# JMM内存模型

JMM（Java内存模型）本身是一种抽象的概念并不真实存在，它描述的是一组规则或规范，通过这组规范定义了程序中各个变量（包括实例字段，静态字段和构成数组对象的元素）的访问方式。

1

JMM三大特性

可见性原子性有序性

2

JMM关于同步的规定

1、线程解锁前，必须把共享变量的值刷新回主内存；

2、线程加锁前，必须读取主内存的最新值到自己的工作内存；

3、加锁解锁是同一把锁；

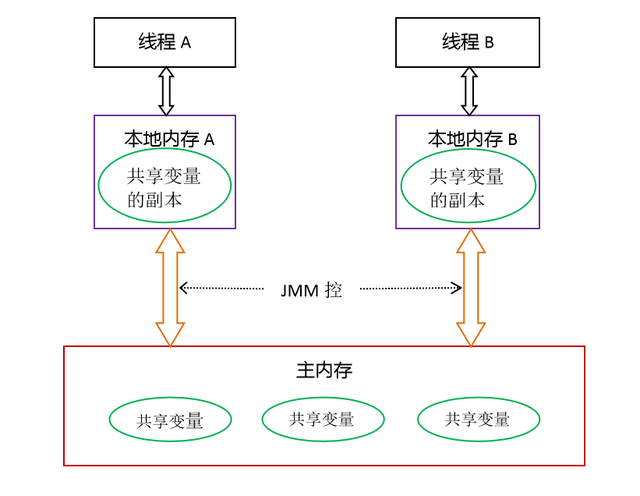
由于JVM运行程序的实体是线程，而每个线程创建时JVM都会为其创建一个工作内存（有些地方称为栈空间），工作内存是每个线程的私有数据区域，而Java内存模型中规定所有的变量都存储在主内存，主内存是共享内存区域，所有的线程都可以访问，但线程对变量的操作（读取赋值等）必须在工作内存中进行，首先要将变量从主内存拷贝到自己的工作内存空间，然后对变量进行操作，操作完成后在将变量写回主内存，不能直接操作主内存中的变量，各个线程的工作内存中存储着主内存中的变量副本拷贝，因此不同的线程间无法访问对方的工作内存，线程间的通信必须通过主内存来完成。

4

模型

JMM是一个抽象的概念，并不是真实的存在，它涵盖了缓冲区，寄存器以及其他硬件和编译器优化。

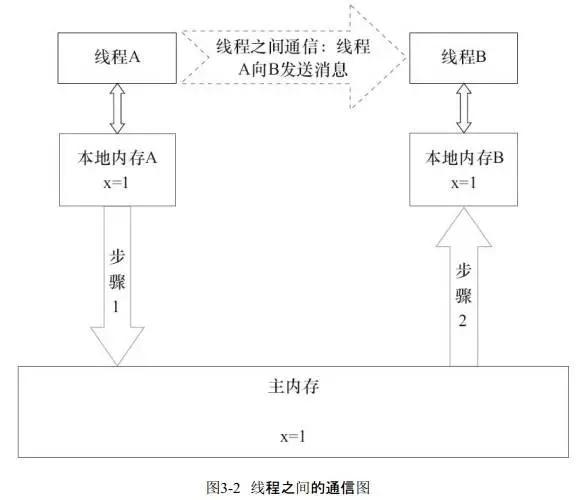
Java内存模型抽象图如下：



从上图可以看出每个线程都有一个本地内存，如果线程想要通信的话要执行一下步骤：

A线程先把本地内存的值写入主内存B线程从主内存中去读取出A线程写的值

再看下面的这个图，表示了A如何向B发送消息



假设这时候有一个共享变量X默认值都是为0，那么线程A把X的值修改为1,这时候如何才能同步到B线程呢。

如果A线程把X修改成1之后，A线程会把X从A的本地内存中写入到主内存中，这样的话主内存的X就等于1了，这时候B线程就会去读取主内存的X变量，存入B的本地内存中，这样B线程的X变量值也就会变成了1。

5

线程同步规则

如果要把一个变量从主内存复制到工作内存，就需要按顺序的执行read和load操作，如果把变量从工作内存同步到主内存中，就要按顺序执行store和write操作。但是Java内存模型只要求上述操作必须按顺序执行，而没有保证必须是连续执行。

不允许read和load、store和write单独出现；

不允许一个线程丢弃它的最近assign的操作，即变量在工作内存中改变了之后必须同步到主内存中；

不允许一个线程无原因（没有发生过任何assign操作）就把数据从工作内存同步到主内存中；

一个新的变量只能在主内存中诞生，不允许在工作内存中直接使用一个未被初始化（load或assign）的变量。也就是对一个变量实施use和store操作之前，必须先执行过assign和load操作。

一个变量在同一时刻只允许一条线程对其进行lock操作，但是lock操作可以被同一条线程重复执行多次，多次执行lock后，只执行相同次数的unlock操作，变量才会被解锁。lock和unlock必须成对出现。

如果对一个变量执行lock操作，将会清空工作内存中此变量的值，在执行引擎使用这个变量前需要重新执行load或assign操作初始化变量的值。

如果一个变量事先没有被lock锁定，则不允许对他执行unlock操作，也不允许去unlock一个被其他线程锁定的变量。

对一个变量执行unlock操作之前，必须事先将此变量同步到主内存中（执行store和write）

