**Java并发**

1. 实现多线程的三种方式

* 实现Runnable接口
* 继承Thread类
* 实现Callable接口

2. 线程的状态

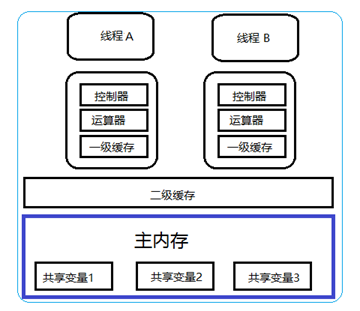
* New
* Runnable
* Blocked
* Waiting
* Timed-Waiting
* Terminated

3. Synchronized和Volatile

**执行控制：**控制代码执行（顺序）及是否可以并发执行。

**内存可见：**控制线程执行结果在内存中对其它线程的可见性。根据Java内存模型的实现，线程在具体执行时，会先拷贝主存数据到线程本地（CPU缓存），操作完成后再把结果从线程本地刷到主存。

* Synchronized关键字解决的是**执行控制**的问题，它会阻止其它线程获取当前对象的监控锁，这样就使得当前对象中被synchronized关键字保护的代码块无法被其它线程访问，也就无法并发执行。更重要的是，synchronized还会**创建一个内存屏障**，内存屏障指令保证了所有CPU操作结果都会直接刷到主存中，从而保证了操作的内存可见性，同时也使得先获得这个锁的线程的所有操作，都happens-before于随后获得这个锁的线程的操作。
* Volatile关键字解决的是**内存可见**性的问题，会使得所有对volatile变量的读写都会直接刷到主存，即保证了变量的可见性。这样就能满足一些对变量可见性有要求而对读取顺序没有要求的需求。



4. Synchronized和Volatile的区别

* volatile本质是在告诉JVM当前变量在寄存器（工作内存）中的值是不确定的，需要从主存中读取； synchronized则是锁定当前变量，只有当前线程可以访问该变量，其他线程被阻塞住
* volatile仅能使用在变量级别；synchronized则可以使用在变量、方法、和类级别的
* volatile仅能实现变量的修改可见性，不能保证原子性；而synchronized则可以保证变量的修改可见性和原子性
* volatile不会造成线程的阻塞；synchronized可能会造成线程的阻塞
* volatile标记的变量不会被编译器优化；synchronized标记的变量可以被编译器优化