# 第十周周日java上机报告

线程安全

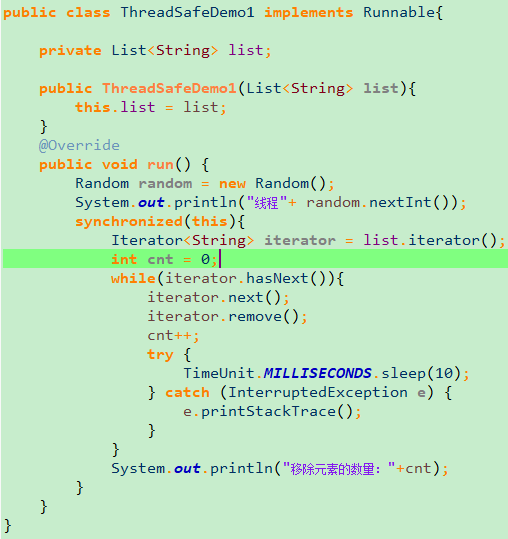
所谓线程不安全，就是多个线程对一个进程的同一个资源进行操作，因为线程自己是没有资源的，这样的话会发生数据对操作的不一致。针对这个特性，解决办法就是的思路就是1、把共享的数据锁起来2、给每个线程分配自己的数据，不用共享的了。

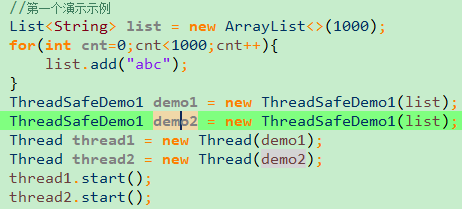
所以根据以上思路，有几个解决办法，我暂时只了解到对于思路一的办法：1、用锁机制synchronized,lock方式，为共享资源加一把锁2.用java类库所提供的线程安全的结合类。

下面看看两个的例子，分别用两种不同的办法来解决。

示例一：

代码：

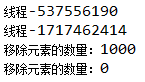




结果：

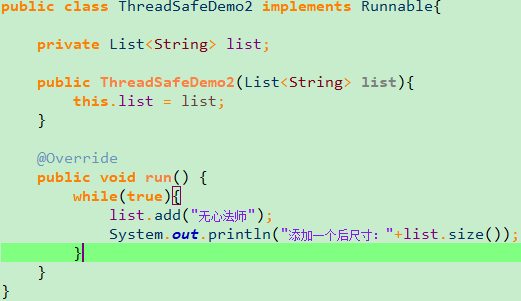
结果抛出异常了，很显然，抛出异常的原因就是两个线程对同一个list进行操作，

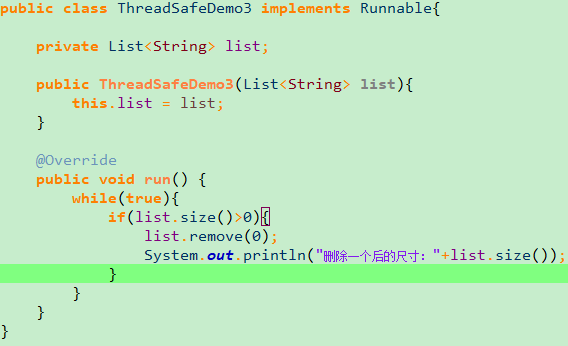
再看改进方法：

将synchronized(this)改成synchronized(this,list)。这样就把共享资源锁住了，再看运行结果：，结果没有在抛出异常了，结果显示，第一个进程把list的所有数据删除完，在删除的同时第二个进程不能对list进行操作，所以，第二个移除元素数量为0。

实例二：

代码：





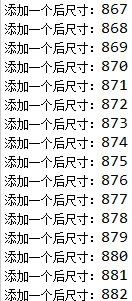


上述代码所演示的就是用十个线程对同一个list进行操作，九个往里面加元素，一个从里面删元素，结果就是list无限增加，如下图所示：

添加进程和删除进程时并发进行的，知识添加进程多一些。

再看改进方法：

将代码里面所有的Arraylist变成BlockingQueue。如：把List<String> list 都换成BlockingQueue<String> queue。结果如下图所示：

可见，删除线程和添加线程都时独立进行的。