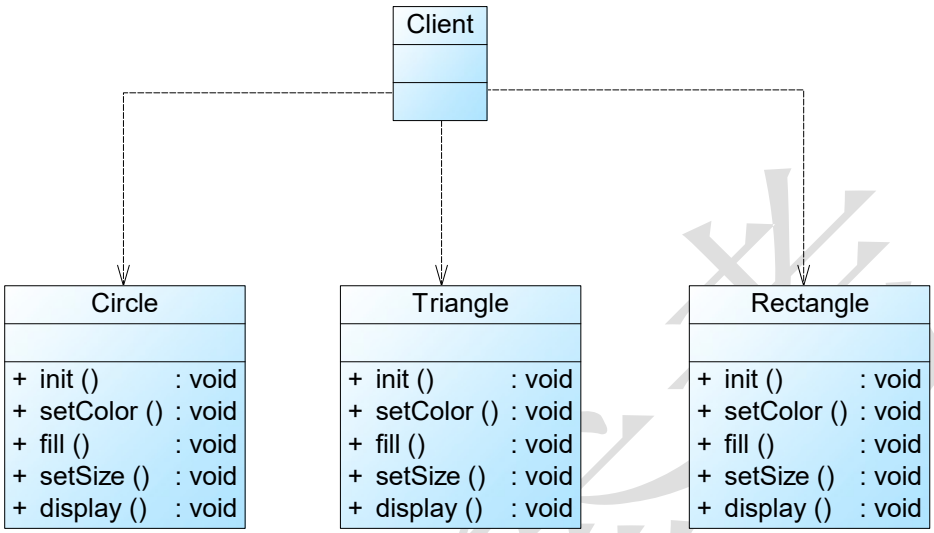
实验 2 设计模式实验一参考答案

【供学习参考，请勿在互联网传播！】

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。

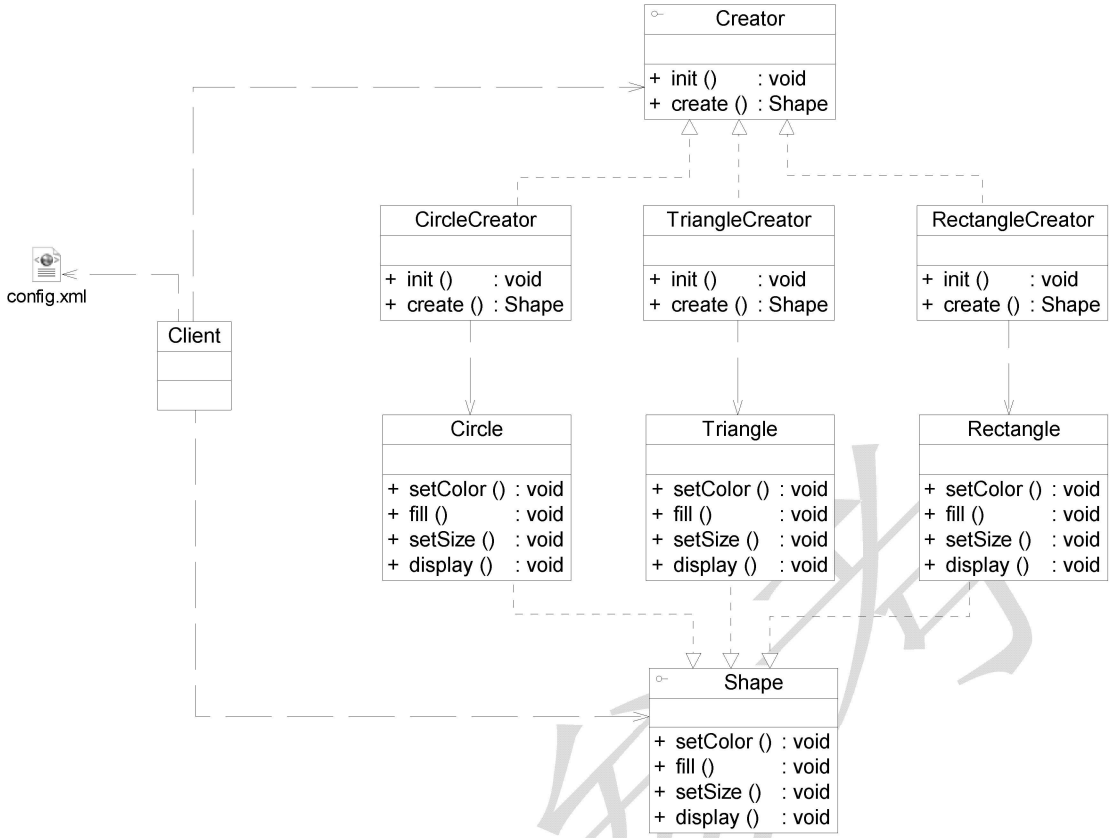
现需要根据面向对象设计原则对该系统进行重构，要求如下：

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。

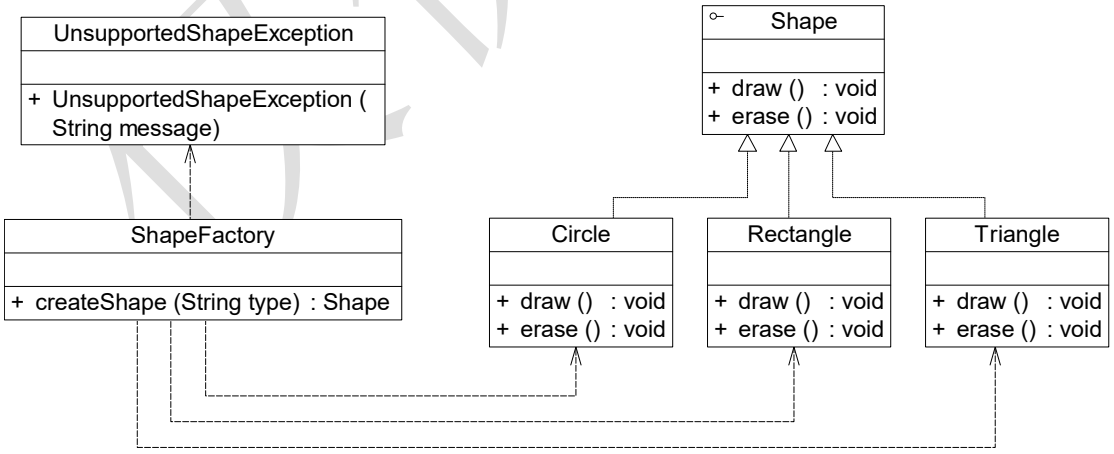
绘制重构之后的类图并说明在重构过程中所运用的面向对象设计原则。

参考答案：



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。

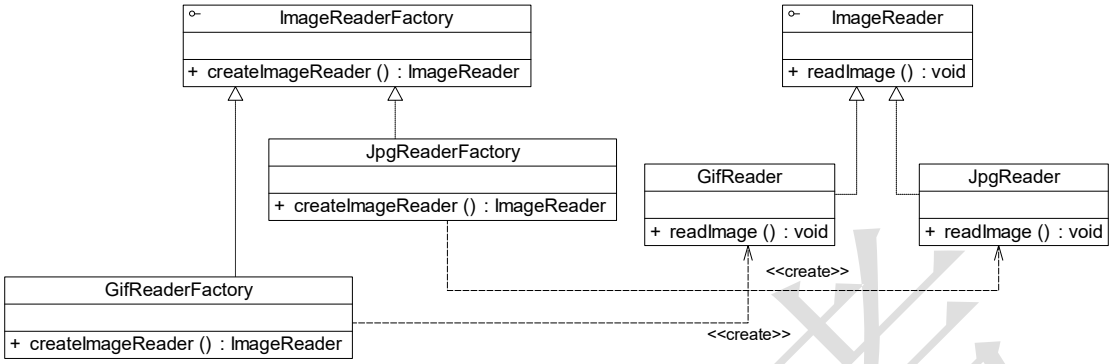
参考答案：



3. 现需要设计一个程序来读取多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)，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。图片读取器对象通过图片读取器工厂 ImageReaderFactory 来创建，ImageReaderFactory 是一个抽象类，用于定义创建图片读取器

的工厂方法，其子类 GifReaderFactory 和 JpgReaderFactory 用于创建具体的图片读取器对象。试使用工厂方法模式设计该程序，绘制类图并编程模拟实现。需充分考虑系统的灵活性和可扩展性。

参考答案：



4. 抽象工厂模式最早的应用之一是用来创建在不同操作系统的图形环境下都能够运行的应用程序，比如同时支持 Windows 与 Linux 操作系统。在每一个操作系统中，都有一个由图形构件组成的构件家族，可以通过一个抽象角色给出功能定义，而由具体子类给出不同操作系统下的具体实现，例如系统中包含两个产品等级结构，分别是 Button 与 Text；同时包含三个产品族：Unix 产品族、Linux 产品族与 Windows 产品族，如图 1 所示。

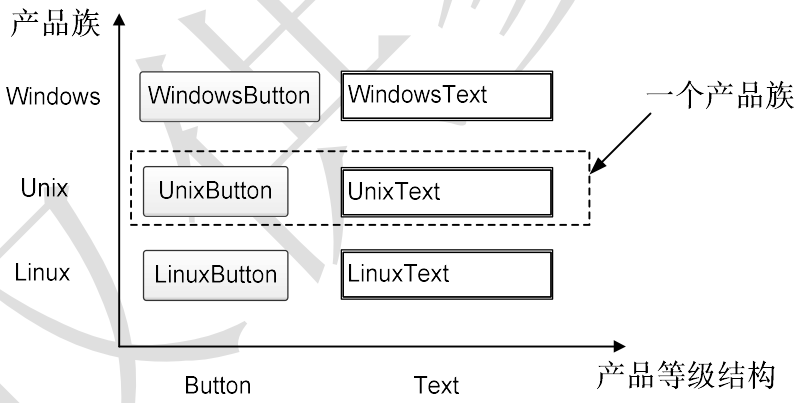
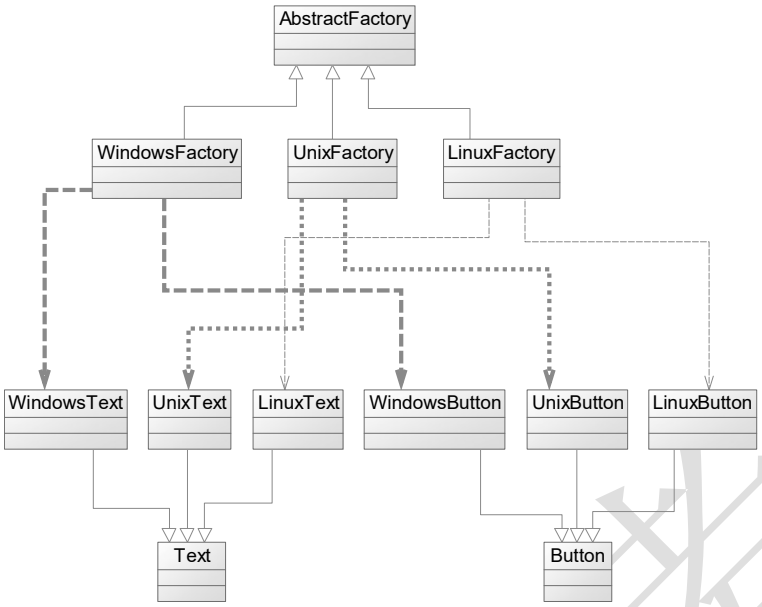


图 1 不同操作系统下不同构件的“产品等级结构-产品族”示意图

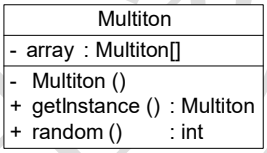
在图 1 中，Windows 中的 Button 和 Text 构成了一个 Windows 产品族，而不同操作系统下的 Button 构成了一个产品等级结构。试使用抽象工厂模式来设计并模拟实现该结构。

参考答案：



5. 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：



多例模式(Multiton Pattern)是单例模式的一种扩展形式，多例类可以有多个实例，而且必须自行创建和管理实例，并向外界提供自己的实例，可以通过静态集合对象来存储这些实例。多例类 Multiton 的代码如下所示：

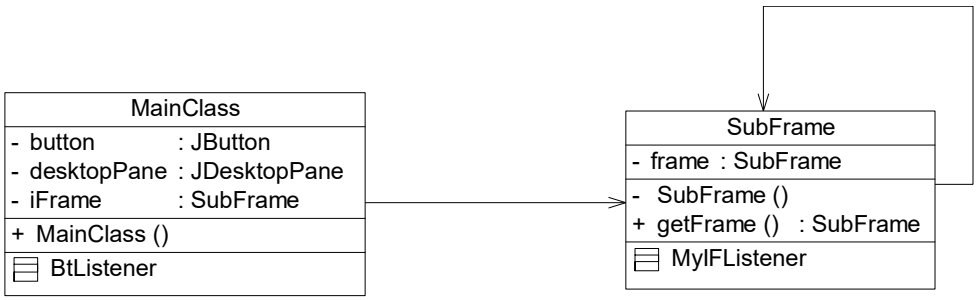
```
import java.util.*;

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);
    }
}
```

6. 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：



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;
            }
        }
```

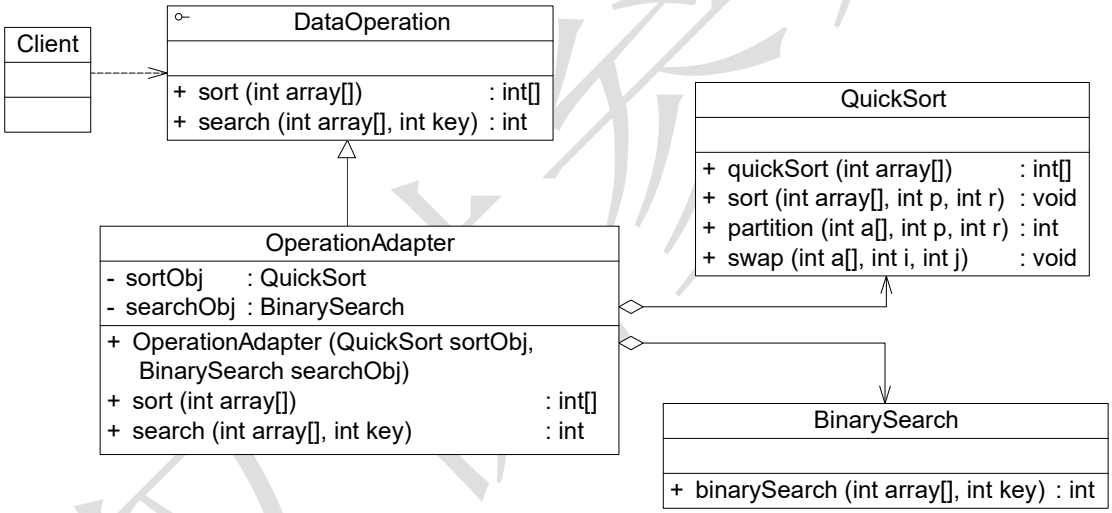
```
    }
}

//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();
    }
}
```

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。

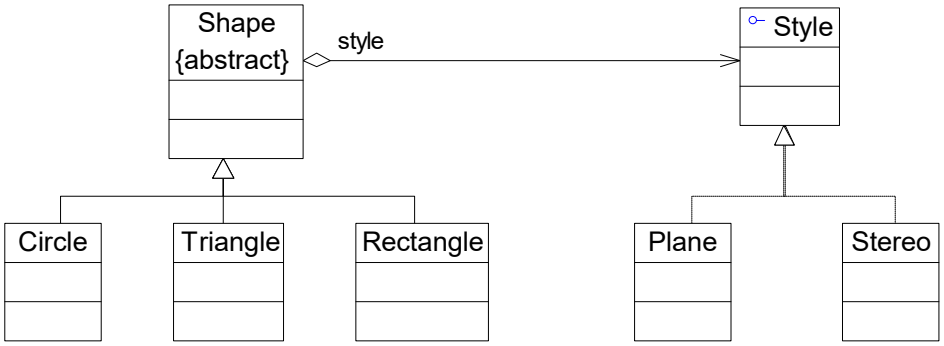
7. 现有一个接口 DataOperation 定义了排序方法 sort(int[]) 和查找方法 search(int[], int)，已知类 QuickSort 的 quickSort(int[])方法实现了快速排序算法，类 BinarySearch 的 binarySearch(int[], int)方法实现了二分查找算法。试使用适配器模式设计一个系统，在不修改源代码的情况下将类 QuickSort 和类 BinarySearch 的方法适配到 DataOperation 接口中。绘制类图并编程实现。（要求实现快速排序和二分查找，使用对象适配器实现）

参考答案：



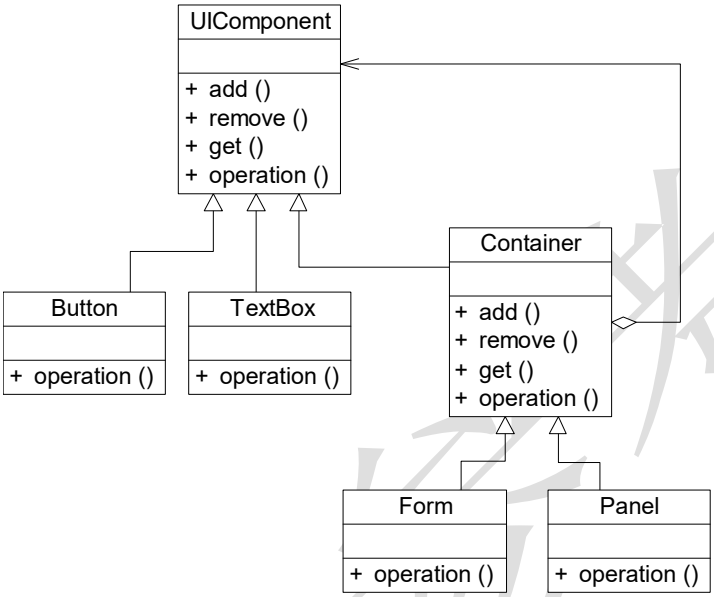
8. 某图形绘制软件提供了多种不同类型的图形，例如圆形、三角形、长方形等，并为每种图形提供了多种样式，例如平面图形、立体图形等。该软件还需经常增加新的图形及新的图形样式，采用桥接模式设计该图形绘制软件，绘制类图并编程模拟实现。

参考答案：



9. 某软件公司欲开发一个界面控件库，界面控件分为两大类，一类是单元控件，例如按钮、文本框等，一类是容器控件，例如窗体、中间面板等，试用组合模式设计该界面控件库，绘制类图并编程模拟实现。

参考答案：【安全组合模式或透明组合模式均可】



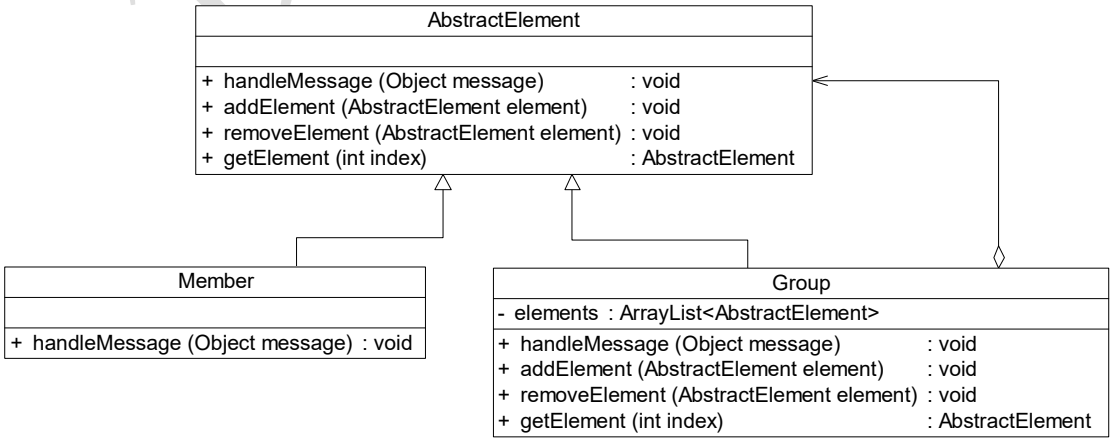
10. 某移动社交软件欲增加一个群组(Group)功能。通过设置，用户可以将自己的动态信息，包括最新动态、新上传的视频以及分享的链接等，分享给某个特定的成员(Member)，也可以分享给某个群组中的所有成员；用户可以将成员添加至某个指定的群组；此外，还允许用户在一个群组中添加子群组，以便更加灵活地实现面向特定人群的信息共享。

选择一种合适的设计模式来设计该群组功能，绘制类图并编程模拟实现。

参考答案：

所选模式：组合模式

参考类图【安全组合模式或透明组合模式均可】:



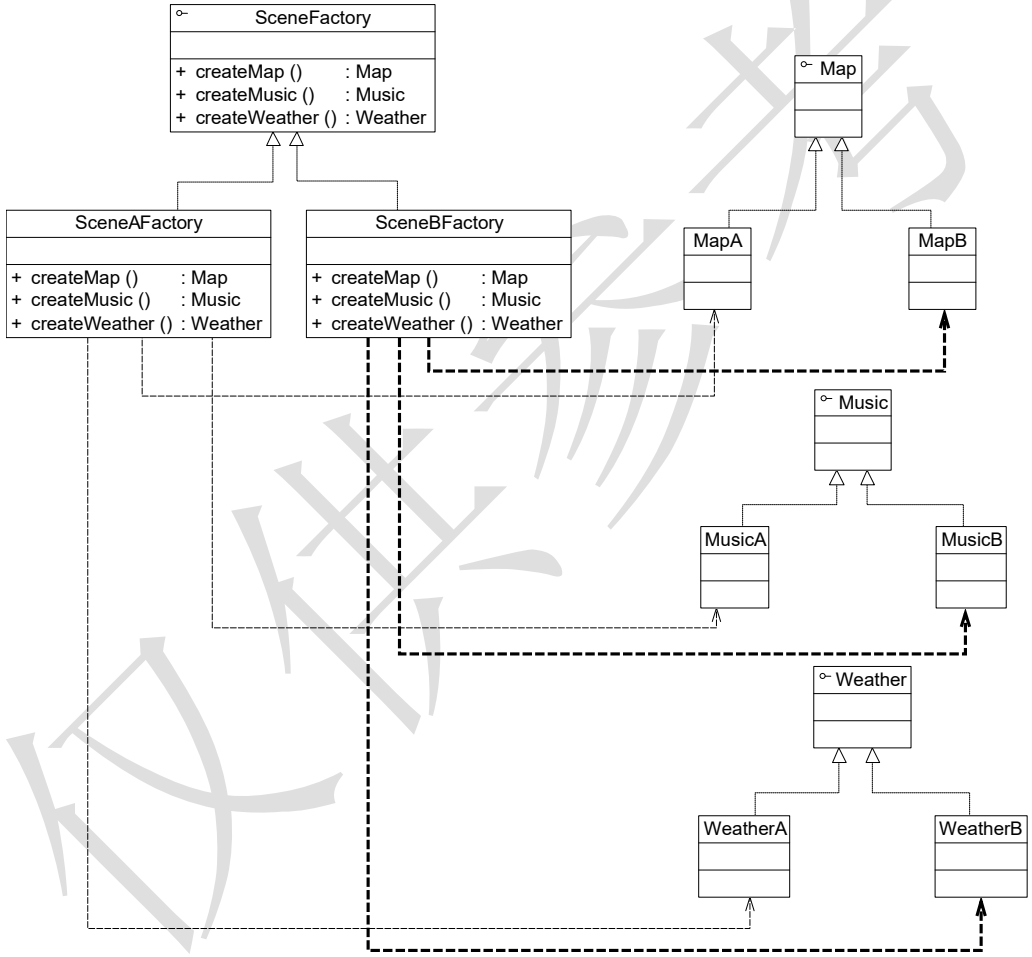
11. 在某 FPS(First-Person Shooting Game，第一人称射击) 游戏中提供了多个不同的游戏场景。在每一个游戏场景中，提供了对应的地图(Map)、天气(Weather)和游戏背景音乐(Sound)等。

请选择一种合适的设计模式对游戏场景进行设计，使得当用户选择游戏场景时，该场景所对应的地图、天气和背景音乐能够同时出现；此外，还可以方便地在该游戏中增加新的游戏场景。绘制类图并编程模拟实现。

参考答案：

所选设计模式：抽象工厂模式

参考类图：



12. In a First-Person Shooting (FPS) Game, a building or a blindage(掩体) is a 3D structure that consists of many 3D Objects such as Cube(立方体), Cylinder(圆柱体), Pyramid(锥体) etc. When we fill a 3D block with color (such as Gray), the same color also gets applied to the Objects in the block. Here a 3D block is made up of different parts and they all have same operations. The parts of a 3D block can be small blocks.

Which design pattern can be used to implement the 3D structure? Draw its structure diagram with this sample and write enough code to describe your solution.

参考答案：

所选设计模式：组合模式

参考类图【安全组合模式或透明组合模式均可】: