组合模式

组合模式是一种结构型模式，用于当我们不得不代表整个层次的一部分。当我们需要创建一个结构以一种在这个结构中的对象都以同样的方式被处理。我们可以应用组合设计模式。

让我们用一个实际生活中的例子来理解它。一个图像是一个结构，包含的对象比如圆形，直线，三角形等，当我们使用颜色来绘图的时候，同一种颜色在绘画的过程中也会被应用到其他对象。这里，绘图由不同的部分组成，它们都拥有相同的操作。

组合模式包含下面几个对象：

- **基础组件** - 基础组件在组合中是所有对象的接口，客户端程序使用基础组件来操作组合中的对象。它可以是一个接口也可以是一个拥有对所有对象都通用的一些方法的抽象类。
- **叶子** - 定义组合中元素的行为。它是组合的构建块并实现了基础组件。他没有对其他组件的引用。
- **组合** - 它包含叶子元素并实现了基础组件中的操作。

基础组件

基础组件为叶子和组合定义了公共方法，我们可以创建一个`Shape`类，并定义一个`draw(String fillColor)`方法来使用给定的颜色绘制图形。

Shape.java

```
package com.journaldev.design.composite;

public interface Shape {

    public void draw(String fillColor);
}
```

叶子对象

Triangle.java

```
package com.journaldev.design.composite;

public class Triangle implements Shape {

    @Override
    public void draw(String fillColor) {
        System.out.println("Drawing Triangle with color
"+fillColor);
    }

}
```

Circle.java

```
package com.journaldev.design.composite;

public class Circle implements Shape {

    @Override
    public void draw(String fillColor) {
        System.out.println("Drawing Circle with color
"+fillColor);
    }

}
```

组合

Drawing.java

```
package com.journaldev.design.composite;

import java.util.ArrayList;
import java.util.List;

public class Drawing implements Shape{

    //collection of Shapes
    private List<Shape> shapes = new ArrayList<Shape>();

    @Override
    public void draw(String fillColor) {
        for(Shape sh : shapes)
        {
            sh.draw(fillColor);
        }
    }

    //adding shape to drawing
    public void add(Shape s){
        this.shapes.add(s);
    }

    //removing shape from drawing
    public void remove(Shape s){
        shapes.remove(s);
    }

    //removing all the shapes
    public void clear(){
        System.out.println("Clearing all the shapes from
drawing");
        this.shapes.clear();
    }
}
```

测试

TestCompositePattern.java

```
package com.journaldev.design.test;

import com.journaldev.design.composite.Circle;
import com.journaldev.design.composite.Drawing;
import com.journaldev.design.composite.Shape;
import com.journaldev.design.composite.Triangle;

public class TestCompositePattern {

    public static void main(String[] args) {
        Shape tri = new Triangle();
        Shape tri1 = new Triangle();
        Shape cir = new Circle();

        Drawing drawing = new Drawing();
        drawing.add(tri1);
        drawing.add(tri1);
        drawing.add(cir);

        drawing.draw("Red");

        drawing.clear();

        drawing.add(tri);
        drawing.add(cir);
        drawing.draw("Green");
    }

}
```

输出

```
Drawing Triangle with color Red
Drawing Triangle with color Red
Drawing Circle with color Red
Clearing all the shapes from drawing
Drawing Triangle with color Green
Drawing Circle with color Green
```

组合模式的关键点

- 仅当一组对象表现像单个的对象时可以使用组合模式。
- 组合模式可以用来创建类似于树的结构。