Task 05:

Removing Duplicates from a Sorted Linked List

A sorted linked list has been constructed with repeated elements. Describe an algorithm to remove all duplicates from the linked list efficiently.

Algorithm-:

1. **Traversal with Two Pointers:**
   - We'll use two pointers to iterate through the linked list:
     - current: This pointer will traverse the entire list.
     - prev: This pointer will point to the previous unique node encountered so far.
2. **Duplicate Check and Skipping:**
   - As we iterate using current:
     - If the value of current is the same as the value of prev, it means we've encountered a duplicate.
     - In this case, we simply skip the duplicate node by updating current to point to the next node (current.next).

3. **Updating prev for Uniques:**
    - If the value of current is different from the value of prev, it means we've encountered a unique element.
        - We update the next pointer of the prev node to point to the current node (prev.next = current).
        - We also update prev to point to the current node (prev = current) to keep track of the previous unique node.
4. **Iterate Until End:**
    - We continue iterating through the list using current until we reach the end (current becomes None).
5. **Handling the Last Node (Optional):**
    - In some cases, the last node might be a duplicate that wasn't handled in the loop.
    - You can optionally add a check after the loop to see if prev.next is not None and set it to None if it is a duplicate.
    -

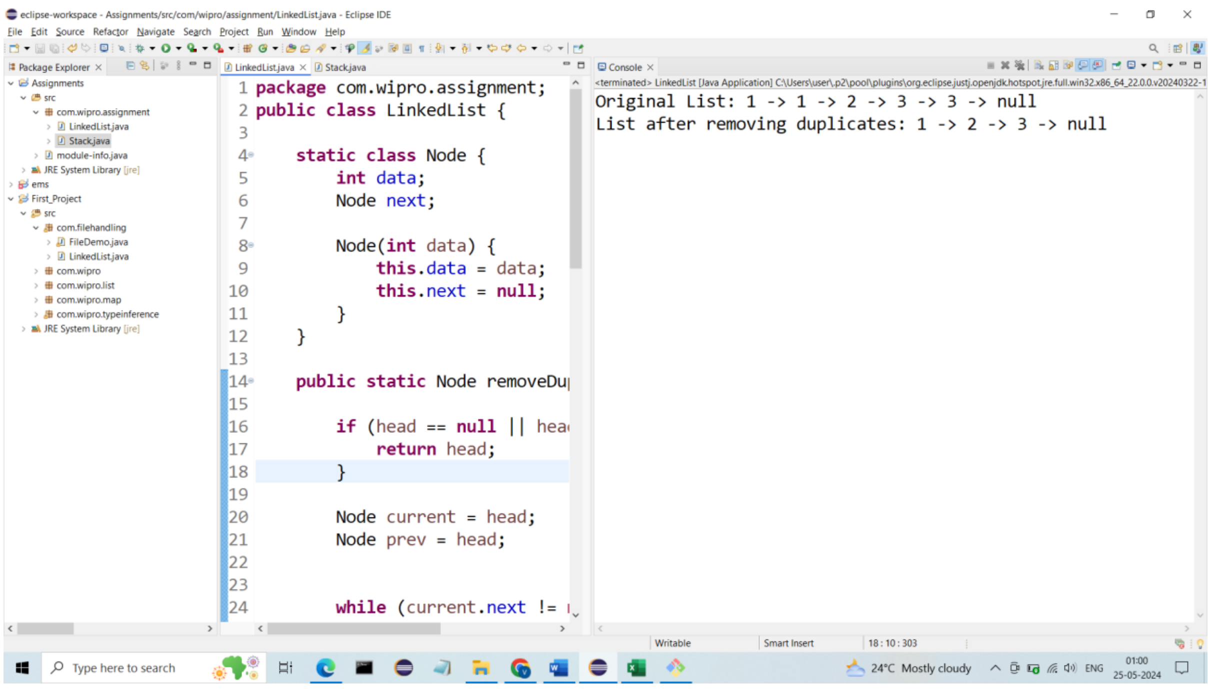
Implementation: -

```
package com.wipro.assignment;
```

```
public class LinkedList {

    static class Node {
        int data;
        Node next;

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

    public static Node removeDuplicates(Node
head) {

        if (head == null || head.next == null)
{
            return head;
        }

        Node current = head;
        Node prev = head;


        while (current.next != null) {
            current = current.next;


            if (current.data != prev.data) {
                prev.next = current;
                prev = current;
            }
        }
        prev.next = null;

        return head;
    }

    public static void main(String[] args) {
        // Sample linked list with duplicates
        Node head = new Node(1);
        head.next = new Node(1);
        head.next.next = new Node(2);
        head.next.next.next = new Node(3);
        head.next.next.next.next = new
Node(3);

        // Print the original list
        System.out.print("Original List: ");
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ->
");
            temp = temp.next;
        }
        System.out.println("null");

        // Remove duplicates
        head = removeDuplicates(head);

        // Print the list after removing
duplicates
        System.out.print("List after removing
duplicates: ");
```

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

Output: -



```
<terminated> LinkedList [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.0.v20240322-1
Original List: 1 -> 1 -> 2 -> 3 -> 3 -> null
List after removing duplicates: 1 -> 2 -> 3 -> null
```

```java
package com.wipro.assignment;
public class LinkedList {

    static class Node {
        int data;
        Node next;

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

    public static Node removeDu

        if (head == null || hea
            return head;
        }

        Node current = head;
        Node prev = head;


        while (current.next != 
```