1. 熟练掌握单例模式的常见写法。

A：饿汉式单例

B：饱汉式单例

C：注册式单例（枚举，注册）

D：ThreadLocal单例

1. 总结每种单例写法的优、缺点。

饿汉式单例

优点：线程安全；。

缺点：在类加载的时候就创建了对象，浪费资源；受到序列化，反射的破坏。

饱汉式单例

优点：在对象调用的时候才去实例化对象，不浪费资源。

缺点：线程不安全；受到序列化，反射的破坏。

注册式单例

优点：把创建好的对象存储在缓冲中，使用方便。

缺点：线程不安全；占用内存空间。

ThreadLocal单例

优点：能保证每个线程中的对象是单例，并且线程安全的，属于伪安全

缺点：

1. 思考破坏单例模式的方式有哪些？并且归纳总结。

破坏单例模式的方式有两种：

1. 反射机制破坏单例：通过Class得到类的私有构造方法进行实例化，这样就会导致不是同一个对象；通过对私有构造函数就行判断，如果对象已经存在就直接抛出异常，来确保单例。
2. 通过序列化与反序列化破坏单例：通过查看源码

objectInputStream.readObject()中

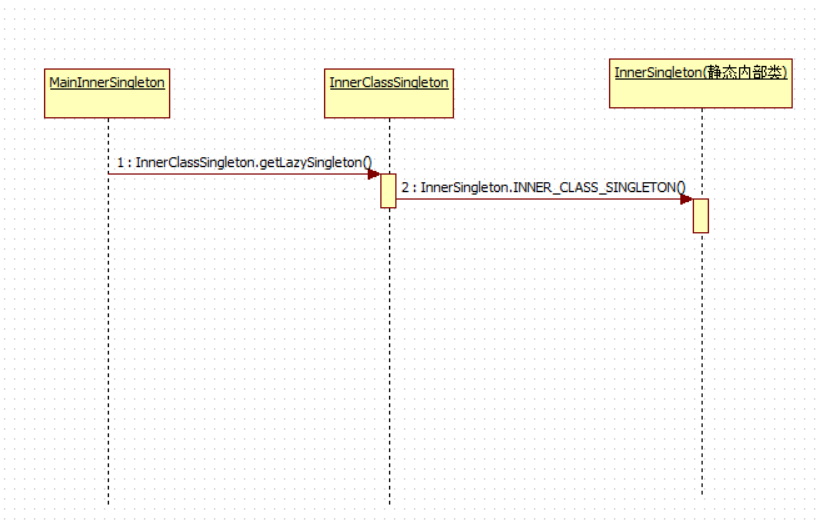
**case *TC\_OBJECT***:  
 **return** checkResolve(readOrdinaryObject(unshared));

到obj = desc.isInstantiable() ? desc.newInstance() : **null**;

得出isInstantiable是true，就会执行newInstance方法实例化对象，但是desc.hasReadResolveMethod()判断是否有这个方法，所有我们可以重写**readResolve 方法无参数**

**readResolveMethod** = *getInheritableMethod*(  
 cl, **"readResolve"**, **null**, Object.**class**);

1. 梳理内部类的执行逻辑，并画出时序图。



在调用外部类的方法getLazySingleton()的时候才去调用静态内部类的

**private static final** InnerClassSingleton ***INNER\_CLASS\_SINGLETON***=**new** InnerClassSingleton();