Linked Lists

Updated: 17th July, 2023

Aims

- To create a general purpose linked list class.
- To convert your previously implemented stack and queue to use the linked list.

Before the Practical

- Read this practical sheet fully before starting.
- Ensure that you have completed the activities from the previous practical, as you will build upon the classes developed last week.

Activities

1. Linked Lists Implementation

Let's create a Linked List class.

- Use the pseudocode from the lecture slides and the textbook to assist you in developing `DSAListNode` and `DSALinkedList`.
- *Be aware that the lecture slide pseudocode is for a **Single-Ended Linked List**. You must upgrade it to be a **Doubly-Linked, Double-Ended Linked List!***
- Start with a Single-ended Singly Linked List ensuring exception handling is implemented.
- Redevelop the list to be Doubly-Linked, Double-Ended - that is, maintain *both* a head and a tail pointer as member fields. This makes the linked list far more efficient for queues.
- Read the notes given below:

Note:

- Java Students: You may decide to make `DSAListNode` a separate .java file or place it as a *private class* within `DSALinkedList`
  Note that the latter will mean that you cannot return `DSAListNode` to any client/user of `DSALinkedList`. This is actually good design since it promotes information hiding (*how* the linked list works under the covers should not be something that clients should know about).

  Refer to the Lecture Slides for how to make a private inner class
- When implementing `insertFirst()`, use linked list diagrams like those in the lectures to help you decide how to maintain tail as well as head.
  Consider the following possible cases:

  (a) Empty list

  (b) One-item list

  (c) Multi-item list

  Some might end up working the same, but you still need to think it through.
- `insertLast()` is your next task - again, drawing diagrams can help.
- `removeFirst()` can be tricky - again, consider all the above three cases, but note that each case must be handled explicitly (in particular, removing the node in a one-item list is a special case since it is *both the first and last node*).
- Ensure that the `peek` and `remove` methods return the *value* of the ListNode!

2. Use DSALinkedList for DSAStack and DSAQueue

Copy your existing `DSAStack` and `DSAQueue` into your Practical 4 directory. You will convert them to use a *linked list* instead of an *array* data structure. This is simple as it is largely just a matter of hollowing-out the existing methods and calling the appropriate methods from your `DSALinkedList` class.

- For `DSAQueue`, have `enqueue()` perform an `insertLast()` in `DSALinkedList`.
  Conversely to `dequeue()`, use a combination of `peekFirst()` and `removeFirst()` to access the first element and remove it. You may reverse these operations and still achieve FIFO behaviour.

- For `DSAStack`, have `push()` perform an `insertFirst()`.
  Conversely to `pop()`, use a combination of `peekFirst()` and `removeFirst()` to access the first element and remove it. Again, you may reverse these operations and still achieve LIFO behaviour.
- Similar simplifications occur for `isEmpty()` and other methods. Meanwhile, some things can even be removed from your classes such as `isFull()`, `count`, `MAX_CAPACITY`, alternate constructor etc.

Remember to cite your previously submitted work, you may leave a comment in your code.

3. Interactive Menu for DSALinkedList

Now that we have a linked list, we want to setup an interactive menu system to explore the operations in a Linked List. Include at least the following options:

(a) InsertFirst/InsertLast on the list

(b) RemoveFirst/RemoveLast on the list

(c) Display the list

Submission Deliverable

- Your code and UML diagrams are due 2 weeks from your current tutorial session.
  - You will demonstrate your work to your tutors during that session
  - If you have completed the practical earlier, you can demonstrate your work during the next session
- You must **submit** your code and any test data that you have been using **electronically via Blackboard** under the *Assessments* section before your demonstration.
  - Java students, please do not submit the *.class files

Marking Guide

Your submission will be marked as follows:

- [8] Your DSALinkedList and DSAListNode are implemented properly.
- [7] Your DSAStack and DSAQueue have been re-developed to use linked lists. *Remember to cite your work.*
- [5] You have an interactive menu.

End of Worksheet