**Assignment: Implementation of Singly Linked List**

**Learning Objective:**

In this assignment, you will learn how to implement your own singly linked list from scratch. There is an in built class called LinkedList in Java does that the same job but in this assignment, you will learn how to implement a LinkedList on your own.

There are three types of linked list:

1. Singly Linked List – Has a link to the next node in the list
2. Doubly Linked List – Has two links: one to the previous node and one to the next node in the list
3. Circularly Linked List – Variation of the linked list in which the last node has a link to the first node.

However, we will focus on only the singly linked list in this assignment. At the end of this assignment you will learn how to perform several operations on a linked list such as insertion and deletion.

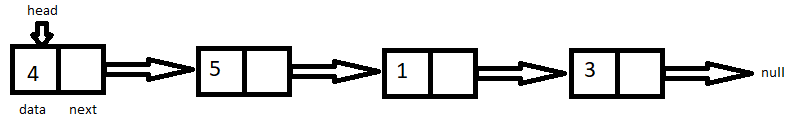
**Applications of Linked List in real world:**

1. Image viewer – Previous and next images are linked, hence can be accessed by next and previous button.
2. Previous and next page in web browser – We can access previous and next url searched in web browser by pressing back and next button since, they are linked as linked list.
3. Music Player – Songs in music player are linked to previous and next song. you can play songs either from starting or ending of the list.

**Tasks to be done:**

You have been given a ‘Node,java’ class that has two attributes ‘data’ of type integer and ‘next’ of type Node class. Do not modify anything in this file.

You have been given a skeleton code for the ‘SinglyLinkedList.java’ class. Your task is to fill in the methods as indicated in this java file. A singly linked list can be pictorially represented as follows:



You will implement six methods in the class called ‘SinglyLinkedList.java’.

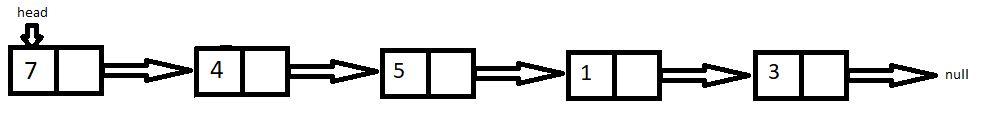
1. insertAtStart

Input parameters: int data

Return type: void

In this method, you will insert a node with the given data at the beginning of a linked list. Remember to update the head when you do so.

Ex: If we have inserted the node with data as 7 to our initial linked list using this method, then the updated linked list would be as follows:



1. insertAtEnd

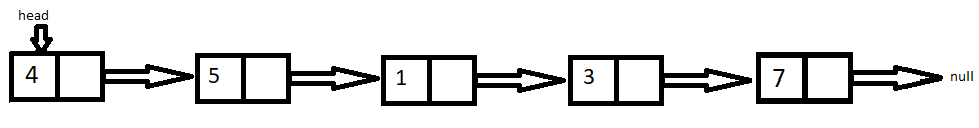
Input parameters: int data

Return type: void

In this method, you will insert a node with the given data at the end of the linked list. While doing this, there are two main points you need to consider:

* How do I handle the case when the linked list is empty?
* How to get to the end of the linked list?

Ex: If we have inserted the node with data as 7 to our initial linked list using this method, then the updated linked list would be as follows:



1. insertAtIndex

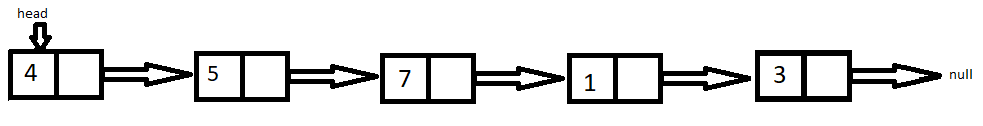
Input parameters: int index, int data

Return type: void

In this method, you will insert a node with the given data at the index specified by the input parameter ‘index’. While doing this, consider the following:

* Values for ‘index’ could be any integer (negative, zero or positive).
* How do I handle cases where I specify an index larger than the size of the linked list?
* We will be using zero indexing here. 0 refers to the first element, 1 refers to the second and so on.

Ex: If we have inserted the node with data as 7 and index as 2 to our initial linked list using this method, then the updated linked list would be as follows:



1. deleteAtIndex

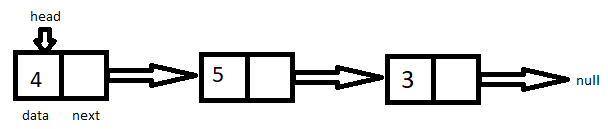
Input parameters: int index

Return type: void

In this method, you will delete a node specified by the index in the input parameter. While doing this, consider the following:

* Values for ‘index’ could be any integer (negative, zero or positive).
* How do I handle cases where I specify an index larger than the size of the linked list?
* We will be using zero indexing here. 0 refers to the first element, 1 refers to the second and so on.

Ex: If we have deleted the node at index 2 in our initial linked list using this method, then the updated linked list would be as follows:



1. isEmpty

Input parameters: None

Return type: Boolean

In this method, you need to check if the linked list is empty of not. You can use this method inside any of your other methods you have implemented to check if your linked list is empty. It should return either true or false.

1. lengthOfLinkedList

Input parameters: None

Return type: int

In this method, you will find out the length or the size of the linked list. It should return an integer indicating the number of nodes in the linked list.

Ex: For our initial linked list, the length should be returned as 4.

**Additional Pointers to help while coding:**

1. Make sure you update the head according to the requirement
2. Check for cases where your linked list is empty
3. Make sure you handle all other edge cases accordingly.

**More about in-built class LinkedList in Java:**

The methods implemented in this assignment are equivalent to the inbuilt methods in the LinkedList class. Few of the methods in the in-built LinkedList class of Java are as follows:

|  |  |  |
| --- | --- | --- |
| Return Type | Method | Description |
| void | addFirst (E e) | Inserts the specified element at the beginning of the list |
| void | addLast (E e) | Inserts the specified element at the end of the list |
| void | add (int index, E element) | Inserts the specified element in the specified index |
| E | remove(int index) | Removes the element at the specified position in the list |
| int | size() | Returns the number of elements in the list |
| boolean | isEmpty() | Checks if the linked list is empty or not |

These are equivalent to the methods we implemented in this assignment. You can learn more about the other methods available in the in-built LinkedList class of Java as an additional exercise.