1. **几种线程池的区别**

* **newFixedThreadPool**

newFixedThreadPool创建一个指定工作线程数量的线程池。每当提交一个任务就创建一个工作线程，如果工作线程数量达到线程池初始的最大数，则将提交的任务存入到池队列中。

* **newCachedThreadPool**

1).工作线程的创建数量几乎没有限制(其实也有限制的,数目为Interger. MAX\_VALUE), 这样可灵活的往线程池中添加线程。   
2).如果长时间没有往线程池中提交任务，即如果工作线程空闲了指定的时间(默认为1分钟)，则该工作线程将自动终止。终止后，如果你又提交了新的任务，则线程池重新创建一个工作线程。

* **newSingleThreadExecutor**

newSingleThreadExecutor创建一个单线程化的Executor，即只创建唯一的工作者线程来执行任务，如果这个线程异常结束，会有另一个取代它，保证顺序执行(我觉得这点是它的特色)。单工作线程最大的特点是可保证顺序地执行各个任务，并且在任意给定的时间不会有多个线程是活动的 。

* **newScheduleThreadPool**

newScheduleThreadPool创建一个定长的线程池，而且支持定时的以及周期性的任务执行，类似于Timer。(这种线程池原理暂还没完全了解透彻)

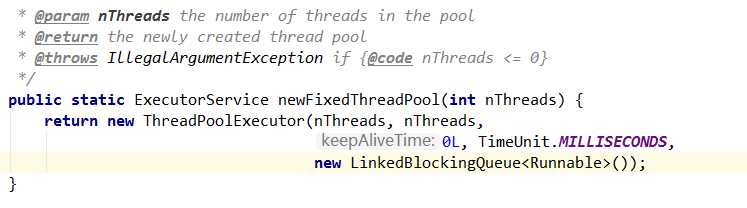
* **newWorkStealingPool**

**1.1** **newFixedThreadPool**

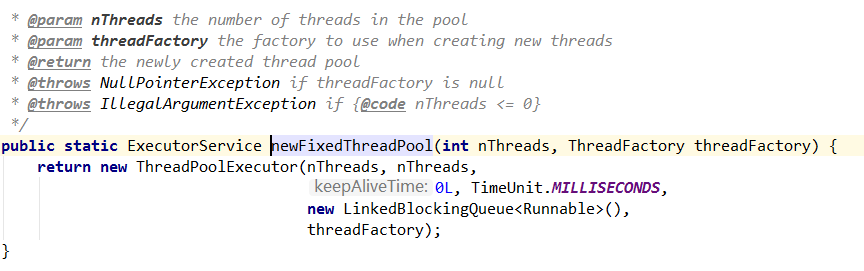
**查看**newFixedThreadPool的构造函数：

NewFixedThreadPool的构造函数有两个：

1)



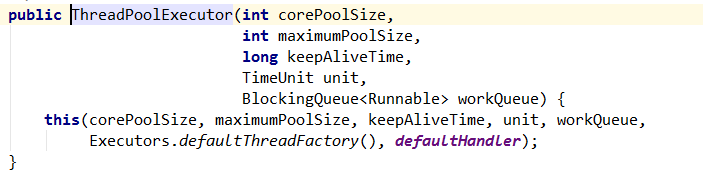
2)



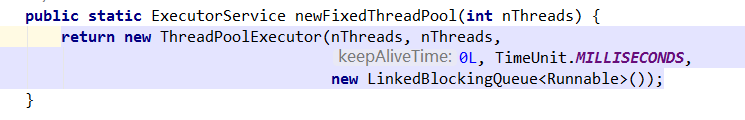
**1.1.1 构造函数情况1**

现在先看第一个构造函数，参数只有一个nThreads，线程池中的线程数，超出线程数的部分会加入LinkedBlockingQueue的等待队列，而LinkedBlockingQueue的理论值最大为int的最大值，那么现在思考一个问题，如果超出这个队列后，还有任务继续过来会怎样，可以查看跟踪查看Executors.newFixedThreadPool(num)和ThreadPoolExecutor的源码：

进行Executors.newFixedThreadPool(num)的时候，代码会返回一个ThreadPoolExecutor实例，查看ThreadPoolExecutor的构造函数，



对比newFixedThreadPool的构造函数：



* ThreadPoolExecutor线程池几个参数如下：

核心线程数：nThreads；

最大线程数：nThreads；

keepAliveTime：0L；(超过keepAliveTime的时间，如果还没有新任务，则释放核心线程数之外的线程，保留核心线程数数目的线程)

时间单位：秒；

等待队列：BlockingQueue；

* 从ThreadPoolExecutor构造函数可以看到这里的核心线程数和最大线程数是相同的；
* 线程池执行器逻辑：

1. 在一开始的时候，线程池是完全没有线程的；
2. 当有任务需要执行，则开始创建线程，并且当线程数少于核心线程数时，该部分的线程会一直保持正常状态而不会回收，线程数量在核心数量以内的，一旦创建就会保持这个数目的线程；
3. 当线程数已经大于或等于核心线程数，但是又继续有新的任务进来，那么会比较当前运行的线程数是否小于最大线程数，如果小于，则继续创建新的线程，超出核心线程数部分的空闲线程，会在超过keepAliveTime时自动回收；
4. 如果正在运行的线程任务已经大于等于最大线程数，而又继续有新的线程进来，那么这些任务就会被加入到等待队列，等待队列的理论值为整数型int的最大值；
5. 如果等待队列已经满了，那么会检查此刻是否有空闲线程，如果有则把任务交给空闲线程去执行，如果没有则抛出异常，告诉使用者已经不能再接收新的任务了；

* 关于队列满了后并且没有空闲的线程，默认的情况下才会抛出异常，因为ThreadPoolExecutor的构造函数实际上可以提供预定义处理策略：

1) 默认的ThreadPoolExecutor.AbortPolicy 处理程序遭到拒绝将抛出运行时RejectedExecutionException；

2) ThreadPoolExecutor.CallerRunsPolicy 线程调用运行该任务的 execute 本身。此策略提供简单的反馈控制机制，能够减缓新任务的提交速度；

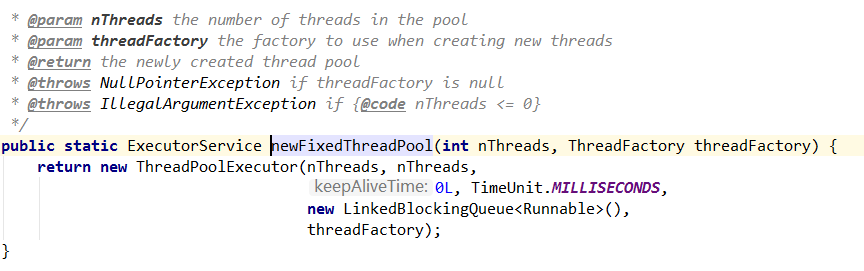
3) ThreadPoolExecutor.DiscardPolicy 不能执行的任务将被删除；

4) ThreadPoolExecutor.DiscardOldestPolicy 如果执行程序尚未关闭，则位于工作队列头部的任务将被删除，然后重试执行程序（如果再次失败，则重复此过程）；

测试demo：NewFixedThreadPoolDemoTest；

**1.1.2 构造函数情况2**

1.1.1已经讨论了不带ThreadFactory构造函数的时候，现在来讨论带有ThreadFactory参数的构造函数，比较两者的差异；



* ThreadFactory的作用是生产线程，可以对生产的线程进行计数统计、设置分组、优先级等等操作；
* 那么其实两者的区别就在于是否拥有一个生产线程的线程工厂；

**2.1 newCachedThreadPool**

newCachedThreadPool跟newFixedThreadPool的构造函数一样有两个，既然从上面可以得知两个的区别，那么在测试中就直接使用带有ThreadFactory参数的构造函数，这样方便监控执行任务的线程是新建的，还是使用了空闲线程；测试demo：NewnewCachedThreadPoolDemoTest；

* 经过测试demo：NewnewCachedThreadPoolDemoTest的测试，可以得出结论：

当有任务来时，线程池会产生线程来执行任务，而当任务执行完成后，线程成为空闲线程，如果有新的任务来临，那么系统就会把该空闲线程用来继续执行任务，而当没有新的任务来临，空闲线程超过了时间（默认是一分钟）没有执行任务，那么该线程就会回收，下次再有任务来临时，系统里如没有了空闲线程，那么就会重新创建新的线程来执行任务；跟newFixedThreadPool的区别在于，线程池创建线程没有核心线程数、最大线程数的概念，线程能够持续创建的线程数是无限的（理论上是有的，数目为整数型的最大值）。

**待解决问题：**

* 守护线程；
* 回想短信网关连接，模拟实现网关监控的守护线程；
* 查看多线程资料，了解yield等方法的使用；
* 四种线程池各自的优点和缺点；