单例模式实验报告

3018216095 郭紫珊 软工二班

1 实验要求

实现单例模式。

2 单例模式的介绍

单例模式（Singleton），也叫单子模式，是一种常用的设计模式。在应用这个模式时，单例对象的类必须保证只有一个实例存在。许多时候，整个系统只需要拥有一个的全局对象，这样有利于我们协调系统整体的行为。比如在某个服务器程序中，该服务器的配置信息存放在一个文件中，这些配置数据由一个单例对象统一读取，然后服务进程中的其他对象再通过这个单例对象获取这些配置信息，显然，这种方式简化了在复杂环境下的配置管理。

综上所述，单例模式就是为确保一个类只有一个实例，并为整个系统提供一个全局访问点的一种方法。

3 设计思路和类图

类图：

设计思路;

单例模式的三要素：1.私有的构造方法；2.指向自己实例的私有静态引用；3.以自己实例为返回值的静态的公有方法。

4 单例模式的两种实现

4.1 立即加载（饿汉式）

饿汉式单例在单例类被加载时候，就实例化一个对象并交给自己的引用；

```
public class Singleton1 {

    // 指向自己实例的私有静态引用，主动创建
    private static Singleton1 singleton1 = new Singleton1();

    // 私有的构造方法
    private Singleton1(){}

    // 以自己实例为返回值的静态的公有方法，静态工厂方法
    public static Singleton1 getSingleton1(){
        return singleton1;
    }
}
```

4.2 延迟加载（懒汉式）

懒汉式单例只有在真正使用的时候才会实例化一个对象并交给自己的引用。

```
public class Singleton2 {

    // 指向自己实例的私有静态引用
    private static Singleton2 singleton2;

    // 私有的构造方法
    private Singleton2(){}

    // 以自己实例为返回值的静态的公有方法，静态工厂方法
    public static Singleton2 getSingleton2(){
        // 被动创建，在真正需要使用时才去创建
        if (singleton2 == null) {
            singleton2 = new Singleton2();
        }
        return singleton2;
    }
}
```

总之，从速度和反应时间角度来讲，饿汉式（又称立即加载）要好一些；从资源利用效率上说，懒汉式（又称延迟加载）要好一些。