jdk源码&多线程&高并发-【阶段2、深入多线程设计模式】

讲师简介

smart哥，互联网悍将，历经从传统软件公司到大型互联网公司的洗礼，入行即在中兴通讯等大型通信公司担任项目leader，后随着互联网的崛起，先后在美团支付等大型互联网公司担任架构师，公派旅美期间曾与并发包大神Doug Lea探讨java多线程等最底层的核心技术。对互联网架构底层技术有相当的研究和独特的见解，在多个领域有着丰富的实战经验。

一、多线程设计模式概论

不是gof的23种设计模式（针对业务的非线程代码）

shi山

线程代码的设计模式（12种），是**doug lea** 和 josh bloch jsr166的规范，jsr133

第三阶段 学习juc的一个理论基础。老外是先有一套方法论，然后在这一套方法论的基础之上进行落地

E.F.CODD 数据库的范式

doug lea的主页：

http://gee.cs.oswego.edu/

根据doug lea的分类，多线程程序的评价标准如下：

1、安全性

不损坏对象

通常是指对象的属性出现了意想不到的情况，例如 i--，在没有保护的情况之下就有可能出现问题

安全性只是状态变化了

安全性和兼容性

arraylist如何安全？？

2、生存性

必要的处理能够被执行，出现假死，最典型的就是死锁

状态没有变，但是程序停了，或者即使程序没有死但是也是不动，卡死了（死锁）

生存性和安全性是相互制约的，一味的强调安全，那么就有可能出现生存性问题

3、可重复性

juc里面有大量的可重复性利用的类

4、性能

doug lea总结了影响性能的关键因素，如下：

1）吞吐量

2）响应性

3）容量

第1,2两点是必要的，第3,4两点是锦上添花

二、Single Threaded Execution模式

所谓“Single Threaded Execution”，即“以一个线程执行”，该模式用于设置限制，以确保同一时间内只让一个线程执行处理。

1、案例：多个人过安检门

结论：只能一个一个的通过，如果大家一起通过的话，那么就有可能出现数据不一致的问题

一个线程执行pass方法会经历1次写操作，2次读操作

2、1个线程执行临界区，能不能最多同时N个线程来执行临界区？

Semaphore 信号量

需求：一个饭店同时最多允许8个人用餐，如何实现？？

自定义一个semaphore

总结：**基于aqs。只要是基于aqs的实现理论上都是可以使用synchronized替代的**

三、Immutable模式

Immutable模式中存在着确保实例状态不发生改变的类（immutable类）。在访问这些实例时并不需要执行耗时的互斥处理，因此若能巧妙利用该模式，定能提高程序性能。

1、案例：和single threaded execution比较

没有写操作（类的属性不会发生变化），只有读操作

属性用final和private修饰 ，更加说明类中不会提供setter方法

总结：不可变模式就是属性没有写，只有读，所有不需要synchronized关键字修饰临界区

2、分析：jdk中哪些类是使用这种模式呢？

有一部分是immutable，还有一部分是mutable

String类和StringBuffer类

String类是immutable。StringBuffer类是mutable

标准类库中的immutable模式

ArrayList(可变的)

CopyOnWriteArrayList（可变部分给拷贝出来）

提问：immutable模式是否能够提升性能？？

可以提升性能，但是测试未必能测的出来，因为synchronized做了大量优化，要具备一定的条件才能测出来

综合案例：外部传入的可变对象如何保证安全？

见视频

四、Guarded Suspension模式

又叫"保护性暂挂"模式，如果执行现在的处理会造成问题，就让执行处理的线程等待，通过让线程等待来保证实例的安全性。

1、提问：保护的是谁？暂时挂起的又是谁？

在tomcat中，client发送request请求，server端接受到请求，然后做解析，缓存起来，然后server端线程会获取到request请求，然后对请求做进一步处理（交给springmvc去做了）

服务器的前端（依然在服务器这一端），专指服务接受请求

服务器后端，请求转发给框架

4个实体：

1）客户端线程，发送request到队列

2）request

3）queue

4）服务端线程，负责到队列里面获取request做进一步处理

保护的是queue，暂时挂起的是消费者线程，说白了就是队列里面如果没有东西，那么消费者线程必须等待

核心代码：

```
while（队列为空）{
    synchronized（queue）{
    wait（ ）；//挂起消费者线程
    }
}
```

这里其实有多个设计模式：

1）有synchronized，那么说明用了单线程执行的设计模式

2）Immutable模式（Request）

3）Guarded Suspension

五、Balking模式

如果现在不合适执行这个操作，或者没必要执行这个操作，就停止处理，直接返回。

有多个（起码2个）线程，某一个线程正在准备做某件事的时候，然后突然呢，另外一个线程抢先做了，那么这个线程就不执行这个动作了。

1、图解