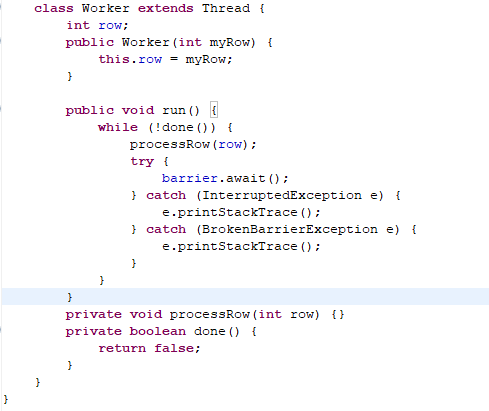
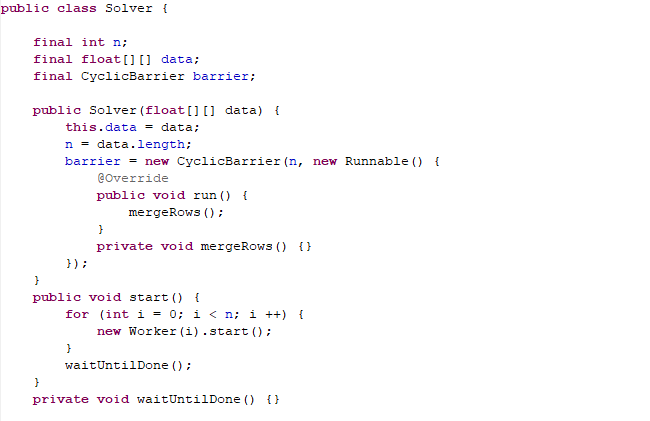
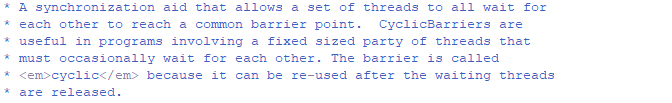
# CyclicBarrier

先来看一个官方的示例：



这是一个合并列的功能，每一列都交给一个子线程处理，当所有子线程都处理完阻塞在barrier.await这个点，执行CyclicBarrier传入的合并列的任务，合并任务执行完唤醒所有的子线程。

看一下官方对于CyclicBarrier的说明：

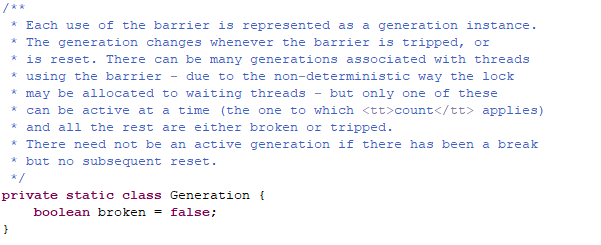


允许一组线程等待彼此执行到某个公共的点，然后在继续执行。CyclicBarrier可以用来在所有子线程都到达了某一个点上然后执行一些公共的操作，例如数组元素的求和，可以将数组拆分为多个子数组，为每个子数组分配一个线程执行对子数组元素的求和过程，最后等待所有子数组都求和完毕执行公共操作，合并所有子数组的和。

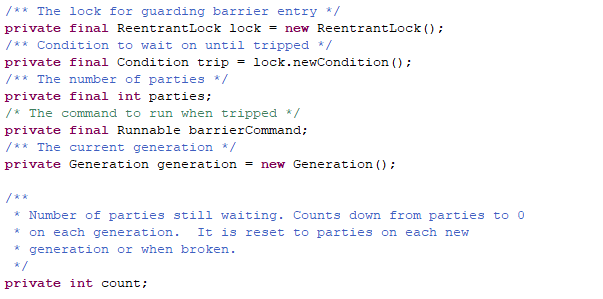
CyclicBarrier可以重置barrier状态，重复使用。

CyclicBarrier使用全破环模型，当子线程在阻塞过程中被中断，其他阻塞或者即将阻塞的线程将终止，并抛出BrokenBarrierException异常。

# 源码分析



每个CyclicBarrier都有一个Generation实例代表，只有在barrier被打破（正常通过或者异常）或者调用reset重置barrier状态时，Generation会改变。



lock保证进入barrier是同步的。

trip调用await的线程会阻塞在该条件上。

parties进入barrier的线程数。

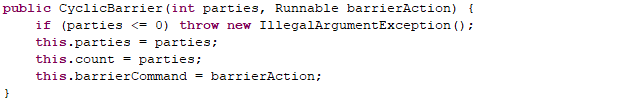
barrierCommand当barrier正常通过时执行的任务。

generation每个barrier的Generation实例代表。

count仍需要阻塞的线程的数量，在每个generation中count从parties减少到0，在新的generation中或者generation被打破之后重置count为parties。

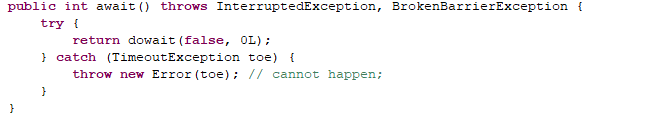
## 1.1构造器





初始化了parties，count，barrierCommand。

## 1.2await

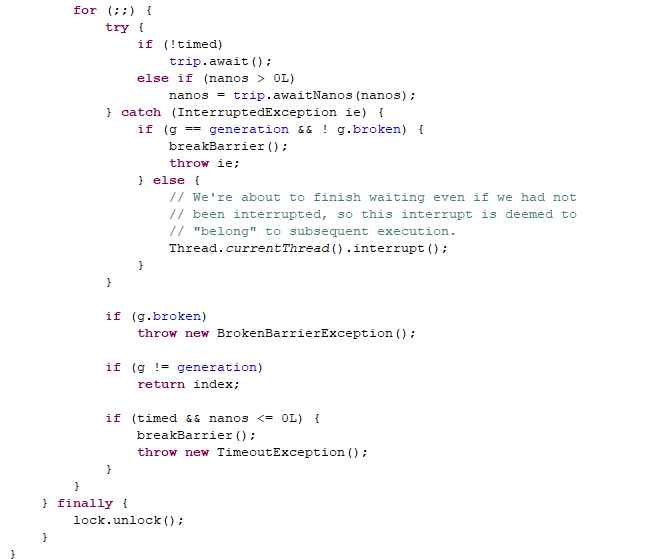


调用await的线程会一直阻塞直到所有相关的线程都调用await之后。

如果当前线程不是最后调用await的线程，以下情况可能导致失败或者唤醒。

1. 所有相关线程已经执行await，线程正常唤醒继续执行。
2. 其他线程中断了当前线程，当前线程抛出InterruptedException异常，其他相关线程抛出BrokenBarrierException异常。
3. 任一相关线程在阻塞期间被中断，被中断线程抛出InterruptedException异常，当前线程与其他相关线程抛出BrokenBarrierException异常。
4. 其他相关线程超时。
5. 其他相关线程调用reset重置barrier。

## 1.3dowait



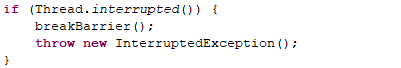
dowait有两个参数，一个timed表示超时等待标志，另一个nanos表示超时时间。在方法执行期间一直处于加锁状态。

下面看一下执行过程：

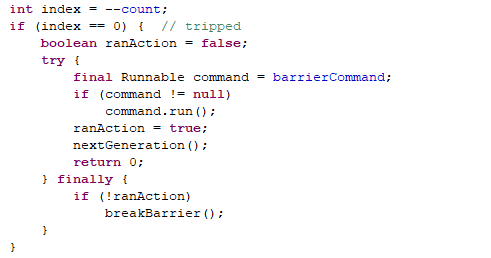
1. 首先检查barrier是否已经被打破，有可能在当前线程调用await之前，其他线程由于被中断打破了barrier。如果当前barrier被打破，抛出BrokenBarrierException异常。



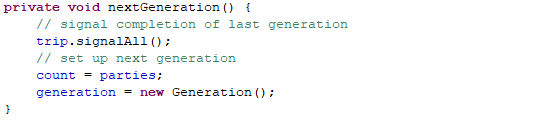
1. 检查当前线程中断状态，如果当前线程已经被中断，打破当前barrier被抛出InterruptedException异常。



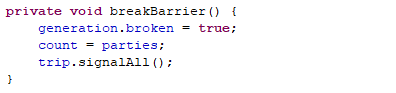
1. count表示还需要阻塞的线程的数量。如果当前线程是最后一个到达的线程，并且在创建CyclicBarrier时指定了要执行的任务barrierCommand，那么执行barrierCommand任务，barrierCommand正常执行完成之后开启新的barrier（CyclicBarrier可以重复使用的原因），如果在barrierCommand任务执行期间出现异常，那么打破当前barrier。



看一下nextGeneration和breakBarrier操作做了哪些事情。

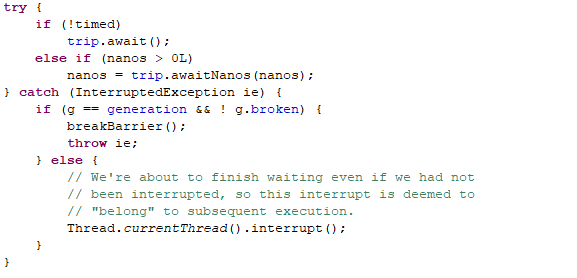


nextGeneration首先唤醒了所有在当前barrier上阻塞的线程。然后将count重置为parties并且将generation指向了一个新的实例。



breakBarrier首先将generation的属性broken设置为true，表示当前的barrier已经被打破，已经阻塞的线程和后续进来的线程在检查到这个标志之后将抛出BrokenBarrierException异常。然后唤醒阻塞在当前barrier上的线程。

1. 当前线程不是最后进入barrier的线程，如果启用了超时等待那么调用awaitNanos，否则调用await。如果在等待期间，当前线程被中断并且barrier没有被打破，那么打破barrier，抛出InterruptedException异常。



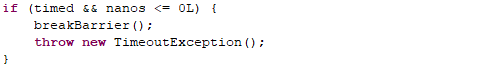
1. 这时线程被唤醒，首先检查barrier是否被打破。



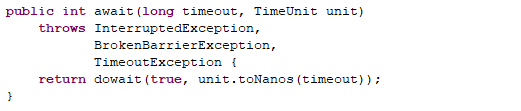
1. 判断当前的barrier是否已经正常结束并返回是第一个到达barrier的。



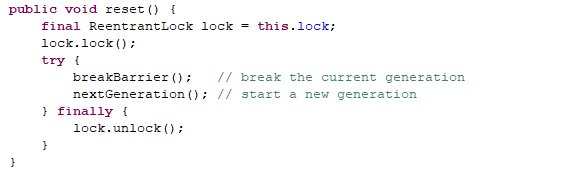
1. 检查当前线程是否已经超时等待了，如果超时了，打破barrier，抛出TimeoutException。



## 1.4超时await



## 1.5reset



首先打破当前barrier，然后开启新的barrier。