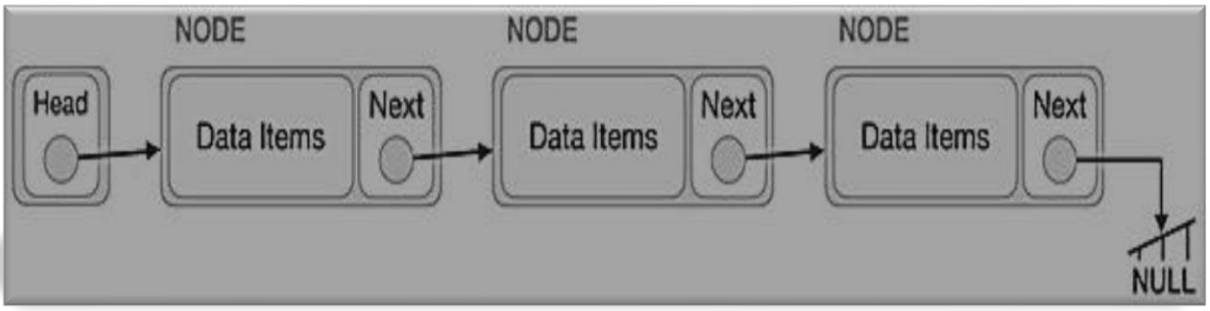
LINKED LIST DATA STRUCTURE (LECTURE 3)

Linked List Data Structure

- A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Linked lists are probably the second most commonly used general purpose storage structures after arrays
- Linked List is a very commonly used linear data structure which consists of group of nodes in a sequence. Each node holds its own data and the address of the next node hence forming a chain like structure. Linked Lists are used to create trees and graphs



Advantages of Linked Lists

- They are dynamic in nature which allocates the memory when required
- Insertion and deletion operations can be easily implemented
- Stacks and queues can be easily executed
- Linked List reduces the access time

Disadvantages of Linked Lists

- The memory is wasted as pointers require extra memory for storage
- No element can be accessed randomly; it has to access each node sequentially
- Reverse Traversing is difficult in linked list

Linked List Operations/ Functions

- **find()** - Find a link with a specified key value
- **insert()** - Insert links anywhere in the list
- **delete()** - Delete a link with the specified value
- **isEmpty()** - check if list is empty

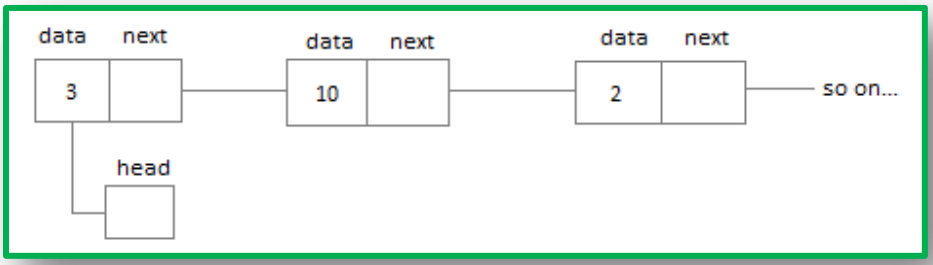
Types of Linked Lists

- There are 3 different implementations of Linked List available, they are:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

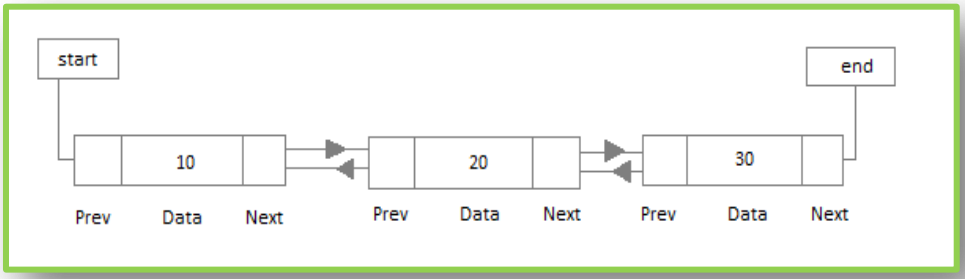
1. Singly Linked List

- Singly linked lists contain nodes which have a **data** part as well as an **address part** i.e. next, which points to the next node in the sequence of nodes



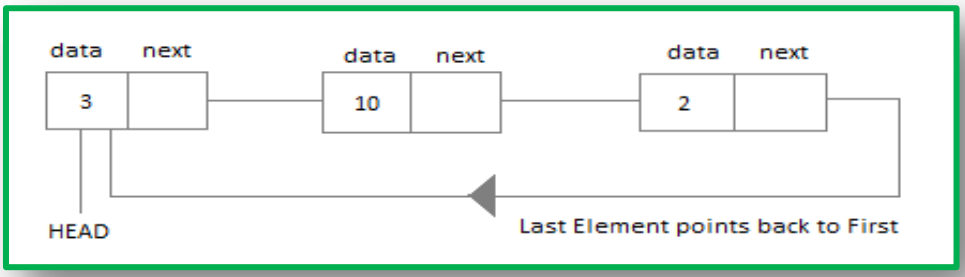
2. **Doubly Linked List**

- Each node contains a **data** part and two addresses, one for the **previous** node and one for the **next** node



3. **Circular Linked List**

- In circular linked list the last node of the list holds the address of the first node hence forming a circular chain.



Linked List implementation

```
class LinkedList {
	private Node head;

	public void insertFirst(int value) {
		Node node = new Node(value);
		node.next = head;
		head = node;
	}//insertFirst

	public void insertAt(int position, int value) {
		Node node = new Node(value);

		if (position == 1) {
			insertFirst(value);
		} else {
			Node temp = head;

			for (int i = 1; i < position-1; i++) {
				temp = temp.next;
			}//for

			node.next = temp.next;
			temp.next = node;
		}//if-else

	}//insertAt

	public void insertLast(int value) {
		Node node = new Node(value);

		if (head == null) {
			head = node;
		} else {
			Node temp = head;
```

```
                while (temp.next != null) {
                        temp = temp.next;
                }//while

                temp.next = node;
        }//if-else

    }//insertLast

    public void deleteAt(int position) {
        if (position == 1) {
                head = head.next;
        } else {
                Node temp = head;
                Node temp1;

                for (int i = 1; i < position-1; i++) {
                        temp = temp.next;
                }//for

                temp1 = temp.next;
                temp.next = temp1.next;
                temp1 = null;
        }
    }//deleteAt

    public void display() {
        Node node = head;

        while (node.next != null) {
                System.out.println(node.dataItem);
                node = node.next;
        }//while

        System.out.println(node.dataItem);
    }//display

    public boolean isEmpty() {
        return (head == null);
    }//isEmpty

}//class LinkedList
```

```
public class Node {
    public int dataItem;
    public Node next;

    public Node(int data) {
        dataItem = data;
        next = null;
    }//constructor

}//class
```

```
public class Main {

    public static void main(String[] args) {
        LinkedList list = new LinkedList();

        list.insertFirst(10);
        list.insertFirst(20);

        list.insertLast(70);
        list.insertLast(80);

        list.insertAt(1, 25);
        list.insertAt(3, 35);

        list.display();
        System.out.println("");
        list.deleteAt(1);
        list.deleteAt(4);

        list.display();
    }//main method

}//class
```