1：单例模式的双重检查锁为何要做2次非空检测？

**Tom：**

我们之前在代码中调试过，我们假设两个线程如果同时进入到方法中，就意味着这两个线程可以同时获得一个判断，就会得到一个结果，即初始化实例可能是空的，这时候，我们就要做一个锁，进入到Synchrionized关键字里面，在里面如果不做判断的话，那么第一次会创建实例，第二个线程执行的时候，第二个判断如果不做非空判断，就会执行一次重复创建，这时候就会导致实例创建两次，从这个结果来看，我们可以断定，第一个判断主要是控制性能优化的作用，当两个线程同时进来时，如果是非空则直接返回，如果不是非空，则会进入锁，锁里面的非空主要是为了防止对象的重复创建，如果不加这个判断，任何线程进来都可能会直接执行创建，如果做了判断，第一个线程创建好了，第二个线程就不会再创建了；

我们得到结果：第一个判断做性能优化，第二个判断是做逻辑控制的。

2：单例模式双重检查锁中的Synchronized为什么锁住的是class，可以是int或者Object吗？

**TOM：**

Class在JVM层面是只load一次的，也就是说有缓存，这个Class的缓存有个全局锁，叫做全局共享锁，一旦锁住这个Class，其他线程再次操作这个Class的相关操作时就会受到保护，也就是说锁住Class后，基于这个对象的所有操作都是线程安全的，如果锁住其他的Class，作用域就有限了，一旦方法中涉及到其他的操作，其他的变量或其他访问的时候，就会导致其他的操作不受线程的保护，因此Synchrionized关键字只能锁住Class，不能锁住Class以外的东西，保证它的作用域是全局的。