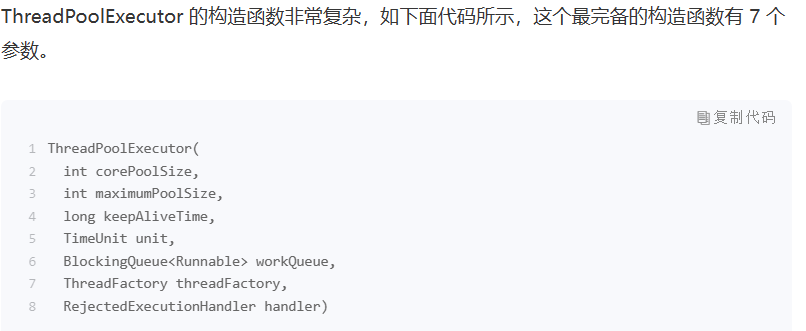
**22 | Executor与线程池：如何创建正确的线程池？**

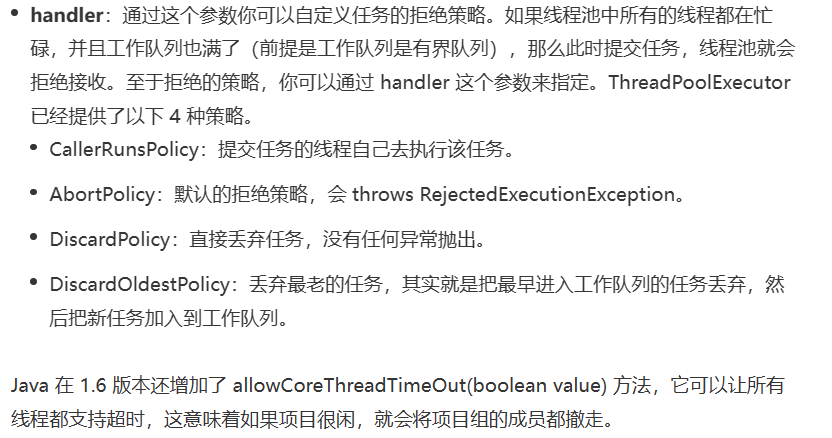
线程池的设计，没有办法直接采用一般意义上池化资源的设计方法。那线程池该如何设计呢？

目前业界线程池的设计，普遍采用的都是**生产者 - 消费者模式**。线程池的使用方是生产者，线程池本身是消费者。

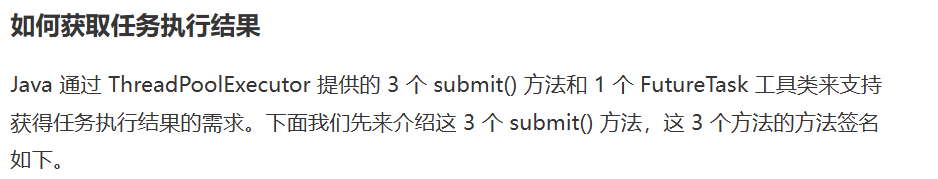
API

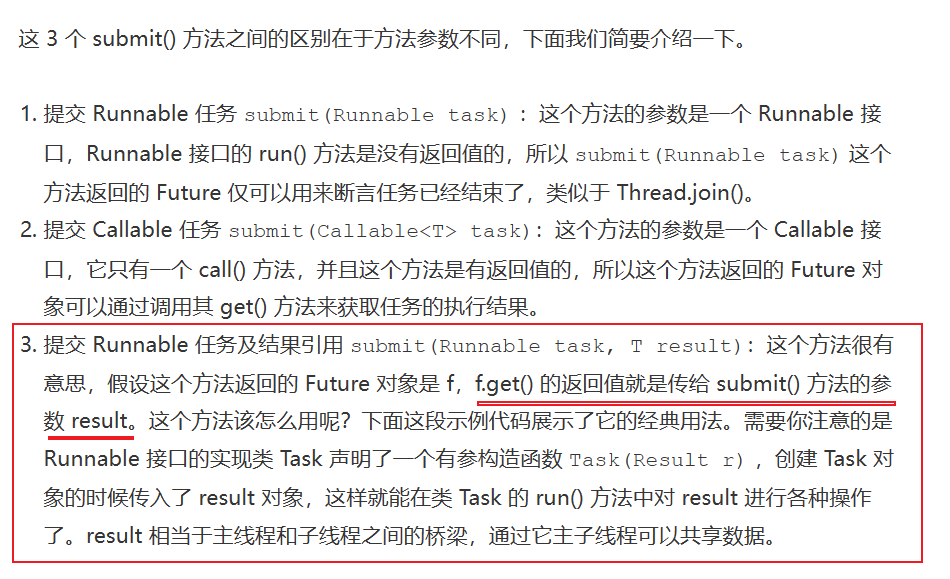


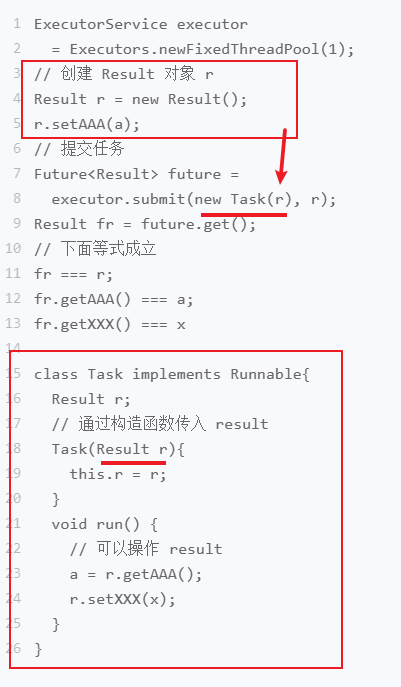




# 23 | Future：如何用多线程实现最优的“烧水泡茶”程序？





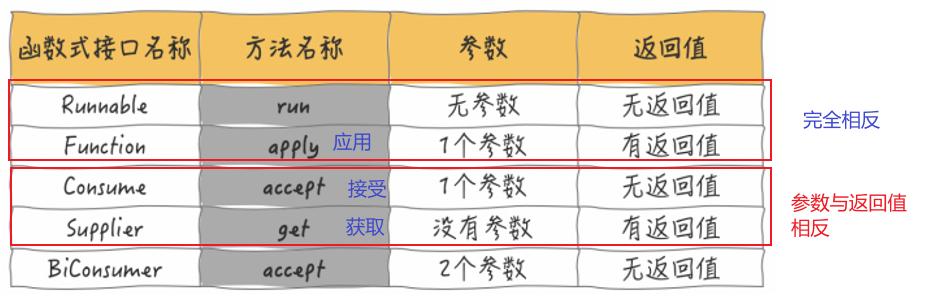


# 24 | CompletableFuture：异步编程没那么难

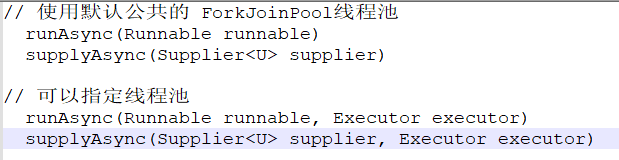
多线程优化性能，其实不过就是将串行操作变成并行操作。如果仔细观察，你还会发现在串行转换成并行的过程中，一定会涉及到异步化.

**异步通俗点来讲就是调用方是否需要等待结果，如果需要等待结果，就是同步；如果不需要等待结果，就是异步**。

## 函数式接口总结

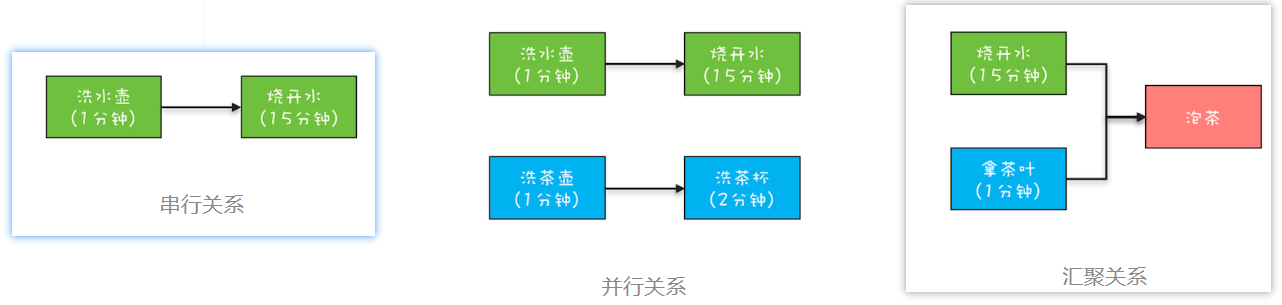


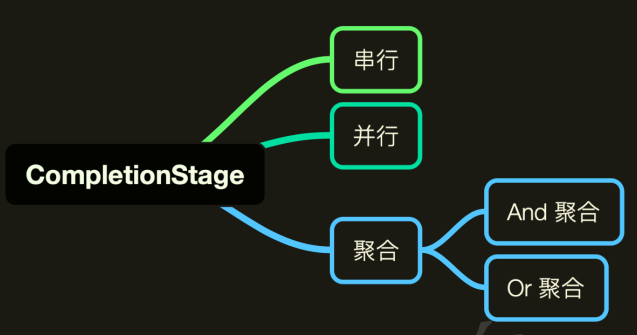
## 创建 CompletableFuture 对象



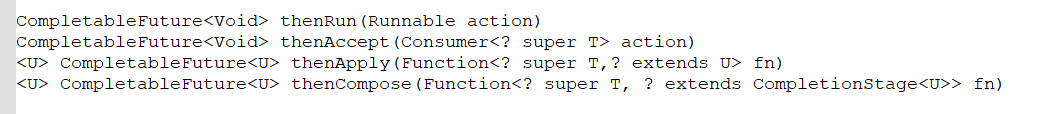
## 如何理解 CompletionStage 接口

可以站在分工的角度类比一下工作流。任务是有时序关系的，比如有**串行关系、并行关系、汇聚关系**等。



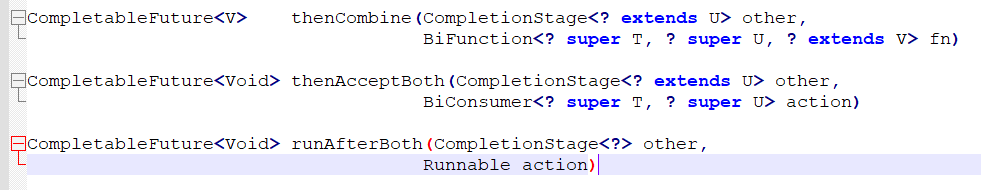


### 串行关系



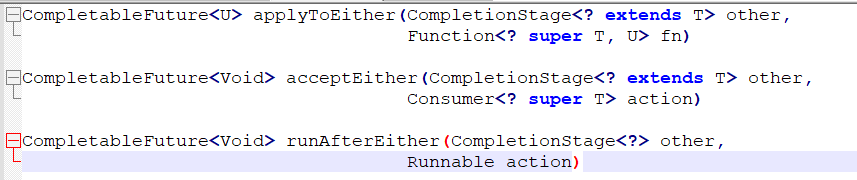
### AND 聚合关系

combine... with... 和 both...and... 都是要求两者都满⾜，也就是 and 的关系了

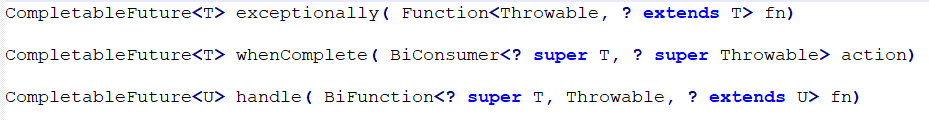


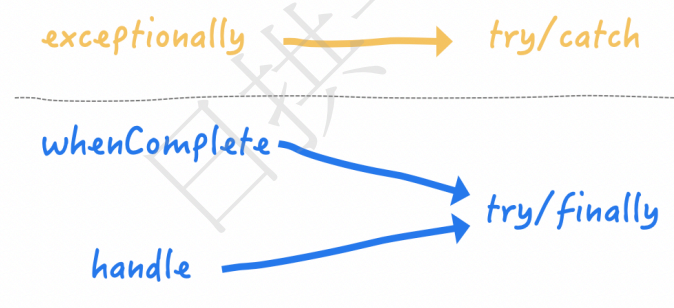
### OR 聚合关系

Either...or... 表示两者中的⼀个，⾃然也就是 Or 的体现了



### 异常处理





## get与join的区别

get：获取结果，必须try catch进行异常处理。

join: 等待线程完成，获取处理结果，不进行强制异常处理。