# Data Structure Lab4 : Singly Linked List 2022-2023

## Topics: Implement Node Class and SinglyLinkedList Class

### 1. Node Class

```
java
Copy code
public class Node<E> {
    E element; // Data stored in this node
    Node<E> next; // Reference to the next node in the list

    public Node(E element, Node<E> next) {
        this.element = element;
        this.next = next;
    }

    public E getElement() {
        return element;
    }

    public Node<E> getNext() {
        return next;
    }

    public void setNext(Node<E> next) {
        this.next = next;
    }
}
```

---

2. SinglyLinkedList Class

```
java
Copy code
public class SinglyLinkedList<E> {
    private Node<E> head = null; // Head node of the list
    private Node<E> tail = null; // Tail node of the list
    private int size = 0; // Number of elements in the list

    public SinglyLinkedList() { }

    // isEmpty
    public boolean isEmpty() {
        return size == 0;
    }

    // size
    public int size() {
        return size;
    }

    // first
    public E first() {
        if (isEmpty()) return null;
        return head.getElement();
    }

    // last
    public E last() {
        if (isEmpty()) return null;
        return tail.getElement();
    }

    // addFirst
    public void addFirst(E element) {
        head = new Node<>(element, head);
        if (size == 0) tail = head; // If list was empty, tail is the same as head
        size++;
    }

    // addLast
    public void addLast(E element) {
        Node<E> newNode = new Node<>(element, null);
        if (isEmpty()) {
            head = newNode;
        } else {
            tail.setNext(newNode);
        }
        tail = newNode;
        size++;
    }

    // removeFirst
    public E removeFirst() {
        if (isEmpty()) return null;
        E removedElement = head.getElement();
        head = head.getNext();
        size--;
        if (size == 0) tail = null; // If list becomes empty, reset tail
        return removedElement;
    }}
```

# Data Structure Lab4 : Singly Linked List 2022-2023

## Homework

### 1. Equals Method Implementation

```
java
Copy code
@Override
public boolean equals(Object o) {
    if (o == null || getClass() != o.getClass()) return false;
    SinglyLinkedList<?> other = (SinglyLinkedList<?>) o;

    if (size != other.size) return false;

    Node<?> currentA = head;
    Node<?> currentB = other.head;

    while (currentA != null) {
        if (!currentA.getElement().equals(currentB.getElement())) return
false;
        currentA = currentA.getNext();
        currentB = currentB.getNext();
    }

    return true;
}
```

### 2. Find the Second-to-Last Node

**Algorithm:**

1. Start at the head.
2. Traverse the list until the next node's next reference is `null`.
3. Return the current node.

```
java
Copy code
public Node<E> secondToLast() {
    if (head == null || head.getNext() == null) return null; // List too
short
    Node<E> current = head;

    while (current.getNext().getNext() != null) {
        current = current.getNext();
    }

    return current;
}
```

### 3. Implement Size Without Instance Variable

```
java
Copy code
public int sizeWithoutInstanceVariable() {
    int count = 0;
    Node<E> current = head;

    while (current != null) {
        count++;
        current = current.getNext();
    }

    return count;
}
```

---

### 4. Rotate Method

**Explanation:** The `rotate()` method removes the first node and adds it to the end without creating new nodes.

```
java
Copy code
public void rotate() {
    if (head == null || head.getNext() == null) return; // Empty or single-
element list

    Node<E> oldHead = head;
    head = head.getNext();
    tail.setNext(oldHead);
    tail = oldHead;
    tail.setNext(null);
}
```

---

# Data Structure Lab4 : Singly Linked List 2022-2023

## 5. Concatenate Two Singly Linked Lists

### Algorithm:

1. If list LLL is empty, return MMM.
2. If MMM is empty, return LLL.
3. Set LLL’s tail to point to MMM’s head.
4. Update LLL’s tail to MMM’s tail.
5. L′L'L′ is now the concatenated list.

```
java
Copy code
public static <E> SinglyLinkedList<E> concatenate(SinglyLinkedList<E> L,
SinglyLinkedList<E> M) {
    if (L.isEmpty()) return M;
    if (M.isEmpty()) return L;

    L.tail.setNext(M.head);
    L.tail = M.tail;
    L.size += M.size;

    return L;
}
```

---

## 6. Reverse a Singly Linked List

### Algorithm:

1. Use three pointers: `prev`, `current`, and `next`.
2. Traverse the list, reversing the direction of the `next` pointer for each node.
3. Update `head` and `tail`.

```
java
Copy code
public void reverse() {
    if (head == null || head.getNext() == null) return; // Empty or single-node list

    Node<E> prev = null;
    Node<E> current = head;

    while (current != null) {
        Node<E> next = current.getNext(); // Store the next node
        current.setNext(prev);            // Reverse the current node's link
        prev = current;                   // Move prev to current
        current = next;                   // Move current to next
    }

    tail = head;
    head = prev;
}
```