为什么需要线程池？如果并发的线程数量太多，并且每个线程都是执行一个很短的任务就结束，频繁创建线程会降低系统的效率，频繁创建线程和销毁线程需要时间。

java.uitl.concurrent.ThreadPoolExecutor 类是线程池中最核心的一个类，在ThreadPoolExecutor类中有四个构造方法。ThreadPoolExecutor继承了AbstractExecutorService类，并提供了四个构造器，前面三个构造器都是调用的第四个构造器进行的初始化工作。

构造器中的七个参数：（1）corePoolSize，核心池的大小，在创建了线程池后，默认情况下，线程池中并没有任何线程，而是等有任务了才创建线程去执行任务。除非调用了prestartAllCoreThreads()或者 prestartCoreThread()方法，即在没有任务到来之前就创建corePoolSize 个线程或者一个线程。默认情况下，在创建了线程池后，线程池中的线程数为0，当有任务来之后，就会创建一个线程去执行任务，当线程池中的线程数目达到 corePoolSize后，就会把到达的任务放到缓存队列当中。（2）maximumPoolSize：线程池最大线程数，它表示在线程池中最多能创建多少个线程。（3）keepAliveTime：表示线程没有任务执行时最多保持多久时间会终止。当线程池中的线程数大于 corePoolSize 时，如果一个线程空闲的时间达到 keepAliveTime，则会终止，直到线程池中的线程数不超过 corePoolSize。但是如果调用了 allowCoreThreadTimeOut(boolean)方法，在线程池中的线程数不大于 corePoolSize 时，keepAliveTime 参数也会起作用，直到线程池中的线程数为 0。（4）unit：参数 keepAliveTime 的时间单位，有 7 种取值，在 TimeUnit 类中有 7 种静态属性。（5）workQueue：一个阻塞队列，用来存储等待执行的任务。ArrayBlockingQueue 和 PriorityBlockingQueue 使用较少，一般使用 LinkedBlockingQueue 和 Synchronous。线程池的排队策略与 BlockingQueue 有关。（6）threadFactory：线程工厂，主要用来创建线程。（7）handler，表示当拒绝处理任务时的策略。ThreadPoolExecutor.AbortPolicy:丢弃任务并抛出 RejectedExecutionException 异常。ThreadPoolExecutor.DiscardPolicy ： 也是丢弃任务，但是不抛出异常。ThreadPoolExecutor.DiscardOldestPolicy：丢弃队列最前面的任务，然后重新尝试执行任务（重复此过程）ThreadPoolExecutor.CallerRunsPolicy：由调用线程处理该任务。

合理配置线程池大小：（1）如果是 CPU 密集型任务，需要尽量压榨 CPU，参考值可以设置为 $N_{CPU}+1$。（2）如果是 IO 密集型任务，参考值可以设置为 $2*N_{CPU}$。