Assignment 3

Doubly-Linked Lists

Introduction

This assignment requires you to implement a list interface using a doubly-linked list implementation. You will write and test non-static methods in the `A3LinkedList` to implement the basic operations of a linked list for Part 1. In Part 2, you will implement two advanced linked list operations, `insertInto` and `rotate`.

Although there is a tester provided for this assignment, it does not include a comprehensive set of sets for each method. You should add your own tests to test cases not considered.

Note: The automated grading of your assignment will include different and additional tests to those found in the A3Tester.java file. You are expected to write additional tests until you are convinced each method has full test coverage. The `displayResults` and test coverage videos provide more information about code testing.

Objectives

Upon finishing this assignment, you should be able to:

- Draw a complete memory trace for a doubly-linked list;
- Write a reference-based implementation of the List ADT;
- Write your own tests using the `displayResults` method.

Submission and Grading

Submit `A3LinkedList.java` to the BrightSpace assignment page. Remember to click submit afterward. You should receive a notification that your assignment was successfully submitted.

If you chose not to complete some of the methods required, you **must** provide a stub for the incomplete method(s) in order for our tester to compile. If you submit files that do not compile with our tester, you will receive a zero grade for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files. Additionally, your code must not be written to specifically pass the test cases in the tester, instead, it must work on all valid inputs. We may change the input values during grading and we will inspect your code for hard-coded solutions. This video explains stubs.

Be sure you submit your assignment, not just save a draft. ALL late and incorrect submissions will be given a ZERO grade. A reminder that it is OK to talk about your assignment with your classmates, but not to share code electronically or visually (on a display screen or paper). We will be using plagiarism detection software.

Instructions

Part 1:

1. Download all of the `.java` files found in the **Assignments -> Assignment 3** page on BrightSpace.
2. Read through the documentation provided in the `A3List.java` interface, there is a lot of very useful information that will help you with your implementation of each method.
3. Compile and run `A3Tester.java`. Work through implementing each method at a time. Debug the method until all of the tests pass for that method before proceeding to the next one.

Part 2:

1. For `insertInto` and `rotate`, make sure you come up with a strategy before writing any code.

2. Draw out your strategy similar to we have drawn out linked lists with boxes and arrows in the lectures and labs. Think about if any temporary reference variables will be needed, and about the order of operations required for your strategy to work. You will save a lot of time if you draw out your strategy before writing any code. I recommend drawing out how the list is updated after each line of code you write. It helps keep track of everything as you progress through the problem.
3. Remember to test that the reference arrows have been correctly updated after calling the methods. The linked lists for this assignments can be traversed from front to back or from back to front.

CRITICAL: Any compile or runtime errors will result in a **zero grade** (if the tester crashes, it will not be able to award you any points for any previous tests that may have passed). Make sure to compile and run your program **before** submitting it!