单例模式

核心作用：

* 保证一个类只有一个实例，并且提供一个访问该实例的全局访问点

常见的几种单例模式实现方式

* 饿汉式（线程安全，调用效率高。但是不能延时加载。）
* 懒汉式（线程安全，调用效率不高。但是可以延时加载）
* 双重检测锁式（由于JVM底层内部模型原因，偶尔会出现问题。不建议使用）
* 静态内部类式（线程安全，调用效率高，可以延时加载）
* 枚举单例（线程安全，调用效率高。不能延时加载）

**饿汉式实现:**

饿汉式单例模式代码中，static变量会在类装载时初始化，此时也不会涉及多个线程对

象访问该对象的问题。虚拟机保证只会装载一次该类，肯定不会发生并发访问的问题，因此可以省略synchronize关键字。

**问题:**如果只是加载该类，而不是要调用getInstance()方法，甚至永远没有调用，则会造成资源浪费。

**懒汉式实现:**

延时加载！懒加载，真正用的时候才加载

**问题：**资源利用效率高了。但是，每次调用getInstance()方法时都要同步，并发效率较低

**静态内部类式（也是一种懒加载方式）：**

要点：

* 外部类没有static属性，则不会像饿汉式那样立即加载对象。
* 只有真正调用getInstance(),才会加载静态内部类。加载类是是线程安全的。Instance是static final类型，保证了内存中只有一个实例存在且只能被赋值一次，从而保证了线程安全性
* 兼备了并发高效调用和延迟加载的优势！