下面就来具体介绍下**happens-before原则（先行发生原则）：**

* 程序次序规则：一个线程内，按照代码顺序，书写在前面的操作先行发生于书写在后面的操作
* 锁定规则：**一个unlock操作先行发生于后面对同一个锁的lock操作。（先解锁，才可以在后面继续上锁）**
* volatile变量规则：对一个变量的写操作先行发生于后面对这个变量的读操作
* 传递规则：如果操作A先行发生于操作B，而操作B又先行发生于操作C，则可以得出操作A先行发生于操作C
* **线程启动**规则**：Thread对象的start()方法先行发生于此线程的每个一个动作**
* **线程中断**规则：对线程**interrupt()方法的调用**先行发生于被中断**线程的代码检测到中断事件的发生**
* **线程终结**规则：线程中**所有的操作都先行**发生于线程的**终止检测**，我们可以通过Thread.join()方法结束、Thread.isAlive()的**返回值手段检测到线程已经终止执行(所有终结的线程都不可再用)**
* **对象终结规则(GC)**：一个**对象的初始化**完成先行发生于**他的finalize()方法的开始**

**这8条原则**摘自《深入理解Java虚拟机》。

　　这8条规则中，前4条规则是比较重要的，后4条规则都是显而易见的。

　　下面我们来解释一下前4条规则：

　　对于程序次序规则来说，我的理解就是一段程序代码的执行在单个线程中看起来是有序的。注意，虽然这条规则中提到“书写在前面的操作先行发生于书写在后面的操作”，这个应该是程序看起来执行的顺序是按照代码顺序执行的，因为虚拟机可能会对程序代码进行指令重排序。虽然进行重排序，但是最终执行的结果是与程序顺序执行的结果一致的，它**只会对不存在数据依赖性的指令进行重排序**。因此，在单个线程中，程序执行看起来是有序执行的，这一点要注意理解。事实上，这个规则是用来保证程序在单线程中执行结果的正确性，但**无法保证程序在多线程中执行的正确性**。

　　第二条规则也比较容易理解，也就是说无论在单线程中还是多线程中，**同一个锁如果出于被锁定的状态，那么必须先对锁进行了释放操作，后面才能继续进行lock操作**。

　　第三条规则是一条比较重要的规则，也是后文将要重点讲述的内容。直观地解释就是，如果一个线程先去写一个变量，然后一个线程去进行读取，那么**写入操作肯定会先行发生于读操作。**

　　第四条规则实际上就是体现happens-before原则**具备传递性**。