一个类 T 什么时候被创建

1.T是一个类，T类型的实例被创建

2.T是一个类，T类中声明的一个静态方法被调用

3.T是一个类，T中声明的一个静态字段被赋值

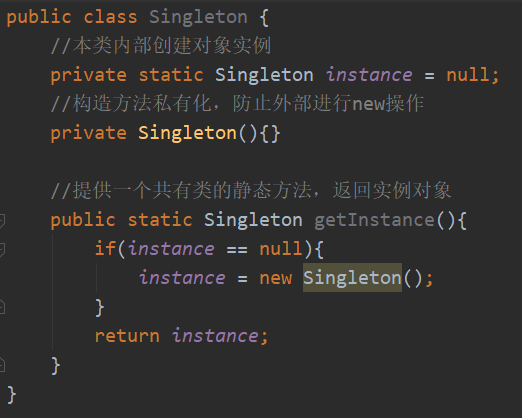
4.T是一个类，T中声明的一个静态字段被使用，并且该字段不是一个常量

5.？T是一个顶级类，而且一个断言语句嵌套在T内部被执行

单例模式的五种分类

1. 懒汉式

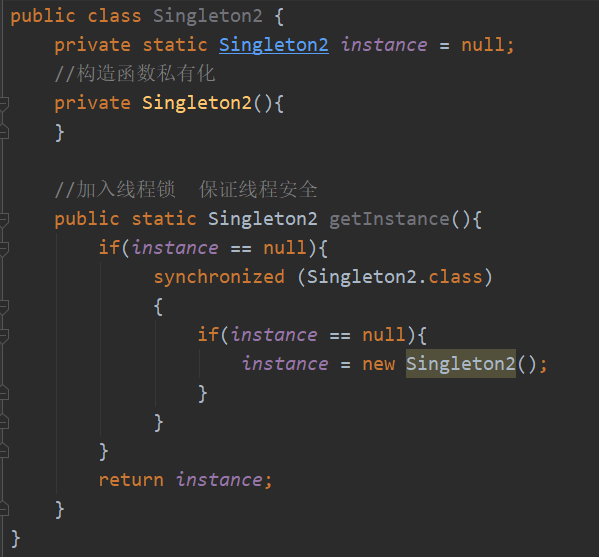
(该类将构造方法私有化，所以不能通过new方法创建，所以只能通过调用其内部静态方法来创建该类，所以懒汉模式就是一开始不加载资源，一直 等，等到第一次使用时才加载，加载完后就一直在。)



1. 懒汉式+同步锁

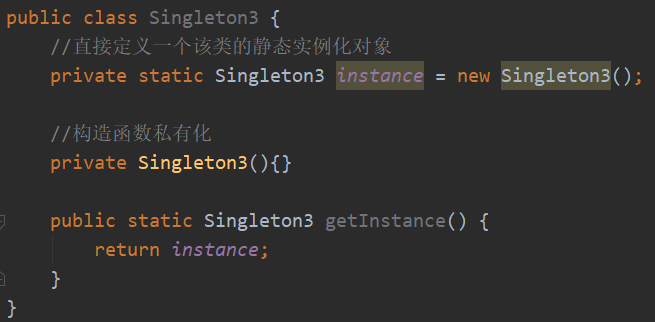
基本原理同懒汉式，就是加入了同步锁并两次空值判断来保证类被创建时的线程安全

Synchronized(Singleton2.Class),将Singleton2这个类上锁，保证每次只能有一个线程能创建该类的实例化对象



1. 饿汉式

不管有没有用到，直接加载资源，用空间换区时间



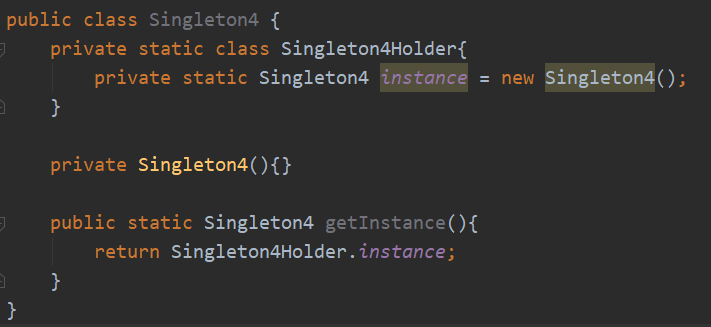
1. **内部类模式**

在类中定义一个静态内部私有类

在开头提到，一个类中的静态变量被实例化的时候，该类才会被创建，并且静态域只会被加载一次(Singleton4Holder这个类和其中的静态变量instance)。并且在创建的时候由jvm来保证线程安全

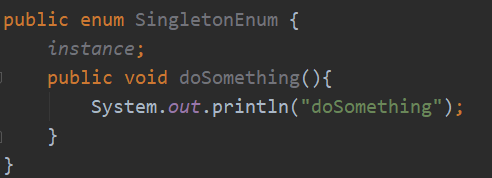
别的大佬对其的解释是

当getInstance方法第一次被调用的时候,它第一次读取SingletonHolder.instance，导致SingletonHolder类得到初始化；而这个类在装载并被初始化的时候，会初始化它的静态域，从而创建Singleton的实例，由于是静态的域，因此只会在虚拟机装载类的时候初始化一次，并由虚拟机来保证它的线程安全性。

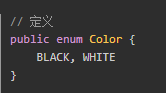


1. **枚举单例模式**

里面写一个instance，就是SingletonEnum类的实例化对象，这是最佳的单例模式



**下面是将枚举类反编译后的结果是个final类参考**



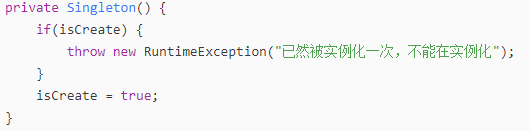


**破坏单例模式的三种方式**

1.反射

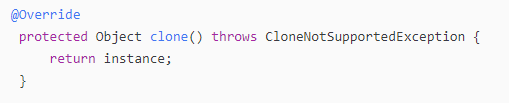
解决方法 定义一个全局变量，当第二次创建的时候抛出异常，这里是isCreate(boolean 类型)





2.克隆

解决方法  重写clone(),直接返回单例对象



3.序列化 重写readResolve,直接返回单例对象

