Solution: Implement Doubly LinkedList

### **Steps to Implement:**

1. **Initialize** a *LinkedList* with the initial tasks.
2. **Add** a new task at the beginning of the day.
3. **Insert** a forgotten task somewhere in the middle of the list.
4. **Remove** the last task from the list.
5. **Traverse** the list both forward and backward, utilizing the *ListIterator* provided by the *LinkedList* to demonstrate the doubly linked nature.

**Solution**

import java.util.LinkedList;

import java.util.ListIterator;

public class TaskManager {

public static void main(String[] args) {

// Step 1: Initialize the LinkedList with the initial tasks

LinkedList<String> tasks = new LinkedList<>();

tasks.add("Check emails.");

tasks.add("Attend team meeting.");

tasks.add("Complete project report.");

// Step 2: Add a new task at the start

tasks.addFirst("Review today's schedule.");

System.out.println("Tasks after adding to the start: " + tasks);

// Step 3: Insert a forgotten task in the middle (at index 2)

tasks.add(2, "Prepare presentation slides.");

System.out.println("Tasks after adding in the middle: " + tasks);

// Step 4: Remove the last task from the list

tasks.removeLast();

System.out.println("Tasks after removing the last task: " + tasks);

// Step 5: Traverse and print the final task list in both directions

// Forward traversal

System.out.println("Final Task List (Forward):");

ListIterator<String> iterator = tasks.listIterator();

int taskNumber = 1;

while (iterator.hasNext()) {

System.out.println(taskNumber + ". " + iterator.next());

taskNumber++;

}

// Backward traversal

System.out.println("Final Task List (Backward):");

taskNumber = 1;

while (iterator.hasPrevious()) {

System.out.println(taskNumber + ". " + iterator.previous());

taskNumber++;

}

}

}