L02-线程池-JUC线程池API

JAVA并发编程

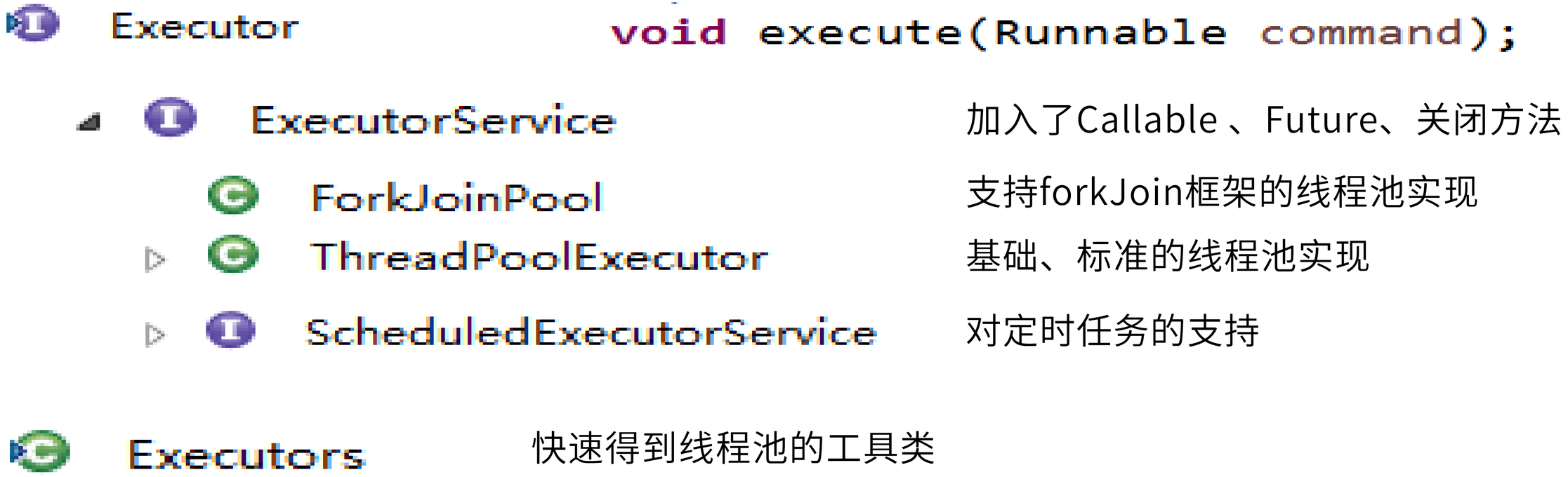
学习目标

- 掌握JUC线程池API
- 掌握ThreadPoolExecutor实现原理
- 掌握Executors的应用
- 掌握ForkJoin框架的使用

JAVA并发包中的线程池API详解

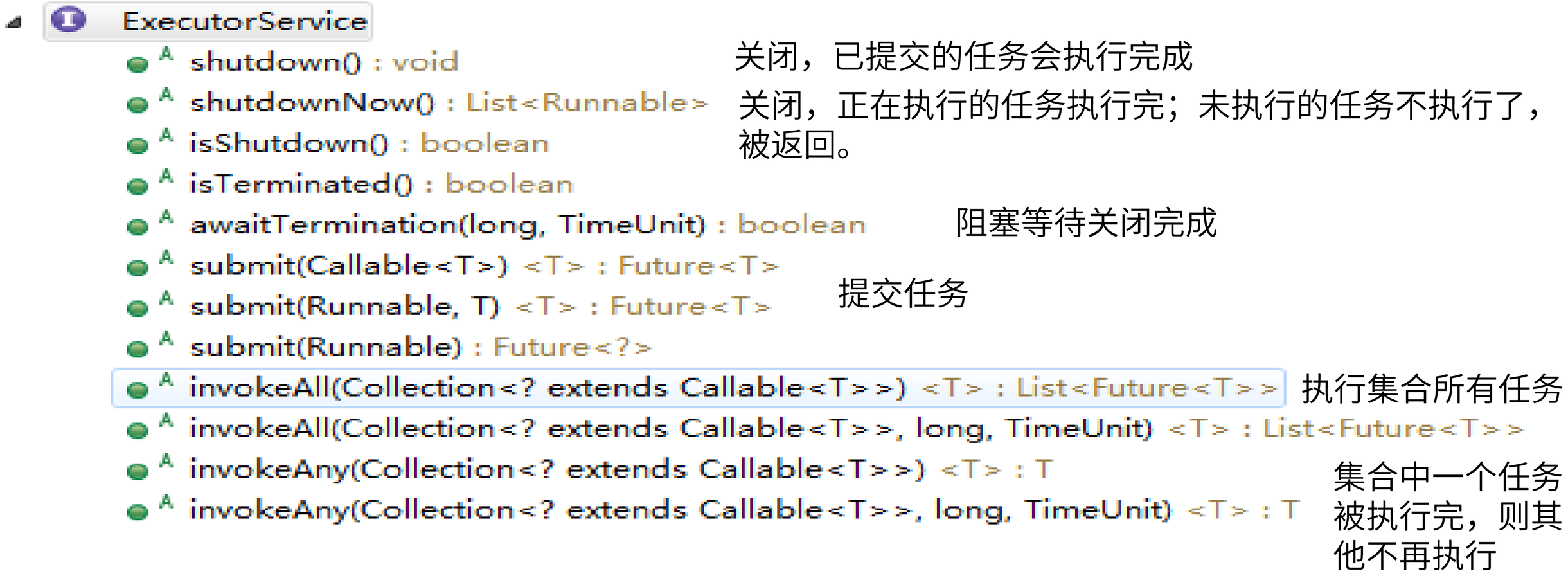
Java线程池API学习

- Java 并发包中提供了丰富的线程池实现！



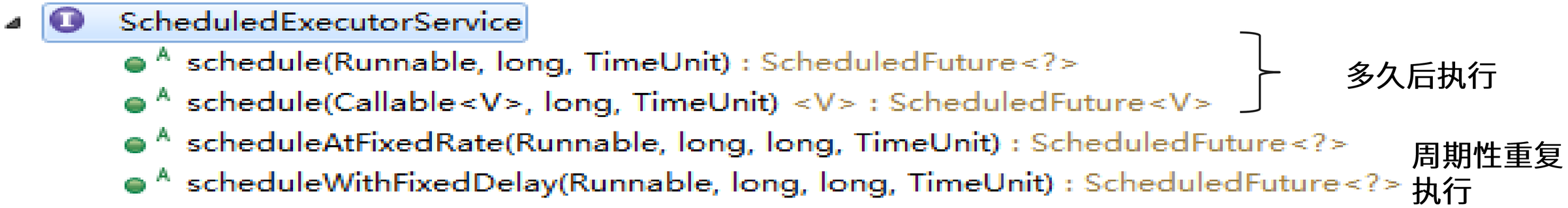
Java线程池API学习

- ExecutorService 加入关闭方法 和对 Callable、Future的支持。



Java线程池API学习

- ScheduleExecutorService 加入对定时任务的支持。



public ScheduledFuture<?> **scheduleAtFixedRate**(Runnable command,
long initialDelay,
long period,
TimeUnit unit);

以固定的时间频率重复执行任务，如每5分钟。
注：当任务本身的执行耗时超过时间频率时，下次执行需等待该次执行完后才能开启，不能并发执行。

public ScheduledFuture<?> **scheduleWithFixedDelay**(Runnable command,
long initialDelay,
long delay,
TimeUnit unit);

以固定的任务间延迟来重复执行任务。下一次执行是在上一次执行完后间隔时延时间再执行。

Java线程池API学习

- ThreadPoolExecutor 线程池标准实现

```
public ThreadPoolExecutor(int corePoolSize,
                          int maximumPoolSize,
                          long keepAliveTime,
                          TimeUnit unit,
                          BlockingQueue<Runnable> workQueue)
```

```
public ThreadPoolExecutor(int corePoolSize,
                          int maximumPoolSize,
                          long keepAliveTime,
                          TimeUnit unit,
                          BlockingQueue<Runnable> workQueue,
                          RejectedExecutionHandler handler)
```

```
public ThreadPoolExecutor(int corePoolSize,
                          int maximumPoolSize,
                          long keepAliveTime,
                          TimeUnit unit,
                          BlockingQueue<Runnable> workQueue,
                          ThreadFactory threadFactory)
```

```
public ThreadPoolExecutor(int corePoolSize,
                          int maximumPoolSize,
                          long keepAliveTime,
                          TimeUnit unit,
                          BlockingQueue<Runnable> workQueue,
                          ThreadFactory threadFactory,
                          RejectedExecutionHandler handler)
```

Java线程池API学习

■ Executors 创建线程池的工厂类，它的工厂方法：

➢ newFixedThreadPool(int nThreads) 创建一个固定大小、任务队列容量无界的线程池。池的核心线程数=最大线程数=nThreads

➢ newCachedThreadPool() 创建的是一个大小无界的缓冲线程池。它的任务队列是一个同步队列。任务加入到池中，如果池中有空闲线程，则用空闲线程执行，如无则创建新线程执行。池中的线程空闲超过60秒，将被销毁释放。池中的线程数随任务的多少变化。缓冲线程池适用于执行耗时较小的异步任务。池的核心线程数=0 最大线程数= Integer.MAX_VALUE

➢ newSingleThreadExecutor() 只有一个线程来执行无界任务队列的单一线程池。该线程池确保任务按加入的顺序一个一个依次执行，任何时刻只有一个任务在执行。当唯一的线程因任务异常中止时，将创建一个新的线程来继续执行后续的任务。单一线程池与newFixedThreadPool(1)的区别在于，单一线程池的池大小是不能再改变的。

Java线程池API学习

Executors 的其他工厂方法

- newScheduledThreadPool(int corePoolSize) 能定时执行任务的线程池。该池的核心线程数由参数指定，最大线程数= Integer.MAX_VALUE

- newWorkStealingPool() 以当前系统可用处理器数作为并行级别创建的 work-stealing thread pool (ForkJoinPool)

- newWorkStealingPool(int parallelism) 以parallelism指定的并行级别创建的 work-stealing thread pool (ForkJoinPool)

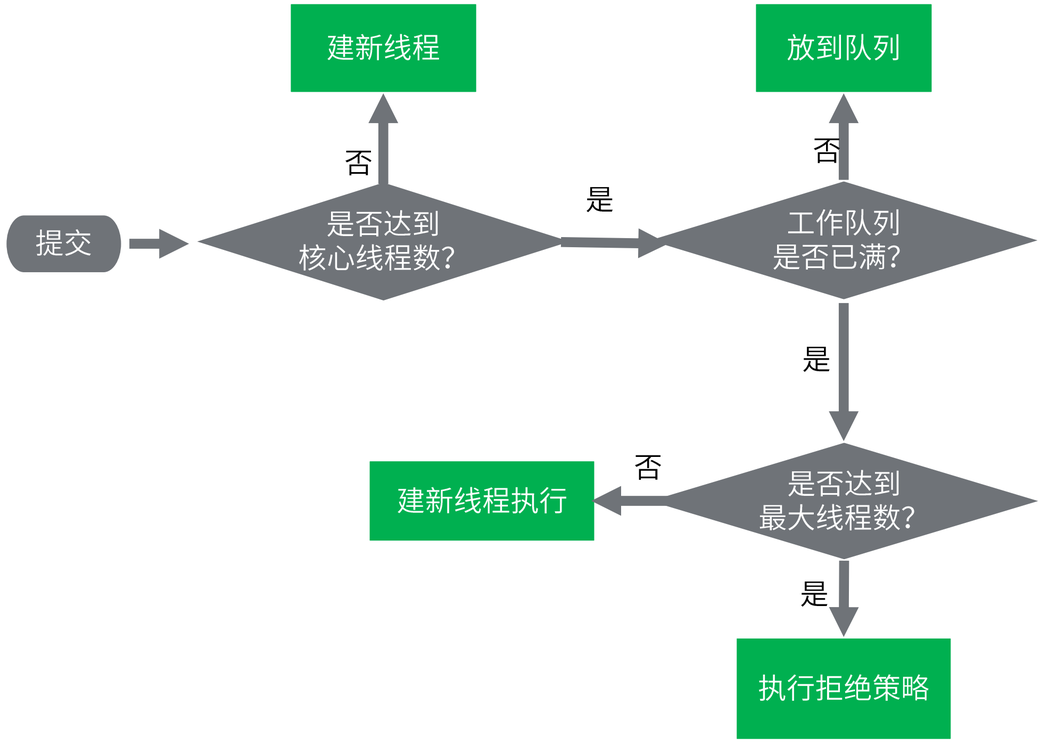
正确使用JUC线程池API

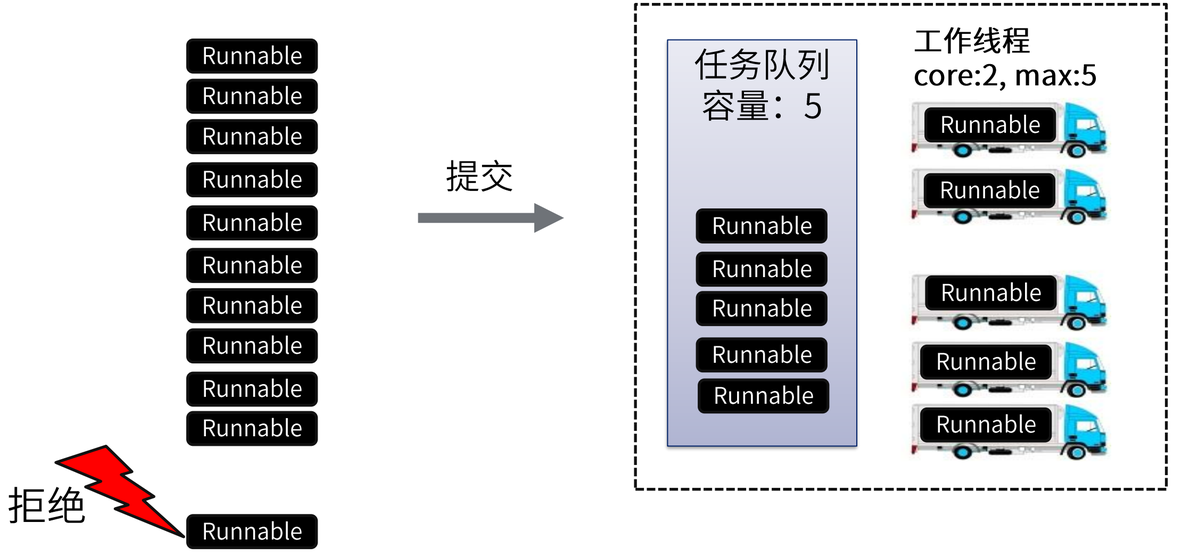
正确使用JUC线程池API

- ThreadPoolExecutor 的 coreSize maxSize queueSize
- ThreadPoolExecutor 的 shutdown
- 慎用Executors

ThreadPoolExecutor-原理-参数

```
new ThreadPoolExecutor(
        2,          //核心线程数
        5,          //最大线程数
        5,          //keepAliveTime,超过核心线程数的线程，空闲超过keepAliveTime，这个线程就会被销毁
        TimeUnit.SECONDS,   //keepAliveTime 的时间单位
        new LinkedBlockingQueue<Runnable>(5)    //传入边界为 5 的工作队列
);
```





ForkJoin框架详解

经典面试题

- 只用 100M 内存，如何快速完成对10G文件中数据的排序

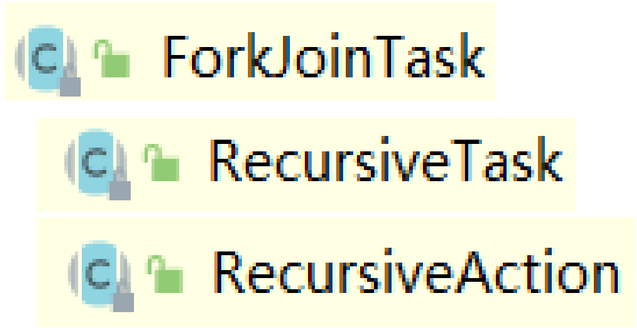
- 考察点：
- 排序算法：外排序、归并排序
- 并发编程（快速完成排序）

Forkjoin框架-先来个简单的

- 快速实现对一个长度百万的数组的排序
- 难点：多线程如何组织 来实现 归并排序

Forkjoin框架-API

- 任务表示：



- ForkJoinPool　　int parallelism　并行度（并行执行几个线程）

动手实践，下节课见