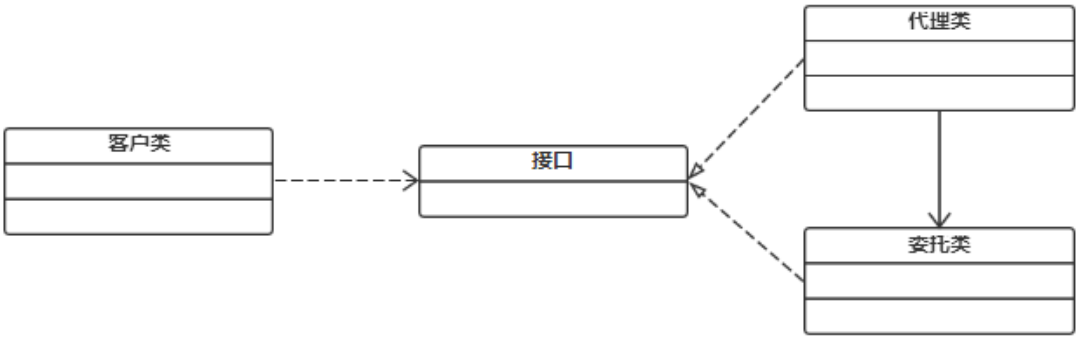
代理模式

代理模式给某一个对象提供一个代理对象，并由代理对象控制对原对象的引用。通俗的来讲代理模式就是我们生活中常见的中介。
比如我们要买一辆二手车,虽然我们可以自己去找哪里有车,但是还有一系列指过户流程,这个流程花费时间太多了,但是你的时间非常值钱

走错片场了,这个时候你就要找二手车代理帮忙,鉴于瓜子是我们部门的,我们先屏蔽优信二手车,选择瓜子二手车作为我们的代理。
瓜子二手车来给我找车源，帮我办理车辆过户流程，我只是负责选择自己喜欢的车，然后付钱就可以了



使用代理模式的好处是啥呢?
设计模式有个原则,就是对拓展是开放的,对修改是关闭的.我们可以给代理类增加一些方法逻辑功能,而不修改委托类,这符合代码的开闭原则.代理类
不真正的实现一个服务,而是调用一个真正的委托类,假如我们想要加入一些计算方法执行时间这样那样的东西,就不要放在委托类里面,直接让代理类实现这个逻辑就行了.

代理从创建的时间来区分,可以分为静态代理和动态代理。

静态代理

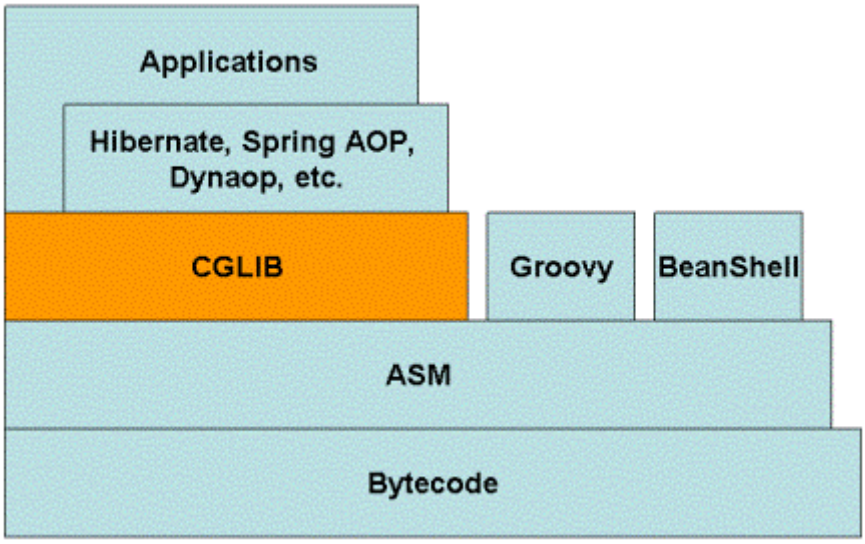
静态代理为程序运行前就已经编译好了class文件

动态代理

程序运行中动态生成class文件为动态代理

- jdk动态代理

- cglib动态代理



  CGLIB底层使用了ASM（一个短小精悍的字节码操作框架）来操作字节码生成新的类。除了CGLIB库外，脚本语言（如Groovy何BeanShell）也使用ASM生成字节码。ASM使用类似SAX的解析器来实现高性能。我们不鼓励直接使用ASM，因为它需要对Java字节码的格式足够的了解

Enhancer创建一个被代理对象的子类并且拦截所有的方法调用（包括从Object中继承的toString和hashCode方法）。
Enhancer不能够拦截final方法，例如Object.getClass()方法，这是由于Java final方法语义决定的。基于同样的道理，Enhancer也不能对fianl类进行代理操作。