DOUBLY LINKED LIST

public static class DLLi {  
  
  
 int size;  
  
 DLLi() {  
 this.size = 0;  
 }  
  
 class Node {  
 int val;  
 Node next;  
 Node prev;  
  
 Node(int val) {  
 this.val = val;  
 this.next = null;  
 this.prev = null;  
 size++;  
 }  
 }  
  
 Node head;  
 Node tail;  
  
 public void addFirst ( int val){  
 Node newn = new Node(val);  
 if (head == null) {  
 head=newn;  
 }  
 else{  
 newn.next = head;  
 head.prev = newn;  
 head = newn;  
 }  
  
 }  
  
 public void addLast ( int val){  
 Node newn = new Node(val);  
 if (head == null) {  
 addFirst(val);  
 }  
 Node curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newn;  
  
 }  
 public void insert(int index,int val){  
 if(head==null){  
 addFirst(val);  
 }  
 Node newn=new Node(val);  
 Node curr=head;  
 for(int i=1;i<index-1;i++){  
 curr=curr.next;  
  
 }  
 newn.next=curr.next;  
 curr.next.prev=newn;  
 curr.next=newn;  
 newn.prev=curr;  
 }  
 public void delFirst(){  
 if(head==null){  
 return;  
 }  
 if(head.next==null){  
 head=null;  
 }  
 head=head.next;  
 }  
 public void delLast(){  
 if(head==null){  
 return;  
 }  
 if(head.next==null){  
 head=null;  
 }  
 Node curr=head;  
 while(curr.next.next!=null){  
 curr=curr.next;  
 }  
 curr.next=null;  
 }  
 public void delete(int index){  
 if(head==null){  
 return;  
 }  
 if(head.next==null){  
 head=null;  
 }  
 Node curr=head;  
 for(int i=1;i<index-1;i++){  
 curr=curr.next;  
 }  
 curr.next=curr.next.next;  
 curr.next.next.prev=curr;  
 }  
  
 public void display () {  
 if (head == null) {  
 System.*out*.println("empty list");  
 return;  
 }  
 if (head.next == null && head.prev == null) {  
 System.*out*.println(head.val);  
 return;  
 }  
 Node curr = head;  
  
 while (curr != null) {  
 System.*out*.print(curr.val + "<-->");  
 curr = curr.next;  
 }  
 System.*out*.println("null");  
 }  
  
}  
  
  
  
  
 public static void main(String[] args) {  
 DLLi list = new DLLi();  
 list.addFirst(6);  
 list.addFirst(5);  
 list.addFirst(4);  
 list.addFirst(3);  
 list.addLast(7);  
 list.addFirst(2);  
 list.addFirst(1);  
 list.insert(4,11);  
// list.display();  
 list.delFirst();  
 list.delLast();  
 list.delete(3);  
 list.display();  
  
  
 }