JAVA内存模型

JAVA虚拟机规范中试图定义一种JAVA内存模型（Java Memory Model, JMM）来屏蔽掉各种硬件和操作系统的内存访问差异，以实现让Java程序在各种平台下都能达到一致的内存访问效果。

Java内存模型的主要目标是定义程序中各个变量的访问规则，即虚拟机中将变量存储到内存和从内存中取出变量这样的底层细节。此处的变量（variables）与Java编程中所说的变量有所区别，它包括实例字段、静态字段和构成数组对象的元素，但不包括局部变量与方法参数，因为后者是线程私有的，不会被共享，自然就不会存在竞争问题。

Java线程

Java线程

Java线程

工作内存

工作内存

工作内存

Save和Load操作

线程、主内存、工作内存三者的交互关系

Java使用的线程调度方式是抢占式调度，抢占式调度中，每个线程将由系统来分配执行时间，线程的切换不由线程本事来决定（在Java中，Thread.yield()可以让出执行时间，但是要获取执行时间的话，线程本身是没有什麽办法的）。我们可以人为地设置线程优先级（Java设置了10个线程优先级）。

Java定义了5种线程状态：新建(New)、运行(Running)、无期限等待(Waiting)、期限等待(Timed Waiting)、阻塞(Blocked)、结束(Terminated)。

run()结束

Sleep()

synchronized

Start()

Wait()

New

Waiting

Blocked

Timed Waiting

Terminated

Runningg

notify()/notifyAll()

线程状态转换关系

阻塞（Blocked）:线程阻塞了，“阻塞状态”与“等待状态”的区别是：“阻塞状态”在等待着获取到一个排它锁，这个事件将在另一个线程放弃这个锁的时候发生；而“等待状态”则是在等待一段时间，或者唤醒动作的发生。在程序等待进入同步区域的时候，线程将进入这种状态。

重排序：

同步将限制编译器、运行时和硬件对内存操作重排序的方式，从而在实施重排序时不会破坏JVM提供的可见性保证。

JMM为程序中所有的操作定义了一个偏序关系，称之为Happens-Before。要想保证执行操作B的线程看到操作A的结果（无论A和B是否在同一个线程中执行），那么在A和B之间必须满足Happens-Before关系。如果两个操作之间缺乏Happens-Before关系，那么JVM可以对它们任意地重排序。

Happens-Before的规则包括：

程序顺序规则。如果程序中操作A在操作B之前，那么在线程中A操作将在B操作之前执行。

监视器锁规则。在监视器锁上的解锁操作必须在同一个监视器锁上的加锁操作之前执行。

volatile变量规则。对volatile变量的写入操作必须在对该变量的读操作之前执行。

线程启动规则。在线程上对Thread.Start的调用必须在该线程中执行任何操作之前执行。

线程结束规则。线程中的任何操作必须在其他线程检测到该线程已经结束之前执行，或者从 Thread.join中成功返回，或者在调用Thread.iaAlive时返回false。

中断规则。当一个线程在另一个线程上调用interrupt时，必须在被中断线程检测到interrupt调用之前执行（通过抛出InterruptedException，或者isInterrupted和interrupted）。

终结器规则。对象的构造函数必须在启动该对象的终结器之前执行。

传递性。如果操作A在操作B之前执行，并且操作B在操作C之前执行，那么操作A必须在操作C之前执行。