public class ThreadPoolExecutor extends AbstractExecutorService

AbstractExecutorService是一个实现了ExecutorService接口中部分方法的抽象类，ExecutorService接口是Executor接口的拓展。

# 接口说明

An [ExecutorService](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ExecutorService.html) that executes each submitted task using one of possibly several pooled threads, normally configured using [Executors](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html) factory methods.

ThreadPoolExecutor实现了ExecutorService接口，使用线程池中的任一线程执行提交的任务，通常使用Executors类中的工厂方法配置并获取ThreadPoolExecutor类的实例。

Thread pools address two different problems: they usually provide improved performance when executing large numbers of asynchronous tasks, due to reduced per-task invocation overhead, and they provide a means of bounding and managing the resources, including threads, consumed when executing a collection of tasks. Each ThreadPoolExecutor also maintains some basic statistics, such as the number of completed tasks.

ThreadPoolExecutor的主要作用：

1. 当需要执行大量异步任务时，通过使用ThreadPoolExecutor能够减少每个任务的调用开销，从而提高性能。
2. 提供了一种方法来约束和管理执行任务集合时消耗的资源（包括线程）。
3. 维护一些基本统计信息，例如已完成任务的数量。

To be useful across a wide range of contexts, this class provides many adjustable parameters and extensibility hooks. However, programmers are urged to use the more convenient [Executors](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html) factory methods [Executors.newCachedThreadPool()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool--) (unbounded thread pool, with automatic thread reclamation), [Executors.newFixedThreadPool(int)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newFixedThreadPool-int-) (fixed size thread pool) and [Executors.newSingleThreadExecutor()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newSingleThreadExecutor--) (single background thread), that preconfigure settings for the most common usage scenarios. Otherwise, use the following guide when manually configuring and tuning this class:

为了支持更多的使用场景，类中提供了很多可以调整的参数和可以拓展的钩子。但是鼓励程序员通过一种更简单的方式使用线程池，即通过Executors类提供的一些工厂方法，如：

[Executors.newCachedThreadPool()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool--) – 线程池大小无限制，自动回收线程。

[Executors.newFixedThreadPool(int)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newFixedThreadPool-int-) – 线程池大小固定

[Executors.newSingleThreadExecutor()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newSingleThreadExecutor--) – 线程池中只有一个线程

这些方法针对一些常用配置做了预先配置。如果你打算自己手动配置和调整ThreadPoolExecutor类是，请阅读以下注意事项：

**Core and maximum pool sizes**

A ThreadPoolExecutor will automatically adjust the pool size (see [getPoolSize()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#getPoolSize--)) according to the bounds set by corePoolSize (see [getCorePoolSize()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#getCorePoolSize--)) and maximumPoolSize (see [getMaximumPoolSize()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#getMaximumPoolSize--)). When a new task is submitted in method [execute(Runnable)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#execute-java.lang.Runnable-), and fewer than corePoolSize threads are running, a new thread is created to handle the request, even if other worker threads are idle. If there are more than corePoolSize but less than maximumPoolSize threads running, a new thread will be created only if the queue is full. By setting corePoolSize and maximumPoolSize the same, you create a fixed-size thread pool. By setting maximumPoolSize to an essentially unbounded value such as Integer.MAX\_VALUE, you allow the pool to accommodate an arbitrary number of concurrent tasks. Most typically, core and maximum pool sizes are set only upon construction, but they may also be changed dynamically using [setCorePoolSize(int)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#setCorePoolSize-int-) and [setMaximumPoolSize(int)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#setMaximumPoolSize-int-).