DSA

Module 9

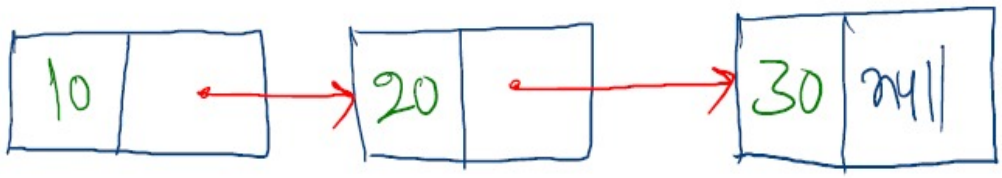
LinkedLists

Author
Srinivas Dande

9. Introduction to LinkedLists

- Linked List is Linear Data Structure.
- Linked List is Used to store collection of elements.
- LinkedLists elements can be accessed in sequential order only.
- LinkedLists elements will be stored with Node representaion.

Ex:



- Linked list has the following properties.
  - ✓ Successive elements are connected by pointers
  - ✓ The last element points to NULL
  - ✓ Can grow or shrink in size during execution of a program
  - ✓ Does not waste memory space. It allocates memory as list grows but takes some extra memory for pointers.
- Types of Linked Lists
  - ✓ Singly Linked Lists
  - ✓ Doubly Linked Lists
  - ✓ Circular Linked Lists

9.1 Why LinkedLists

- There are many other data structures that do the same thing as linked lists.
- Before discussing linked lists it is important to understand the difference between linked lists and arrays.
- Both linked lists and arrays are used to store collections of data, and since both are used for the same purpose, we need to differentiate their usage.
- That means in which cases arrays are suitable and in which cases linked lists are suitable.

Advantages of Arrays:

- Arrays are Used to store collection of elements of Same type
- Arrays elements are stored in contiguous locations
- Arrays elements will be stored with index representaion.
- Arrays allow Random access to elements.
- Search Operation is faster because we can apply Binary Search.
- Arrays are Cache Friendly.

DisAdvantages of Arrays:

- Arrays are fixed size.
  It means Once you create the array, you can not increse the size.
- Momory waste when you allocate more space and store less elements.
- Momory shortage when you allocate less space and store more elements.
  You need to resize the array when you want to store more elements.
  Resizing is an expensive again.
- Insert Operation is expensive because it takes time to shift elements
- Delete Operation is expensive because it takes time to shift elements

Advantages of LinkedLists:

- LinkedLists are Used to store collection of elements
- LinkedLists elements will not be stored in contiguous locations
- LinkedLists elements will be stored with Node representaion.
- LinkedLists allow Only sequential acess to elements, No Random access.
- Insert Operation is faster
- Delete Operation is faster
- LinkedLists are dynamic. i.e No Node Creation inadvance

DisAdvantages of LinkedLists:

- No Random access.
- Search Operation is expensive
  we can not apply Binary Search on LinkedList because of Sequential Access.
- Momoey wastage for Node addresses.
- LinkedLists are Not Cache Friendly.

Comparison of Linked Lists with Arrays & Dynamic Arrays

Parameter	Arrays	ArrayList	LinkedList
Get Elemenet	O(1)	O(1)	O(n)
Insert at Begining	O(n)	O(n)	O(1)
Delete at Begining	O(n)	O(n)	O(1)
Insert at End	O(1)	O(1)	O(n)
Delete at End	O(1)	O(1)	O(n)
Insert at Middle	O(n)	O(n)	O(n)
Delete at Middle	O(n)	O(n)	O(n)

Use-cases:

1. I am writing some programs which has to store very large number of elements.I may not get the contiguous memory because memory may be available as chunks. Which one to choose to store large number of elements.

   Ans: LinkedList

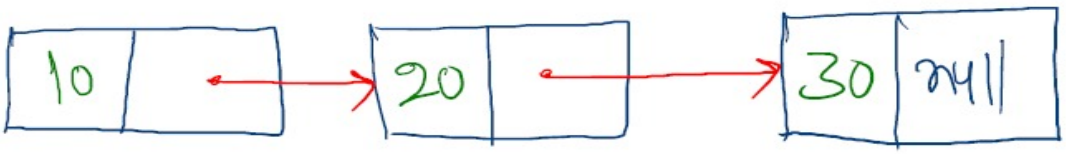
2. Stacks and Queues can be implemented with Arrays and LinkedLists.
   Ex: I want to Implement Round Rabin Scheduling.

   Ans: Use Queue with LinkedLists

9.2. Singly Linked List

- Singly Linked List consists of a number of nodes
- Each Node has will have 2 parts
  a) Data
  b) Address of Next Node
- Address of the last node in the list is NULL, which indicates the end of the list.

Ex:



- Following is a type declaration for a linked list:

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

        Node(int data) {
                this.data = data;
                this.next = null;
        }
}
```

- Main Linked Lists Operations
  - ✓ Traverse: Access the elements in the list
  - ✓ Count: returns the number of elements in the list
  - ✓ Find nth node from the end of the list
  - ✓ Insert: inserts an element into the list
  - ✓ Delete: removes the specified position element from the list

9.2.1. Traverse the LinkedList using Iterative Style

Lab1.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab1 {

	static void displayList(Node headNode) {

		Node currentNode = headNode;

		while(currentNode != null) {
			System.out.print(currentNode.data+"\t");
			currentNode=currentNode.next;
		}

		System.out.println("\n");
	}

	public static void main(String[] args) {

		Node head = new Node(10);
		head.next =  new Node(20);
		head.next.next =  new Node(30);
		head.next.next.next =  new Node(40);
		head.next.next.next.next =  new Node(50);

		displayList(head);

	}
}
```

9.2.2. Traverse the LinkedList using Recursive Style

Lab2.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab2 {

	static void displayList(Node currentNode) {

		//Base Condition
		if(currentNode==null)
			return ;

		System.out.print(currentNode.data+"\t");
		displayList(currentNode.next); //Recursive Call

		System.out.println("\n");
	}

	public static void main(String[] args) {

		Node head = new Node(10);
		head.next =  new Node(20);
		head.next.next =  new Node(30);
		head.next.next.next =  new Node(40);
		head.next.next.next.next =  new Node(50);

		displayList(head);

	}
}
```

9.2.3. Count the No. of Nodes in the LinkedList

Lab3.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab3 {
	static int length(Node headNode) {

		int length = 0;
		Node currentNode = headNode;
		while (currentNode != null) {
			length++;
			currentNode = currentNode.next;
		}
		return length;
	}

	public static void main(String[] args) {
		Node head = new Node(10);
		head.next = new Node(20);
		head.next.next = new Node(30);
		 head.next.next.next = new Node(40);
		 head.next.next.next.next = new Node(50);

		int len = length(head);
		System.out.println(len);

		Node myhead = null;
		int mylen = length(myhead);
		System.out.println(mylen);
	}
}
```

9.2.4. Insert the Node at the begining of LinkedList

Lab4.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab4 {
	static Node insertFirst(Node headNode,int data) {

		Node temp = new Node(data);
		temp.next = headNode;
		return temp;
	}
	static void displayList(Node headNode) {
		Node currentNode = headNode;
		while(currentNode != null) {
			System.out.print(currentNode.data+"\t");
			currentNode=currentNode.next;
		}
		System.out.println("\n");
	}
	public static void main(String[] args) {

			Node head = null;
			head = insertFirst(head,10);
			head = insertFirst(head,20);
			head = insertFirst(head,30);
			head = insertFirst(head,40);
			head = insertFirst(head,50);

			displayList(head);
	}
}
```

9.2.5. Insert the Node at the end of LinkedList

Lab5.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab5 {
	static Node insertLast(Node headNode,int data) {

		Node temp = new Node(data);
		if(headNode==null){
			return temp;
		}

		//To Reach the Last Node of LL
		Node currentNode = headNode;
		while(currentNode.next != null) {
			currentNode=currentNode.next;
		}

		currentNode.next = temp;

		return headNode;
	}

	static void displayList(Node headNode) {
		Node currentNode = headNode;
		while(currentNode != null) {
			System.out.print(currentNode.data+"\t");
			currentNode=currentNode.next;
		}

		System.out.println("\n");
	}
```

```
        public static void main(String[] args) {

                        Node head = null;
                        head = insertLast(head,10);
                        head = insertLast(head,20);
                        head = insertLast(head,30);
                        head = insertLast(head,40);
                        head = insertLast(head,50);

                        displayList(head);
        }
}
```

9.2.6. Insert the Node at given position of LinkedList

Lab6.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab6 {
        static Node insert(Node headNode,int position, int data) {

                Node temp = new Node(data);

                if(position==1){
                        temp.next = headNode;
                        return temp;
                }

                Node currentNode = headNode;
                for(int i=1;i<=position-2 && currentNode != null;i++) {
                        currentNode=currentNode.next;
                }
```

```
            if(currentNode==null) {
                return headNode;
            }

            temp.next=currentNode.next;
            currentNode.next = temp;

            return headNode;
        }
        static void displayList(Node headNode) {
            //Copy from Lab5
        }
        static Node insertLast(Node headNode,int data) {
            //Copy from Lab5
        }
        public static void main(String[] args) {
                    Node head = null;
                    head = insertLast(head,10);
                    head = insertLast(head,20);
                    head = insertLast(head,30);
                    head = insertLast(head,40);
                    head = insertLast(head,50);

                    displayList(head);
                    System.out.println("-----------------");

                    head = insert(head,3,99);
                    displayList(head);
                    System.out.println("-----------------");
                    head = insert(head,4,88);
                    displayList(head);
        }
}
```

9.2.7. Delete the First Node of LinkedList

Lab7.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab7 {

	static Node deleteFirst(Node headNode) {

		if(headNode==null)
				return null;

		Node newHead = headNode.next;
		headNode.next = null;
		return newHead;
	}

	static Node insertLast(Node headNode,int data) {
		//Copy from Lab5
	}

	static void displayList(Node headNode) {
		//Copy from Lab5
	}

	public static void main(String[] args) {

			Node head = null;
			head = insertLast(head,10);
			head = insertLast(head,20);
			head = insertLast(head,30);
			head = insertLast(head,40);
			head = insertLast(head,50);
```

```
                displayList(head);
                System.out.println("--------------");

                head= deleteFirst(head);
                displayList(head);
                System.out.println("--------------");
                head= deleteFirst(head);
                displayList(head);
        }
}
```

9.2.8. Delete the Last Node of LinkedList

Lab8.java

```
package com.jlcindia.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab8 {

        static Node deleteLast(Node headNode) {

                if(headNode==null)
                        return null;

                Node newHead = headNode.next;
                headNode.next = null;
                return newHead;
        }

        static Node insertLast(Node headNode,int data) {
                //Copy from Lab5
        }
```

```
        static void displayList(Node headNode) {
                //Copy from Lab5
        }

        public static void main(String[] args) {

                        Node head = null;
                        head = insertLast(head,10);
                        head = insertLast(head,20);
                        head = insertLast(head,30);
                        head = insertLast(head,40);
                        head = insertLast(head,50);

                        displayList(head);
                        System.out.println("--------------");

                        head= deleteLast(head);
                        displayList(head);
                        System.out.println("--------------");
                        head= deleteLast(head);
                        displayList(head);
        }

}
```