

```

1 package LinkedList;
2 public class DSA1_LinkedList {
3     public static class Node{
4         private int data;
5         private Node next;
6     }
7     public static class LinkedList{
8         private Node head;
9         private Node tell;
10        private int size;
11
12        public void addLast(int val){
13            Node node=new Node();
14            node.data=val;
15            node.next=null;
16            if(size==0){
17                head=tell=node;
18            } else{
19                tell.next=node;
20                tell=node;
21            }
22            size++;
23        }
24
25        public void addFirst(int val){
26            Node node=new Node();
27            node.data=val;
28            node.next=head;
29            head=node;
30            size++;
31        }
32        public void addAtIndex(int val,int index){
33            Node node=new Node();
34            node.data=val;
35            Node temp=head;
36            for(int i=0;i<index-1;i++){
37                temp=temp.next;
38            }
39            node.next=temp.next;
40            temp.next=node;
41            size++;
42        }
43        public void removeFirst(){
44            if(this.size==0){
45                System.out.println("LinkedList is empty");
46            } else if(size==1){
47                head=tell=null;
48                size=0;
49            } else{
50                head=head.next;
51                size--;
52            }
53        }
54    }
55    public void removeLast(){
56        Node temp=head;
57        for(int i=0;i<size-2;i++){
58            temp=temp.next;
59        }
60        tell=temp;
61        tell.next=null;
62        size--;
63    }
64
65    void display(){
66        Node temp=head;
67        while(temp!=null){
68            System.out.print(temp.data+"->");
69            temp=temp.next;
70        }
71        System.out.println();
72    }
73    private void reverseLinkedList(){
74        int i=0;
75        int j=size-1;
76        while(i<j){
77            Node left=getNodeAt(i);
78            Node right=getNodeAt(j);
79            int temp=left.data;
80            left.data=right.data;
81            right.data=temp;
82            i++;
83            j--;
84        }
85    }
86    public void size(){
87        System.out.println("Size of LinkedList->"+size);
88    }
89    public int getFirst(){
90        return head.data;
91    }
92    public int getLast(){
93        return tell.data;
94    }
95    public int getElement(int index){
96        if(size==0){
97            System.out.println("List is empty");
98            return 0;
99        } else if(index<0 || index>size){
100            System.out.println("Invalid Input");
101            return 0;
102        }
103        Node temp=head;
104        for(int i=0;i<index;i++){
105            temp=temp.next;
106        }
107        return temp.data;
108    }
109    private Node getNodeAt(int index){
110        if(size==0){
111            System.out.println("List is empty");
112            // return 0;
113        } else if(index<0 || index>size){
114            System.out.println("Invalid Input");
115            // return 0;
116        }
117        Node temp=head;
118        for(int i=0;i<index;i++){
119            temp=temp.next;
120        }
121        return temp;
122    }
123
124    public void removeIndexElement(int index){
125        Node temp=head;
126        for(int i=0;i<index-1;i++){
127            temp=temp.next;
128        }
129        temp.next=temp.next.next;
130        size--;
131    }
132
133    public void normalPrint(int getElement){
134        System.out.print("*****");
135        System.out.print("1st Element ->"+getFirst());
136        System.out.print("\nLast Element ->"+getLast());
137        System.out.print("\nElement "+getElement+" at Index ->"+getElement(getElement));
138        System.out.print("\nSize ->"+size);
139        System.out.print("*****");
140    }
141 }
142
Run | Debug
143 public static void main(String[] args) throws NullPointerException{
144     LinkedList lst=new LinkedList();
145     for(int i=1;i<10;i++){
146         lst.addLast(i*10);
147     }
148     lst.size();
149     lst.addFirst(64);
150     lst.addLast(34);
151     lst.addAtIndex(43, 8);
152     lst.display();
153     lst.removeLast();
154     lst.removeIndexElement(2);
155     lst.reverseLinkedList();
156     lst.display();
157     lst.normalPrint(5);
158 }

```