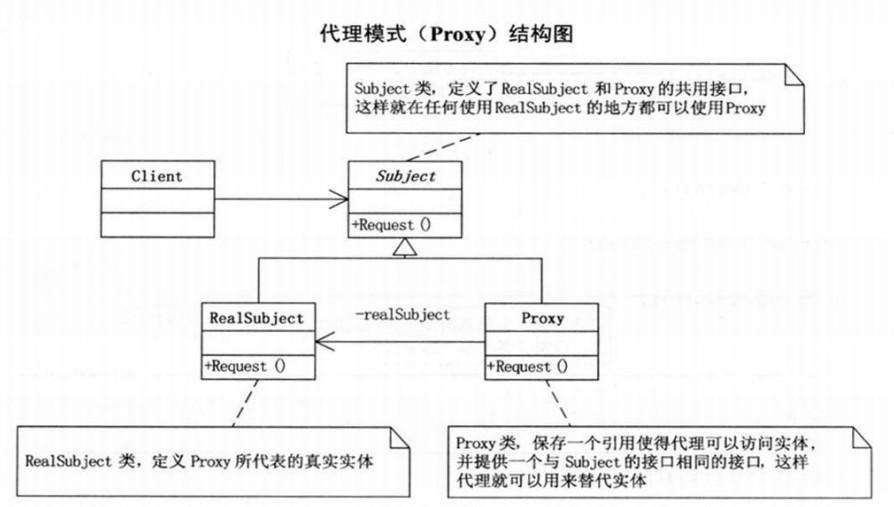
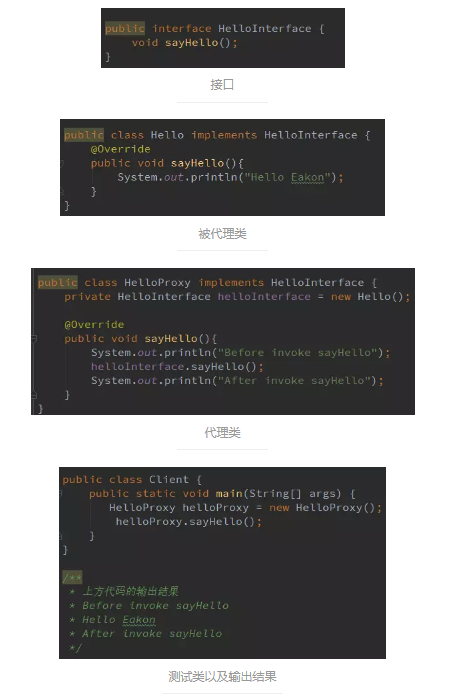
## 代理模式

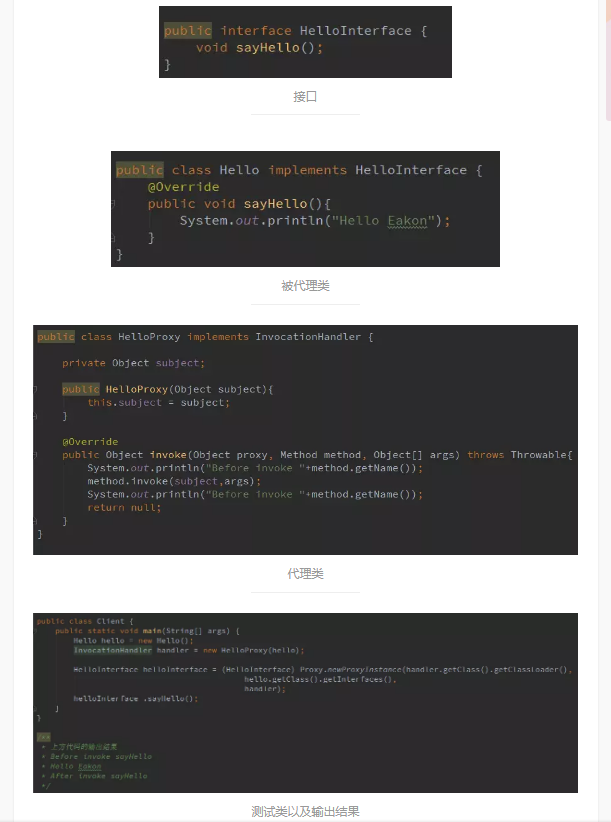
JDK静态代理、JDK动态代理以及CGLIB动态代理



JDK静态代理



我们可以看出，使用JDK静态代理很容易就完成了对一个类的代理操作。但是JDK静态代理的缺点也暴露了出来：由于代理只能为一个类服务，如果需要代理的类很多，那么就需要编写大量的代理类，比较繁琐。



JDK动态代理其实也是基本接口实现的。因为通过接口指向实现类实例的多态方式，可以有效地将具体实现与调用解耦，便于后期的修改和维护。更具源码可知设置系统属性sun.misc.ProxyGenerator.saveGeneratedFiles为ture时可以打印出动态生成的代理类如public final class $Proxy0 extends Proxy implements Student {...}，可知jdk动态代理都继承了Proxy这个类，所以必须是接口。

CGLIB动态代理

CGLIB底层：使用字节码处理框架ASM，来转换字节码并生成新的类。

应用

动态代理十分广泛如mybatis mapper代理、spring aop原理、事务管理原理都是动态代理实现的



每个用@Transactional注解的service层的类实际上运行时的实例都是代理类，当类.方法时，此方法是代理方法会从上面invoke进入，而在方法里调用方法则是普通方法。