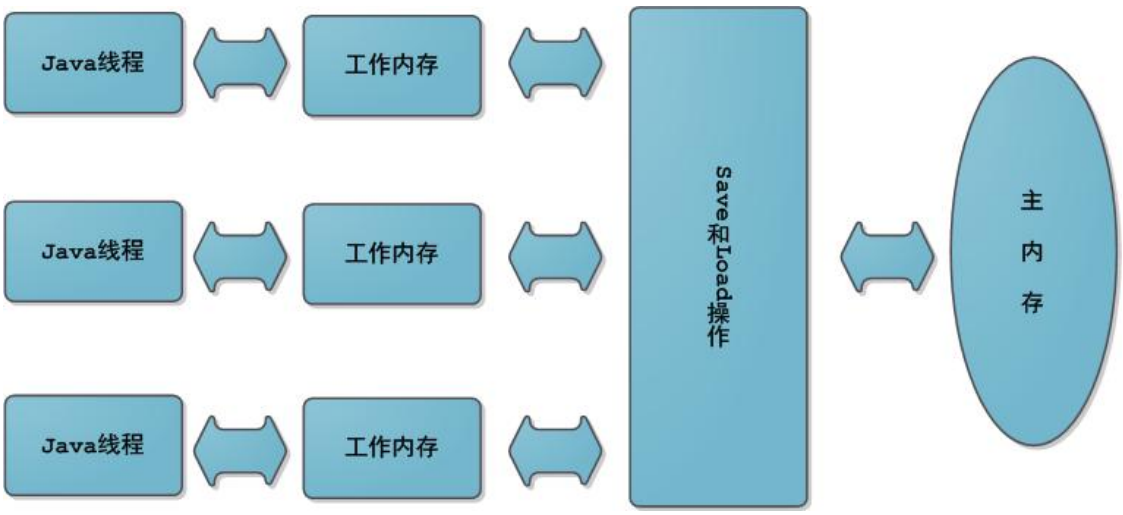
一、java内存模型

java定义的内存模型是类似物理计算机的三级缓存模型

1.主内存与工作内存

java内存模型定义的是变量的访问规则，即在虚拟机中将变量存储到内存和从内存读取变量这样的底层细节。
1、java内存模型规定所有的变量都存储在主内存。
2、每条线程都有自己的工作内存，该线程的工作内存中存储该线程使用到的变量的主内存副本拷贝。
3、线程对变量的读取赋值等操作都是在工作内存中，而不能直接读写主存。不同线程之间不能直接访问对方的工作线程，而是通过主内存进行交互。



java并发会带来三个问题，java中会有一系列措施来解决这三个问题

> #### 1.可见性
>
> #### 2.有序性
>
> #### 3.原子性

2.内存间交互操作

1. lock（锁定）：作用于主存的变量，它将一个变量标识为一条线程独占的状态
2. unlock（解锁）：作用于主内存，它将一个处于锁定状态的变量释放出来，释放后的变量能继续被其他线程使用。
3. read（读取）：作用于主内存变量。将一个变量从主内存传输到线程的工作内存，以便随后的load操作
4. load（载入）：作用于工作内存的变量。将从主内存读取到的变量值放入到工作内存的副本变量中。
5. use（使用）：将工作内存的变量值传递到执行引擎供虚拟机使用
6. assign（赋值）：作用于工作内存的变量。将执行引擎得到的赋给工作内存的变量。
7. store（存储）：作用于工作内存变量。将工作内存的变量值传递回主内存中，以便随后的write操作。
8. write（写入）：作用于主内存变量。将工作内存得到的变量值赋给主内存变量。

以上8个操作会有等价happens-before原则

3.volatile关键字

volatile用于修饰变量，很好的解决了可见性和有序性的问题。但是，*并没有保证原子性*

1. 可见性：当一个线程修改了一个变量的值，该变量值得修改对其他线程来说总是能立即看到。

> 可见性依赖的条件：
> （1）运算的结果并不依赖变量的当前值,或者能够确保只有单一的线程修改变量的值
> （2）变量不依赖其他变量值。

2. 有序性：被volatile修饰的变量，虚拟机在变量前后会保证指令执行顺序不会被优化重排。（volatile相当于在该处增加了一道内存屏障，禁止前后指令顺序跨越）

综上，volatile的作用等价于以下：

> (1)每次读取变量时都到主内存刷新最新值，以保证对其他线程修改变量的可见
> (2)每次线程对变量修改后都应立即同步回主内存，以让其他线程能够看到自己

> 对变量的修改
> (3)volatile修饰的变量不会被指令重排序优化，保证代码的执行顺序与程序的顺序相同。

4. 原子性、可见性、有序性

（1）原子性

java提供基本的原子性变量操作，read，load，assign，use，store，write。
更大范围的原子性操作则需要加锁。

> lock，unlock。 sychronized。

（2）可见性

当一个线程修改了变量的值，该修改的值对其他线程是立即可见的。

> lock unlock
> synchronized,基于锁的原理，unlock之前必须把变量同步回主存
> volatile
> final：虚拟机会对final修饰的字段进行优化，被final修饰的字段在构造器中一旦初始化完成，并且构造器没有把this引用传递出去，那么其他线程中就能看见final字段的值。

（3）有序性

java中的有序性是指线程内串行指令。
指令重排序
主内存与工作内存同步延迟

> volatile 本身具有禁止指令重排序的功能
> synchronized 加锁的原理，一个变量同一时刻只允许一条线程对其进行lock操作
> lock， unlock