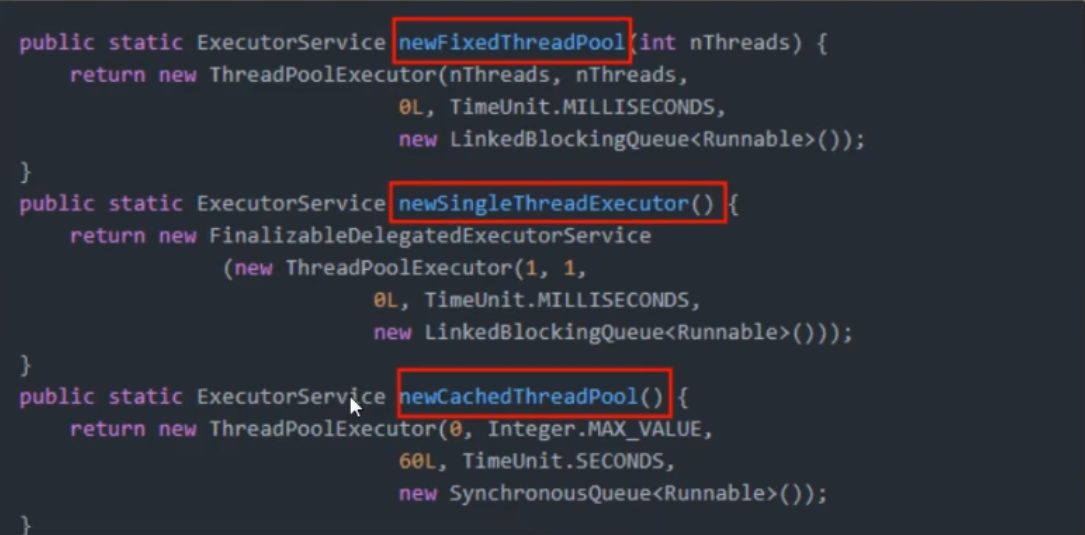
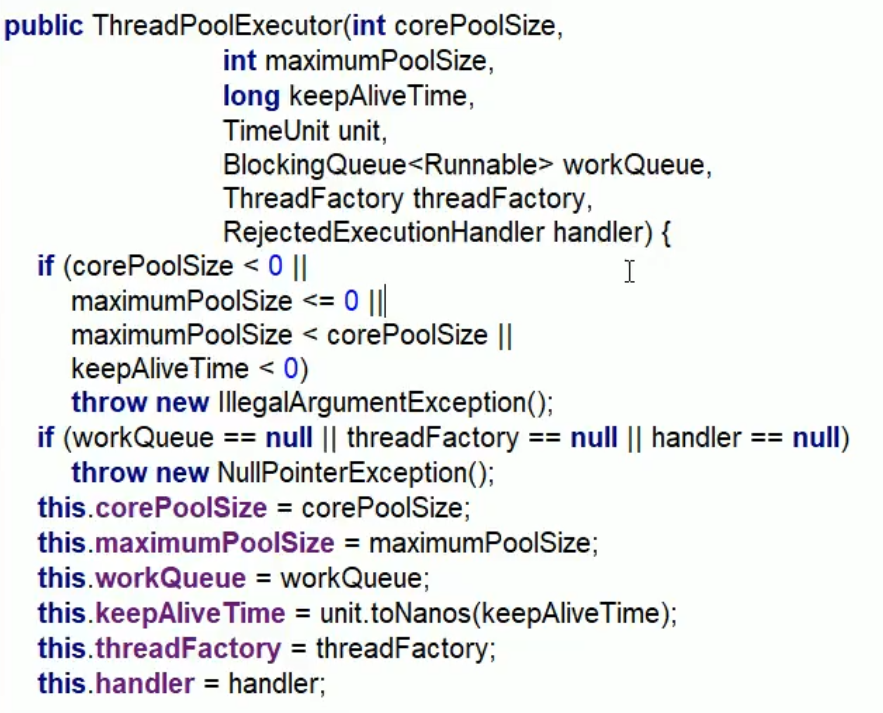
1. ThreadPoolExecutor
   1. 在上一章中我们学习到Executors中有三个创建线程池的方法：实际上这些方法的底层都是由一个方法实现（ThreadPoolExecutor）。



* 1. 由上面的代码可以看出Executor的底层代码是由ThreadPool实现，而ThreadPoolExecutor的底层又是说什么呢？

接下来我们看看ThreadPoolExecutor的源代码。

我们首先来分析线程池ThreadPoolExecutor的七大参数;

·corePoolSize:线程池中常驻核心线程数

·maximumPoolSize：线程池中能容纳的同时执行的最大线程数，此值必须大于1。

·keepAliveTime:多余的空闲线程的存活时间。当前池中线程数超过corePoolSize时，当前空闲时间达到keepAliveTime时，多余线程会被销毁直到只剩下corePoolSize个线程为止。

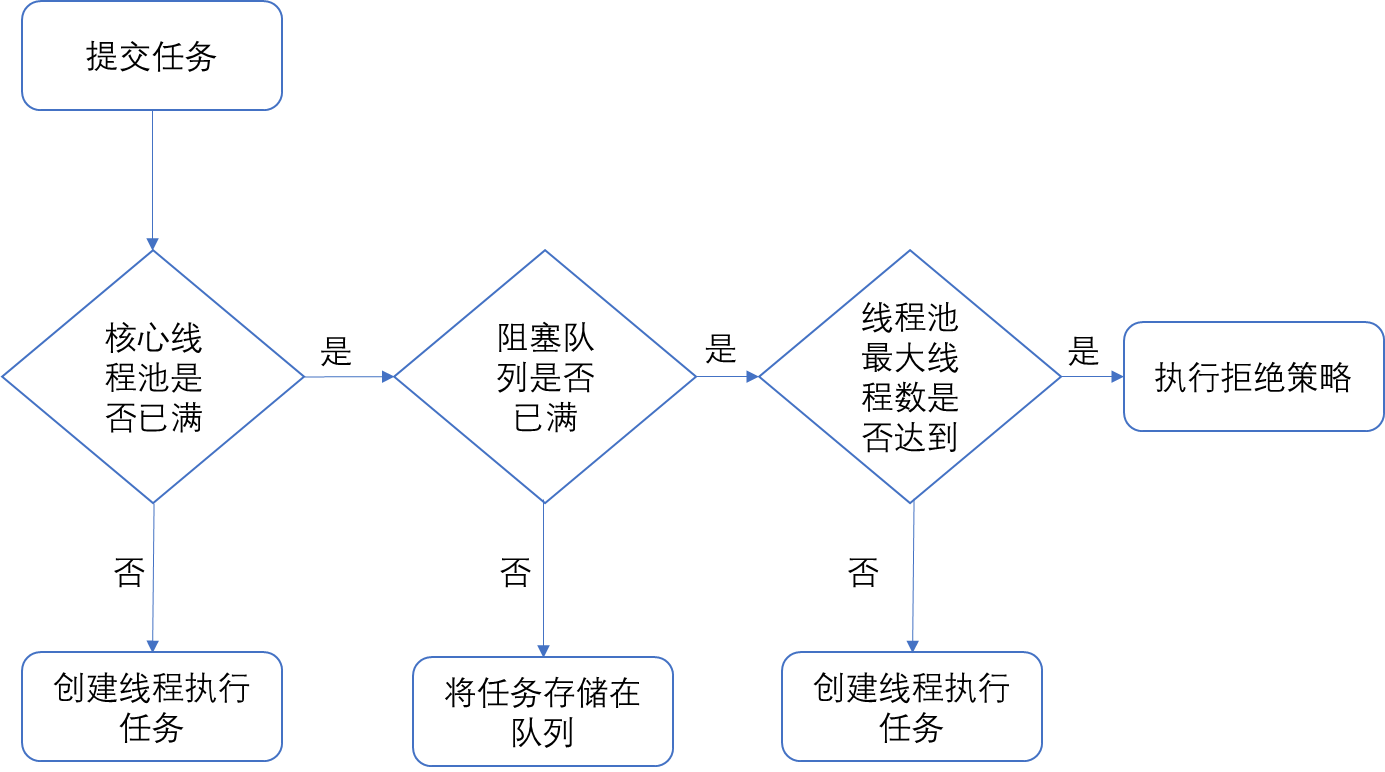
·unit: keepAliveTime的单位

·workQueue:任务队列，被提交但是还没执行的任务。（当需要的线程超过线程池中已有的那么没有抢到线程的任务，就会进入阻塞队列，当有空闲的时候就会取出。）

·threadFactory:表示生成线程池中工作线程的线程工厂，用于创建线程，一般默认值就可以了。

·hander：拒绝策略，表示当前队列满了，并且工作线程大于等于线程池中的最大线程数（maximumPoolSize）时如何来拒绝请求执行的runnable的策略。

线程池执行流程

 总结--以下是重点:

1、在创建了线程池后，开始等待请求。

2、当调用execute()方法添加一个请求任务时，线程池会做出如下判断:

2.1如果正在运行的线程数量小于corePoolSize，那么马上创建线程运行这个任务;2.2如果正在运行的线程数量大于或等于corePoolSize，那么将这个任务放入队列;

2.3如果这个时候队列满了且正在运行的线程数量还小于maximumPoolSize，那么还是要创建非核心线程立刻运行这个任务;2.4如果队列满了且正在运行的线程数量大于或等于maximumPoolSize，那么线程池会启动饱和拒绝策略来执行。

3、当一个线程完成任务时，它会从队列中取下一个任务来执行。

4、当一个线程无事可做超过一定的时间(keepAliveTime）时，线程会判断:

如果当前运行的线程数大于corePoo1Size，那么这个线程就被停掉。所以线程池的所有任务完成后，它最终会收缩到corePoolSize的大小。