1. 代理Proxy模式

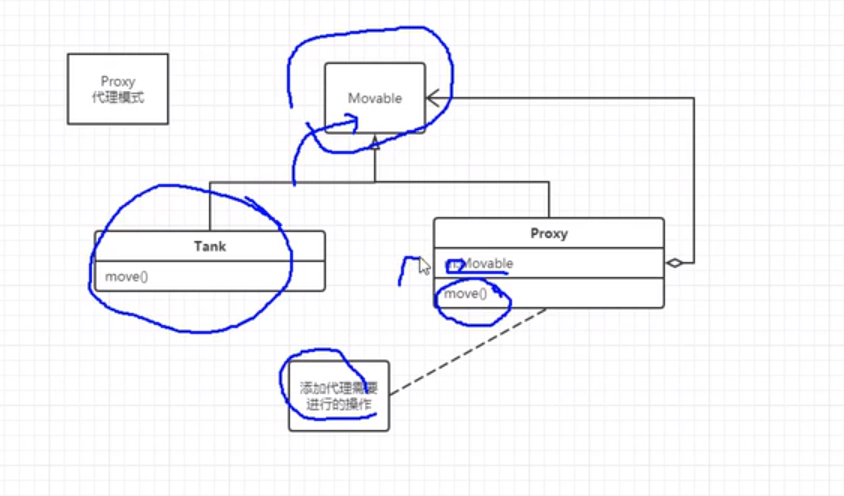
静态代理、动态代理、Spring AOP

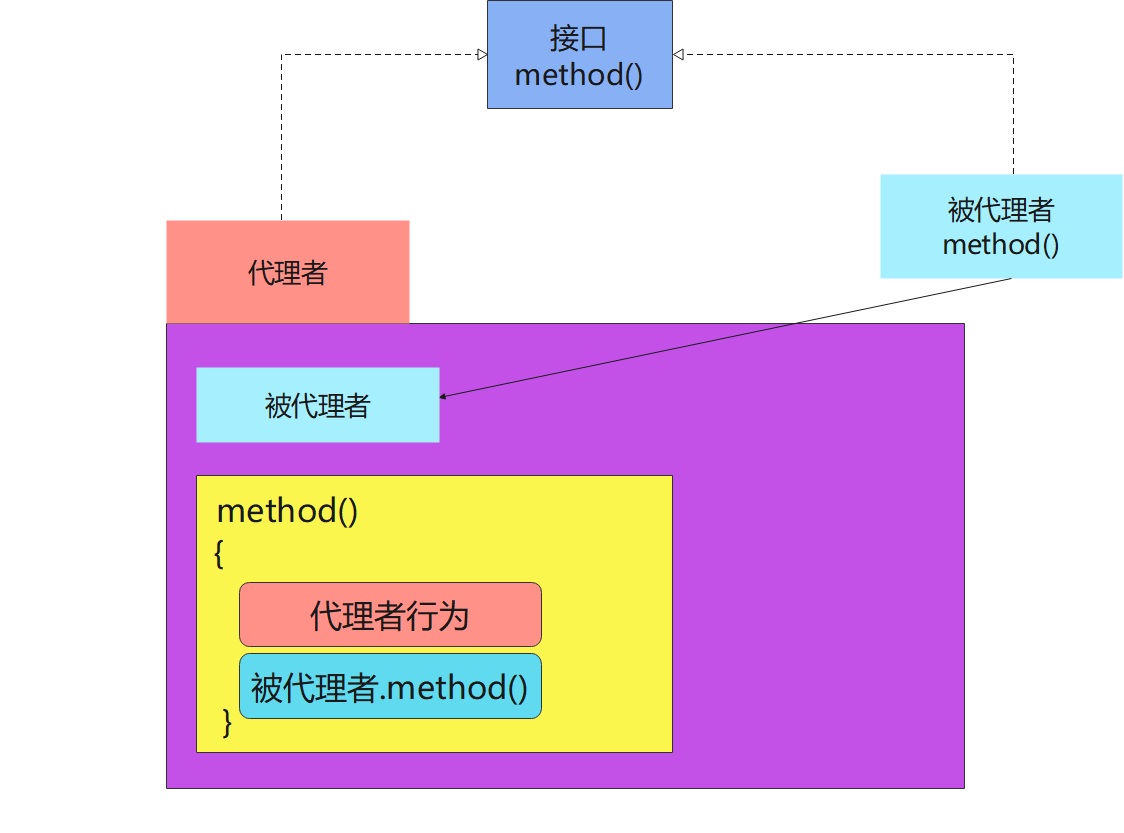
代理者动作：自己的动作（打印log、记录时间等等）+被代理的动作

分离代理行为与被代理对象

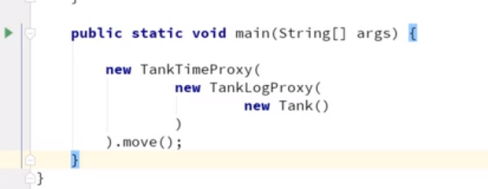
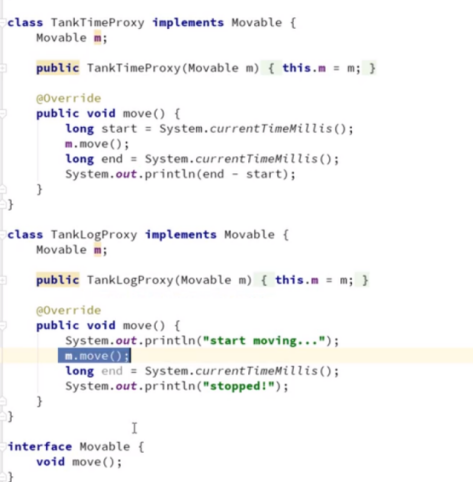
1. 静态代理：通过实现接口，静态生成代理对象（需要手写类）

缺点：只能代理某一类对象，实现某个接口，无法代理所有对象





1. 注册被代理者（构造函数）
2. 方法：代理者行为+被代理者方法

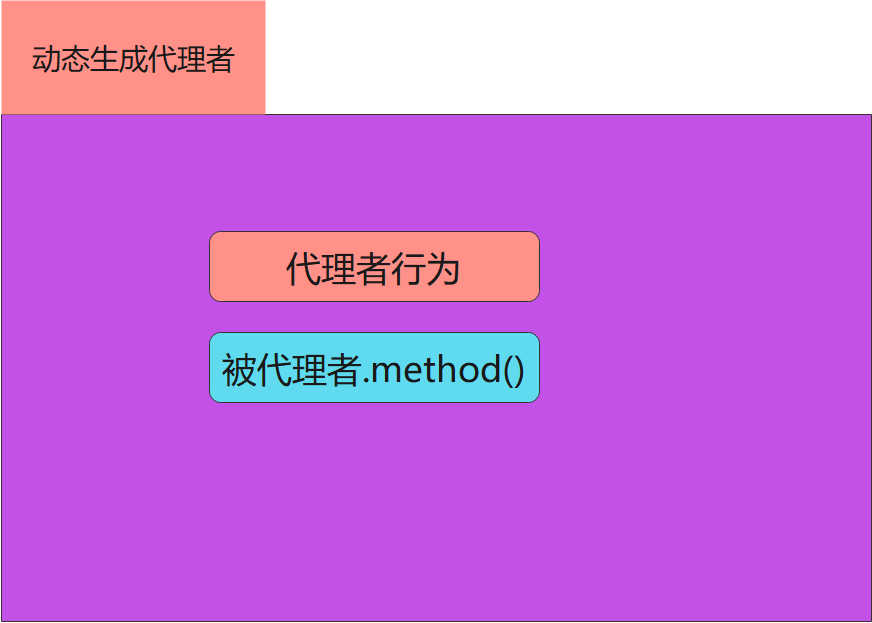


1. 动态代理：动态生成代理对象（不需要手写类，asm动态生成字节码class）

解决静态代理的缺点，只能代理某一类对象。

代理对象不手写，通过反射机制动态生成class，包装被代理者行为

（A）Proxy.proxyNewInstance

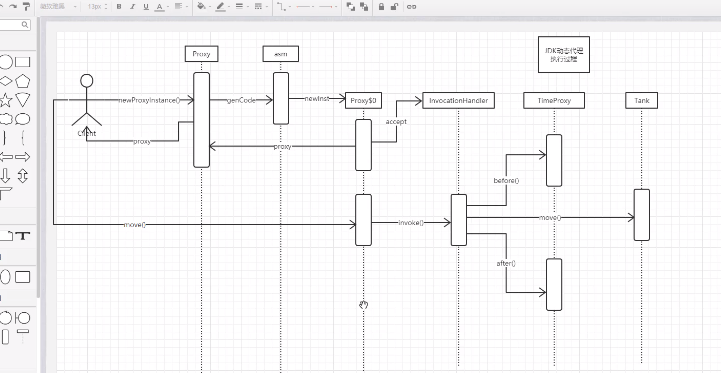


需要：

1. 被代理对象（手动实现）
2. 代理对象（动态生成）：类加载器、实现的接口、代理方法

过程：

1. 通过Proxy.newProxyNewInstance动态生成代理对象，需要传入代理行为proxyMethod
2. 调用代理生成的对象方法，即可调用到proxyMethod



1. Instruments

class文件被加载到内存之前，修改class文件结构

1. Cglib

设置Enhancer的父类为代理者，但是不能代理final类

