1. 总结每种单例写法的优、缺点

|  |  |  |
| --- | --- | --- |
| 单例类型 | 优点 | 缺点 |
| 饿汉式单例 | 可以快速创建实例 | 在未使用时就初始化实例，占用内存空间 |
| 懒汉式单例 | 只有在使用的时候，才会创建 | 需要考虑多线程的安全问题，需要进行双重校验保证实例只被创建一次。但是添加锁后，会有性能问题 |
| 内部类方式单例 | 不需要考虑线程安全，没有性能问题 | 可通过反射和反序列化方式暴力破坏单例 |
| 枚举类方式单例（注册式单例） | 既能满足单例要求，又无法被反射和序列化方式破解 | 相对目前最优实现单例方式 |
| 容器注入方式单例（注册式单例） | 作为容器，可存放多种对象的单个实例 | 需要考虑多线程安全问题，添加锁 |

1. 思考破坏单例模式的方式有哪些？并且归纳总结

目前破坏单例模式的方式主要有反射和序列化。

反射：普通java类可以通过反射获取到构造方法，从而调用构造方法创建新对象。

但是枚举类方式单例不能通过发射破坏，因为枚举单例反编译后可以看到，枚举类实际内部没有无参构造方法。另外，如果反射想通过调用有参构造方法，去创建实例时，反射机制的调用源码已经做了判断，判断当前调用类是否是枚举，如果是枚举类，则不允许进行反射调用构造方法创建实例对象。

序列化：序列化方式，当读取文件流生成对象时，如果是object类型，会去检查是否含有构造方法，如果有，会重新newInstance创建新对象。如果是枚举类型，会没有调用new 对象的动作，而是根据枚举名称直接获取枚举值。所以枚举类方式单例不会被反序列化破解，普通单例可以。

3、梳理内部类的执行逻辑，并画出时序图。

