1、在main方法中使用了线程池，任务跑完了，main方法还是不能结束，原因是线程池没有关闭，阻碍了main方法结束。

解决办法：pool.shutdownNow(); 这句话可以写在finally块中，防止程序异常，而不能执行到。

ThreadPoolExecutor pool = new ThreadPoolExecutor(10,

10, 1000, TimeUnit.MILLISECONDS, new LinkedBlockingQueue<>(),

new DefaultThreadFactory("product-pool"), new ThreadPoolExecutor.DiscardOldestPolicy());

ThreadPoolExecutor threadPoolExecutor = new ThreadPoolExecutor(1000,  
 1000, 60, TimeUnit.*SECONDS*, new LinkedBlockingQueue<>(),  
 new NamedThreadFactory("test-rp-"), new ThreadPoolExecutor.AbortPolicy());

这里的DefaultThreadFactory 和 NamedThreadFactory不是java中的默认实现。NamedThreadFactory在lucene-core包中

扩展：Timer, 线程池, 线程操作中的sleep,wait,suspend等方法都能阻止我们的线程结束

线程池使用ThreadPoolExecutor而不是用FixedThreadPool,SingleThreadPool,CachedThreadPool, ScheduledThreadPool。

摘自阿里巴巴开发手册：

　　　　　　【强制】线程池不允许使用 Executors 去创建，而是通过 ThreadPoolExecutor 的方式，这样 的处理方式让写的同学更加明确线程池的运行规则，规避资源耗尽的风险。 说明：Executors 返回的线程池对象的弊端如下： 1）FixedThreadPool 和 SingleThreadPool: 允许的请求队列长度为 Integer.MAX\_VALUE，可能会堆积大量的请求，从而导致 OOM。 2）CachedThreadPool 和 ScheduledThreadPool: 允许的创建线程数量为 Integer.MAX\_VALUE，可能会创建大量的线程，从而导致 OOM。

很好的一片文章

参考文档：<https://www.cnblogs.com/sunhaoyu/articles/6955923.html>

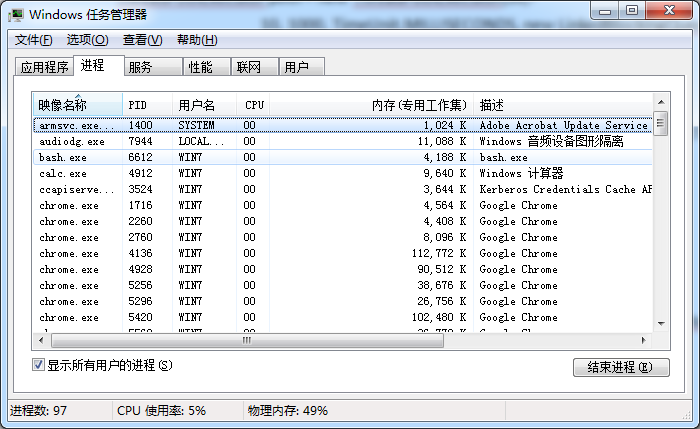
2、线程池和数据库连接池的关系

3、多线程调试、问题查看工具

jconsole

jstack：jstack -l 10115 >> jstackFile.text

说明：windows环境在cmd输入上面命令，可以将线程信息输出到文件jstackFile.text。其中10115是进程id，就是任务管理器中的pid



jstack命令用法

可以在命令行输入jstack，会有提示

待解决问题：mysql并发插入的时候，执行一会儿，程序就卡住了，没报错，也没再往数据库写数据。线程全都time\_wait——》wait，druid连接池，mysql的连接都空闲了（show PROCESSLIST）。

添加任务到线程池   
通过 execute(Runnable)方法被添加到线程池，任务就是一个 Runnable类型的对象，任务的执行方法就是 Runnable类型对象的run()方法。   
当一个任务通过execute(Runnable)方法欲添加到线程池时：   
如果此时线程池中的数量小于corePoolSize，即使线程池中的线程都处于空闲状态，也要创建新的线程来处理被添加的任务。   
如果此时线程池中的数量等于 corePoolSize，但是缓冲队列 workQueue未满，那么任务被放入缓冲队列。   
如果此时线程池中的数量大于corePoolSize，缓冲队列workQueue满，并且线程池中的数量小于maximumPoolSize，建新的线程来处理被添加的任务。   
如果此时线程池中的数量大于corePoolSize，缓冲队列workQueue满，并且线程池中的数量等于maximumPoolSize，那么通过 handler所指定的策略来处理此任务。   
也就是：处理任务的优先级为：   
核心线程corePoolSize、任务队列workQueue、最大线程maximumPoolSize，如果三者都满了，使用handler处理被拒绝的任务。   
当线程池中的线程数量大于 corePoolSize时，如果某线程空闲时间超过keepAliveTime，线程将被终止。这样，线程池可以动态的调整池中的线程数。   
unit可选的参数为java.util.concurrent.TimeUnit中的几个静态属性：NANOSECONDS、MICROSECONDS、MILLISECONDS、SECONDS。   
workQueue常用的是：java.util.concurrent.ArrayBlockingQueue   
handler有四个选择：   
ThreadPoolExecutor.AbortPolicy()： 抛出java.util.concurrent.RejectedExecutionException异常   
ThreadPoolExecutor.CallerRunsPolicy(): 重试添加当前的任务，他会自动重复调用execute()方法   
ThreadPoolExecutor.DiscardOldestPolicy(): 抛弃旧的任务   
ThreadPoolExecutor.DiscardPolicy(): 抛弃当前的任务