[Create] Doubly Linked List

```
public class MainClass {
    public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList();
        list.addNode(0);
        list.addNode(1);
        list.addNode(2);
        list.addNode(3);
        list.addNode(4);
        list.addNode(5);

        list.printNodesFromHead();
        list.printNodesFromTail();

        list.removeNode(2);
        list.addLast(7);
        list.removeLast();
        list.printNodesFromHead();
    }
}

class Node {
    public Node nextNode;
    public Node previousNode;
    public int value;

    public Node(int value) {
        this.value = value;
        this.nextNode = null;
        this.previousNode = null;
    }
}

class DoublyLinkedList {
    Node head;
    Node tail;

    public void addNode(int value) {
        Node node = new Node(value);
        if (head == null) {
```

```
            head = node;
            tail = node;
        } else {
            node.previousNode = tail;
            tail.nextNode = node;
            tail = node;
        }
    }

    public boolean removeNode(int value) {
        Node temp = head;
        Node prev = null;
        if (temp == null) {
            return false;
        }
        // If the value is head.
        if (temp.value == value) {
            head = head.nextNode;
            return true;
        }
        // Search node.
        while (temp != null && temp.value != value) {
            prev = temp;
            temp = temp.nextNode;
        }
        // No item is available.
        if (temp == null) {
            return false;
        }
        prev.nextNode = temp.nextNode;
        if (temp.nextNode != null) {
            Node afterTempNode = temp.nextNode;
            afterTempNode.previousNode = prev;
        }
        return true;
    }

    public void printNodesFromHead() {
        Node temp = head;
        System.out.print("Head: [");

        // No item available.
        if (head == null) {
```

```
            return;
        }

        while (temp != null) {
            if (temp.nextNode != null) {
                System.out.print(temp.value + ", ");
            } else {
                System.out.print(temp.value);
            }
            temp = temp.nextNode;
        }
        System.out.print("]");
        System.out.println();
    }

    public void printNodesFromTail() {
        Node temp = tail;
        System.out.print("Tail: [");

        // No item available.
        if (tail == null) {
            return;
        }

        while (temp != null) {
            if (temp.previousNode != null) {
                System.out.print(temp.value + ", ");
            } else {
                System.out.print(temp.value);
            }
            temp = temp.previousNode;
        }
        System.out.print("]");
        System.out.println();
    }

    public void addFirst(int value) {
        Node node = new Node(value);
        if (head == null) {
            head = node;
            tail = node;
        } else {
            head.previousNode = node;
```

```
            node.nextNode = head;
            head = node;
        }
    }

    public void addLast(int value) {
        Node node = new Node(value);
        if (head == null) {
            head = node;
            tail = node;
        } else {
            node.previousNode = tail;
            tail.nextNode = node;
            tail = node;
        }
    }

    public void removeFirst() {
        if (head == null) {
            return;
        }
        head = head.nextNode;
    }

    public void removeLast() {
        if (head == null) {
            return;
        }
        Node temp = head;
        Node prev = null;
        while (temp.nextNode != null) {
            prev = temp;
            temp = temp.nextNode;
        }
        prev.nextNode = null;
    }
}
```