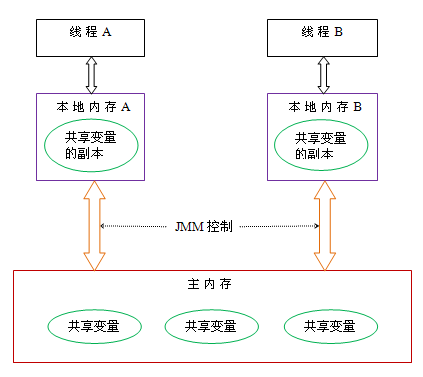
# 概述

主要掌握知识点：

* 可见性
* 重排序

## 内存模型抽象结构



* 所有的java对象都存放在内存中（也就是不管jvm哪个区的对象，都在主内存）；
* 每个线程都有其独立的本地内存（逻辑概念），本地内存存放的是主内存对象的副本；
* 共享变量包括（堆上对象，静态对象、数组对象）；
* 局部变量、方法参数、异常处理参数都不被共享，属于线程私有；
* 线程对所有共享变量的操作都是在本地内存，不会直接操作主内存中的共享变量；
* 不同线程无法直接访问其他线程本地内存中的对象，线程间的共享变量需要主内存进行传递；

# 知识点

## 可见性

一个线程对共享变量的更新，修改可以被其他线程所“看见”（也就是可以被访问）。

可见性是指当多个线程访问同一个变量时，一个线程修改了这个变量的值，其他线程能够立即看得到修改的值。

为了达到共享变量的可见性，java提供volatile关键字保证被修改的变量立即更新到主内存，而不会保存在线程的私有内存中。

普通变量不能保证可见性，对普通变量值的修改无法确定更新到主内存的时间；因此其他线程读取该变量时，可能仍然读取到的是旧值，从而无法保证其可见性；

Synchronize和Lock也可以保证可见性。Synchronize和Lock保证同一时刻只有一个线程可以访问临界区内的代码，别去锁在释放的时候会见变量的值更新到主内存；

### 实现

## 原子性

## 重排序

* 对没有数据依赖性的操作指令进行重新排序；
* 编译器重排序和处理器重排序，针对单个处理器和单个线程中的操作指令，不同处理器和不同线程之间的指令不在考虑范围内；
* 无论如何重排序，（单线程）程序执行的结果保持不变（as-if-serial规则）

## 顺序一致性

一个理想化的内存模型，包括两个特性：

* 一个线程中的所有操作都严格按照程序的顺序执行（也就是不会出现重排序现象）；
* （不管程序是否同步）所有线程都只能看到一个单一的操作执行顺序；

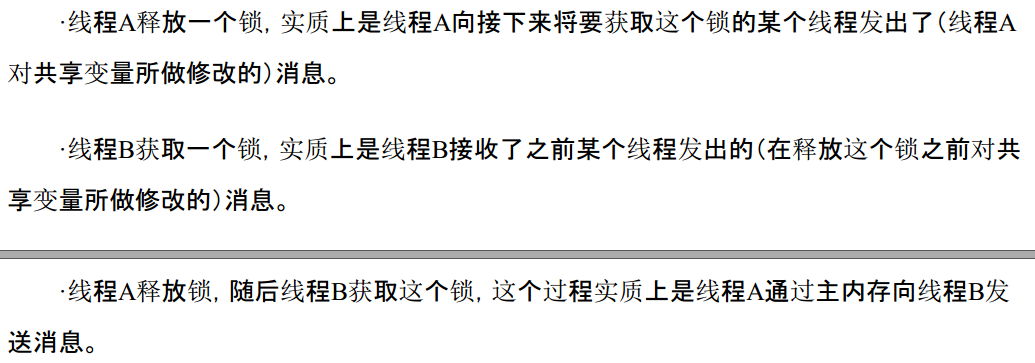
这两点特性中，第一点表明程序不会出现重排序，第二点比较重要，表明每个操作都具有原子性并且立刻对所有线程可见。

单一操作执行顺序就是指不管程序同步与否，所有线程看到的程序执行过程都是一样的（这一点有特性2保证，如果不是立即可见，某一个操作的结果可能存放在线程的本地内存，对其他线程来说这个操作被认作为没有执行）。

## 锁

* 当线程释放锁时，JMM会把该线程对应的本地内存中共享变量刷新到主内存；
* 当线程获取锁时，JMM会把该线程对应的本地内存中共享变量置为无效，以此保证线程在执行临界区代码时必须从主内存中获取共享变量的最新值；

锁释放-获取的内存语义和volatile写-读的内存语义一样（注意是释放对应写，获取对应读）；



## Happen-before

## Volatile

volatile写的内存语义：

当对一个volatile变量的写操作时，JMM内存模型会将该线程对应的本地内存中共享变量的值刷新到主内存；

Volatile读的内存语义：

当对一个volatile变量的读操作时，JMM内存模型会把该线程本地内存置为无效，然后从主内存中读取共享变量的值；

特性：

* 可见性：对一个volatile变量的读总能看到（任意线程）对这个volatile变量最后的写操作；
* 原子性：对单个volatile变量的读写操作具有原子性，复合操作不具有原子性（比如volatile++就是一个复合操作）

## Final

# 参考

系列文章<http://ifeve.com/java-memory-model-1/>

《java并发编程艺术》第三章