《软件体系结构》实验指导书

-参考答案

刘　伟

weiliu@csu.edu.cn

实验 1《UML 软件需求建模实验》

实验学时：_____　实验地点：_____ 实验日期：_____

一、实验目的

1．安装和使用建模工具 PowerDesigner，熟练使用 PowerDesigner 绘制常用的 UML 图形，熟悉常用的 UML 符号；

2．使用用例模型来描述软件需求，包括绘制用例图，撰写用例文档并制作用例检查矩阵；

3. 使用活动图为业务流程建模，使用顺序图描述用例交互细节，使用状态图描述对象的状态及转换，使用领域类图和对象图构造领域模型；

4. 使用用例图、活动图、顺序图、状态图、领域类图和对象图等构造软件系统的需求模型。

二、实验内容

1．根据以下场景绘制用例图：

某企业为了方便员工用餐，为企业餐厅开发了一个订餐系统（COS：Cafeteria Ordering System），企业员工可通过企业内联网使用该系统。该系统功能描述如下：

(1) 企业的任何员工都可以查看菜单和今日特价；

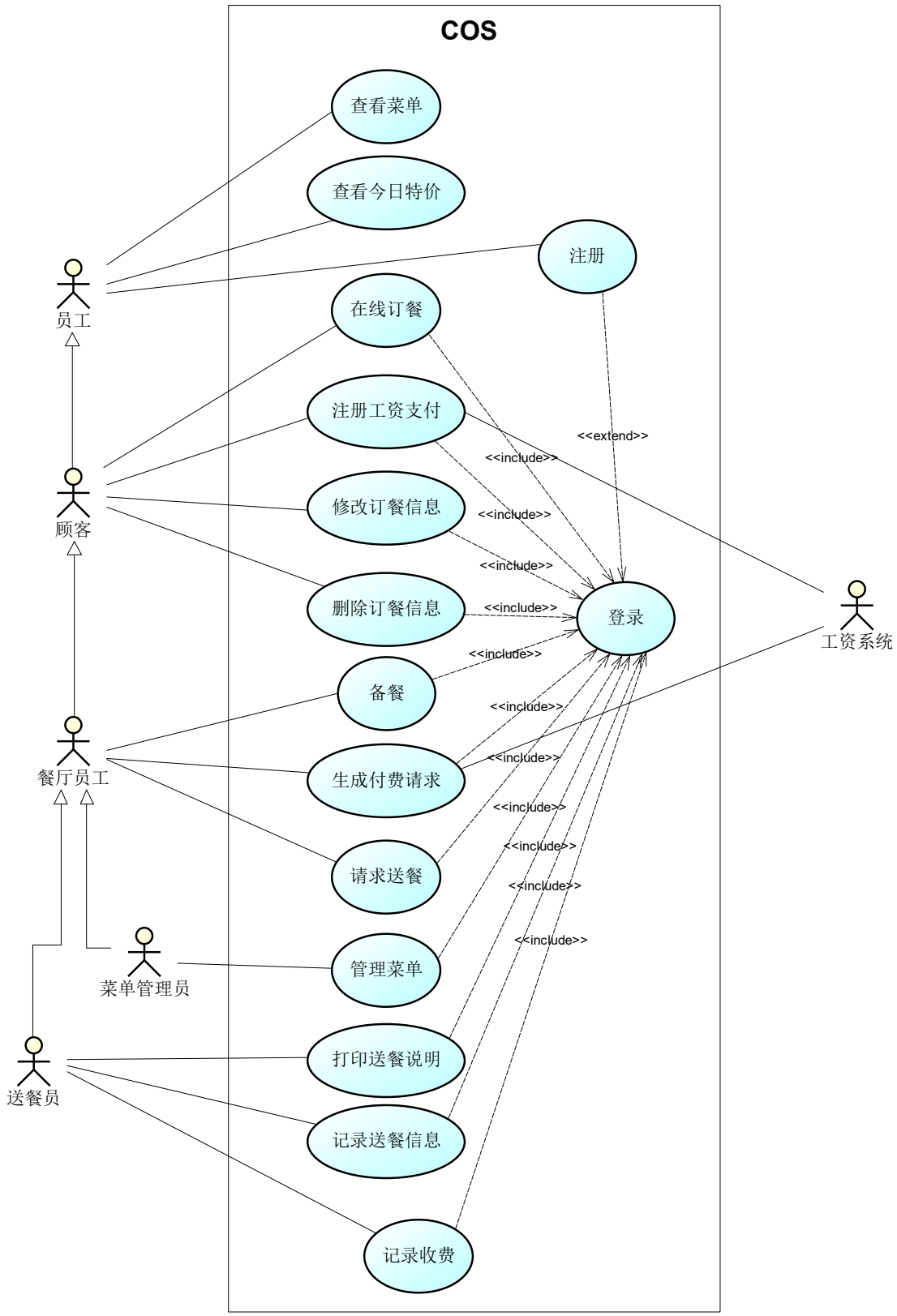
(2) 系统的顾客是注册到系统的员工，可以在线订餐（以下操作均需先登录）、注册工资支付、修改订餐信息和删除订餐信息，在注册工资支付时需要通过工资系统进行身份验证；

(3) 餐厅员工是特殊的顾客，可以进行备餐（系统记录备餐信息）、生成付费请求和请求送餐，其中对于注册使用工资支付的顾客生成付费请求并发送给工资系统；

(4) 菜单管理员是餐厅员工的一种，可以管理菜单；

(5) 送餐员也是餐厅员工的一种，可以打印送餐说明、记录送餐信息（如送餐时间）以及记录收费（对于没有注册工资支付的顾客，由送餐员收取现金后记录）。

参考答案：



2．某银行准备开发一个网上信用卡管理系统 CCMS，该系统的基本功能为：

(1) 信用卡申请。非信用卡客户填写信用卡申请表，说明所要申请的信用卡类型及申请者的基本信息，提交 CCMS 登录。如果信用卡申请被银行接受，客户会收到银行的确认函，

并告知用户信用卡的有效期及信贷限额；否则银行会发送一封拒绝函给该客户。客户收到确认函后，需再次登录 CCMS，用信用卡号和密码激活该信用卡。激活操作结束后，CCMS 将激活通知发送给客户，告知客户其信用卡是否被成功地激活。

(2) 月报表生成。在每个月第一天的零点，CCMS 为每个信用卡客户创建一份月报表，对该客户上月的信用卡交易情况及交易额进行统计。信用卡客户可以登录 CCMS 查看月报表，也可以要求 CCMS 提供打印出的月报表。

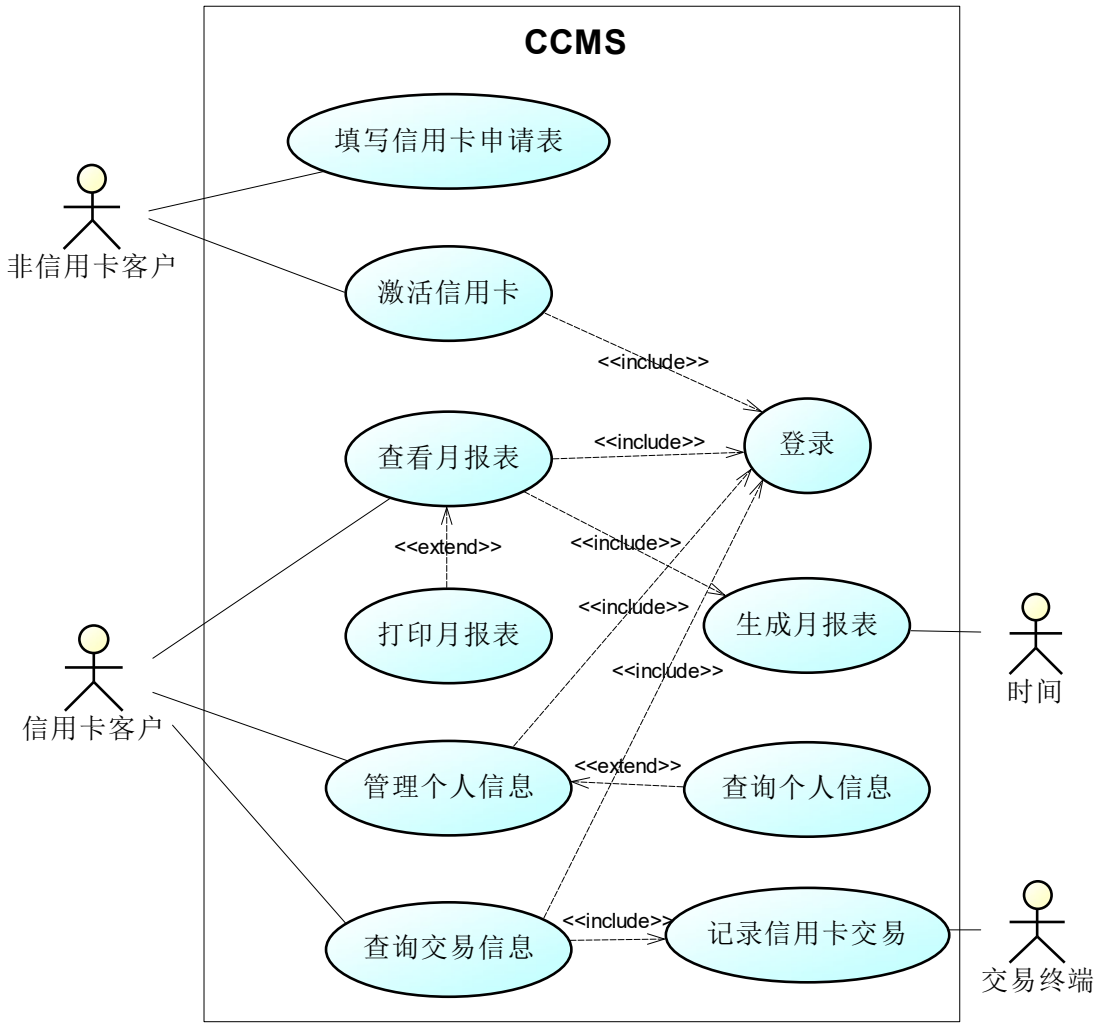
(3) 信用卡客户信息管理。信用卡客户的个人信息可以在 CCMS 中进行在线的管理。每个信用卡客户可以在线查询其个人信息。

(4) 信用卡交易记录。信用卡客户使用信息卡进行的每一笔交易都会记录在 CCMS 中。

(5) 交易信息查询。信用卡客户可以登录 CCMS 查询并核实其信用卡交易记录及交易额。

构造该系统的用例模型，要求绘制用例图，编写相应的用例文档，还需提供用例追踪矩阵。

参考答案：



3. 根据以下描述绘制某订餐系统中顾客订餐过程的活动图（要求绘制泳道）：

(1) 顾客请求查看菜单；

(2) 系统显示菜单和今日特价；

(3) 顾客选菜；

(4) 系统显示订单和价格；

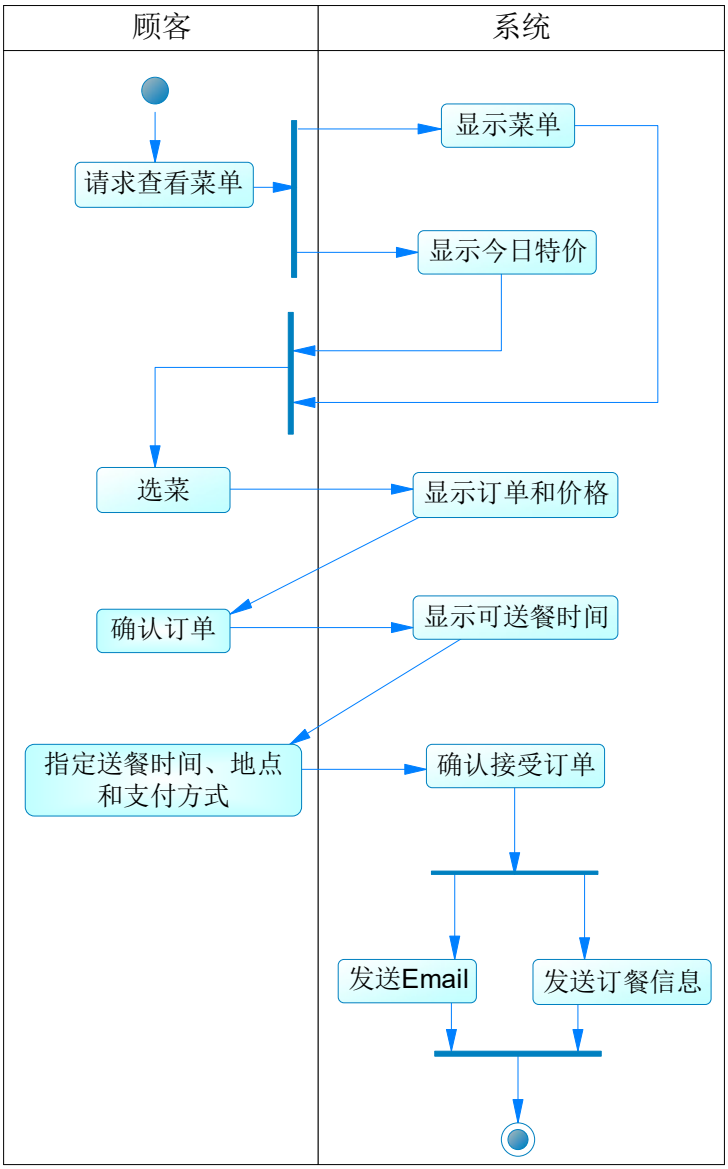
(5) 顾客确认订单；

(6) 系统显示可送餐时间；

(7) 顾客指定送餐时间、地点和支付方式；

(8) 系统确认接受订单，然后发送 Email 给顾客以确认订餐，同时发送相关订餐信息通知给餐厅员工。

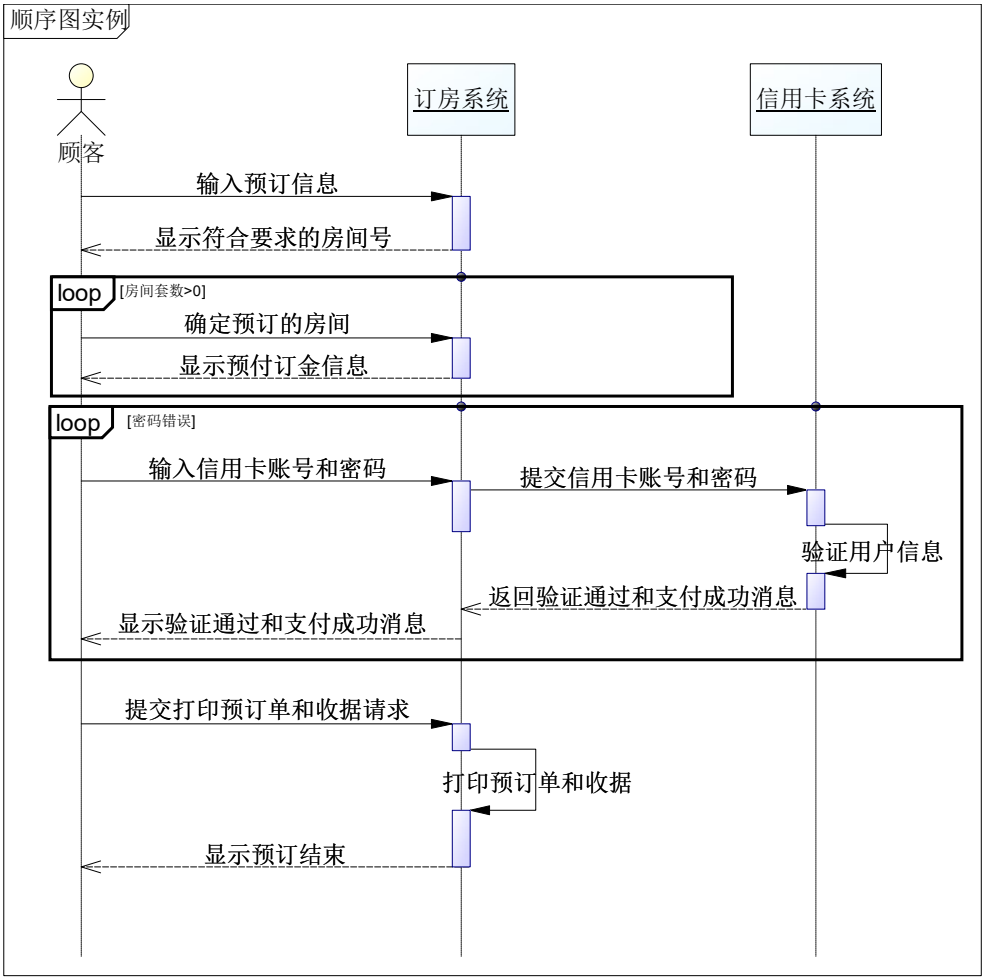
参考答案：



4. 根据如下描述绘制顺序图：

在某在线订房系统中，顾客输入房间套数、房间类型、入住时间、入住天数等信息，系统显示符合要求的房间号；顾客确定预订的房间，系统显示预付订金信息；顾客输入信用卡账号和密码，系统请求银行信用卡系统提供支付服务；银行信用卡系统认证用户信息并返回认证通过和支付成功消息，顾客请求系统打印预订单和收据，系统打印相关资料；预订结束。

参考答案：



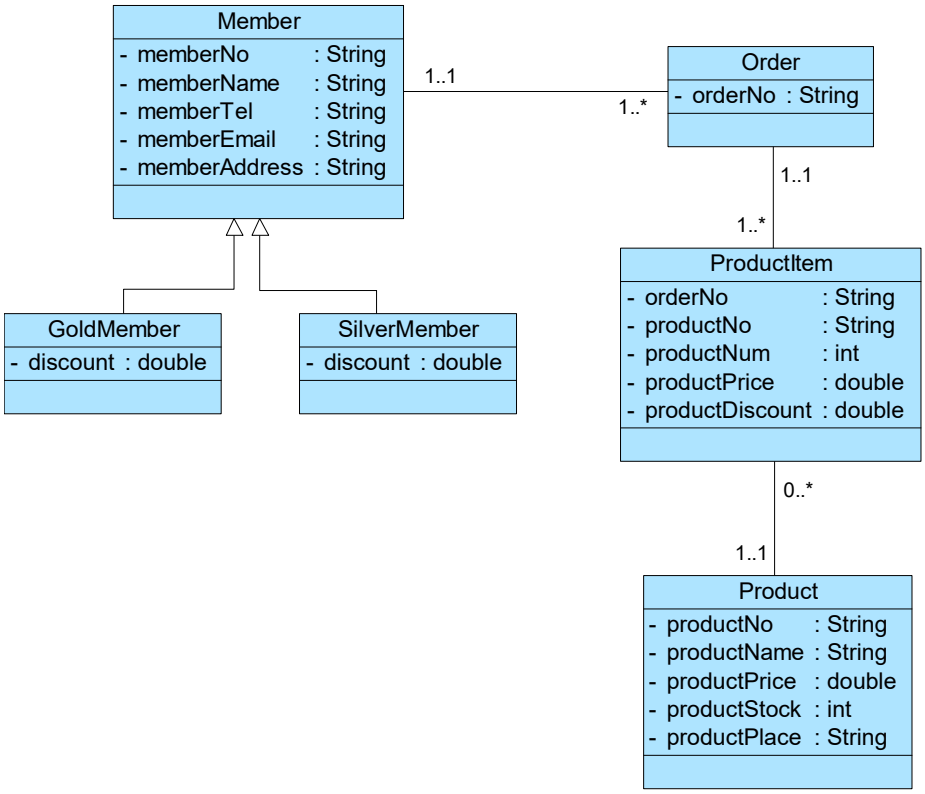
5. 根据如下描述绘制请假条的状态图：

在某 OA 系统中，员工提出请假申请，请假条进入“申请”状态；部门经理审批该请假条，如果请假时间小于等于 5 天且请假条填写完整、合理（例如请假理由合理，请假对工作无重大影响等），则请假条修改为“部门经理已审批”状态；如果因为请假时间超过 5 天，部门经理无法审批，则状态改为“部门经理已查阅”，系统自动将请假条转至公司主管人事行政的副总，如果人事行政副总同意该请假条，则请假条改为“公司副总已审批”状态，否则将进入“拒绝”状态；如果部门经理或者公司副总认为请假条填写不完整或者不合理（例如请假理由不充分），则请假条直接进入“拒绝”状态，此时，员工可以放弃请假申请，也可以选择修改请假申请，修改之后请假条将重新进入“申请”状态。

6. 根据如下描述绘制领域类图，并绘制一个相对应的对象图。

某商场会员管理系统包含一个会员类，会员的基本信息包括会员编号、会员姓名、联系电话、电子邮箱、地址等，会员可分为金卡会员和银卡会员两种，不同类型的会员在购物时可以享受不同的折扣；每个会员可以拥有一个或多个订单，每一个订单又可以包含至少一条商品销售信息，商品销售信息包括订单编号、商品编号、商品数量、商品单价和折扣等；每一条商品销售信息对应一类商品，商品信息包括商品编号、商品名称、商品单价、商品库存量、商品产地等。

参考类图：



三、实验方法

1. 安装和使用建模工具 PowerDesigner，熟练使用 PowerDesigner 绘制常用的 UML 图形，熟悉常用的 UML 符号；

2. 使用用例模型来描述软件需求，包括绘制用例图，撰写用例文档并制作用例检查矩阵；

3. 使用活动图为业务流程建模，使用顺序图描述用例交互细节，使用状态图描述对象的状态及转换，使用领域类图和对象图构造领域模型；

4. 根据项目需求描述，使用 PowerDesigner 绘制用例图、活动图、顺序图、状态图、领域类图和对象图等 UML 图形，使用 UML 构造软件系统的需求模型。

四、实验步骤

1. 安装 PowerDesigner；

2. 熟悉 PowerDesigner 的常用功能；

3. 分析订餐系统需求，使用 PowerDesigner 绘制用例图；

4. 分析需求，构造网上信用卡管理系统 CCMS 的需求模型，使用 PowerDesigner 绘制用例图，撰写用例文档并创建用例追踪矩阵；

5. 分析订餐系统中，顾客订餐过程的流程，使用 PowerDesigner 绘制活动图；

6. 分析在线订房系统中顾客与系统之间的交互过程，使用 PowerDesigner 绘制顺序图；

7. 分析 OA 系统中请假条的状态及其转换，使用 PowerDesigner 绘制状态图；

8. 分析并识别商场会员管理系统中的领域类，使用 PowerDesigner 绘制相应的领域类图和对象图。

五、实验结果

注：有程序的要求附上程序源代码，有图表的要有截图并有相应的文字说明和分析。

1. 绘制并提交订餐系统用例图；

2. 构造网上信用卡管理系统 CCMS 的需求模型，提交用例图、用例文档和用例追踪矩阵；

3. 绘制并提交订餐系统活动图；

4. 绘制并提交在线订房系统的顺序图；

5. 绘制并提交 OA 系统中请假条的状态图；

6. 绘制并提交商场会员管理系统的领域类图和对象图。

六、实验小结

给出本次实验的体会，如学会了什么，遇到哪些问题，如何解决这些问题，存在哪些有待改进的地方。

实验 2《UML 软件设计建模实验》

实验学时：_____ 实验地点：_____ 实验日期：_____

一、实验目的

1. 学习从系统中识别类并将分析类映射到设计类，学习将类封装为包，绘制包图，并通过包图体现系统的分层架构；

2. 使用顺序图描述对象之间的交互次序，学习如何将类图和顺序图映射为源代码；

3. 使用构件图描述物理文件及其相互关系，使用部署图描述硬件结构及其驻留构件；

4. 使用类图、包图、顺序图、构件图、部署图等构造软件系统的设计模型。

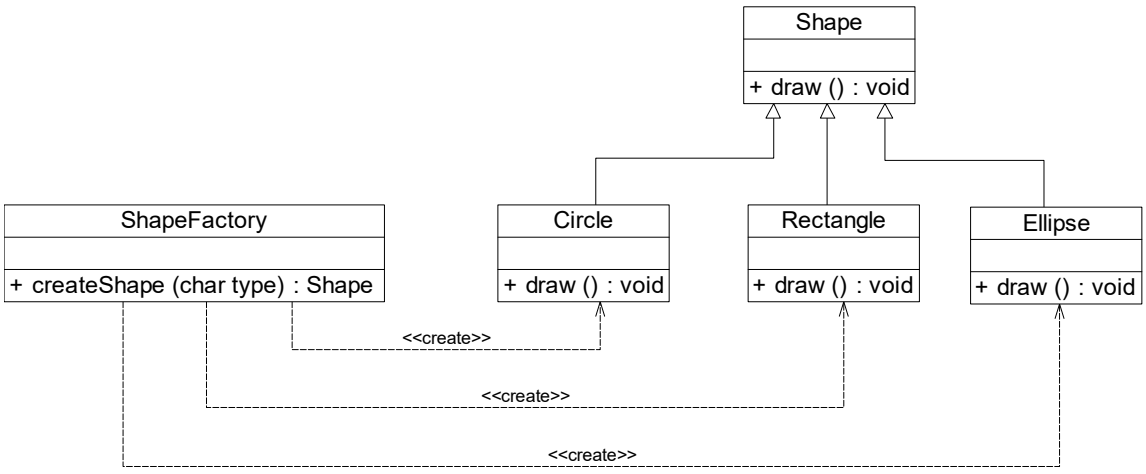
二、实验内容

1．根据以下描述绘制类图，再正向工程生成 Java 源代码（也可生成其他面向对象语言的源代码，如 C++或 C#等）：

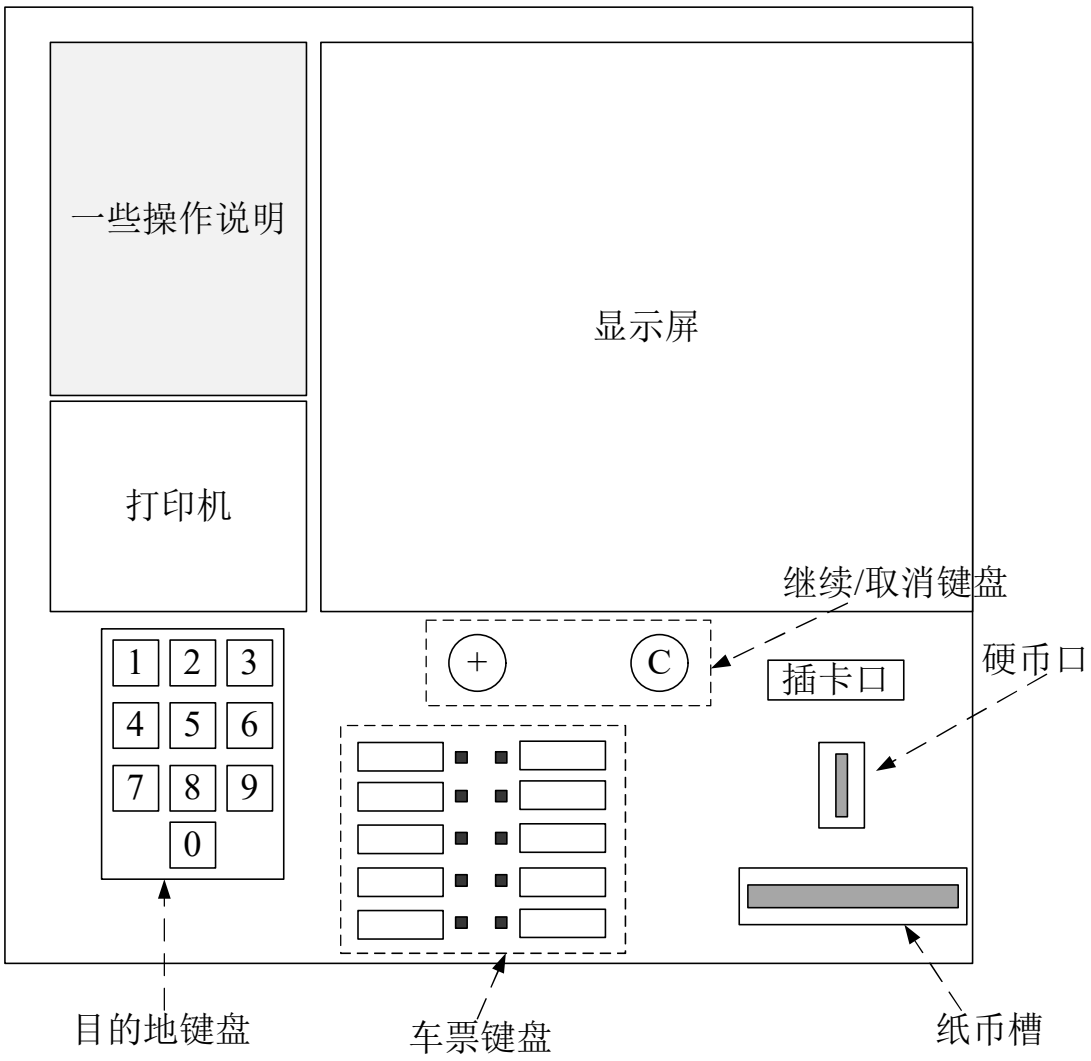
图形(Shape)可分为圆形(Circle)、矩形(Rectangle)、椭圆形(Ellipse)等具体图形，在 Shape 类中提供了一个抽象的 draw()方法用于绘制图形，而在具体的图形类中实现该抽象 draw()方法。

提供一个图形工厂类(ShapeFactory)，该类提供一个静态方法 createShape(char type)，其返回类型为 Shape，参数 type 为所需绘制图形对应的代码，如“c”表示圆形，“r”表示矩形，“e”表示椭圆形，在 createShape()方法中，可以使用条件语句来判断所需绘制图形的类型，并根据参数的不同返回不同的具体形状对象。

参考答案：



2．某运输公司决定为新的售票机开发车票销售的控制软件，下图给出了售票机的面板示意图以及相关的控制部件。



售票机相关部件的作用如下所述：

(1) 目的地键盘用来输入行程目的地的代码（例如，200 表示总站）。

(2) 乘客可以通过车票键盘选择车票种类（单程票、多次往返票和座席种类）。

(3) 继续/取消键盘上的取消按钮用于取消购票过程，继续按钮允许乘客连续购买多张票。

(4) 显示屏显示所有的系统输出和用户提示信息。

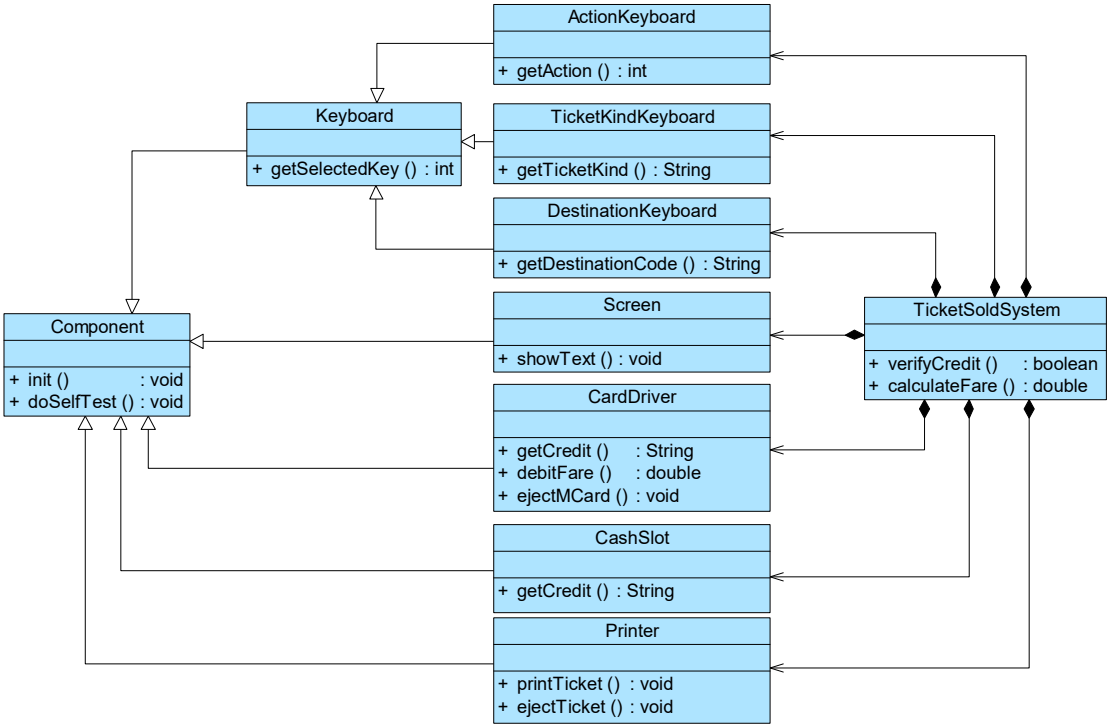
(5) 插卡口接受 MCard（现金卡），硬币口和纸币槽接受现金。

(6) 打印机用于输出车票。

(7) 所有部件均可实现自检并恢复到初始状态。

现采用面向对象方法开发该系统，使用 UML 进行建模，绘制该系统的核心类图并尽量分析出每一个类所包含的方法。

参考答案：



类说明：

类　名	说　明
Component	抽象部件类，所有部件类的父类
Keyboard	抽象键盘类
ActionKeyboard	继续/取消键盘类
TicketKindKeyboard	车票种类键盘类
DestinationKeyboard	目的地键盘类
Screen	显示屏类
CardDriver	卡驱动器类
CashSlot	现金（硬币/纸币）槽类
Printer	打印机类
TicketSoldSystem	售票系统类

方法说明：

方法名	说　明
Component 的 init()方法	初始化部件
Component 的 doSeltTest()方法	自检
Keyboard 的 getSelectedKey()方法	获取按键值
ActionKeyboard 的 getAction()方法	继续/取消键盘事件处理
TicketKindKeyboard 的 getTicketKind()方法	车票种类键盘事件处理
DestinationKeyboard 的 getDestinationCode()方法	目的地键盘事件处理
Screen 的 showText()方法	显示信息
CardDriver 的 getCredit()方法	获取金额
CardDriver 的 debitFare()方法	更新卡余额
CardDriver 的 ejectMCard()方法	退卡
CashSlot 的 getCredit()方法	获取金额

Printer 的 printTicket()方法	打印车票
Printer 的 ejectTicket()方法	出票
TicketSoldSystem 的 verifyCredit()方法	验证金额
TicketSoldSystem 的 calculateFare()方法	计算费用

3. 某基于 C/S 的即时聊天系统的注册和登录模块功能描述如下：

(1) 注册功能：用户通过注册界面(RegisterForm)输入新帐号，系统检测该帐号是否已存在，如果不存在则可注册成功，否则提示“帐号已存在”，用户再次输入帐号；用户输入其他个人信息；系统保存用户个人信息；用户个人信息包括帐号、密码、姓名、性别、年龄、电话、电子邮箱等。

(2) 登录功能：用户通过登录界面(LoginForm)输入账号和密码，系统将输入的账号和密码与存储在数据库(User)表中的用户信息进行比较，验证用户输入是否正确，如果输入正确则进入主界面(MainForm)，否则提示“输入错误”。

现对这两个模块进行设计，要求如下：

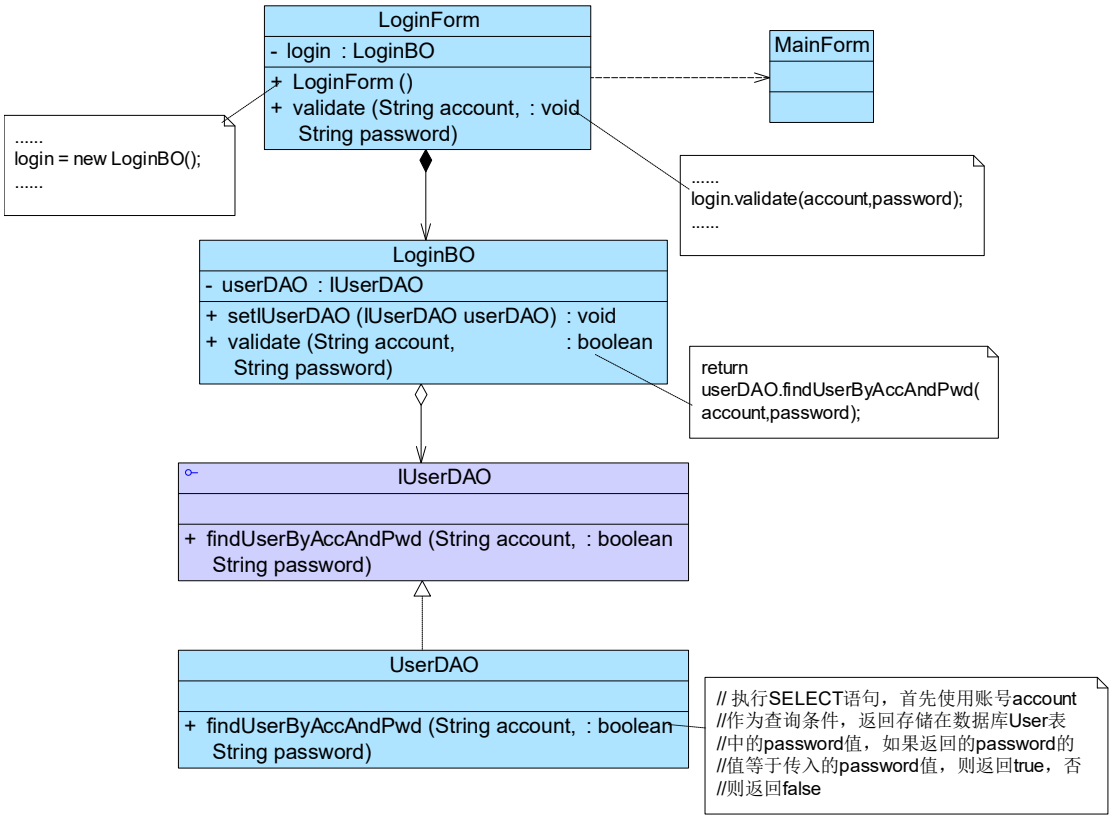
(1) 根据以上描述绘制类图，要求分析出每个类中的主要方法；

(2) 需要提供独立的业务逻辑层类和数据访问类，考虑到数据库的移植性，需提供抽象的数据访问接口；

(3) 尽量使用数据传输对象(DTO)来传递参数，减少方法中参数的个数；

(4) 将所得到的类分包，绘制相应的包图，要求通过包图来体现系统的分层架构。

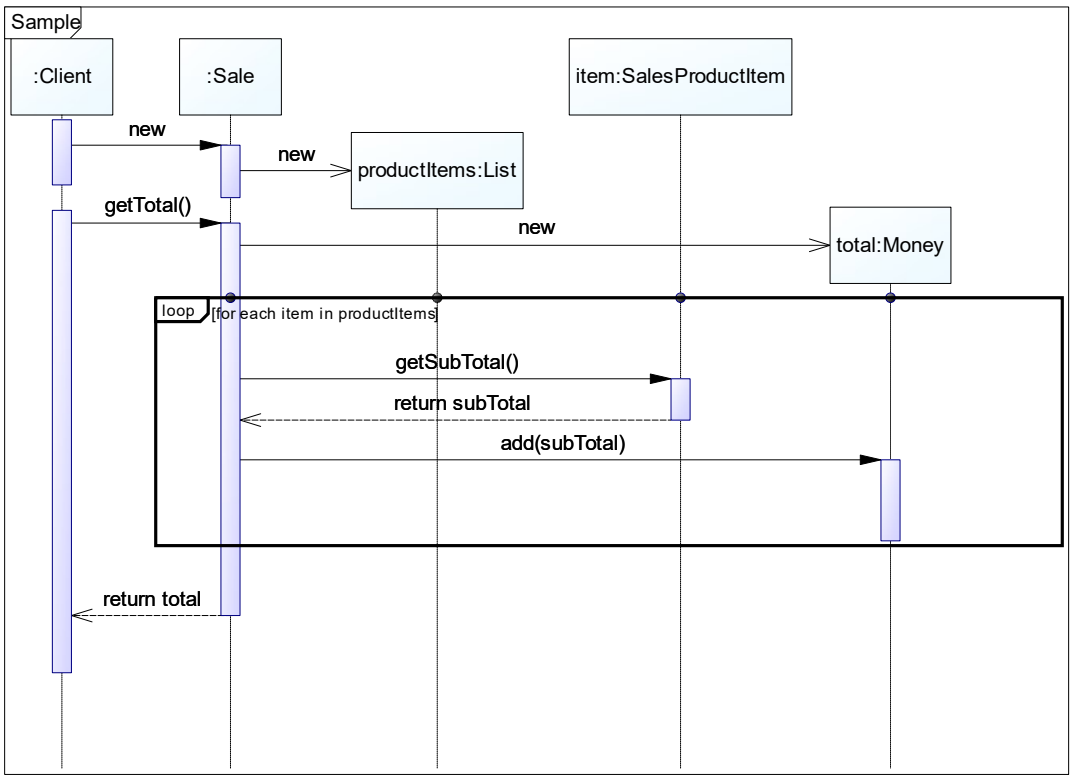
参考类图（登录模块）：



4. 根据以下 Java 源代码绘制相应的顺序图：

```
public class Sale {
    private List<SalesProductItem> productItems = new ArrayList<SalesProductItem>;

    public Money getTotal() {
        Money total = new Money();
        Money subTotal = null;

        for (SalesProductItem item : productItems) {
            subTotal = item.getSubTotal();
            total.add(subTotal);
        }

        return total;
    }
}
```

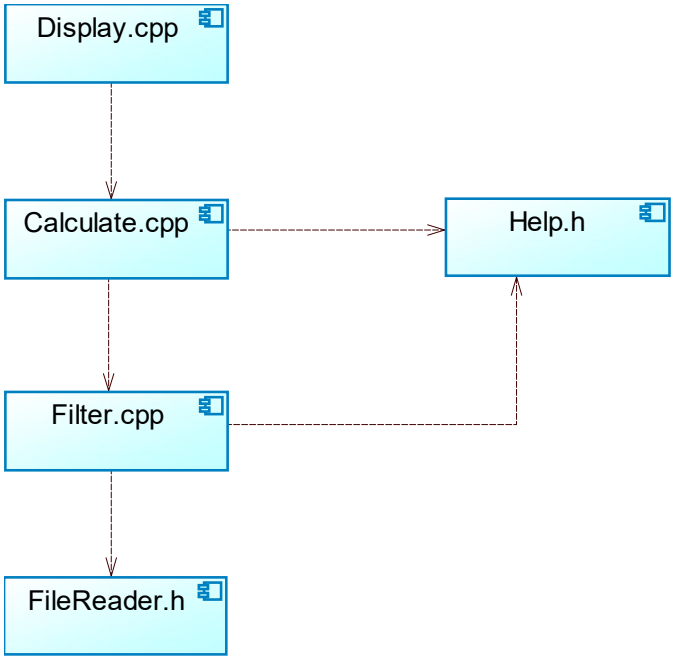
参考答案：



5. 根据以下描述，使用构件图对所述源代码文件进行建模：

Help.h 是一个头文件，被 Calculate.cpp 和 Filter.cpp 引用，而 Calculate.cpp 又引用了 Filter.cpp。此外，Filter.cpp 还引用了头文件 FileReader.h，Display.cpp 又依赖于 Calculate.cpp 的运算结果。

参考答案：



6. 某电子商务网站的硬件部署情况如下：

(1) 客户可通过浏览器访问该网站，以实现查询商品、在线购物、注册、登录等功能；

(2) 电子商务网站的界面文件和业务逻辑代码以 war 文件的方式部署在一台 Web 服务器上，该服务器使用 Tomcat 作为 Web 服务器中间件；

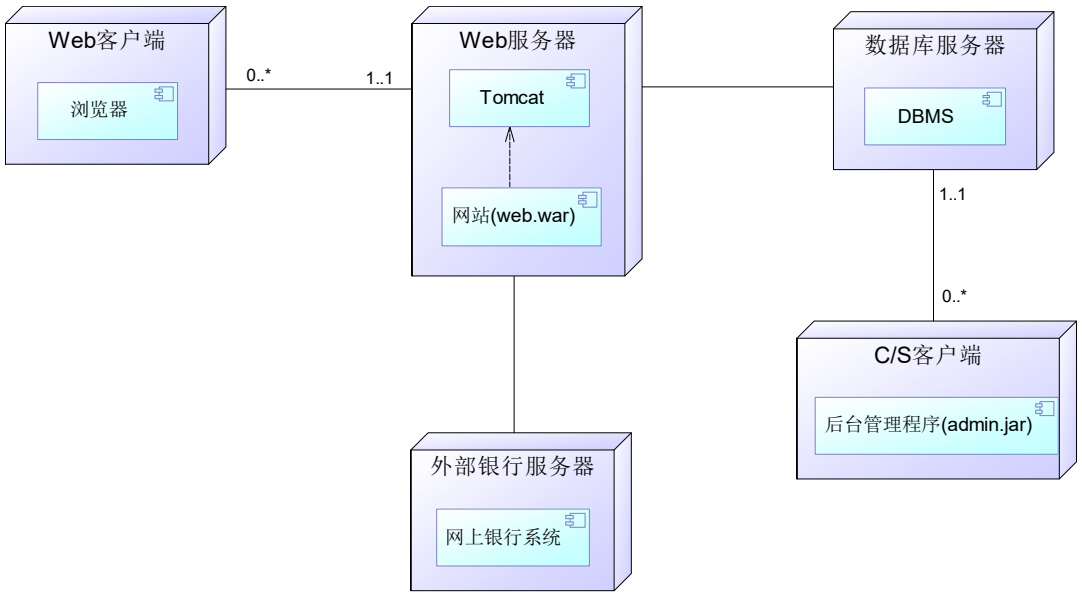
(3) 为了降低 Web 服务器的负载，该网站的数据库部署在另一台服务器中，Web 服务器通过网络连接数据库服务器；

(4) 为了实现在线支付功能，网站需要与银行提供的网上银行系统进行通信；

(5) 考虑到系统的安全性，系统的后台管理通过 C/S 方式来实现，后台管理程序打包成 jar 文件，管理员在自己的工作电脑中通过该 jar 文件中的数据库访问模块直接访问数据库服务器，以实现对网站数据的增删改查等操作和管理。

根据以上描述绘制该电子商务网站的部署图。

参考答案：



三、实验方法

1. 从系统中识别类并将分析类映射到设计类，将类封装为包，绘制包图，并通过包图体现系统的分层架构；
2. 使用顺序图描述对象之间的交互次序，将类图和顺序图映射为源代码；
3. 使用构件图描述物理文件及其相互关系，使用部署图描述硬件结构及其驻留构件；
4. 使用类图、包图、顺序图、构件图、部署图等构造软件系统的设计模型。

四、实验步骤

1. 根据描述绘制类图并正向工程生成源代码；
2. 分析车票销售控制软件实例，绘制该系统的核心类图；
3. 分析某基于 C/S 的即时聊天系统的注册和登录模块的功能，绘制对应的类图和包图；
4. 根据源代码绘制相应的顺序图；
5. 根据描述，使用构件图对所述源代码文件进行建模；
6. 根据某电子商务网站的硬件部署情况描述，绘制对应的部署图。

五、实验结果

注：有程序的要求附上程序源代码，有图表的要有截图并有相应的文字说明和分析。

1. 绘制并提交对应的类图，提交正向工程生成的源代码；
2. 绘制并提交车票销售控制软件的核心类图；
3. 绘制并提交即时聊天系统的注册和登录模块的功能的类图和包图；
4. 绘制并提交源代码所相应的顺序图；
5. 绘制并提交对所述源代码文件进行建模后所得到的构件图；
6. 绘制并提交电子商务网站的部署图。

六、实验小结

给出本次实验的体会，如学会了什么，遇到哪些问题，如何解决这些问题，存在哪些有待改进的地方。

实验 3《创建型模式实验》

实验学时：_____ 实验地点：_____ 实验日期：_____

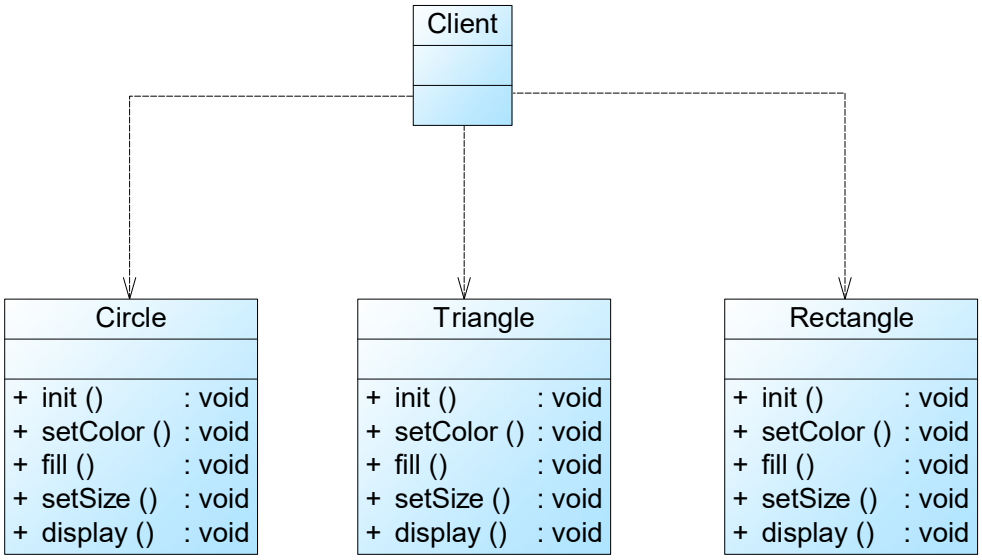
一、实验目的

1．熟练使用面向对象设计原则对系统进行重构；

2．熟练使用 PowerDesigner 和任意一种面向对象编程语言实现几种常见的创建型设计模式，包括简单工厂模式、工厂方法模式、抽象工厂模式、原型模式和单例模式，理解每一种设计模式的模式动机，掌握模式结构，学习如何使用代码实现这些模式。

二、实验内容

1．在某图形库 API 中提供了多种矢量图模板，用户可以基于这些矢量图创建不同的显示图形，图形库设计人员设计的初始类图如下所示：



在该图形库中，每个图形类（如 Circle、Triangle 等）的 init()方法用于初始化所创建的图形，setColor()方法用于给图形设置边框颜色，fill()方法用于给图形设置填充颜色，setSize()方法用于设置图形的大小，display()方法用于显示图形。

客户类(Client)在使用该图形库时发现存在如下问题：

① 由于在创建窗口时每次只需要使用图形库中的一种图形，因此在更换图形时需要修改客户类源代码；

② 在图形库中增加并使用新的图形时需要修改客户类源代码；

③ 客户类在每次使用图形对象之前需要先创建图形对象，有些图形的创建过程较为复杂，导致客户类代码冗长且难以维护。

现需要根据面向对象设计原则对该系统进行重构，要求如下：

① 隔离图形的创建和使用，将图形的创建过程封装在专门的类中，客户类在使用图形

时无须直接创建图形对象，甚至不需要关心具体图形类类名；

② 客户类能够方便地更换图形或使用新增图形，无须针对具体图形类编程，符合开闭原则。

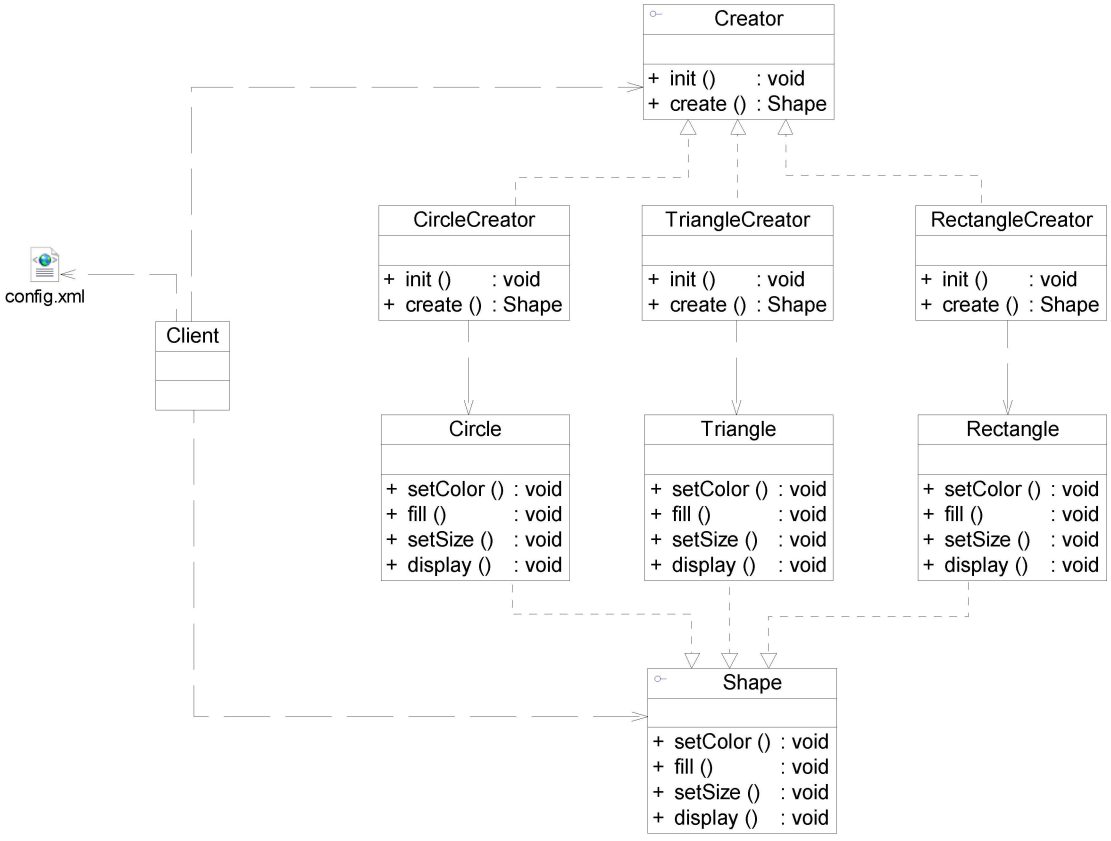
绘制重构之后的类图并说明在重构过程中所运用的面向对象设计原则。

参考答案：本题可以通过单一职责原则和依赖倒转原则进行重构，具体过程可分为如下两步：

(1) 由于图形对象的创建过程较为复杂，因此可以将创建过程封装在专门的类中（这种专门用于创建对象的类称为工厂类），将对象的创建和使用分离，符合单一职责原则；

(2) 引入抽象的图形类 Shape，并对应提供一个抽象的创建类，将具体图形类作为 Shape 的子类，而具体的图形创建类作为抽象创建类的子类，根据依赖倒转原则，客户端针对抽象图形类和抽象图形创建类编程，而将具体的图形创建类类名存储在配置文件中。

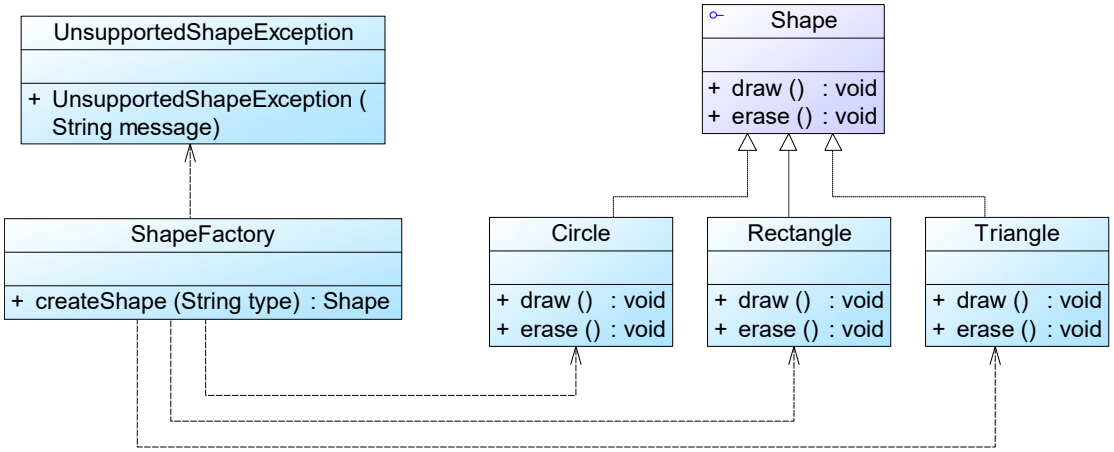
重构之后的类图如下所示：



通过上述重构，在抽象类 Creator 中声明了创建图形对象的方法 create()，在其子类中实现了该方法，用于创建具体的图形对象。客户端针对抽象 Creator 编程，而将其具体子类类名存储在配置文件 config.xml 中，如果需要更换图形，只需在配置文件中更改 Creator 的具体子类类名即可；如果需要增加图形，则对应增加一个新的 Creator 子类用于创建新增图形对象，再修改配置文件，在配置文件中存储新的图形创建类类名。更换和增加图形都无须修改源代码，完全符合开闭原则。（注：本重构方法即为工厂方法模式。）

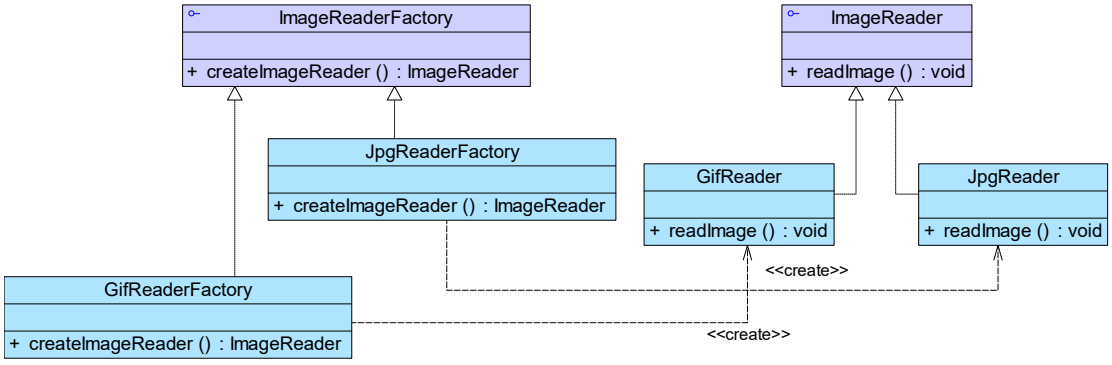
2．使用简单工厂模式设计一个可以创建不同几何形状(Shape)，如圆形(Circle)、矩形(Rectangle)和三角形(Triangle)等的绘图工具类，每个几何图形均具有绘制 Draw()和擦除 Erase()两个方法，要求在绘制不支持的几何图形时，抛出一个 UnsupportedShapeException 异常，绘制类图并编程模拟实现。

参考答案：

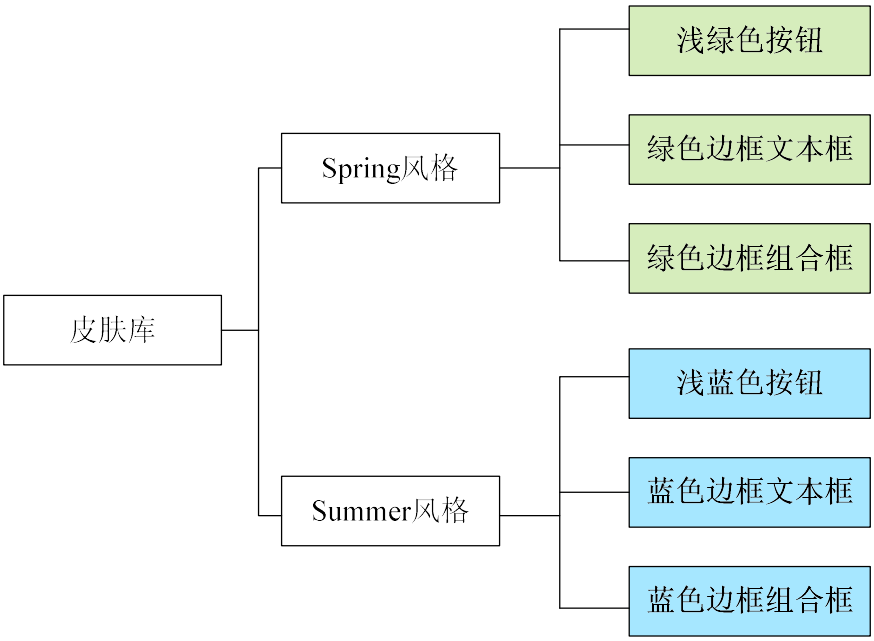


3. 使用工厂方法模式设计一个程序用来读取各种不同类型的图片格式，针对每一种图片格式都设计一个图片读取器(ImageReader)，例如 GIF 图片读取器(GifReader)用于读取 GIF 格式的图片、JPG 图片读取器(JpgReader)用于读取 JPG 格式的图片。需充分考虑系统的灵活性和可扩展性。

参考答案：

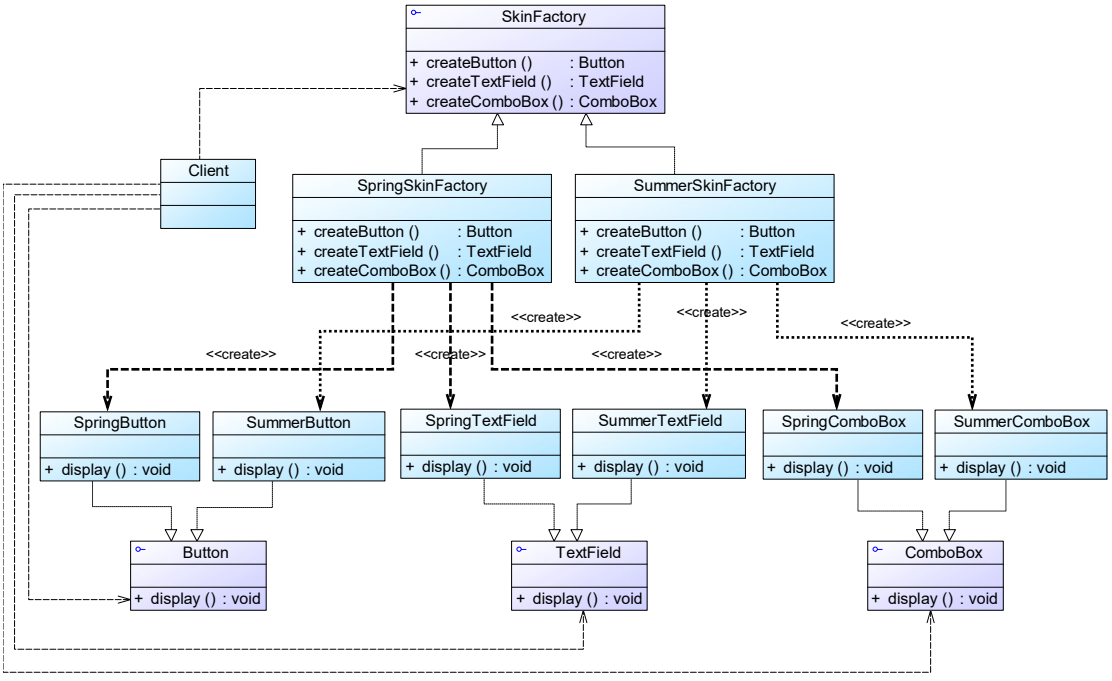


4. 某软件公司欲开发一套界面皮肤库，可以对桌面软件进行界面美化。不同的皮肤将提供视觉效果不同的按钮、文本框、组合框等界面元素，其结构如下图所示：



该皮肤库需要具备良好的灵活性和可扩展性，用户可以自由选择不同的皮肤，开发人员可以在不修改既有代码的基础上增加新的皮肤。试使用抽象工厂模式设计该皮肤库，绘制类图并编程模拟实现。

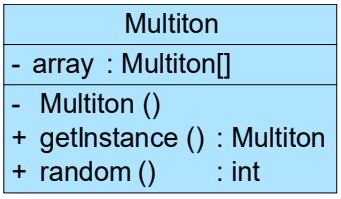
参考答案：



5. 某公司欲开发一个即时聊天软件，用户可以使用该软件同时与多位好友聊天。由于在私聊时系统将产生多个聊天窗口，为了提高聊天窗口的创建效率，要求根据第一个窗口快速创建其他窗口。试使用原型模式设计并模拟实现该即时聊天软件的聊天窗口，要求提供浅克隆和深克隆两套实现方案。

6. 使用单例模式的思想实现多例模式，确保系统中某个类的对象只能存在有限个，例如两个或三个，设计并编写代码实现一个多例类。

参考答案：



多例类 Multiton 的代码如下所示：

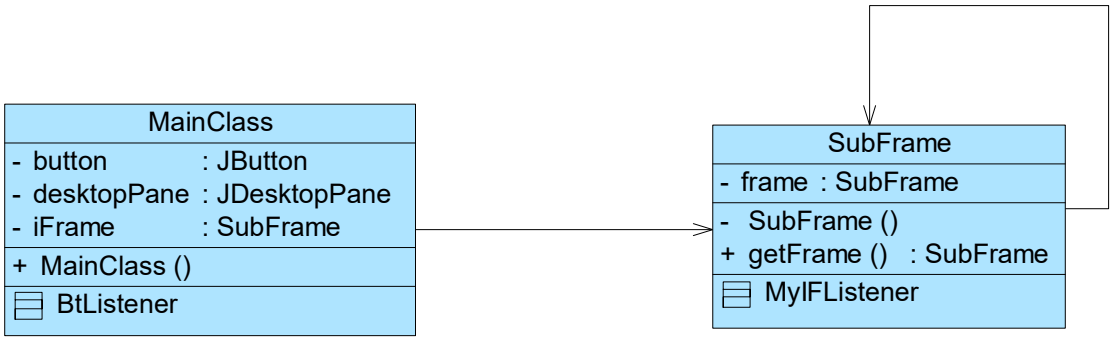
```
import java.util.*;

public class Multiton
{
    //定义一个数组用于存储四个实例
    private static Multiton[] array = {new Multiton(), new Multiton(), new Multiton(), new Multiton()};
    //私有构造函数
    private Multiton()
    {
    }
    //静态工厂方法，随机返回数组中的一个实例
    public static Multiton getInstance()
    {
        return array[random()];
    }
    //随机生成一个整数作为数组下标
    public static int random()
    {
        Date d = new Date();
        Random random = new Random();
        int value = Math.abs(random.nextInt());
        value = value % 4;
        return value;
    }
    public static void main(String args[])
    {
        Multiton m1,m2,m3,m4;
        m1 = Multiton.getInstance();
        m2 = Multiton.getInstance();
        m3 = Multiton.getInstance();
        m4 = Multiton.getInstance();

        System.out.println(m1==m2);
```

```
            System.out.println(m1==m3);
            System.out.println(m1==m4);
        }
}
```

7. 使用单例模式设计一个多文档窗口（注：在 Java AWT/Swing 开发中可使用 JDesktopPane 和 JInternalFrame 来实现），要求在主窗体中某个内部子窗体只能实例化一次，即只能弹出一个相同的子窗体，如下图所示，编程实现该功能。

（注：用 C#或 C++实现类似功能也可以）

参考答案：



在本练习中，SubFrame 类充当单例类，在其中定义了静态工厂方法 getFrame()，代码如下所示：

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

//子窗口：单例类
class SubFrame extends JInternalFrame
{
    private static SubFrame frame;//静态实例

    //私有构造函数
    private SubFrame()
```

```
    {
        super("子窗体", true, true, true, false);
        this.setLocation(20,20);  //设置内部窗体位置
        this.setSize(200,200);  //设置内部窗体大小
        this.addInternalFrameListener(new MyIFListener());//监听窗体事件
        this.setVisible(true);
    }

    //工厂方法，返回窗体实例
    public static SubFrame getFrame()
    {
        //如果窗体对象为空，则创建窗体，否则直接返回已有窗体
        if(frame==null)
        {
            frame=new SubFrame();
        }
        return frame;
    }

    //事件监听器
    class MyIFListener extends InternalFrameAdapter
    {
        //子窗体关闭时，将窗体对象设为 null
        public void internalFrameClosing(InternalFrameEvent e)
        {
            if(frame!=null)
            {
                frame=null;
            }
        }
    }
}

//客户端测试类
class MainClass extends JFrame
{
    private JButton button;
    private JDesktopPane desktopPane;
    private SubFrame iFrame=null;

    public MainClass()
    {
        super("主窗体");
        Container c=this.getContentPane();
```

```
        c.setLayout(new BorderLayout());

        button=new JButton("点击创建一个内部窗体");
        button.addActionListener(new BtListener());
        c.add(button, BorderLayout.SOUTH);

        desktopPane = new JDesktopPane(); //创建 DesktopPane
        c.add(desktopPane);

        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);
        this.setSize(400,400);
        this.show();
    }

    //事件监听器
    class BtListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            if(iFrame!=null)
            {
                desktopPane.remove(iFrame);
            }
            iFrame=SubFrame.getFrame();
            desktopPane.add(iFrame);
        }
    }

    public static void main(String[] args)
    {
        new MainClass();
    }
}
```

在本实例中，SubFrame 类是 JInternalFrame 类的子类，在 SubFrame 类中定义了一个静态的 SubFrame 类型的实例变量，在静态工厂方法 getFrame()中创建了 SubFrame 对象并将其返回。在 MainClass 类中使用了该单例类，确保子窗口在当前应用程序中只有唯一一个实例，即只能弹出一个子窗口。

三、实验方法

1. 选择合适的面向对象设计原则对系统进行重构，正确无误地绘制重构之后的类图；

2. 结合实例，正确无误地绘制简单工厂模式、工厂方法模式、抽象工厂模式、原型模式和单例模式的模式结构图；

3. 使用任意一种面向对象编程语言实现简单工厂模式、工厂方法模式、抽象工厂模式、原型模式和单例模式实例，代码运行正确无误。

四、实验步骤

1. 选择合适的面向对象设计原则对系统进行重构，使用 PowerDesigner 绘制重构之后的类图；

2. 结合实例，使用 PowerDesigner 绘制简单工厂模式实例结构图并用面向对象编程语言实现该模式实例；

3. 结合实例，使用 PowerDesigner 绘制工厂方法模式实例结构图并用面向对象编程语言实现该模式实例；

4. 结合实例，使用 PowerDesigner 绘制抽象工厂模式实例结构图并用面向对象编程语言实现该模式实例；

5. 结合实例，使用 PowerDesigner 绘制原型模式实例结构图并用面向对象编程语言实现该模式实例；

6. 结合实例，使用 PowerDesigner 绘制多例模式实例结构图并用面向对象编程语言实现该模式实例；

7. 结合实例，使用 PowerDesigner 绘制单例模式实例结构图并用面向对象编程语言实现该模式实例。

五、实验结果

注：有程序的要求附上程序源代码，有图表的要有截图并有相应的文字说明和分析。

1. 提供重构之后的类图，简要说明重构过程中所运用的面向对象设计原则；

2. 提供简单工厂模式实例结构图及实现代码；

3. 提供工厂方法模式实例结构图及实现代码；

4. 提供抽象工厂模式实例结构图及实现代码；

5. 提供原型模式实例结构图及实现代码（包括浅克隆和深克隆）；

6. 提供多例模式实例结构图及实现代码；

7. 提供单例模式实例结构图及实现代码。

六、实验小结

给出本次实验的体会，如学会了什么，遇到哪些问题，如何解决这些问题，存在哪些有待改进的地方。

实验 4《结构型模式实验》

实验学时：_____ 实验地点：_____ 实验日期：_____

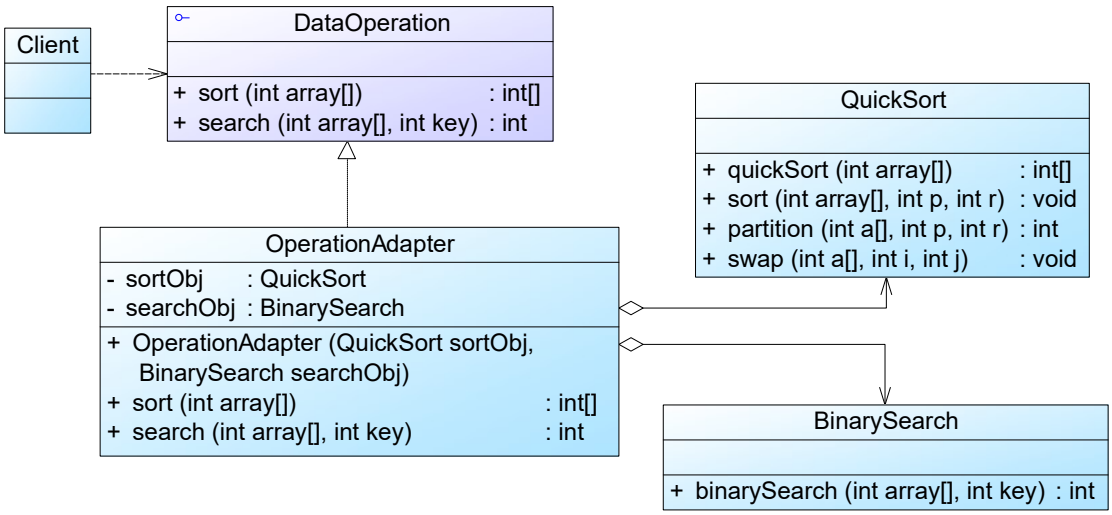
一、实验目的

熟练使用 PowerDesigner 和任意一种面向对象编程语言实现几种常见的结构型设计模式，包括适配器模式、桥接模式、组合模式、装饰模式、外观模式和代理模式，理解每一种设计模式的模式动机，掌握模式结构，学习如何使用代码实现这些模式。

二、实验内容

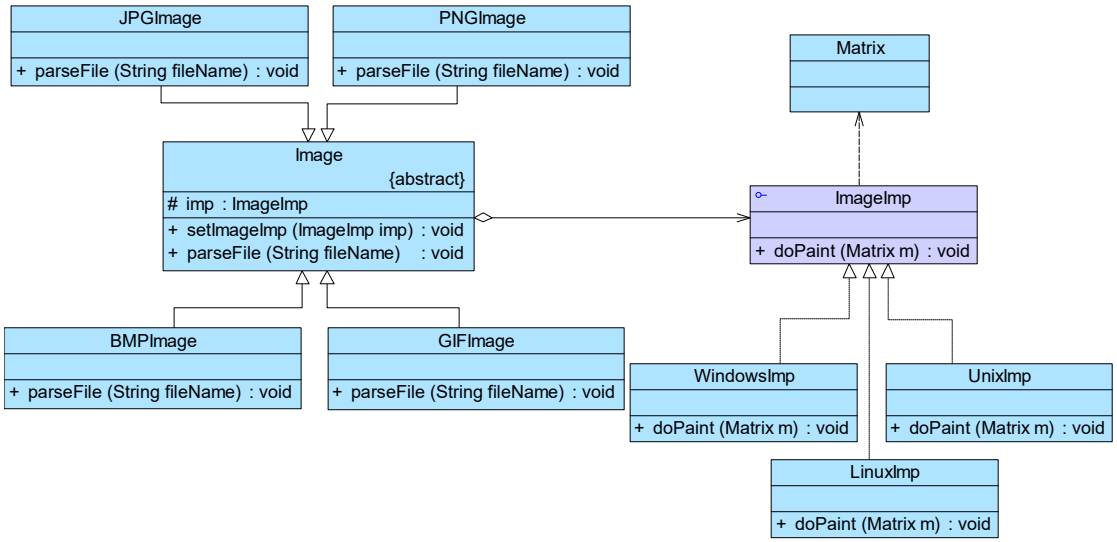
1．现有一个接口 DataOperation 定义了排序方法 sort(int[]) 和查找方法 search(int[], int)，已知类 QuickSort 的 quickSort(int[])方法实现了快速排序算法，类 BinarySearch 的 binarySearch(int[], int)方法实现了二分查找算法。试使用适配器模式设计一个系统，在不修改源代码的情况下将类 QuickSort 和类 BinarySearch 的方法适配到 DataOperation 接口中。绘制类图并编程实现。（要求实现快速排序和二分查找）

参考答案：

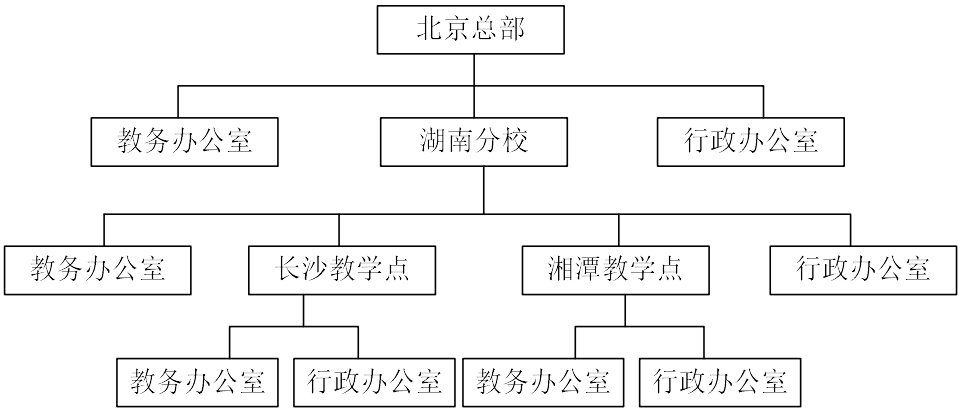


2. 某软件公司欲开发一个跨平台图像浏览系统，要求该系统能够显示 BMP、JPG、GIF、PNG 等多种格式的文件，并且能够在 Windows、Linux、Unix 等多个操作系统上运行。系统首先将各种格式的文件解析为像素矩阵(Matrix)，然后将像素矩阵显示在屏幕上，在不同的操作系统中可以调用不同的绘制函数来绘制像素矩阵。系统需具有较好的扩展性，以便在将来支持新的文件格式和操作系统。试使用桥接模式设计该跨平台图像浏览系统，绘制类图并编程模拟实现。

参考答案：

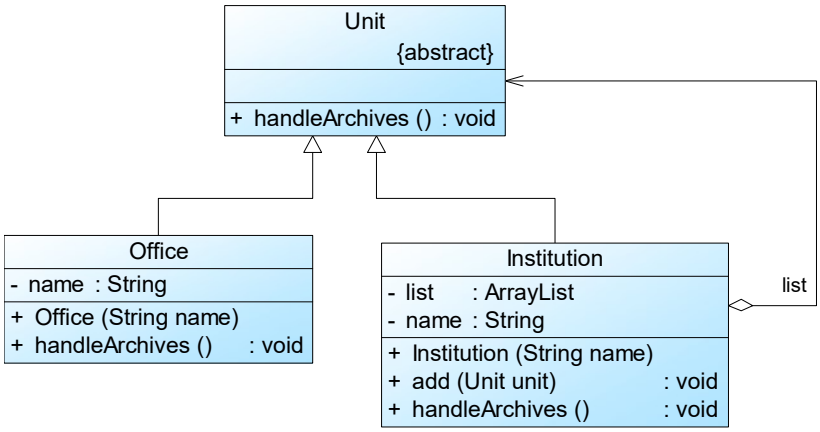


3. 某教育机构组织结构如下图所示：



在该教育机构的 OA 系统中可以给各级办公室下发公文，试采用组合模式设计该机构的组织结构，绘制相应的类图并编程模拟实现，在客户端代码中模拟下发公文。

参考答案：参考类图如下图所示。



参考类图

此处使用了安全组合模式，其中，Unit 充当抽象构件角色，Office 充当叶子构件角色，Institution 充当容器构件角色。本实例代码如下：

```
abstract class Unit {
    public abstract void handleArchives();
```

```
}

class Office extends Unit {
    private String name;
    public Office(String name) {
        this.name = name;
    }
    public void handleArchives() {
        System.out.println(this.name + "处理公文！");
    }
}

class Institution extends Unit {
    private ArrayList list = new ArrayList();
    private String name;
    public Institution(String name) {
        this.name = name;
    }
    public void add(Unit unit) {
        list.add(unit);
    }
    public void handleArchives() {
        System.out.println(this.name + "接收并下发公文：");
        for(Object obj : list) {
            ((Unit)obj).handleArchives();
        }
    }
}
```

在客户类中创建树形结构，代码如下所示：

```
class Client {
    public static void main(String args[]) {
        Institution bjHeadquarters,hnSubSchool,csTeachingPost,xtTeachingPost;
        Unit tOffice1,tOffice2,tOffice3,tOffice4,aOffice1,aOffice2,aOffice3,aOffice4;
        bjHeadquarters = new Institution("北京总部");
        hnSubSchool = new Institution("湖南分校");
        csTeachingPost = new Institution("长沙教学点");
        xtTeachingPost = new Institution("湘潭教学点");
        tOffice1 = new Office("北京教务办公室");
        tOffice2 = new Office("湖南教务办公室");
        tOffice3 = new Office("长沙教务办公室");
        tOffice4 = new Office("湘潭教务办公室");
        aOffice1 = new Office("北京行政办公室");
        aOffice2 = new Office("湖南行政办公室");
        aOffice3 = new Office("长沙行政办公室");
```

```
            aOffice4 = new Office("湘潭行政办公室");
            csTeachingPost.add(tOffice3);
            csTeachingPost.add(aOffice3);
            xtTeachingPost.add(tOffice4);
            xtTeachingPost.add(aOffice4);
            hnSubSchool.add(csTeachingPost);
            hnSubSchool.add(xtTeachingPost);
            hnSubSchool.add(tOffice2);
            hnSubSchool.add(aOffice2);
            bjHeadquarters.add(hnSubSchool);
            bjHeadquarters.add(tOffice1);
            bjHeadquarters.add(aOffice1);
            bjHeadquarters.handleArchives();
        }
}
```

4. 某咖啡店在卖咖啡时，可以根据顾客的要求在其中加入各种配料，咖啡店会根据所加入的配料来计算总费用。咖啡店所供应的咖啡及配料的种类和价格如表 1 所示：

表 1 咖啡及配料的种类和价格表

咖啡	价格/杯(￥)		配料	价格/份(￥)
浓缩咖啡(Espresso)	25		摩卡(Mocha)	10
混合咖啡(House Blend)	30		奶泡(Whip)	8
重烘焙咖啡(Dark Roast)	20		牛奶(Milk)	6

试使用装饰模式来为该咖啡店设计一个程序以实现计算费用的功能，输出每种饮料的详细描述及花费。

输出结果示例如下：

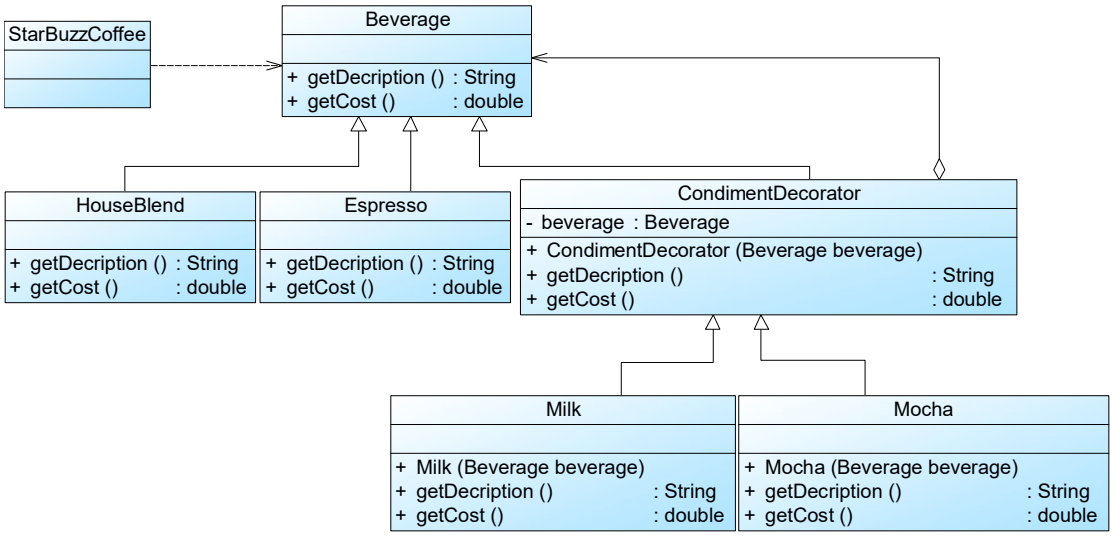
```
浓缩咖啡，摩卡，牛奶  ￥41
```

饮料类 Beverage 代码如下：

```
abstract class Beverage
{
    public abstract string GetDescription();
    public abstract int GetCost();
}
```

要求画出对应的类图，并编程模拟实现。

参考答案：



在类图中，Beverage 充当抽象组件，HouseBlend 和 Espresso 充当具体组件，CondimentDecorator 充当抽象装饰器，Milk 和 Mocha 充当具体装饰器，StarBuzzCoffee 充当客户端。本题完整代码示例如下所示：

```
abstract class Beverage       //抽象组件
{
      public abstract String getDescription();
      public abstract double getCost();
}

class HouseBlend extends Beverage       //具体组件
{
      public String getDescription()
      {
            return "HouseBlend 咖啡";
      }
      public double getCost()
      {
            return 10.00;
      }
}

class Espresso extends Beverage       //具体组件
{
      public String getDescription()
      {
            return "Espresso 咖啡";
      }
      public double getCost()
      {
            return 20.00;
      }
```

```
}

class CondimentDecorator extends Beverage       //抽象装饰器
{
    private Beverage beverage;
    public CondimentDecorator(Beverage beverage)
    {
        this.beverage = beverage;
    }
    public String getDescription()
    {
        return beverage.getDescription();
    }
    public double getCost()
    {
        return beverage.getCost();
    }
}

class Milk extends CondimentDecorator       //具体装饰器
{
    public Milk(Beverage beverage)
    {
        super(beverage);
    }
    public String getDescription()
    {
        String decription = super.getDescription();
        return decription + "加牛奶";
    }
    public double getCost()
    {
        double cost = super.getCost();
        return cost + 2.0;
    }
}

class Mocha extends CondimentDecorator       //具体装饰器
{
    public Mocha(Beverage beverage)
    {
        super(beverage);
    }
    public String getDescription()
```

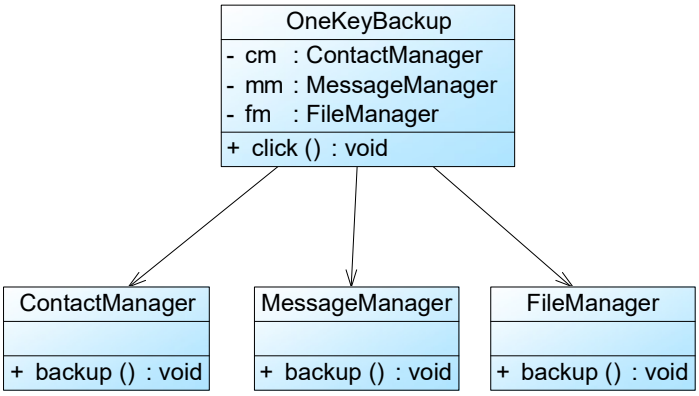
```
    {
        String decription = super.getDescription();
        return decription + "加摩卡";
    }
    public double getCost()
    {
        double cost = super.getCost();
        return cost + 3.0;
    }
}

class StarBuzzCoffee       //客户端测试类
{
    public static void main(String args[])
    {
        String decription;
        double cost;
        Beverage beverage_e;

        beverage_e = new Espresso();
        decription = beverage_e.getDescription();
        cost = beverage_e.getCost();
        System.out.println("饮料：" + decription);
        System.out.println("价格：" + cost);
        System.out.println("---------------------");

        Beverage beverage_mi;
        beverage_mi = new Milk(beverage_e);
        decription = beverage_mi.getDescription();
        cost = beverage_mi.getCost();
        System.out.println("饮料：" + decription);
        System.out.println("价格：" + cost);
        System.out.println("---------------------");

        Beverage beverage_mo;
        beverage_mo = new Mocha(beverage_mi);
        decription = beverage_mo.getDescription();
        cost = beverage_mo.getCost();
        System.out.println("饮料：" + decription);
        System.out.println("价格：" + cost);
        System.out.println("---------------------");
    }
}
//输出结果如下：
```

```
//饮料：Espresso 咖啡
//价格：20.0
//--------------------
//饮料：Espresso 咖啡加牛奶
//价格：22.0
//--------------------
//饮料：Espresso 咖啡加牛奶加摩卡
//价格：25.0
```

5. 某软件公司为新开发的智能手机控制与管理软件提供了一键备份功能，通过该功能可以将原本存储在手机中的通信录、短信、照片、歌曲等资料一次性全部拷贝到移动存储介质（例如 MMC 卡或 SD 卡）中。在实现过程中需要与多个已有的类进行交互，例如通讯录管理类、短信管理类等，为了降低系统的耦合度，试使用外观模式来设计并编程模拟实现该一键备份功能。

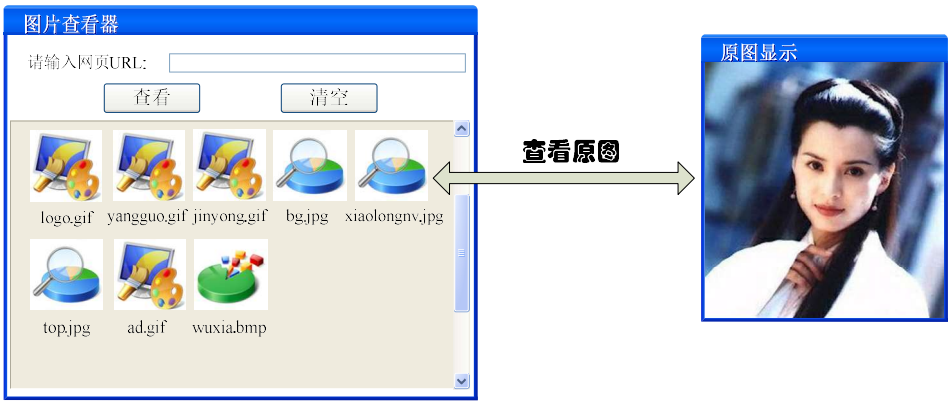
参考答案：

参考类图如下所示：



其中，OneKeyBackup 充当外观角色，ContactManager、MessageManager 和 FileManager 充当子系统角色。

6. 某软件公司欲开发一款基于 C/S 的网络图片查看器，具体功能描述如下：用户只需在图片查看器中输入网页 URL，程序将自动将该网页所有图片下载到本地，考虑到有些网页图片比较多，而且某些图片文件比较大，因此将先以图标的方式显示图片，不同类型的图片使用不同的图标，并且在图标下面标注该图片的文件名，用户单击图标后可查看真正的图片，界面效果如下图所示。试使用虚拟代理模式设计并实现该图片查看器。（注：可以结合多线程机制，使用一个线程显示小图标，同时启动另一个线程在后台加载原图。）



三、实验方法

1. 结合实例，正确无误地绘制适配器模式、桥接模式、组合模式、装饰模式、外观模式和代理模式的模式结构图；

2. 使用任意一种面向对象编程语言实现适配器模式、桥接模式、组合模式、装饰模式、外观模式和代理模式实例，代码运行正确无误。

四、实验步骤

1. 结合实例，使用 PowerDesigner 绘制适配器模式实例结构图并用面向对象编程语言实现该模式实例；

2. 结合实例，使用 PowerDesigner 绘制桥接模式实例结构图并用面向对象编程语言实现该模式实例；

3. 结合实例，使用 PowerDesigner 绘制组合模式实例结构图并用面向对象编程语言实现该模式实例；

4. 结合实例，使用 PowerDesigner 绘制装饰模式实例结构图并用面向对象编程语言实现该模式实例；

5. 结合实例，使用 PowerDesigner 绘制外观模式实例结构图并用面向对象编程语言实现该模式实例；

6. 结合实例，使用 PowerDesigner 绘制代理模式实例结构图并用面向对象编程语言实现该模式实例。

五、实验结果

注：有程序的要求附上程序源代码，有图表的要有截图并有相应的文字说明和分析。

1. 提供适配器模式实例结构图及实现代码；

2. 提供桥接模式实例结构图及实现代码；

3. 提供组合模式实例结构图及实现代码；

4. 提供装饰模式实例结构图及实现代码；

5. 提供外观模式实例结构图及实现代码；

6. 提供代理模式实例结构图及实现代码。

六、实验小结

给出本次实验的体会，如学会了什么，遇到哪些问题，如何解决这些问题，存在哪些有待改进的地方。

实验 5《行为型模式实验》

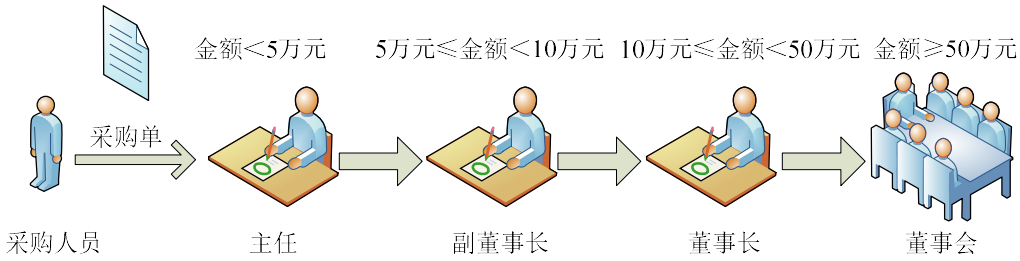
实验学时：_____ 实验地点：_____ 实验日期：_____

一、实验目的

熟练使用 PowerDesigner 和任意一种面向对象编程语言实现几种常见的行为型设计模式，包括职责链模式、命令模式、迭代器模式、中介者模式、备忘录模式、观察者模式、状态模式、策略模式、模板方法模式和访问者模式，理解每一种设计模式的模式动机，掌握模式结构，学习如何使用代码实现这些模式。

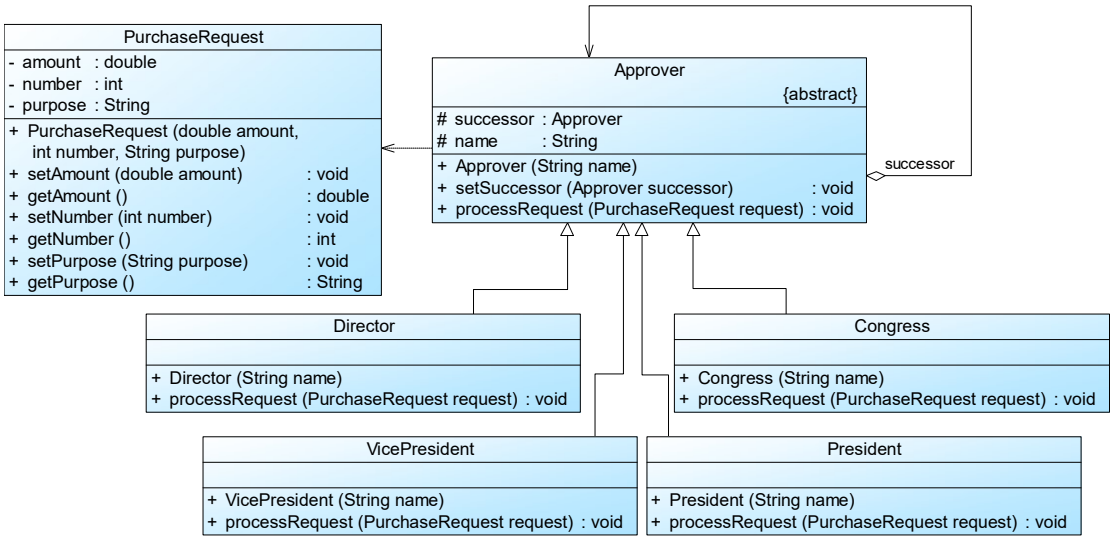
二、实验内容

1．某企业的 SCM(Supply Chain Management，供应链管理)系统中包含一个采购审批子系统。该企业的采购审批是分级进行的，即根据采购金额的不同由不同层次的主管人员来审批，主任可以审批 5 万元以下（不包括 5 万元）的采购单，副董事长可以审批 5 万元至 10 万元（不包括 10 万元）的采购单，董事长可以审批 10 万元至 50 万元（不包括 50 万元）的采购单，50 万元及以上的采购单就需要开董事会讨论决定。如下图所示：



试使用职责链模式设计并实现该系统。

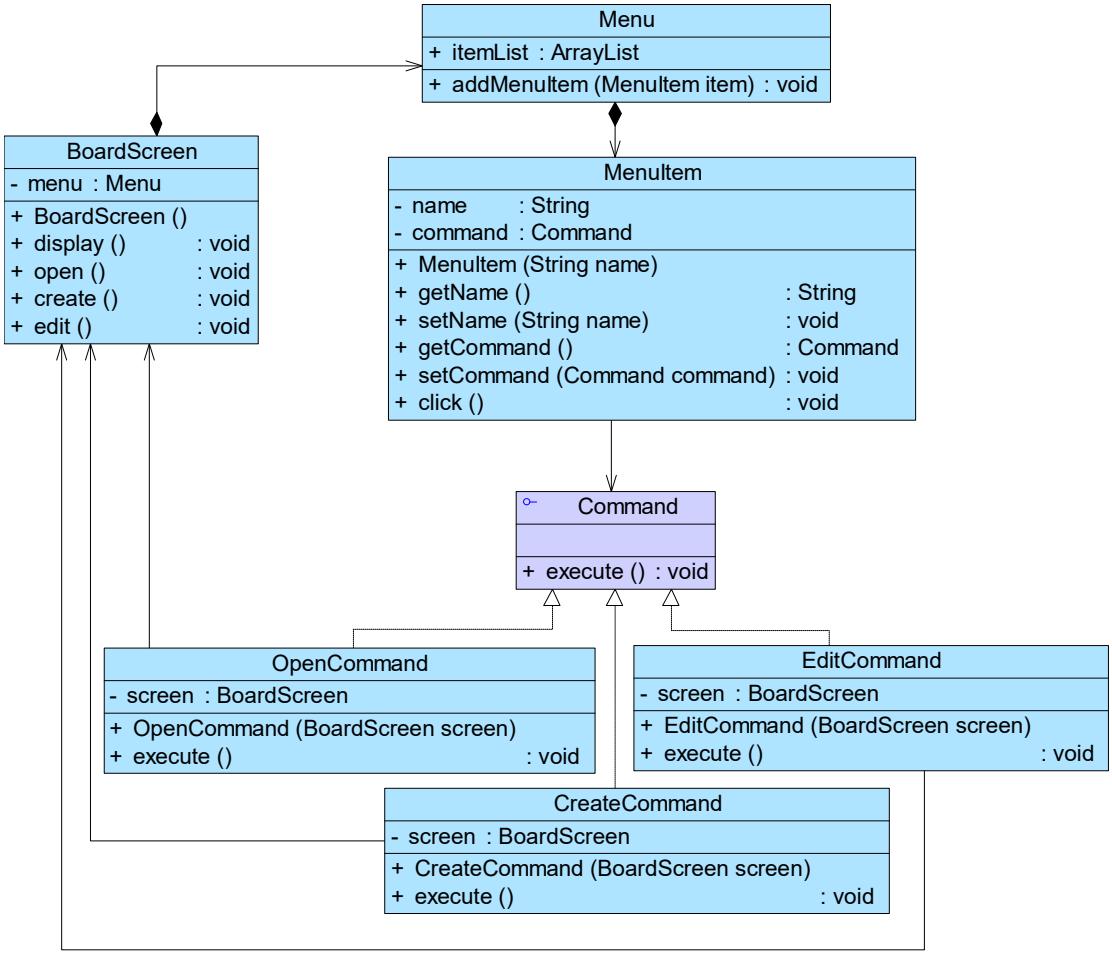
参考答案：



2. 某软件公司欲开发一个基于 Windows 平台的公告板系统。系统提供一个主菜单

(Menu)，在主菜单中包含了一些菜单项(MenuItem)，可以通过 Menu 类的 addMenuItem()方法增加菜单项。菜单项的主要方法是 click()，每一个菜单项包含一个抽象命令类，具体命令类包括 OpenCommand(打开命令)，CreateCommand(新建命令)，EditCommand(编辑命令)等，命令类具有一个 execute()方法，用于调用公告板系统界面类(BoardScreen)的 open()、create()、edit()等方法。现使用命令模式设计该系统，使得 MenuItem 类与 BoardScreen 类的耦合度降低，绘制类图并编程实现。

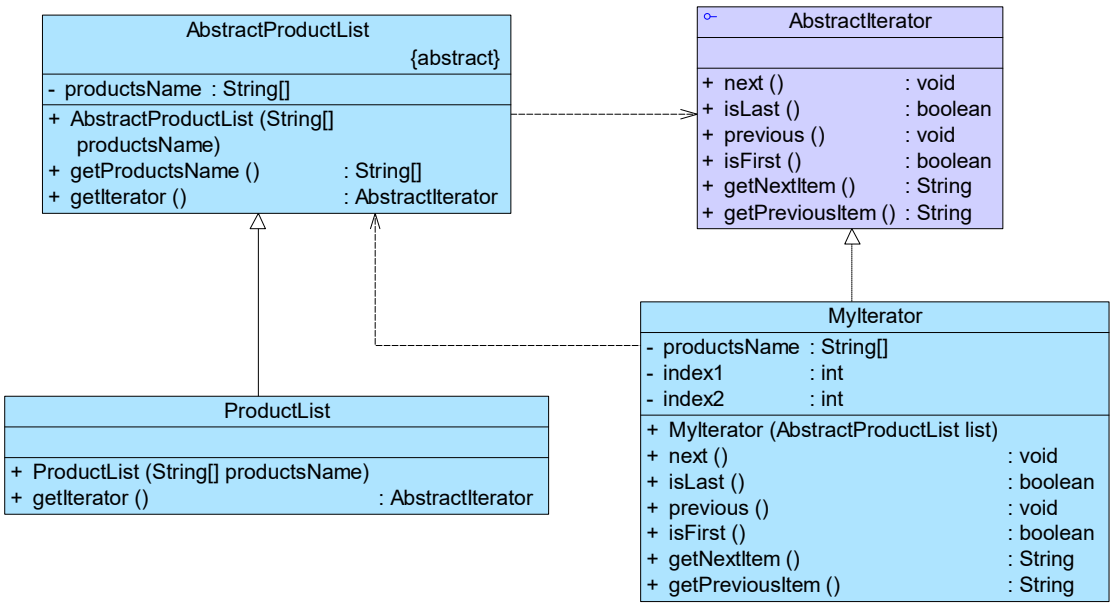
参考答案：



其中，BoardScreen 充当接收者角色，MenuItem 充当调用者角色，Command 充当抽象命令角色，OpenCommand、CreateCommand 和 EditCommand 充当具体命令角色。

3. 某商品管理系统的商品名称存储在一个字符串数组中，现需要自定义一个双向迭代器(MyIterator)实现对该商品名称数组的双向（前向和后向）遍历。绘制类图并编程实现（设计方案必须符合 DIP）。

参考答案：



抽象类 AbstractProductList 充当抽象聚合角色，ProductList 类充当具体聚合角色，AbstractIterator 接口充当抽象迭代器角色，MyIterator 类充当具体迭代器角色。

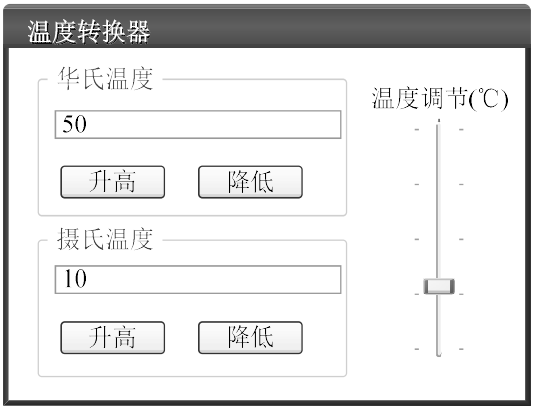
参考代码：

```
//抽象商品集合：抽象聚合类
abstract class AbstractProductList
{
    private String[] productsName;
    public AbstractProductList(String[] productsName)
    {
        this.productsName = productsName;
    }
    public String[] getProductsName()
    {
        return this.productsName;
    }
    public abstract AbstractIterator getIterator();
}

//商品集合：具体聚合类
class ProductList extends AbstractProductList
{
    public ProductList(String[] productsName)
    {
        super(productsName);
    }
    public AbstractIterator getIterator()
    {
        return new MyIterator(this);
    }
```

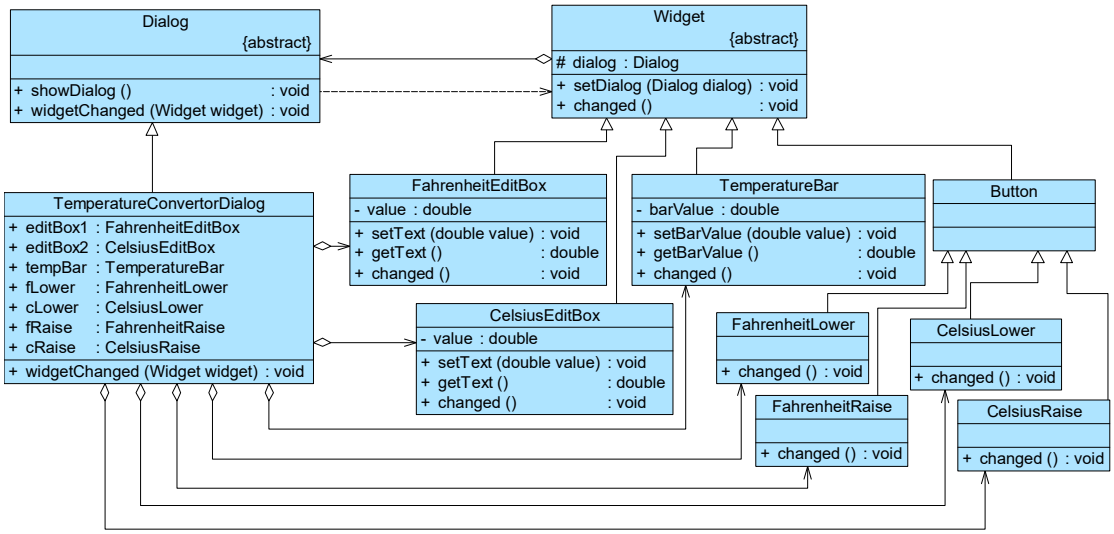
```
}

//抽象迭代器类
interface AbstractIterator
{
	public void next();
	public boolean isLast();
	public void previous();
	public boolean isFirst();
	public String getNextItem();
	public String getPreviousItem();
}

//具体迭代器类
class MyIterator implements AbstractIterator
{
	private String[] productsName;
	private int index1;
	private int index2;
	public MyIterator(AbstractProductList list)
	{
		productsName = list.getProductsName();
		index1 = 0;
		index2 = productsName.length-1;
	}
	public void next()
	{
		if(index1<productsName.length)
		{
			index1++;
		}
	}
	public boolean isLast()
	{
		return (index1==productsName.length);
	}
	public void previous()
	{
		if(index2>-1)
		{
			index2--;
		}
	}
	public boolean isFirst()
```

```
    {
        return (index2==-1);
    }
    public String getNextItem()
    {
        return productsName[index1];
    }
    public String getPreviousItem()
    {
        return productsName[index2];
    }
}
```

4. 如下图所示温度转换器程序，该程序在同一个界面上显示华氏温度(Fahrenheit)和摄氏温度(Celsius)。用户可以通过“升高”、“降低”按钮或右边的温度调节条来调节温度，也可以直接通过文本框来设置温度，摄氏温度和华氏温度将同时改变，且温度调节条也将一起被调节。使用中介者模式设计该系统，绘制类图并编程模拟实现。



参考类图：

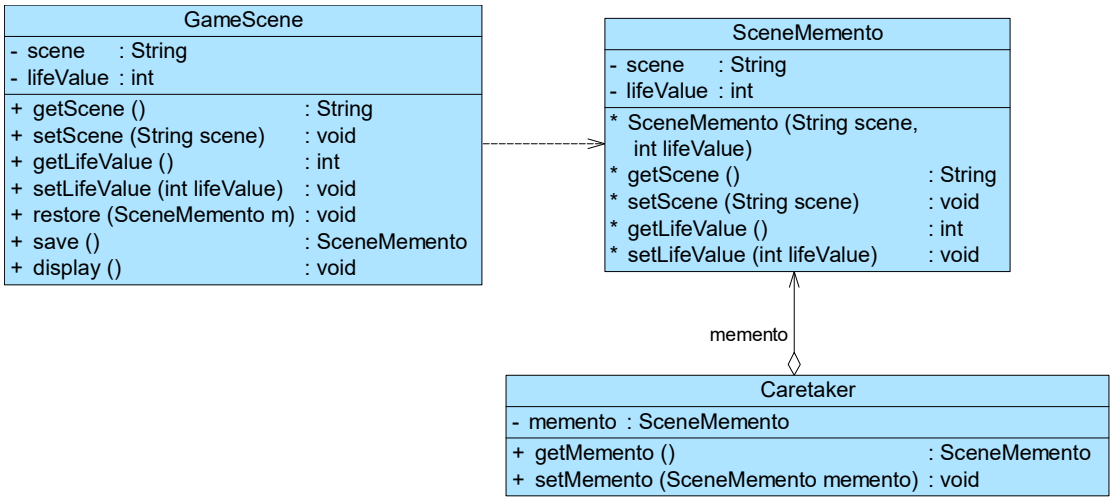


Dialog 充当抽象中介者角色，TemperatureConvertorDialog 充当具体中介者角色，Widget 充当抽象同事角色，FahrenheitEditBox、CelsiusEditBox、TemperatureBar、FahrenheitLower、

FahrenheitRaise、CelsiusLower 和 CelsiusRaise 充当具体同事角色。

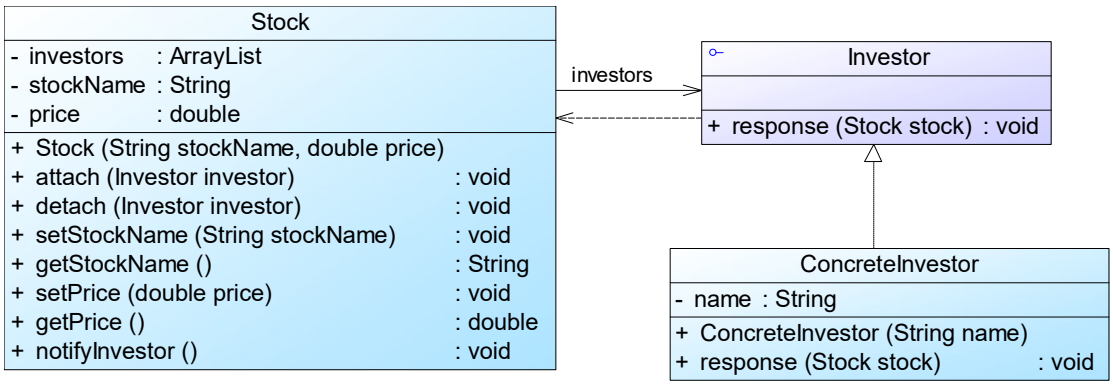
5. 某软件公司正在开发一款 RPG 网游，为了给玩家提供更多方便，在游戏过程中可以设置一个恢复点，用于保存当前的游戏场景，如果在后续游戏过程中玩家角色“不幸牺牲”，可以返回到先前保存的场景，从所设恢复点开始重新游戏。试使用备忘录模式设计该功能，要求绘制相应的类图并编程模拟实现。

参考类图：

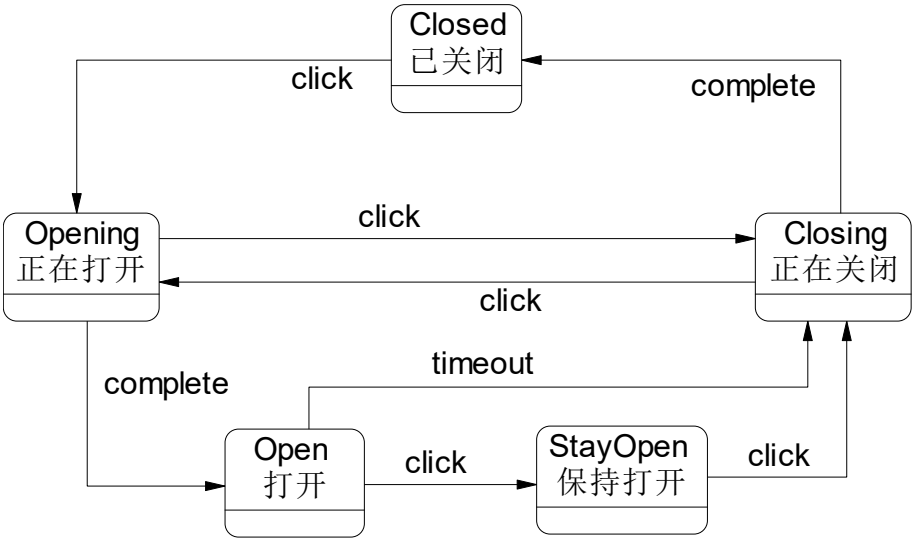


6. 某实时在线股票软件需要提供如下功能：当股票购买者所购买的某支股票价格变化幅度达到 5%时，系统将自动发送通知（包括新价格）给购买该股票的所有股民。试使用观察者模式设计并实现该系统，要求绘制相应的类图并编程模拟实现。

参考答案：



7. 传输门是传输系统中的重要装置。传输门具有 Open（打开）、Closed（关闭）、Opening（正在打开）、StayOpen（保持打开）、Closing（正在关闭）五种状态。触发状态的转换事件有 click、complete 和 timeout 三种。事件与其相应的状态转换如下图所示。



试使用状态模式对传输门进行状态模拟，要求绘制相应的类图并编程模拟实现。

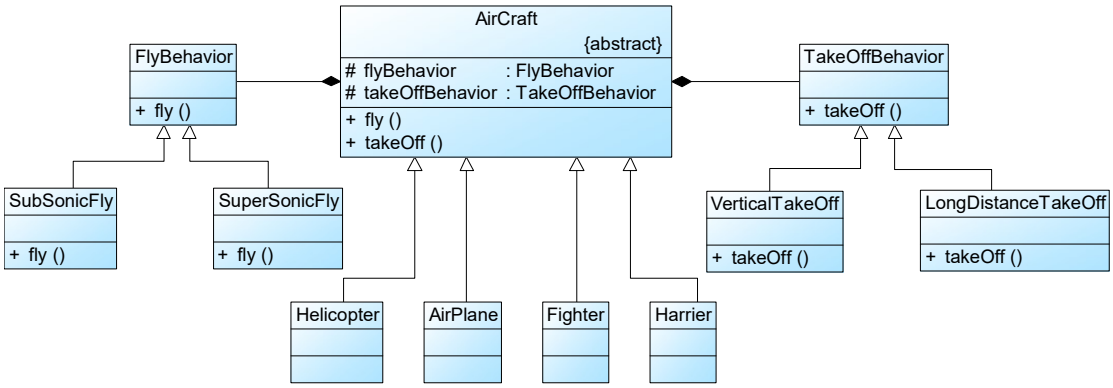
8. 某软件公司欲开发一款飞机模拟系统，该系统主要模拟不同种类飞机的飞行特征与起飞特征，需要模拟的飞机种类及其特征如表 1 所示：

表 1 飞机种类及特征一览表

飞机种类	起飞特征	飞行特征
直升机(Helicopter)	垂直起飞(VerticalTakeOff)	亚音速飞行(SubSonicFly)
客机(AirPlane)	长距离起飞(LongDistanceTakeOff)	亚音速飞行(SubSonicFly)
歼击机(Fighter)	长距离起飞(LongDistanceTakeOff)	超音速飞行(SuperSonicFly)
鹞式战斗机(Harrier)	垂直起飞(VerticalTakeOff)	超音速飞行(SuperSonicFly)

为将来能够模拟更多种类的飞机，试采用策略模式设计并模拟实现该飞机模拟系统。

参考答案：



9. 某软件公司欲为某银行的业务支撑系统开发一个利息计算模块，利息计算流程如下：

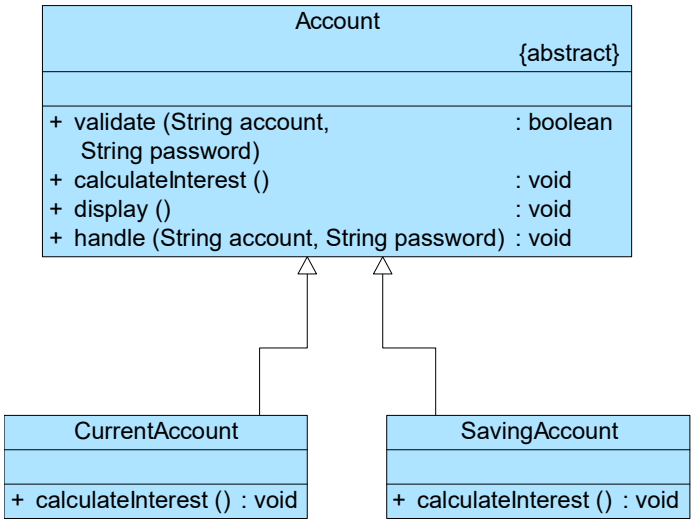
(1) 系统根据账号和密码验证用户信息，如果用户信息错误，系统显示出错提示；

(2) 如果用户信息正确，则根据用户类型的不同使用不同的利息计算公式计算利息（如活期账户和定期账户具有不同的利息计算公式）；

(3) 系统显示利息。

试使用模板方法模式设计并模拟实现该利息计算模块。

参考答案：



10. 某公司 OA 系统中包含一个员工信息管理子系统，该公司员工包括正式员工和临时工，每周人力资源部和财务部等部门需要对员工数据进行汇总，汇总数据包括员工工作时间、员工工资等。该公司基本制度如下：

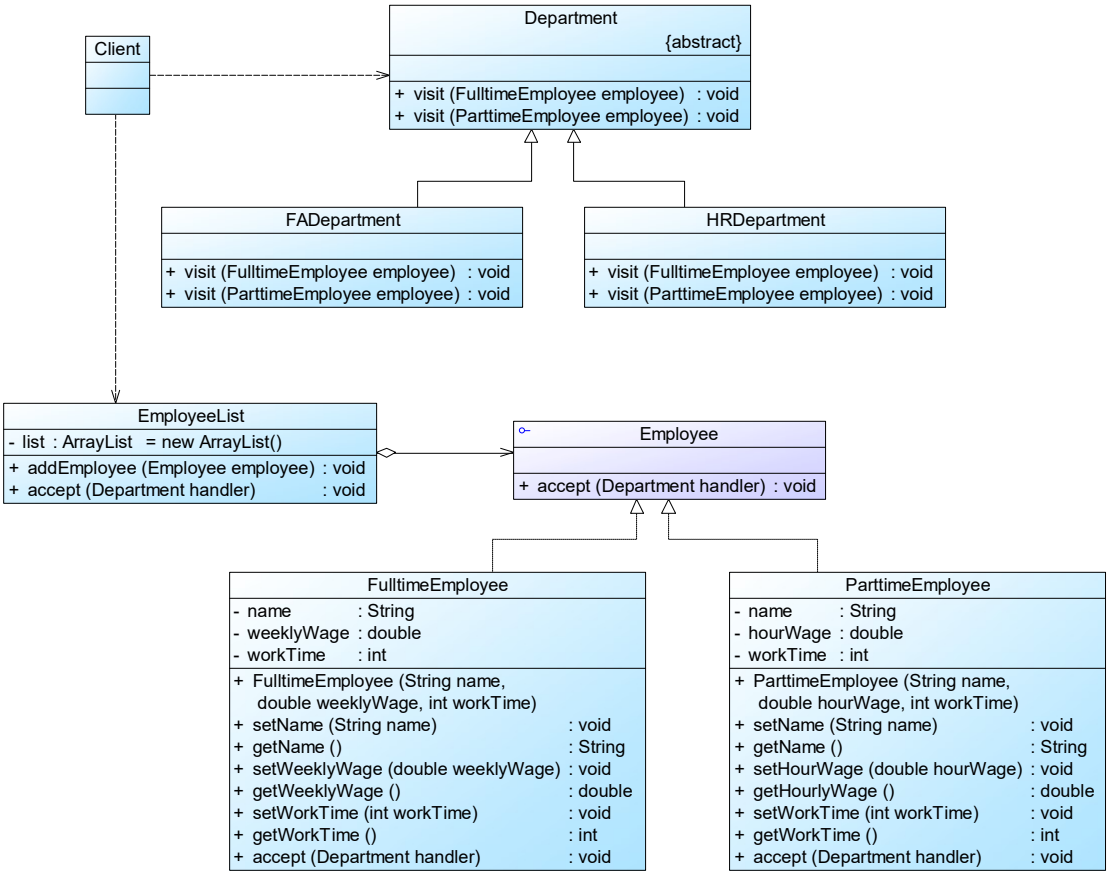
(1) 正式员工每周工作时间为 40 小时，不同级别、不同部门的员工每周基本工资不同；如果超过 40 小时，超出部分按照 100 元/小时作为加班费；如果少于 40 小时，所缺时间按照请假处理，请假所扣工资以 80 元/小时计算，直到基本工资扣除到零为止。除了记录实际工作时间外，人力资源部需记录加班时长或请假时长，作为员工平时表现的一项依据。

(2) 临时工每周工作时间不固定，基本工资按小时计算，不同岗位的临时工小时工资不同。人力资源部只需记录实际工作时间。

人力资源部和财务部工作人员可以根据各自的需要对员工数据进行汇总处理，人力资源部负责汇总每周员工工作时间，而财务部负责计算每周员工工资。

现使用访问者模式设计该系统，绘制类图并编码实现。

参考答案：



三、实验方法

1. 结合实例，正确无误地绘制职责链模式、命令模式、迭代器模式、中介者模式、备忘录模式、观察者模式、状态模式、策略模式、模板方法模式和访问者模式的模式结构图；

2. 使用任意一种面向对象编程语言实现职责链模式、命令模式、迭代器模式、中介者模式、备忘录模式、观察者模式、状态模式、策略模式、模板方法模式和访问者模式实例，代码运行正确无误。

四、实验步骤

1. 结合实例，使用 PowerDesigner 绘制职责链模式实例结构图并用面向对象编程语言实现该模式实例；

2. 结合实例，使用 PowerDesigner 绘制命令模式实例结构图并用面向对象编程语言实现该模式实例；

3. 结合实例，使用 PowerDesigner 绘制迭代器模式实例结构图并用面向对象编程语言实现该模式实例；

4. 结合实例，使用 PowerDesigner 绘制中介者模式实例结构图并用面向对象编程语言实现该模式实例；

5. 结合实例，使用 PowerDesigner 绘制备忘录模式实例结构图并用面向对象编程语言实现该模式实例；

6. 结合实例，使用 PowerDesigner 绘制观察者模式实例结构图并用面向对象编程语言实现该模式实例；

7. 结合实例，使用 PowerDesigner 绘制状态模式实例结构图并用面向对象编程语言实现该模式实例；

8. 结合实例，使用 PowerDesigner 绘制策略模式实例结构图并用面向对象编程语言实现该模式实例；

9. 结合实例，使用 PowerDesigner 绘制模板方法模式实例结构图并用面向对象编程语言实现该模式实例；

10. 结合实例，使用 PowerDesigner 绘制访问者模式实例结构图并用面向对象编程语言实现该模式实例。

五、实验结果

注：有程序的要求附上程序源代码，有图表的要有截图并有相应的文字说明和分析。

1. 提供职责链模式实例结构图及实现代码；
2. 提供命令模式实例结构图及实现代码；
3. 提供迭代器模式实例结构图及实现代码；
4. 提供中介者模式实例结构图及实现代码；
5. 提供备忘录模式实例结构图及实现代码；
6. 提供观察者模式实例结构图及实现代码；
7. 提供状态模式实例结构图及实现代码；
8. 提供策略模式实例结构图及实现代码；
9. 提供模板方法模式实例结构图及实现代码；
10. 提供访问者模式实例结构图及实现代码。

六、实验小结

给出本次实验的体会，如学会了什么，遇到哪些问题，如何解决这些问题，存在哪些有待改进的地方。