**Task 2: Linked List Middle Element Search:**

**You are given a singly linked list. Write a function to find the middle element without using any extra space and only one traversal through the linked list.**

**Sol:**

To find the middle element of a singly linked list in Java with only one traversal and without using extra space, you can use the two-pointer technique. One pointer (let's call it slow) will move one step at a time, while the other pointer (let's call it fast) will move two steps at a time. When the fast pointer reaches the end of the list, the slow pointer will be at the middle element.

**Program:**

class ListNode {

int val;

ListNode next;

ListNode(int x) {

val = x;

next = null;

}

}

public class LinkedListMiddle {

public static ListNode findMiddle(ListNode head) {

if (head == null) {

return null;

}

ListNode slow = head;

ListNode fast = head;

// Move fast pointer two steps and slow pointer one step

while (fast != null && fast.next != null) {

slow = slow.next;

fast = fast.next.next;

}

// When fast pointer reaches the end, slow pointer will be at the middle

return slow;

}

public static void main(String[] args) {

// Create a sample linked list: 1 -> 2 -> 3 -> 4 -> 5

ListNode head = new ListNode(1);

head.next = new ListNode(2);

head.next.next = new ListNode(3);

head.next.next.next = new ListNode(4);

head.next.next.next.next = new ListNode(5);

ListNode middle = findMiddle(head);

if (middle != null) {

System.out.println("The middle element is: " + middle.val);

} else {

System.out.println("The list is empty.");

}

}

}