线程池：

使用线程池的好处：

- 线程是稀缺资源，不能频繁的创建
- 将其放入一个池子中可以给其他任务进行复用
- 解耦作用，线程创建与执行完全分开，方便维护

1.任务提交给线程池后的处理策略：

- 如果当前线程池中的线程数目小于corePoolSize，则创建一个线程去执行任务；
- 如果当前线程池中的线程数目>=corePoolSize，则将任务添加到BlockingQueue中
- 如果队列已满，则创建新的线程来处理任务
- 如果创建线程将使当前线程超出maximumPoolSize,则会采取任务拒绝策略进行处理

java.uitl.concurrent.ThreadPoolExecutor类是线程池中最核心的一个类
corePoolSize：核心池的大小
maximumPoolSize：线程池最大线程数， 表示在线程池中最多能创建多少个线程（更像是一种补救措施）

2.ThreadPoolExecutor中execute()方法和submit()方法

execute()用于提交不需要返回值的任务，submit()用于提交需要返回值的任务，返回future类型对象,通过future对象可以获取任务执行结果

3.任务拒绝策略：

- AbortPolicy ： 直接抛出 RejectedExecutionException
- DiscardPolicy :不处理，丢弃掉
- DiscardOldestPolicy ： 丢弃队列中最近的一个任务，并执行当前任务
- CallerRunsPolicy :只用调用者所在线程来运行任务

4.关闭线程池

shutdown:不会立即终止线程池，而是要等所有任务缓存队列中的任务都执行完后才终止，但再也不会接受新的任务

shutdownNow:立即终止线程池，并尝试打断正在执行的任务，并且清空任务缓存队列，返回尚未执行的任务

5.合理地配置线程池

CPU密集型任务：配置尽可能小的线程池，如配置N+1个线程

IO密集型任务：配置尽可能多的线程，如2*N

6.线程池的几种方式

Java 5+中的Executor接口定义一个执行线程的工具。它的子类型即线程池接口是ExecutorService。要配置一个线程池是比较复杂的，尤其是对于线程池的原理不是很清楚的情况下，因此在工具类Executors面提供了一些静态工厂方法，生成一些常用的线程池，如下所示：

- newSingleThreadExecutor：创建一个单线程的线程池。这个线程池只有一个线程在工作，也就是相当于单线程串行执行所有任务。如果这个唯一的线程因为异常结束，那么会有一个新的线程来替代它。此线程池保证所有任务的执行顺序按照任务的提交顺序执行。
- newFixedThreadPool：创建固定大小的线程池。每次提交一个任务就创建一个线程，直到线程达到线程池的最大大小。线程池的大小一旦达到最大值就会保持不变，如果某个线程因为执行异常而结束，那么线程池会补充一个新线程。
- newCachedThreadPool：创建一个可缓存的线程池。如果线程池的大小超过了处理任务所需要的线程，那么就会回收部分空闲（60秒不执行任务）的线程，当任务数增加时，此线程池又可以智能的添加新线程来处理任务。此线程池不会对线程池大小做限制，线程池大小完全依赖于操作系统（或者说JVM）能够创建的最大线程大小。
- newScheduledThreadPool：创建一个大小无限的线程池。此线程池支持定时以及周期性执行任务的需求。