Lab5:

## Singly Linked Lists

* Node
* Attributes of Singly Linked List (head,tail,size)
* Methods (size,isEmpty,first,last,addFirst,addLast,removeFirst)
* Inserting an Element at the Head of a Singly Linked List
* Inserting an Element at the Tail of a Singly Linked List
* Removing an Element from a Singly Linked List
* public class SinglyLinkedList<E> {  
    
   private static class Node<E>{  
    
   private E element;  
   private Node<E> next;  
    
   public Node(E element, Node<E> next) {  
   this.element = element;  
   this.next = next;  
   }  
    
   public E getElement() {  
   return element;  
   }  
    
   public void setElement(E element) {  
   this.element = element;  
   }  
    
   public Node<E> getNext() {  
   return next;  
   }  
    
   public void setNext(Node<E> next) {  
   this.next = next;  
   }  
    
   }  
    
   private Node<E> head = null;  
   private Node<E> tail=null;  
   private int size=0;  
    
   public SinglyLinkedList() {  
   }  
    
   public boolean isEmpty(){return size==0;}  
    
   public int Size(){ return size;}  
    
   public E first(){  
   if(isEmpty())return null;  
   return head.getElement();  
   }  
    
   public E last(){  
   if(isEmpty())return null;  
   return tail.getElement();  
   }  
    
   public void addFirst(E element){  
   head= new Node<E>(element,head);  
   if(size==0)tail=head;  
   size++;  
   }  
    
   public void addLast(E element){  
   Node<E> newest=new Node<E>(element,null);  
   if(size==0)head=newest;  
   else tail.setNext(newest);  
   tail=newest;  
   size++;  
   }  
    
   public E removeFirst(){  
   if (isEmpty()) return null;  
    
   E deleted=head.getElement();  
   head=head.getNext();  
   size--;  
   if(size==0)  
   tail=null;  
    
   return deleted;  
   }  
    
   public String print (){  
   Node<E> i=head;  
   String all="";  
   while (i!=null){  
   all=all+i.getElement().toString()+"\n";  
   i=i.getNext();  
   }  
   return all;  
   }  
  }

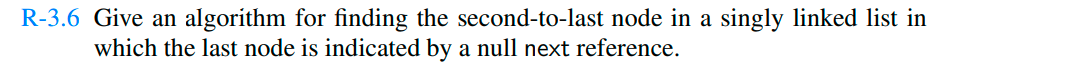
public class Lab5 {  
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list= new SinglyLinkedList<>();  
 SinglyLinkedList<Integer> list2=new SinglyLinkedList<>();  
  
 list.addFirst(11);  
 list.addFirst(12);  
 list.addLast(13);  
  
 //System.out.println(list.print());  
  
 int n=list.Size();  
 for (int i = 0; i <n ; i++) { // int x=list.removeFirst();  
 System.*out*.println(x);  
 list2.addLast(x);  
 }  
 System.*out*.println(list.Size());  
 System.*out*.println(list2.Size());  
 }  
}

Read from file:

import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
  
public class ReadTextToSingly {  
 public static void main(String[] args) {  
 SinglyLinkedList<String> list = new SinglyLinkedList<>();  
  
 try {  
 Scanner input = new Scanner(new File("D:\\Aya\\level2.2\\DS\\Lab5.txt"));  
 while (input.hasNext()) {  
 list.addFirst(input.nextLine());  
 }  
 System.*out*.println(list.print());  
 System.*out*.println(list.Size());  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
}

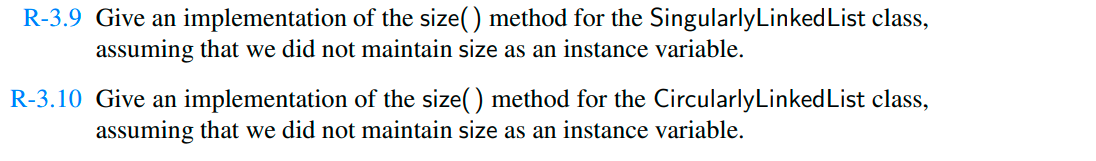
Tasks Lab5:

Try answering five of the following :



public E Second\_2\_Last()  
{  
 Node<E> newNode = head ;  
 while (newNode.getNext()!=tail)  
 {  
 newNode=newNode.getNext();  
 }  
 return (E) newNode.getElement();  
}

public class testsectolast {  
  
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println(list.print());  
 System.*out*.println("Second to Last = "+list.Second\_2\_Last());  
 }  
  
  
}



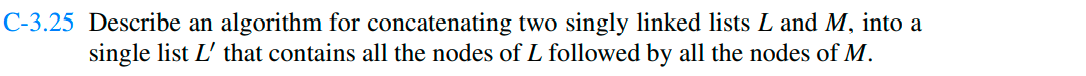
1.

public int SizeQ()  
{  
 Node<E> temp = head ;  
 int Size = 0 ;  
 if(head==null)  
 return Size ;  
 else  
 {  
 Size++ ;  
 while (temp.next!=null)  
 {  
 Size++;  
 temp = temp.next ;  
 }  
 }  
 return Size ;  
}

public class testsizeQ {  
   
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println("The Size is : "+list.SizeQ());  
 }  
   
}

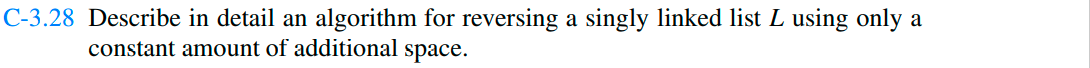
public void Rotate() {  
 if (head != null) {  
 Node current = head;  
 while (current.getNext() != null) {  
 tail = current;  
 current = current.getNext();  
 }  
 if (tail != null) {  
 tail.setNext(null);  
 current.setNext(head);  
 head = current;  
 }  
 }  
}

public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println(list.print());  
 list.Rotate();  
 System.*out*.println("After Rotation : \n");  
 System.*out*.println(list.print());  
}



public Node<E> concatenation(E head1 , E head2)  
{  
 Node<E> temp = null ;  
 if (head1==null)  
 return (Node<E>) head2;  
 if (head2==null)  
 return (Node<E>) head1;  
 temp= head.getNext();  
 while (temp.getNext()!=null)  
 temp = temp.next;  
 temp.next= tail.getNext();  
 return (Node<E>) head1;  
}

public class testconcatenation {  
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list = new SinglyLinkedList<>();  
 list.addLast(1);  
 list.addLast(2);  
 list.addLast(3);  
 list.addLast(4);  
 SinglyLinkedList<Integer> list2 = new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 list.concatenation(list.first(),list2.last());  
 System.*out*.println(list.print());  
 }  
}



public Node<E> Reverse()  
{  
 if (head==null)  
 {  
 return head;  
 }  
 Node<E> current = head ;  
 Node<E> previous = null ;  
 Node<E> next = current.next ;  
  
  
 while (current!=null)  
 {  
 next = current.next ;  
 current.next=previous ;  
 previous = current ;  
 current = next ;  
 }  
  
 head = previous ;  
 return previous ;  
}

public class Task5 {  
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println(list.print());  
 System.*out*.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  
 list.Reverse();  
 System.*out*.println(list.print());  
 }  
   
}