Task 7: Merging Two Sorted Linked Lists

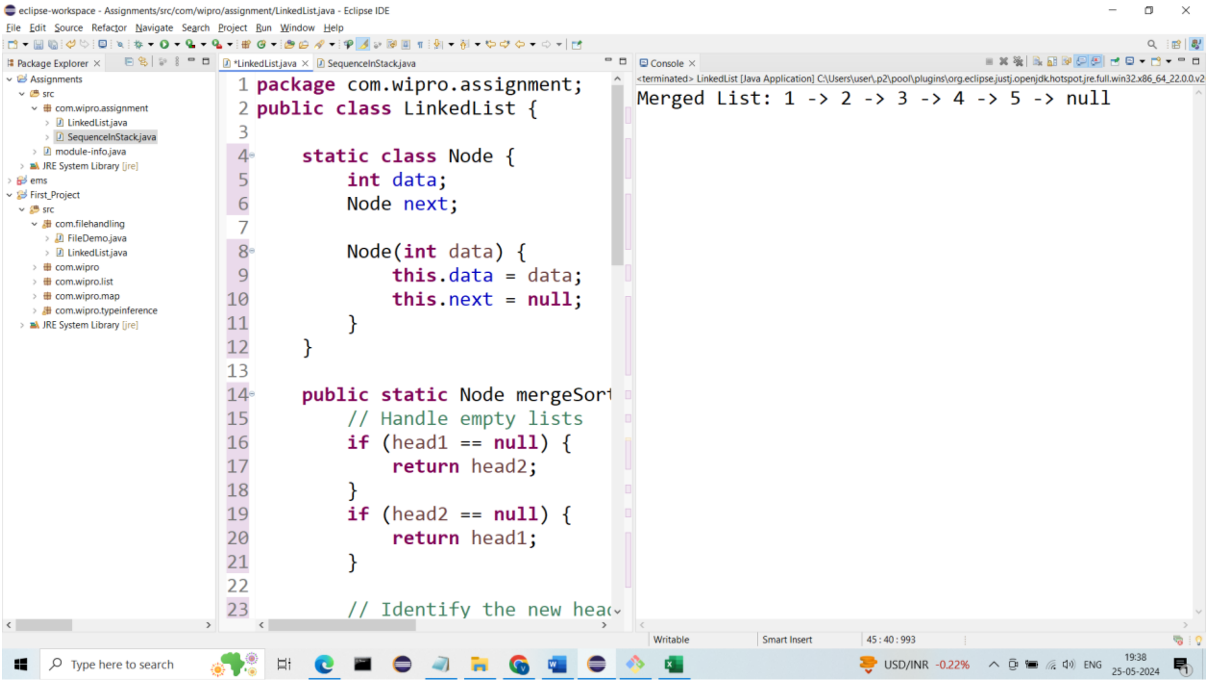
You are provided with the heads of two sorted linked lists. The lists are sorted in ascending order. Create a merged linked list in ascending order from the two input lists without using any extra space (i.e., do not create any new nodes).

Answer: -

```
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 mergeSortedLists(Node head1, Node head2) {
        // Handle empty lists
        if (head1 == null) {
            return head2;
        }
        if (head2 == null) {
            return head1;
        }
```

```java
        // Identify the new head of the merged list
        Node dummy = new Node(0); // Dummy node for
easier handling
        Node tail = dummy;

        while (head1 != null && head2 != null) {
            // Compare elements and attach the
smaller one to the tail
            if (head1.data < head2.data) {
                tail.next = head1;
                head1 = head1.next;
            } else {
                tail.next = head2;
                head2 = head2.next;
            }
            tail = tail.next;
        }

        // Append remaining elements from either
list
        tail.next = head1 != null ? head1 : head2;

        return dummy.next; // Return the actual
head (skip dummy)
    }

    public static void main(String[] args) {
        // Sample linked lists
        Node head1 = new Node(1);
        head1.next = new Node(3);
        head1.next.next = new Node(5);

        Node head2 = new Node(2);
        head2.next = new Node(4);

        // Merge the lists
        Node mergedHead = mergeSortedLists(head1, 
head2);

        // Print the merged list
        System.out.print("Merged List: ");
        while (mergedHead != null) {
            System.out.print(mergedHead.data + " -> 
");
            mergedHead = mergedHead.next;
        }
        System.out.println("null");
    }
}
```

Output: -


eclipse-workspace - Assignments/src/com/wipro/assignment/LinkedList.java - Eclipse IDE

```
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 mergeSort
        // Handle empty lists
        if (head1 == null) {
            return head2;
        }
        if (head2 == null) {
            return head1;
        }

        // Identify the new heac
```

<terminated> LinkedList [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.0.v2

```
Merged List: 1 -> 2 -> 3 -> 4 -> 5 -> null
```

Explanation:

1. **Node Class:** Defines the basic structure of a node in the linked list with data and a pointer to the next node.
2. **mergeSortedLists Function:**

- Takes the heads of the two sorted linked lists (head1 and head2) as input.
- Handles empty lists: If either head1 or head2 is null, it returns the other list as the merged list.
- Creates a dummy node (dummy) to simplify handling the head of the merged list.
- Initializes a tail pointer that points to the last node of the merged list so far (initially the dummy node).
- Iterates through both lists (head1 and head2) using a while loop until one of them reaches the end:
  - Compares the data of the nodes at the head of each list:
    - If the data in head1 is less than the data in head2, it attaches head1 to the next of tail and advances head1.
    - Otherwise, it attaches head2 to the next of tail and advances head2.
  - In both cases, it advances tail to point to the newly attached node.
- After the loop, any remaining elements in the non-exhausted list are appended to the end of the merged list using the tail.next assignment.
- Returns the next pointer of the dummy node (dummy.next), which is the actual head of the merged list (skipping the dummy node).

3. **Main Method:**
   - Creates two sample sorted linked lists head1 and head2.

- Calls the mergeSortedLists function to merge them.
- Prints the merged list.