目录

[01\_JUC教程简介 1](#_Toc121583445)

[02\_为什么学好用多线程如此重要 2](#_Toc121583446)

[03\_start线程开启C源码分析 3](#_Toc121583447)

[Openjdk源码 3](#_Toc121583448)

[Thread.c 3](#_Toc121583449)

[Jvm.cpp 4](#_Toc121583450)

[Thread.cpp 5](#_Toc121583451)

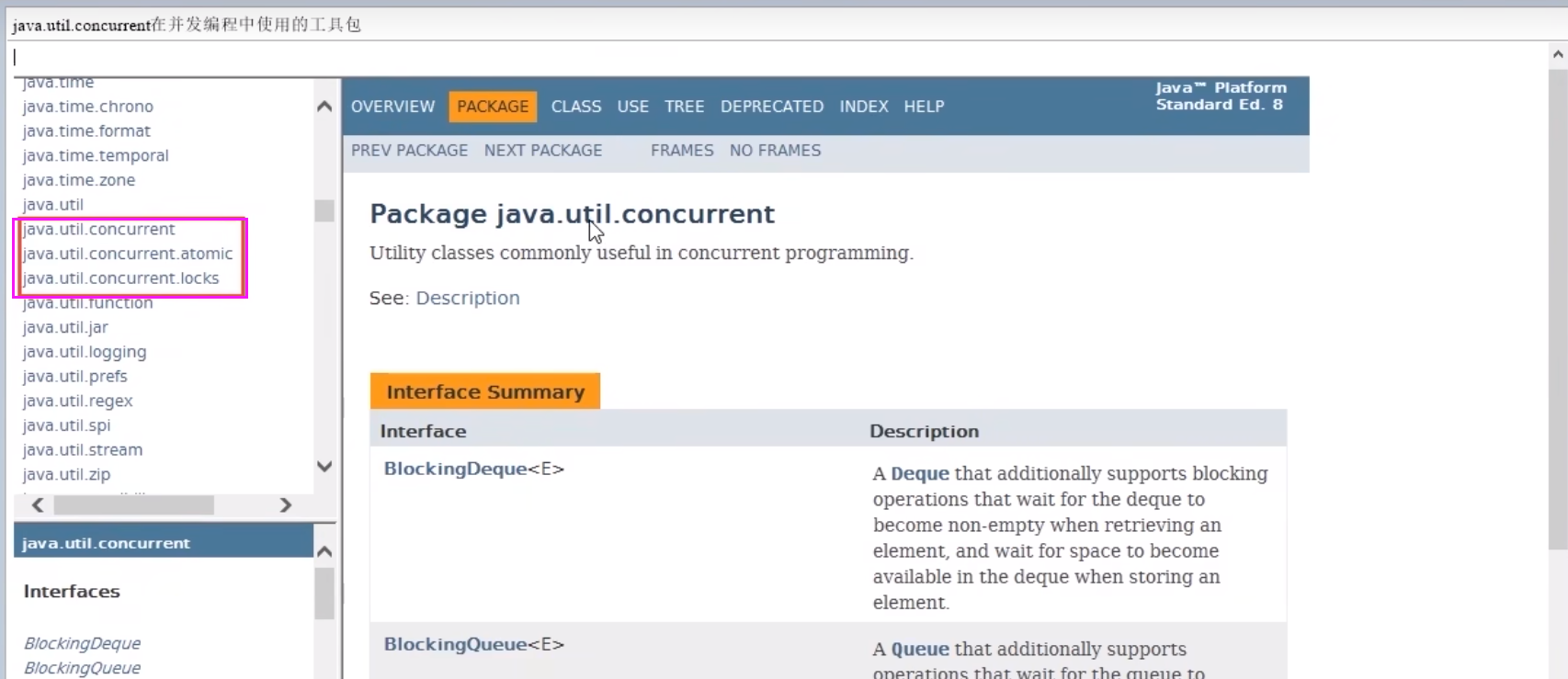
[总结： 5](#_Toc121583452)

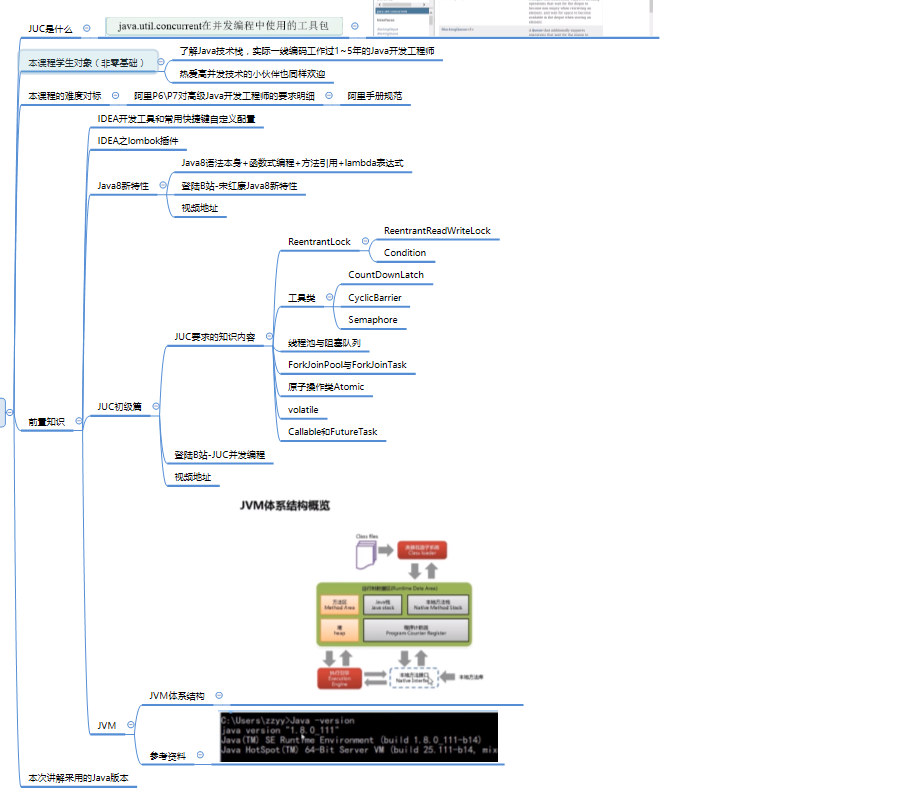
[05\_用户守护线程理论 5](#_Toc121583453)

[06\_用户守护线程代码演示和总结 5](#_Toc121583454)

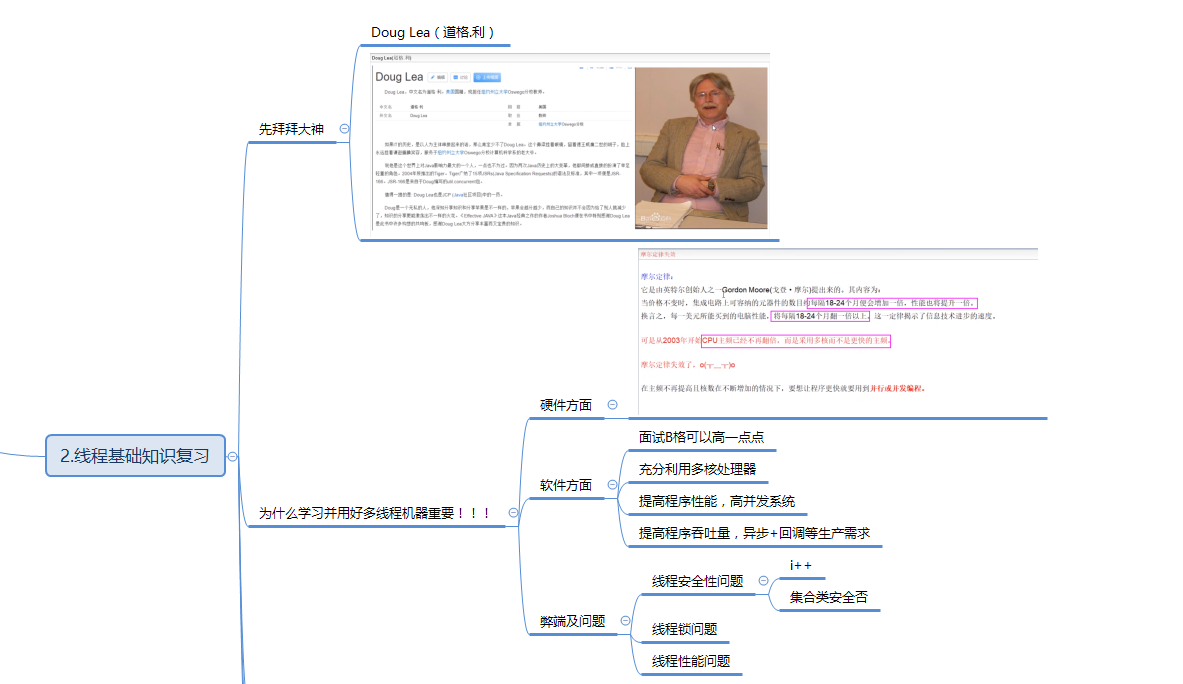
# 01\_JUC教程简介



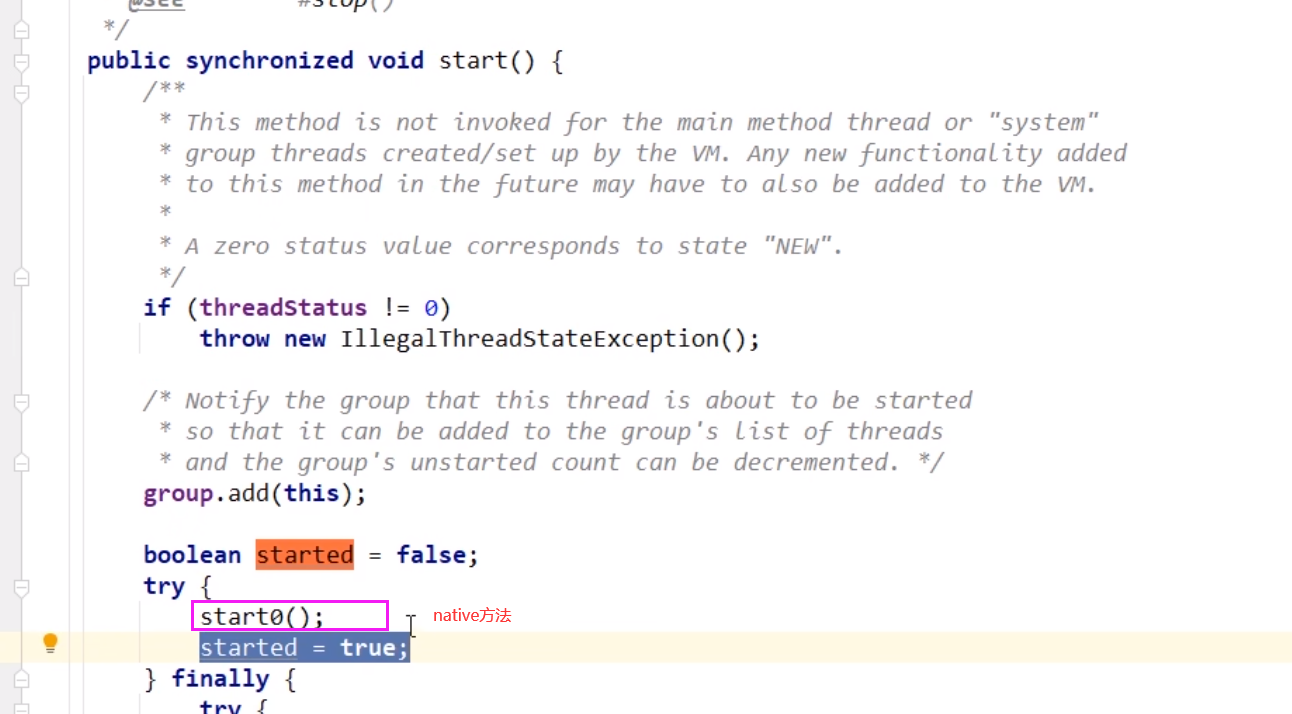




# 02\_为什么学好用多线程如此重要



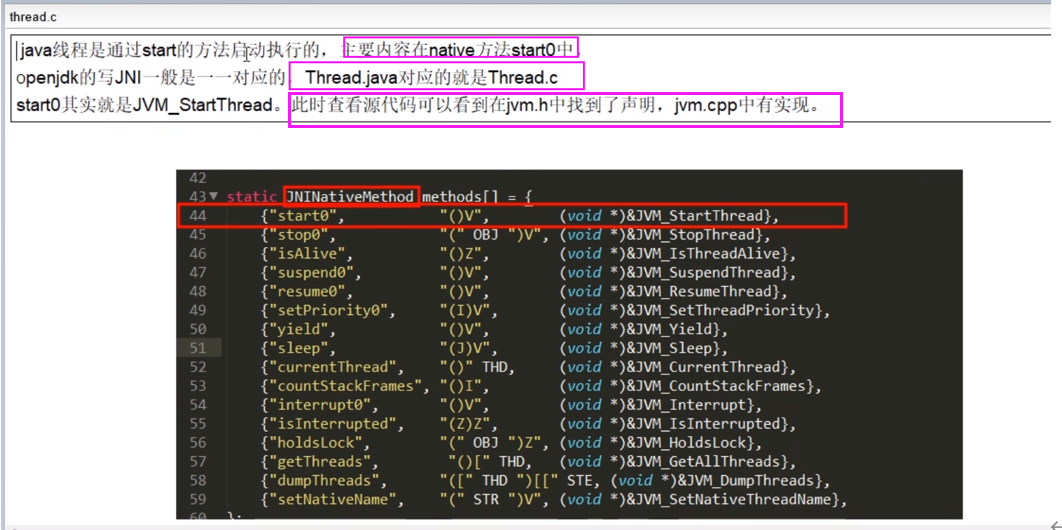
# 03\_start线程开启C源码分析



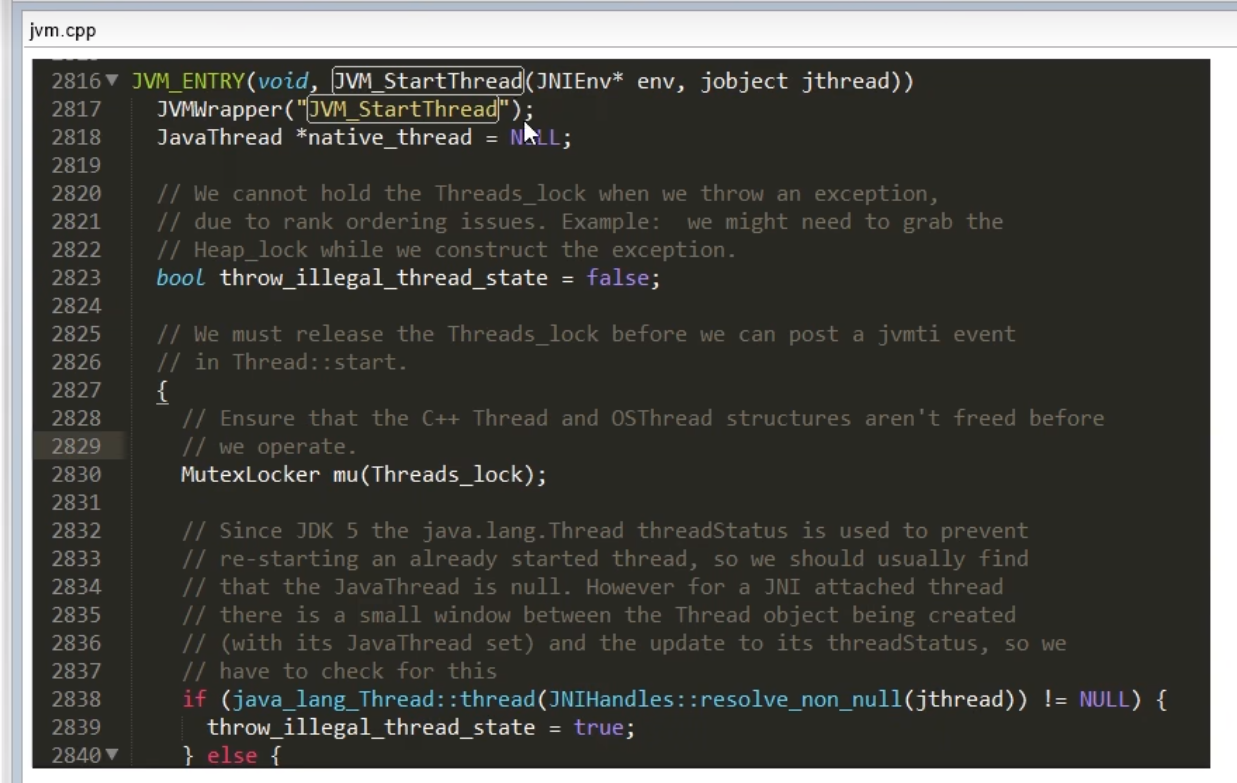


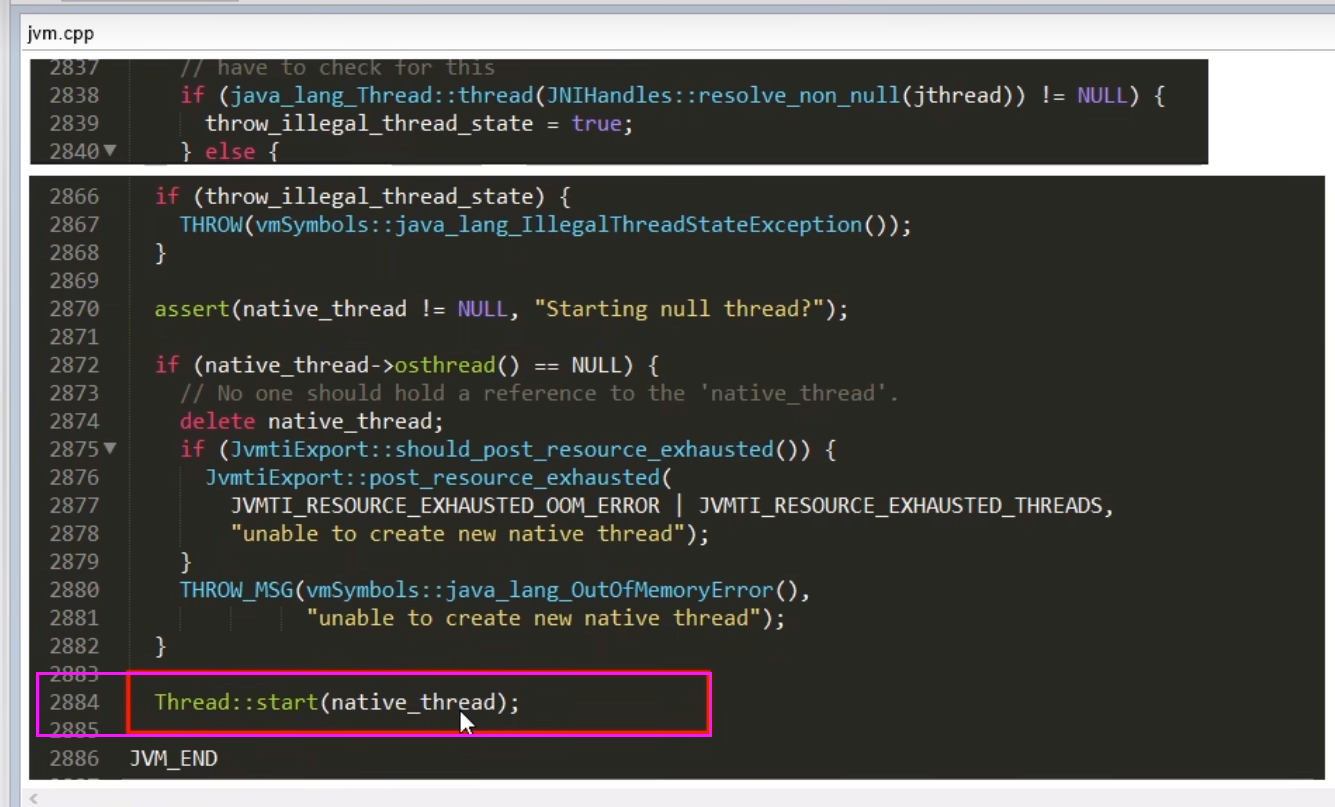
## Openjdk源码

### Thread.c

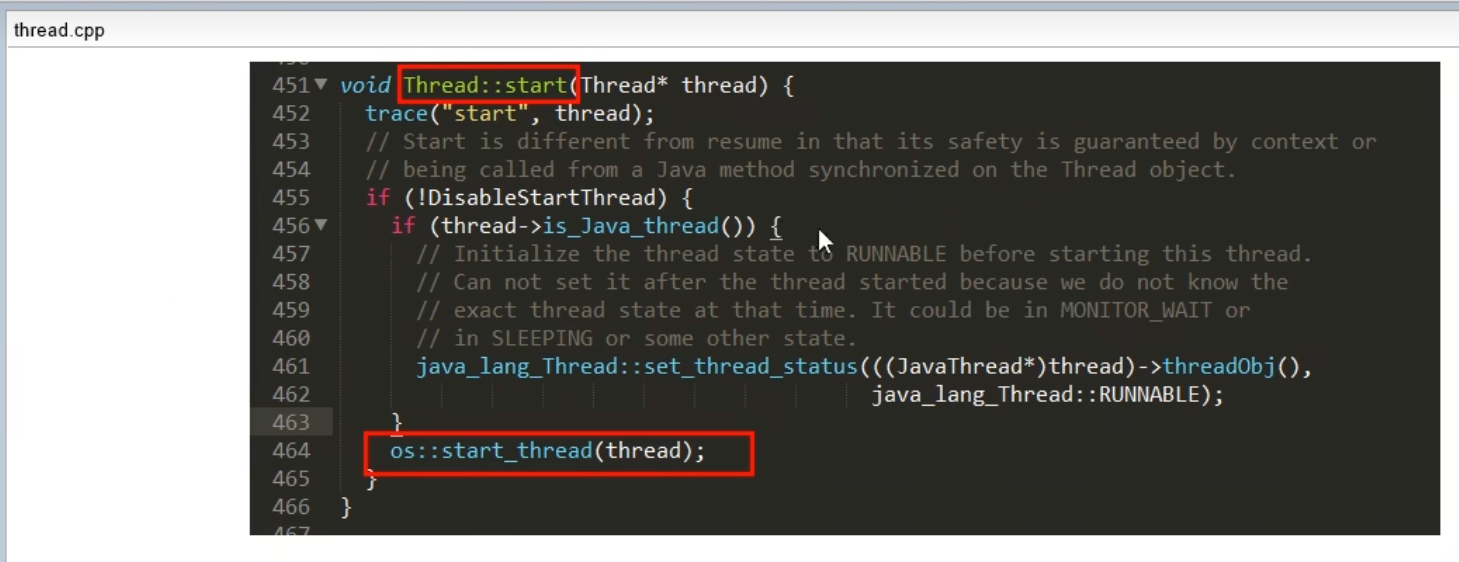


### Jvm.cpp





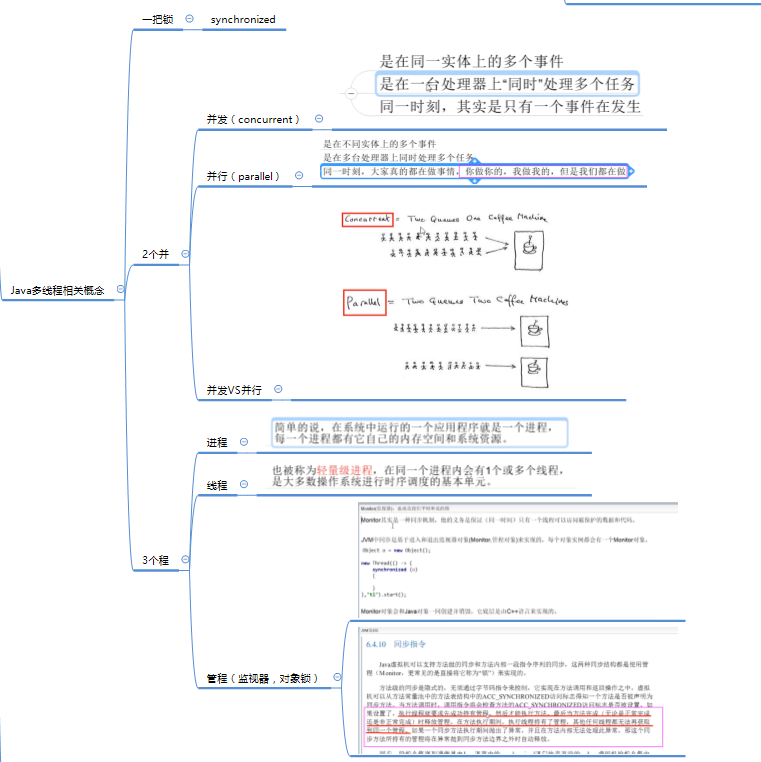
### Thread.cpp



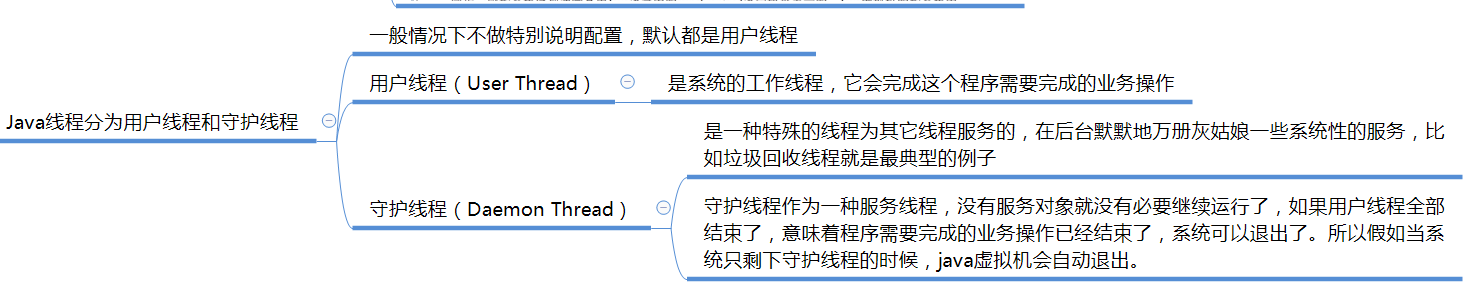
### 总结：

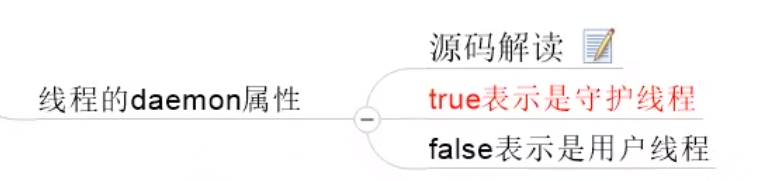
对于start0这个native方法，通过jvm配合操作系统，底层由操作系统分配的原生基础线程来帮助你来启动

# 04\_基础概念复习



# 05\_用户守护线程理论

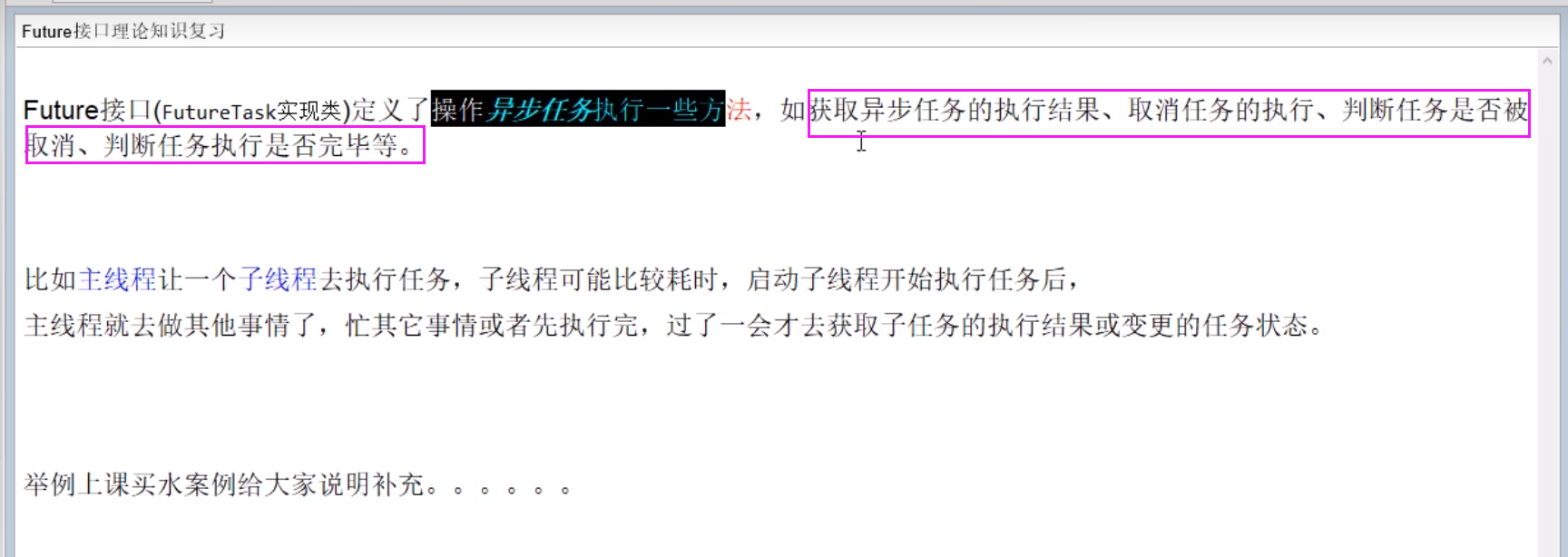




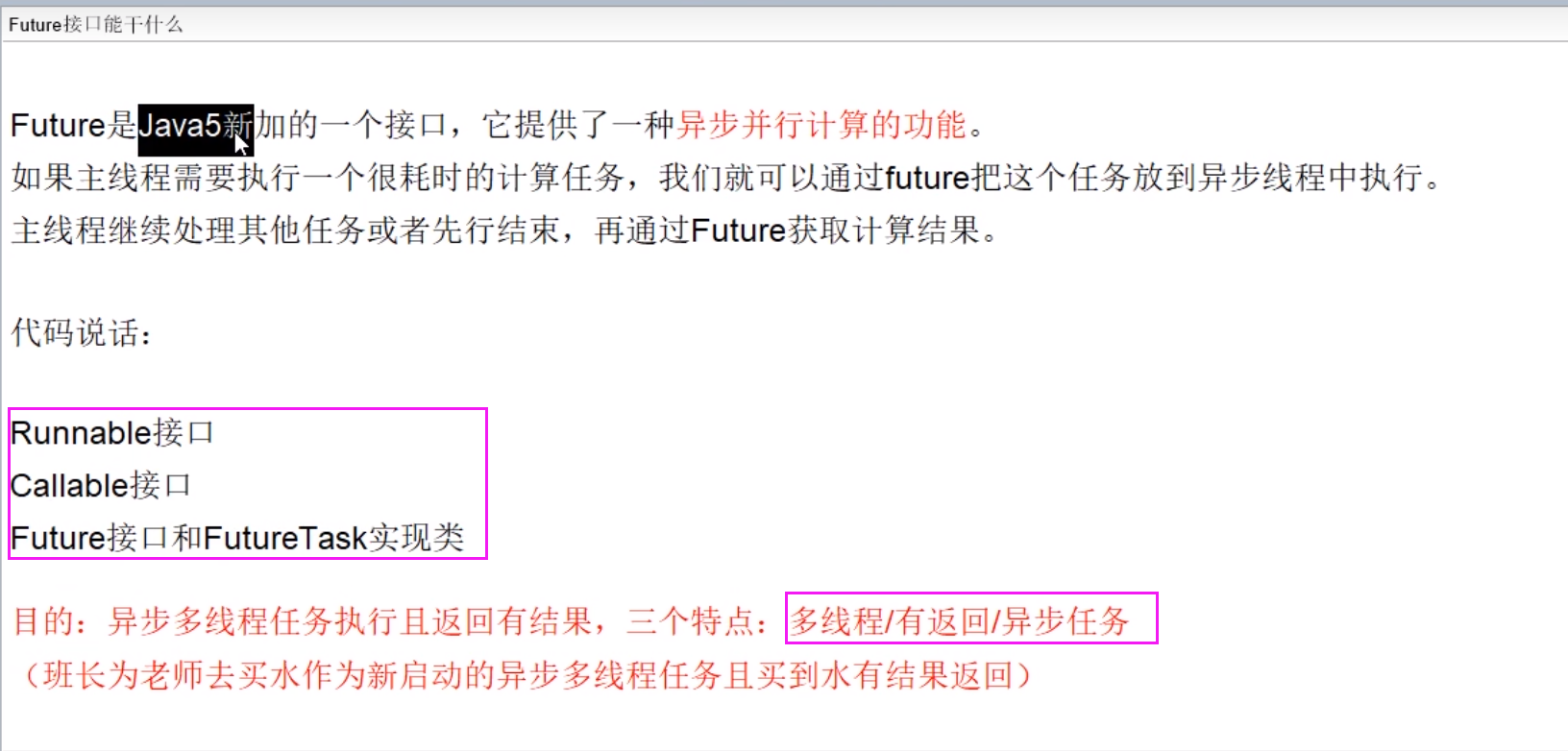
# 06\_用户守护线程代码演示和总结

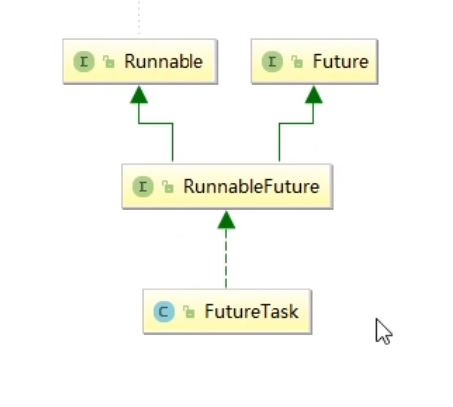


# 07CompletetableFuture之Future为什么出现

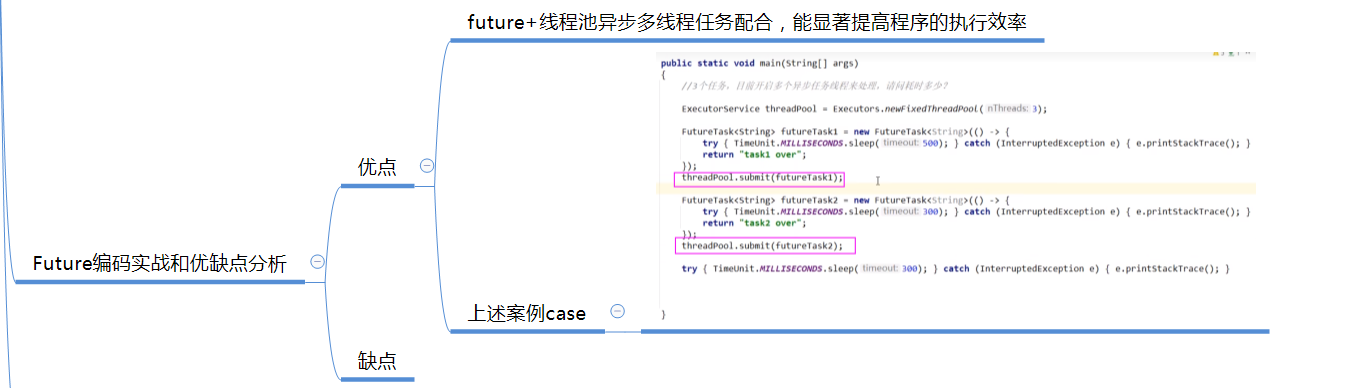


# 08/09/10CompletableFuture之引出FutureTask-上集

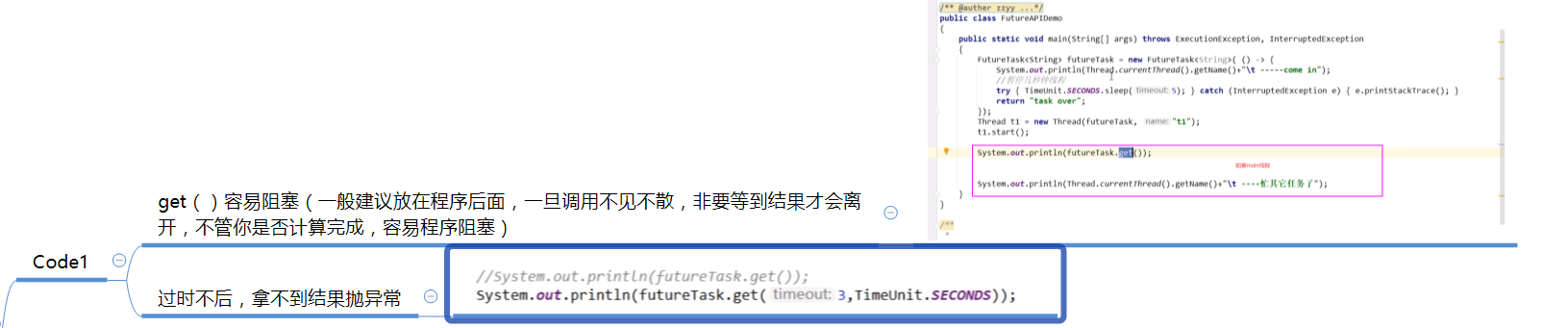




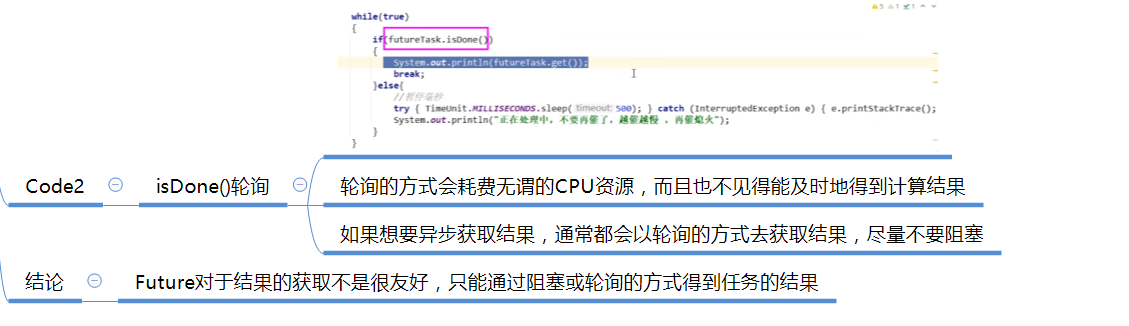
# 11CompletableFuture之FutureTask结合线程池提升性能



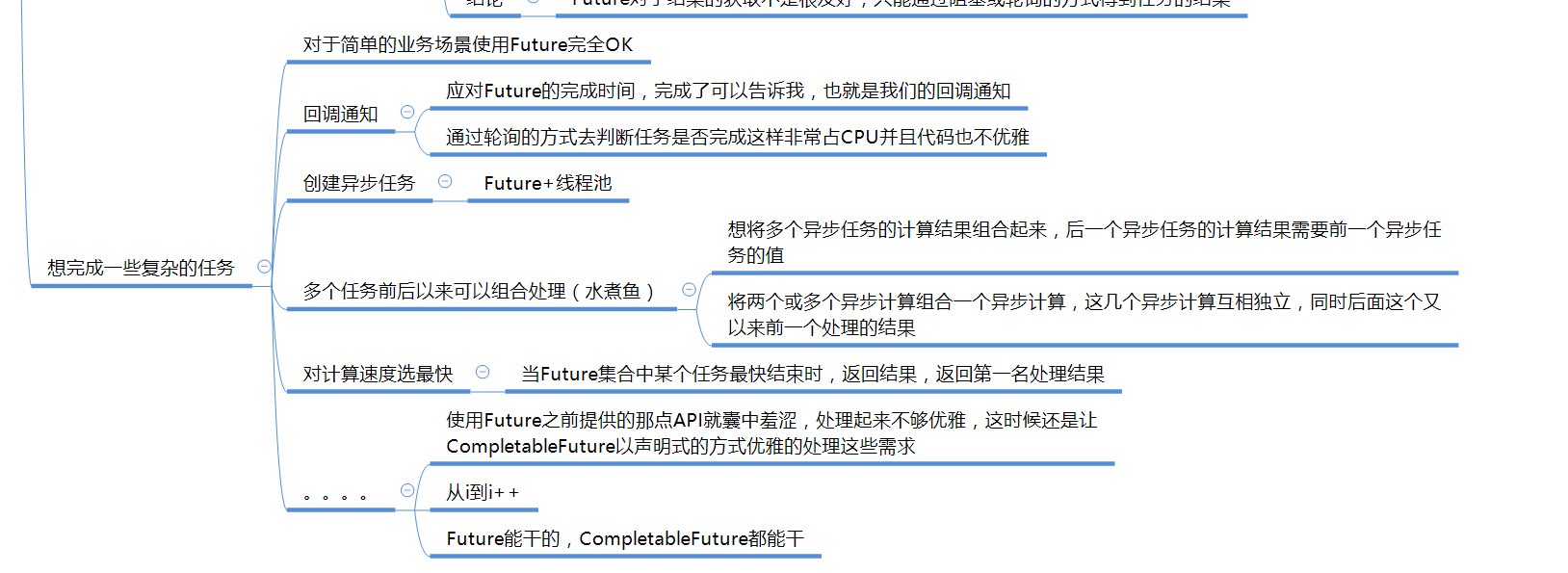
# 12CompletableFuture之get获取容易阻塞



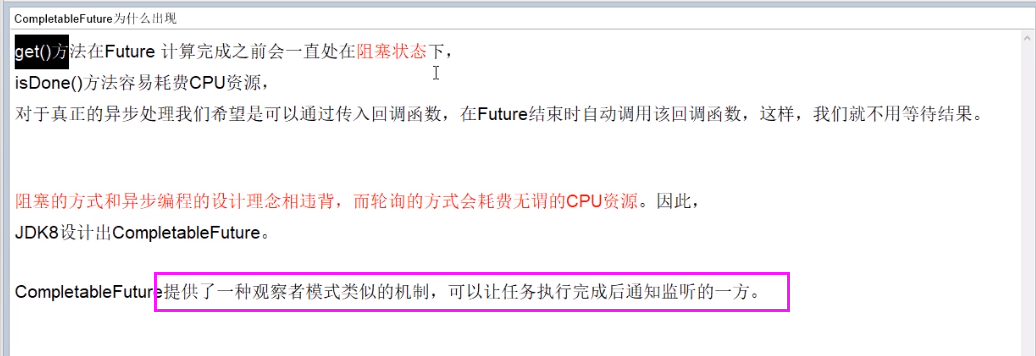
# 13CompletableFuture之轮询耗费CPU

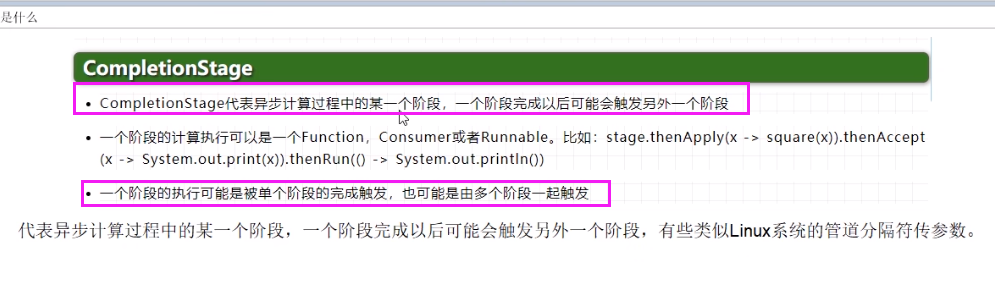


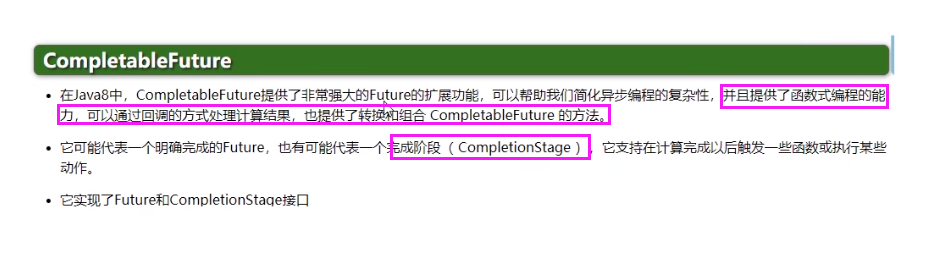
# 14.CompletableFuture之Future异步优化思路



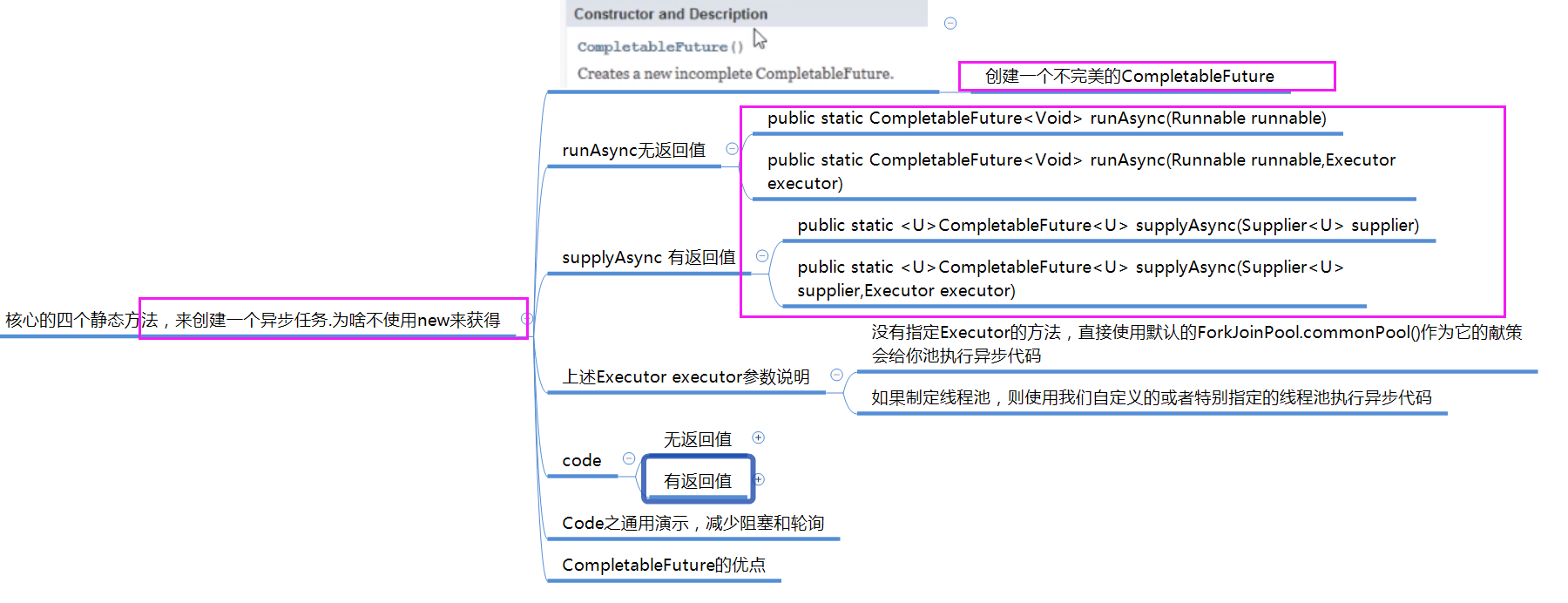
# 15CompletableFuture CompletionStage源码



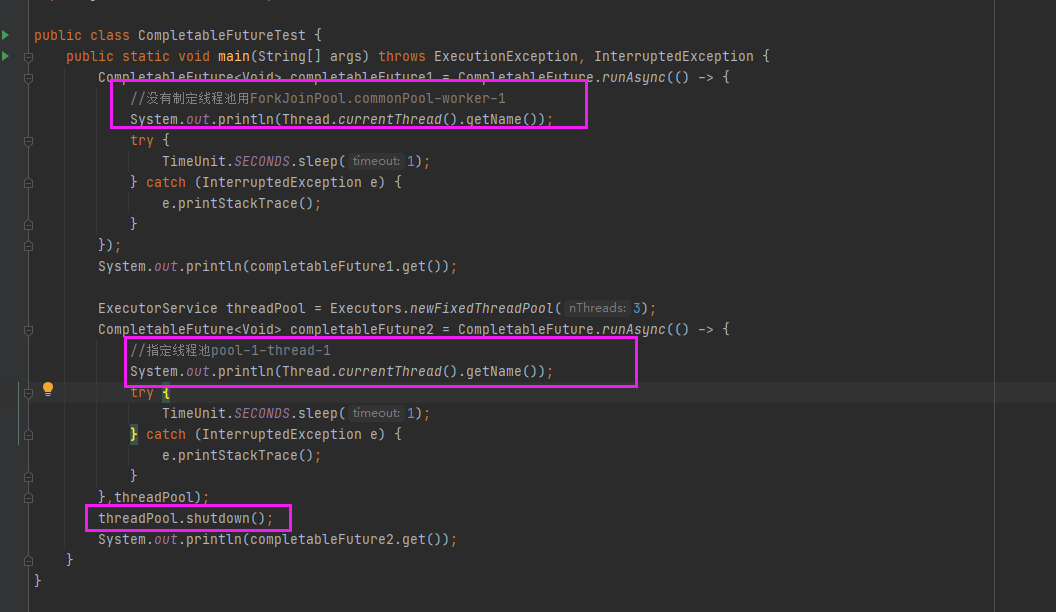


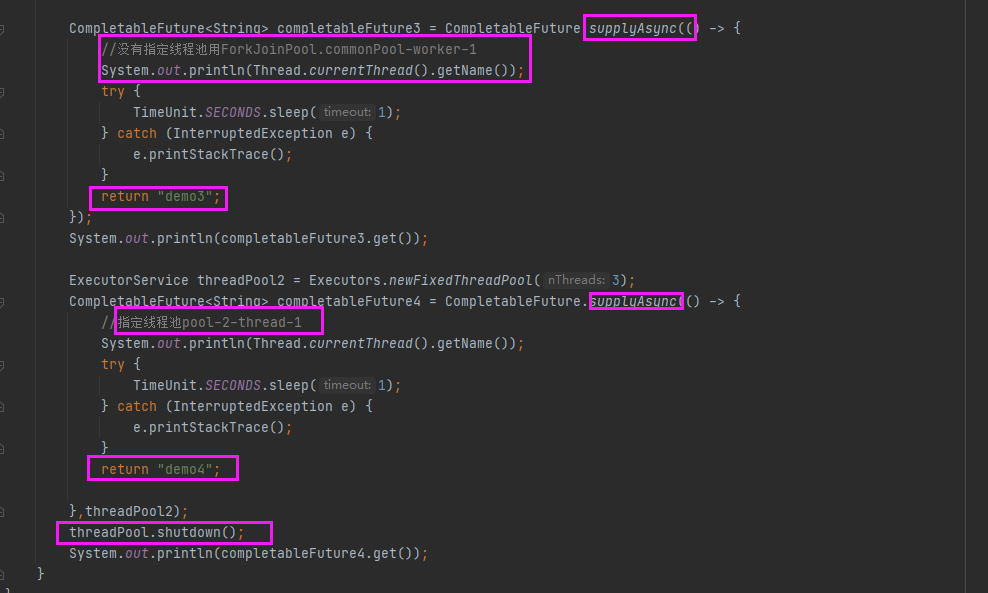


# 16CompletableFuture之四大静态方法初讲



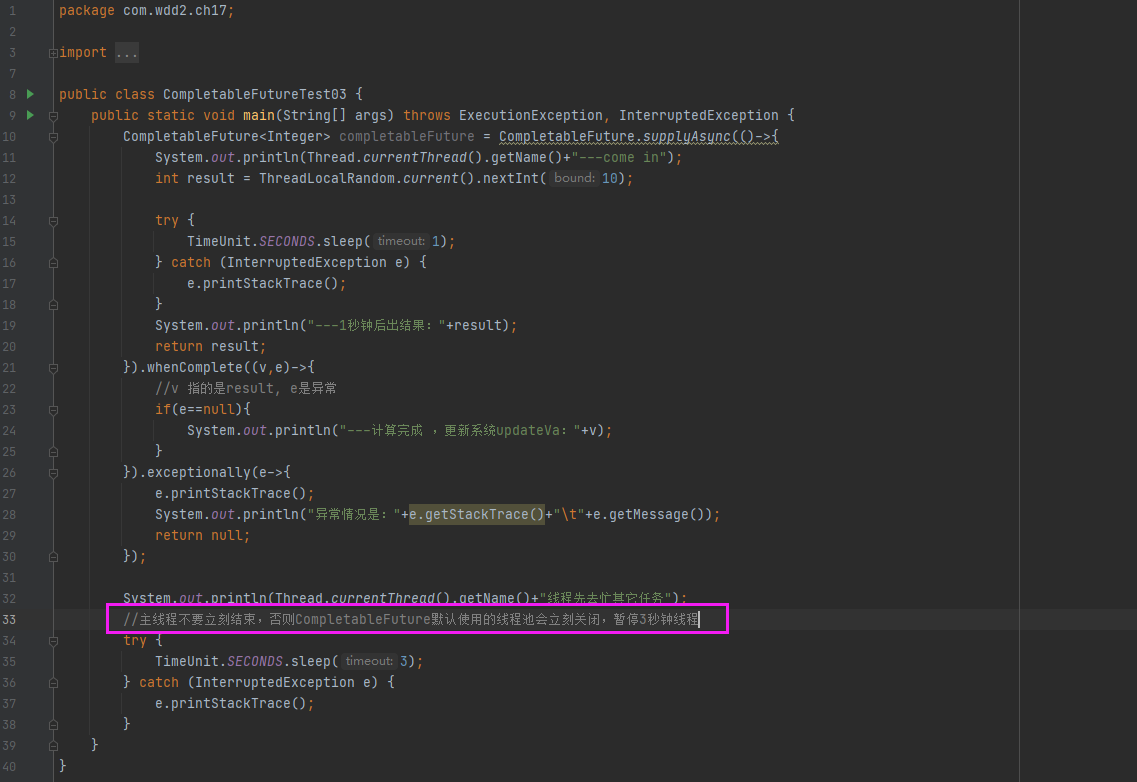
## Code



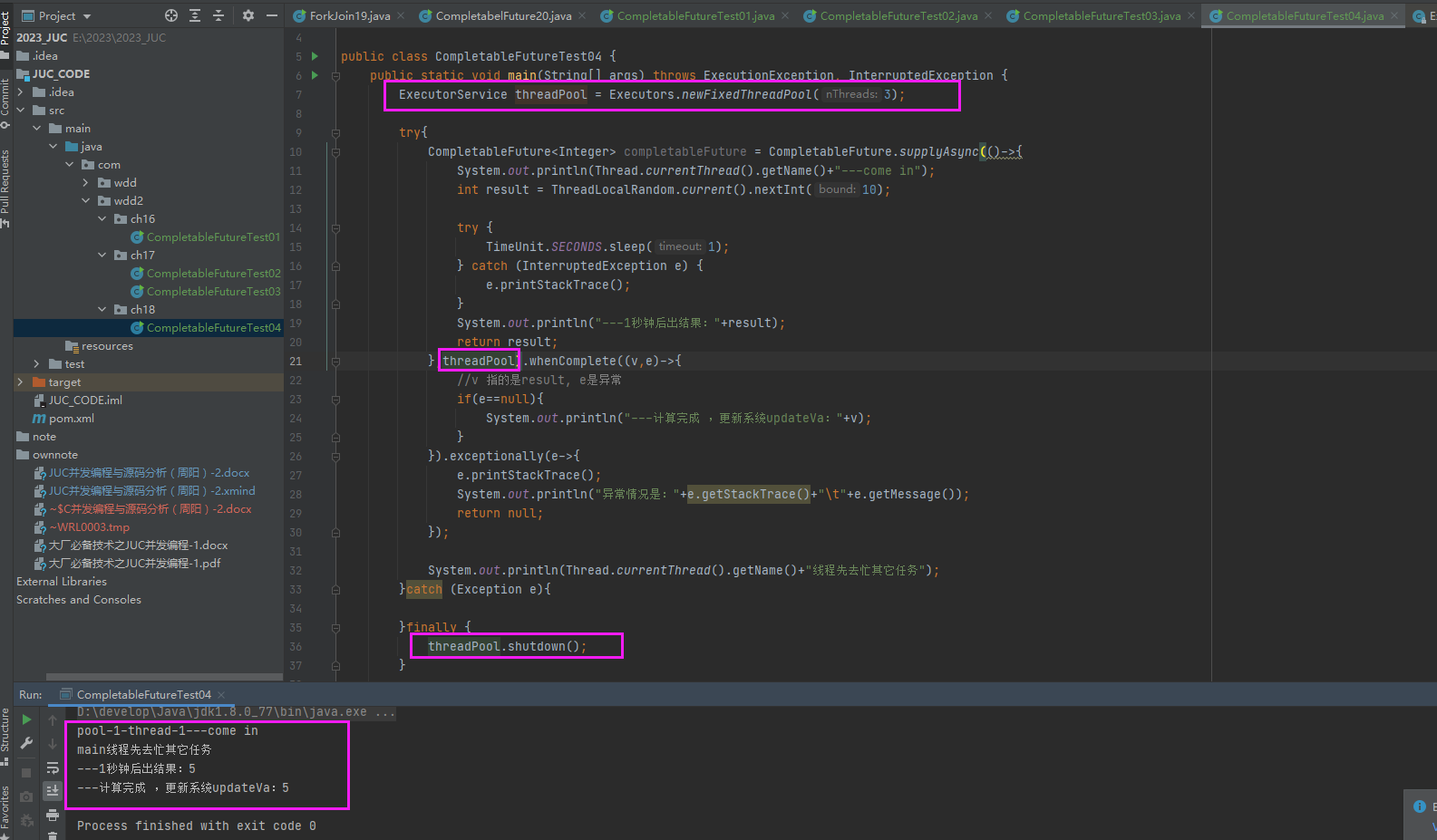


# 17CompletableFuture之通用异步编程-上集

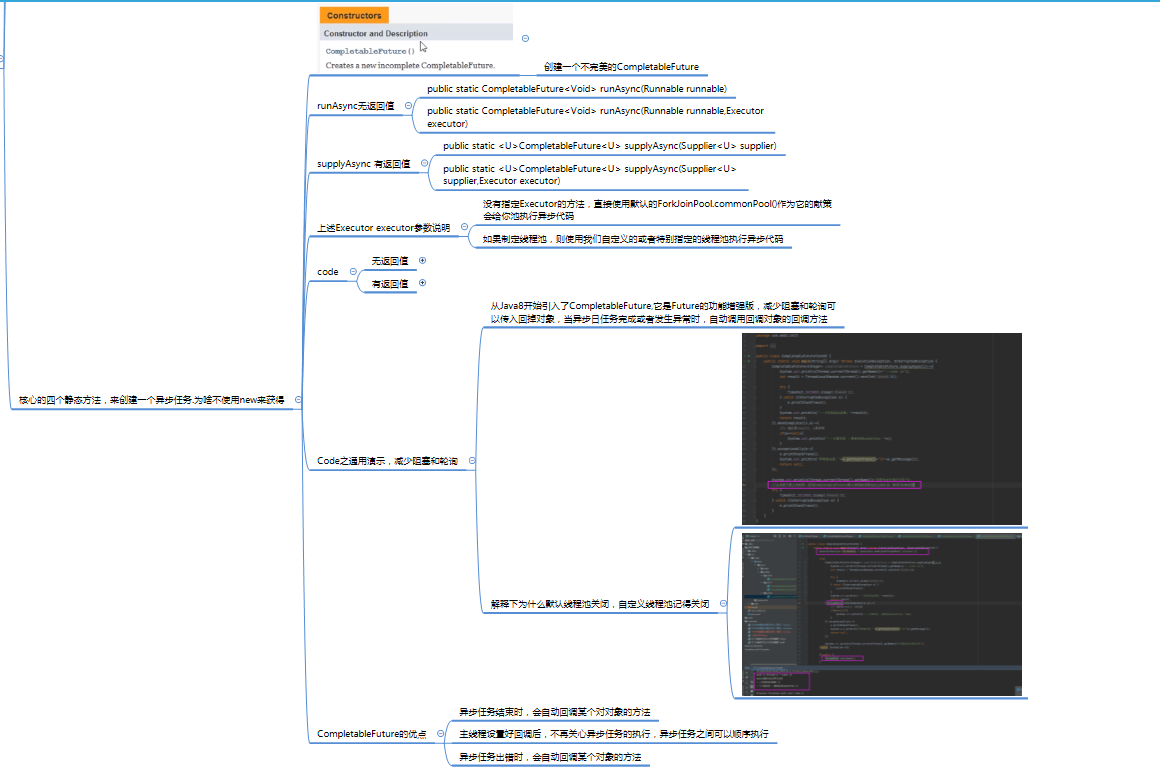
## 使用默认线程池



## 使用自定义线程池



# 18CompletableFuture之通用异步编程-下集



# 19CompletableFuture之链式语法和join方法介绍