https://mp.weixin.qq.com/s/kypZ8bi8YKP5MLiexMQOuA

# 获取线程结果的六种结果

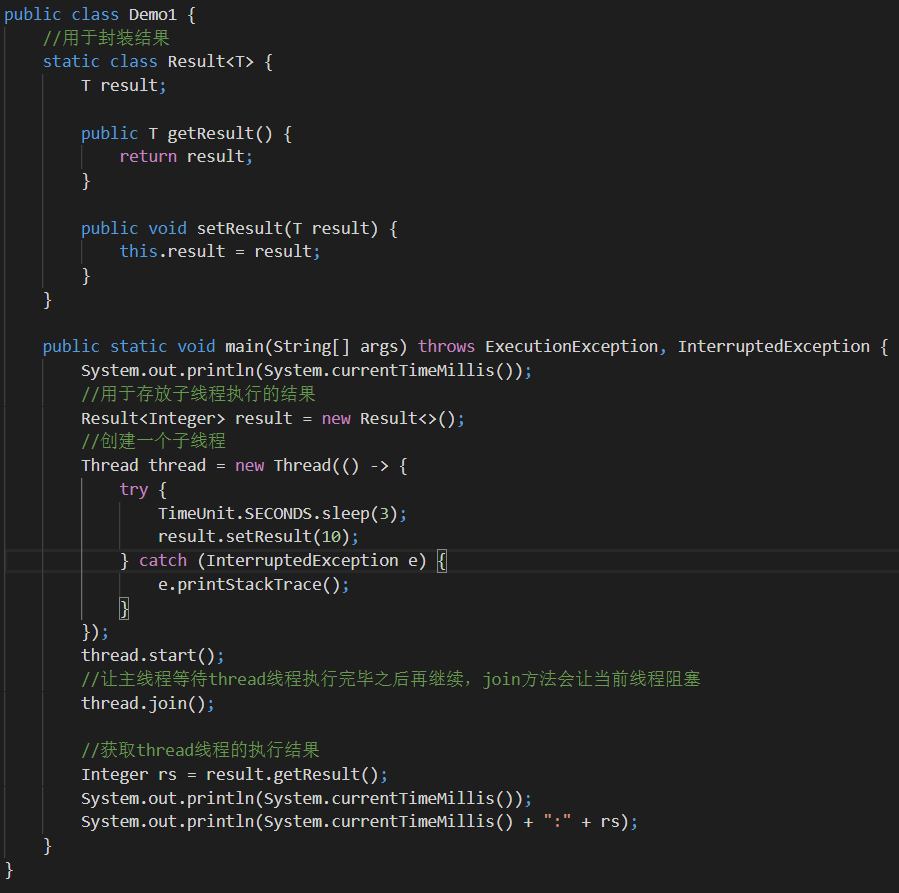
java高并发系列已经学了不少东西了，本篇文章，我们用前面学的知识来实现一个需求：

**在一个线程中需要获取其他线程的执行结果，能想到几种方式？各有什么优缺点？**

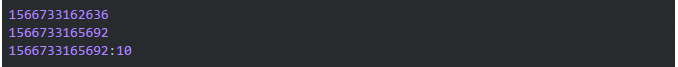
结合这个需求，我们使用**6种方式**，来对之前学过的知识点做一个回顾，加深记忆。

## 方式1：Thread的join()方法实现

代码：



输出：



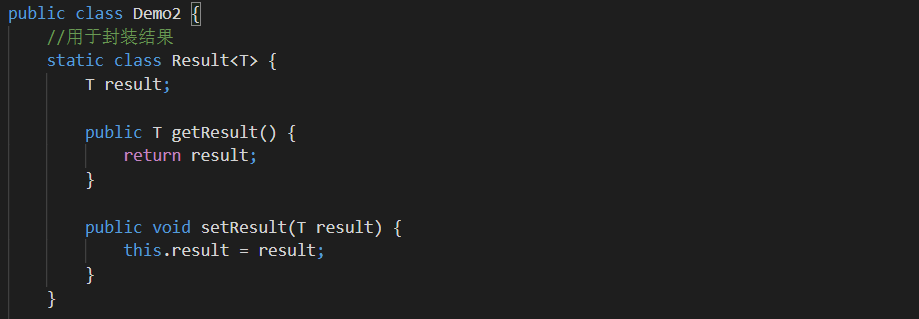
代码中通过join方式阻塞了当前主线程，当thread线程执行完毕之后，join方法才会继续执行。

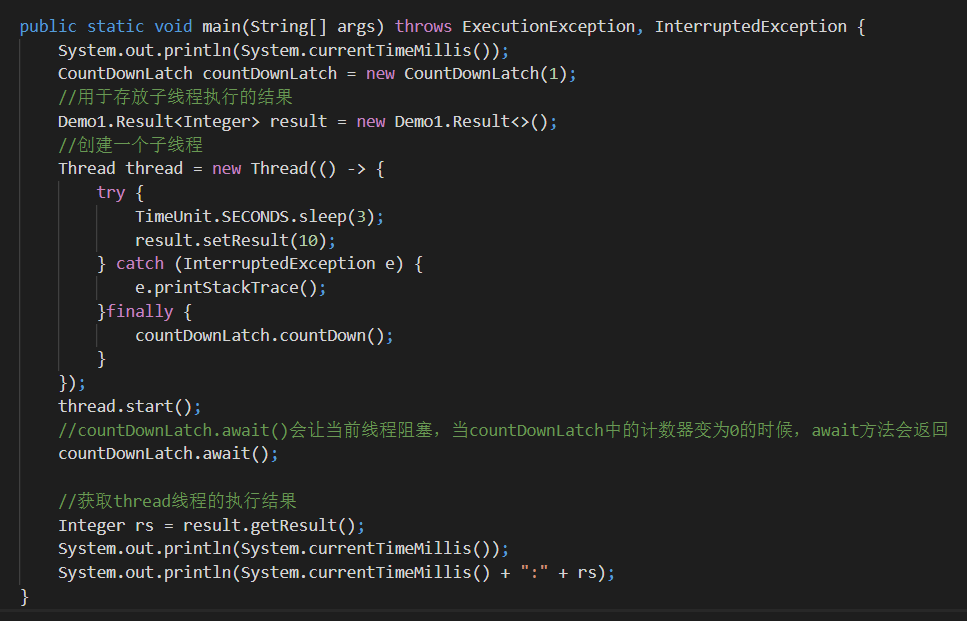
此行结论有误，请跳过：join的方式，只能阻塞一个线程，如果其他线程中也需要获取thread线程的执行结果，join方法无能为力了。

关于join()方法和线程更详细的使用，可以参考：[线程的基本操作](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933082&idx=1&sn=e940c4f94a8c1527b6107930eefdcd00&chksm=88621ae4bf1593f270991e6f6bac5769ea850fa02f11552d1aa91725f4512d4f1ff8f18fcdf3&token=2041017112&lang=zh_CN&scene=21#wechat_redirect)

## 方式2：CountDownLatch实现

代码：





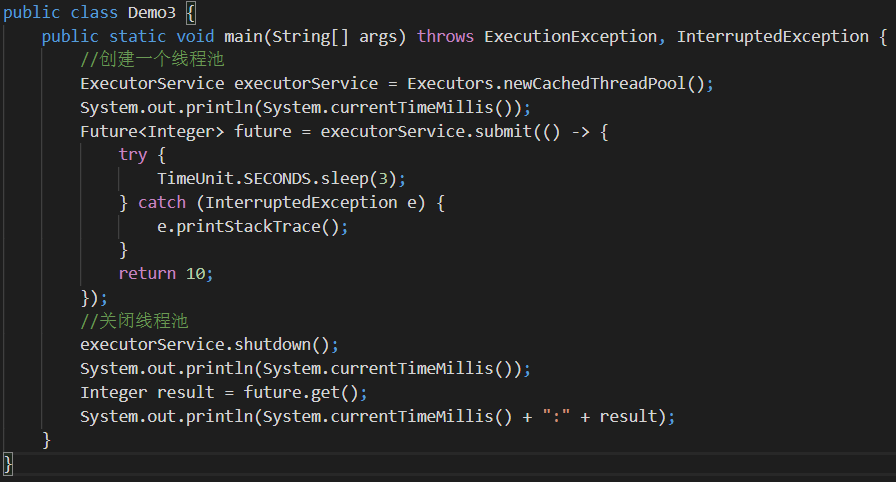
输出：



上面代码也达到了预期效果，使用CountDownLatch可以让一个或者多个线程等待一批线程完成之后，自己再继续；CountDownLatch更详细的介绍见：[JUC中等待多线程完成的工具类CountDownLatch，必备技能](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933134&idx=1&sn=65c2b9982bb6935c54ff33082f9c111f&chksm=88621b30bf159226d41607292a1dc83186f8928744dbc44acfda381266fa2cdc006177b44095&token=773938509&lang=zh_CN&scene=21#wechat_redirect)

## 方式3：ExecutorService.submit方法实现

代码：



输出：



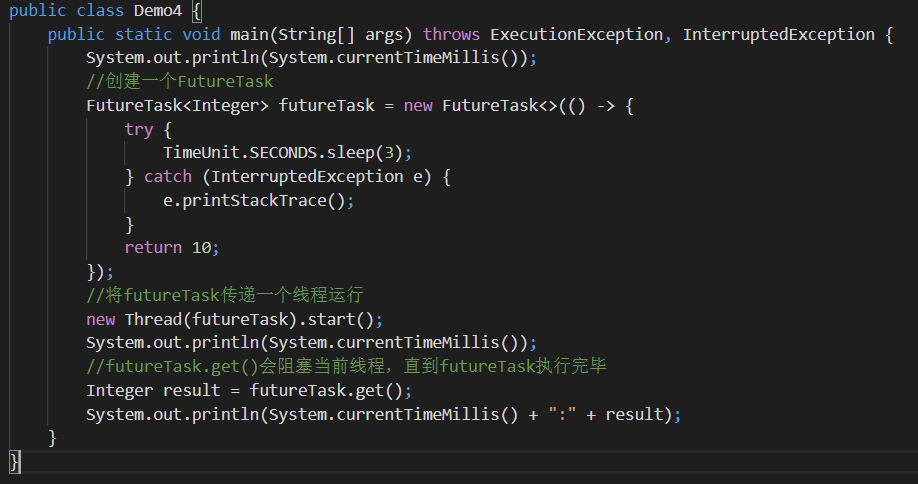
使用ExecutorService.submit方法实现的，此方法返回一个Future，future.get()会让当前线程阻塞，直到Future关联的任务执行完毕。

相关知识：

1. [JAVA线程池，这一篇就够了](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933151&idx=1&sn=2020066b974b5f4c0823abd419e8adae&chksm=88621b21bf159237bdacfb47bd1a344f7123aabc25e3607e78d936dd554412edce5dd825003d&token=995072421&lang=zh_CN&scene=21#wechat_redirect)
2. [JUC中的Executor框架详解1](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933156&idx=1&sn=30f7d67b44a952eae98e688bc6035fbd&chksm=88621b1abf15920c7a0705fbe34c4ce92b94b88e08f8ecbcad3827a0950cfe4d95814b61f538&token=995072421&lang=zh_CN&scene=21#wechat_redirect)
3. [JUC中的Executor框架详解2](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933160&idx=1&sn=62649485b065f68c0fc59bb502ed42df&chksm=88621b16bf159200d5e25d11ab7036c60e3f923da3212ae4dd148753d02593a45ce0e9b886c4&token=42900009&lang=zh_CN&scene=21#wechat_redirect)

## 方式4：FutureTask方式1

代码：



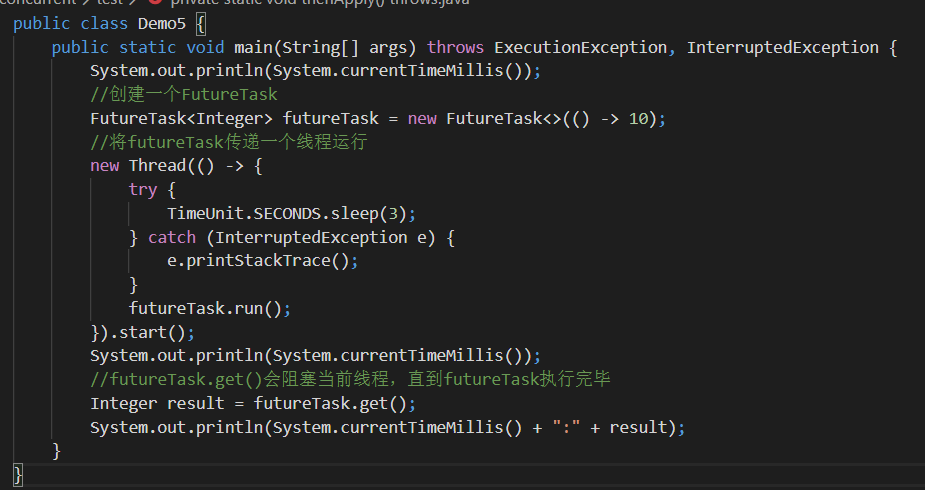
输出：



代码中使用FutureTask实现的，FutureTask实现了Runnable接口，并且内部带返回值，所以可以传递给Thread直接运行，futureTask.get()会阻塞当前线程，直到FutureTask构造方法传递的任务执行完毕，get方法才会返回。关于FutureTask详细使用，请参考：[JUC中的Executor框架详解1](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933156&idx=1&sn=30f7d67b44a952eae98e688bc6035fbd&chksm=88621b1abf15920c7a0705fbe34c4ce92b94b88e08f8ecbcad3827a0950cfe4d95814b61f538&token=995072421&lang=zh_CN&scene=21#wechat_redirect)

## 方式5：FutureTask方式2

代码：



输出：



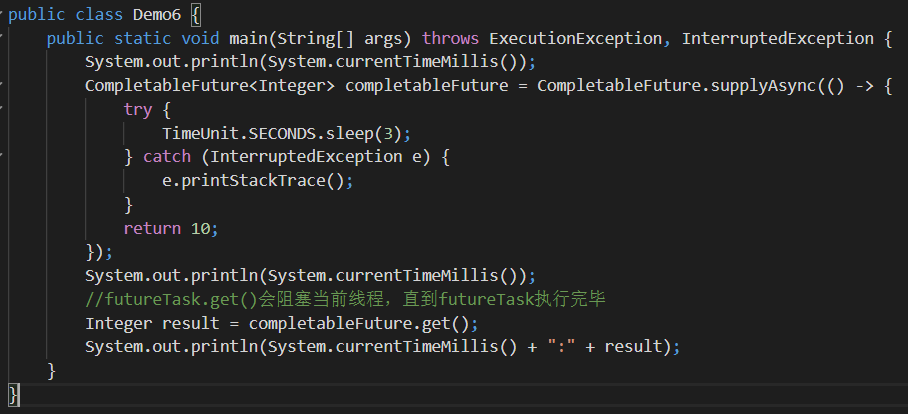
创建了一个FutureTask对象，调用futureTask.get()会阻塞当前线程，子线程中休眠了3秒，然后调用futureTask.run();当futureTask的run()方法执行完毕之后，futureTask.get()会从阻塞中返回。

注意：这种方式和方式4的不同点。

关于FutureTask详细使用，请参考：[JUC中的Executor框架详解1](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933156&idx=1&sn=30f7d67b44a952eae98e688bc6035fbd&chksm=88621b1abf15920c7a0705fbe34c4ce92b94b88e08f8ecbcad3827a0950cfe4d95814b61f538&token=995072421&lang=zh_CN&scene=21#wechat_redirect)

## 方式6：CompletableFuture方式实现

代码：



输出：



CompletableFuture.supplyAsync可以用来异步执行一个带返回值的任务，调用completableFuture.get()

会阻塞当前线程，直到任务执行完毕，get方法才会返回。

关于CompletableFuture更详细的使用见：[JUC中工具类CompletableFuture，必备技能](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933221&idx=1&sn=1af60b8917df6494b7c6b05c9eaebfe7&chksm=88621b5bbf15924d403e66e6d442d6b5897757471368b8d3a28c5de6e264cef104338dba1811&token=2098378399&lang=zh_CN&scene=21#wechat_redirect)