**对单体模式进行扩展以达到**

**适用多线程的情形**

学号： 2017282160103

姓名： 杨一帆

专业班级： 软件工程8班

授课老师： 胡启平

# 引言

单例模式是一种常见的设计模式。

单例模式确保某个类只有一个实例，而且自行实例化并向整个系统提供这个实例。在计算机系统中，线程池、缓存、日志对象、对话框、打印机、显卡的驱动程序对象常被设计成单例。这些应用都或多或少具有资源管理器的功能。每台计算机可以有若干个打印机，但只能有一个Printer Spooler，以避免两个打印作业同时输出到打印机中。每台计算机可以有若干通信端口，系统应当集中管理这些通信端口，以避免一个通信端口同时被两个请求同时调用。总之，选择单例模式就是为了避免不一致状态，避免政出多头。

单例模式特点：  
　　1、单例类只能有一个实例。  
　　2、单例类必须自己创建自己的唯一实例。  
　　3、单例类必须给所有其他对象提供这一实例。

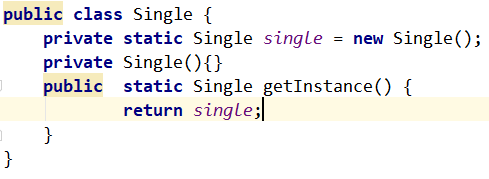
单例模式保证了全局对象的唯一性，比如系统启动读取配置文件就需要单例保证配置的一致性。

# 实现方式

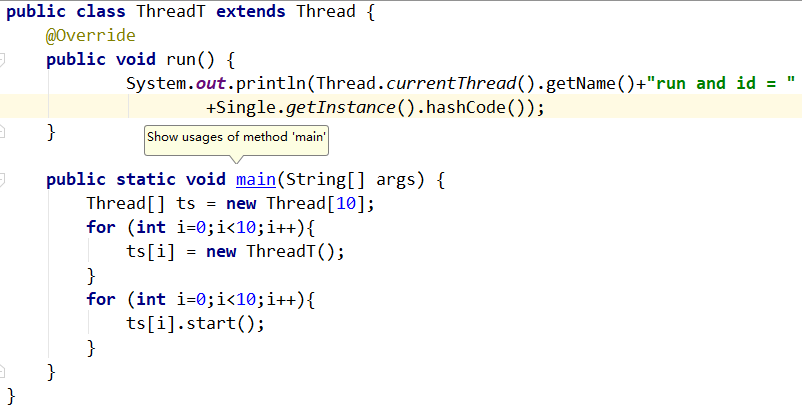
## 2.1饿汉式

本文实例全部通过java编写

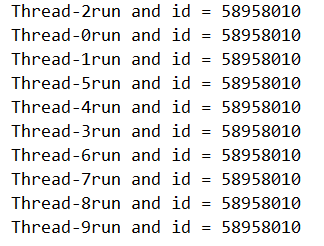
对于单例设计模式的饿汉式实现方式，在方法调用前，实例就已经创建好了。如下方的示例类Single：



对此种实现方法使用多线程调用：

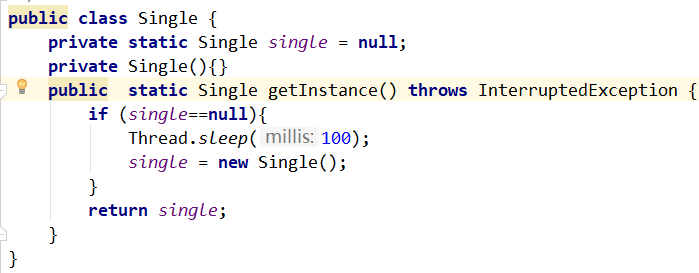


从运行结果可以看出实例变量额hashCode值一致，这说明对象是同一个，饿汉式单例实现了

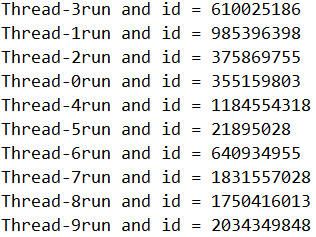


## 2.2懒汉式

懒汉式单例是指在方法调用获取实例时才创建实例，因为相对饿汉式显得“不急迫”，所以被叫做“懒汉模式”。下面是实现代码：



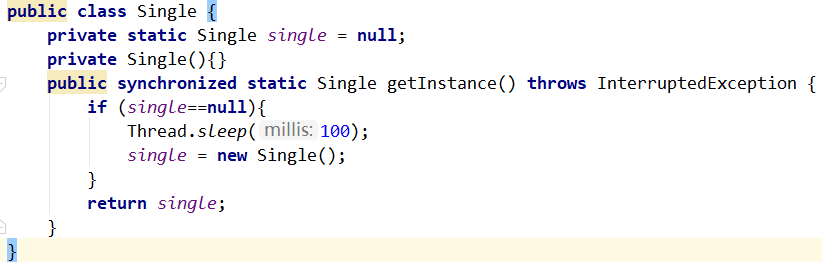
然而我们在运行可以明显发现，在多线程调用时懒汉式会出错，不能保证单例：



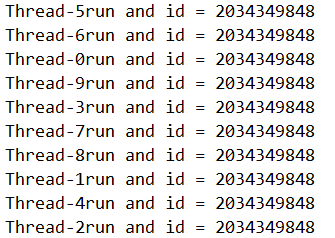
## 2.3解决方法

1、为了实现同步，我们在Single类的getInstance()方法里加入synchronized关键字，这样加锁后，在同一时间就只允许一个线程进入该方法，就不会有创建多个实例的错误出现了。

改变后的Single类如下：

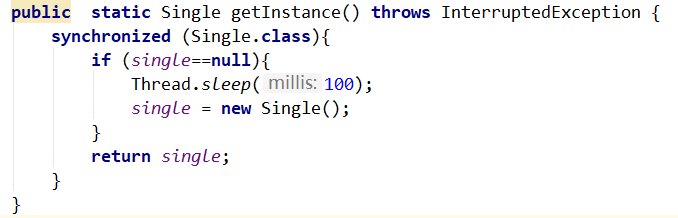


运行结果可以看出实例的hashcode没有变化，是成功的：

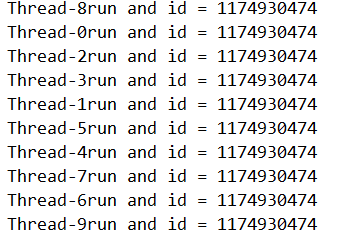


2、同理，我们也可以将getInstance()中的代码用同步代码块包起来，同样可以实现多线程下的单例模式安全调用。

代码如下：



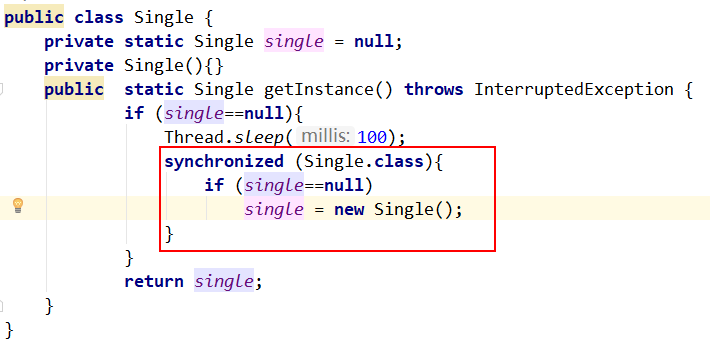
运行结果：



但是，以上两种解决办法，本质上都是通过同步方法的形式确保多线程下单例模式的实现，这种实现方法会使得程序的运行效率很低。

3、只将getInstance()方法中特定的需要进行同步的部分放入同步代码块。

改进后的代码如下：



注意，在将single = new Single()语句放入同步代码块后，务必在同步代码块中再进行一次关于single==null的判断，因为在原程序中我们为了模拟创建实例前的一些准备工作和耗时工作，在代码中加入了一行Thread.sleep(100)，如果不进行二次判断，则有可能在后续线程等待的过程中前边的线程已经创建实例完毕，这样后续线程就也可以获得锁，进入同步代码块中，最终导致创建多个实例。

运行结果：

