1. Java中的并发工具类
2. 概述

juc中提供了几个并发工具类，如CountDownLatch，CyclicBarrier和Semaphore工具类提供了一种并发流程控制的手段，Exchanger工具类提供了在线程之间交换数据的手段。

1. CountDownLatch

CountDownLatch允许一个或多个线程等待其他的线程完成操作，也就是在某些线程完成操作后指定的线程才会继续执行。这个功能可以通过join()方法实现，join()方法的作用就是让当前执行线程等待join线程执行结束，就是不停地检查join线程是否存活，如果join线程存活则让当前线程一直等待，wait(0)表示永远等待下去。

JDK5之后的juc包中的CountDownLatch也可以使用join()的功能，它的构造函数接受一个int整数作为计数器，如果想要等待N个点完成操作，就传入N。当其他的线程调用CountDownLatch的countDown方法时计数器就会-1，而await()方法则会阻塞当前的线程直到计数器变为0，当然await()方法也有带有计时时间的重载方法，以免永远等待下去。

注意，CountDownLatch的计数器是无法重新初始化的或者修改计数器的值。

1. 同步屏障CyclicBarrier

它的作用于CountDownLatch类似，也是等待一组线程执行，但是它是作为一个屏障存在的，即每个线程到达该屏障后阻塞，直到所有需要等待的线程都执行完毕后，这些阻塞的线程才能继续执行。同时，它的构造函数还有另一种，如下所示：

CyclicBarrier(int parties, Runnable barrierAction)

这个Runnable在所有阻塞的线程到达屏障后优先执行，然后放行所有阻塞的线程。同步屏障可以用于多线程计算数据，然后合并计算结果的场景。CyclicBarrier的计数器可以使用reset()方法重置。

1. 信号量Semaphore

信号量是用来控制同时访问特定资源的线程数量，构造函数传入指定的线程数量N，那么最多只有N个线程能够访问指定的资源。要使用信号量的线程首先通过Semaphore的acquire()方法获取一个许可证，使用完毕后调用release()方法归还许可证，还可以使用tryAcquire()尝试获取许可证。

1. 数据交换Exchanger

Exchanger是一个用于线程间数据交换的工具类，提供一个同步点，在这个同步点处，两个线程可以交换彼此的数据，通过exchange()方法交换，如果第一个线程先执行exchange()方法则它会持续等待第二个线程，等到两个线程都执行了exchange()方法后，这两个线程就可以交换数据了。

1. 线程池技术
2. 线程池处理流程
3. 提交一个任务到线程池，线程池判断核心线程池中的线程是否都在执行任务，如果不是则创建一个新的工作线程来执行任务。如果核心线程池里的线程都在执行任务，则进入下一个流程；
4. 线程池判断工作队列是否满了，如果没有则将新提交的任务放入工作队列中，如果工作队列满了，则进入下一个流程；
5. 判断线程数是否到达了最大线程数，如果没有则创建一个新的工作线程来执行任务，否则交给拒绝策略处理这个任务。
6. 工作线程

线程池创建线程时，会将线程封装为工作线程Worker，Worker在执行完任务后，还会循环获取工作队列中的任务来执行。

1. 线程池的使用
2. 创建线程池

线程池有7个参数：

1. corePoolSIze

线程池核心线程数，线程池中至少有这些线程。

1. maximumPoolSIze

最大线程数量，如果工作队列满了，且创建的线程数量小于最大线程数，则线程池会再创建新的线程，注意如果使用了无界队列则该参数没有意义。

1. KeepAliveTime

线程空闲的存活时间。

1. TimeUnit

线程存活时间单位。

1. ThreadFactory

线程工厂，可以给每个线程设置名称。

1. runnableTaskQueue

任务队列，用于保存等待执行的任务的阻塞队列。

1. RejectedExecutrionHandler

拒绝策略，当队列和线程池都满了，需要采用拒绝策略来拒绝任务请求。

1. 向线程池提交任务

可以使用两个方法向线程池提交任务，分别是execute()和submit()方法。execute()方法提交不需要返回值的任务，无法判断任务是否执行；submit()方法提交需要返回值的任务，返回一个Future类型的对象，通过该对象可以判断任务是否执行成功，可以通过Future对象的get()方法获取返回值，get()方法会一直阻塞当前线程直到任务完成。

1. 关闭线程池

shutdown()或者shutdownNow()方法都可以关闭线程池，遍历线程池中的工作线程，逐个调用线程的interrupt方法中断线程，无法响应中断的任务可能永远无法终止。shutdownNow首先将线程池的状态设置为STOP，然后尝试停止所有正在执行或暂停的任务；shutdown设置状态为SHUTDOWN，然后中断所有没有执行的任务线程。通常使用的是shutdown方法，让已经执行的任务都执行完成，如果不需要让任务都执行完成，则使用shutdownNow方法。