fail-fast 机制是java集合(Collection)中的一种错误机制。当多个线程对同一个集合的内容进行操作时，就可能会产生fail-fast事件。
例：当某一个线程A通过iterator去遍历某集合的过程中，若该集合的内容被其他线程所改变了；那么线程A访问集合时，就会抛出ConcurrentModificationException异常，产生fail-fast事件。

原因：调用 next() 和 remove()时，都会执行checkForComodification() 。出现“**modCount 不等于 expectedModCount**”抛出该异常。

解决办法

在多线程环境下使用fail-fast机制的集合，建议使用“java.util.concurrent并发包下的类”去取代“java.util包下的类”。
CopyOnWriteArrayList
读写分离
写操作在一个复制的数组上进行，读操作还是在原始数组中进行，读写分离，互不影响。写操作需要加锁，防止并发写入时导致写入数据丢失。写操作结束之后需要把原始数组指向新的复制数组。适合读多写少的应用场景。
CopyOnWriteArrayList 缺陷：

- 内存占用：在写操作时需要复制一个新的数组，使得内存占用为原来的两倍左右；
- 数据不一致：读操作不能读取实时性的数据，因为部分写操作的数据还未同步到读数组中。

所以 CopyOnWriteArrayList 不适合内存敏感以及对实时性要求很高的场景。

ArrayList和CopyOnWriteArrayList：

(01) 和ArrayList继承于AbstractList不同，CopyOnWriteArrayList没有继承于AbstractList，它仅仅只是实现了List接口。

(02) ArrayList的iterator()函数返回的Iterator是在AbstractList中实现的；而CopyOnWriteArrayList是自己实现Iterator。

(03) ArrayList的Iterator实现类中调用next()时，会“调用checkForComodification()比较‘expectedModCount’和‘modCount’的大小”；但是，CopyOnWriteArrayList的Iterator实现类中，没有所谓的checkForComodification()，更不会抛出ConcurrentModificationExcep tion异常！