ConcurrentModificationException是基于java集合中的 快速失败（fail-fast） 机制产生的，在使用迭代器遍历一个集合对象时，如果遍历过程中对集合对象的内容进行了增删改，就会抛出该异常。
快速失败机制使得java的集合类不能在多线程下并发修改，也不能在迭代过程中被修改。

如何避免ConcurrentModificationException异常
modCount只在集合的内部迭代器中使用，所以规避该异常只需在使用迭代器时考虑迭代器内部的expectedModCount与list的modCount值是否相等，游标cursor是否可能大于数组长度。

基于此思考归纳出以下几点可能抛该异常的情景：

- 迭代器遍历过程中，调用了集合的remove或add等改变了集合结构的方法；
- foreach循环遍历集合，实际上隐式调用了迭代器遍历，同样调用集合的remove或add等方法会抛异常；
- 多线程环境下，迭代器遍历+iterator.remove也可能导致异常，因此多线程环境下建议使用并发集合或做好线程间的同步。

Java集合的快速失败机制 “fail-fast”？
是java集合的一种错误检测机制，当多个线程对集合进行结构上的改变的操作 时，有可能会产生 failfast 机制。

例如：假设存在两个线程（线程1、线程2），线程1通过Iterator在遍历集合A中 的元素，在某个时候线程2修改了集合A的结构（是结构上面的修改，而不是简 单的修改集合元素的内容），那么这个时候程序就会抛出ConcurrentModificationException 异常，从而产生fail-fast机制。

原因：迭代器在遍历时直接访问集合中的内容，并且在遍历过程中使用一个 modCount 变量。集合在
被遍历期间如果内容发生变化，就会改变modCount 的值。每当迭代器使用hashNext()/next()遍历下一个元素之前，都会检测 modCount变量是否为expectedmodCount值，是的话就返回遍历；否则抛出异常，终止遍历。

解决办法： 1. 在遍历过程中，所有涉及到改变modCount值得地方全部加上synchronized。 2. 使用CopyOnWriteArrayList来替换ArrayList