**JAVA上机报告**

**代码：**

ThreadMain.Java:

**package** com.captain.demo.threadNoSafe;

**import** java.util.ArrayList;

**import** java.util.List;

/\*\*

\* Created by captain on 2017/7/27.

\*/

**public** **class** ThreadMain {

**public** **static** **void** main(String[] args) {

//第一个的演示示例

List<String> list = **new** ArrayList<>(100);

**for**(**int** cnt=0;cnt<100;cnt++){

list.add("abc");

}

ThreadSafeDemo1 demo1 = **new** ThreadSafeDemo1(list);

ThreadSafeDemo1 demo2 = **new** ThreadSafeDemo1(list);

Thread thread1 = **new** Thread(demo1);

Thread thread2 = **new** Thread(demo2);

thread1.start();

thread2.start();

}

ThreadSafeDemo1.java:

**package** com.captain.demo.threadNoSafe;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Random;

**import** java.util.concurrent.TimeUnit;

/\*\*

\* 线程安全演示类

\* Created by captain on 2017/7/27.

\*/

**public** **class** ThreadSafeDemo1 **implements** Runnable{

**private** List<String> list;

**public** ThreadSafeDemo1(List<String> list){

**this**.list = list;

}

@Override

**public** **void** run() {

Random random = **new** Random();

System.***out***.println("线程："+ random.nextInt());

**synchronized**(**this**){

Iterator<String> iterator = list.iterator();

**int** cnt = 0;

**while**(iterator.hasNext()){

iterator.next();

iterator.remove();

cnt++;

**try** {

TimeUnit.***MILLISECONDS***.sleep(10);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

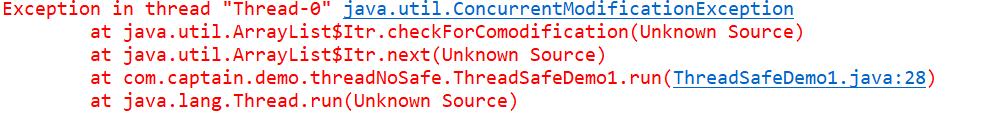
System.***out***.println("移除元素的数量："+cnt);

}

}

}

运行抛出异常:

****

原因：虽然Vector的方法采用了synchronized进行了同步，但是实际上通过Iterator访问的情况下，每个线程里面返回的是不同的iterator，也即是说expectedModCount是每个线程私有。假若此时有2个线程，线程1在进行遍历，线程2在进行修改，那么很有可能导致线程2修改后导致Vector中的modCount自增了，线程2的expectedModCount也自增了，但是线程1的expectedModCount没有自增，此时线程1遍历时就会出现expectedModCount不等于modCount的情况了。

　　因此一般有2种解决办法：

　　1）在使用iterator迭代的时候使用synchronized或者Lock进行同步；

　　2）使用并发容器CopyOnWriteArrayList代替ArrayList和Vector。