

Data Structures

LAB No: 02

Instructor: Marina Gul

Objective of Lab No. 2:

After performing lab2, students will be able to:

- Implementation of Linked list

Practice 02

We have already covered Linked List. Some of the basic operation of linked list are following

- a) Create linked list
- b) Implementation of Insert, show, isEmpty and delete operation

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

    //Constructor
    Node(int data)
    {
        this.data=data;
        this.next=null;
    }
}
class LinkedList{

    Node head;
    void addToBack(int data)
    {
        Node node=new Node(data);

        if(head==null)
            head=node;
        else{
            Node n=head;
            while(n.next!=null)
                n=n.next;
            n.next=node;
        }
    }
    void printList() {
        Node = head;
        while (node.next != null){
            System.out.print(node.data + " ");
        }
    }
}
```

```

        node = node.next;
    }
    System.out.print(node.data + " ");
}

void addToFront(int data)
{
    Node =new Node(data);

    node.next=head;
    head=node;

}

void addMiddle(int index,int data)
{
    Node =new Node(data);
    node.data=data;
    node.next=null;
    Node n=head;
    for(int i=1;i<index-1;i++)
    {
        n=n.next;
    }
    node.next=n.next;
    n.next=node;
}
void removeFromFront()
{
    head=head.next;
}
void removeFromBack()
{
    Node n=head;
    while (n.next.next!=null)
        n=n.next;
    n.next=null;
}
void removeAt(int index)
{
    if (index==1)
        removeFromFront();
    else{
        Node n=head;
        Node temp=null;
        for(int i=1;i<index-1;i++)
        {
            n=n.next;
        }
        temp=n.next;
        n.next=temp.next;
    }
}
boolean isEmpty(){
return head == null;
}

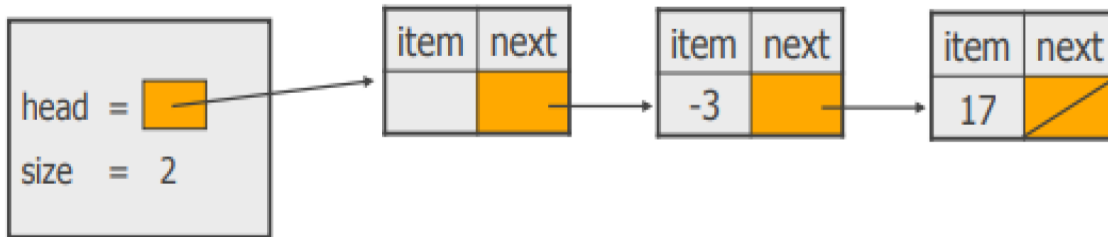
```

```
}  
  
public class Main {  
    public static void main(String[] args) {  
        LinkedList ll=new LinkedList();  
        ll.addToBack(5);  
        ll.addToBack(19);  
        /*  
        ll.addLast(19);  
        ll.show();  
        System.out.println("");  
        ll.addStart(20);  
        ll.show();  
  
        System.out.println("");  
        ll.addMiddle(2,200);  
        ll.show();  
  
        System.out.println("");  
        ll.deleteStart();  
        ll.show();  
        */  
    }  
}
```

Exercise

1. Implement a function to search for a specific element in a linked list.
2. Write a function to find the length of a linked list.
3. In this task you will write a program that implements a variant of a linked list. This variant has a dummy node pointed to by the head link as shown in the following figure:

Linked list with a dummy first node:



This trick will allow your code to be a little simpler, not requiring a special case for add or remove operations. Your constructor method will be:

```
public LinkedList() {
    head = new Node(null);
    size = 0;
}
```

You need to write a class called `LinkedList` that implements the following `List` interface:

```
// a list interface
public interface List {
    public boolean isEmpty();
    // returns true if the list is empty, false otherwise

    public int size();
    // returns the number of items in the list

    public void add(Object item);
    // adds an item to the list
    // item is added at the end of the list

    public void add(int index, Object item);
    // adds an item to the list at the given index
    // item is added at the given index;
    // the indices start from 1.

    public void remove(int index);
    // removes the item from the list that has the given index
}
```

```

public void remove(Object item);
// removes an item from the list
// removes the first item in the list whose equal method matches
// that of the given item

public List duplicate();
// creates a duplicate of the list
// returns a copy of the linked list

public List duplicateReversed();
// creates a duplicate of the list with the nodes in reverse order
// returns a copy of the linked list with the nodes in reverse order
}

```

In addition to the interface, your LinkedList class needs to implement a toString() method that prints the list in the format

[size: the_size_of_the_list - item1, item2,]

Note: Your Node class should be an inner class within the LinkedList class. **Make sure your class implements the interface as specified, i.e. your class should begin with** public class LinkedList implements List.