《Java 设计模式》实验指导书

主讲教师：程细柱

韶关学院计算机科学技术学院

2012 年 2 月 22 号

目　录

实验一：Rose的使用与类图的设计 3

实验二：五类“创建型模式”的应用 5

实验三：七类“结构型模式”的应用 7

实验四：十一类“行为型模式”的应用 9

实验一　Rose 的使用与类图的设计

一、实验目的

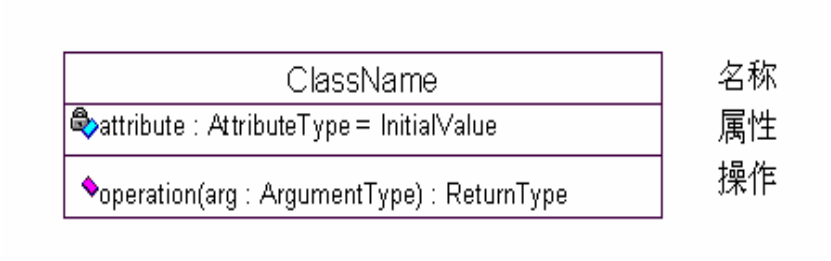
1、理解类的基本概念，掌握如何从需求分析中抽象出类的方法。

2、解类间关系的基本概念，掌握如何分析类间关系的方法。

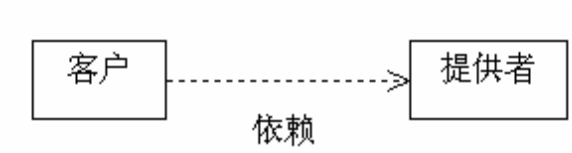
3、掌握在 Rational Rose 中绘制类以及类关系的操作方法。

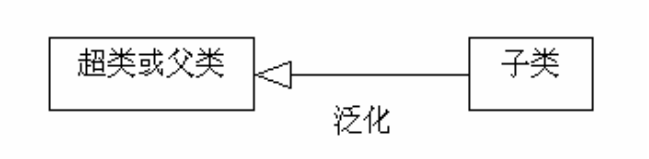
二、实验原理

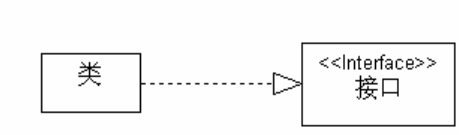
1、类是面向对象系统组织结构的核心。对一组具有相同属性、操作、关系和语义的对象的抽象。包括名称部分（Name）、属性部分（Attribute）和操作部分（Operation）。



2、类之间存在以下的关系：依赖关系、泛化关系、关联关系、实现关系。







三、实验内容

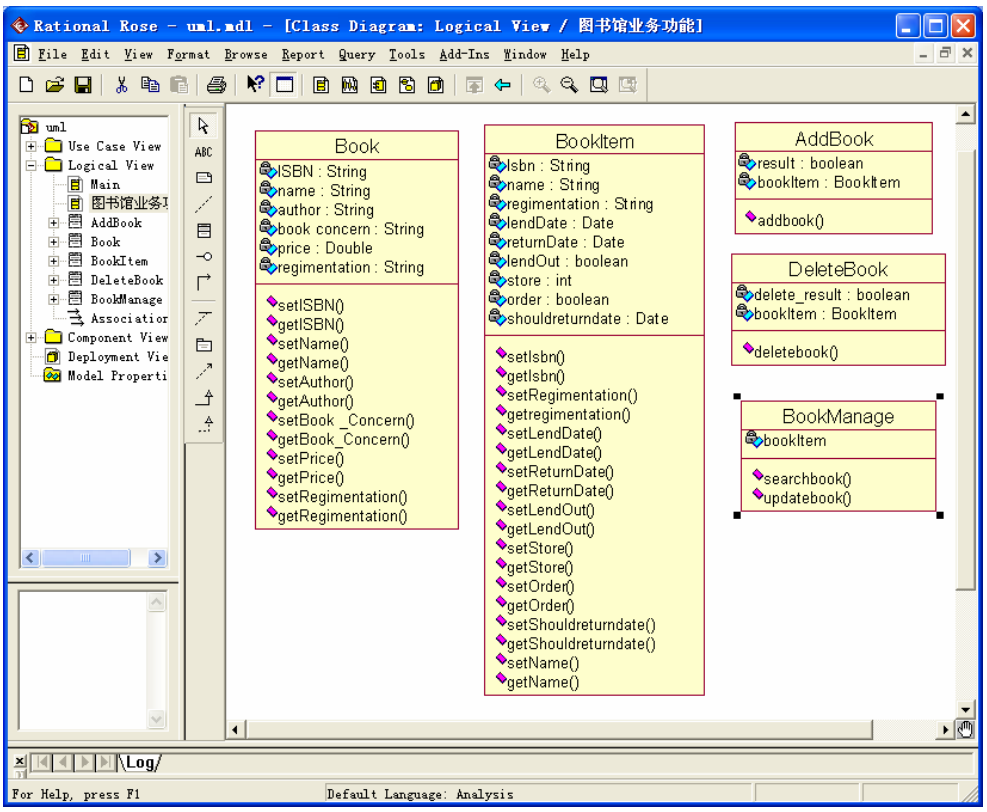
1、通过对“图书馆管理系统”的需求的初步分析，寻找和抽象出书籍管理功能中的类。

2、对书籍管理功能中的类的关系建模。

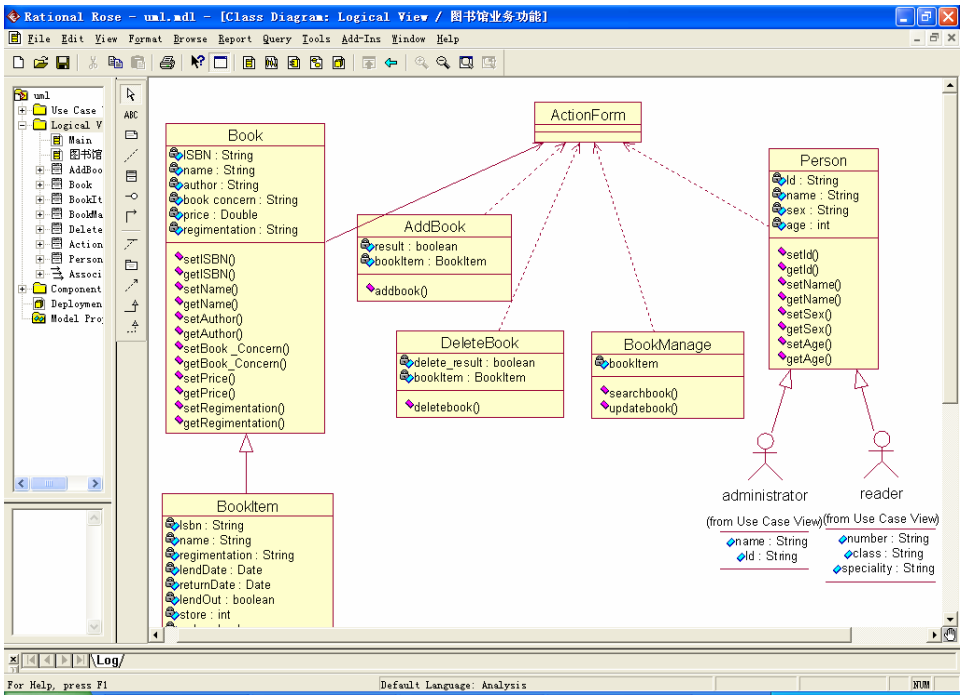
四、实验步骤

1、对图书馆管理系统中的书籍管理功能进行分析，提取出书籍信息类、书目类、新增书籍界面类、修改书籍界面类、删除书籍界面类和书籍管理类 6 个类。

2、利用 Rose 画出这 6 个类的类图如下：



3、分析以上类，确定它们之间的关系，并利用 Rose 画出类关系图如下：



4、整理实验结果，小结实验心得体会。

实验二　五类“创建型模式”的应用

一、实验目的

1、以本实验指导中的 Factory Method 模式为实验实例，掌握五类“创建型模式”的工作原理和应用环境。

2、掌握工厂方法模式（Factory Method）、抽象工厂模式（Abstract Factory）、建造者模式（Builder）、原型模式（Prototype）、单例模式（Singleton）等五类“创建型模式”的实验过程。

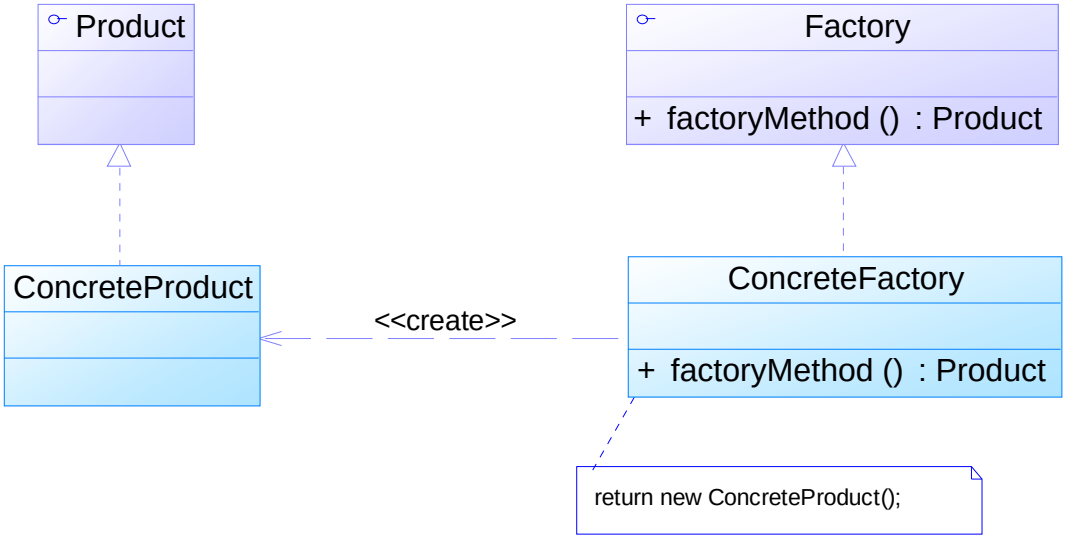
二、实验原理

1、创建型模式的工作原理

创建型模式隐藏了类的实例的创建细节，通过隐藏对象如何被创建和组合在一起达到使整个系统独立的目的。 创建型模式分为：工厂方法模式（Factory Method）、抽象工厂模式（Abstract Factory）、建造者模式（Builder）、原型模式（Prototype）、单例模式（Singleton）等五类。

2、Factory Method 模式的工作原理

工厂方法模式(Factory Method Pattern)：也叫虚拟构造器(Virtual Constructor)模式或者多态工厂(Polymorphic Factory)模式，在工厂方法模式中，工厂父类负责定义创建产品对象的公共接口，而工厂子类则负责生成具体的产品对象，这样做的目的是将产品类的实例化操作延迟到工厂子类中完成。它的 UML 类图如下：



工厂方法模式包含如下角色：

- Product：抽象产品
- ConcreteProduct：具体产品
- Factory：抽象工厂
- ConcreteFactory：具体工厂

Factory Method 模式的特点为当系统扩展需要添加新的产品对象时，仅仅需要添加一个具体产品对象以及一个具体工厂对象，原有工厂对象不需要进行任何修改，也不需要修改客户端，很好地符合了“开闭原则” 。

3、抽象工厂模式（Abstract Factory）、建造者模式（Builder）、原型模式（Prototype）、单例模式（Singleton）的工作原理请参考教材。

三、实验内容

1、用 Factory Method 模式设计一个电视机工厂

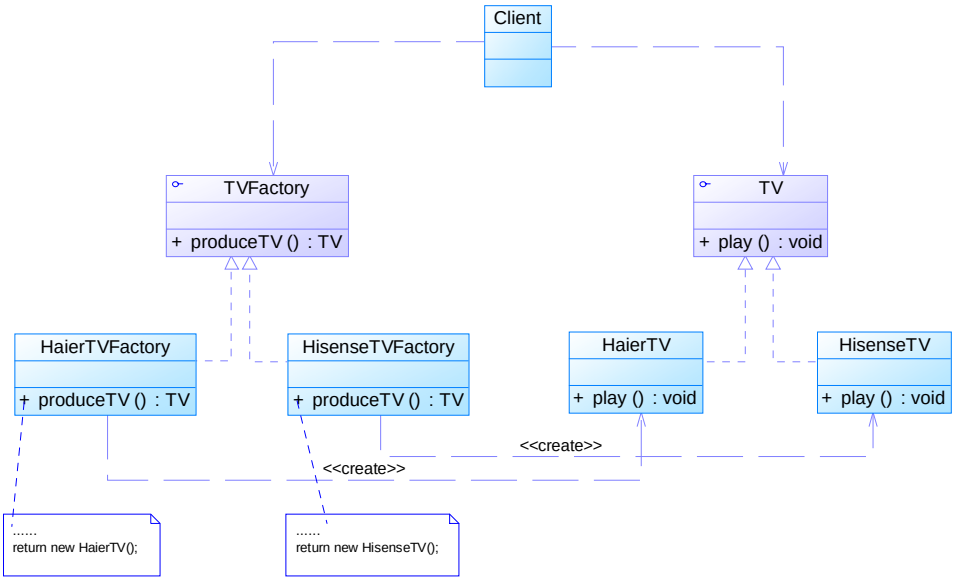
要求为每种品牌的电视机提供一个子工厂，如：海尔工厂专门负责生产海尔电视机，海信工厂专门负责生产海信电视机，如果需要生产 TCL 电视机或创维电视机，只需要对应增加一个新的 TCL 工厂或创维工厂即可，原有的工厂无须做任何修改，使得整个系统具有更加的灵活性和可扩展性。

2、参考以上实例设计抽象工厂模式（Abstract Factory）、建造者模式（Builder）、原型模式（Prototype）、单例模式（Singleton）的程序实例。

四、实验步骤

1、用 UML 设计“电视机工厂”的类图。

“电视机工厂”的参考类图如下：



2、根据类图写出“电视机工厂”的源代码。

参考代码下载地址： http://download.csdn.net/user/cflynn

3、上机测试程序，写出运行结果。

4、按同样的步骤设计其他“创建型模式”的程序实例。

实验三　七类“结构型模式”的应用

一、实验目的

1、以设计适配器模式(Adapter)为实验实例，掌握“结构型模式”的工作原理、应用环境和应用方法。

2、掌握适配器模式(Adapter)、桥接模式(Bridge)、组合模式(Composite) 、装饰模式(Decorator)、外观模式(Facade) 、享元模式(Flyweight) 、代理模式(Proxy 等七类“结构型模式”的实验过程。

二、实验原理

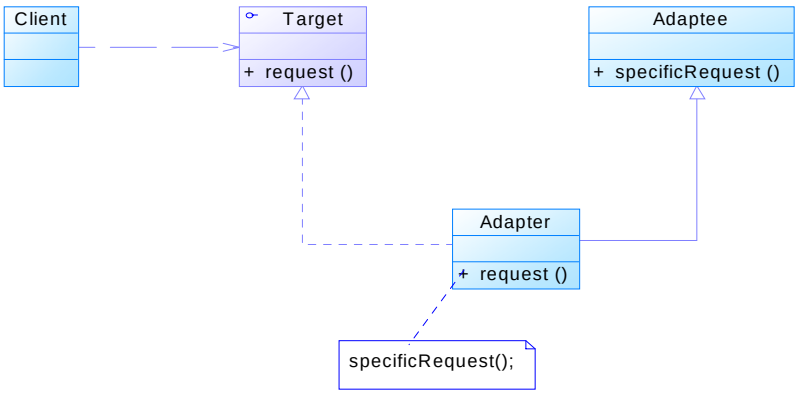
1、结构型模式的工作原理

结构型模式(Structural Pattern)描述如何将类或者对象结合在一起形成更大的结构，就像搭积木，可以通过简单积木的组合形成复杂的、功能更为强大的结构。结构型模式可以分为类结构型模式和对象结构型模式。 也可分为：适配器模式(Adapter)、桥接模式(Bridge)、组合模式(Composite) 、装饰模式(Decorator)、外观模式(Facade) 、享元模式(Flyweight) 、代理模式(Proxy)等七类。

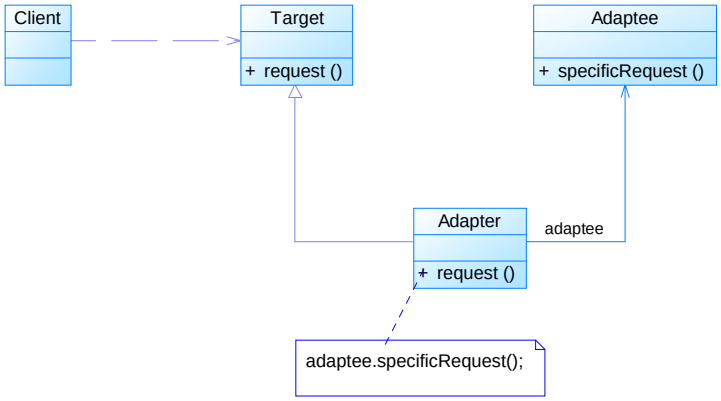
2、Adapter 模式的工作原理

适配器模式(Adapter Pattern) ：将一个接口转换成客户希望的另一个接口，适配器模式使接口不兼容的那些类可以一起工作，其别名为包装器(Wrapper)。适配器模式既可以作为类结构型模式，也可以作为对象结构型模式。

类结构型模式的 UML 类图如下：



对象结构型模式的 UML 类图如下：



适配器模式包含如下角色：

Target： 目标抽象类

Adapter：适配器类

Adaptee：适配者类

Client： 客户类

3、桥接模式(Bridge)、组合模式(Composite) 、装饰模式(Decorator)、外观模式(Facade) 、享元模式(Flyweight) 、代理模式(Proxy)的工作原理请参考教材。

三、实验内容

1、用 Adapter 模式设计一个仿生机器人

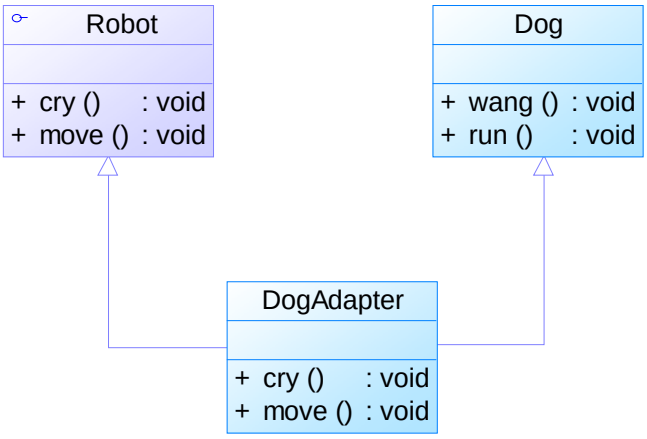
要求机器人可以模拟各种动物行为，在机器人中定义了一系列方法，如机器人叫喊方法 cry()、机器人移动方法 move()等。如果希望在不修改已有代码的基础上使得机器人能够像狗一样汪汪叫，像狗一样快跑，或者像鸟一样叽叽叫，像鸟一样快快飞，使用适配器模式进行系统设计。

2、参考以上实例设计桥接模式(Bridge)、组合模式(Composite) 、装饰模式(Decorator)、外观模式(Facade) 、享元模式(Flyweight) 、代理模式(Proxy)的实例程序。

四、实验步骤

1、用 UML 设计“仿生机器人”的类图。

“仿生机器人”的参考类图如下：



2、根据类图写出“仿生机器人”的源代码。

参考代码下载地址： http://download.csdn.net/user/cflynn

3、上机测试程序，写出运行结果。

4、按同样的步骤设计其他“结构型模式”的程序实例。

实验四　十一类“行为型模式”的应用

一、实验目的

1、以设计职责链模式(Chain of Responsibility)为实验实例，掌握“行为型模式”的工作原理、应用环境和应用方法。

2、掌握职责链模式(Chain of Responsibility)、命令模式(Command)、解释器模式(Interpreter)、迭代器模式(Iterator)、中介者模式(Mediator)、备忘录模式(Memento)、观察者模式(Observer)、状态模式(State)、策略模式(Strategy)、模板方法模式(Template Method)、访问者模式(Visitor)等十一类“行为型模式”的实验过程。

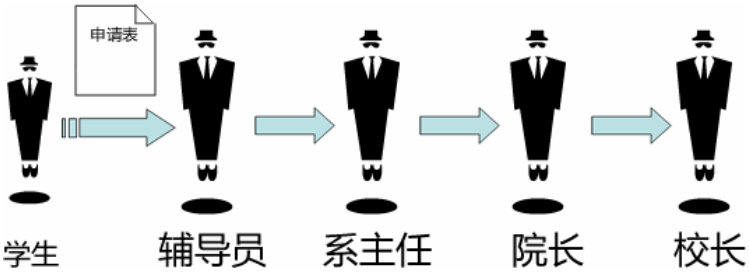
二、实验原理

1、行为型模式的工作原理

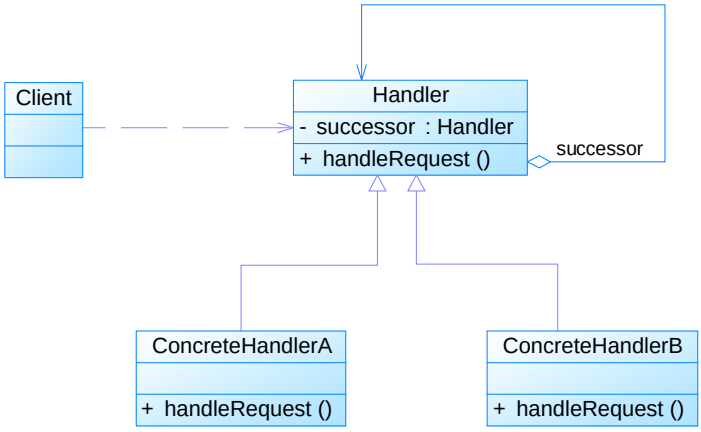
行为型模式(Behavioral Pattern)是对在不同的对象之间划分责任和算法的抽象化。行为型模式分为：类行为型模式和对象行为型模式两种。其中，类的行为型模式使用继承关系在几个类之间分配行为，类行为型模式主要通过多态等方式来分配父类与子类的职责；对象的行为型模式则使用对象的聚合关联关系来分配行为，对象行为型模式主要是通过对象关联等方式来分配两个或多个类的职责。行为型模式也可以分为：职责链模式(Chain of Responsibility)、命令模式(Command)、解释器模式(Interpreter)、迭代器模式(Iterator)、中介者模式(Mediator)、备忘录模式(Memento)、观察者模式(Observer)、状态模式(State)、策略模式(Strategy)、模板方法模式(Template Method)、访问者模式(Visitor)等十一类。

2、Chain of Responsibility 模式的工作原理

职责链模式(Chain of Responsibility)：为了避免将请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止。如学生请假的职责链如下：



职责链模式的 UML 类图如下：



职责链模式包含如下角色：

Handler:　抽象处理者

ConcreteHandler:　具体处理者

Client:　客户类

3、命令模式(Command)、解释器模式(Interpreter)、迭代器模式(Iterator)、中介者模式(Mediator)、备忘录模式(Memento)、观察者模式(Observer)、状态模式(State)、策略模式(Strategy)、模板方法模式(Template Method)、访问者模式(Visitor)的工作原理请参考教材。

三、实验内容

1、用职责链模式设计一个审批员工假条的子模块

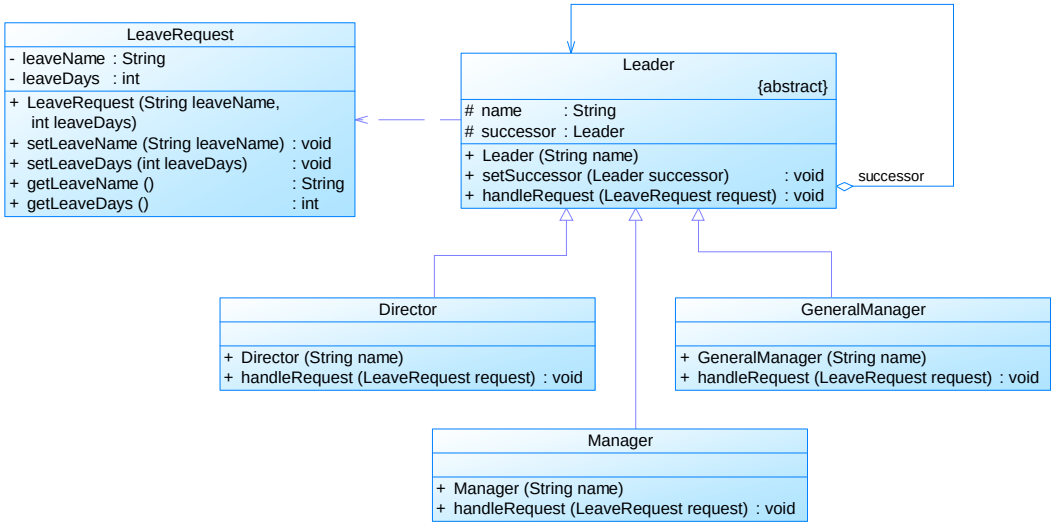
某 OA 系统需要提供一个假条审批的模块，如果员工请假天数小于 3 天，主任可以审批该假条；如果员工请假天数大于等于 3 天，小于 10 天，经理可以审批；如果员工请假天数大于等于 10 天，小于 30 天，总经理可以审批；如果超过 30 天，总经理也不能审批，提示相应的拒绝信息。

2、参考以上实例设计命令模式(Command)、解释器模式(Interpreter)、迭代器模式(Iterator)、中介者模式(Mediator)、备忘录模式(Memento)、观察者模式(Observer)、状态模式(State)、策略模式(Strategy)、模板方法模式(Template Method)、访问者模式(Visitor)的实例程序。

四、实验步骤

1、用 UML 设计“假条审批模块”的类图。

“假条审批模块”的参考类图如下：



2、根据类图写出“假条审批模块”的源代码。

参考代码下载地址：　http://download.csdn.net/user/cflynn

3、上机测试程序，写出运行结果。

4、按同样的步骤设计其他“行为型模式”的程序实例。