```java
/*Author: Bochen (mddboc@foxmail.com)
Last Modified: Tue Apr 10 22:28:44 CST 2018*/

/*You are given two non-empty linked lists representing two non-negative integers.
The digits are stored in reverse order and each of their nodes contain a single
digit. Add the two numbers and return it as a linked list.

        You may assume the two numbers do not contain any leading zero, except the
        number 0 itself.

        Example

        Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
        Output: 7 -> 0 -> 8
        Explanation: 342 + 465 = 807.*/


import java.lang.System;
import java.util.*;
import java.lang.Math;
import java.util.HashMap;


class ListNode
{
    int val;
    ListNode next;

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



public class Main
{
    public static void main(String[] args)
    {
        ListNode l1 = new ListNode(2);
        l1.next = new ListNode(4);
        l1.next.next = new ListNode(3);

        ListNode l2 = new ListNode(5);
        l2.next = new ListNode(6);
        l2.next.next = new ListNode(4);

        Solution solution = new Solution();

        ListNode receive = solution.addTwoNumbers(l1, l2);

        System.out.println("haha");

    }


}


class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

        if ( l1 == null || l2 == null )
        {
            return (l1 == null ? l2 : l1);
        }

        int additionBit = 0;
        int currentSum = 0;
        ListNode head = new ListNode(0);
        ListNode pointer = head;

```

```
71            while ( l1 != null && l2 != null )
72            {
73                currentSum = l1.val + l2.val + additionBit;
74
75                pointer.next = new ListNode(currentSum % 10);
76                additionBit = currentSum / 10;
77
78                pointer = pointer.next;
79                l1 = l1.next;
80                l2 = l2.next;
81            }
82
83            ListNode l = (l1 == null) ? l2 : l1;
84            while (l != null)
85            {
86                currentSum = l.val + additionBit;
87
88                pointer.next = new ListNode(currentSum % 10);
89                additionBit = currentSum / 10;
90
91                pointer = pointer.next;
92                l = l.next;
93            }
94
95            if (additionBit!=0)
96            {
97                pointer.next = new ListNode(additionBit);
98            }
99
100           return head.next;
101       }
102   }
103
104
105
```