Learning Outcomes

- Linked lists
- Handling user input
- Conditionals

Introduction

You have been hired by Ms. List (first name Doubly, middle name Linked), the CEO at the renowned 'Linked List Company' to build a new program for their dedicated customers. This new product is called the 'Doubly-Linked List Editor', and is to be written in Java (convenient, I know).

The idea is that a user can run the program and build their very own doubly-linked list one node at a time. The user should also be able to remove nodes, reverse the doubly-linked list, delete the current doubly-linked list, and print the contents of the doubly-linked list all through user input. Ms. List provides you with a file, 'DoublyLinkedListEditor.java', that handles collecting and handling user input. You just need to write the functionality in the DoublyLinkedList.java and Node.java files.

Provided Files

The following is a list of files provided to you for this assignment. **Do NOT alter these files**.

- DoublyLinkedListEditor.java (containing the main method)
- TestDoublyLinkedList.java to test your code

Classes to Implement

For this assignment, you must implement two java classes: **Node** and **DoublyLinkedList**. Follow the guidelines for each class below.

In these classes, you can implement more private (helper) methods if you want to. You may not, however implement more public methods.

You may not add instance variables other than the ones specified below nor change the variable types or accessibility (i.e. making a variable public when it should be private). Penalties will be applied if you implement additional instance variables or change the variable types of modifiers from what is described here.

Node.java

This class represents a single node that will be used in the doubly-linked list.

The class must have the following public variables:

- int value (the value stored by the linked list)
- Node next (the 'next' pointer)
- Node prev (the 'prev' pointer)

The class must have a single constructor:

- public Node(int value) [constructor]
    - Set the instance variable 'value' to the passed value
    - Set both pointers to null

DoublyLinkedList.java

This class represents the doubly-linked list.

The class must have the following **private** variables:

- Node head (keeps track of the head node)
- Node tail (keeps track of the tail node)

The class must have the following **public** methods:

- public void add(int value)
    - Adds a new node with the given value to the end of the doubly linked list. If the list is empty, the new node becomes both the

head and the tail. Otherwise, the new node is linked as the next node of the current tail, and the tail is updated to the new node.

- public boolean remove(int value)
    - Removes the first node in the list that contains the specified value. It searches for the node with the given value, updates the links of neighboring nodes to bypass the node being removed, and adjusts the head or tail if necessary. Returns **true** if the node was found and removed, otherwise returns **false**.
- public void reverse()
    - Reverses the order of the nodes in the doubly-linked list. It iterates through the list, swapping the next and previous pointers of each node. After the reversal, it updates the head to the new first node.
- public void print()
    - Prints the values of all the nodes in the doubly linked list from head to tail. It iterates through the list, printing the value of each node followed by a space, and ends with a new line.
- public void deleteList()
    - Deletes the entire doubly-linked list by setting both the head and tail pointers to null, effectively removing all references to the nodes and allowing them to be garbage collected.

The class has the following methods provided to you:

- public void printToString(StringBuilder sb)
    - Appends the values of all the nodes in the doubly-linked list to the provided StringBuilder object. It iterates through the list, appending each node's value followed by a space. This method is used in the TestDoublyLinkedList class (so you don't necessarily need to understand it, but please leave it there so you'll pass the given tests).

    - Copy and paste the following method into your DoublyLinkedList code:

```java
// helper method to print to a StringBuilder (for testing purposes)
    public void printToString(StringBuilder sb) {
        Node current = head;
        while (current != null) {
            sb.append(current.value).append(" ");
            current = current.next;
        }
    }
```

Once completed, you can run the provided 'DoublyLinkedListEditor.java' file to interact with your functional Doubly-Linked list editor!

Marking Notes

Functional Specifications

- Does the program behave according to specification?
- Does the program produce the correct output and pass all tests?
- Are the classes implemented properly?
- Does the code run on another machine (i.e., is anything hardcoded? It shouldn't be)
- Does the code produce compilation or runtime errors?
- Does the program follow all stated rules and match the specifications laid out in the 'Classes to Implement' section?

Non-Functional Specifications

- Are there comments throughout the code?
- Are variables within the methods given appropriate and meaningful names?
- Is the code clean and readable with proper indenting and white space?
- Is the code consistent regarding formatting and naming conventions?
- Submission errors (i.e., missing files, too many files, etc.) will receive a penalty of 5%.
- Including a 'package' line at the top of a file will receive a penalty of 5%.

** Remember you must do all the work on your own. Do not copy or even look at the work of another student. All submitted code will be run through similarity detection software.

Rules

- Please only submit the files specified below.
- Do not upload the .class files! A 5% penalty will be applied for this.
- Submit your assignment on time. Late submissions will receive a penalty of 10% per day up to three days.
- Forgetting to submit is not a valid excuse for submitting late.
- Submissions must be done through Gradescope.
- Assignment files are not to be emailed to the instructor or TAs. They will not be marked if sent by email.

Testing your Code

The TestDoublyLinkedList class tests the functionality of your methods (so you can see if your code is working properly). These tests are similar, but not identical, to the tests that will be used to grade your work. You do not need to understand all the code presented in the TestDoublyLinkedList.java file. Just compile and execute it.

Test 1: add

Test 2: remove

Test 3: removing a non-existing node

Test 4: reverse

Test 5: printing list contents

Test 6: deleteList

Test 7: Adding and removing all nodes (linked list should be empty)

Test 8: reversing an empty list

To test your code, compile and execute TestDoublyLinkedList.java. If everything is working properly, you should see:

```
Test 1 Passed
Test 2 Passed
Test 3 Passed
Test 4 Passed
Test 5 Passed
Test 6 Passed
Test 7 Passed
Test 8 Passed
```

Otherwise, check while tests you are failing and that should give you an indication of where you should be focusing your attention.

Files to Submit

- Node.java
- DoublyLinkedList.java

** Note that you do not need to submit the DoublyLinkedListEditor.java file.

Using the Tests

To use the TestDoublyLinkedList.java file provided, place the TestDoublyLinkedList.java file in the same location on your machine as your DoublyLinkedList.java, Node.java, and DoublyLinkedListEditor.java files and recompile and run the TestDoublyLinkedList.java file. You should be passing all tests to achieve full marks.

Grading Criteria

Total Marks [6]

Passing custom tests (similar but not identical to the ones provided) **[5.1]**

- There will be 8 tests of increasing value.

Following document specifications [0.9]

- Does your code comply with all ‘non-functional specifications’ discussed above?

Good luck, and have fun!