**CSCD 300 Homework One**

**Rules**

**If you have skipped one or more video lectures relevant to this homework assignment without an acceptable excuse, the instructor will refuse to answer ANY questions about this homework. An acceptable excuse is usually related to an emergency or a documented excuse. Please refer to the syllabus for more information regarding this rule.**

Warmup/Readiness assignment - Linked List Basics and Exceptions (100 points)

Javadoc Standard Documentation \*NOT\* Required -- basic documentation is (all source files must include author name and a description of the class(es) contained in the file; any code another developer might not understand at first look should also be documented). Also, follow Java's naming conventions, utilize whitespace/indentation liberally in your code.

**Purpose**

The purpose of this assignment is to allow you to demonstrate basic knowledge of linked lists and exceptions. These operations include (directly or indirectly): inserting, deleting, finding and modifying and "printing." This assignment should be done by you without outside help (other than from your instructor). **If you are unable to complete this assignment in the allotted time, it is a strong indication you are not ready for the material in this class and you MUST meet with the instructor to discuss your continued enrollment in the class.**

**Specifics**

* As defined in the MyLinkedList class, you have to use a dummy node for your **singly** LinkedList -- this will actually simplify the amount of code you need for many operations.
* Please implement all the methods in the MyLinkedList class except for the AddFirst() and toString() methods. The requirements and specifications for each method are provided in the comments above each method header.
* Please do NOT change the interface and signature (the list of arguments and their type, and return type) of the methods in the MyLinkedList class.
* Please do NOT change the provided AddFirst() and toString() methods in the MyLinkedList class.
* Please do NOT change any code in the provided MyLinkedListTester class. However, during your **testing and debugging**, you can comment out some method calls in the main() of MyLinkedListTester class in order to single out the method that you are currently debugging. **After finish debugging, please make sure all methods are called (uncommented) in the main() of MyLinkedListTester class, as it was initially provided by the instructor.**
* Please read and understand the design and implementation of the MyLinkedListTester class. So that you can imitate its design in other project when testing each method of your implementation.

**List of methods you have to implement**

1. public Object removeFirst() throws Exception
2. public boolean contains(Object o)
3. public boolean remove(Object o)
4. public boolean removeAllCopies( Object o )
5. public static MyLinkedList interleave(MyLinkedList A, MyLinkedList B)
6. public void add(int index, Object o)
7. public Object get(int index)
8. public Object remove(int index) throws IndexOutOfBoundsException
9. public boolean add(Object e)

**Rubrics:**

1. Each method implementation above weighs 7%. So it is 63% for all nine method implementations.
2. A report for this assignment weights 37%. A report is a separate PDF file that will be due days before the entire project is due. A separate assignment named **hw1Logic** will be created on canvas. Please turn in your report for this homework there. Please check the description of homework **hw1Logic on Canvas for more details.**

**EXTRA CREDIT (up to 10 points possible)**

You have to use the generic parameter E for the myLinkedList class and its ListNode class, as well as specified in the java documentation. You have to rewrite MyLinkedList class so it supports generics (the stuff with the Es as type parameters). This means that when creating a LinkedList you will specify the type of data it will hold (e. g. LinkedList<String> myList = new LinkedList<String>();) The previous code specifies your LinkedList will only ever contain String objects as its data. The compiler will enforce this, which is a wonderful thing.

In this case, you can rewrite/change the MyLinkedListTester class to support the generic parameter E testing. But test results(program outputs) should be same as what were provided for the case when generic parameter E is not used.

**If you do this extra credit, be sure and clearly document it at the top of your MyLinkedList class AND you leave a comment “Extra credits attempted” at the homework submission page. Failure to do so will result in no extra points awarded. Your teacher will NOT help you with the extra credit portion of this homework.**

**To Turn In**

Turn in all your source code (.java file) and a **screenshot(picture)** of your program output that was generated after you execute your program. Zip up those files into a single zip file and turn in on the **EWU Canvas** by going to CSCD300-02 course page on Canvas, then clicking Assignments🡪hw1->submit.

This assignment must be submitted in working order by the due data on the top of this assignment. Submit a zip file with your source files only. Source files are those that end in .java. Check your zip file before submitting and make sure it has only your source files. Do not submit .class files; they do the grader no good. You will not receive credit if your submission contains only .class files. Name your zip file with your last name, followed by the first initial of your first name, followed by hw1. For example, if you are John Smith, name you file as smithjhw1.zip.

The grader should be able to open your zip file, compile your code, and run your program from **the command line using commands, javac \*.java and java** **MyLinkedListTester** version 1.7 or greater (**so make sure you try this yourself before you submit**). **If your code has compile-time error when USING COMMAND LINE tool to compile, you get a ZERO for this homework!**

Get started right away on this assignment or you WILL NOT finish :-(

**CORRECT OUTPUT of the program is provided below. Please make sure your output matches the correct output.**

------------------testAddLast()----

{A}

{A->B}

{A->B->null}

{A->B->null->C}

-----------------------------

------------testRemoveFirst()------

{apple->bad->null}, removed:null

{bad->null}, removed:apple

{null}, removed:bad

{}, removed:null

java.lang.Exception: Exception: LinkedList is empty!

-----------------------------

----------------testContains()------

true

true

true

false

false

false

false

false

-----------------------------

----------------testRemove(Object)---

{2:Morning->3:Abby->4:Tim->5:Tom->6:Tony}

{2:Morning->3:Abby->4:Tim->5:Tom->6:Tony}

{2:Morning->3:Abby->4:Tim->5:Tom}

{2:Morning->3:Abby->4:Tim->5:Tom}

{2:Morning->3:Abby->5:Tom}

{null->apple->bad->null}

{apple->bad->null}

{apple->bad}

{}

-----------------------------

---------------Test RemoveAllCopies()---

{B->A->A->A}

{A->A->A}

{}

{A}

-----------------------------

-----------------testInterleave()---

{null->1:Good->apple->2:Morning->bad->3:Abby->null->4:Tim->5:Tom->6:Tony}

{null->apple->bad->null}

-----------------------------

--------------testAddIndex()------

{A0->null->apple->bad->null}

{A0->null->apple->bad->null->A5}

{A0->null->apple->A3->bad->null->A5}

{B0}

java.lang.IndexOutOfBoundsException: Index Passed in not valid!

-----------------------------

--------------testRemoveIndex()------

{apple->bad->null}

{apple->bad}

java.lang.IndexOutOfBoundsException: Provided index is out of bounds! 2

{apple}

{}

-----------------------------

--------------testGetIndex()------

null

apple

bad

null

java.lang.IndexOutOfBoundsException: Provided index is out of bounds! 4

4:Tim

-----------------------------