软件体系结构

Zhenyan Ji

Beijing Jiaotong University

结构型模式--装饰器模式

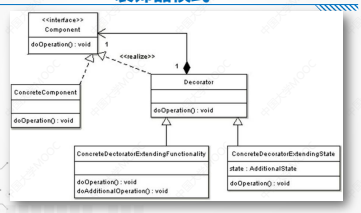
装饰器模式

01 装饰器模式可修改单个对象的行为，却无需创建新的派生类。

02 通过创建对原始类进行包裹的装饰器类来实现。

03 目的：动态地向对象添加职责。

装饰器模式



装饰器模式

参与者

- Component
  - 定义可以动态添加职责的对象的接口。
- ConcreteComponent
  - 定义一个可以添加职责的对象。

装饰器模式

Decorator

- 维护对组件对象的引用，并定义符合组件接口的接口。

ConcreteDecorator

- 向组件添加职责。

装饰器模式和子类

装饰器模式可避免创建过多的子类。

- 子类在编译时添加行为，更改可影响原始类的所有实例。
  - 需要为每个可能的组合都创建一个新类。

装饰器模式和子类

- 装饰器可在运行时为单个对象添加新的行为。
  - 装饰器是运行时创建的对象，可以根据使用进行组合。

例子：装饰器模式

例子：装饰器基类

```
public class Decorator extends Jcomponent {
public Decorator(JComponent c) {
        setLayout(new BorderLayout());
        //add component to container
        add("Center", c);
  }
}
```

例子：具体装饰类

```
public class CoolDecorator extends Decorator{
boolean mouse_over;
   //true when mouse over button
JComponent thisComp;
public CoolDecorator(JComponent c) {
        super(c);
        mouse_over = false;
        thisComp = this; //save this component
        //catch mouse movements in inner class
```

例子：具体装饰类

```
c.addMouseListener(new MouseAdapter()        {
        public void mouseEntered(MouseEvent e) {
                mouse_over=true;
                //set flag when mouse over
                thisComp.repaint();
        }
        public void mouseExited(MouseEvent e) {
                mouse_over=false;
                //clear if mouse not over
                thisComp.repaint();
        }
  });
}
```

例子：具体装饰类

```
public void paint(Graphics g) {
        super.paint(g);
        if(! mouse_over) {
                Dimension size = super.getSize();
                g.setColor(Color.lightGray);
                g.drawRect(0, 0, size.width-1, size.height-1);
                g.drawLine(size.width-2, 0, size.width-2,
size.height-1);
                g.drawLine(0, size.height-2, size.width-2,
size.height-2);
        }
    } }
```

例子：装饰器模式

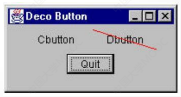
```
super ("Deco Button");
JPanel jp = new JPanel();
getContentPane().add(jp);
jp.add( new CoolDecorator(new JButton("Cbutton")));
jp.add( new CoolDecorator(new JButton("Dbutton")));
jp.add(Quit = new JButton("Quit"));
Quit.addActionListener(this);
```

例子：装饰器模式

```
public class SlashDecorator extends Decorator {
        int x1, y1, w1, h1; //saved size and posn
        public SlashDecorator(JComponent c) {
                super(c);
        }
        //-----------------------------------------------
        public void setBounds(int x, int y, int w, int h) {
                x1 = x; y1= y; //save coordinates
                w1 = w; h1 = h;
                super.setBounds(x, y, w, h);
        }
```

例子：装饰器模式

```
//-----------------------------------------------
    public void paint(Graphics g) {
            super.paint(g); //draw button
            g.setColor(Color.red); //set color
            g.drawLine(0, 0, w1, h1); //draw red line
    }
}
```



例子：装饰器模式

```
jp.add(new SlashDecorator( new CoolDecorator(new
JButton("Dbutton"))));
```