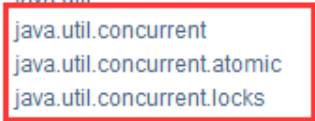
**JUC**

**概述**

JUC全称java util concurrent，指的是java util包的子包concurrent中提供给编程接口。



**进程和线程**

进程是应用程序的一次动态执行过程，也就是正在运行的程序，由操作系统调度

线程是进程执行任务一条路径，进程想要执行任务，就得依赖线程。线程是进程的基本单元，也是cpu调度的最小单位

**线程的生命周期经过几个状态**

java在枚举类State（State是在Thread类中定义的一个枚举类）中为线程定义了6个状态，分别是

新建、运行、等待、超时等待、阻塞、死亡

在操作系统层面来讲，就5个状态，分别是

新建、就绪、运行、阻塞、死亡

所以我认为理论上是6个状态，但总体来说是5个状态

**并发和并行**

并发是指一段时间内，cpu调度多个线程轮流执行

并行是指一个时间点，多个线程同时执行

**wait方法和sleep方法的区别**

1. 来自不同的类

wait是Object类中的方法，sleep方法是Thread类中的方法

1. 关于锁的释放

wait为释放锁，sleep不会

1. 使用的范围不同

wait只能在同步代码块中使用，sleep可以在任何地方使用

**Synchronized和Lock区别**

1. synchronized是一个关键字，Lock是一个接口
2. synchronized会自动使用释放锁，Lock必须手动释放锁
3. 对于synchronized来说，在多线程并发时，如果一个线程进入阻塞就无法释放锁，其它线程拿不到锁，就会一直等下。而Lock，在遇到这种情况，可以尝试获取锁，以确保程序正常运行。
4. synchronized使用的非公平锁，不可更改；Lock默认使用非公平锁，可以更该为公平锁

**函数式接口**

只有一个抽象方法的接口，被称为函数式接口。函数式接口分4大类：分别是Function接口（函数型接口），Predicate接口（断定型接口）

函数式接口可以使用lambda表达式简化。

lambda表达式的格式：(方法参数) -> {方法体}

若没有方法参数可以省略，即（）-> {方法体}

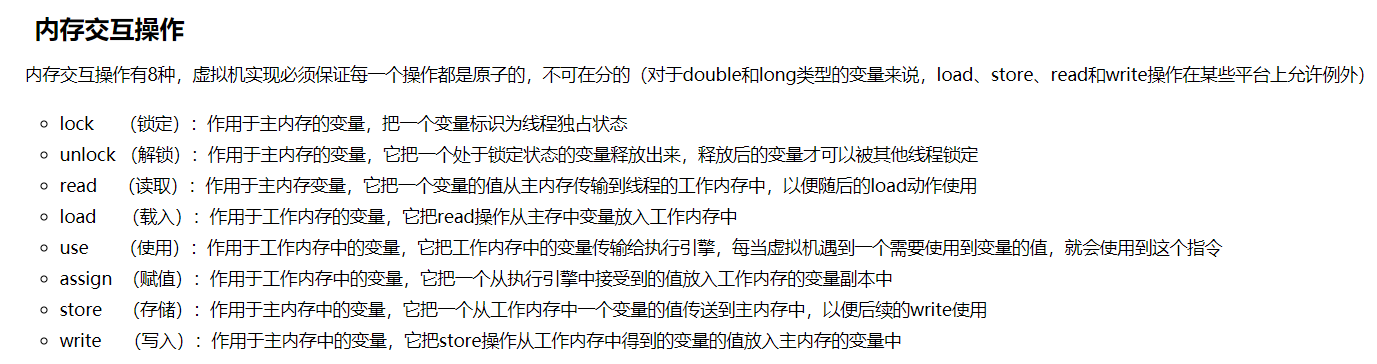
若方法只有一个参数，并且方法体中只有一条return语句，则（）和 { } 都可

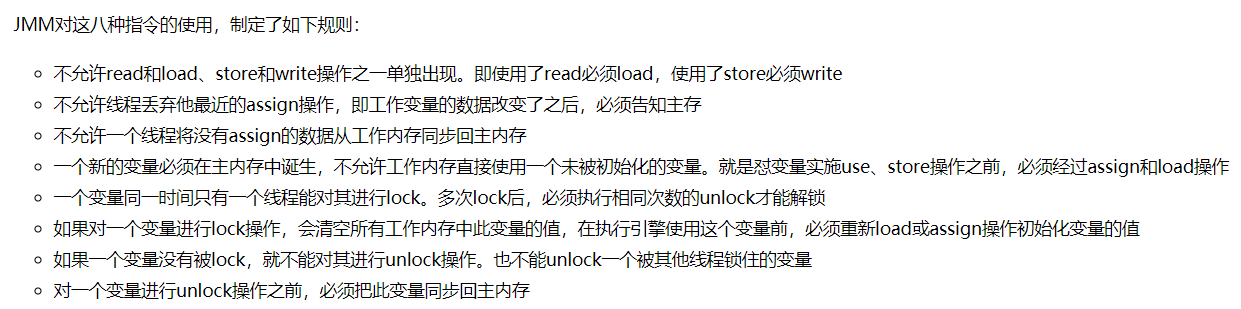
以省略，方法体中的return关键也可省略。

即：方法参数 -> 返回值

**JMM**

JMM（Java memory model），java内存模型。JMM规定了内存主要划分为主内存和工作内存两种。JVM在设计时候考虑到，如果JAVA线程每次读取和写入变量都直接操作主内存，对性能影响比较大，所以每条线程拥有各自的工作内存，工作内存中的变量是主内存中的一份拷贝，线程对变量的读取和写入，直接在工作内存中操作，而不能直接去操作主内存中的变量。但是这样就会出现一个问题，当一个线程修改了自己工作内存中的共享变量，对其他线程是不可见的，会导致线程不安全的问题（在共享变量上加volatile关键字可解决）。因为JMM制定了一套标准来保证开发者在编写多线程程序的时候，能够控制什么时候内存会被同步给其他线程。





**Volatile**

Volatile 是 Java 虚拟机提供轻量级的同步机制，特征如下：

1. 保证可见性

可见性是指：多个线程访问同一个变量时，一个线程修改这个变量的值，其

它线程能够感知到变量值的变化。

1. 不保证原子性
2. 禁止/防止指令重排

**CAS**

CAS（compare and swap）比较和替换是设计并发算法时用到的一种技术。简单来说，比较和替换是使用一个期望值和一个变量的当前值进行比较，如果当前变量的值与我们期望的值相等，就使用一个新值替换当前变量的值。