java上机1

源代码：

/\*\*

\* 线程安全演示类

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

\*/

**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**.list){

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("error");

}

}

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

}

}

}

/\*\*

\* 线程安全演示类

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

\*/

**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。作者：KY主创们（来自豆瓣来源：https://www.douban.com/note/630084526/不过，专门研究“人猫交流（cat-human communication)”的学者认为这种印象也不完全是精确的——是的，这是一个真实存在的研究学科。研究宠物行为的学科曾经一度被取消，因为人们认为这种研究的本质是一种拟人论（anthropomorphism），但现在研究这个学科的学者已经不再这样认为了（即宠物行为研究有其自身本质的意义）。");

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

}

}

}

/\*\*

\* 线程安全演示类

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

\*/

**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());

}

}

}

}

**public** **class** ThreadTestt {

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

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();

}

}

当注释掉ThreadSafeDemo1类中run方法中的synchronized关键字的时候，运行此代码时，出现异常，运行结果如下：

线程：1106785531

线程：350489956

Exception in thread "Thread-0" java.util.ConcurrentModificationException

at java.util.ArrayList$Itr.checkForComodification(ArrayList.java:901)

at java.util.ArrayList$Itr.remove(ArrayList.java:865)

at java\_test.ThreadSafeDemo1.run(ThreadTestt.java:75)

at java.lang.Thread.run(Thread.java:748)

移除元素的数量：997

说明出现了两个线程同时对ArrayList操作的情况，而在一个线程中改变了ArrayList，另一个线程在执行remove的时候，发现ArrayList被更改过了，于是抛出了异常。这种时候的数据是不安全的，因为存在多个线程同时对一个数据进行操作。

当加上synchronized关键字的时候，运行结果如下：

线程：1757181329

线程：1653449687

移除元素的数量：1000

移除元素的数量：0

说明当用synchronized关键字将list锁住的时候，当一个线程拿到这个锁的时候，其他线程就不能再对list进行操作，所以只有一个线程对list进行操作，当这个线程操作完的时候，其他线程就可以操作了，所以第一个拿到锁的线程执行完后，移除元素的数量应该是1000，而此时list里面无元素了，所以第二个线程移除的元素数量为0。这个时候就不存在多个线程同时对一个数据进行操作，这个时候线程就是安全的。

将main函数里面的代码用下面代码替换：

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();

}

运行结果如下所示：

添加一个后尺寸：375488

添加一个后尺寸：375489

添加一个后尺寸：375490

添加一个后尺寸：375491

添加一个后尺寸：375492

添加一个后尺寸：375493

添加一个后尺寸：375494

添加一个后尺寸：375495

添加一个后尺寸：375496

添加一个后尺寸：375497

添加一个后尺寸：375498

添加一个后尺寸：375499

添加一个后尺寸：375500

添加一个后尺寸：375501

添加一个后尺寸：375502

删除一个后尺寸：373750

添加一个后尺寸：373731

添加一个后尺寸：375504

添加一个后尺寸：375505

添加一个后尺寸：375506

结果太多，只复制了这一段。

可以发现有多个线程同时对list进行操作，而打印出来的信息不一致，这个时候list是不准确的，将ThreadSafeDemo2中的run方法做了一个改进：

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

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

**synchronized** (**this**.list) {

list.add(

"版权归作者所有，任何形式转载请联系作者。作者：KY主创们（来自豆瓣来源：https://www.douban.com/note/630084526/不过，专门研究“人猫交流（cat-human communication)”的学者认为这种印象也不完全是精确的——是的，这是一个真实存在的研究学科。研究宠物行为的学科曾经一度被取消，因为人们认为这种研究的本质是一种拟人论（anthropomorphism），但现在研究这个学科的学者已经不再这样认为了（即宠物行为研究有其自身本质的意义）。");

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

}

}

}

将ThreadSafeDemo3中的run方法也做了一些改进：

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

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

**synchronized** (**this**.list) {

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

list.remove(0);

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

}

}

}

}

然后将main中的代码改为：

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

ThreadSafeDemo2 demo1 = **new** ThreadSafeDemo2(list);

ThreadSafeDemo2 demo9 = **new** ThreadSafeDemo2(list);

ThreadSafeDemo3 demo10 = **new** ThreadSafeDemo3(list);

**new** Thread(demo9).start();

**new** Thread(demo10).start();

运行结果如下：

添加一个后尺寸：40634

添加一个后尺寸：40635

添加一个后尺寸：40636

添加一个后尺寸：40637

添加一个后尺寸：40638

添加一个后尺寸：40639

添加一个后尺寸：40640

添加一个后尺寸：40641

添加一个后尺寸：40642

添加一个后尺寸：40643

添加一个后尺寸：40644

添加一个后尺寸：40645

添加一个后尺寸：40646

添加一个后尺寸：40647

添加一个后尺寸：40648

添加一个后尺寸：40649

添加一个后尺寸：40650

添加一个后尺寸：40651

删除一个后尺寸：40650

删除一个后尺寸：40649

删除一个后尺寸：40648

删除一个后尺寸：40647

删除一个后尺寸：40646

删除一个后尺寸：40645

删除一个后尺寸：40644

可以看到数据是有序的，也就是说不管是list添加数据还是移除数据，都要等拿到锁的线程处理完之后才能处理，所以每个线程只有在每次循环结束后到下一次循环开始时这一段时间内才可能拿到锁，这也就保证了一个时刻只会有一个线程对数据进行操作，保证了数据的正确性。