Introduction to Linked List
Java

Linked List

LinkedList class Implementation (Collection Framework)

```java
import java.util.*;

class LL {
   public static void main(String args[]) {
       LinkedList<String> list = new LinkedList<String>();
       list.add("is");
       list.add("a");
       list.addLast("list");
       list.addFirst("this");
       list.add(3, "linked");
       System.out.println(list);

       System.out.println(list.get(0));
       System.out.println(list.size());
       list.remove(3);
       list.removeFirst();
       list.removeLast();

       System.out.println(list);
   }
}
```

Scratch Implementation (Important for BEGINNERS)

```
class LL {



   Node head;

   private int size;



   LL () {

       size = 0;

   }



   public class Node {

       String data;

       Node next;



       Node(String data) {

           this.data = data;

           this.next = null;

           size++;

       }

   }



   public void addFirst(String data) {

       Node newNode = new Node(data);

       newNode.next = head;

       head = newNode;

   }
```

```java
public void addLast(String data) {
    Node newNode = new Node(data);

    if(head == null) {
        head = newNode;
        return;
    }

    Node lastNode = head;
    while(lastNode.next != null) {
        lastNode = lastNode.next;
    }

    lastNode.next = newNode;
}

public void printList() {
    Node currNode = head;

    while(currNode != null) {
        System.out.print(currNode.data+" -> ");
        currNode = currNode.next;
    }

    System.out.println("null");
```

```
    }

    public void removeFirst() {
        if(head == null) {
            System.out.println("Empty List, nothing to delete");
            return;
        }

        head = this.head.next;
        size--;
    }

    public void removeLast() {
        if(head == null) {
            System.out.println("Empty List, nothing to delete");
            return;
        }

        size--;
        if(head.next == null) {
            head = null;
            return;
        }

        Node currNode = head;
        Node lastNode = head.next;
```

```
        while(lastNode.next != null) {
            currNode = currNode.next;
            lastNode = lastNode.next;
        }

        currNode.next = null;
    }

    public int getSize() {
        return size;
    }

    public static void main(String args[]) {
        LL list = new LL();
        list.addLast("is");
        list.addLast("a");
        list.addLast("list");
        list.printList();

        list.addFirst("this");
        list.printList();
        System.out.println(list.getSize());

        list.removeFirst();
```

```
        list.printList();



        list.removeLast();

        list.printList();

    }

}
```

How to insert in the middle of a Linked List (at a specified index ‘i’) ?

Scratch

```
public void addInMiddle(int index, String data) {

        if(index > size || index < 0) {

            System.out.println("Invalid Index value");

            return;

        }

        size++;



        Node newNode = new Node(data);

        if(head == null || index == 0) {

            newNode.next = head;

            head = newNode;

            return;

        }

         Node currNode = head;

        for(int i=1; i<size; i++) {

            if(i == index) {
```

```
                Node nextNode = currNode.next;

                currNode.next = newNode;

                newNode.next = nextNode;

                break;

            }

            currNode = currNode.next;

        }

    }
```

LinkedList class

```
import java.util.*;



class LL {

    public static void main(String args[]) {

        LinkedList<String> list = new LinkedList<String>();



        list.addFirst("shradha");

        list.addFirst("name");

        list.addFirst("my");

        System.out.println(list);



        list.add(2, "is");

        System.out.println(list);

    }

}
```

Homework Problems

1. Make a Linked List & add the following elements to it : (1, 5, 7, 3 , 8, 2, 3). Search for the number 7 & display its index.
2. Take elements(numbers in the range of 1-50) of a Linked List as input from the user. Delete all nodes which have values greater than 25.