**可见性**

可见性：一个线程对共**享变量**值的修改，能够及时被其他线程看到。

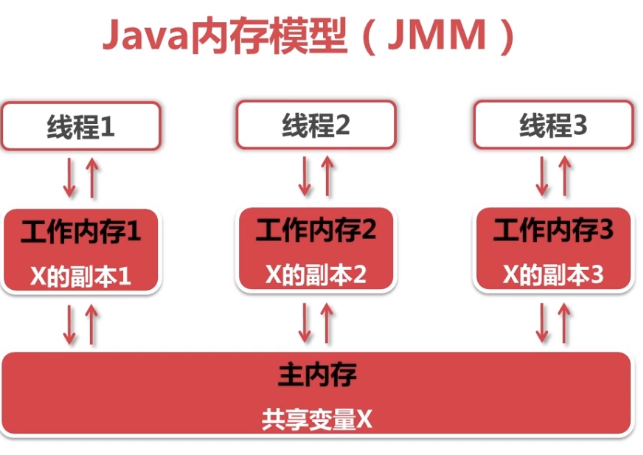
共享变量：如果一个变量在多个线程的**工作内存**中都存在副本，那么这个变量就是在这几个线程的共享变量。

什么是工作内存？这是JMM里概念。

**Java内存模型（JMM）--类似一种规范**

描述了Java程序中各种变量（线程共享变量）的访问规则，以及在JVM中将变量存储到内存和从内存中读取出变量（共享变量）这样的底层细节。

所有的变量都存储在主内存中，每个线程都有自己独立的工作内存，里面保存该线程使用到的变量的副本。



**两条规定**

1、线程对共享变量的所有操作都必须在自己的工作内存中进行，不能直接从主内存中读写。

2、不同线程之间无法直接访问其他线程工作内存中的变量，线程间变量值的传递需要通过主内存来完成。

**共享变量可见性实现的原理**

1. 把工作内存中更新过的共享变量刷新到主内存中
2. 将主内存中更新的共享变量的值更新到另一个工作内存中、

**可见性的实现方式**

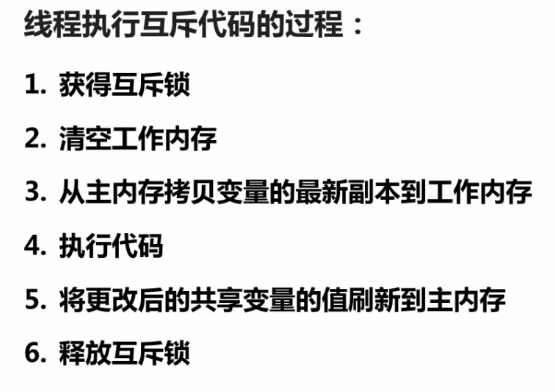
**Java语言层面支持的可见性实现方式：**

Syncronized：

能实现原子性、可见性

线程解锁前，必须把共享变量的最新值刷新到主内存中。

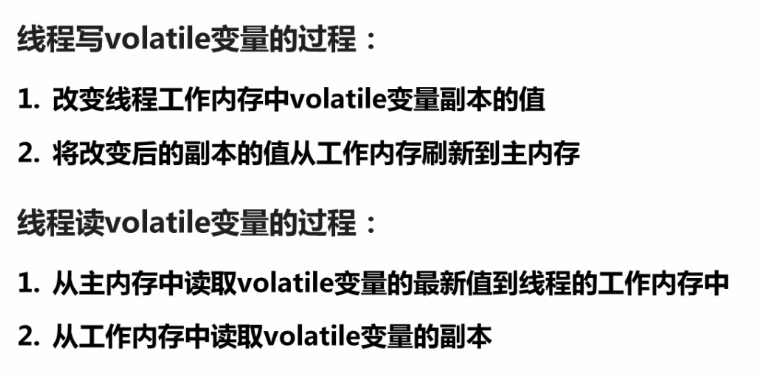
线程加锁时，将清空工作内存中共享变量的值，从而使用共享变量时需要从主内存中重新读取最新过的值。



Volatile：

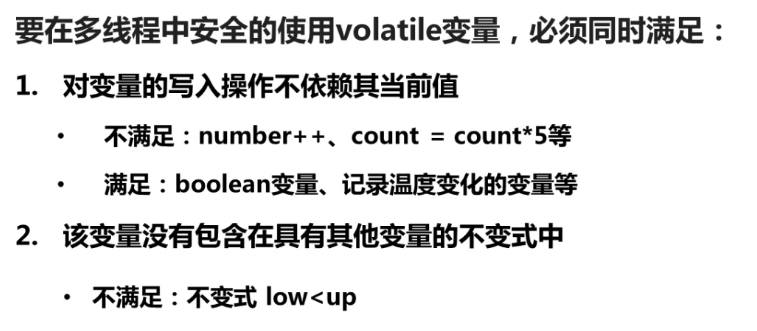
能够保证Volatile变量的可见性，不能保证Volatile变量的复合操作的原子性（i++）

通过加入内存屏障和禁止重排序优化来实现的。



保证number自增操作的原子性：

1. 使用Syncronized关键字
2. 使用ReentrantLock
3. 使用AtomicInterger



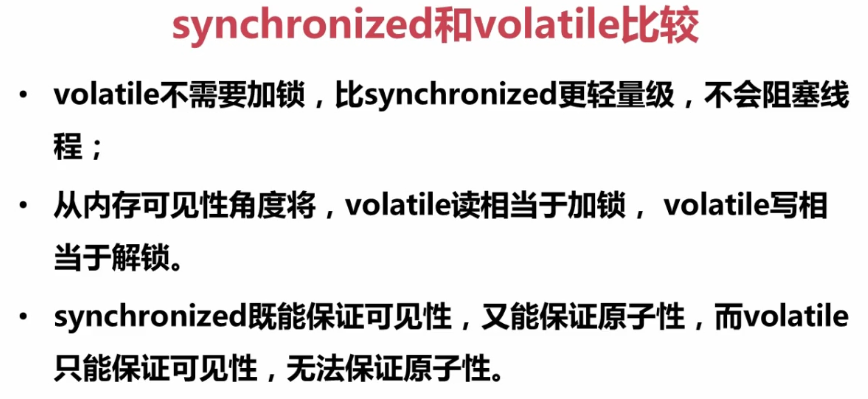
涉及到的知识点：

**重排序：**代码书写的顺序与实际执行的顺序不同，指令重排序是编译器或处理器为了提高程序性能而做的优化。

1. 编译器优化的重排序---单线程中保证结果正确的情况下进行的优化。
2. 指令级并行重排序
3. 内存系统的重排序---处理器对读写缓存的重排序

**As-if-serial语义**

Java编译器、运行时和处理器都会保证Java在**单线程**下遵循as-if-serial语义。



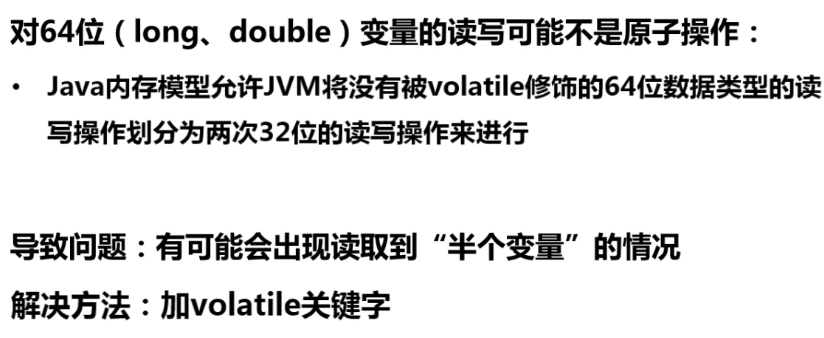
**实现可见性的方式：**

**Synchronized和volatile**

**Final也可以保证内存可见性。**

**有时候即使没有保证可见性的操作，很多时候共享变量依然能够在主内存和工作内存见得到及时的更细？**

一般只有在短时间内高并发的情况下才会出现变量得不到及时更新的情况，因为CPU在执行时会很快地率性缓存，所以一般很难看到。



其实现在很多平台商务虚拟机对long，double当做原子对待了。

**总结：volatile比synchronized更轻量级，volatile没有synchronized使用广泛**