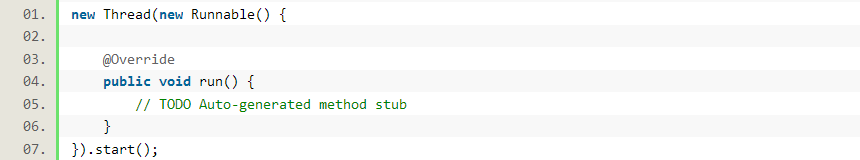
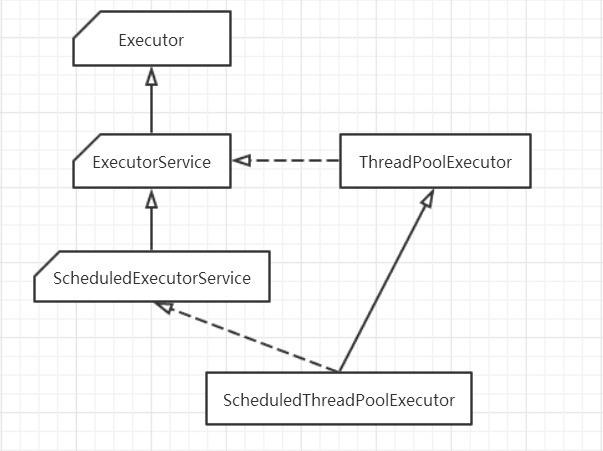
直接new Thread()方式创建线程代码为：



new Thread的弊端如下：  
a. 每次new Thread新建对象性能差  
b. 线程缺乏统一管理，可能无限制新建线程，相互之间竞争，及可能占用过多系统资源导致死机或OOM  
c. 缺乏更多功能，如定时执行、定期执行、线程中断

相比new Thread，Java提供的**四种线程池**的好处在于：  
a. 重用存在的线程，减少对象创建、消亡的开销，性能佳  
b. 可有效控制最大并发线程数，提高系统资源的使用率，同时避免过多资源竞争，避免堵塞  
c. 提供定时执行、定期执行、单线程、并发数控制等功能



Java通过Executors提供四种线程池，分别为：

newCachedThreadPool

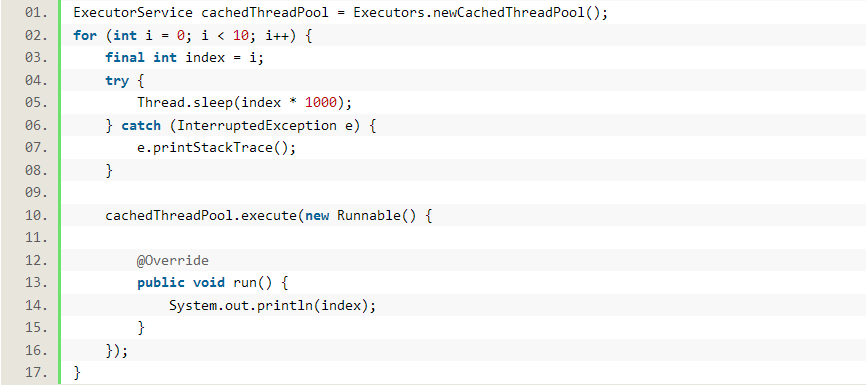
创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收线程，则新建  
newFixedThreadPool

创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待  
newScheduledThreadPool

创建一个定长线程池，支持定时及周期性任务执行  
newSingleThreadExecutor

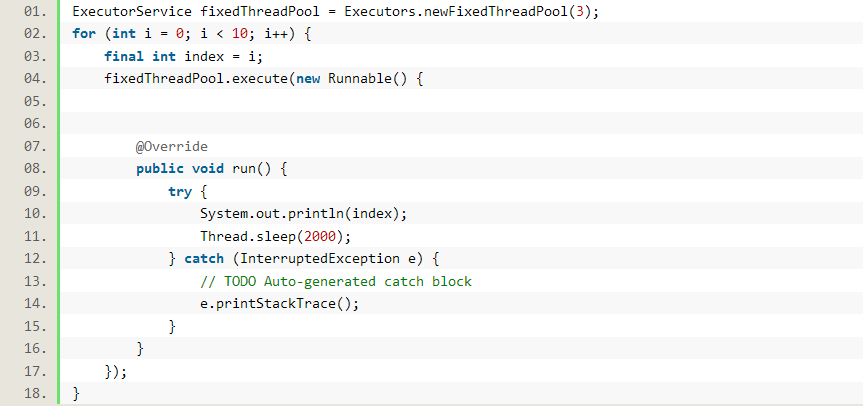
创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行

newCachedThreadPool:



线程池为无限大，当执行第二个任务时第一个任务已经完成，会复用执行第一个任务的线程，而不用每次新建线程

newFixedThreadPool:



第1个线程打印出0，然后中止；

第2个线程打印出1，然后中止；

第3个线程打印出2，然后中止；

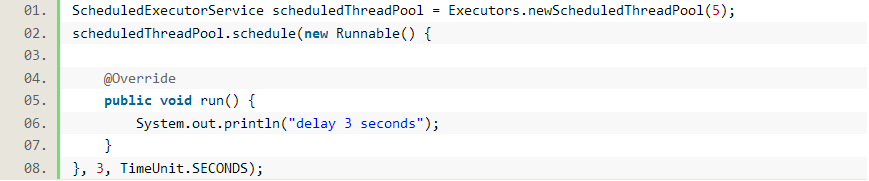
睡眠2秒后：

第1个线程打印出3，然后中止;

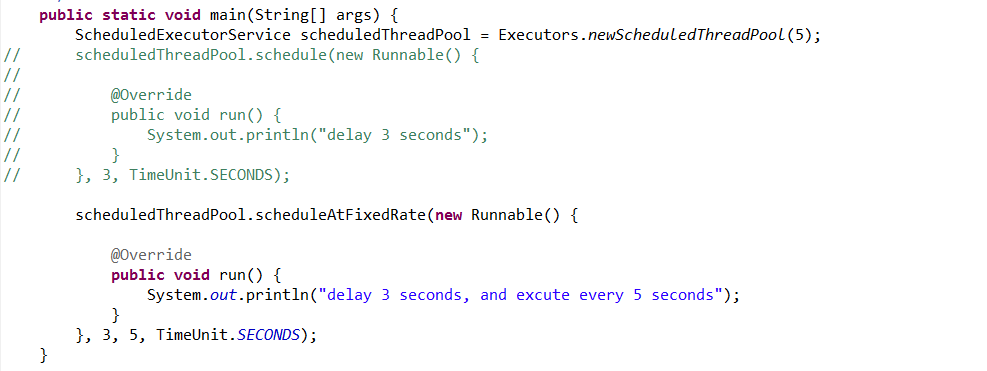
... ...

线程池大小为3，每个任务输出index后sleep 2秒，所以每两秒打印3个数字

newScheduledThreadPool :

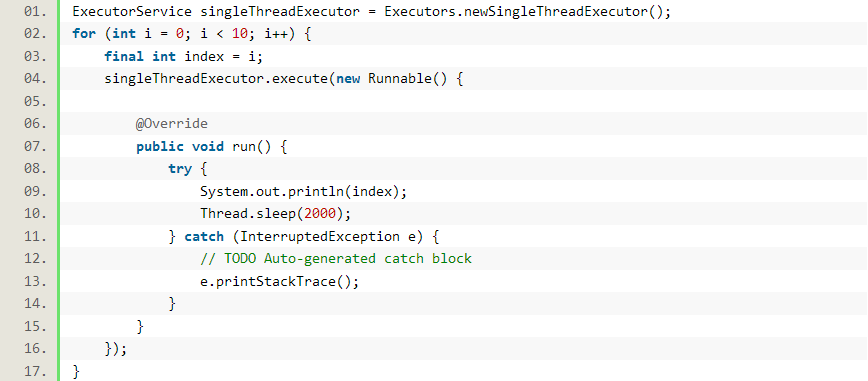


表示延迟3秒执行



表示延迟3秒后每5秒执行一次

newSingleThreadExecutor ：



结果依次输出，相当于顺序执行各个任务