1. Volatile关键字
2. volatile的作用

volatile关键字的作用是强制从公共堆栈中取得变量的值，而不是从线程私有堆栈中取得变量的值。当一个线程中新建一个变量时，该值在该线程的私有堆栈中存在，也在公共堆栈中存在。volatile关键字的作用就是同步这两个堆栈中的值，保证值会不断刷新。

当JVM设置为 Server服务器的环境时，为了线程运行的效率，线程就会只在私有堆栈中获取变量的值，如果该变量并没有用volatile关键字来修饰的话，那么线程获取到的值域公共堆栈中是不同的。

1. volatile与synchronized对比
2. 关键字volatile是线程同步的轻量级实现，所以volatile的性能比synchronized要好。且volatile只能修饰变量，而synchronized可以修饰方法和代码块。
3. 多线程访问volatile不会发生阻塞，而synchronized会发生阻塞。
4. volatile可以保证数据的可见性，但是不能保证数据的原子性，即数据是不同步的，非线程安全的。而synchronized可以保证原子性，也可以间接保证数据的可见性，通过私有内存和公共内存实现。
5. volatile关键字的使用场景

关键字volatile的主要使用场合是在多个线程中可以感知到共享的实例变量被修改了，并且可以获得该变量最新的值。注意，仅仅是读取哦，修改变量操作是非线程安全的，需要使用synchronized关键字进行同步操作。

因此，可以看到volatile关键字的作用只是将线程的私有堆栈数据同步到公共堆栈以及从公共堆栈中读取数据。修改变量的操作会引发线程的安全问题，因为在内存中，当一个线程将变量从公共堆栈复制加载到线程的私有堆栈中，并在私有堆栈中进行修改时，如果公共堆栈中的变量在此时也被其他线程修改了，由于私有堆栈中的变量已经加载不会产生对应的变化，也就是说此时的私有堆栈与公共堆栈的变量值已经是不同步的了，这就出现了非线程安全问题。

JVM只能保证volatile修饰的变量从公共堆栈加载到线程的私有堆栈的值是最新的，但是无法保证多线对该变量的修改是同步的。因此，当多个线程访问同一个实例变量时需要加锁同步。

1. 原子类

除了使用synchronized关键字实现同步外，还可以使用以Atomic开头的原子类实现同步。原子操作是不能分割的整体，没有其他线程能够中断或检查正在原子操作的变量。一个原子类型就是一个原子操作可用的类型，它可以在没有锁的情况下做到线程安全。不过，原子类也不是完全安全的，即使使用的是原子类，但是在代码具有逻辑性的情况下，不同方法之间的调用并不是同步的，需要手动设置synchronized关键字进行同步处理。