**CIS163 Lab on Double Linked list**

**Preparation:**

Attended class and review your notes on Double Linked list.

**Objectives** (after completing the lab you will be able to do:)

* To write simple methods the modify a linked list

**Activities:**

# ***Task’*** Cut and paste the code below into IntelliJ.

* Create the needed projects and packages.

# ***Task’*** Complete the following methods 5 methods.

# Note (code has been provided):

* public int getLen() {
* public void insertBefore (int index, String data) {
* public void insertAfter(int index, String data) {
* public String removeTop () {
* public boolean delAt(int index) {

# Get checked off by the instructor

# If you have time, ***Task’*** write JUnits to test out your list.

public class DNode {  
 private String data;  
 private DNode next;  
 private DNode prev;  
  
 public DNode(String data, DNode next, DNode prev) {  
 this.data = data;  
 this.next = next;  
 this.prev = prev;  
 }  
  
 public DNode() {  
 }  
  
 public String getData() {  
 return data;  
 }  
  
 public void setData(String data) {  
 this.data = data;  
 }  
  
 public DNode getNext() {  
 return next;  
 }  
  
 public void setNext(DNode next) {  
 this.next = next;  
 }  
  
 public DNode getPrev() {  
 return prev;  
 }  
  
 public void setPrev(DNode prev) {  
 this.prev = prev;  
 }  
}

public class *MyIllegalArgumentException* extends RuntimeException {  
}

public class DoubleLinkListLab {  
 private DNode top;

private DNode tail;  
  
 public DoubleLinkListLab() {  
 top = null;

tail = null;  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*  
 \* Determines the size, that is, the number of elements in the list  
 \*  
 \* @return the size of the list  
 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* public int getLen() {  
 *// place your code here  
 // GO FROM TAIL UP, NOT FROM TOP DOWN.* return 0;  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*  
 \* Inserts a node before a specific index. When the list is empty  
 \* that is, top = null, then the index must be 0. After the first  
 \* element is added, index must be: 0 <= index < size of list  
 \*  
 \* @param index a specific index into the list.  
 \*  
 \* @throws MyIllegalArgumentException if index < 0 or  
 \* index >= size of the list  
  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* public void insertBefore (int index, String data) {  
 *// place your code here* }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*  
 \* Inserts a node after a specific index. When the list is empty  
 \* that is, top = null, then the index must be 0. After the first  
 \* element is added, index must be: 0 <= index < size of list  
 \*  
 \* @param index a specific index into the list.  
 \*  
 \* @throws MyIllegalArgumentException if index < 0 or  
 \* index >= size of the list  
  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* public void insertAfter (int index, String data) {  
 *// place your code here* }  
  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*  
 \* Removes the top element of the list  
 \*  
 \* @return returns the element that was removed.  
 \*  
 \* @throws MyIllegalArgumentException if top == null, that is,  
 \* there is no list.  
 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* public String removeTop () {

*// place your code here*  
 return null;  
 }  
  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*  
 \* This Method removes a node at the specific index position. The  
 \* first node is index 0.  
 \*  
 \* @param index the position in the linked list that is to be  
 \* removed. The first position is zero.  
 \*  
 \* @throws MyIllegalArgumentException if index < 0 or  
 \* index >= size of the list  
 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* public String delAt(int index) {

*// place your code here*  
 return null;  
 }  
  
*// A simple testing program. Not complete but a good start.* public static void main (String[] args){  
 DoubleLinkListLab list = new DoubleLinkListLab();  
  
 list.display();  
 System.*out*.println ("Current length = " + list.getLen());  
  
 list.insertBefore(0, "apple");  
 list.insertBefore(0, "pear");  
 list.insertBefore(1, "peach");  
 list.insertBefore(1, "cherry");  
 list.insertBefore(3, "donut");  
 list.display();  
  
 list.insertAfter(0, "apple");  
 list.insertAfter(0, "pear");  
 list.insertAfter(1, "peach");  
 list.insertAfter(1, "cherry");  
 list.insertAfter(3, "donut");  
 list.display();  
  
 list.removeTop();  
 System.*out*.println("Deleted pos 4 with value of: " +list.delAt(4));  
 System.*out*.println("Deleted pos 2 with value of: " +list.delAt(2));  
 System.*out*.println("Deleted pos 0 with value of: " +list.delAt(0));  
 list.removeTop();  
 list.removeTop();  
  
 list.display();

System.*out*.println ("Current length = " + list.getLen());  
 }  
// Once you are done!

// Change this method to go from the tail up!

public void display() {  
 DNode temp = top;  
  
 System.*out*.println ("\_\_\_\_\_\_\_\_\_\_\_ List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 while (temp != null) {  
 System.*out*.println (temp.getData());  
 temp = temp.getNext();  
 }  
 }  
}

Output:

\_\_\_\_\_\_\_\_\_\_\_ List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Current length = 0

\_\_\_\_\_\_\_\_\_\_\_ List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

pear

cherry

peach

donut

apple

\_\_\_\_\_\_\_\_\_\_\_ List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

pear

pear

cherry

peach

donut

apple

cherry

peach

donut

apple

Deleted pos 4 with value of: apple

Deleted pos 2 with value of: peach

Deleted pos 0 with value of: pear

\_\_\_\_\_\_\_\_\_\_\_ List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cherry

peach

donut

apple

Current length = 4