深圳大学实验报告

课程名称：软件体系结构与设计模式

实验项目名称：实验 2 OO 设计原则应用

学院：计算机与软件学院

专业：软件工程

指导教师：毛斐巧

报告人：郑彦薇 学号：2020151022 班级：软件工程 01 班

实验时间：2023 年 3.22、29、4.5、4.12 日（周三）

实验报告提交时间：2023/4/6

教务部制

一、 实验目的与要求：

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. 熟练绘制重构后的类图。

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图形，图形库设计人员设计的初始类图如图 1 所示。

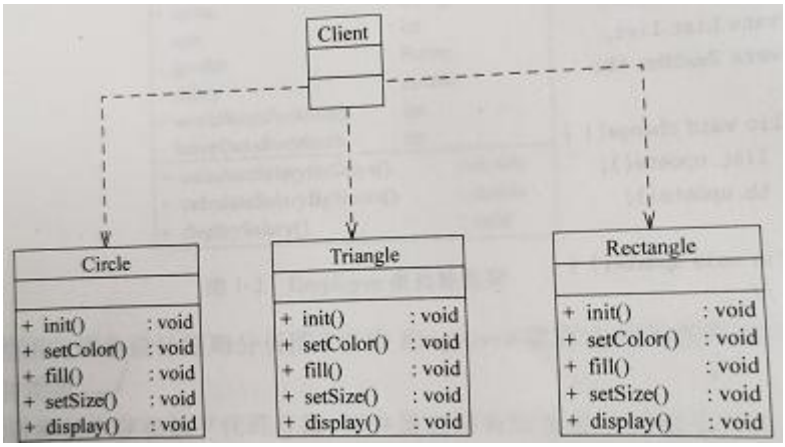


图 1 图形库初始类图

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，set Size( )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。

请绘制重构后的结构类图。

2. 在某公司财务系统的初始设计方案中，存在如图 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方法也不一样，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工资，因此在 Employee 类中提供了两个方法 calculateSalaryByDays( )和 calculateSalaryByHours( )，分别用于按照天数和时数计算工资，此外，还提供了方法 displaySalary( )用于显示工资。

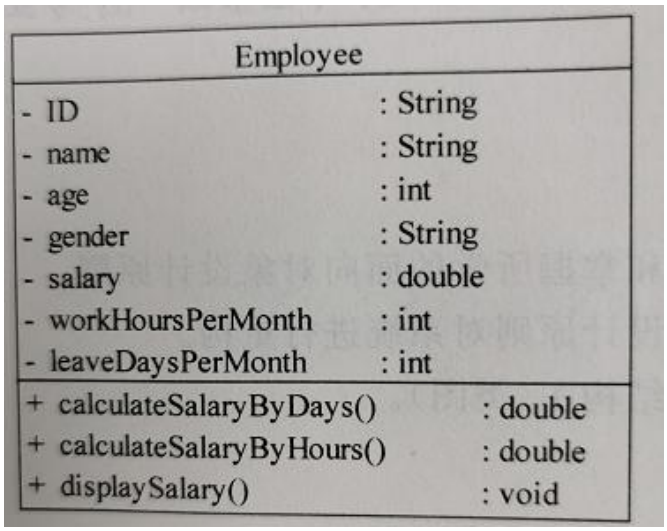


图 2 Employee 类初始类图

试采用所学面向对象设计原则分析图 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() {
        //...
    }
    //...
}
```

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

//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 3 所示的初始类图。

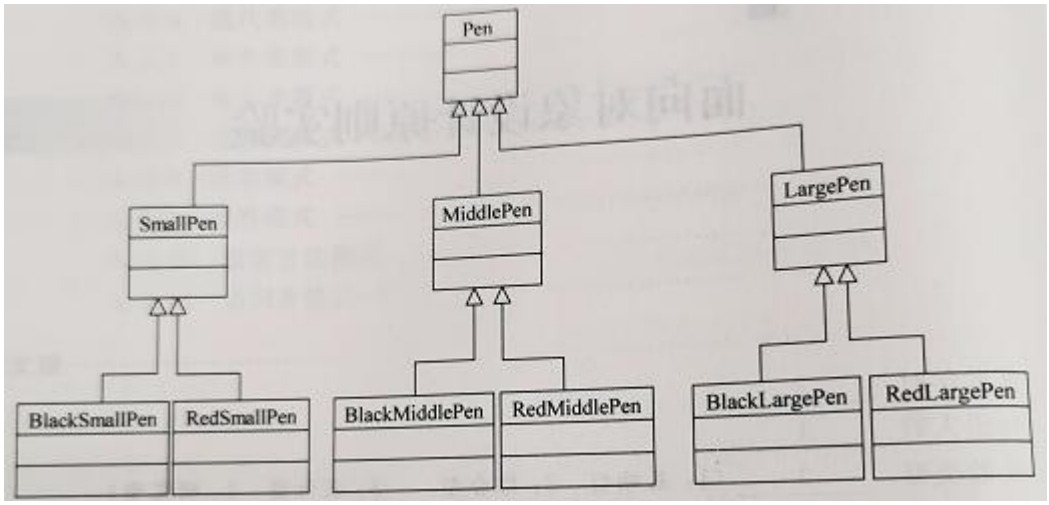


图 3 画笔结构初始类图

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类图。

三、理解和分析报告、实践过程及结果等

1. 重构客户类 Client 结构类图

1.1. 对图形库初始类图进行分析

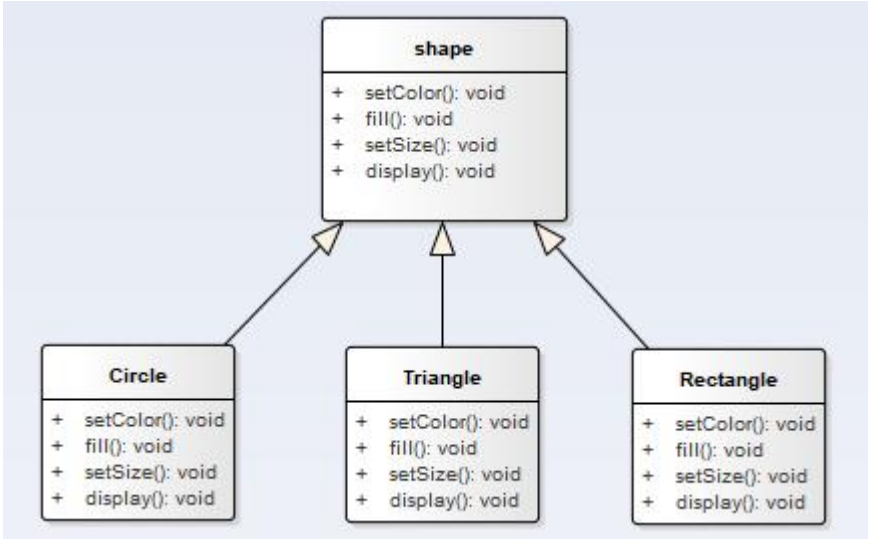
在图形库中的 3 个图形类都具有：init()方法，用于初始化所创建的图形；setColor()方法，给图形设置边框颜色；fill()方法，给图形设置填充颜色；setSize()方法，设置图形大小以及 display()方法显示图形共 5 个方法。根据方法的作用，我们可以看到在这 5 个方法中，除了图形的创建方法需要根据图形种类进行区分，其余方法在图形创建完成后，其操作都是相同的。

1.2. 重构图形系统结构类图

通过上述 1.1 中的分析，我们可以将图形的创建方法和使用方法进行隔离，对于 3 种图形的创建方法，可以定义一个创建类 ShapeCreator，包含图形的创建方法 create(String shapeName)。于是当客户端想要执行图形的创建操作时，就可以定义一个 ShapeCreator 类，然后调用 create 方法，传入需要创建的图形名称，执行对应的创建操作。

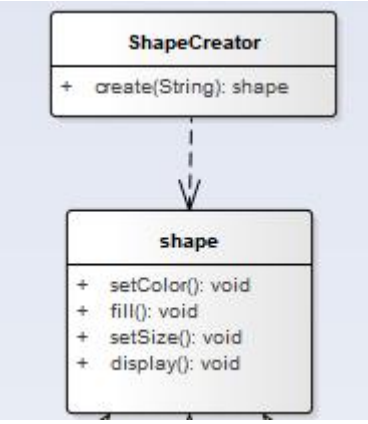
名称	参数	返回类型	作用域	构造型	别名
create	shapeName: String		Public		
新操作...					

在 1.1 的分析中，针对不同类型的图形，其使用方法都是相同的，所以接下来我们可以定义一个 shape 类，包含图形的使用方法——fill()、setSize()、setColor()以及 display()。然后，保留 Circle 类、Triangle 类以及 Rectangle 类，这三个类都继承定义的 shape 类，以使用其中的方法。

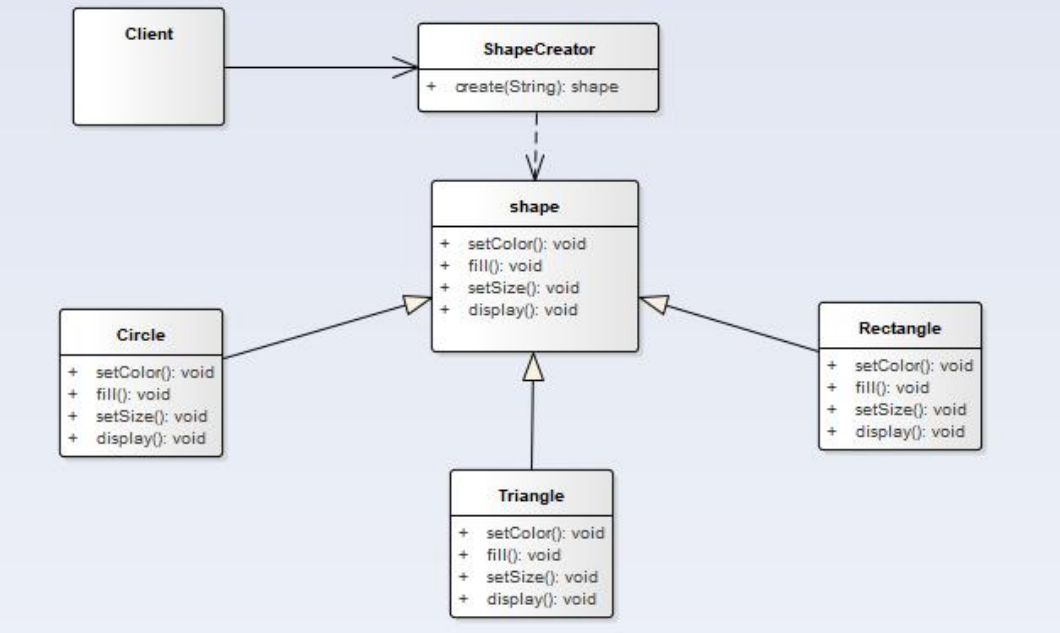


在 shape 类定义完成后，我们就可以建立 ShapeCreator 与 shape 之间的依赖关系，并使 create()

方法的创建结果为一个 shape 类。使得图形的创建与使用分隔开，且客户端能够直接使用图形。



1.3. 完成重构的结构类图如下



2. 重构 Employee 类图

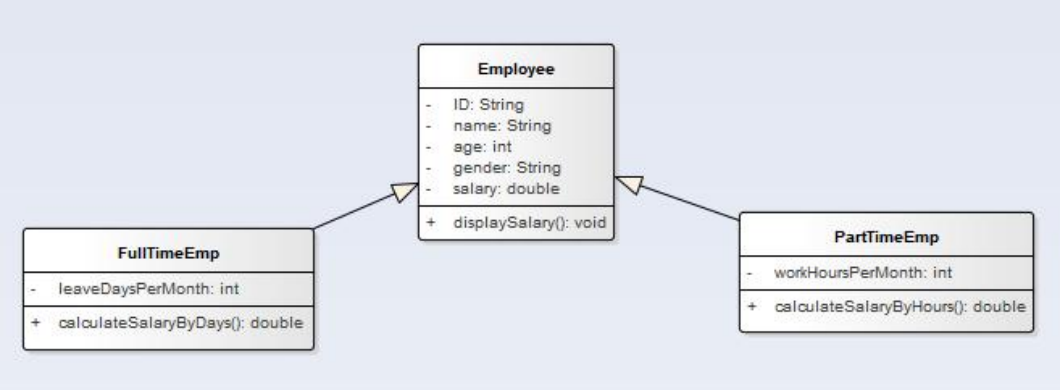
2.1. 对 Employee 初始类图进行分析

根据系统描述，对于全职员工和兼职员工的工作时长以及工作薪水对应着不同的属性和计算方法。在原系统中，对于全职员工，不需要每月的工作时数这一属性，同样的，对于兼职员工，也不需要每月请假天数这一属性。其次，在 Employee 类中，包含了两个针对不同类型员工的工资计算方法。根据单一职责原则，应该将这两部分分开。

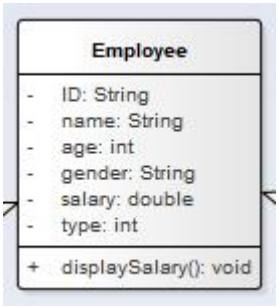
2.2. 重构财务系统结构类图

根据 2.1 中的分析，我们可以首先定义一个全职员工 FullTimeEmp 类以及一个兼职员工 PartTimeEmp 类，同时继承 Employee 类。在 FullTimeEmp 类中，需要包含属性每月请假天数 leaveDaysPerMonth，同时，去掉原先在 Employee 类中的该属性；另外，还应包含统计该

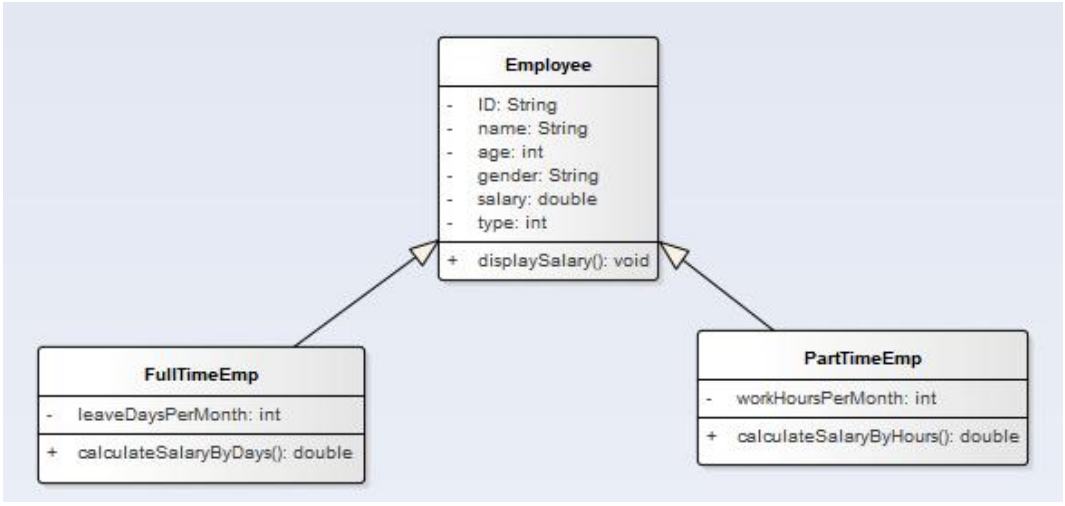
类员工的工资计算方法 calculateSalaryByDays，同样去掉 Employee 中的该方法。对于兼职员工 PartTimeEmp 类，同样从 Employee 中抽取出对应属性的工资计算方法。



由于我们已经将 Employee 中的全职员工和兼职员工各自隔离，在 Emplyee 中，我们可以加入一个用于标识员工类型的属性值 type，type 为 0 表示兼职员工，1 表示全职员工。



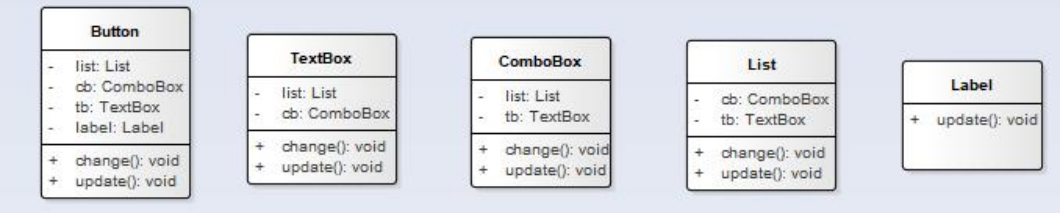
2.3. 完成重构的结构类图如下



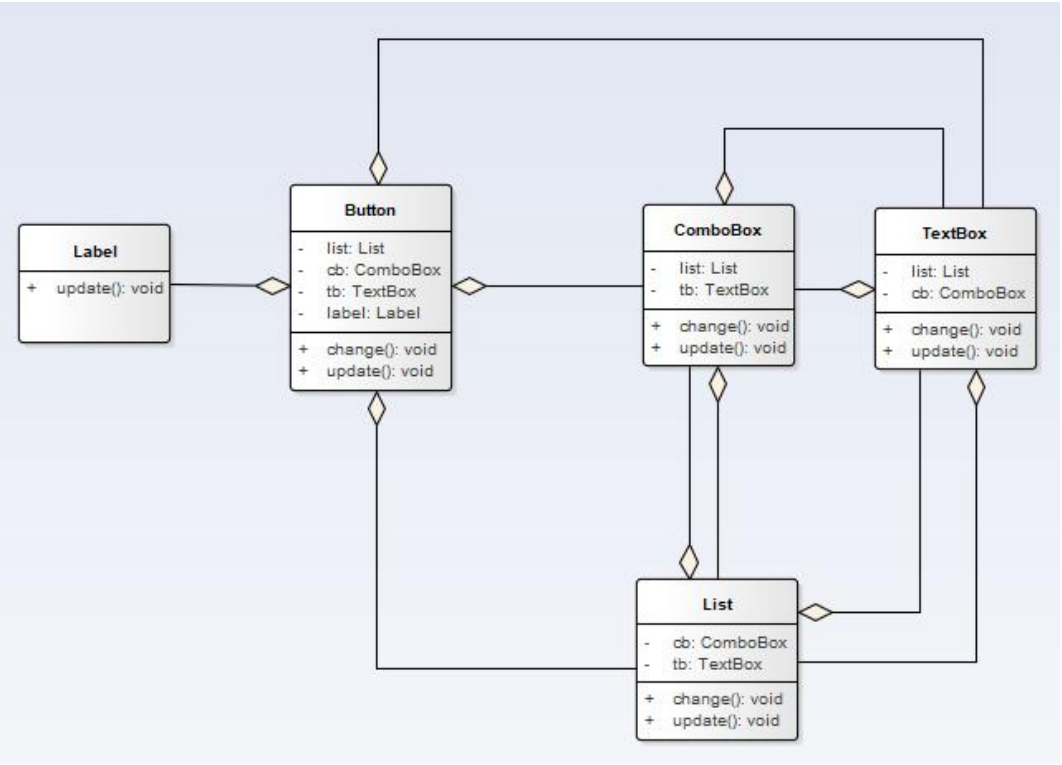
3. 基于图形界面代码，完成以下两个练习

3.1. 绘制出代码对应的类图

根据代码，我们可以得到在该系统中包含 5 个类，其名称、包含的属性和操作如下图所示。



在 Button 中，List 类、ComboBox 类、TextBox 类以及 Label 类都作为 Button 类的属性，即这些类都与 Button 类关联且被其引用。同理，根据其他类之间的引用关系，我们可以对类进行关联，得到类图如下。

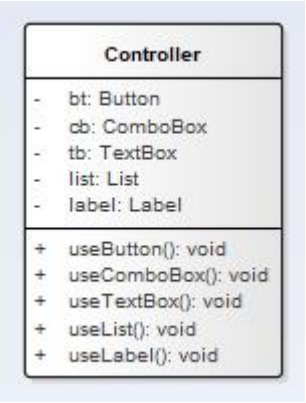


3.2. 根据迪米特法则对上述类图进行重构，降低组件之间的耦合度，绘制重构后的类图

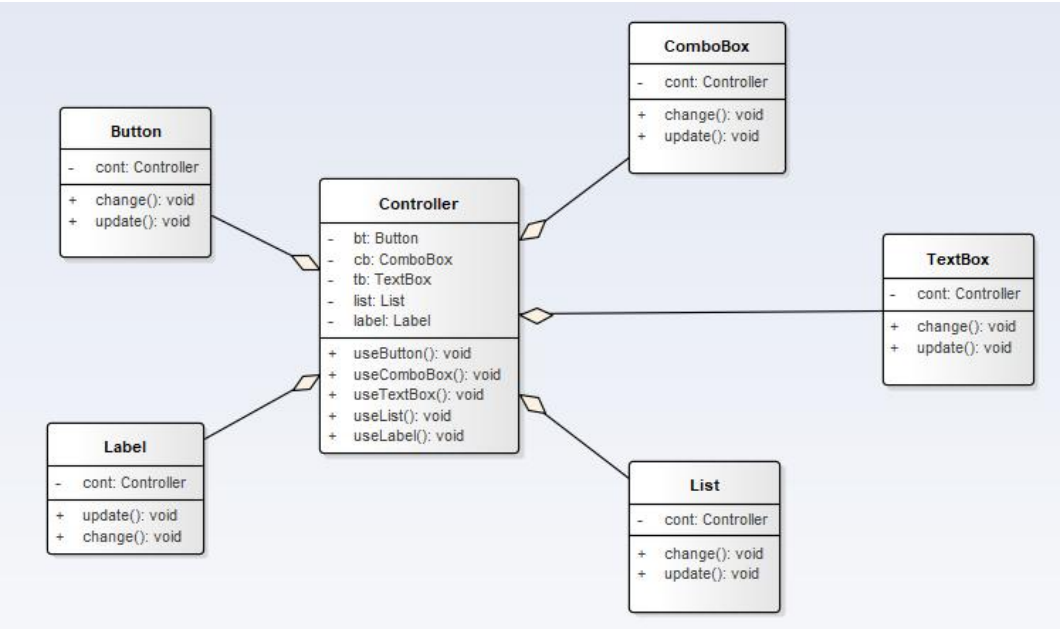
迪米特法则要求系统中不必彼此直接通信的对象不应该发生任何直接的相互作用，对于对象之间的相互调用，可以引入第三者转发这个调用。通过第三者的引入，降低现有对象之间的耦合度。

在上图中，5 个对象之间都存在相互调用，使整个系统的调用关系较为复杂，根据迪米特法则，可以在系统中引入一个中间类 Controller 对这些类之间的调用进行转发。

对于引入的中间类 Controller，需要包含对应这 5 个类的属性值以及调用方法，设计中间类如下

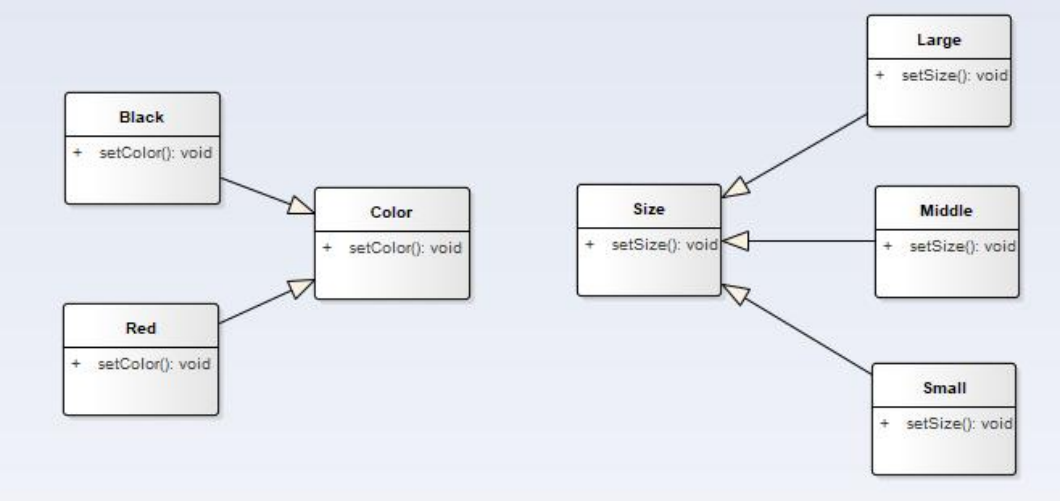


此时，对于这 5 个类之间的调用关系，就可以通过中间类 Controller 完成。在这 5 个类中，只需保证包含 Controller 类的属性值即可。另外，对于 Label 类，因为此时引用了 Controllrt 类，也需要加入 change()方法。完成重构的类图绘制如下



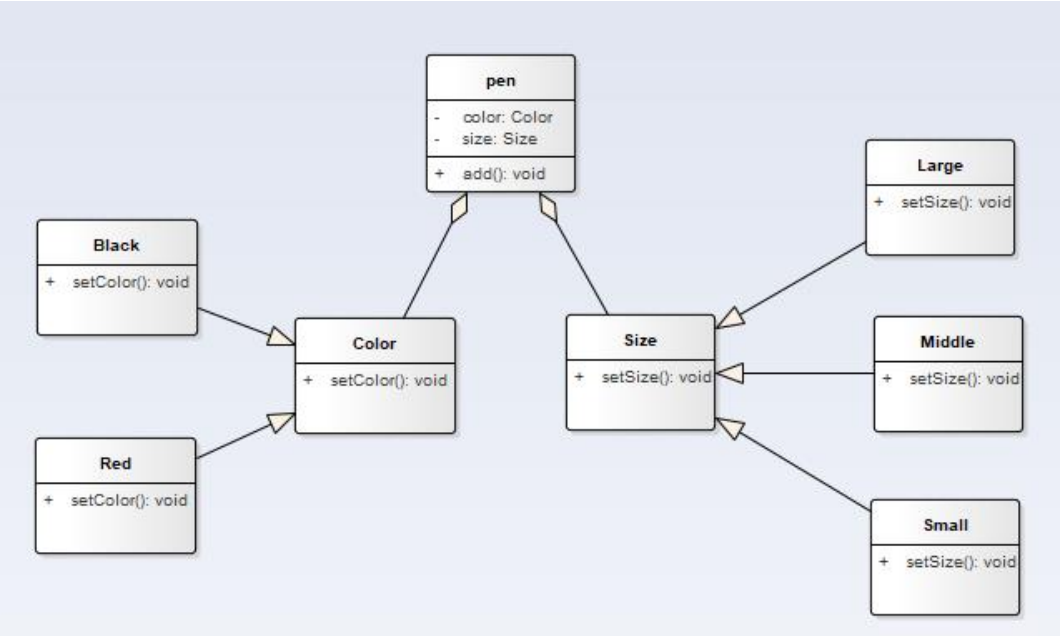
4. 根据依赖倒转原则和合成复用原则，对画笔结构初始类图进行重构。使得增加新的大小或颜色的笔都较为方便。

在初始类图中，该系统若增加一种绿色的笔，则需要在每个表示大小的类下增加一个新类，那个此时表示大小的类的代码就需要进行修改，以使用新的类，违背了开闭原则。根据依赖倒转原则，我们可以定义一个表示颜色的类 Color，在 Color 种定义设置颜色的方法 setColor()，那么增加一种颜色的笔，就可以直接在 Color 类中增加该种颜色。同样的，可以定义一个表示大小的类 Size，并定义方法 setSize()，对笔的大小进行设置。



对于一支笔，在初始类图中，表示不同大小的类分别继承 pen 类，当我们定义了表示颜色的 Color 类和表示大小的 Size 类之后，根据合成复用原则，可以使用聚合关系实现对笔的颜色和大小类的功能复用。此时则要求 pen 类中具有 color 和 size 两个属性，并具备一个添加新笔的方法 add()，可以调用 Color 中的 setColor 方法和 Size 类中的 setSize 方法，实现对新笔的添加。

完成重构的类图如下



四、实验总结与体会

通过本次实验，认识并理解了单一职责原则、开闭原则、依赖倒转原则和合成复用等 OO 设计原则。

对于设计存在问题的类图，能够根据这些原则对类图中存在的问题进行分析，并对其进行重构。对类图的设计和分析有了更深的掌握。

五、成绩评定及评语

1.指导老师批阅意见：

2.成绩评定：

指导教师签字：毛斐巧

2023 年　　月　　日