# Module 2 Assignment: Singly-Linked List

Marcar esta página



Coding Assignment fecha límite Aug 21, 2023 05:50 -05

## Singly-Linked List

For this assignment, you will be implementing a *Singly-Linked List* with a head and tail reference. Recall that a Linked List is a collection of nodes, with each node containing a data item and reference(s) to other node(s). In a Singly-Linked List, each node will only reference the next node. Your Singly-Linked List should follow the requirements stated in the javadocs of each method you are to implement.

**IMPORTANT:**

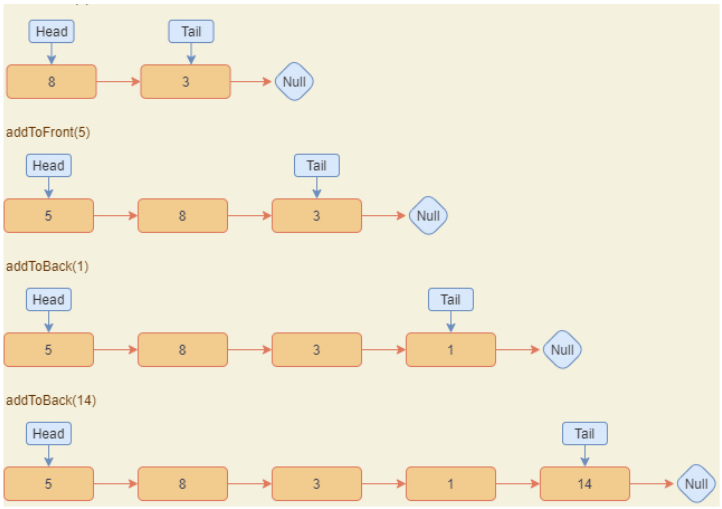
- **You will be given 5 attempts on this assignment, with a 30 minute cooldown between submissions.**
- **Please run your code before each submission to ensure that there are no formatting errors! If there are formatting errors in your code, your code will not be graded and a submission attempt will be logged. For more information, please review the Vocareum overview below.**

## Nodes

A **SinglyLinkedListNode** class is provided to you and will be used to represent the nodes in your Singly-Linked List. This file should be treated as **read-only** and should not be modified in any way. This SinglyLinkedListNode class contains getter and setter methods to access and mutate the structure of the nodes. Please make sure that you understand how this class works, as interaction with this class is crucial for this assignment.

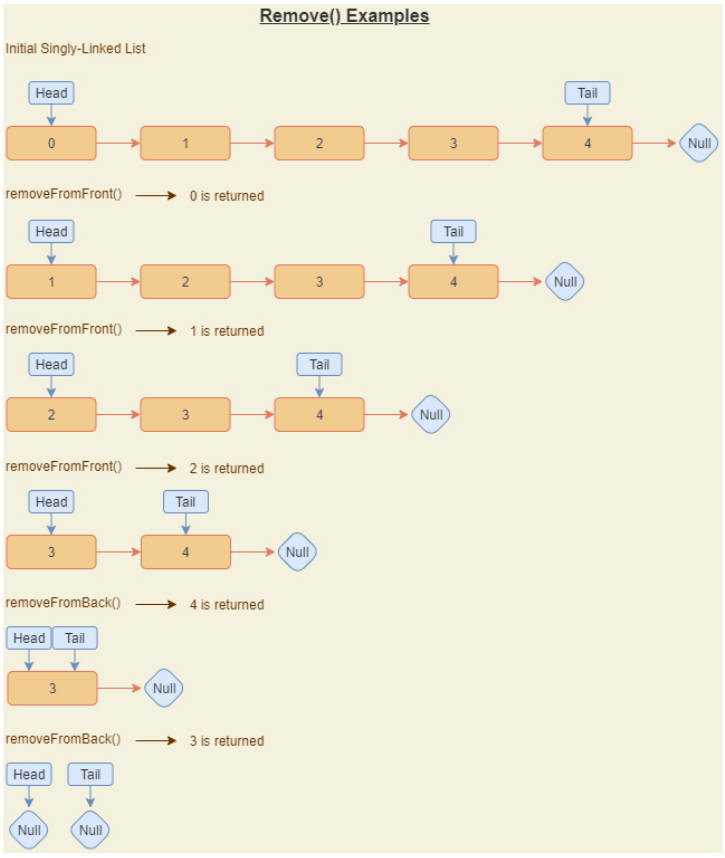
## Adding

You will implement two *add()* methods. One will add to the front and one will add to the back. Be sure to correctly reassign the head and/or tail pointers when adding an element.




### Removing

You will also implement two *remove()* methods. One will remove from the front and one will remove from the back. Be sure to correctly reassign the head and/or tail pointers when removing an element.




### Garbage Collection

Java will automatically mark objects for garbage collection based on whether there is any means of accessing the object. In other words, if we want to remove a node from the Singly-Linked List, we must remove all references to that node. What the "next" reference of that node points to does not particularly matter. As long as no references can reach the node, the node will be garbage collected eventually.

### General Tips

- Be sure to consider all edge cases! For example, when you add to the front or back of an empty Singly-Linked List, both the head and tail should be pointing to the same node. Similarly, when removing from the front or back of a Singly-Linked List containing only one node, both the head and tail should be set to null.

- In *add()* methods, if the passed in data is null, an exception should be thrown (see details in javadocs) and your Singly-Linked List should **not** be modified in any way. Where is the most optimal location in the method to check for this situation? Should this check be performed at the beginning or at the end?

to check for this situation? Should this check be performed at the beginning or at the end?

- We highly recommend copying the starter code and working in your preferred IDE in order to have access to features such as code completion, auto-formatting, and much more!

---

**Here are general assignment guidelines that should be followed.**

- Do not include any package declarations in your classes.
- Do not change any existing class headers, constructors, instance/global variables, or method signatures. For example, do not add throws to the method headers since they are not necessary. Instead, exceptions should be thrown as follows: `throw new InsertExceptionHere("Error: some exception was thrown");`
- All helper methods you choose to write should be made private. Recall the idea of Encapsulation in Object-Oriented Programming!
- Do not use anything that would trivialize the assignment. (e.g. Don't import/use java.util.ArrayList for an ArrayList assignment.)
- Always be very conscious of efficiency. Even if your method is to be $O(n)$, traversing the structure multiple times is considered inefficient unless that is absolutely required (and that case is extremely rare).
- If applicable, use the generic type of the class; do not use the raw type of the class. For example, use `new LinkedList<Integer>()` instead of `new LinkedList()`.

**Use of the following statements should be avoided at all times.**

| | | |
|---|---|---|
| package | System.arraycopy() | clone() |
| assert() | Arrays class | Array class |
| Thread class | Collections class | Collection.toArray() |
| Reflection APIs | Inner or nested classes | Lambda Expressions |

---

**The Vocareum (code editor) interface has six main components:**

- The **Drop-Down** in the top left. This lets you choose from multiple available files. Note that this drop-down will only be visible in assignments that require multiple files.
- The **Run** button. This will compile your code and run a file scan. Running your code will not count towards your total allowed submission attempts, therefore you are free to run as many times as needed.
- The **Submit** button. This will compile your code, run a file scan, grade your assignment, and output results to console. Note that for most assignments in this class, you will only be allowed a limited number of submissions. A submission is counted when the submit button is clicked, regardless of whether or not your code can compile or if there are any file issues. Therefore, we **highly recommend** that you run your code before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results.
- The **Reset** button. This will revert all your changes and reset your code to the default code template.
- The **Code Window**. This is where you will write your code. For large coding assignments, we highly recommend copying the starter code and working in your preferred IDE to have access to features such as code completion, auto-formatting, and much more!
- The **Output Window**. This window will appear whenever you run or submit your code and will display the output for you to view.

**For additional help, please visit the Vocareum information page located in the course information module!**

‹ Anterior  Siguiente ›