

软件设计模式实验报告

**2021/2022(2)**



实验题目 多例模式的应用

学生姓名 黄益妙

学生学号 201906062108

学生班级 软件工程1904

任课教师 黄洪

提交日期 2022.4.27

**计算机科学与技术学院、软件学院**

**一、设计问题**

**单例模式的扩展及应用。**

编写一个类LimitInstanceClass，该类最多可以实例化指定个数实例。实例的个数用配置文件InstanceLimit.cfg指定。例如，如果InstanceLimit.cfg的内容为2，则LimitInstanceClass最多可以同时存在2个对象。

LimitInstanceClass的对象有一个整型成员变量id，保存对象的编号；有一个boolean型变量isBusy，如果该变量的值为true，表示该对象正在被使用，否则该对象空闲。

getInstance()方法判断是否存在空闲的对象，存在将该空闲对象的isBusy置为true，并返回该对象；如果不存在空闲对象则返回null。LimitInstanceClass有一个release()方法，该方法将对象的isBusy置为false。

LimitInstanceClass还有一个String类型的成员变量accessMessage，以及一个成员方法writeAccessMessage(String message)，该方法将参数message追加到accessMessage。LimitInstanceClass的printAccessMessage()方法输出accessMessage的内容。

编写一个线程类AccessLimitInstanceClassThread，在其run()方法中获取一个LimitInstanceClass对象，调用获得的对象的writeAccessMessage(String message)将自己的线程名写入accessMessage，随机休眠0-5秒，再调用printAccessMessage()，最后调用release（）方法。

编写一个UseLimitInstanceClass类，在其main方法中实例化10个AccessLimitInstanceClassThread线程对象，并启动各个线程。

设置InstanceLimit.cfg的内容为3，写出你的程序的运行结果。

**二、问题分析与模式选用**

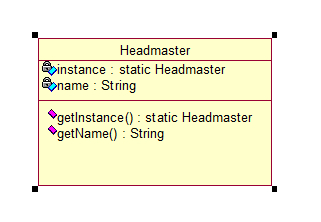
* 多例模式的几个关键点就在于：

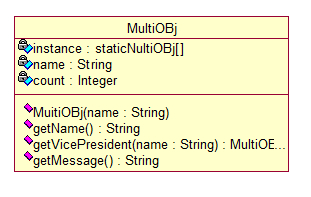
1. 通过实例容器保存容器。

2. 利用私有构造阻止外部构造。

3. 提供getInstantce()方法获取实例。

其UML类图如下：





**三、设计方案**

* 由问题描述可知：

1. 有一个LimitInstanceClass类，该类的成员变量和成员方法如下：

成员变量：

static int i;//用来得到InstanceLimit.cfg指定的对象个数

int id=0;//保存对象的编号

boolean isBusy;//如果该变量的值为true，表示该对象正在被使用，否则该对象空闲

String accessMessage=new String();

成员方法：

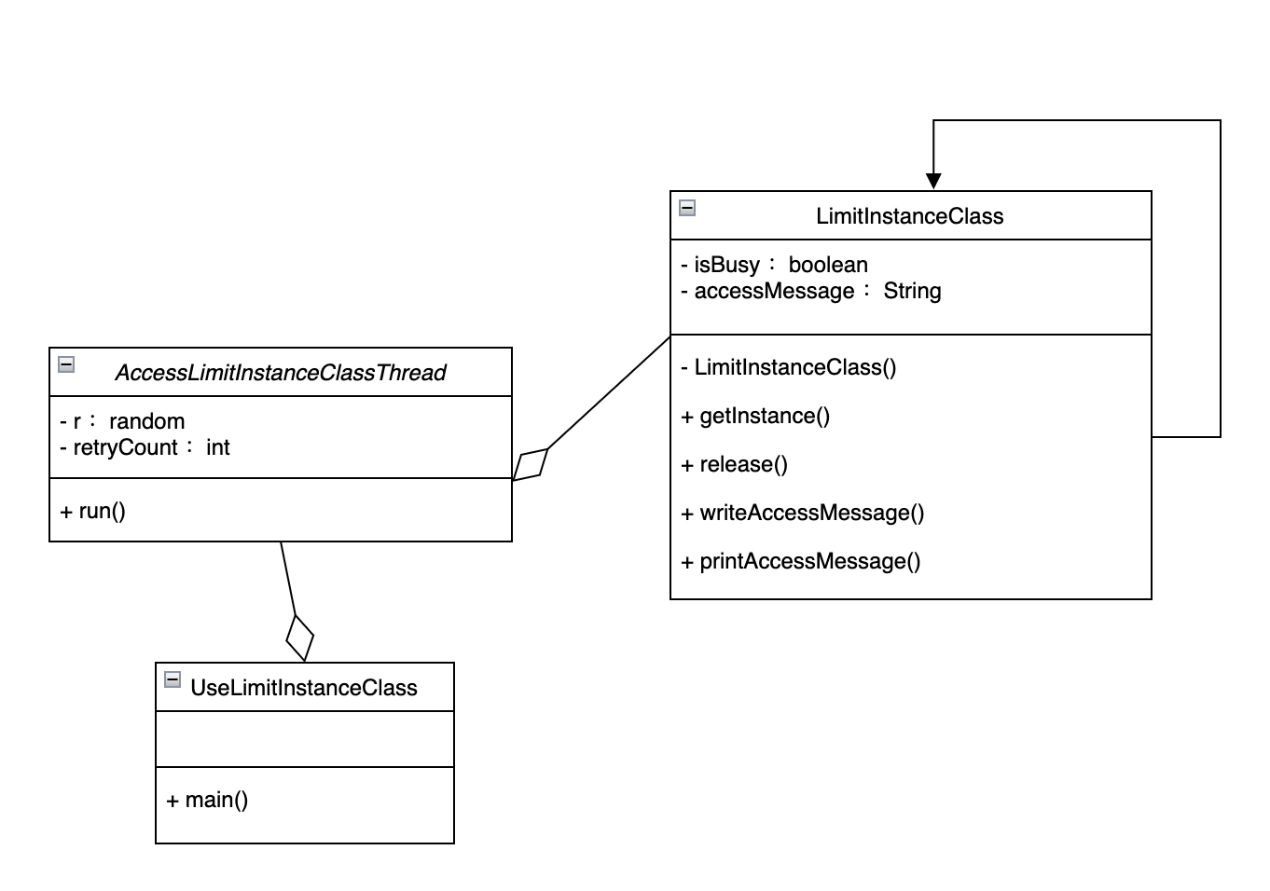
public void release()

public String printAccessMessage()

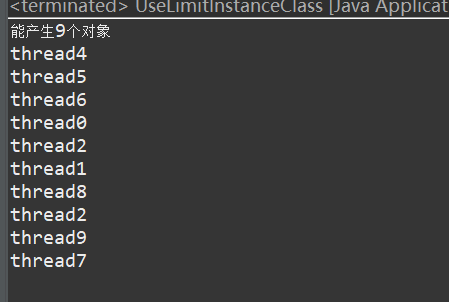
public void writeAccessMessage(String message)

public static LimitInstanceClass getInstance()

1. 有一个线程类AccessLimitInstanceClassThread，该类有一个run()方法获取一个LimitInstanceClass对象，调用获得的对象的writeAccessMessage(String message)将自己的线程名写入accessMessage，随机休眠0-5秒，再调用printAccessMessage()，最后调用release（）方法。
2. 有一个UseLimitInstanceClass类，其main方法中实例化10个AccessLimitInstanceClassThread线程对象，并启动各个线程。



**四、运行结果及效果分析**



* 效果分析：

该程序从InstanceLimit.cfg中读取指定对象个数，然后允许同时存在i个对象，并在UseLimitInstanceClass中for循环实例化了十个线程，并在屏幕上输出实例化的各个线程。

**五、实验体会**

本次实验，我学会了**多例模式的创建和使用，对线程也有了更深的理解。**