# JUC 的学习心得

1. 多线程的实现

Thread Runnable 都是去实现Runnable接口

实现线程池

实现callable接口（juc）

1. 线程同步

syn 不灵活 只能用在方法上和代码块上

Lock unlock (这是代替syn 去做锁的功能)

1. 线程通讯

Wait() notify()

Lock.condition

进程和线程的概念：

什么线程—》：java里面的 计算机原理里面的多线程

Thread1, 2, 3

4

程序： 一块数据

进程：独立程序的活动

线程：线程是进程的某一次进行过程

1. 线程同步锁

>1.syn:

>2.lock:

ReentrantLock， 可重入锁 相当于syn

ReentrantReadWriteLock.ReadLock 有读锁的线程们可以共享 读读可共享

ReentrantReadWriteLock.WriteLock 有写锁的线程不和读锁或者写锁共享， 读写，写写，不共享

1. 线程的实现方式

》1.继承Thread

》2.实现runnable接口

》3.线程池

》4.实现callable

1. Callable接口的实现：

可以和主线程通过用户自定义的操作来实现数据交互 实现call（）方法

并且可以获得线程call的执行结果

1.Get() 会阻塞主线程的执行

2.对于同一个任务， 多条线程反复执行， 只执行一遍

3.如果反复执行同一个任务， 需要再新建一个任务对象，然后新建线程来执行

Runnable 和Callable 的区别

1Java.lang Callable.util.concurrent

2Runnable 实现run方法，没有返回值， Callable实现call方法，有返回值

Syn 普通的同步关键字 锁对象 普通方法和同步方法不同步

Static syn 静态同步， 锁类 普通同步和静态同步不同步

lock

lock

lock

wait

notifyall

Syn

1. JUC的辅助工具类

1.数据库 读写锁

写写 同步 互斥 排他 阻塞

读写 异步 共享 非阻塞

2.Java程序的多线程

写写 同步 互斥 排他 阻塞

读写 juc的读写锁 同步 互斥 排他 阻塞

读读 异步 共享 非阻塞

**定制化线程**

CountDownLatch countDownLatch = **new** CountDownLatch(1)//创建object

countDownLatch.countDown();//执行完之后释放

countDownLatch.await();//执行前等待上面结束

>>>>>>>>>>>>>上面在主线程里面实现顺序执行程序<<<<<<<<<<

Lock lock =**new** ReentrantLock();//建立锁

**final** Condition cut = lock.newCondition();//创建对象锁位置

**final** Condition cook = lock.newCondition();//创建两一个锁的位置

lock.lock();//程序执行前就开始锁住

lock.unlock();//使用try finally里面去释放

cut.await();//等待释放

cook.signal();//通知另一个开始执行

>>>>>>>>>>>>>上面可以实现一个接另一个执行完成<<<<<<<<<<

实现有返回值的多线程

写一个类是实现接口Callable<Double> 并重写call()方法

MyCallable myCallable2 = **new** MyCallable();//创建实现callable接口的类的对象

FutureTask<Double> futureTask1 = **new** FutureTask<Double>(myCallable2);

**new** Thread(futureTask1).start();//线程开始执行程序

futureTask1.get() //这里是获取当前线程的返回值