# 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：

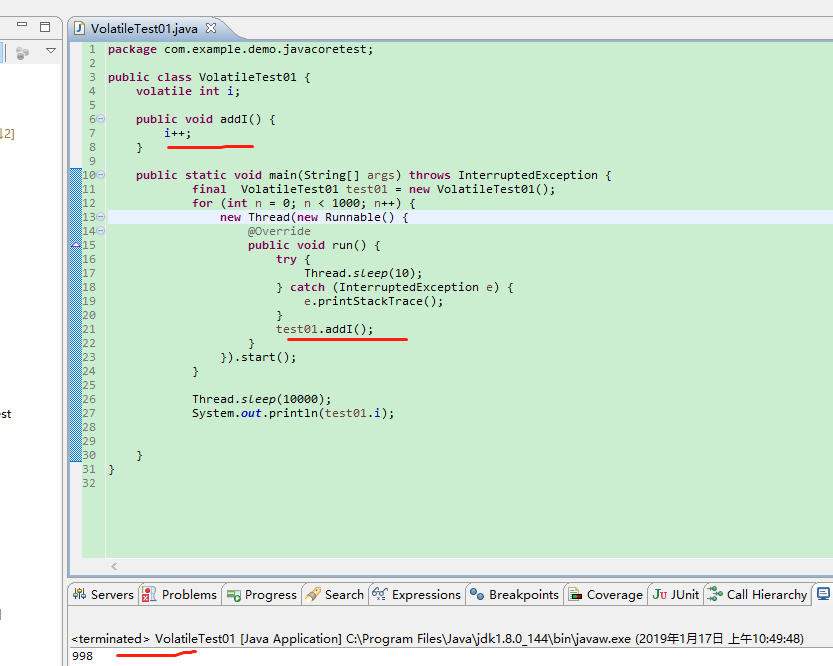
## Concurrent包

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

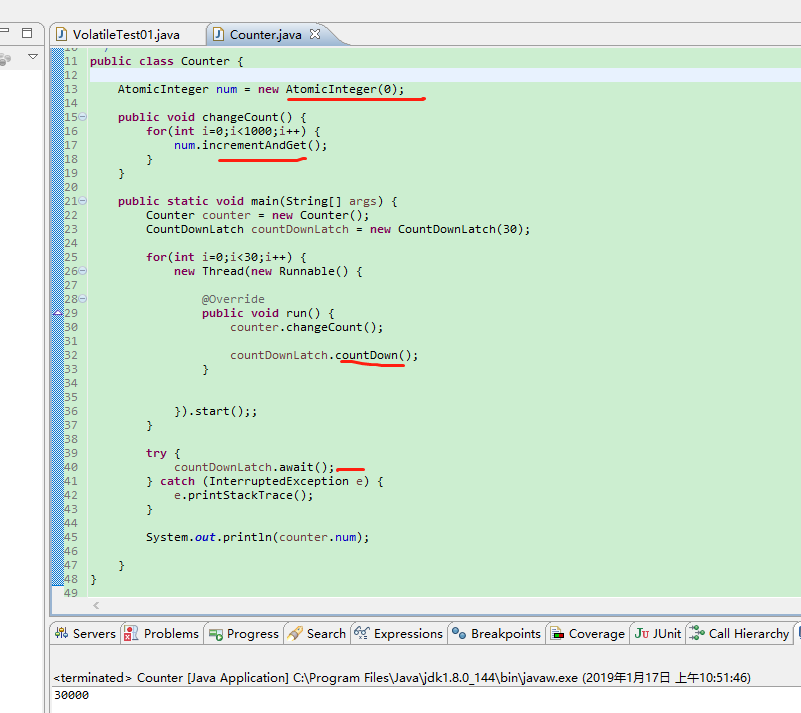
参考文档：https://www.cnblogs.com/my376908915/p/6758681.html

### 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

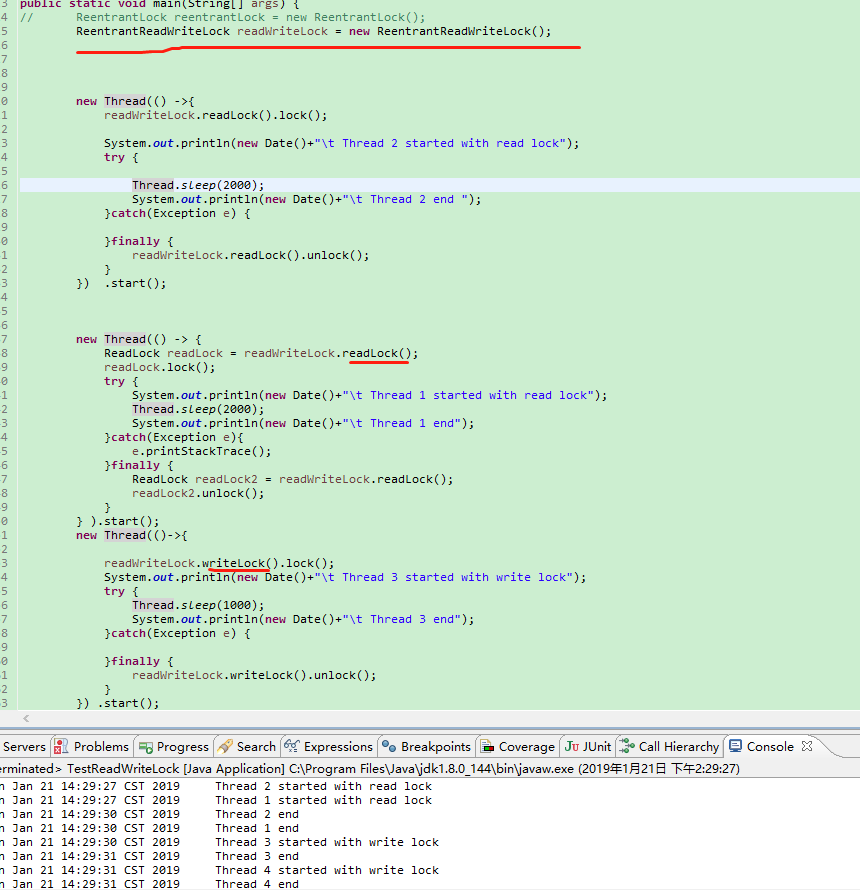


#### **读写锁**

**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)。

### 线程池

#### 为什么使用线程池

如果每次使用线程都创建一个，并发的线程数量太多， 每个线程执行任务的时间又短，频繁的创建和销毁线程会大大降低系统的效率， 所以使得线程能够复用， 就是执行完一个线程后，不被销毁，可以继续执行其他的任务。

#### ThreadPoolExecutor

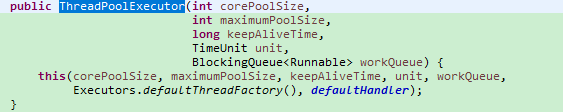








创建一个线程池：



* **corePoolSize（线程池的基本大小）**：当提交一个任务到线程池时，线程池会创建一个线程来执行任务，即使其他空闲的基本线程能够执行新任务也会创建线程，等到需要执行的任务数大于线程池基本大小时就不再创建。如果调用了线程池的prestartAllCoreThreads()或者prestartCoreThread()方法，线程池会提前创建并启动所有基本线程。
* **maximumPoolSize（线程池最大大小）**：线程池允许创建的最大线程数。如果**队列满了**，并且已创建的线程数小于最大线程数，则线程池会再创建新的线程执行任务。值得注意的是**如果使用了无界的任务队列这个参数就没什么效果**。
* **keepAliveTime（线程活动保持时间）**：线程池的工作线程空闲后，保持存活的时间。所以如果任务很多，并且每个任务执行的时间比较短，可以调大这个时间，提高线程的利用率。默认情况下，只有当线程池中的线程数**大于corePoolSize时，keepAliveTime才会起作用**，直到线程池中的线程数不大于corePoolSize，即当线程池中的线程数大于corePoolSize时，如果一个线程空闲的时间达到keepAliveTime，则会终止，直到线程池中的线程数不超过corePoolSize。但是如果调用了**allowCoreThreadTimeOut(boolean)方法**，在线程池中的线程数不大于corePoolSize时，keepAliveTime参数也会起作用，直到线程池中的线程数为0；
* **TimeUnit（线程活动保持时间的单位）**：可选的单位有天（DAYS），小时（HOURS），分钟（MINUTES），毫秒(MILLISECONDS)，微秒(MICROSECONDS, 千分之一毫秒)和毫微秒(NANOSECONDS, 千分之一微秒)。
* **runnableTaskQueue（任务队列）**：用于保存等待执行的任务的阻塞队列。可以选择以下几个阻塞队列。**一般使用LinkedBlockingQueue和Synchronous。**

1. ArrayBlockingQueue：是一个基于数组结构的有界阻塞队列，此队列按 FIFO（先进先出）原则对元素进行排序。
2. **LinkedBlockingQueue**：一个基于链表结构的阻塞队列，此队列按FIFO （先进先出） 排序元素，吞吐量通常要高于ArrayBlockingQueue。**静态工厂方法Executors.newFixedThreadPool()使用了这个队列**。
3. **SynchronousQueue**：一个不存储元素的阻塞队列。每个插入操作必须等到另一个线程调用移除操作，否则插入操作一直处于阻塞状态，吞吐量通常要高于LinkedBlockingQueue，静态工厂方法Executors.newCachedThreadPool使用了这个队列。
4. PriorityBlockingQueue：一个具有优先级得无限阻塞队列。

* **ThreadFactory**：用于设置创建线程的工厂，可以通过线程工厂给每个创建出来的线程**设置更有意义的名字**，Debug和定位问题时非常又帮助。
* **RejectedExecutionHandler（饱和策略）**：当队列和线程池都满了，说明线程池处于饱和状态，那么必须采取一种策略处理提交的新任务。这个策略默认情况下是AbortPolicy，表示无法处理新任务时抛出异常。以下是JDK1.5提供的四种策略。

ThreadPoolExecutor.AbortPolicy:丢弃任务并抛出RejectedExecutionException异常。

ThreadPoolExecutor.DiscardPolicy：也是丢弃任务，但是**不抛出异常**。

ThreadPoolExecutor.DiscardOldestPolicy：丢弃队列**最前面的任务**，然后重新尝试执行任务（重复此过程）

ThreadPoolExecutor.CallerRunsPolicy：由调用线程处理该任务

execute()

submit()

shutdown()

shutdownNow()

execute()方法，这个方法是ThreadPoolExecutor的核心方法，通过这个方法可以向线程池提交一个任务，交由线程池去执行。

submit()方法，这个方法也是用来向线程池提交任务的，但是它和execute()方法不同，它能够返回任务执行的结果，去看submit()方法的实现，会发现它实际上还是调用的execute()方法，只不过它**利用了Future来获取任务执行结果**。

我们可以通过这个future来判断任务是否执行成功，通过future的get方法来获取返回值，**get方法会阻塞住直到任务完成。**而使用get(long timeout, TimeUnit unit)方法则会阻塞一段时间后立即返回，这时有可能任务没有执行完。

Object s = future.get();

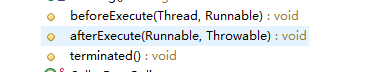
shutdown()，原理是只是将线程池的状态设置成SHUTDOWN状态，然后中断所有没有正在执行任务的线程。

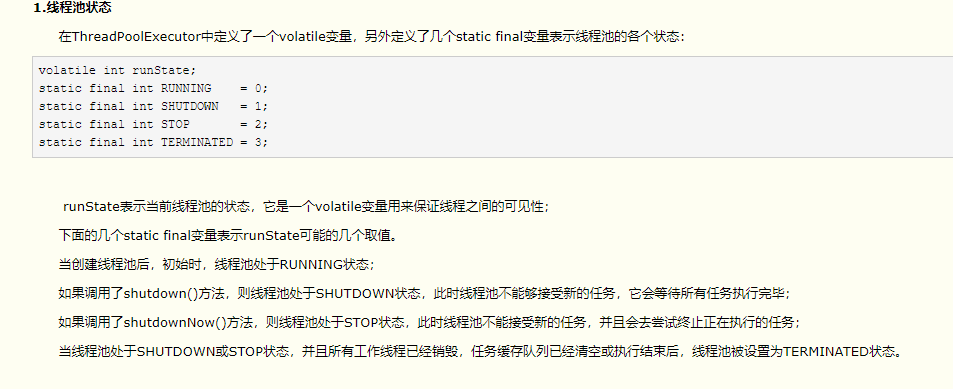
shutdownNow()，原理是遍历线程池中的工作线程，然后逐个调用线程的interrupt方法来中断线程，所以无法响应中断的任务可能永远无法终止。shutdownNow会首先将线程池的状态设置成STOP，然后尝试停止所有的正在执行或暂停任务的线程，并返回等待执行任务的列表。

只要调用了这两个关闭方法的其中一个，isShutdown方法就会返回true。

当所有的任务都已关闭后,才表示线程池关闭成功，这时调用isTerminaed方法会返回true

#### 线程池的原理

1. 当前线程数小于corePoolSize, 每来一个任务， 就创建一个线程
2. 当当前线程>= corePoolSize, 但是队列还没满， 每来一个任务就丢到任务队列
3. 当当前线程>=corePoolSize, 并且对列已满，但是线程池中没有达到maximumPoolSize, 就创建新的线程，去执行任务
4. 当线程池中的线程已经达到maximumPoolSize， 且线程池已经满， 就会**采取任务拒绝策略**进行处理。
5. 可以自定义任务拒绝策略
6. 也可以extends ThreadPoolExecutor ,监控线程池的状态，beforeExecute， 
7. 线程池的状态有：



#### 使用



### LinkedTransferQueue

一个由链表结构组成的无界阻塞队列， 当存在一个正在等待的消费线程， 则立即移交， 如果没有，则插入元素e到队列的尾部，并进入阻塞状态到有消费者线程取走该元素

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