**单例模式**

单例模式（Singleton Pattern）是指确保一个类在任何情况下都绝对只有一个实例，并

提供一个全局访问点

**饿汉**：饿汉式单例是在类加载的时候就立即初始化，并且创建单例对象。绝对线程安全，在线

程还没出现以前就是实例化了，不可能存在访问安全问题。

优点：没有加任何的锁、执行效率比较高，在用户体验上来说，比懒汉式更好。

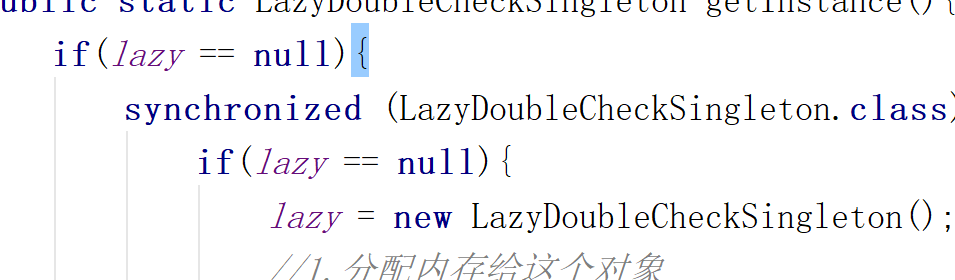
缺点：类加载的时候就初始化，不管用与不用都占着空间，浪费了内存

2种写法：典型写法和静态代码块写法

**懒汉**：在懒汉模式下，虽然是在用户调用再加载，**但是多线程下，也会出现**

**多次实例化影响线程安全**，所以我们需要加上synconized来加锁控制，但是可能会出现效率降低,

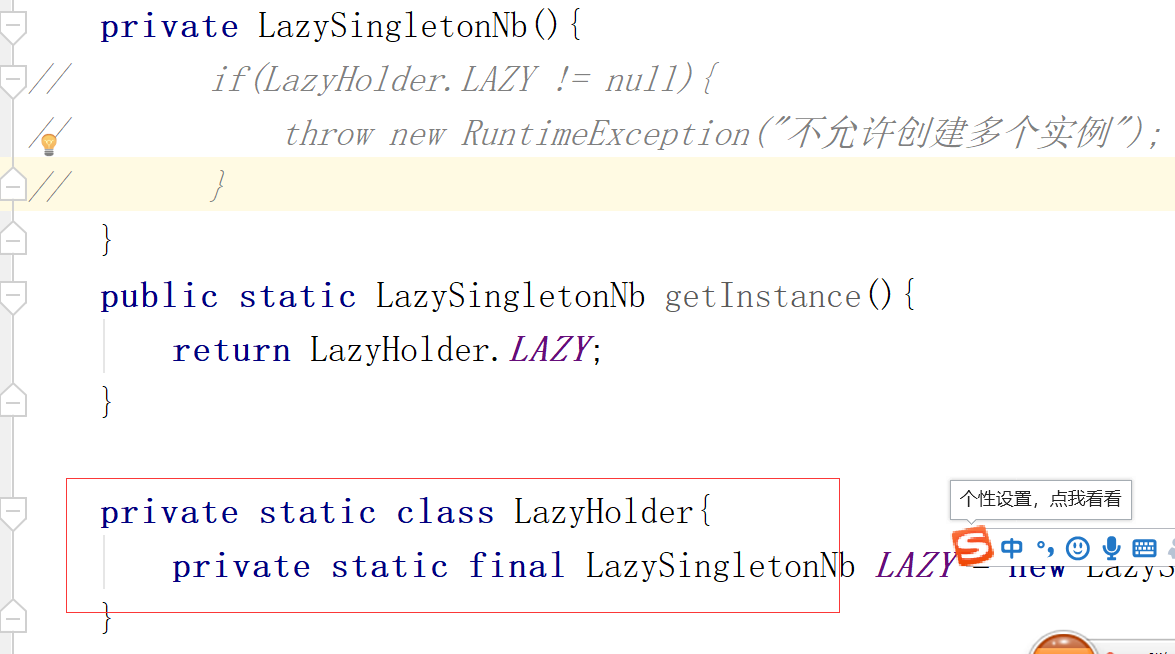
所以采用双重锁来解决



这种方式也是许多源码的写法，这种2种判断有各自的作用，称之为双重锁。

不过在懒汉下还有一种方式可以实现，**采用静态内部类的方式,内部类一定是要在方**

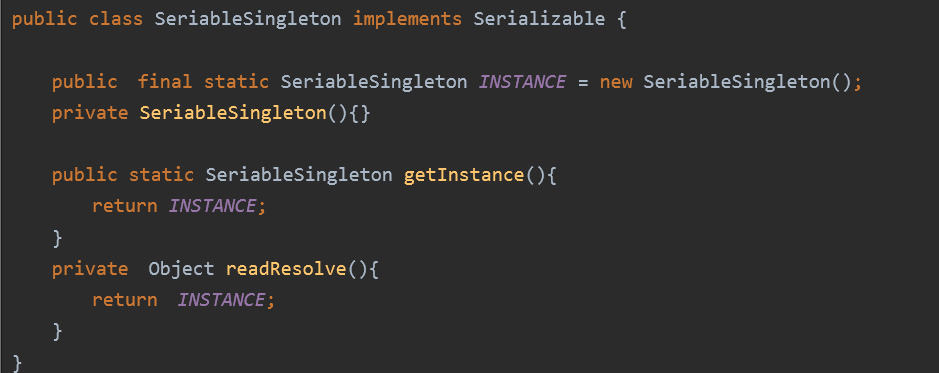
**法调用之前初始化，巧妙地避免了线程安全问题**



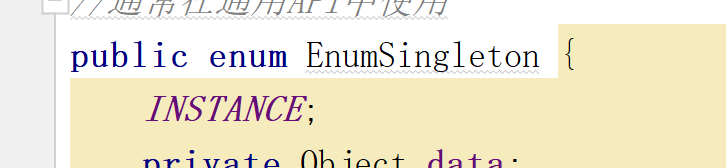
**但是很有可能反射导致实例化不同,**所以需要在private方法加入判断，抛出异常即可解决。

最后还有一个会影响单例，**就是序列化。**

**其实很简单**，**只需要增加readResolve()方法即可**。来看优化代码：



**枚举单例：**

****

其实就是正常的枚举,但是jdk自行优化，反射调用会报错而序列化也不会导致多次实例化.

**ThreadLocal线程单例**

ThreadLocal 不能保证其

创建的对象是全局唯一，但是能保证在单个线程中是唯一的，天生的线程安全

