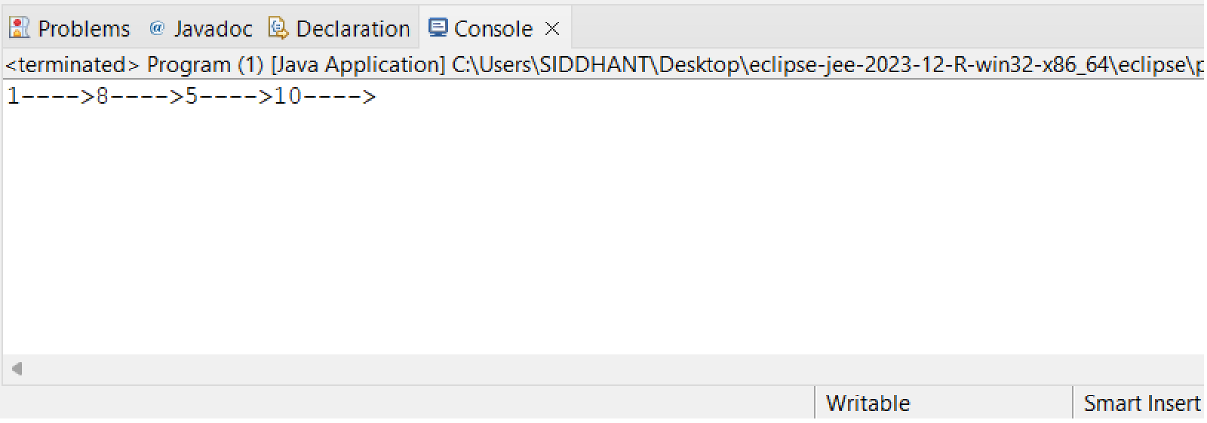
1.Implement singly linked list in java

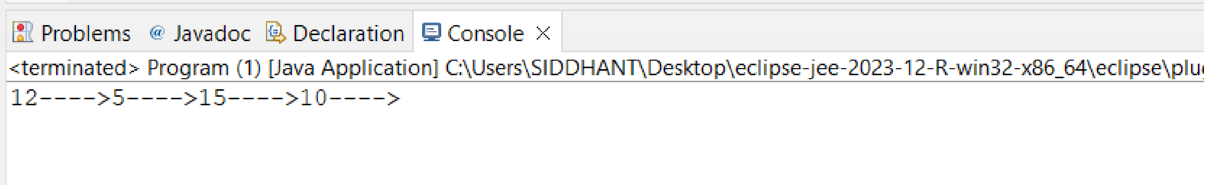
```
package com.example.main;

class SLL {

    Node head;

    static class Node{
        int data;
        Node next;

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

    void display(){
        Node n=head;

        while(n!=null){
            System.out.print(n.data+"---->");
            n=n.next;
        }
    }

    void insert(int newData){
        Node newNode=new Node(newData);
        newNode.next=head;
        head=newNode;
    }
}

class Program{
    public static void main(String[] args) {
        SLL sl=new SLL();
        sl.head=new SLL.Node(10);
        sl.insert(5);
        sl.insert(8);
        sl.insert(1);

        sl.display();

    }
}
```



```
<terminated> Program (1) [Java Application] C:\Users\SIDDHANT\Desktop\eclipse-jee-2023-12-R-win32-x86_64\eclipse\p
1---->8---->5---->10---->
```

2.Implement Doubly linked list in java

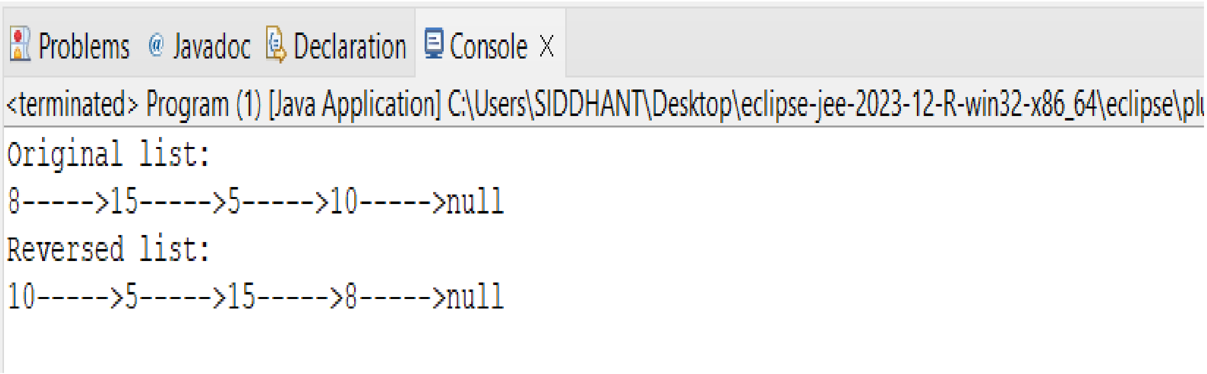
```
package com.example.main;

class DLL {
    Node head;
    static class Node {
        int data;
        Node next,prev;
        Node(int data){
            this.data=data;
            next=prev=null;
        }
    }

    void display(Node head){
        if(head==null) return;

        while(head!=null){
            System.out.print(head.data+"---->");
            head=head.next;
        }
    }

    void insert(int newData){
        Node newNode=new Node(newData);

        if(head==null) {
            head=newNode;
            return;
        }

        Node ptr=head;

        while(ptr.next!=null){
            ptr=ptr.next;
        }

        newNode.next=head;
        head=newNode;
    }
}

class Program{
    public static void main(String[] args) {
        DLL dl=new DLL();
        dl.head=new DLL.Node(10);
        dl.insert(15);
        dl.insert(5);
        dl.insert(12);

        dl.display(dl.head);
    }
}
```



```
<terminated> Program (1) [Java Application] C:\Users\SIDDHANT\Desktop\eclipse-jee-2023-12-R-win32-x86_64\eclipse\plu
12---->5---->15---->10---->
```

3.How to reverse a linked list in java

```
package com.example.main;

class SLL {

    Node head;
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            next = null;
        }
    }
    void display() {
        Node n = head;
        while (n != null) {
            System.out.print(n.data + "----->");
            n = n.next;
        }
        System.out.println("null");
    }

    void insert(int newData) {
        Node newNode = new Node(newData);
        newNode.next = head;
        head = newNode;
    }
    void reverse() {
        Node prev = null;
        Node curr = head;
        Node nextNode;
        while (curr != null) {
            nextNode = curr.next;
            curr.next = prev;
            prev = curr;
            curr = nextNode;
        }

        head = prev;
    }
}

class Program {
    public static void main(String[] args) {
        SLL sll = new SLL();
        sll.insert(10);
        sll.insert(5);
        sll.insert(15);
        sll.insert(8);
        System.out.println("Original list:");
        sll.display();

        sll.reverse();
        System.out.println("Reversed list:");
        sll.display();
    }
}
```



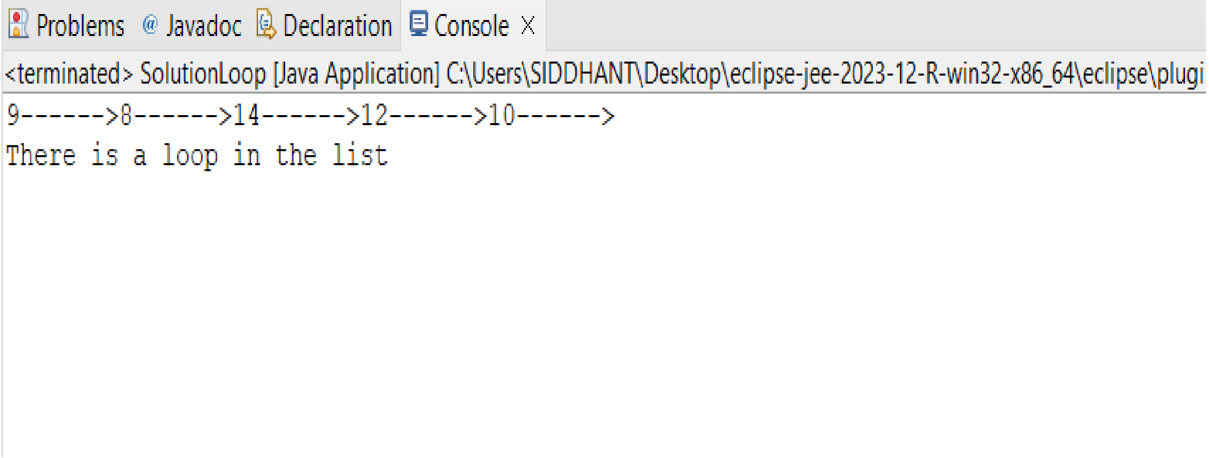
<terminated> Program (1) [Java Application] C:\Users\SIDDHANT\Desktop\eclipse-jee-2023-12-R-win32-x86_64\eclipse\plu

```
Original list:
8----->15----->5----->10----->null
Reversed list:
10----->5----->15----->8----->null
```

4.How to detect a loop in linked list in java

```
package com.example.main;

class Loop {

    Node head;

    static class Node {
        int data;
        Node next;

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

    void display() {
        if (head == null)
            return;
        else {
            Node n = head;
            while (n != null) {
                System.out.print(n.data + "------>");
                n = n.next;
            }
        }
    }

    void add(int newData) {
        Node newNode = new Node(newData);

        if (head == null)
            head = newNode;
        else {
            newNode.next = head;
            head = newNode;
        }
    }

    boolean detectLoop() {
        Node slow = head;
        Node fast = head;

        if (head == null)
            return false;

        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;

            if (slow == fast)
                return true;
        }

        return false;
    }

    void loop() {
        if (head == null)
```

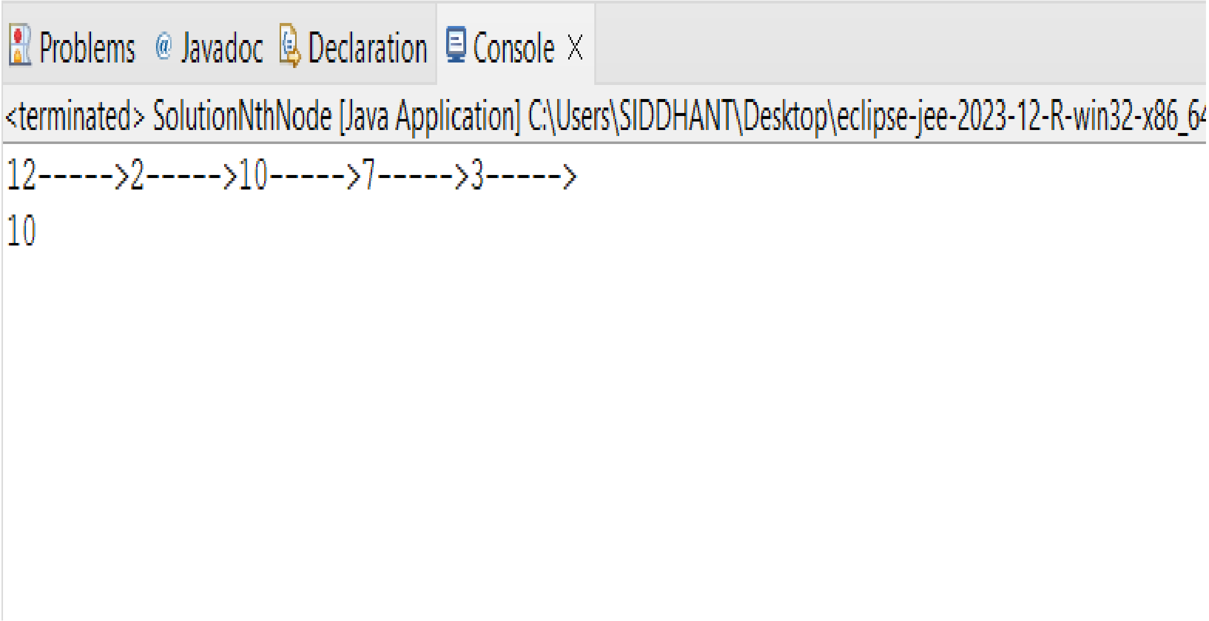
```
            return;

        Node n = head;

        while (n.next != null) {
            n = n.next;
        }

        n.next = head;
    }
}

public class SolutionLoop {
    public static void main(String[] args) {
        Loop l = new Loop();

        l.add(10);
        l.add(12);
        l.add(14);
        l.add(8);
        l.add(9);

        l.display();
        System.out.println();

        // Create a loop in the list
        l.loop();

        // Now check if there is a loop
        if (l.detectLoop())
            System.out.println("There is a loop in the list");
        else
            System.out.println("There is no loop in the list");
    }
}
```



```
<terminated> SolutionLoop [Java Application] C:\Users\SIDDHANT\Desktop\eclipse-jee-2023-12-R-win32-x86_64\eclipse\plugi
9------>8------>14------>12------>10------>
There is a loop in the list
```

5.How to find nth element from end of linked list

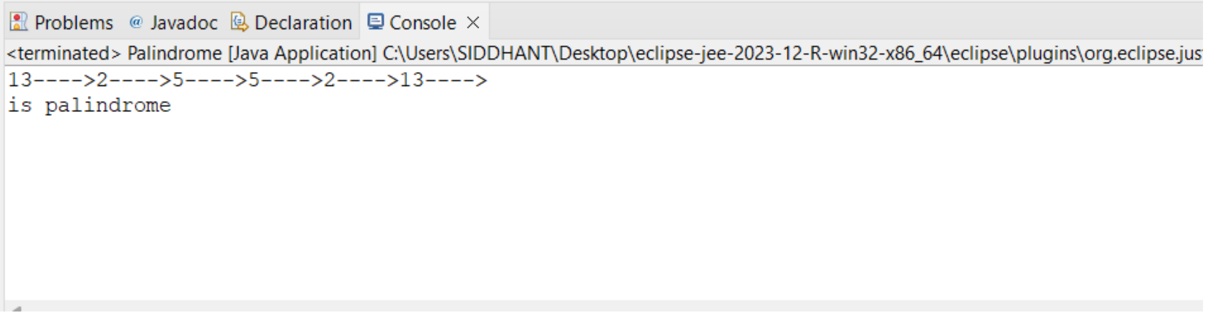
```
package com.example.main;

public class SolutionNthNode {
    Node head;

    static class Node {
        int data;
        Node next;

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

    void display() {
        if (head == null) return;

        Node n = head;

        while (n != null) {
            System.out.print(n.data + "----->");
            n = n.next;
        }
    }

    void add(int newData) {

        Node newNode = new Node(newData);

        if (head == null) {
            head = newNode;
            return;
        }

        Node n = head;

        while (n.next != null) {
            n = n.next;
        }

        n.next = newNode;
    }

    Node find(int n) {
        if (head == null || n <= 0) {
            return null;
        }

        Node ptr1 = head;
        Node ptr2 = head;

        for (int i = 0; i < n; i++) {
            if (ptr1 == null) return null;
            ptr1 = ptr1.next;
        }


        while (ptr1 != null) {
            ptr1 = ptr1.next;
            ptr2 = ptr2.next;
        }

        return ptr2;
    }

    public static void main(String[] args) {
        SolutionNthNode l = new SolutionNthNode();

        l.add(12);
        l.add(2);
        l.add(10);
        l.add(7);
        l.add(3);

        l.display();
        System.out.println();

        Node node = l.find(2);

        if (node != null) System.out.println(node.data);
        else System.out.println("no node");
    }
}
```



```
<terminated> SolutionNthNode [Java Application] C:\Users\SIDDHANT\Desktop\eclipse-jee-2023-12-R-win32-x86_6
12----->2----->10----->7----->3----->
10
```

6.How to check if linked list is palindrome in java

```
package com.example.main;

public class Palindrome {
    Node head;

    static class Node {
        int data;
        Node next;

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

    void display() {
        if (head == null)
            return;

        Node n = head;

        while (n != null) {
            System.out.print(n.data + "---->");
            n = n.next;
        }
    }

    void add(int newData) {

        Node newNode = new Node(newData);

        if (head == null) {
            head = newNode;
            return;
        }

        Node n = head;

        while (n.next != null) {
            n = n.next;
        }

        n.next = newNode;
    }

    Node reverse(Node node) {
        if (node == null)
            return null;

        Node prev = null;
        Node curr = node;
        Node next = null;
        while (curr != null) {
            next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }

    boolean isPalindrome() {
        if (head == null || head.next == null)
            return true;

        Node slow = head;
        Node fast = head;

        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }

        Node mid = reverse(slow);

        Node p1 = head;
        Node p2 = mid;

        while (p2 != null) {
            if (p1.data != p2.data)
                return false;

            p1 = p1.next;
            p2 = p2.next;
        }

        return true;
    }

    public static void main(String[] args) {
        Palindrome l = new Palindrome();

        l.add(13);
        l.add(2);
        l.add(5);
        l.add(5);
        l.add(2);
        l.add(13);

        l.display();
        System.out.println();

        if (l.isPalindrome()) System.out.println("is palindrome");
        else System.out.println("not palindrome");
    }
}
```



<terminated> Palindrome [Java Application] C:\Users\SIDDHANT\Desktop\eclipse-jee-2023-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jus

```
13---->2---->5---->5---->2---->13---->
is palindrome
```