyValue***. For example, if a list has 3 nodes with key values

**//** of 5, 10, and 15, insertion of a new node which has a key value of 12 would

**//** require that the new node was inserted after the node with key value of 10.

**//** If the key being inserted is already in the list, throw an Exception with a

// meaningful error string. Make sure that you test all possible cases for insertion.

public void remove(int key) throws Exception { … }

// Traverse the list, keeping track of where the precedessor for each node

// is located. When the value is found in some node in the list, make the

// predecessor node bypass the node which holds the value. If the key value

// is not found, throw an Exception that indicates the error has occurred.

public int listCount( ) { … } // Returns the number of nodes in the list.

public String getValue(int key) throws Exception { … }

// Traverse the list until you locate the requested key value. Return the

// payload from the node which has the requested key value. If the key value

// is not found, throw an Exception that indicates the error has occurred.

public String toString( ) { … }

// Override the toString method and create a string that contains the information

// from all the nodes currently in the list. The string should be constructed such

// that the data from each node will appear on a separate line. If the list is empty

// the returned string should indicate that the list was empty.

}

Implement all the methods of this class and **write a test program** which **thoroughly** tests all of the methods in this class. Carefully consider each method and what meaningful test cases are needed to fully test each method. Make sure to **identify each test case** in your test driver class with a comment. You can write your test program any way you like, but your program must **thoroughly** exercise your class. It can be interactive with the user, or just have some specific hardwired test cases. Make sure that the output generated by your test program clearly indicates what each of your test cases is doing and what the results are. By running your test program, you should be convinced that every method of your class is running correctly under all circumstances. Your grade will partially be determined by how good of a job you do in testing your class.

**Documentation**

You must provide complete documentation for the OrderedLinkedList class. Javadoc comments must be provided at the class level that include the author plus an overview of the purpose of the class. Javadoc comments must be provided for each method of the OrderedLinkedList class. The method level comments must include parameter explanations, return explanations, and exception explanations. Any preconditions should be explained with the parameters and exceptions, and postconditions should be explained with the return.

You must generate the Javadoc documentation from your comments. You do this by clicking on **Run** in the NetBeans menu bar. Then click on Generate Javadoc. An HTML page will appear that contains all the comments extracted from the classes in your project.

**Development Strategy**

Setup your project with the OrderedLinkedList class with all the methods specified containing no code so that the class compiles initially. You will need dummy return statements in some of the methods to get things to compile. Set up your test driver program to create an OrderedLinkedList object. Work on the insert method first followed by the toString method. Once you get those two methods written, you can start adding test cases to your test program to add items to the list and then use the toString method to return the content of the list to be displayed by your test program. Debug and test these methods first before moving on to the other methods. Once these methods are working, then implement and test one OrderedLinkedList class method at a time.

**Submitting**

Submit only 2 files to the drop box. Copy and paste your source code into ONE text file (Wordpad or Notepad). Put the OrderedLinkedList code first, followed by the test program code. Upload your ONE text file to the Pilot drop box. Save your lab assignment by creating one ZIP file of your entire project. Upload your ZIP file to the Pilot drop box before the lab due date. Ask your TA for help if you have any questions.

**Failure to properly submit these files will result in a grade of zero! If your project does not compile, you will receive a zero. Use of any static variables will result in an automatic 25% point penalty. Use of static methods in any class other than your test program will result in an automatic 25% point penalty.**

**Adding any member variables to the OrderedLinkedList class will result in a 20% point deduction.**

**Rubric:**

* **OrderedLinkedList class (25 pts)**
  + **The Node Class (1 pt)**
  + **Constructor (1 pt)**
  + **Empty method (1 pt)**
  + **GetFirst method (2 pts)**
  + **GetLast method (3 pts)**
  + **Insert method (5 pt)**
  + **Remove method (4 pt)**
  + **ListCount method (2 pt)**
  + **GetValue method (3 pt)**
  + **toString method (3 pt)**
* **Exception Handling (5 pts)**
  + **GetFirst and GetLast methods throw Exception (2 pts)**
  + **GetValue and Remove methods throw Exception (2 pts)**
  + **Test program catches and handles all the exceptions (1 pt)**
* **Your Test Program (20 pts)**
  + **Tests all methods when list is empty (3 pts)**
  + **Inserts into the list at front, in middle, at end (3 pts)**
  + **Removes objects from your list at front, in middle, at end (3 pts)**
  + **Searches for objects that are at front, in middle, at end (3 pts)**
  + **Tests remove and search of key not in list (2 pts)**
  + **Tests getFirst and getLast (3 pts)**
  + **Test program output clearly identifies test cases and results (3 pts)**
* **Commenting and Code Formatting (10 pts)**
  + **Class and method level Javadoc for the OrderedLinkedList class (4 pts)**
  + **Class level Javadoc for the test program. (1 pt)**
  + **Test program has comments specifying all test cases. (3 pts)**
  + **Generated Javadoc HTML displays all Javadoc comment content. (2 pts)**