Java线程上机

测试代码：

package day07;

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("移除元素数量："+cnt);

//}

}

}

**package** day07;

**import** java.util.List;

/\*\*

\* 线程安全演示类

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

\*/

**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");

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

}

}

}

**package** day07;

**import** java.util.List;

/\*\*

\* 线程安全演示类

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

\*/

**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());

}

}

}

}

package day07;

import java.util.ArrayList;

import java.util.List;

/\*\*

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

\*/

public class ThreadMain {

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();



注释ThreadSafeDemo1中run方法的synchronized关键字，出现上图结果。

ThreadSafeDemo1中Iterator是工作在一个独立的线程中，并且拥有一个 mutex锁，就是说Iterator在工作的时候，是不允许被迭代的对象被改变的。当出现多线程时，索引指针找不到迭代对象对象，出现错误。



加上synchronized关键字后，Iterator就相当于有了一个锁将list锁住，当一个线程访问时，其他线程无法操作，只有一个线程访问完，其他线程才可以访问，因此第一个线程访问元素应为1000，其他线程访问应为空0,。

/\* 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();

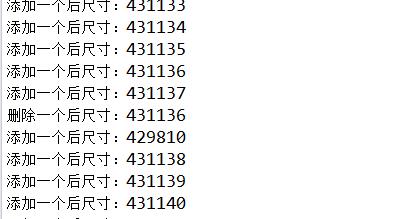
new Thread(demo8).start();

new Thread(demo9).start();

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

}

}



多线程对list操作时，发现打印的结果添加中出现了删除，打印结果出现异常，在ThreadSafeDemo2中修改语句，添加synchronized字段



修改后结果



同理，ThreadSafeDemo3一样添加synchronized关键字在相应位置，运行结果为