Java线程安全上机

班级：软工11601

姓名：王嘉欣

序号：31号

## 程序代码

### ThreadMain.java

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

**import** java.util.ArrayList;

**import** java.util.List;

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

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

//第一个的演示示例

List<String> list = **new** ArrayList<>(1000);

**for**(**int** cnt=0;cnt<1000;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();

//第二个的演示示例

/\*

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

ThreadSafeDemo2 demo1 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo2 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo3 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo4 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo5 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo6 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo7 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo8 = new ThreadSafeDemo2(list);

ThreadSafeDemo2 demo9 = new ThreadSafeDemo2(list);

ThreadSafeDemo3 demo10 = new ThreadSafeDemo3(list);

new Thread(demo1).start();

new Thread(demo2).start();

new Thread(demo3).start();

new Thread(demo4).start();

new Thread(demo5).start();

new Thread(demo6).start();

new Thread(demo7).start();

new Thread(demo8).start();

new Thread(demo9).start();

new Thread(demo10).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

\*/

**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**)//线程同步

{

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);

}

}

}

### ThreadSafeDemo2.java

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

**import** java.util.List;

/\*\*

\* 线程安全演示类

\*/

**public** **class** ThreadSafeDemo2 **implements** Runnable{

**private** List<String> list;

**public** ThreadSafeDemo2(List<String> list){

**this**.list = list;

}

@Override

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

**while**(**true**){

list.add("版权归作者所有，任何形式转载请联系作者。");

System.***out***.println("添加一个后尺寸："+list.size());

}

}

}

### ThreadSafeDemo3.java

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

**import** java.util.List;

/\*\*

\* 线程安全演示类

\*/

**public** **class** ThreadSafeDemo3 **implements** Runnable{

**private** List<String> list;

**public** ThreadSafeDemo3(List<String> list){

**this**.list = list;

}

@Override

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

**while**(**true**){

**if**(list.size()>0){

list.remove(0);

System.***out***.println("删除一个后尺寸："+list.size());

}

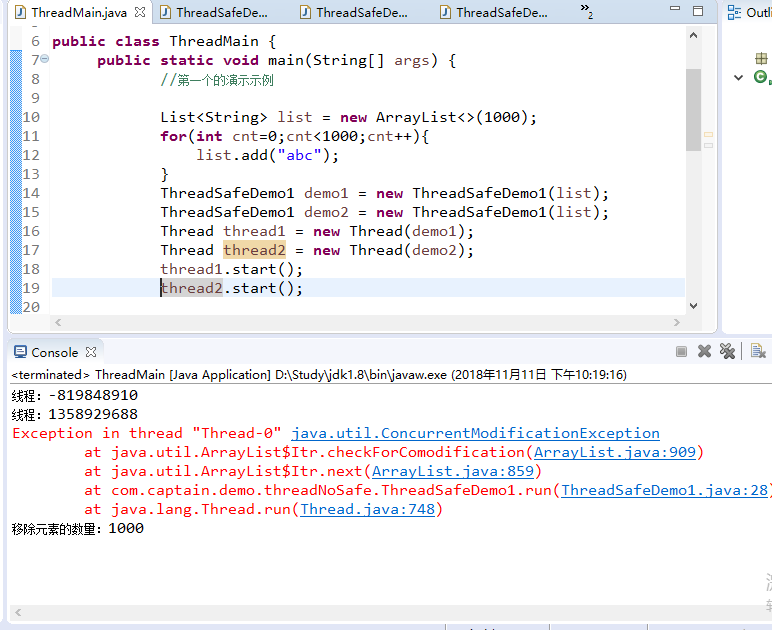
}

}

}

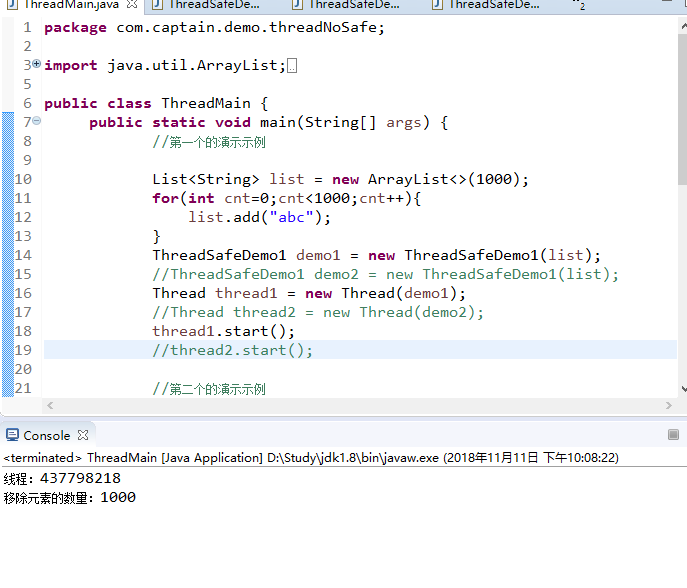
## 对于ThreadSafeDemo1

运行结果截图



只new一个demo时，程序不抛出异常

截图为



为了便于查看，将输出语句变为

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

…

…

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

相应的，main函数中的语句变为

Thread thread1 = **new** Thread(demo1,"A");

Thread thread2 = **new** Thread(demo2,"B");

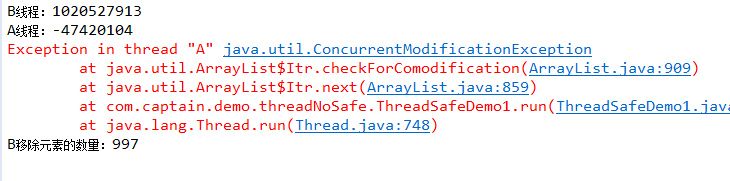
此时再看运行结果

截图为

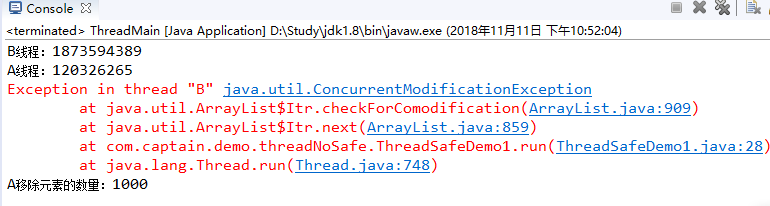
图一：



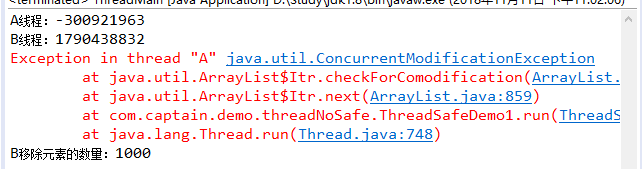
图二：



图三：



图四：



可以看到每次执行的线程顺序都不同

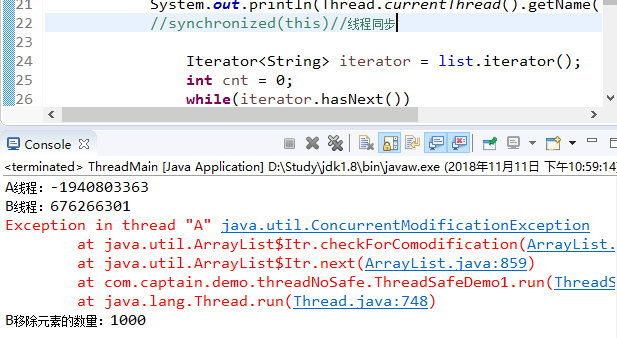
针对synchronized(this)同步代码块

当两个并发线程访问同一个对象object中的这个synchronized(this)同步代码块时，一个时间内只能有一个线程得到执行。另一个线程必须等待当前线程执行完这个代码块以后才能执行该代码块。

然而，当一个线程访问object的一个synchronized(this)同步代码块时，另一个线程仍然可以访问该object中的非synchronized(this)同步代码块。

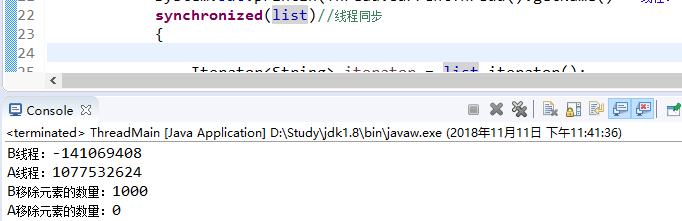
将synchronized(this)同步代码块注释

运行结果为



可以 看到同步代码块没有起作用

将synchronized(this)同步代码块改为synchronized(list)，即把iterator遍历过程加同步锁，锁住整个List



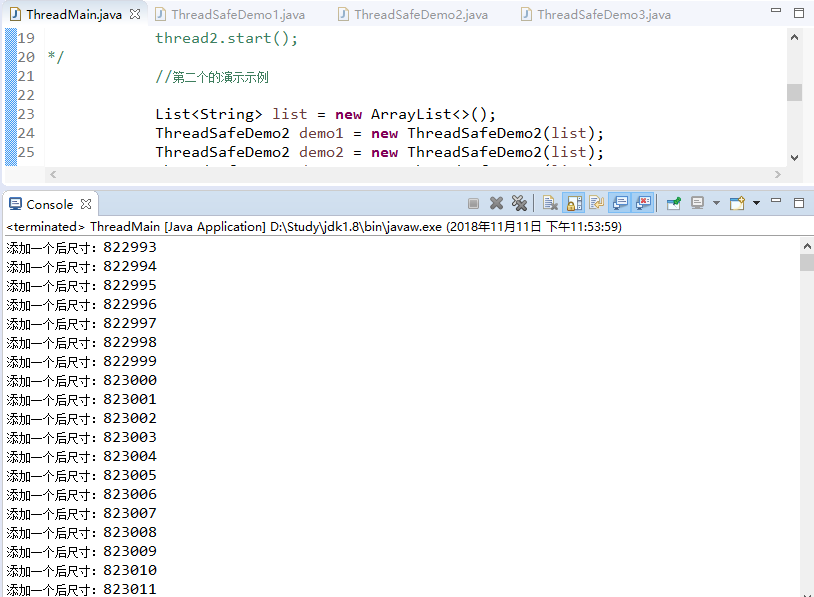
此时不抛出异常。

这种方案本质上是将多线程通过加锁来转变成单线程操作，确保同一时间内只有一个线程去使用itetator遍历list，其他线程等待。这种方案的效率显然只有单线程的效率

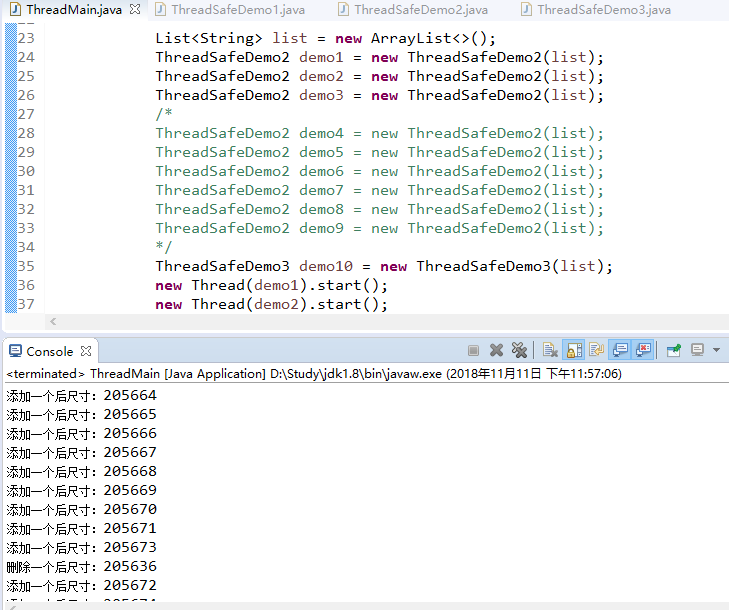
## 对于ThreadSafeDemo2

程序运行结果为

（强制停止后）



注释一部分对象，截图



可以看到程序运行结果与预期的三个添加一个删除不符。

且程序一直不停的进行下去。