Doubly LinkedList

```
public class DoublyLinkedList {
    private Node head;
    private Node tail;

    // Node inner class for doubly linked list
    private class Node {
        int data;
        Node prev;
        Node next;

        public Node(int data) {
            this.data = data;
            this.prev = null;
            this.next = null;
        }
    }

    // Constructor
    public DoublyLinkedList() {
        head = null;
        tail = null;
    }

    // Method to add an element at the front of the list
    public void addFirst(int data) {
        Node newNode = new Node(data);
        if (head == null) { // If the list is empty
```

```
            head = newNode;
            tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }

    // Method to add an element at the end of the list
    public void addLast(int data) {
        Node newNode = new Node(data);
        if (tail == null) { // If the list is empty
            head = newNode;
            tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

    // Method to remove an element from the front of the list
    public int removeFirst() {
        if (head == null) throw new RuntimeException("Cannot remove from an empty list");
        int data = head.data;
        if (head == tail) { // Only one element in the list
            head = null;
            tail = null;
```

```
    } else {
        head = head.next;
        head.prev = null;
    }
    return data;
}

// Method to remove an element from the end of the list
public int removeLast() {
    if (tail == null) throw new RuntimeException("Cannot remove from an empty list");
    int data = tail.data;
    if (head == tail) { // Only one element in the list
        head = null;
        tail = null;
    } else {
        tail = tail.prev;
        tail.next = null;
    }
    return data;
}

// Method to print elements from front to back
public void printForward() {
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " ");
        current = current.next;
    }
    System.out.println();
```

```
    }

    // Method to print elements from back to front
    public void printBackward() {
        Node current = tail;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.prev;
        }
        System.out.println();
    }
}

// Main class to run examples
public class Main {
    public static void main(String[] args) {
        DoublyLinkedList dll = new DoublyLinkedList();
        dll.addFirst(10);
        dll.addFirst(20);
        dll.addLast(5);
        dll.addLast(1);

        System.out.println("List from front to back:");
        dll.printForward();  // Outputs: 20 10 5 1

        System.out.println("List from back to front:");
        dll.printBackward();  // Outputs: 1 5 10 20

        dll.removeFirst();
```

```
        dll.removeLast();

        System.out.println("List after removing first and last:");
        dll.printForward();  // Outputs: 10 5
    }
}
```