

# 代理模式

代理模式是指，为其他对象提供一种代理以控制对这个对象的访问。在某些情况下，一个对象不适合或者不能直接引用另一个对象，而代理对象可以在客户类和目标对象之间起到中介的作用。

百度百科《代理模式》

换句话说，使用代理对象，是为了在不修改目标对象的基础上，增强主业务逻辑。

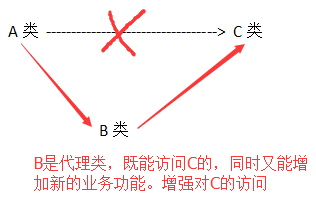
客户类真正的想要访问的对象是目标对象，但客户类真正可以访问的对象是代理对象。客户类对目标对象的访问是通过访问代理对象来实现的。当然，代理类与目标类要实现同一个接口。

例如： 有A，B，C三个类， A原来可以调用C类的方法， 现在因为某种原因C类不允许A类调用其方法，但B类可以调用C类的方法。A类通过B类调用C类的方法。这里B是C的代理。 A通过代理B访问C.

原来的访问关系：



通过代理的访问关系：



Window系统的快捷方式也是一种代理模式。快捷方式代理的是真实的程序，双击快捷方式是启动它代表的程序。

## 代理模式作用

##### 控制访问

##### 增强功能

## 代理模式分类

可以将代理分为两类：静态代理与动态代理

## 代理的实现方式

**静态代理和动态代理**

## 需求

需求：用户需要购买u盘， u盘厂家不单独接待零散购买，厂家规定一次最少购买1000个以上，用户可以通过淘宝的代理商，或者微商哪里进行购买。

淘宝上的商品，微商都是 u 盘工厂的代理商， 他们代理对u盘的销售业务。

用户购买-------代理商（淘宝，微商）----- u 厂家（金士顿，闪迪等不同的厂家）

设计这个业务需要的类：

1. 商家和厂家都是提供 sell 购买u盘的方法。定义购买u盘的接口 UsbSell
2. 金士顿（King）对购买1千以上的价格是 85, 3千以上是80, 5千以上是75。 单个120元。定义UsbKingFactory类，实现UsbSell
3. 闪迪（San）对购买1千以上的价格是 82, 3千以上是78, 5千以上是72。 单个120元。定义UsbSanFactory类，实现UsbSell
4. 定义淘宝的代理商 TaoBao ，实现UsbSell
5. 定义微商的代理商 WeiShang, 实现UsbSell
6. 定义测试类，测试通过淘宝， 微商购买u盘

## 静态代理

静态代理是指，代理类在程序运行前就已经定义好.java源文件，其与目标类的关系在程序运行前就已经确立。在程序运行前代理类已经编译为.class文件。

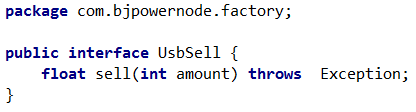
### 静态代理

在idea中创建java工程，

工程名称：ch01-staticproxy

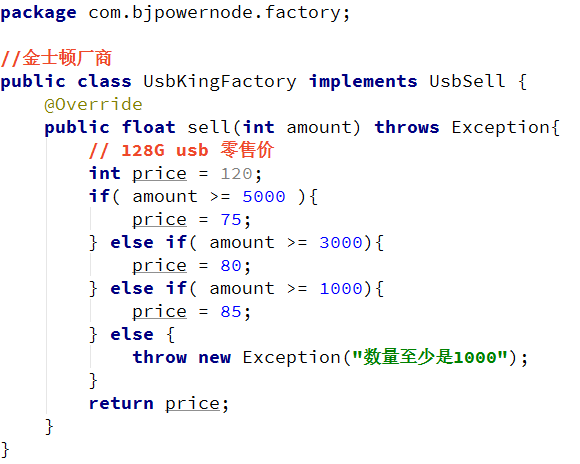
#### 定义业务接口

定义业务接口UsbSell（目标接口），其中含有抽象方法sell(int amount)， sell是目标方法。



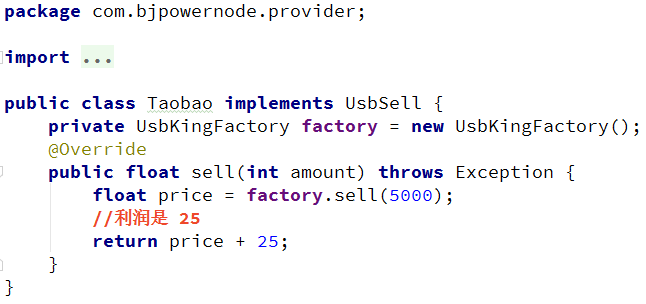
#### 定义接口实现类

目标类UsbKingFactory(金士顿u盘)，该类实现了业务接口。



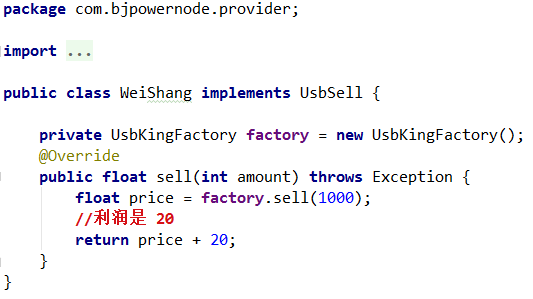
#### 代理商TaoBao

TaoBao就是一个代理类， 代理厂家销售u盘

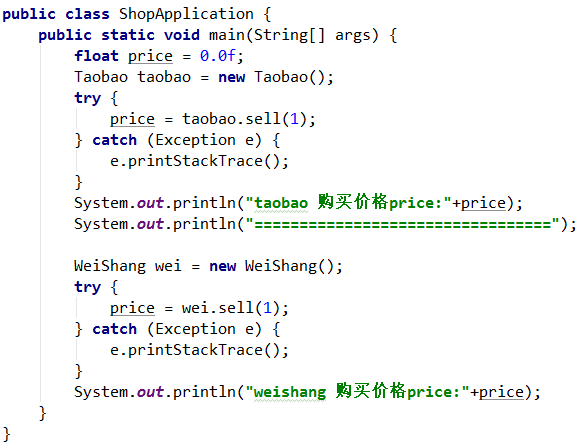


#### 代理商WeiShang

WeiShang就是一个代理类， 代理厂家销售u盘



#### 客户端调用者，购买商品类



使用代理的访问关系图：

client调用者

UsbKingFactory 真正的对象

Taobao代理对象

### 静态代理的缺点

#### 代码复杂，难于管理

代理类和目标类实现了相同的接口，每个代理都需要实现目标类的方法，这样就出现了大量的代码重复。如果接口增加一个方法，除了所有目标类需要实现这个方法外，所有代理类也需要实现此方法。增加了代码维护的复杂度。

#### 代理类依赖目标类，代理类过多

代理类只服务于一种类型的目标类，如果要服务多个类型。势必要为每一种目标类都进行代理，静态代理在程序规模稍大时就无法胜任了，代理类数量过多。

## 动态代理

动态代理是指代理类对象在程序运行时由JVM根据反射机制动态生成的。动态代理不需要定义代理类的.java源文件。

动态代理其实就是jdk运行期间，动态创建class字节码并加载到JVM。

动态代理的实现方式常用的有两种：使用JDK代理代理，与通过CGLIB动态代理。

### jdk的动态代理

jdk动态代理是基于Java的反射机制实现的。使用jdk中接口和类实现代理对象的动态创建。

Jdk的动态要求目标对象必须实现接口，这是java设计上的要求。

从jdk1.3以来，java语言通过java.lang.reflect包提供三个类支持代理模式Proxy, Method和InovcationHandler。

#### InvocationHandler接口

InvocationHandler接口叫做调用处理器，负责完调用目标方法，并增强功能。

通过代理对象执行目标接口中的方法，会把方法的调用分派给调用处理器(InvocationHandler)的实现类，执行实现类中的invoke()方法，我们需要把功能代理写在invoke（）方法中 。



接口中只有一个方法：



在invoke方法中可以截取对目标方法的调用。在这里进行功能增强。Java的动态代理是建立在反射机制之上的。

实现了InvocationHandler接口的类用于加强目标类的主业务逻辑。这个接口中有一个方法invoke()，具体加强的代码逻辑就是定义在该方法中的。通过代理对象执行接口中的方法时，会自动调用invoke()方法。

invoke()方法的介绍如下：

|  |
| --- |
| public Object invoke ( Object proxy, Method method, Object[] args)  proxy：代表生成的代理对象  method：代表目标方法  args：代表目标方法的参数 |

第一个参数proxy是jdk在运行时赋值的，在方法中直接使用，第二个参数后面介绍，第三个参数是方法执行的参数， 这三个参数都是jdk运行时赋值的，无需程序员给出。

#### Method 类

invoke()方法的第二个参数为Method类对象，该类有一个方法也叫invoke()，可以调用目标方法。这两个invoke()方法，虽然同名，但无关。

|  |
| --- |
| public Object invoke ( Object obj, Object... args)  obj：表示目标对象  args：表示目标方法参数，就是其上一层invoke方法的第三个参数 |

该方法的作用是：调用执行obj对象所属类的方法，这个方法由其调用者Method对象确定。

在代码中，一般的写法为

method.invoke(target, args);

其中，method为上一层invoke方法的第二个参数。这样，即可调用了目标类的目标方法。

#### Proxy类

通过JDK的java.lang.reflect.Proxy类实现动态代理，会使用其静态方法newProxyInstance()，依据目标对象、业务接口及调用处理器三者，自动生成一个动态代理对象。

|  |
| --- |
| public static newProxyInstance ( ClassLoader loader, Class<?>[] interfaces,  InvocationHandler handler)  loader：目标类的类加载器，通过目标对象的反射可获取  interfaces：目标类实现的接口数组，通过目标对象的反射可获取  handler：调用处理器。 |

### jdk动态代理实现

jdk动态代理是代理模式的一种实现方式，其只能代理接口。

实现步骤

1、新建一个接口，作为目标接口

2、为接口创建一个实现类，是目标类

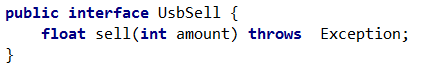
3、创建类实现java.lang.reflect.InvocationHandler接口，调用目标方法并增加其他功能代码

4、创建动态代理对象，使用Proxy.newProxyInstance()方法，并把返回值强制转为接口类型。

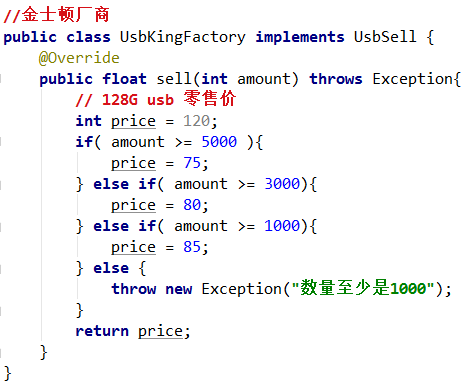
idea创建java project

工程名称：ch02-dynamicproxy

#### 定义目标接口

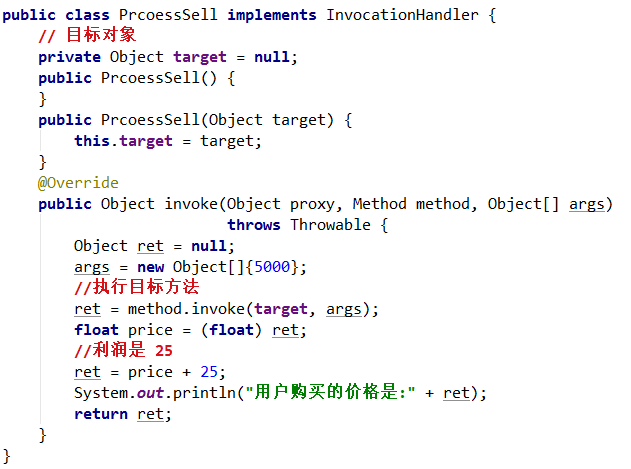


#### 定义目标接口实现类

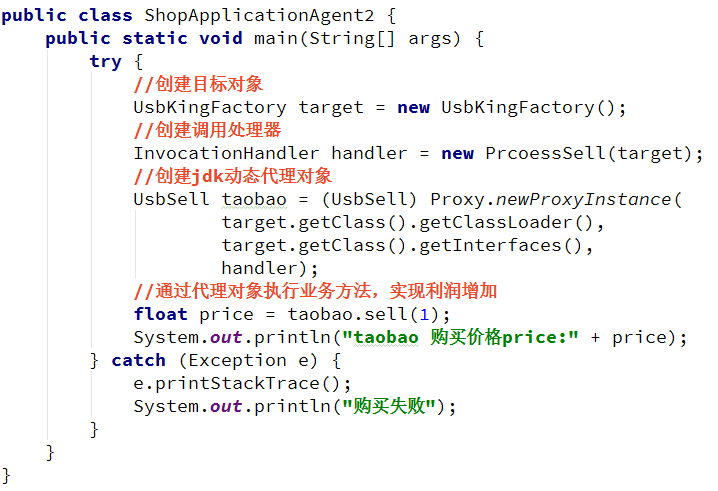


#### 定义调用处理程序

调用处理程序是实现了InvocationHandler的类，在invoke方法中增加业务功能。还需要创建有参构造，参数是目标对象。为的是完成对目标对象的方法调用。



#### 创建动态代理对象



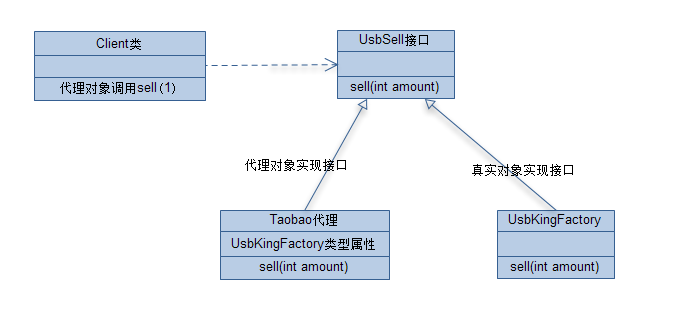
执行流程：

client

MyInvocationHandler:调用处理器截取对方法的调用。执行invoke()方法。

动态代理对象Proxy

类图：



### cgLib代理

CGLIB(Code Generation Library)是一个开源项目。是一个强大的，高性能，高质量的Code生成类库，它可以在运行期扩展Java类与实现Java接口。它广泛的被许多AOP的框架使用，例如Spring AOP。

使用JDK的Proxy实现代理，要求目标类与代理类实现相同的接口。若目标类不存在接口，则无法使用该方式实现。

但对于无接口的类，要为其创建动态代理，就要使用CGLIB来实现。CGLIB代理的生成原理是生成目标类的子类，而子类是增强过的，这个子类对象就是代理对象。所以，使用CGLIB生成动态代理，要求目标类必须能够被继承，即不能是final的类。

cglib经常被应用在框架中，例如Spring ，Hibernate等。Cglib的代理效率高于Jdk。对于cglib一般的开发中并不使用。做了一个了解就可以。