1. 什么是单例模式

【只有一个实例】

因程序需要，有时我们只需要某个类同时保留一个对象，不希望有更多对象，此时，我们则应考虑单例模式的设计。

二. 单例模式的特点

1. 单例模式只能有一个实例。

2. 单例类必须创建自己的唯一实例。

3. 单例类必须向其他对象提供这一实例。

三. 单例模式VS静态类

在知道了什么是单例模式后，我想你一定会想到静态类，“既然只使用一个对象，为何不干脆使用静态类？”，这里我会将单例模式和静态类进行一个比较。

1. 单例可以继承和被继承，方法可以被override，而静态方法不可以。

2. 静态方法中产生的对象会在执行后被释放，进而被GC清理，不会一直存在于内存中。

3. 静态类会在第一次运行时初始化，单例模式可以有其他的选择，即可以延迟加载。

4. 基于2， 3条，由于单例对象往往存在于DAO层（例如sessionFactory），如果反复的初始化和释放，则会占用很多资源，而使用单例模式将其常驻于内存可以更加节约资源。

5. 静态方法有更高的访问效率。

6. 单例模式很容易被测试。

几个关于静态类的误解：

误解一：静态方法常驻内存而实例方法不是。

实际上，特殊编写的实例方法可以常驻内存，而静态方法需要不断初始化和释放。

误解二：静态方法在堆(heap)上，实例方法在栈(stack)上。

实际上，都是加载到特殊的不可写的代码内存区域中。

静态类和单例模式情景的选择：

情景一：不需要维持任何状态，仅仅用于全局访问，此时更适合使用静态类。

情景二：需要维持一些特定的状态，此时更适合使用单例模式。

四. 单例模式的实现

1. 懒汉模式

2. 线程安全的懒汉模式

3. 饿汉模式

4. 静态类内部加载

但是，上面提到的所有实现方式都有两个共同的缺点：

* 都需要额外的工作(Serializable、transient、readResolve())来实现序列化，否则每次反序列化一个序列化的对象实例时都会创建一个新的实例。
* 可能会有人使用反射强行调用我们的私有构造器（如果要避免这种情况，可以修改构造器，让它在创建第二个实例的时候抛异常）。

5. 枚举方法（推荐）

最后，不管采取何种方案，请时刻牢记单例的三大要点：

* 线程安全
* 延迟加载
* 序列化与反序列化安全