1. 快速失败

fail-fast 机制是java容器（Collection和Map都存在fail-fast机制）中的一种错误机制。在遍历一个容器对象时，当容器结构被修改，很有可能会抛出ConcurrentModificationException，产生fail-fast。

在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修改（增加、删除、修改），则会抛出Concurrent Modification Exception。

因此，面对并发的修改，迭代器快速而干净利落地失败，而不是在不确定的情况下冒险。

但是迭代器的remove并不会抛出该异常错误。

原理：

list类中保存着一个变量modCount,用于表示修改次数，即每次集合结构发生改变的时候，执行modCount++；在迭代器初始化过程中会执行`int expectedModCount = modCount;`来记录迭会通过checkForComodification()方法判断modCount和expectedModCount 是否相等，如果不相等就表示已经有线程修改了集合结构。

使用迭代器的remove()方法修改集合结构不会触发ConcurrentModificationException，现在可以在源码中看出来是为什么。在remove()方法的最后会执行`expectedModCount = modCount;`，这样itr.remove操作后modCount和expectedModCount依然相等，就不会触发ConcurrentModificationException了。

解决办法：

使用*java.util.concurrent*包下的类去取代java.util包下的类。所以，本例中只需要将Vector替换成java.util.concurrent包下对应的类即可。

2. 安全失败

采用安全失败机制的集合容器，在遍历时不是直接在集合内容上访问的，而是先复制原有集合内容，在拷贝的集合上进行遍历。

原理：由于迭代时是对原集合的拷贝进行遍历，所以在遍历过程中对原集合所作的修改并不能被迭代器检测到，所以不会触发Concurrent Modification Exception。

缺点：基于拷贝内容的优点是避免了Concurrent Modification Exception，但同样地，迭代器并不能访问到修改后的内容，即：迭代器遍历的是开始遍历那一刻拿到的集合拷贝，在遍历期间原集合发生的修改迭代器是不知道的。

场景：java.util.concurrent包下的容器都是安全失败，可以在多线程下并发使用，并发修改