Task 2: Linked List Middle Element SearchYou 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.

1. **Two Pointers:**
   - Use two pointers, slow and fast.
   - Initially, both pointers are at the head of the list.
   - Move slow one step at a time.
   - Move fast two steps at a time.
   - When fast reaches the end of the list, slow will be at the middle.
2. **Implementation in Java**
   - Here is the Java code to find the middle element of a singly linked list:

```
class LinkedList {
        Node head; // head of the list
        static class Node {
        int data;
        Node next;
                // Constructor to create a new node
        Node(int d) {
        data = d;
        next = null;
                }
        }
   // Function to print the middle element of the linked list
   void printMiddle() {
      Node slow = head;
      Node fast = head;
      if (head != null) {
         while (fast != null && fast.next != null) {
            fast = fast.next.next;
            slow = slow.next;
         }
         System.out.println("The middle element is [" + slow.data + "] \n");
```

```
    }
}

// Function to add a new node at the end of the list
public void addToTheLast(Node node) {
    if (head == null) {
        head = node;
    } else {
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = node;
    }
}

// Function to print the linked list
void printList() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data + " ");
        temp = temp.next;
    }
    System.out.println();
}

 // Main method to test the linked list
public static void main(String[] args) {
    LinkedList llist = new LinkedList();
```

```
        // Adding nodes to the list
        llist.addToTheLast(new Node(1));
        llist.addToTheLast(new Node(2));
        llist.addToTheLast(new Node(3));
        llist.addToTheLast(new Node(4));
        llist.addToTheLast(new Node(5));

        // Printing the list
        System.out.println("The linked list is:");
        llist.printList();

        // Printing the middle element
        llist.printMiddle();
    }
}
```

Explanation

3. **Node Class:**
   - The Node class represents each node in the linked list and contains an integer data and a reference to the next node.
4. **LinkedList Class:**
   - Contains a reference to the head node of the list.
   - addToTheLast(Node node) method adds a new node at the end of the list.
   - printList() method prints all the elements of the list.
   - printMiddle() method finds and prints the middle element using the tortoise and hare approach.
5. **Main Method:**
   - Creates an instance of LinkedList.
   - Adds nodes to the linked list.
   - Prints the linked list.
   - Finds and prints the middle element.