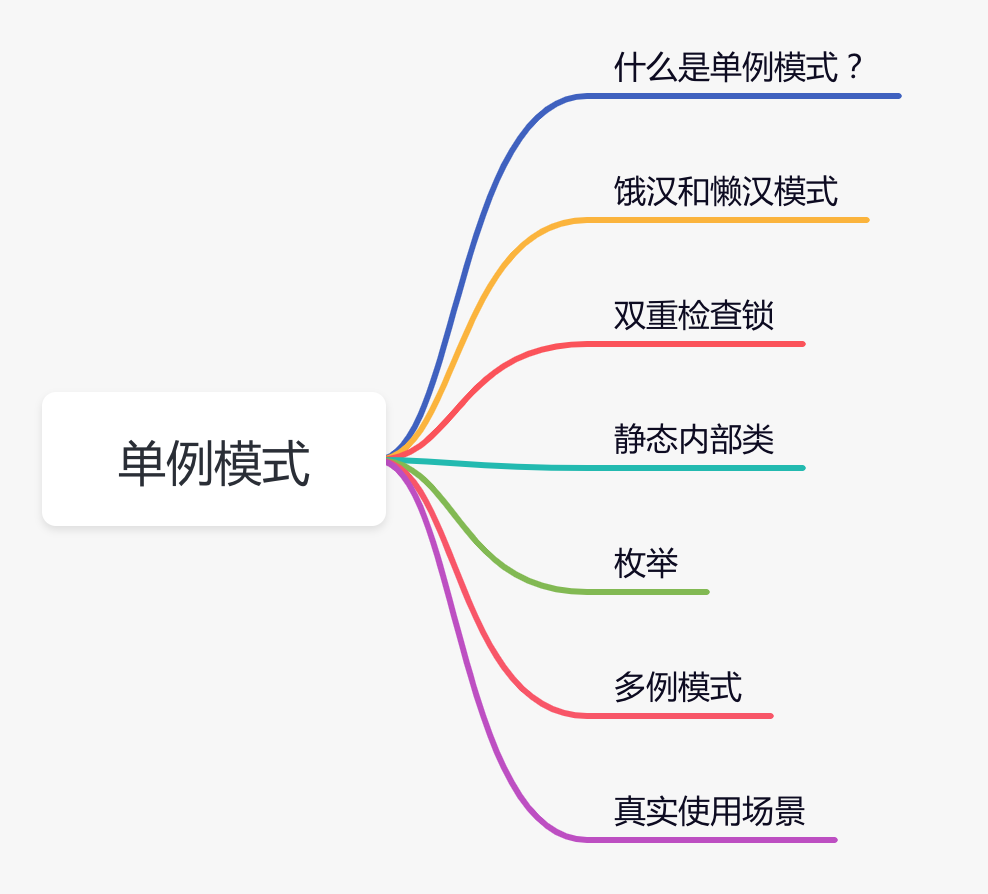
<https://mp.weixin.qq.com/s/ztTWmTWyKr5Sni75LT8o7Q>



**1 什么是单例模式？**

单例模式是一种非常常用的软件设计模式，它定义是单例对象的类只能允许一个实例存在。

该类负责创建自己的对象，同时确保只有一个对象被创建。一般常用在工具类的实现或创建对象需要消耗资源的业务场景。

单例模式的特点：

* 类构造器私有
* 持有自己类的引用
* 对外提供获取实例的静态方法

我们先用一个简单示例了解一下单例模式的用法。

## 2 饿汉和懒汉模式

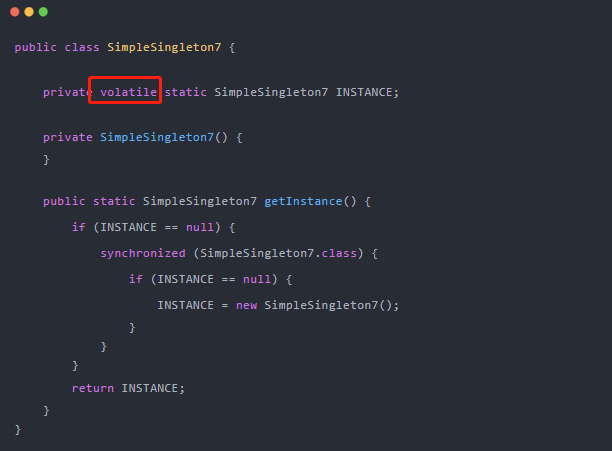
## 3 双重检查锁

双重检查锁顾名思义会检查两次：在加锁之前检查一次是否为空，加锁之后再检查一次是否为空。

那么，它是如何实现单例的呢？



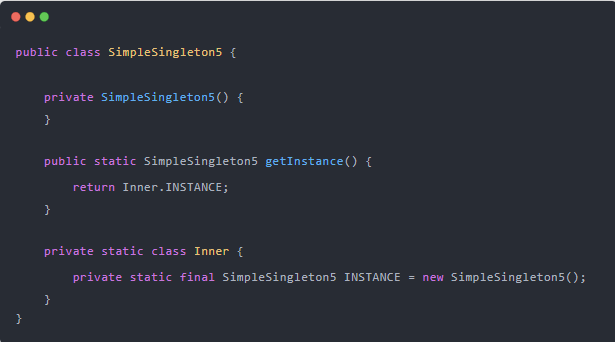
防止指令重排序：



volatile关键字可以保证多个线程的可见性，但是不能保证原子性。同时它也能禁止指令重排。

双重检查锁的机制既保证了线程安全，又比直接上锁提高了执行效率，还节省了内存空间。

## 4 静态内部类



我们看到在SimpleSingleton5类中定义了一个静态的内部类Inner。在SimpleSingleton5类的getInstance方法中，返回的是内部类Inner的实例INSTANCE对象。

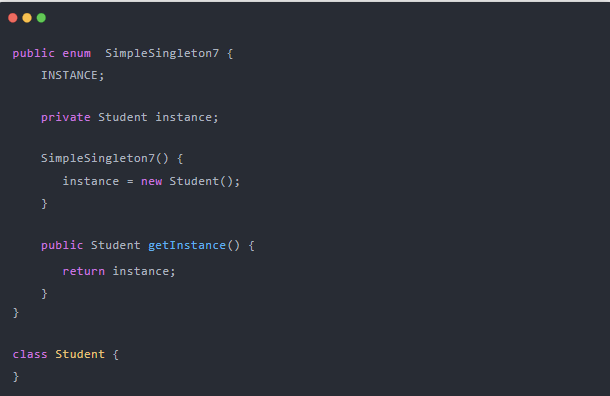
只有在程序第一次调用getInstance方法时，虚拟机才加载Inner并实例化INSTANCE对象。

java内部机制保证了，只有一个线程可以获得对象锁，其他的线程必须等待，保证对象的唯一性。

反射漏洞，在构造方法中进行处理；

反序列化漏洞，重写readResolve方法

## 5 枚举

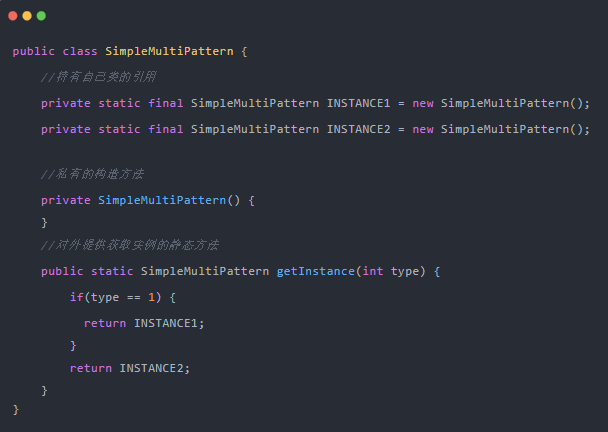


## 6 多例模式

我们之前聊过的单例模式，都只会产生一个实例。但它其实还有一个变种，也就是我们接下来要聊的：多例模式。

多例模式顾名思义，它允许创建多个实例。但它的初衷是为了控制实例的个数，其他的跟单例模式差不多。

具体实现代码如下：



有些朋友可能会说：既然多例模式也是为了控制实例数量，那我们常见的池技术，比如：数据库连接池，是不是通过多例模式实现的？

答：不，它是通过享元模式实现的。

那么，多例模式和享元模式有什么区别？

* 多例模式：跟单例模式一样，纯粹是为了控制实例数量，使用这种模式的类，通常是作为程序某个模块的入口。
* 享元模式：它的侧重点是对象之间的衔接。它把动态的、会变化的状态剥离出来，共享不变的东西。