To implement the removeNthFromEnd method correctly in Java, you can use the **two-pointer approach**. The idea is to use a **dummy node** and two pointers (fast and slow) to find the node to be removed in a single pass.

Here is the **correct and complete implementation**:

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode removeNthFromEnd(ListNode head, int n) {

// Create a dummy node that points to the head

ListNode dummy = new ListNode(0, head);

ListNode fast = dummy;

ListNode slow = dummy;

// Move the fast pointer n+1 steps ahead so that the gap between fast and slow is n

for (int i = 0; i <= n; i++) {

fast = fast.next;

}

// Move both pointers until fast reaches the end

while (fast != null) {

fast = fast.next;

slow = slow.next;

}

// Remove the nth node from the end

slow.next = slow.next.next;

// Return the new head (might be different if head was removed)

return dummy.next;

}

}