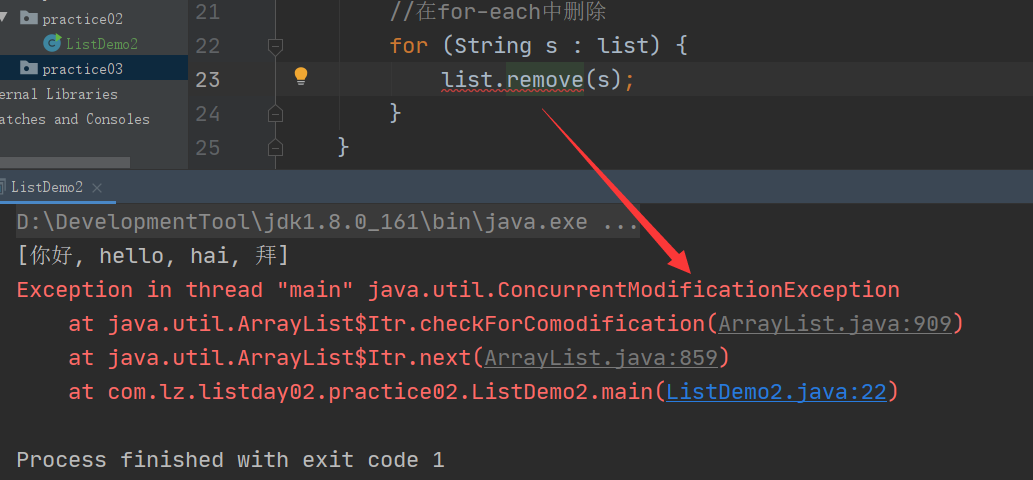
1. 使用for-each遍历数组 执行删除



报java.util.ConcurrentModificationException 并行修改异常



1. 使用普通的for循环

规约提示：

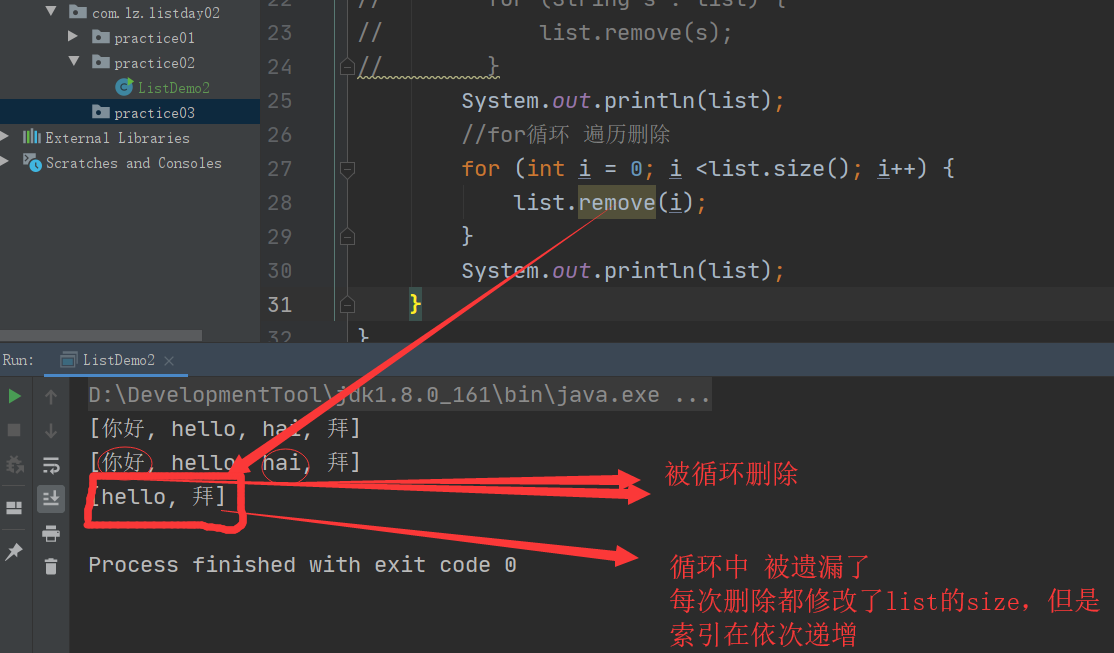
Suspicious 'List.remove()' in the loop

Inspection info: Reports when list.remove(index) is called inside the ascending counted loop. This is suspicious as list becomes shorter after that and the element next to removed will not be processed. Simple fix is to decrease the index variable after removal, but probably removing via iterator or using removeIf method (since Java 8) is a more robust alternative. If you don't expect that remove will be called more than once in a loop, consider adding a break command after it.

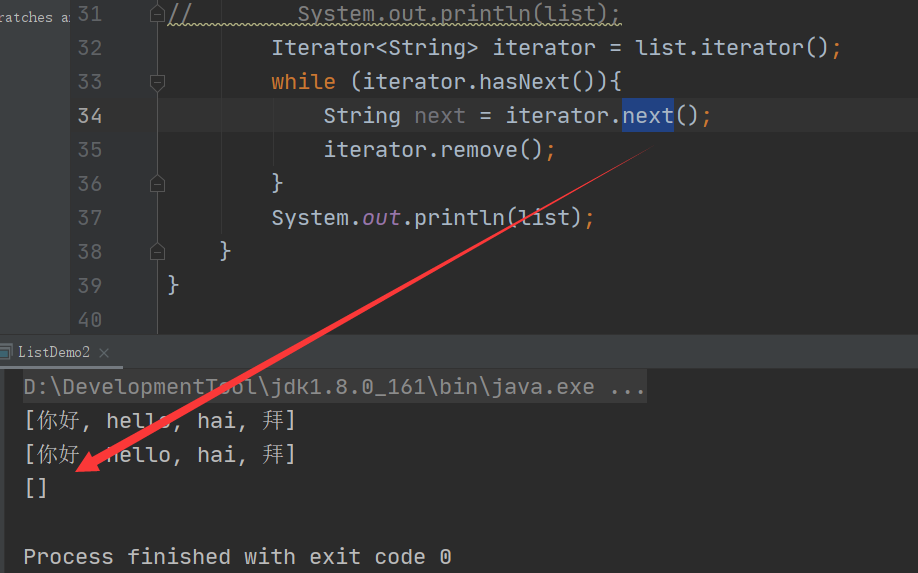
附：

循环中可疑的“ List.remove（）”  
检查信息：报告在递增计数循环内何时调用list.remove（index）。 这很可疑，因为列表在此之后变得更短，并且移除后的元素将不被处理。 一个简单的解决方法是在删除后减少index变量，但是可能通过迭代器或使用removeIf方法（由于Java 8）删除是一个更可靠的选择。 如果您不希望remove在循环中被多次调用，请考虑在其后添加break命令。

测试结果：



1. 使用迭代器完成删除是循环删除的最保险方式



先通过hasnext()方法判断当前位置后面有没有元素，如果有，通过next（）方法拿到下个元素，再进行remove操作。迭代器迭代只能使用迭代器的remove方法，集合遍历只能使用集合的remove方法。

迭代器里的remove方法，执行的是结合的remove方法，但是加入了针对源集合的修改的版本控制，任何一次修改都会被记录

final void checkForComodification() {

if (modCount != expectedModCount)

throw new ConcurrentModificationException();

}