Homework Turnin

Email:	rgalanos@fcps.edu
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	04-02
Receipt ID:	d183936cfd36c7749b5e4f6225e5df0c

Execution failed with return code 1 (general error). (Expected for JUnit when any test fails.)

Warning: Your program failed to compile :

```
ListNode.java:2: error: duplicate class: ListNode
    public class ListNode
           ^
1 error
```

Please correct your file(s), go back, and try to submit again. If you do not correct this problem, you are likely to lose a large number of points on the assignment. Please contact your TA if you are not sure why your code is not compiling successfully.

Turnin Failed! (See above)

There were some problems with your turnin. Please look at the messages above, fix the problems, then Go Back and try your turnin again.

GradeIt has a copy of your submission, but we believe that you will want to fix the problems with your submission by resubmitting a fixed version of your code by the due date.

We have received the following file(s):

ListLabReverse.java (6378 bytes)

```
//name:          date:

/*****************************************
Demonstrates many ways to reverse a list made of ListNodes.
******************************************/
public class ListLabReverse
{
   public static void main(String[] args)
   {
      ListNode head = new ListNode("hello", null);
      head = new ListNode("foo", head);
      head = new ListNode("boo", head);
      head = new ListNode("nonsense", head);
      head = new ListNode("computer",
         new ListNode("science",
         new ListNode("java",
         new ListNode("coffee", head))));

      System.out.print("original: \t\t\t");
      print(head);

      System.out.print("recur and print: \t\t");
      recurAndPrint(head);

      System.out.println();
      System.out.print("original: \t\t\t");
      print(head);

      System.out.print("reverse by building a new list: ");
      head = reverseBuildNew(head);
      print(head);

      System.out.print("iterate with 3 pointers:\t");
      head = iterateThreePointers(head);
      print(head);

      System.out.print("recur with 2 pointers: \t\t");
      head = recurTwoPointers(null, head);
      print(head);

      System.out.print("recur with pointers and append: ");
      head = recurPointersAppend(head);
      print(head);

      System.out.print("Mind Bender reverse:\t\t");
      head = mindBender(head);
      print(head);
   }
   public static void print(ListNode head)
   {
      System.out.print("[");
      while(head != null)
      {
         System.out.print(head.getValue());
         head = head.getNext();
         if(head != null)
            System.out.print(", ");
      }
      System.out.println("]");
   }
   /**********************************************
   1. This approach doesn't actually reverse the list.  It only prints
   the list in reverse order.  recurAndPrint() prints the square
   brackets and calls helper().  helper() is recursive.
   ******************************************************/
   public static void recurAndPrint(ListNode h)
   {
      System.out.print("[");
      helper(h);
      System.out.print("]");
   }
   private static void helper(ListNode p)
   {
      if(p.getNext()==null)
         System.out.print(p.getValue());
      else
      {
         helper(p.getNext());
         System.out.print(", " + p.getValue());
      }
   }

   /**********************************************
   2. This iterative method (for or while) produces a copy of the
   reversed list.  For each node going forward, make a new node and
   link it to the list. The list will naturally be in reverse.
   ***********************************************************/
   public static ListNode reverseBuildNew(ListNode head)
   {
      ListNode head2 = new ListNode(head.getValue(), head.getNext());
      ListNode copy = new ListNode(head2.getValue(),null);
      while(head2.getNext()!=null)
      {
         head2 = head2.getNext();
         copy = new ListNode(head2.getValue(),copy);
      }
      return copy;
   }

   /********************************************
   3a. This iterative method (while) uses 3 pointers to reverse
   the list in place.   The two local pointers are called
   prev and next.
   *********************************************************/
   public static ListNode iterateThreePointers(ListNode head)
   {
      ListNode prev = null;
      ListNode next = null;

      while(head!=null)
      {
         next = head.getNext();
         head.setNext(prev);
         prev = head;
         if(next == null)
            break;
         head = next;
      }

      return head;
   }

   /***************************************************
   3b. This method uses two pointers as arguments to reverse
   the list in place. It is recursive.
   **********************************************************/
   public static ListNode recurTwoPointers(ListNode prev, ListNode head)
   {
      ListNode next = head.getNext();
      head.setNext(prev);

      if(next == null);
      else
         head = recurTwoPointers(head, next);
      return head;
   }
   /***********************************************
   3c. On each recursive level, find pointerToLast() and
   nextToLast(). Make a new last. On way back, append()
   one level to the other.
   *********************************************************/
   public static ListNode recurPointersAppend(ListNode head)
   {
      if(head.getNext()==null)
         return head;
      else
      {
         ListNode last = pointerToLast(head);
         nextToLast(head).setNext(null);
         append(last, recurPointersAppend(head));
         return last;
      }
   }
   private static ListNode pointerToLast(ListNode head)
   {
      ListNode head2 = head;
      while(head2.getNext()!=null)
         head2 = head2.getNext();
      return head2;
   }
   private static ListNode nextToLast(ListNode head)
   {
      ListNode head2 = head;
      ListNode head3 = head;
      while(head2.getNext()!=null)
         head2 = head2.getNext();
      while(head3.getNext()!=head2)
         head3 = head3.getNext();
      return head3;
   }
   private static ListNode append(ListNode h1, ListNode h2)
   {
      h1.setNext(h2);
      return h1;
   }


   /**********************************************
   3d. This difficult method reverses the list in place, using one
   local pointer. Start with pointerToLast(). The helper method
   is recursive.
   ******************************************************/
   public static ListNode mindBender(ListNode head)
   {
      ListNode temp = pointerToLast(head);
      mindBenderHelper(head);
      head.setNext(null);
      return temp;
   }
   public static void mindBenderHelper(ListNode head)
   {
      if(head.getNext()==null);
      else
      {
         mindBenderHelper(head.getNext());
         head.getNext().setNext(head);
      }
   }
}

//the College Board's standard ListNode class
class ListNode
{
   private Object value;
   private ListNode next;
   public ListNode(Object v, ListNode n)
   {
      value=v;
      next=n;
   }
   public Object getValue()
   {
      return value;
   }
   public ListNode getNext()
   {
      return next;
   }
   public void setValue(Object newv)
   {
      value=newv;
   }
   public void setNext(ListNode newn)
   {
      next=newn;
   }
}
```