LeetCode #2

作者：friedwm
原文链接：https://ld246.com/article/1536112995125
来源网站：链滴

问题：给定两个**非空**链表来表示两个非负整数。位数按照**逆序**方式存储，它们的每个节点只存储单个字。将两数相加返回一个新的链表。

你可以假设除了数字 0 之外，这两个数字都不会以零开头。

示例：

输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)

输出：7 -> 0 -> 8

原因：342 + 465 = 807

思路：逐位相加，记录进位。可以认为已经遍历完的链表能继续推进，但是值是0，一直推进直到两节点值和进位都变成0.

代码：

```
package xyz.quxiao.play.lab.leetcode;

/**
 * 输入两个链表表示的数（逆序表示），输出之和的链表 Input: (2 -> 4 -> 3) + (5 -> 6 -> 4) Outp
t: 7 -> 0 -> 8 * * @author 作者 :quxiao 创建时间：2017/11/6 13:49
 */public class Problem2 {

  public static ListNode buildNode(long i) {
    char[] chars = String.valueOf(i).toCharArray();
    ListNode head = null;
    for (int j = 0; j < chars.length; j++) {
      if (head == null) {
        head = new ListNode(Integer.parseInt(String.valueOf(chars[j])));
      } else {
        ListNode tmp = new ListNode(Integer.parseInt(String.valueOf(chars[j])));
        tmp.next = head;
        head = tmp;
      }
    }
    return head;
  }

  public static long parseNode(ListNode node) {
    int idx = 0;
    long result = 0;
    while (node != null) {
      result += (node.val * pow(idx++));
      node = node.next;
    }
    return result;
  }

  public static int pow(int i) {
    int ret = 1;
    if (i == 0) {
      return 1;
    }
```

```
    for (int j = 0; j < i; j++) {
      ret *= 10;
    }
    return ret;
  }

  public static void printNode(ListNode listNode) {
    StringBuilder sb = new StringBuilder();
    while (listNode != null) {
      if (sb.length() != 0) {
        sb.append(" -> ");
      }
      sb.append(listNode.getVal());
      listNode = listNode.next;
    }
    System.out.println(sb.toString());
  }

  public static void main(String[] args) {
    ListNode l1 = buildNode(100);
    ListNode l2 = buildNode(41231);
    printNode(l1);
    printNode(l2);
    ListNode result = addTwoNumbers(l1, l2);
    printNode(result);
    System.out.println(result);
  }

  //    思路：构造一条最低位为链首的单项链表，从最低位相加并进位
  public static ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    int carry = 0;
    ListNode ret = null;
    ListNode last = null;
    while (l1 != null || l2 != null || carry != 0) {
      int result = (l1 != null ? l1.val : 0) + (l2 != null ? l2.val : 0) + carry;
      carry = result / 10;
      result = result % 10;
      if (ret == null) {
        ret = new ListNode(result);
        last = ret;
      } else {
        last.next = new ListNode(result);
        last = last.next;
      }
      if (l1 != null) {
        l1 = l1.next;
      }
      if (l2 != null) {
        l2 = l2.next;
      }
    }
    return ret;
  }
```

```
  public static class ListNode {

    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }

    public int getVal() {
      return val;
    }

    public ListNode setVal(int val) {
      this.val = val;
      return this;
    }

    public ListNode getNext() {
      return next;
    }

    public ListNode setNext(ListNode next) {
      this.next = next;
      return this;
    }
  }
}
```