```java
/* Given a singly linked list where elements are sorted in ascending order, convert it to a height balanced BST.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

Example:

Given the sorted linked list: [-10,-3,0,5,9],

One possible answer is: [0,-3,9,-10,null,5], which represents the following height balanced BST:

      0
     / \
   -3   9
   /   /
 -10  5 */


class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
        val = x;
    }
}

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}

class Solution {
    public TreeNode sortedListToBST(ListNode head) {

        if (head == null) {
            return null;
        }

        return sortedListToBSTHelper(head, null);
    }

    private TreeNode sortedListToBSTHelper(ListNode head, ListNode tail) {

        if (head == tail || head == null) {
            return null;
        }

        ListNode middleListNode = findMiddleListNode(head, tail);

        TreeNode root = new TreeNode(middleListNode.val);

        root.left = sortedListToBSTHelper(head, middleListNode);
        root.right = sortedListToBSTHelper(middleListNode.next, tail);

        return root;
    }

    private ListNode findMiddleListNode(ListNode head, ListNode tail) {

        ListNode slowPointer = head, fastPointer = head;

        while (fastPointer != tail && fastPointer.next != tail) {

            fastPointer = fastPointer.next.next;
            slowPointer = slowPointer.next;
```

```
        }

        return slowPointer;
    }


    public static void main(String[] args) {

        int[] input = {-10, -3, 0, 5, 9};
        ListNode head = new ListNode(-10);
        head.next = new ListNode(-3);
        head.next.next = new ListNode(0);
        head.next.next.next = new ListNode(5);
        head.next.next.next.next = new ListNode(9);

        TreeNode root = new Solution().sortedListToBST(head);

        System.out.print("");
    }
}
```