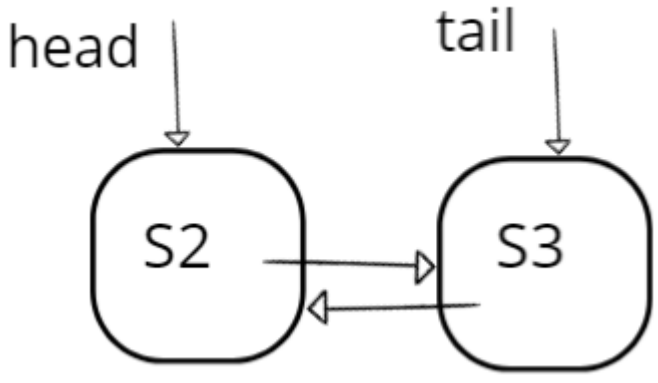
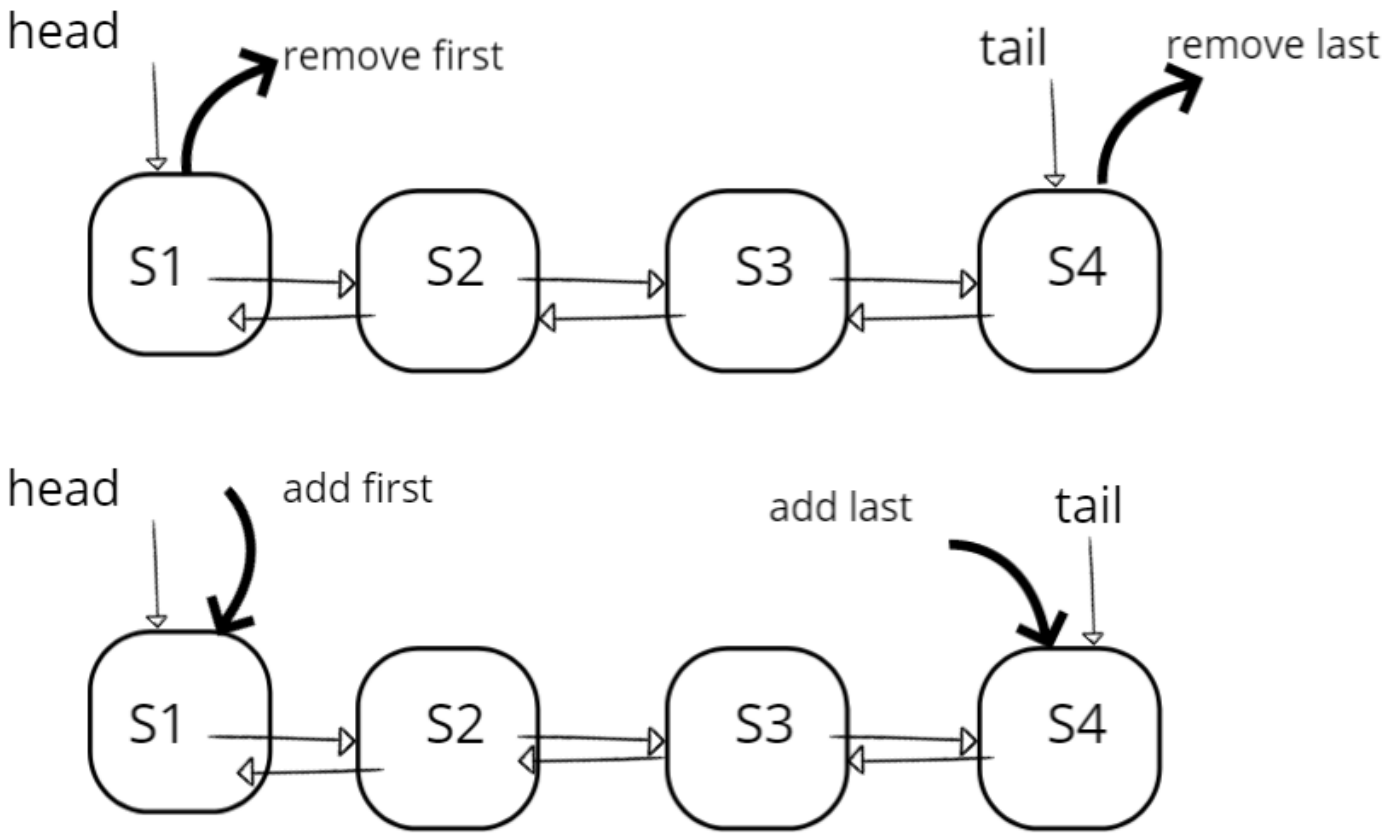
```
import java.util.NoSuchElementException;

public class SimpleLinkedList<E> {

    private Node head;//mientras no se les asigne un valor estan en null
    private Node tail;//hasta la primera inserccion ya valen diferente a null
    private int size;

    public SimplyLinkedList() {
        size = 0;
    }
    /**
     * this class keeps track of each element information
     * @author java2novice
     *
     */
    private class Node {
        E element;
        Node next;//estos nodos valen null
        Node prev;

        public Node(E element, Node next, Node prev) {
            this.element = element;
            this.next = next;
            this.prev = prev;
        }
    }
    /**
     * returns the size of the linked list
     * @return
     */
    public int size() { return size; }

    /**
     * return whether the list is empty or not
     * @return
     */
    public boolean isEmpty() { return size == 0; }

    /**
     * adds element at the starting of the linked list
     * @param element
     */
   public void addFirst(E element) {
        System.out.println("Adding first*********");
        Node tmp = new Node(element, head, null);
        System.out.printf("TEMP: %s, %s\n", tmp.element, tmp);
        System.out.printf("HEAD ANTES: %s\n", head);
        System.out.printf("TAIL ANTES: %s\n", tail);
        if(head != null ) { head.prev = tmp;}
        head = tmp;
        if(tail == null) { tail = tmp;}
```

```
        System.out.printf("HEAD DESUES: %s\n", head);
        System.out.printf("TAIL DESPUES: %s\n", tail);
        size++;
        System.out.println("adding: "+element);
    }

    /**
     * adds element at the end of the linked list
     * @param element
     */
   public void addLast(E element) {
            System.out.println("Adding last*********");
        Node tmp = new Node(element, null, tail);
        System.out.printf("TEMP: %s, %s\n", tmp.element, tmp);
        System.out.printf("HEAD ANTES: %s\n", head);
        System.out.printf("TAIL ANTES: %s\n", tail);
        if(tail != null) {tail.next = tmp;}
        tail = tmp;
        if(head == null) { head = tmp;}
        System.out.printf("HEAD DESUES: %s\n", head);
        System.out.printf("TAIL DESPUES: %s\n", tail);
        size++;
        System.out.println("adding: "+element);
    }

    /**
     * this method walks forward through the linked list
     */
    public void iterateForward(){
        System.out.println("iterating forward..");
        Node tmp = head;
        while(tmp != null){
            System.out.println(tmp.element);
            tmp = tmp.next;
        }
    }

    /**
     * this method walks backward through the linked list
     */
    public void iterateBackward(){
        System.out.println("iterating backword..");
        Node tmp = tail;
        while(tmp != null){
            System.out.println(tmp.element);
            tmp = tmp.prev;
        }
    }



    /**
```

```
     * this method removes element from the start of the linked list
     * @return
     */
    public E removeFirst() {
        if (size == 0) throw new NoSuchElementException();
        Node tmp = head;
        head = head.next;
        head.prev = null;
        size--;
        System.out.println("deleted: "+tmp.element);
        return tmp.element;
    }

    /**
     * this method removes element from the end of the linked list
     * @return
     */
    public E removeLast() {
        if (size == 0) throw new NoSuchElementException();
        Node tmp = tail;
        tail = tail.prev;
        tail.next = null;
        size--;
        System.out.println("deleted: "+tmp.element);
        return tmp.element;
    }

    public static void main(String a[]){

        DoublyLinkedList<Integer> dll = new DoublyLinkedList<Integer>();
        dll.addFirst(10);
        dll.addFirst(34);
        dll.addLast(56);
        dll.addLast(364);
        dll.iterateForward();
        dll.removeFirst();
        dll.removeLast();
        dll.iterateBackward();
    }
}
```




head
remove first
tail
remove last
S1
S2
S3
S4
head
add first
add last
tail
S1
S2
S3
S4