University of New Brunswick
CS1083 – Open Access Version

Module 9 Assignment

The purpose of this assignment is to give you practice:

- writing Java code to work with linked lists.

For all assignments:

- Follow the instructions in the document "Java Coding Guidelines.pdf" available within the "Start Here" module, under the "Assignments" topic.
- For each .java file you create, include a javadoc comment (one that begins with /**) at the beginning of the file that describes that Java class. This comment block should include a line that begins with @author followed by your name and student number on the same line. (Note: inclusion of this comment is part of the instructions given in "Java Coding Guidelines.pdf")
- Include comments throughout your programs to explain any non-obvious portions of your code.
- It is recommended that you create a separate folder on your computer for each assignment. You might even wish to create separate sub-folders within an assignment folder if the assignment has multiple parts. Keeping your work organized makes it easier to find things later when you want to review what you have done.
- Few things in life are more frustrating than losing your work while working on an assignment. Get in the habit of saving frequently when working on an assignment. Also, regularly make backup copies of any files you create as part of your work for this course.
- Each module has a link to the discussion forum where you can ask questions and seek help in completing your assignments.
- Submitting your assignment involves creating a pdf file and uploading that single file to the assignment drop box on D2L. In general, that file will tend to include all Java code you wrote for the assignment, plus any output from running your programs that the assignment instructions specify you should capture. Specific submission instructions are included at the end of each assignment.
- To create the pdf file for each assignment, begin by opening a new document using the word processing program of your choice. Save the document using the name “Your Name CS1083 Module x Assignment Submission.docx” Replace x with the correct module number. “docx” may be different depending on which word processing software you choose to use.

- At the beginning of your submission document enter your name, your student number, CS1083, the assignment name (this one is “Module 2 Assignment”), and the date. It doesn’t matter if the date is when you started working on the assignment or when you submit it – either will do.
- You can add content to your submission document as you work on the various questions in the assignment. Clearly label each part of the assignment (“Part A” etc.) in your document. When your document is complete, save / export as a pdf file. Always make sure your pdf file opens properly before uploading it.
- To include Java code in your submission document, copy all the text in your .java file and then paste that text into your submission document. Use a monospaced font (e.g.: Consolas, Courier) for your code to maintain proper indentation.
- To include output from running your program in your submission document, you have two choices. You can either (a) copy and paste the text from your command prompt window, or (b) capture a screen shot of the output and include that as a picture / image in your submission document. Either approach is fine.
- To copy text from your command prompt window, try selecting the desired text and then pressing either command-c (Mac) or control-c (Windows or Linux). If you have issues, you can always use Google to see how to do this on your specific type of computer.
- To capture a screen shot of a selected portion of your command prompt window, try command-shift-4 (Mac), WindowsKey-shift-s (Windows), or shift-PrtScrn (Linux).
- Once you have pasted your output text into your submission document, make sure it is in a monospaced font (e.g.: Consolas, Courier) so you retain the alignment of output.

Part A – Testing the LinkedList Class

The following files are provided for download earlier in this module:

- **Node.java** – A simple class for linked list nodes, where each node contains a person's name as a String.
- **LinkedList.java** – Contains some simple functionality for managing a linked list of Node objects.
- **ListTest.java** – Performs simple testing for some of the functionality in the LinkedList class.

All three of these classes will be updated as part of this assignment.

Part A involves updating the **ListTest** class to perform the following additional testing on the LinkedList class. In your output, clearly label the output from each test; if I don't specify what to provide as output, decide for yourself what appropriate output should be:

- After displaying the empty list, attempt to remove a name from the empty list. Place this call to the remove() method in an 'if-else' statement and either print "Correct – remove() returned false" or "Failed – improper result returned by remove()".
- Call getHead() on the empty list and display whether the result is null (null is correct, anything else is an error).
- Confirm that isEmpty() returns true when the list is empty.
- Add "Tom" to the empty list, then test in this order: display(), getHead(), isEmpty(), remove("Tom"), and isEmpty() again. In each case provide appropriate output to indicate whether each method behaved correctly.
- After adding the five names to the list and displaying them (ListTest already does this), call remove() three times as follows (with a display() after each to confirm whether the remove worked properly, and also print something like "Attempting to remove Ben" before each of these tests):
    i. Attempt to remove a node from the middle of the list;
    ii. Attempt to remove the first node in the list; and
    iii. Attempt to remove the last node in the list.

Execute your updated ListTest class at this point and capture the output so you can insert it into your assignment submission document.

Part B – Enhancing the Linked List

Update and test the **LinkedList**, **Node**, and **ListTest** classes as follows:

NOTE: For Part B, I recommend creating a new copy of those three java files in a separate directory from the work you did for Part A. That way you don't lose the work you did for Part A – for instance, in case you realize you want to modify what you did for Part A.

- Turn the list into a doubly linked list so Nodes are linked in both directions, forward and backward. This entails:
    - Adding a “previous” pointer to each Node object, along with getPrevious() and setPrevious() methods in the Node class;
    - Adding a “tail” pointer in the LinkedList class, along with a getTail() method; and
    - Updating whatever existing methods in the LinkedList class should be changed so the values of “tail” and the “previous” pointers will be maintained appropriately during all actions on the LinkedList.
    - Make sure all of the existing tests in ListTest execute properly after making those updates to Node and LinkedList. If your updates have been done correctly, this should be as simple as executing your ListTest class from Part A with your updated Node and LinkedList classes.
- Add a new instance variable called “size” to the LinkedList class. This variable must keep track of the number of Nodes in the list at all times.
    - Update the LinkedList constructor to set this variable appropriately when a new list is created.
    - Add a getSize() method that accesses the value of this variable. *(Note: It would not be appropriate to have a mutator method for this variable.)*
    - Update whatever existing methods in the LinkedList class should be changed so the value of “size” will be maintained appropriately during all actions on the LinkedList.
    - Add getSize() calls to your ListTest class as follows:
        - When the list is empty. Use an if statement to print either "Correct – size() returns 0" or "Incorrect – size() should return 0"
        - In Part A you added "Tom" to the empty list. Call getSize() when only this one node is in the list and use an if statement to print either "Correct – size returns 1" or "Incorrect – size() should return 1"

        - In Part A you added five names to the list. Call getSize() after doing so and use an if statement to print either "Correct – size returns 5" or "Incorrect – size() should return 5"

- Add the following methods to the LinkedList class. Make sure all instance variables in the Node and LinkedList classes are handled appropriately by these methods:
    - displayInReverse() – Displays the list in the same fashion as the current display() method, but in reverse order.
    - Add tests for displayInReverse() in ListTest when (a) the list is empty, (b) the list has one node, and (b) the list has five nodes.
    - insertAtHead(String nameIn) – Inserts a new Node with the given name at the beginning of the list. There is no need to check whether this operation keeps the list in sorted order, nor is there any need to check whether this operation results in duplicate names in the list.
    - insertAtTail(String nameIn) – Inserts a new Node with the given name at the end of the list. There is no need to check whether this operation keeps the list in sorted order, nor is there any need to check whether this operation results in duplicate names in the list.
    - removeAtHead() – removes the first Node in the list. Returns true if successful, false otherwise (which should only happen when attempting to removeAtHead() from an empty list).
    - removeAtTail() – removes the last Node in the list. Returns true if successful, false otherwise (which should only happen when attempting to removeAtTail() from an empty list).
    - Add tests for insertAtHead(), insertAtTail(), removeAtHead(), and removeAtTail() in ListTest for when (a) the list is empty, and (b) the list contains one or more nodes (your choice). Within the ListTest code, be sure to position these tests so you don't interfere with any tests that require the nodes to be in order – such as when you're testing insertInOrder().

(Submission instructions are provided on the next page)

Submission Instructions

Include the following in your submission document:

- Your name, student number, CS1083, Module 9 Assignment, and the date
- Part A:
  - Complete code for the updated ListTest class
    (The Node and LinkedList classes are unchanged for Part A, so these unchanged versions do not need to be included in your submission document)
  - Output from executing the updated ListTest class
- Part B:
  - Complete code for the updated ListTest, LinkedList, and Node classes
    (Note – These are the versions you produced for Part B)
  - Sample output from executing the updated (for Part B) ListTest class

D2L DROPBOX SUBMISSION INSTRUCTIONS

Remember to save / export your submission document as a pdf file, and then upload that one pdf file to D2L as follows:

1. In the top-navigation bar on the course screen, select 'Assessments' and then 'Assignments'.
2. Select the assignment title and follow the instructions to upload your submission document.