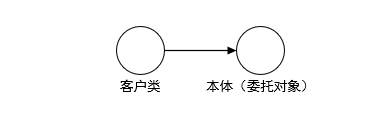
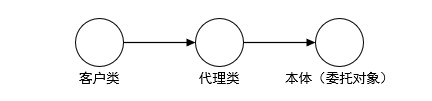
1. 定义

代理模式给某对象提供一个代理对象，由代理对象来控制对原对象的引用。比如火车票代购，代办保险等。

不使用代理



使用代理

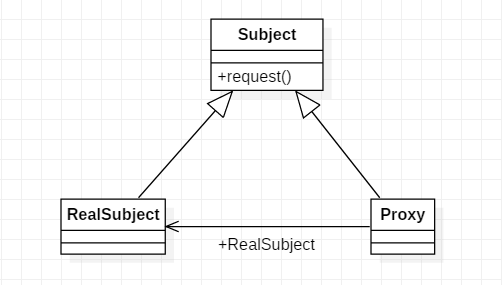


使用场景:客户类不想或不能引用委托对象，这时使用代理类充当中介作用，这种情况下代理类和客户类实现的是同一个接口。或者我们想增强委托类也可以用代理模式。

1. 组成角色

代理一般分两种，静态代理和动态代理(jdk动态代理和cglib动态代理)

* 1. 静态代理



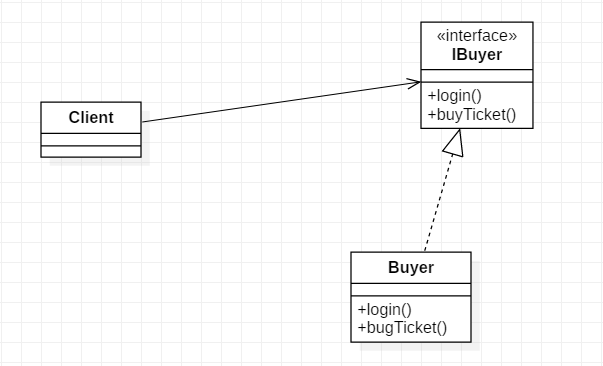
\* 抽象主题角色（subject）:抽象主题角色一般是一个抽象类或接口，用来定义被委托类即真实2的业务处理类和代理类的一些通用操作；

\* 具体的主题角色（RealSubject）: 该类实现Subject,s是真实被委托的类，即具体的业务逻辑真正的执行者；

\* 代理类或委托类（proxy）:该类同样实现subject，再客户类和本地之间充当中介作用，将客户端的业务操作委托给RealSubject执行，并在执行前后做一些预处理或者善后工作。

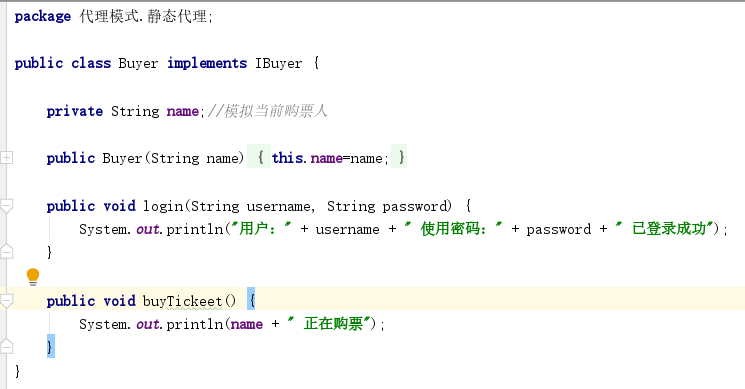
* 1. 实例

以购票为例,定义一个购票接口IBuyer,然后定义一个具体的实现类Buyer

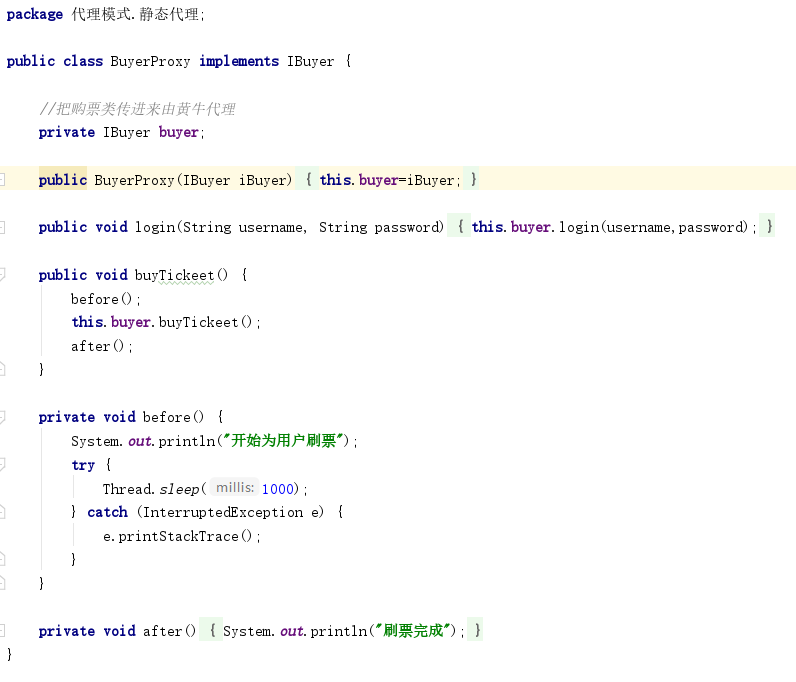


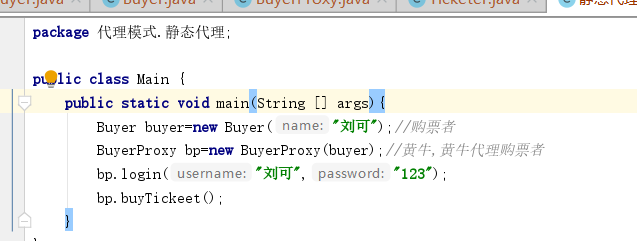


购票的具体实现类



代理类



测试类  


1. jdk动态代理



1. cglib代理

客户类



测试类(引入spring-core的jar包)

