|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **单例模式** | **优点** | **缺点** | **防反射破坏方案** | **防序列化破坏方案** |
| 饿汉模式 | 类初始化的时候就创建实例对象，因此就是线程安全的，不需要手动去实现线程安全。 | 类初始化的时候就创建了实例对象，过早分配了内存，会造成内存浪费。  会有被反射破坏单例和被序列化破坏单例的问题。 | 无参构造器加判断，如果实例不为空则抛出异常。 | 类中增加readResolve方法，返回单例对象。 |
| 懒汉模式 | 每次使用的时候才会去创建实例对象，避免了内存浪费。 | 需要手动实现线程安全。  会有被反射破坏单例和被序列化破坏单例的问题。 | 无参构造器加判断，如果实例不为空则抛出异常（instance在构造方法内赋值）。 | 类中增加readResolve方法，返回单例对象。 |
| 静态内部类模式 | 巧妙的利用了jvm的类加载机制式，内部类只在第一次使用的时候才会去加载和创建实例对象。 | 会有被反射破坏单例和被序列化破坏单例的问题。 | 无参构造器加判断，如果实例不为空则抛出异常。 | 类中增加readResolve方法，返回单例对象。 |
| 枚举模式 | 实现简单，多线程安全 |  |  |  |