[Create] Singly Linked List

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

class Node {
    public Node nextNode;

    public int value;

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

class SinglyLinkedList {
    Node head;
    Node tail;

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

```
public boolean removeNode(int value) {
    Node temp = head;
    Node prev = null;

    // If head is value.
    if (temp != null && temp.value == value) {
        head = temp.nextNode;
        return true;
    }

    // Search value
    while (temp != null && temp.value != value) {
        prev = temp;
        temp = temp.nextNode;
    }

    // Search to the end and doesn't find any value
    if (temp == null) {
        return false;
    }

    // Replace prev with the next value of temp.
    prev.nextNode = temp.nextNode;
    return true;
}

public void printNodes() {
    Node iterator = head;
    System.out.print("[");
    while (iterator != null) {
        if (iterator.nextNode != null) {
            System.out.print(iterator.value + ", ");
        } else {
            System.out.print(iterator.value + "");
        }

        iterator = iterator.nextNode;
    }
    System.out.println("]");
}

public void addFirst(int value) {
```

```
        Node node = new Node(value);
        if (head == null) {
            head = node;
            tail = node;
        } else {
            node.nextNode = head;
            head = node;
        }
    }

    public void addLast(int value) {
        Node node = new Node(value);
        if (head == null) {
            head = node;
            tail = node;
        } else {
            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;
        tail = prev;
    }
}
```