package com.iimtiaz.day\_11;  
  
public class RemoveLinkedListElements {  
 public static void main(String[] args) {  
 // Create a linked list  
 ListNode head = new ListNode(1);  
 head.next = new ListNode(2);  
 head.next.next = new ListNode(2);  
 head.next.next.next = new ListNode(3);  
  
 // Print the original linked list  
 *printList*(head);  
  
 // Call the removeElements method  
 Solution solution = new Solution();  
 ListNode newHead = solution.removeElements(head, 2);  
  
 // Print the modified linked list  
 *printList*(newHead);  
 }  
  
 // Helper method to print a linked list  
 public static void printList(ListNode head) {  
 ListNode current = head;  
 while (current != null) {  
 System.*out*.print(current.val + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
}  
  
*/\*\*  
 \* Time Complexity: O(n)  
 \*  
 \* O(n): The while loop iterates through each node in the linked list once, regardless of the number of elements removed.  
 \* O(1): Operations within the loop (node comparisons, pointer assignments) are constant time.  
 \* Space Complexity: O(1)  
 \*  
 \* Uses a few fixed-size variables (node, temp, head), irrespective of the input linked list size.  
 \* Creates a new dummy node, but this doesn't affect overall space complexity as it's constant in size.  
 \*/*class Solution {  
 public ListNode removeElements(ListNode head, int val) {  
 if (head == null) return null;  
 ListNode node=new ListNode(0);  
 node.next=head;  
 ListNode temp=node;  
 while(temp.next!=null){  
 if(temp.next.val==val){  
 temp.next=temp.next.next;  
 }else{  
 temp=temp.next;  
 }  
 }  
 return node.next;  
 }  
}  
  
class ListNode {  
 int val;  
 ListNode next;  
  
 ListNode() {  
 }  
  
 ListNode(int val) {  
 this.val = val;  
 }  
  
 ListNode(int val, ListNode next) {  
 this.val = val;  
 this.next = next;  
 }  
}