# 校名

**本科生实验报告**

**实验课程 软件体系结构与设计**

**学院名称 计算机与网络安全学院（牛津布鲁克斯学院）**

**专业名称 软件工程**

**学生姓名 徐睿航**

**学生学号 202013160210**

**指导教师 单武扬**

**实验地点 C075-05-32**

**实验成绩**

**二〇二二年四月 二〇二二年六月**

# 实验九 结构型模式应用

## **一、实验目的**

1. 理解“结构型模式”的基本概念、基本原理和应用场景

2. 掌握适配器模式（Adapter）、组合模式（Composite）、代理模式（Proxy）、外观模式（Facade）的使用方法

## **二、实验器材**

1. 计算机一台
2. Rational Rose 工具软件
3. Eclipse集成开发环境

三、实验内容

（1）理解“结构型模式”的基本概念、基本原理和应用场景。

1. 掌握适配器模式(Adapter)、组合模式（Composite)、代理模式（Proxy）

的使用方法。

## **四、实验步骤**

1. 适配器模式

利用适配器模式，完成以下程序设计(UML+JAVA)：

1. 为电源插座(工具箱)开发一个适配器，使该插头可以同时为交流电设备(三相、两相)供电
2. 为该电源插座(工具箱)开发另一个适配器，可以为Usb 设备供电。
3. 分别采用类适配器和对象适配器实现
4. 组合模式

利用组合模式，完成以下程序设计(UML+JAVA)：

1. 人品最好的单老师需要有人帮忙完成一项工作(姑且当作是擦黑板吧)，但是她不知道谁可以帮忙，于是将该指令传达给大班班长。
2. 大班班长、分班班长、寝室长、寝室成员，逐层询问查找,工作会落实到具体的成员身上(该同学具有擦黑板的功能)。
3. 在查找过程中，不管是中间的管理人员,还是后面的执行人员，都无差别地执行相应程序
4. 请将该班级模型描述出来,以便以后单老师无差别、不用思考地分配其他的任务。
5. 代理模式

利用代理模式，完成以下程序设计(UML+IAVA)：

1. 冰淇淋厂商可以生产冰淇淋、销售冰淇淋、查询冰淇淋价格
2. 冰淇淋零售商是冰淇淋厂商的代理，他对外代理的冰淇淋厂商服务为：价格查询、冰淇淋销售。冰淇淋作为一个独立对象，包含5种不同种类
3. 用户通过零售商完成冰淇淋购买,
4. 在代理行为中，冰淇淋零售商做了额外行为;

提价：如查询价格为3元的冰淇淋,代理行为后价格变为4元.

销售统计：按照不同冰淇淋种类统计销量(数量或总)

## **五、实验过程**

1. 适配器模式

分析：

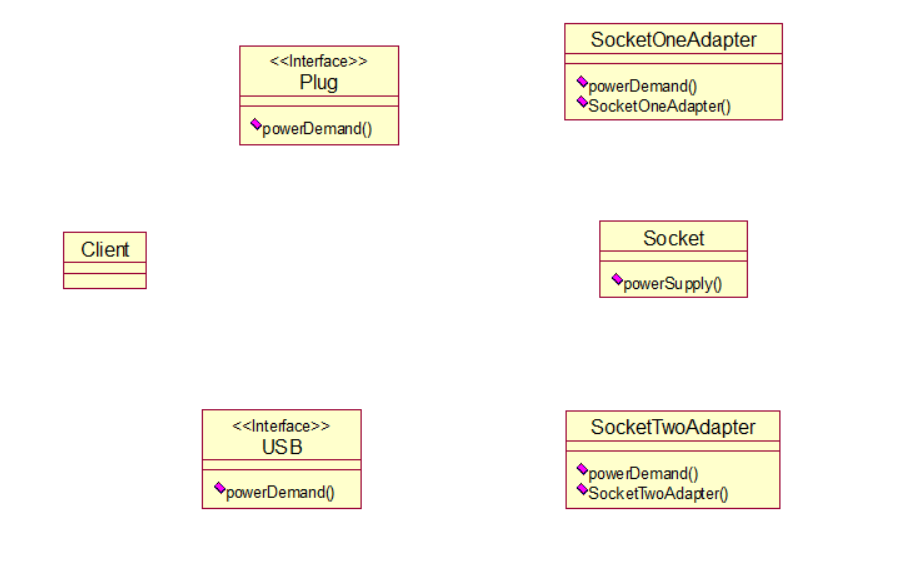
类适配器：为电源插座Socket（工具箱）开发一个适配器SocketOneAdapter，继承Socket，实现Plug接口，使其可同时为交流电设备充电，开发另一个适配器SocketTwoAdapter，继承Socket，实现USB接口，使其可为Usb设备供电。

对象适配器：为插座Socket（工具箱）开发一个适配器SocketOneAdapter，继承Plug，关联Socket，使其可同时为交流电设备充电，开发另一个适配器SocketTwoAdapter，继承USB接口，关联Socket，使其可为Usb设备供电。

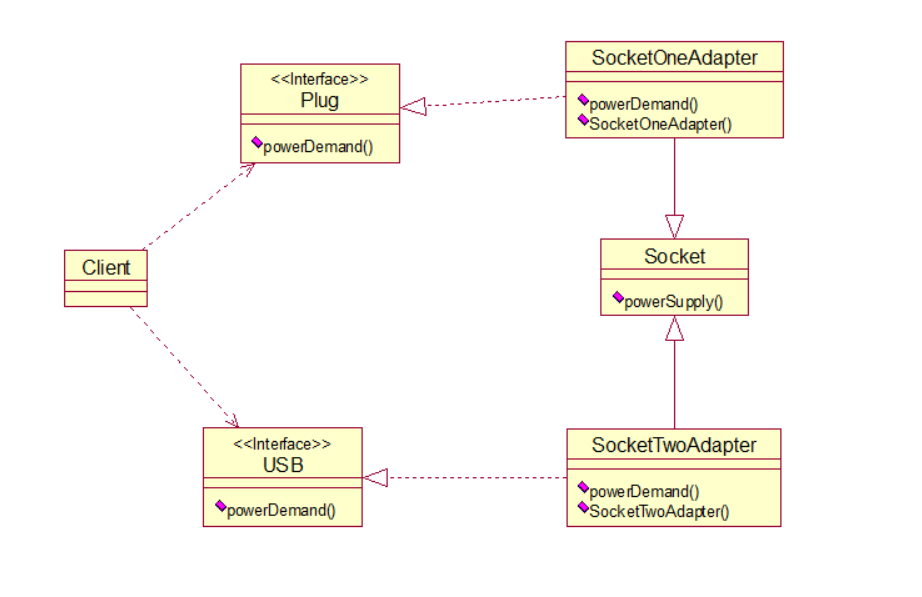
类适配器：

1. 画UML类图

添加Socket、SocketOneAdapter、SocketTwoAdapter、Client类，Plug、USB接口；



为它们之间建立关系，建立Client类与Plug、USB类之间的依赖关系，SocketOneAdapter、SocketTwoAdapter类与Socket类之间的继承关系，与Plug、USB接口的实现关系。



1. 编写Java代码

编写Socket类，其中有方法powerSupply()；

编写Plug接口，其中有方法powerDemand();

USB接口，同样有方法powerDemand();

编写SocketOneAdapter继承Socket类，并实现Plug接口，在powerDemand()方法中调用powerSupply()方法；

编写SocketTwoAdapter继承Socket类，并实现USB接口，在powerDemand()方法中调用powerSupply()方法；

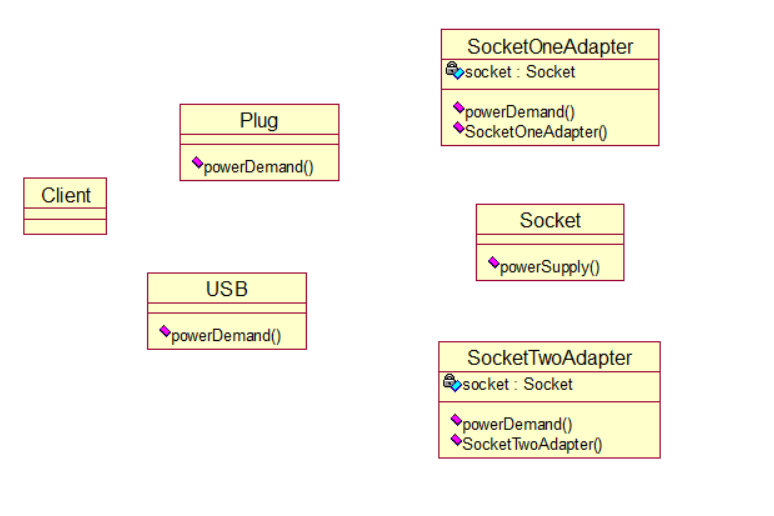
编写Client类，通过适配器为不同的设备供电。

Java代码

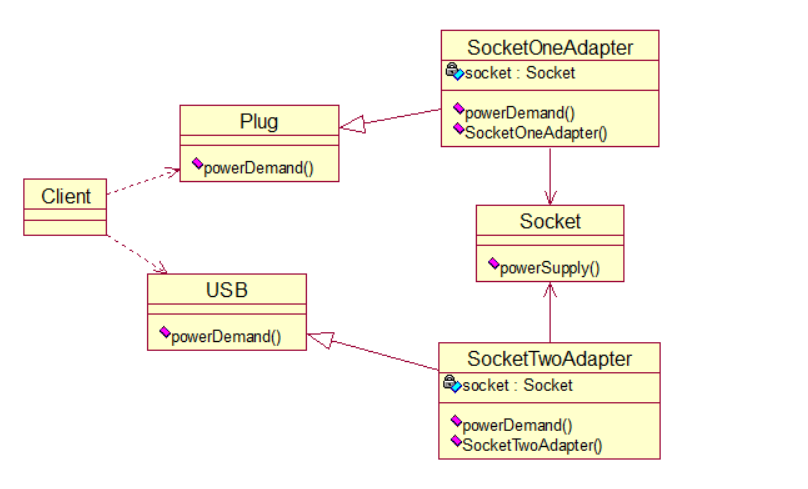
对象适配器：

1. 画UML类图

添加Socket、SocketOneAdapter、SocketTwoAdapter、Client、Plug、USB类；



为它们之间建立关系，建立Client类与Plug、USB类之间的依赖关系，SocketOneAdapter、SocketTwoAdapter类与Socket类之间的关联关系，与Plug、USB接口的继承关系。



1. 编写Java代码

编写Socket类，其中有方法powerSupply()；

编写Plug类，抽象类，其中有powerDemand()抽象方法；

USB类，抽象类，同样有powerDemand()抽象方法；

编写SocketOneAdapter继承Plug类，其数据成员为Socket类的对象socket，在powerDemand()方法中socket调用powerSupply()方法；

编写SocketTwoAdapter继承USB类，其数据成员为Socket类的对象socket，在powerDemand()方法中socket调用powerSupply()方法；

编写Client类，通过适配器为不同的设备供电。

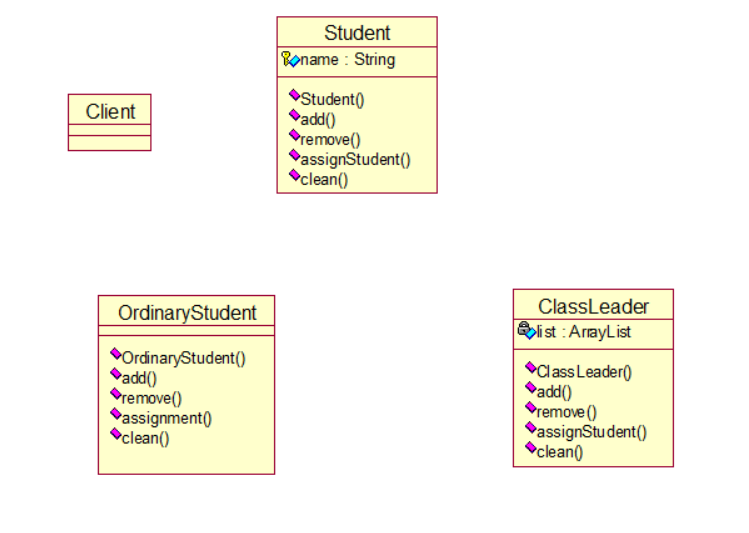
1. 组合模式

分析：

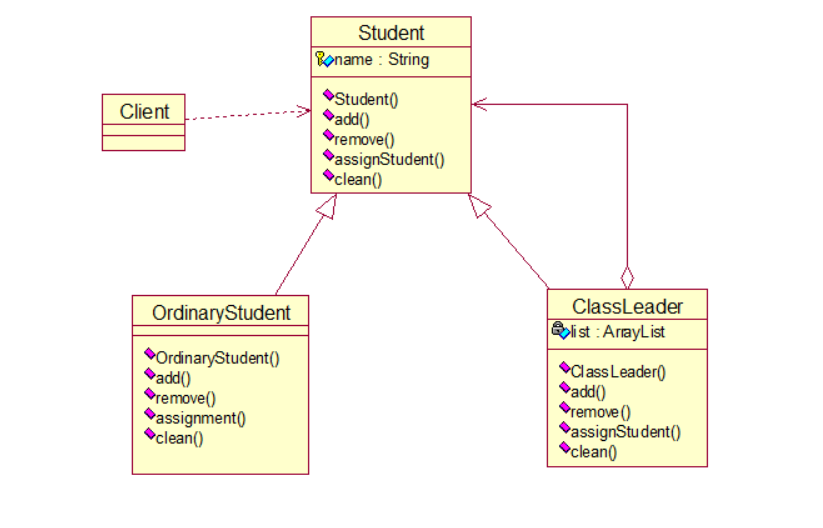
班级里，学生Student对象包含普通学生OrdinaryStudent对象可以檫黑板和班干部ClassLeader对象可以檫黑板和指派同学，班干部ClassLeader具有递归组合特性，可以逐层询问查找。

1. 画UML类图

添加Student、OrdinaryStudent、ClassLeader、Client类；



为它们建立关系，Client类与Student类之间的依赖关系，Student类与OridinaryStudent、ClassLeader之间的继承关系，Student类与ClassLeader类之间的聚合关系。



1. 编写Java代码

编写Student类，抽象类，有add(Student s)、remove(Student s)、clean()、assignStudent(int i)抽象方法；

编写OrdinaryStudent类继承Student类，并实现这些方法，其中add(Student s)、remove(Student s)方法体为空，assignStudnet(int i)方法中返回null；

ClassLeader类同样继承Student类，并实现这些方法，其数据成员为ArrayList<Student>对象；

编写Client类，进行指派同学檫黑板。

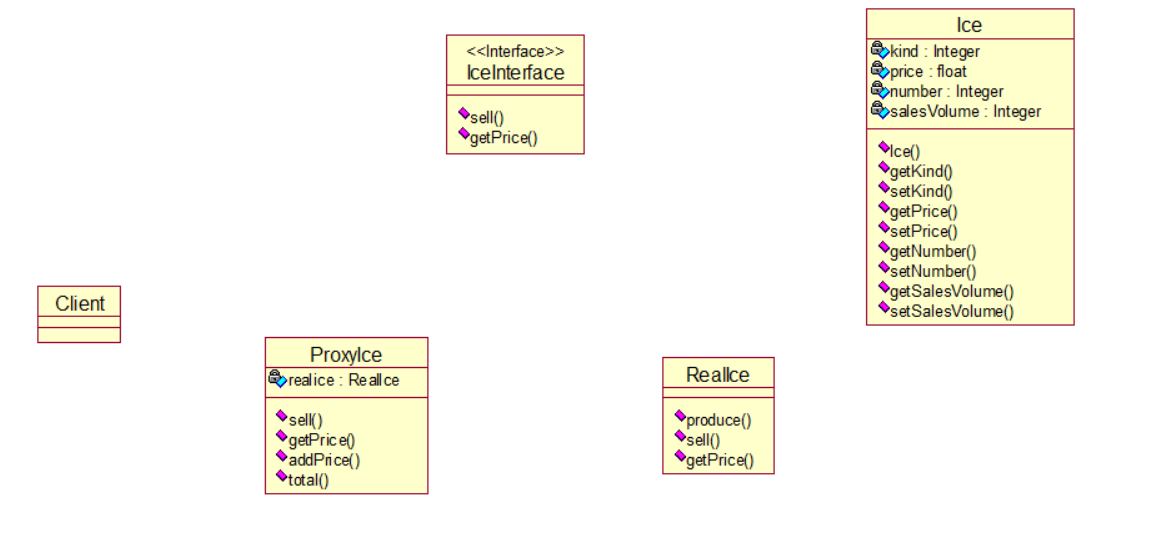
1. 代理模式

分析：

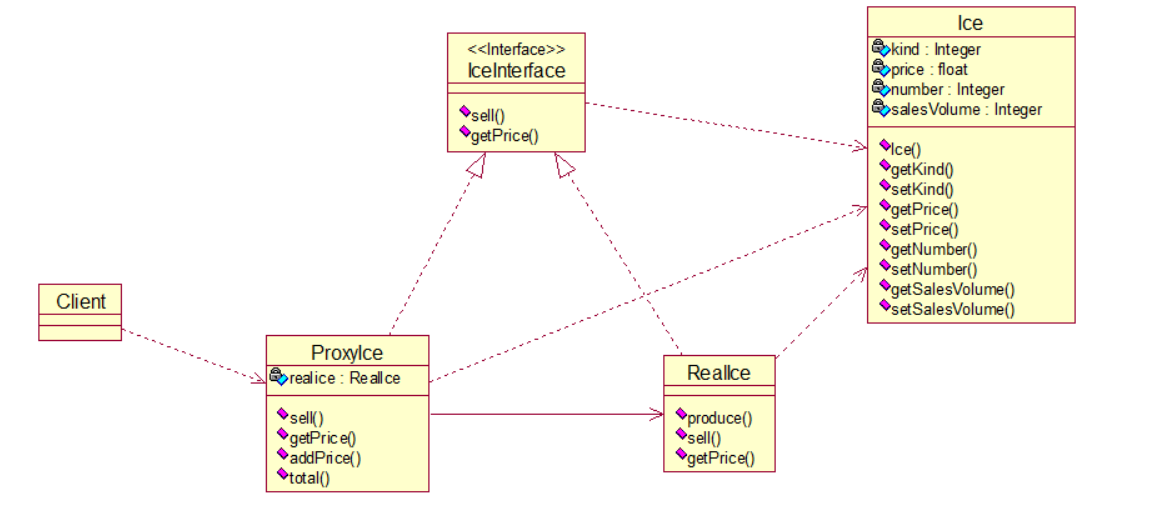
冰淇淋厂商RealIce（被代理对象）可以生产冰淇淋、销售冰淇淋、查询冰淇淋价格，冰淇淋零销商ProxyIce（代理对象）可以价格查询、冰淇淋销售、提价、销售统计，去掉用户不能看见的内容生成冰淇淋，冰淇淋接口IceInterface（公用接口）为RealIce、ProxyIce的公共接口。

1. 画UML类图

添加Client、Ice、RealIce、ProxyIce类，IceInterface接口；



为它们建立关系，建立Client类与ProxyIce类之间的依赖关系，ProxyIce、RealIce类与IceInterface接口之间的实现关系，ProxyIce类与RealIce类之间的关联关系，Ice类与IceInterface接口、ProxyIce、RealIce类之间的依赖关系。



1. 编写Java代码

编写Ice类，其中有数据成员kind表示种类，number表示剩余个数，salesVolume表示销售个数，还有getter()、setter()方法；

编写IceInterface接口，有sell(Ice ice, int n)、getPrice(Ice ice)方法；

编写RealIce类实现IceInterface接口，实现sell()、getPrice()方法，另还有produce()方法；

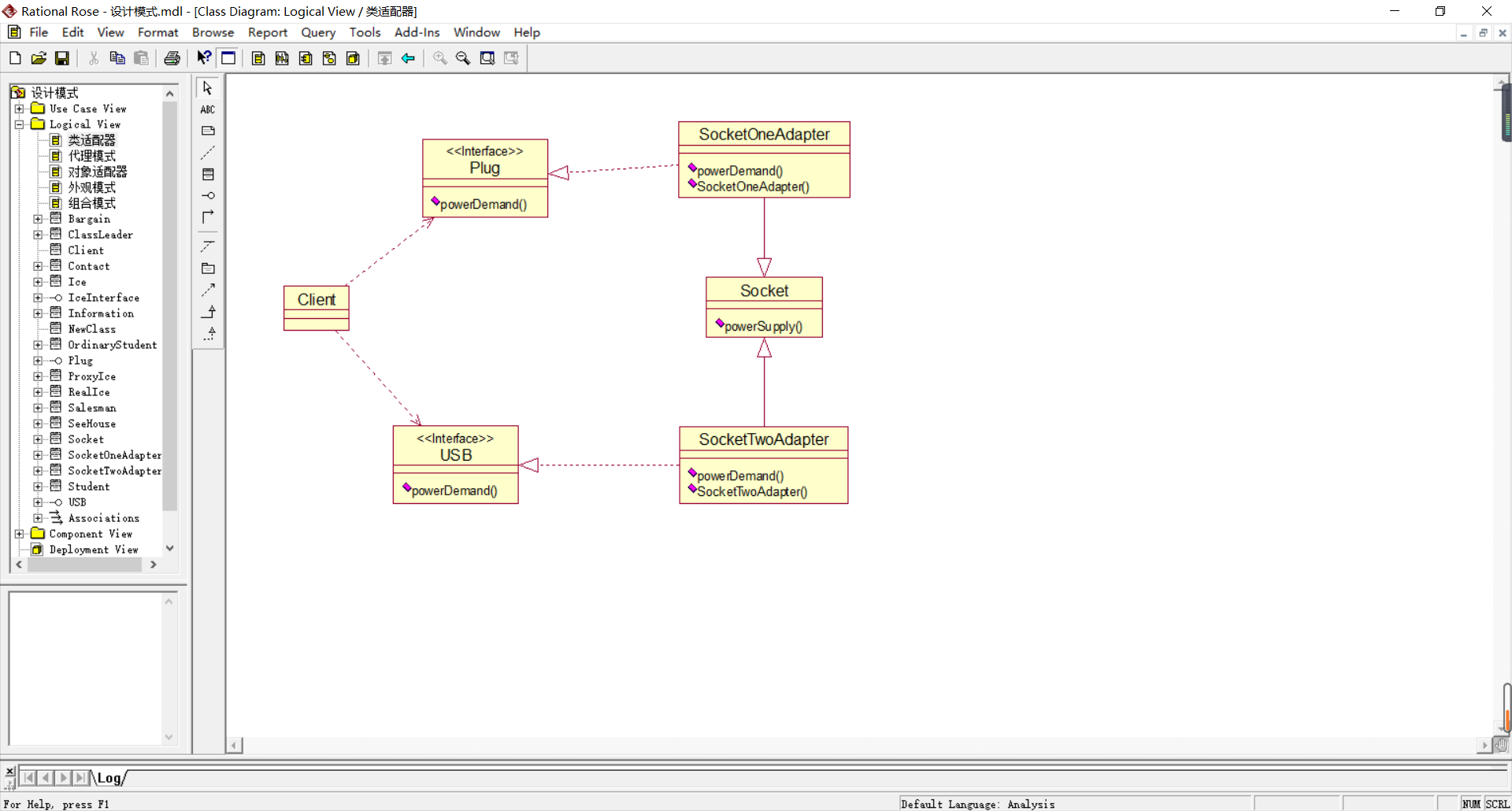
编写ProxyIce类实现IceInterface接口，其数据成员为RealIce类的对象realice，实现sell()、getPrice()方法，另还有addPrice()、total()方法；

编写Client类，进行冰淇淋代理。

## **六、实验结果或核心代码**

1. 适配器模式

类适配器：



Socket.java

**package** classadapter;

**public** **class** Socket {

**public** **void** powerSupply() {

System.***out***.println("充电中");

}

}

Plug.java

**package** classadapter;

**public** **interface** Plug {

**public** **void** powerDemand();

}

USB.java

**package** classadapter;

**public** **interface** USB {

**public** **void** powerDemand();

}

SocketOneAdapter.java

**package** classadapter;

**public** **class** SocketOneAdapter **extends** Socket **implements** Plug {

**public** **void** powerDemand() {

powerSupply();

}

}

SocketTwoAdapter.java

**package** classadapter;

**public** **class** SocketTwoAdapter **extends** Socket **implements** USB {

**public** **void** powerDemand() {

powerSupply();

}

}

Client.java

**package** classadapter;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

Plug plug = **new** SocketOneAdapter();

System.***out***.print("交流电设备：");

plug.powerDemand();

USB usb = **new** SocketTwoAdapter();

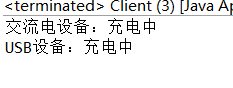
System.***out***.print("USB设备：");

usb.powerDemand();

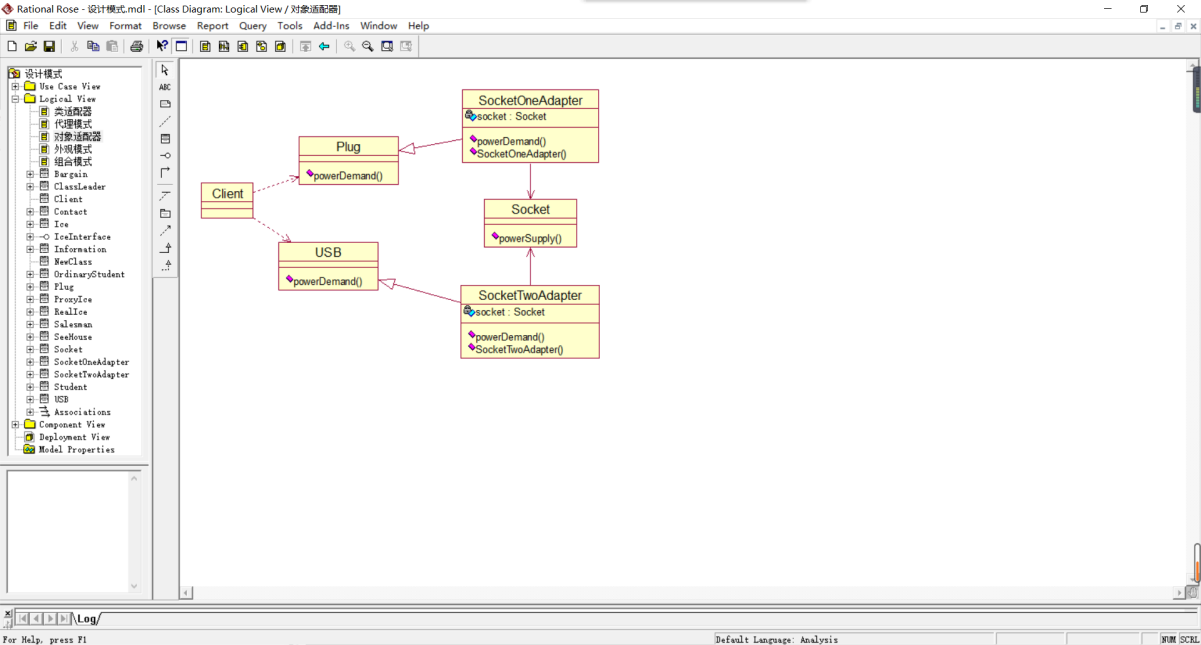
}

}

运行结果：



对象适配器：



Socket.java

**package** objectadapter;

**public** **class** Socket {

**public** **void** powerSupply() {

System.***out***.println("充电中");

}

}

Plug.java

**package** objectadapter;

**public** **abstract** **class** Plug {

**public** **abstract** **void** powerDemand();

}

USB.java

**package** objectadapter;

**public** **abstract** **class** USB {

**public** **abstract** **void** powerDemand();

}

SocketOneAdapter.java

**package** objectadapter;

**public** **class** SocketOneAdapter **extends** Plug {

**private** Socket socket;

**public** SocketOneAdapter(Socket socket) {

**this**.socket = socket;

}

**public** **void** powerDemand() {

socket.powerSupply();

}

}

SocketTwoAdapter.java

**package** objectadapter;

**public** **class** SocketTwoAdapter **extends** USB {

**private** Socket socket;

**public** SocketTwoAdapter(Socket socket) {

**this**.socket = socket;

}

**public** **void** powerDemand() {

socket.powerSupply();

}

}

Client.java

**package** objectadapter;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

Socket socket = **new** Socket();

Plug plug = **new** SocketOneAdapter(socket);

System.***out***.print("交流电设备：");

plug.powerDemand();

USB usb = **new** SocketTwoAdapter(socket);

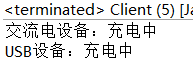
System.***out***.print("USB设备：");

usb.powerDemand();

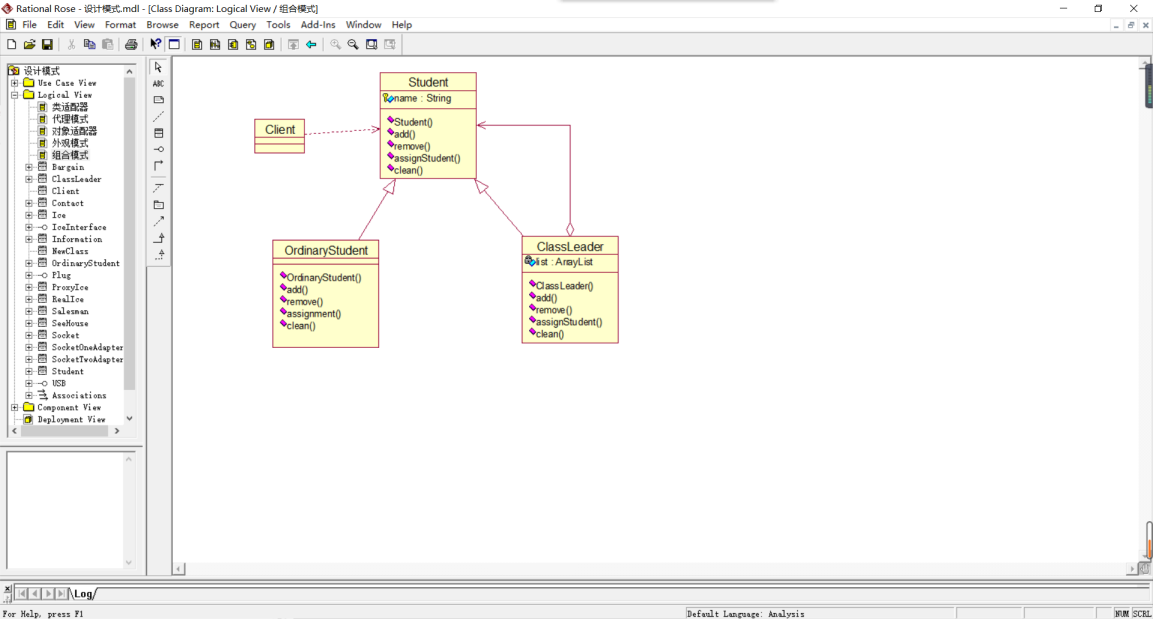
}

}

运行结果：



1. 组合模式



Student.java

**package** composite;

**public** **abstract** **class** Student {

**protected** String name;

**public** Student(String name) {

**this**.name = name;

}

**public** **abstract** **void** add(Student s);

**public** **abstract** **void** remove(Student s);

**public** **abstract** Student assignStudent(**int** i);

**public** **abstract** **void** clean();

}

OrdinaryStudent.java

**package** composite;

**public** **class** OrdinaryStudent **extends** Student {

**public** OrdinaryStudent(String name) {

**super**(name);

}

**public** **void** add(Student s) {}

**public** **void** remove(Student s) {}

**public** Student assignStudent(**int** i) {

**return** **null**;

}

**public** **void** clean() {

System.***out***.println(name + "檫黑板中...");

}

}

ClassLeader.java

**package** composite;

**import** java.util.ArrayList;

**public** **class** ClassLeader **extends** Student {

**private** ArrayList<Student> list = **new** ArrayList<Student>();

**public** ClassLeader(String name) {

**super**(name);

}

**public** **void** add(Student s) {

list.add(s);

}

**public** **void** remove(Student s) {

list.remove(s);

}

**public** Student assignStudent(**int** i) {

**return** (Student)list.get(i);

}

**public** **void** clean() {

System.***out***.println("name" + "檫黑板中...");

}

}

Client.java

**package** composite;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

Student bigMonitor = **new** ClassLeader("大班班长");

Student smallMonitorOne = **new** ClassLeader("1班班长");

Student smallMonitorTwo = **new** ClassLeader("2班班长");

bigMonitor.add(smallMonitorOne);

bigMonitor.add(smallMonitorTwo);

Student dormAdminOne = **new** ClassLeader("1班1号寝室长");

Student dormAdminTwo = **new** ClassLeader("1班2号寝室长");

Student dormAdminThree = **new** ClassLeader("2班1号寝室长");

Student dormAdminFour = **new** ClassLeader("2班2号寝室长");

smallMonitorOne.add(dormAdminOne);

smallMonitorOne.add(dormAdminTwo);

smallMonitorTwo.add(dormAdminThree);

smallMonitorTwo.add(dormAdminFour);

Student studentOne = **new** OrdinaryStudent("1班1号寝室成员1号");

Student studentTwo = **new** OrdinaryStudent("1班2号寝室成员1号");

Student studentThree = **new** OrdinaryStudent("2班1号寝室成员1号");

Student studentFour = **new** OrdinaryStudent("2班2号寝室成员1号");

dormAdminOne.add(studentOne);

dormAdminTwo.add(studentTwo);

dormAdminThree.add(studentThree);

dormAdminFour.add(studentFour);

Student cleaner = bigMonitor.assignStudent(1).assignStudent(1).assignStudent(0);

cleaner.clean();

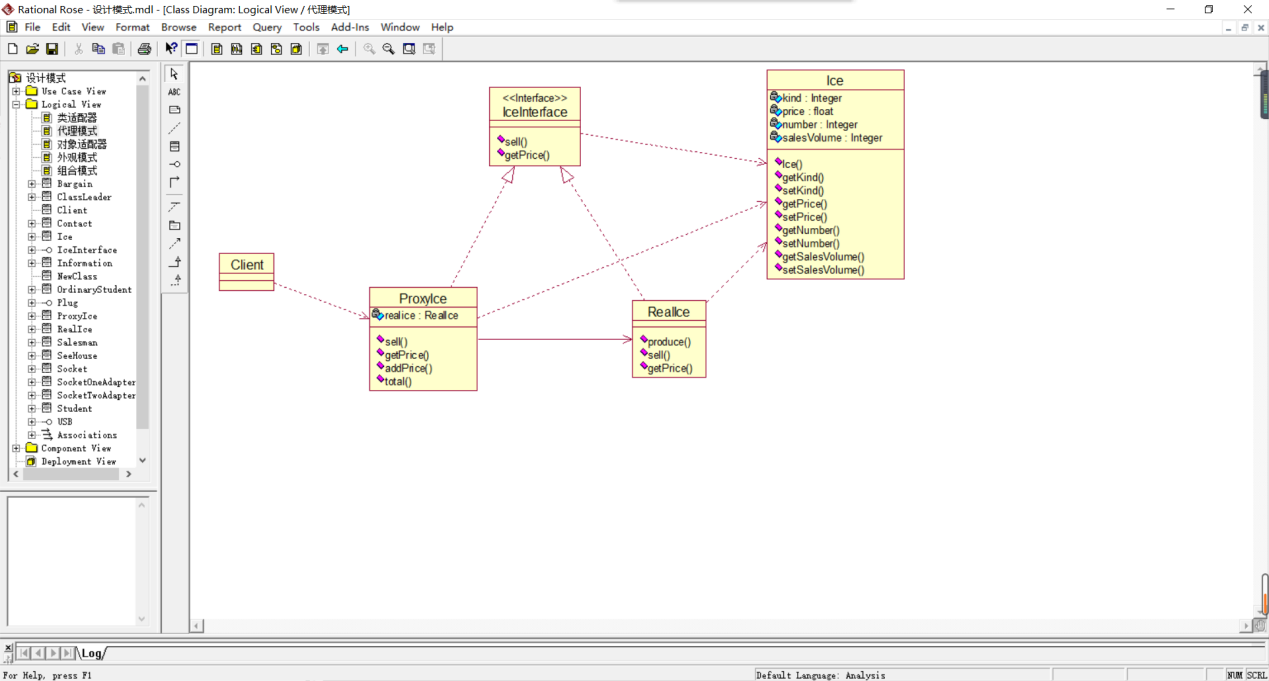
}

}

运行结果：



1. 代理模式



Ice.java

**package** proxy;

**public** **class** Ice {

**private** **int** kind;

**private** **float** price;

**private** **int** number;

**private** **int** salesVolume = 0;

**public** Ice(**int** kind, **float** price, **int** number) {

**this**.setKind(kind);

**this**.setPrice(price);

**this**.number = number;

}

**public** **int** getKind() {

**return** kind;

}

**public** **void** setKind(**int** kind) {

**this**.kind = kind;

}

**public** **float** getPrice() {

**return** price;

}

**public** **void** setPrice(**float** price) {

**this**.price = price;

}

**public** **int** getNumber() {

**return** number;

}

**public** **void** setNumber(**int** number) {

**this**.number = number;

}

**public** **int** getSalesVolume() {

**return** salesVolume;

}

**public** **void** setSalesVolume(**int** salesVolume) {

**this**.salesVolume = salesVolume;

}

}

IceInterface.java

**package** proxy;

**public** **interface** IceInterface {

**public** **void** sell(Ice ice, **int** n);

**public** **float** getPrice(Ice ice);

}

RealIce.java

**package** proxy;

**public** **class** RealIce **implements** IceInterface {

**public** Ice produce(**int** kind, **float** price, **int** number) {

Ice ice = **new** Ice(kind, price, number);

System.***out***.println("生产了" + price + "元的" + kind + "号冰淇淋" + number + "个");

**return** ice;

}

**public** **void** sell(Ice ice, **int** n) {

**int** number = ice.getNumber();

**int** kind = ice.getKind();

**double** price = ice.getPrice();

**int** salesVolume = ice.getSalesVolume();

**if**(number >= n) {

number -= n;

salesVolume += n;

ice.setNumber(number);

ice.setSalesVolume(salesVolume);

**double** sum = n \* price;

System.***out***.println(kind + "号冰淇淋卖出" + n + "个，共" + sum + "元");

}

**else** {

System.***out***.println(kind + "号冰淇淋数量不够！");

}

}

**public** **float** getPrice(Ice ice) {

**float** price = ice.getPrice();

**return** price;

}

}

ProxyIce.java

**package** proxy;

**public** **class** ProxyIce **implements** IceInterface {

**private** RealIce realice = **new** RealIce();

**public** **void** sell(Ice ice, **int** n) {

realice.sell(ice, n);

}

**public** **float** getPrice(Ice ice) {

**float** price = realice.getPrice(ice);

**return** price;

}

**public** **void** addPrice(Ice ice, **float** m) {

**float** price = ice.getPrice();

price += m;

ice.setPrice(price);

System.***out***.println(ice.getKind() + "号冰淇淋涨价" + m + "元，现为" + price +"元");

}

**public** **int** total(Ice ice) {

**int** salesVolume = ice.getSalesVolume();

**return** salesVolume;

}

}

Client.java

**package** proxy;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

RealIce realice = **new** RealIce();

Ice ice = realice.produce(2, 5.5f, 100);

ProxyIce it = **new** ProxyIce();

**double** price = it.getPrice(ice);

System.***out***.println(ice.getKind() + "号冰淇淋价格为" + price + "元");

it.sell(ice, 3);

it.addPrice(ice, 0.5f);

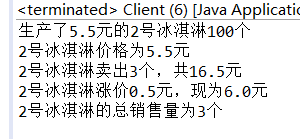
**int** salesVolume = it.total(ice);

System.***out***.println(ice.getKind() + "号冰淇淋的总销售量为" + salesVolume + "个");

}

}

运行结果：



|  |  |
| --- | --- |
| **学生实验 心得** | 这次实验我更加理解了，结构型模式的应用。  学生（签名）：徐睿航  2022 年 6月 10日 |
| **指导**  **教师**  **评语** | 成绩评定：  指导教师（签名）： 年 月 日 |