# 一面

1. 自我介绍
2. 介绍一下项目
3. Java多线程有哪几种实现方式

https://mp.weixin.qq.com/s/39Sb60J0DC2mNN6DYvpFtg

## CountDownLatch

**CountDownLatch** : **一个线程**(或者多个)， 等待另外**N个线程**完成**某个事情**之后才能执行。  **CyclicBarrier**        : **N个线程**相互等待，任何一个线程完成之前，所有的线程都必须等待。  
这样应该就清楚一点了，对于CountDownLatch来说，重点是那个**“一个线程”**, 是它在等待， 而另外那N的线程在把**“某个事情”**做完之后可以继续等待，可以终止。而对于CyclicBarrier来说，重点是那**N个线程**，他们之间任何一个没有完成，所有的线程都必须等待。

两个看上去有点像的类，都在java.util.concurrent下，都可以用来表示代码运行到某个点上，二者的区别在于：

1）CyclicBarrier的某个线程运行到某个点上之后，该线程即停止运行，直到所有的线程都到达了这个点，所有线程才重新运行；CountDownLatch则不是，某线程运行到某个点上之后，只是给某个数值-1而已，该线程继续运行。

2）CyclicBarrier只能唤起一个任务，CountDownLatch可以唤起多个任务。

3) CyclicBarrier可重用，CountDownLatch不可重用，计数值为0该CountDownLatch就不可再用了。

### CountdownLatch原理

1.new CountDownLatch(int count)实际上



1. Sync 继承了AbstractQueuedSynchronizer（共享锁）
2. CountDownLatch 调用await() 实际上是一直在死循环，判断state的状态，state是volatile修饰，在各线程中都是可见的，只有当state=0时，

1. Spring的核心技术
2. 数据库事务
3. 登录验证的技术问题

# 二面：

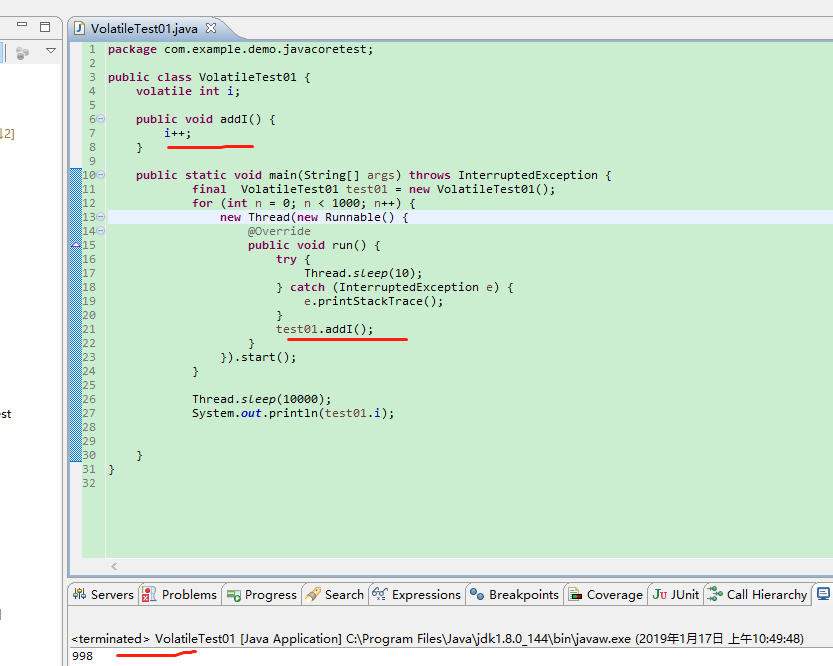
## Concurrent包

Java 并发包 Concurrent 的包结构共可分为五个部分：   
- 原子变量类   
- 锁   
- collection并发集合框架   
- excutor线程池   
- 同步工具

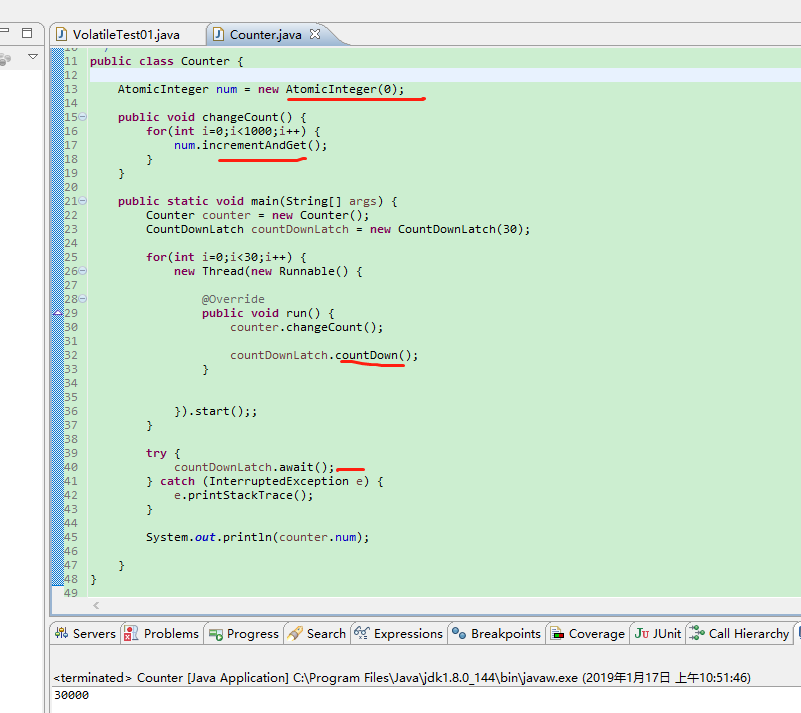
参考文档：https://www.cnblogs.com/my376908915/p/6758681.html

### Volatile/原子类操作

Volatile 保证了一个变量在各个线程间是可见的， 线程A 对变量做了修改，会立刻强刷到JAVA Memory Model中，线程B中的对变量的缓存就会失效， 但是要线程安全，volatil不能做复合性的操作， 如i++;



可以使用JAVA中的原子性的操作类Atomic。。。 通过CAS 方式保证原子性



### **Concurrent包之重入锁/读写锁/条件锁**

#### 重入锁

Java 中的重入锁即ReentrantLock, 与JVM中的synchronized 一样，是一种排它锁

ReentranLock 提供了定时锁，



可以被Interrupt的中断锁, synchronized是不能Interrupt的

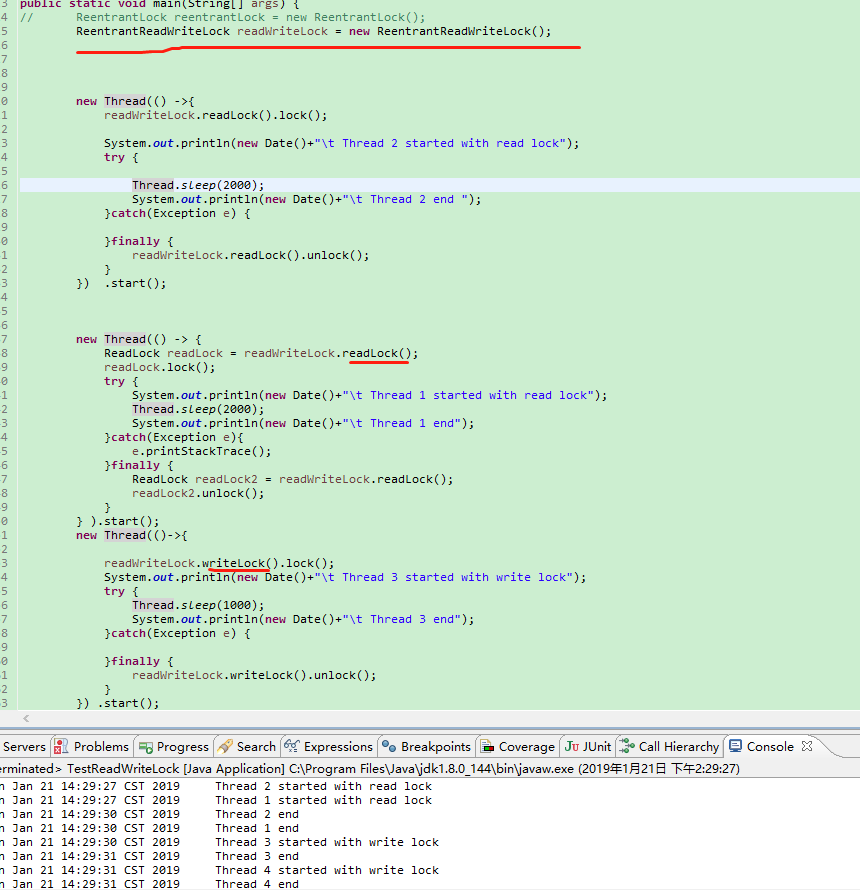


#### **读写锁**

**适用于读多写少得场景**

**锁可以保证原子性和可见性**

**原子性是针对写操作而言， 对于读多写少的场景，一个读操作无需阻塞其他读操作， 只要保证读和写， 写与写不同时发生即可。**



读写锁的规则是：

获得读锁后，其他线程可以获取读锁，但是不能获取写锁

获取写锁后，其他线程读锁，写锁都不能获取

#### **条件锁**

**重入锁可以创建若干个条件对象，signal ()或是singalAll()只能唤醒相同条件等待的线程，**

**可以更加细粒度地控制线程间的通信**

### ReentranLock的原理

ReentranLock的实现前提是AbstractQueueSynchronizer ,同步队列控制器，AQS

AQS自己维护的队列是当前等待资源的队列，AQS会在被释放后，依次唤醒队列中从前到后的所有节点，使他们对应的线程恢复执行，直到队列为空

1. 线程1调用reentrantLock.lock时，尝试获取锁。如果成功，则返回，从AQS的队列中移除线程；否则阻塞，保持在AQS的等待队列中。  
   2. 线程1调用await方法被调用时，对应操作是被加入到Condition的等待队列中，等待signal信号；同时释放锁。  
   3. 锁被释放后，会唤醒AQS队列中的头结点，所以线程2会获取到锁。  
   4. 线程2调用signal方法，这个时候Condition的等待队列中只有线程1一个节点，于是它被取出来，并被加入到AQS的等待队列中。注意，这个时候，线程1 并没有被唤醒，只是被加入AQS等待队列。  
   5. signal方法执行完毕，线程2调用unLock()方法，释放锁。这个时候因为AQS中只有线程1，于是，线程1被唤醒，线程1恢复执行。

发送signal信号只是将Condition队列中的线程加到AQS的等待队列中。

只有到发送signal信号的线程调用reentrantLock.unlock()释放锁后，这些线程才会被唤醒。  
  
可以看到，整个协作过程是靠结点在AQS的等待队列和Condition的等待队列中来回移动实现的，

Condition作为一个条件类，很好的自己维护了一个等待信号的队列，并在适时的时候将结点加入到AQS的等待队列中来实现的唤醒操作。

signal就是唤醒Condition队列中的第一个非CANCELLED节点线程，

而signalAll就是唤醒所有非CANCELLED节点线程，本质是将节点从Condition队列中取出来一个还是所有节点放到AQS的等待队列。

尽管所有Node可能都被唤醒，但是要知道的是仍然只有一个线程能够拿到锁，其它没有拿到锁的线程仍然需要自旋等待，就上上面提到的第4步(acquireQueued)。

把所知道的java的concurrent包的技术全部说出来（volatile ， 锁重入， LinkedTransferQueue字节追加提高并发度技术，ConcurrentHashMap结合volatile的Happen-before读取优化）

1. Redis的配置文件（AOF 和napshot 和主从复制）
2. Websocke的常连接问题
3. 秒杀业务场景的设计（事务，逻辑调整， 行级锁， 数据库并发度， mybatis调用存储过程）
4. 会不会写前端（jquery, css3）
5. 平时用什么开发工具，IDE 相比Eclipse有什么好
6. 讨论后台大并发量的问题
7. 你今后的研究方向是什么
8. 你有什么想问的

作者：要去看酷玩演唱会！  
链接：[https://www.nowcoder.com/discuss/72122](https://www.nowcoder.com/discuss/72122" \t "_blank)  
来源：牛客网

一面

jvm堆和栈都存的啥，区别，特点

- 静态变量存在哪

- 老年代，新生代是啥，以及相关知识

- hashmap原理

- mysql的联合索引

- mysql引擎有啥，各是啥特点，区别

- jvm调优

- 内存泄露，内存溢出

- 创建线程的方法，哪个更好，为什么

- future了解么

- 看过啥书，未来规划

- 你有什么想问我的

二面

- 为什么选java

- 写的技能都了解多少

- arp，arp攻击

- icmp

- 虚拟内存

- 项目遇到的问题

- 你有什么想问的