天津大学

面向对象设计原则实验报告

学　　院　智算学部

专　　业　软件工程

学　　号　3019213043

姓　　名　刘京宗

1.1 实验目的

1. 通过实例深入理解和掌握所学的面向对象设计原则。

2. 熟练使用面向对象设计原则对系统进行重构。

3. 熟练绘制重构后的结构图(类图)。

1.2 实验内容

1.在某绘图软件中提供了多种大小不同的画笔(Pen),并且可以给画笔指定不同颜色，某设计人员针对画笔的结构设计了如图 1-1 所示的初始类图。

通过仔细分析,设计人员发现该类图存在非常严重的问题,即如果需要增加一种新的大小或颜色的笔,就需要增加很多子类,例如增加一种绿色的笔,则对应每一种大小的笔都需要增加一支绿色笔,系统中类的个数急剧增加。

试根据依赖倒转原则和合成复用原则对该设计方案进行重构,使得增加新的大小或颜色的笔都较为方便,请绘制重构之后的结构图(类图)。

2. 在某公司财务系统的初始设计方案中存在如图 1-2 所示的 Employee 类,该类包含员工编号(ID),姓名(name),年龄(age).性别(gender)、薪水(salary)、每月工作时数(workHoursPerMonth),每月请假天数(leaveDaysPerMonth)等属性。该公司的员工包括全职和兼职两类,其中每月工作时数用于存储兼职员工每个月工作的小时数,每月请假天数用于存储全职员工每个月请假的天数。系统中两类员工计算工资的万法也不一样,全职员工按照工作日数计算工资，兼职员工按照工.作时数计算上资，内此在 Employee 类中提供了两个方法 calculateSalaryByDays()和 calculateSalaryByHours(),分别用于按照大数和时数计算工资,此外,还提供了方法 displaySalary()用于显示工资。

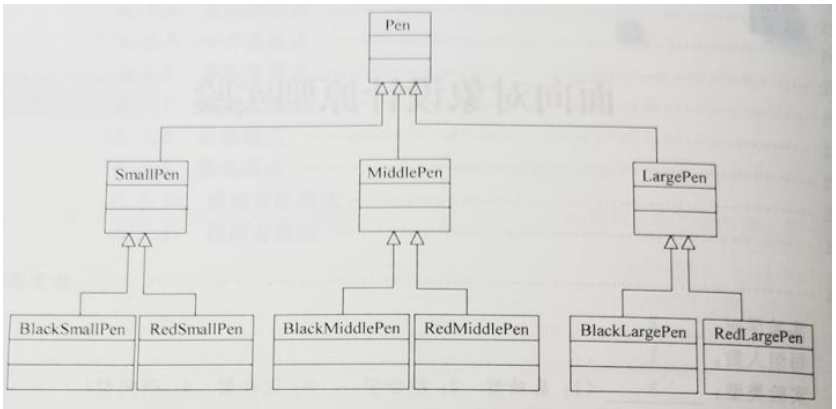


图 1-1　画笔结构初始类图

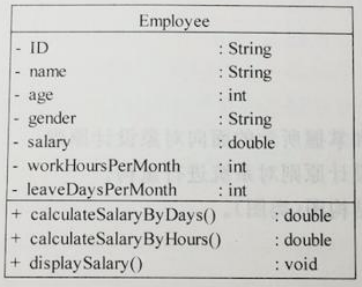
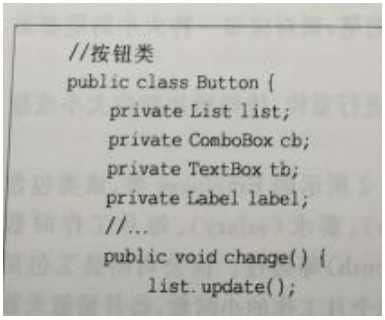


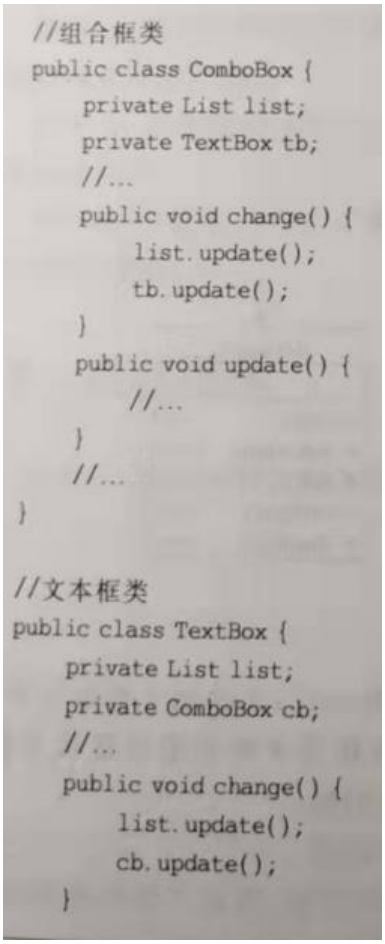
图 1-2　Employee 类初始类图

试采用所学面向对象设计原则分析图 1-2 中 Employee 类存在的问题并对其进行重构绘制重构之后的类图。

3.在某图形界面中存在如下代码片段,组件类之间有较为复杂的相互引用关系:



```
//按钮类
public class Button {
    private List list;
    private ComboBox cb;
    private TextBox tb;
    private Label label;
    //...
    public void change() {
        list.update();
        cb.update();
        tb.update();
        label.update();
    }
    public void update() {
        //...
    }
    //...
}
```



```
//组合框类
public class ComboBox {
    private List list;
    private TextBox tb;
    //...
    public void change() {
        list.update();
        tb.update();
    }
    public void update() {
        //...
    }
    //...
}

//文本框类
public class TextBox {
    private List list;
    private ComboBox cb;
    //...
    public void change() {
        list.update();
        cb.update();
    }
    public void update() {
        //...
    }
    //...
}

//列表框类
public class List {
    private ComboBox cb;
    private TextBox tb;
    //...
    public void change() {
        cb.update();
        tb.update();
    }
    public void update() {
        //...
    }
    //...
}

//文本标签类
public class Label {
    //...
    public void update() {
        //...
    }
    //...
}
```

如果在上述系统中增加一个新的组件类,则必须修改与之交互的其他组件类的源代码,将导致多个类的源代码需要修改。

基于上述代码,请结合所学知识完成以下两道练习题:(1)绘制上述代码对应的类图。

(2)根据迪米特法则对所绘制的类图进行重构,以降低组件之间的耦合度,绘制重构后的类图。

4.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

不同的图形,图形库设计人员设计的初始类图如图 1-3 所示。

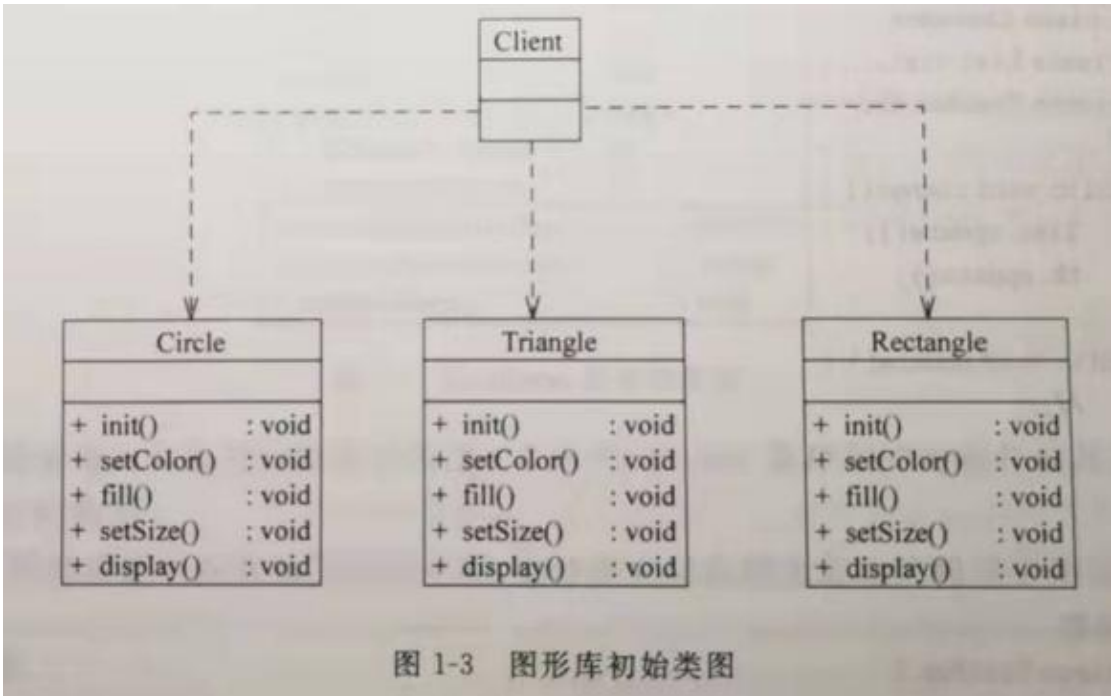


图 1-3 图形库初始类图

在该图形库中,每个图形类(例如 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。

用户在客户类(Client)中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,导致客广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 I 图形对象,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。

请绘制重构后的结构图(类图)。

1.3 实验要求

1、选择合适的面向对象设计原则对系统进行重构。绘制重构之后的类图。

1.4 实验步骤

练习 1:分析初始设计方案存在的问题,根据依赖倒转原则和合成复用原则对初始设计方案进行重构,绘制重构之后的结构图(类图)。

存在的问题:如果需要增加一种新的大小或颜色的笔,就需要增加很多子类,例如增加一种绿色的笔,则对应每一种大小的笔都需要增加一支绿色笔,系统中类的个数急剧增加。

类图见 1.5

练习 2:采用所学面向对象设计原则分析初始设计方案中存在的问题并对其进行重构，绘制重构之后的类图。

存在的问题：违反了单一职责原则。

类图见 1.5

练习 3:绘制初始代码对应的类图,再根据迪米特法则对所绘制的类图进行重构,绘制重构后的类图。

类图见 1.5

练习 4;分析初始设计方案存在的问题,根据面向对象设计原则进行重构并绘制重构后的结构图(类图)。

存在的问题:

(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。

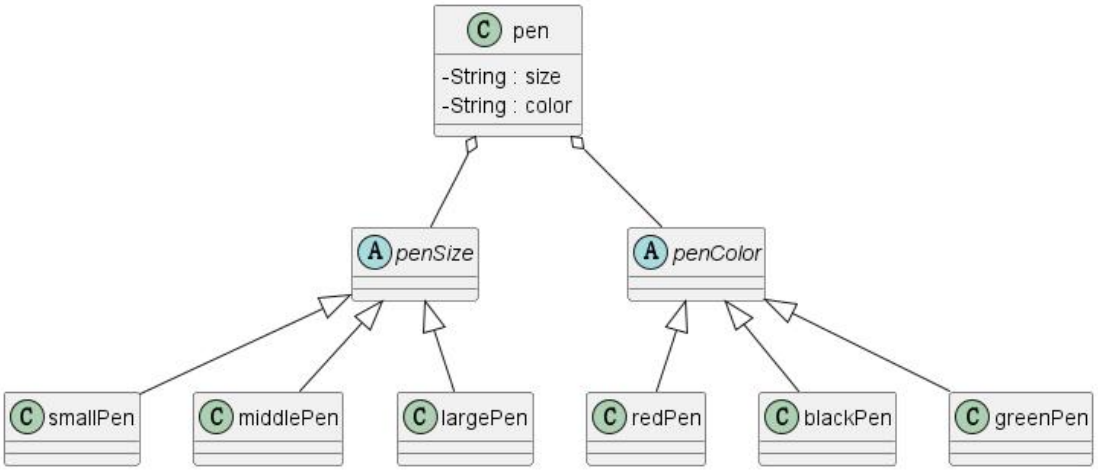
(2)在图形库中增加并使用新的图形时,需要修改客户类源代码。

(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,导致客广户类代码冗长且难以维护。

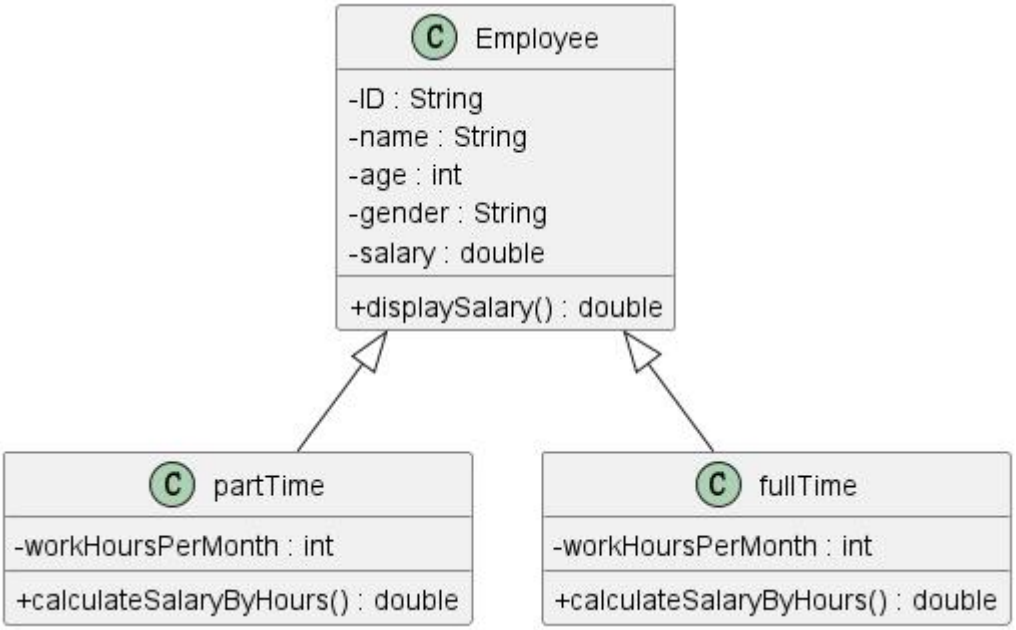
类图见 1.5

1.5 实验结果

练习 1:需要绘制重构之后的结构图(类图)。

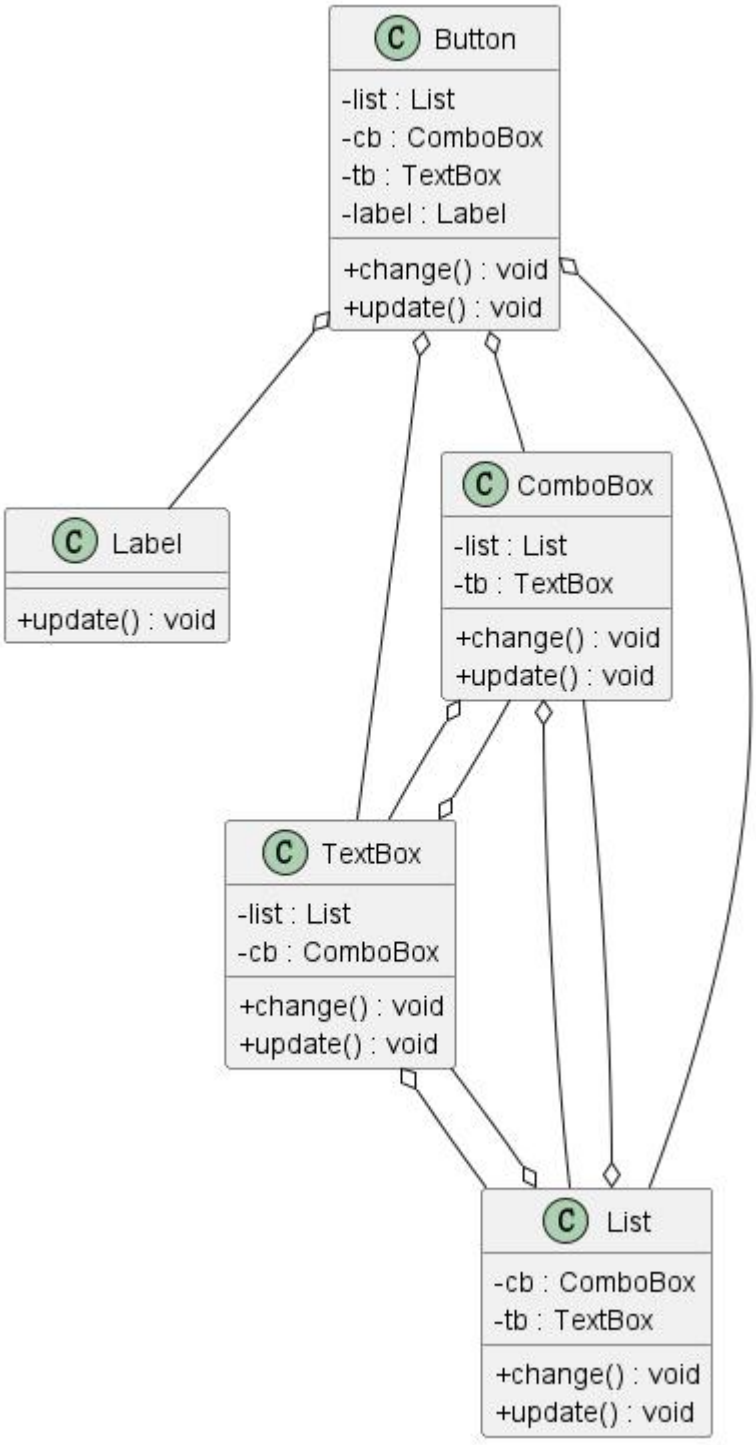


练习 2:需要绘制重构之后的结构图(类图)

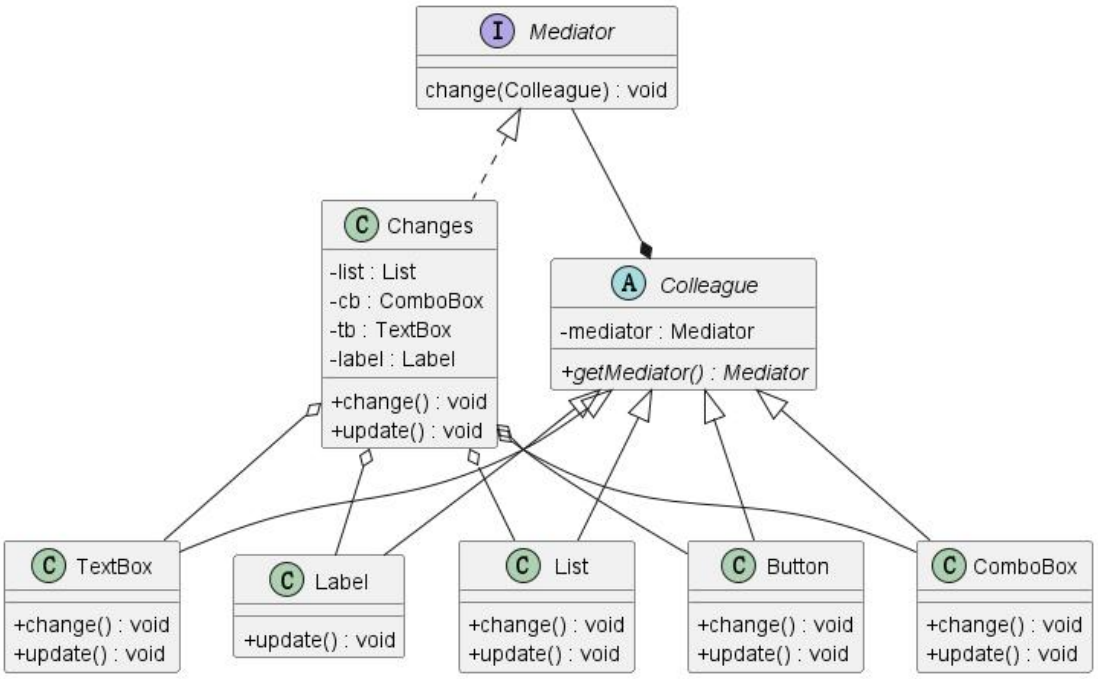


练习 3:需要绘制初始代码对应的类图和重构之后的类图。

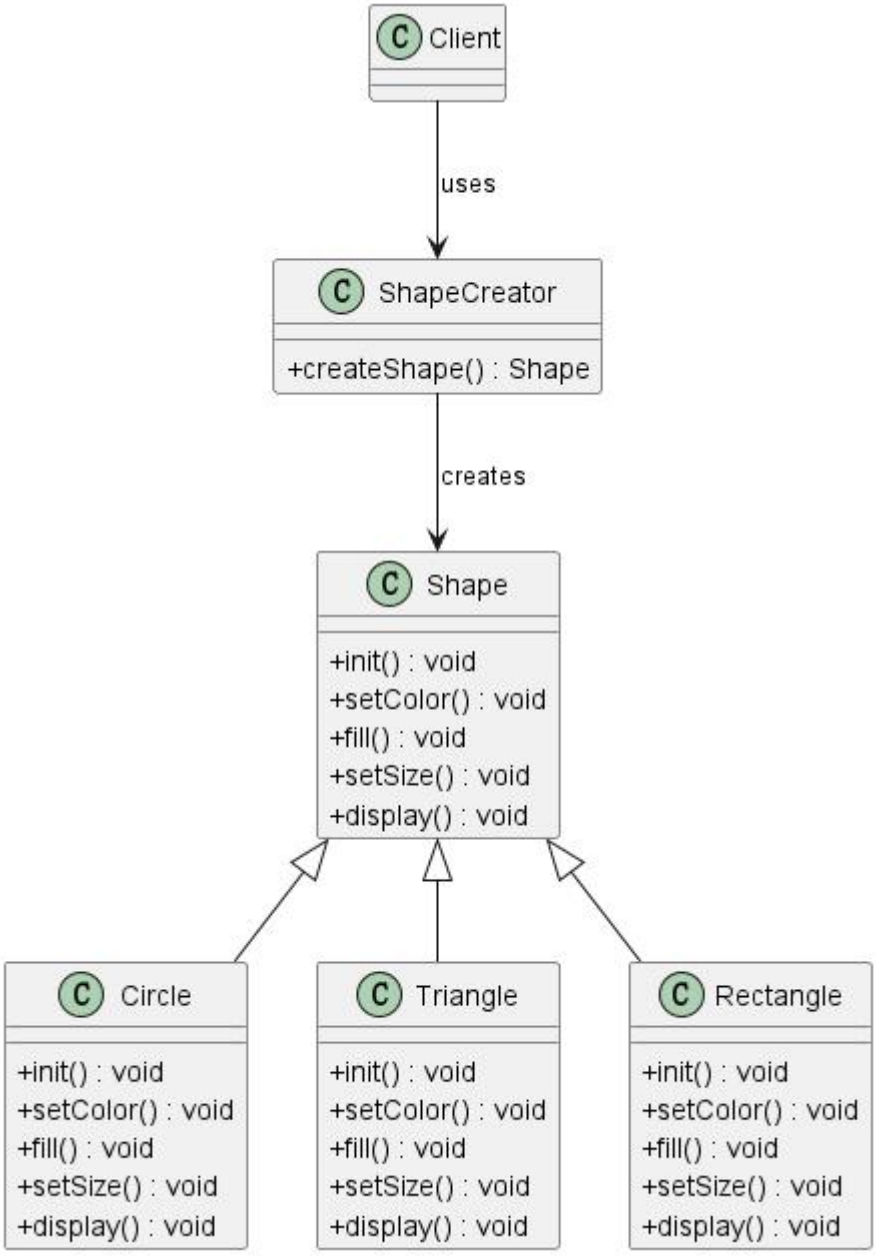
初始代码对应的类图:



修改后的类图：


Mediator
change(Colleague) : void
Changes
-list : List
-cb : ComboBox
-tb : TextBox
-label : Label
+change() : void
+update() : void
Colleague
-mediator : Mediator
+getMediator() : Mediator
TextBox
+change() : void
+update() : void
Label
+update() : void
List
+change() : void
+update() : void
Button
+change() : void
+update() : void
ComboBox
+change() : void
+update() : void

练习 4:需要绘制重构之后的结构图(类图)。



1.6 实验小结

通过这次实验我复习了面向对象设计的相关原则，也熟悉了 uml 类图的绘制，收益匪浅。