线程上机小结

线程：

1. 进程：进程是指运行中的应用程序，每个进程都有自己独立的地址空间（内存空间），比如用户点击桌面的IE浏览器，就启动了一个进程，操作系统就会为该进程分配独立的地址空间。当用户再次点击桌面的浏览器时，有启动了一个进程，操作系统将为心得进程分配新的独立的地址空间，目前操作系统都支持多进程。
2. 是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源只拥有一点在运行中比不可少的资源，但它可与同属一个进程的其他线程共享进程所拥有的全部资源，一个人线程可以创建和撤销另一个线程，同一进程中的多个线程之间可以并发执行，线程有就绪，阻塞，运行三种基本状态

新建状态（new）,就绪状态（Runnable）运行状态（Running），阻塞状态（Blocked）,死亡状态（Dead）

(3) 线程和进程的区别

线程是进程的子集，一个进程可以有很多线程，每条线程并发执行不同的任务，不同的进程使用不同的内存空间，而所有的线程共享一片相同的内存空间

（4）Thread类中的Start()方法和run()方法有什么区别

Start（）方法被用来启动新创建的线程，而且start（）内部调用了run（）方法，这和直接调用run（）方法是不一样样的，当调用run（）方法的时候，只会在原来的线程中调用，没有新的线程启动，start（）方法才会启动新的线程

(5) 线程安全问题

线程安全就是代码在多线程下执行和单线程下执行永远都获得一样的结果

线程安全级别:

1. 不可变

像String，Integer,Long这些，都是final类型的类，任何一个线程都改变不了他的值，要改变除非新创建一个，因此这些不可变得对象不需要任何同步手段就可以直接在多线程环境下使用

1. 绝对线程安全

不管运行环境如何，调用者都不需要额外的同步措施。例如CopyOnWriteArraryList,,CopyOnWriteArraySet

1. 相对线程安全

也就是我们通常意义上所说的线程安全，像vector这种，add,remove方法都是原子操作，不会被打断，但是页仅限与此，如果有个线程遍历某个vector,有个线程同时在add这个Vector,99%的情况下会出现ConcurrenModificationException,也就是fail-fast机制

1. 线程非安全

ArraryList,LinkedLst,HashMap等都是线程非安全的

（6）怎么解决线程安全问题？

通常我们使用同步（关键字为synchronized）来解决这种由于多线程同时操作的共享数据带来的线程安全问题

同步可以理解为：我们将多条操作共享数据的语句代码包成一个整体，让某个线程执行时，其他线程不能执行

同步方案包括三种方式，它们对应的锁对象事不一样的，另外我们可以通过加锁代码块，解决安全问题

注意：同步可以解决问题的根本原因就在于锁对象上，因此要避免线程安全问题，多个线程就必须使用同一个锁对象，否则，不能解决问题。

（7）解决方案：

1.同步代码块

Synchronized(对象) {

//需要被同步的代码

}

//这里的锁对象可以是任意的。

2.同步方法

就是在方法上加synchronizaed关键字，只时的锁对象是this。

3.静态同步方法

格式：将同步加在静态方法上，此时的锁对象为当前类的字节码文件对象

4.加锁Lock

将需要一次执行完代码块用lock（）和unlock（）包裹，来保证线程安全。

（8）java中线程安全的类

1，Vector与ArrartyList

Vector属于线程安全级别的，但大多数情况下都不用Vector,因为线程安全需要更大的系统开销，

2.HashTable与HashMap

HashTable中的方法是同步的，而HashMap中的方法是在缺省情况下是非同步的，再多线程并发的坏境下，可以直接用hashtable，但是要使用HashMap的话就要自己增加同步处理，HashTable 就是线程安全的，即HashTable都提供了同步机制，HashMap不是线程安全的，即不提供同步机制，HashTable不允许插入空值，HashMap允许

3.StringBuilder与StringBuffer

StringBuilder和StringBuffer的方法是一模一样的，StringBuffer线程安全，StringBuilder线程不安全，但是性能更高。

实验验证小结：

实验实例代码：

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.

\* Iterator迭代器的作用是：对集合进行迭代的迭代器 作用是将集合中的元素

\*/

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(list){ //家同步锁的作用

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);

}

}

}

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

**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，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());

}

}

}

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

**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 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();\*/

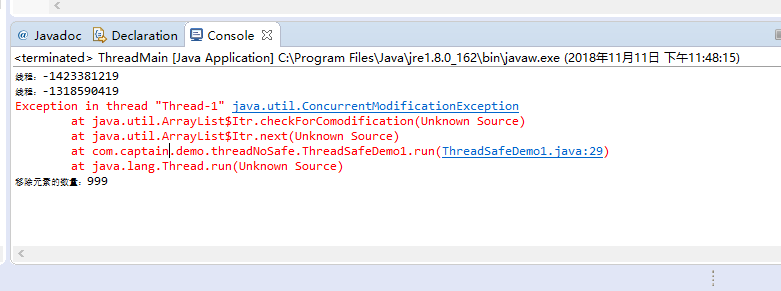
}

}

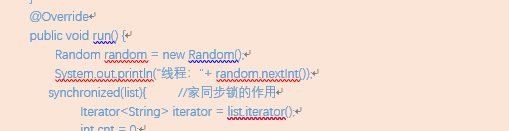
程序运行错误：

1. 在运行public class ThreadMain {

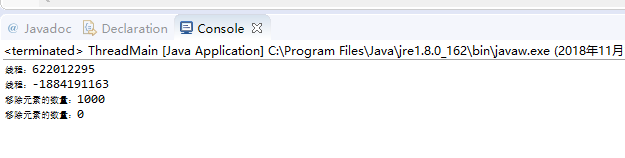
}中的第一个演示示例时出现如下错误：



然后将ThreadSafeDemo1中的this该为list 或者将所有的list改为vector，则锁住的对象this也改为vector



再次运行的结果为：



1. ThreadSafeDemo2和ThreadSafeDemo3是一个死循环，所以修改代码如下：

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

**import** java.util.List;

/\*\*

\* 线程安全演示类

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

\*/

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

**private** Vector<String>vector;

**public** ThreadSafeDemo2(Vector<String> vector){

**this**.vector = vector;

}

@Override

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

**while**(**vector.size()<1000**){

vector.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："+vector.size());

}

}

}

              }

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

**import** java.util.List;

/\*\*

\* 线程安全演示类

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

\*/

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

**private** Vector<String> vector;

**public** ThreadSafeDemo3(Vector<String> vector){

**this**.vector = vector;

}

@Override

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

**while**(**vector.size()>0**){

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

vector.remove(0);

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

}

}

}

}

运行结果：

