# Data Structure Lab6 : Doubly Linked List 2022-2023

## Topics: DoublyLinkedList Implementation

### 1. Node Class

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

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

    public E getElement() {
        return element;
    }

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

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

    public Node<E> getPrev() {
        return prev;
    }

    public void setPrev(Node<E> prev) {
        this.prev = prev;
    }
}
```

---

# Data Structure Lab6 : Doubly Linked List 2022-2023

2. DoublyLinkedList Class

```
java
Copy code
public class DoublyLinkedList<E> {
    private Node<E> header; // Sentinel before the first element
    private Node<E> trailer; // Sentinel after the last element
    private int size = 0; // Number of elements in the list

    public DoublyLinkedList() {
        header = new Node<>(null, null, null);
        trailer = new Node<>(null, header, null);
        header.setNext(trailer);
    }

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

    public int size() {
        return size;
    }

    public E first() {
        if (isEmpty()) return null;
        return header.getNext().getElement();
    }

    public E last() {
        if (isEmpty()) return null;
        return trailer.getPrev().getElement();
    }

    public void addFirst(E element) {
        addBetween(element, header, header.getNext());
    }

    public void addLast(E element) {
        addBetween(element, trailer.getPrev(), trailer);
    }

    public E removeFirst() {
        if (isEmpty()) return null;
        return remove(header.getNext());
    }

    public E removeLast() {
        if (isEmpty()) return null;
        return remove(trailer.getPrev());
    }

    private void addBetween(E element, Node<E> predecessor, Node<E> successor) {
        Node<E> newNode = new Node<>(element, predecessor, successor);
        predecessor.setNext(newNode);
        successor.setPrev(newNode);
        size++;
    }

    private E remove(Node<E> node) {
        Node<E> predecessor = node.getPrev();
        Node<E> successor = node.getNext();
        predecessor.setNext(successor);
        successor.setPrev(predecessor);
        size--;
        return node.getElement();
    }
}
```

# Data Structure Lab6 : Doubly Linked List 2022-2023

## Homework

### 1. Find the Middle Node by Link Hopping

**Algorithm:**

1. Use two pointers: one moves two steps at a time (`fast`), and the other moves one step at a time (`slow`).
2. When the `fast` pointer reaches the end, the `slow` pointer will be at the middle.
3. If the list has an even number of nodes, return the node slightly left of center.

**Implementation:**

```
java
Copy code
public Node<E> findMiddle() {
    if (isEmpty()) return null;

    Node<E> slow = header.getNext();
    Node<E> fast = header.getNext();

    while (fast != trailer && fast.getNext() != trailer) {
        slow = slow.getNext();
        fast = fast.getNext().getNext();
    }

    return slow; // Middle node
}
```

---

### 2. Implement `size()` Without Maintaining a Size Variable

Traverse the list to count nodes:

```
java
Copy code
public int calculateSize() {
    int count = 0;
    Node<E> current = header.getNext();

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

    return count;
}
```

---

# Data Structure Lab6 : Doubly Linked List 2022-2023

### 3. Implement `equals()`

Two lists are equal if they have the same size and corresponding elements are equal.

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

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

    Node<E> currentA = header.getNext();
    Node<?> currentB = other.header.getNext();

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

    return true;
}
```

---

### 4. Concatenate Two DoublyLinkedLists

**Algorithm:**

1. Remove the `trailer` of the first list.
2. Link the `trailer` of the first list to the `header` of the second list.
3. Update the `trailer` of the concatenated list.

**Implementation:**

```
java
Copy code
public void concatenate(DoublyLinkedList<E> other) {
    if (other.isEmpty()) return;
    if (this.isEmpty()) {
        this.header = other.header;
        this.trailer = other.trailer;
    } else {
        this.trailer.getPrev().setNext(other.header.getNext());
        other.header.getNext().setPrev(this.trailer.getPrev());
        this.trailer = other.trailer;
    }
    this.size += other.size();
}
```

# Data Structure Lab6 : Doubly Linked List 2022-2023

5. Single Sentinel DoublyLinkedList

Reimplement the doubly linked list using a single sentinel node.

```
java
Copy code
public class SingleSentinelDoublyLinkedList<E> {
    private Node<E> sentinel; // Single sentinel node
    private int size = 0;

    public SingleSentinelDoublyLinkedList() {
        sentinel = new Node<>(null, null, null);
        sentinel.setNext(sentinel);
        sentinel.setPrev(sentinel);
    }

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

    public int size() {
        return size;
    }

    public E first() {
        if (isEmpty()) return null;
        return sentinel.getNext().getElement();
    }

    public E last() {
        if (isEmpty()) return null;
        return sentinel.getPrev().getElement();
    }

    public void addFirst(E element) {
        addBetween(element, sentinel, sentinel.getNext());
    }

    public void addLast(E element) {
        addBetween(element, sentinel.getPrev(), sentinel);
    }

    public E removeFirst() {
        if (isEmpty()) return null;
        return remove(sentinel.getNext());
    }

    public E removeLast() {
        if (isEmpty()) return null;
        return remove(sentinel.getPrev());
    }

    private void addBetween(E element, Node<E> predecessor, Node<E> successor) {
        Node<E> newNode = new Node<>(element, predecessor, successor);
        predecessor.setNext(newNode);
        successor.setPrev(newNode);
        size++;
    }

    private E remove(Node<E> node) {
        Node<E> predecessor = node.getPrev();
        Node<E> successor = node.getNext();
        predecessor.setNext(successor);
        successor.setPrev(predecessor);
        size--;
        return node.getElement();
    }
}
```

# Data Structure Lab6 : Doubly Linked List 2022-2023

6. Circular DoublyLinkedList with Rotate Methods

```
java
Copy code
public class CircularDoublyLinkedList<E> {
    private Node<E> tail; // Tail of the list
    private int size = 0;

    public CircularDoublyLinkedList() { }

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

    public int size() {
        return size;
    }

    public E first() {
        if (isEmpty()) return null;
        return tail.getNext().getElement();
    }

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

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

    public void addLast(E element) {
        addFirst(element);
        tail = tail.getNext();
    }

    public E removeFirst() {
        if (isEmpty()) return null;
        Node<E> head = tail.getNext();
        if (head == tail) {
            tail = null;
        } else {
            tail.setNext(head.getNext());
            head.getNext().setPrev(tail);
        }
        size--;
        return head.getElement();
    }

    public E removeLast() {
        if (isEmpty()) return null;
        tail = tail.getPrev();
        return removeFirst();
    }

    public void rotate() {
        if (tail != null) tail = tail.getNext();
    }

    public void rotateBackward() {
        if (tail != null) tail = tail.getPrev();
    }
}
```

## 7. Clone Method

```
java
Copy code
@Override
public DoublyLinkedList<E> clone() {
    DoublyLinkedList<E> cloned = new DoublyLinkedList<>();
    Node<E> current = header.getNext();

    while (current != trailer) {
        cloned.addLast(current.getElement());
        current = current.getNext();
    }

    return cloned;
}
```