概念

一块蛋糕，加上一个草莓，那么就成了水果蛋糕，如果在加上巧克力，那么就成了水果巧克力蛋糕，如果在加上几个蜡烛写上生日祝福，就果蛋糕，水果巧克力蛋糕，生日蛋糕，它们的本质都是蛋糕。 草莓，巧克力，蜡烛，文字都是装饰品，在不改变蛋糕本质的情况下增加功能

角色

Component：被装饰的角色接口
ConcreteComponent：被装饰的具体角色类
Decorator：装饰器接口
ConcreteDecorator：具体装饰物类

代码

```
/**
 * 显示文字的接口
 */
public abstract class Display {
   public abstract int getColumns();        //获取横向字符数
   public abstract int getRows();          //获取纵向行数
   public abstract String getRowText(int row); //获取第row行的字符串

   public final void show(){
       for(int i = 0 ; i < getRows() ; i ++){
          System.out.println(getRowText(i));
       }
    }
}

/**
 * 显示文字具体实现类
 */
public class StringDisplay extends Display{

   private String string;

   public StringDisplay(String string){
      this.string = string;
   }

   @Override
   public int getColumns() {
      return string.getBytes().length;
   }

   @Override
   public int getRows() {
      return 1;
   }

   @Override
```

```
38      public String getRowText(int row) {
39          if(row == 0){
40              return string;
41          }
42          return null;
43      }
44
45  }
46
47  /**
48   * 装饰文字的接口
49   */
50  public abstract class Border extends Display{
51
52      protected Display display;
53
54      public Border(Display display) {
55          this.display = display;
56      }
57
58  }
59
60  /**
61   * 装饰文字实现类：在文字前后加上边框
62   */
63  public class SideBorder extends Border{
64
65      private char borderChar;
66
67      public SideBorder(Display display ,char ch) {
68          super(display);
69          this.borderChar = ch;
70      }
71
72      @Override
73      public int getColumns() {
74          return 1 + display.getColumns() + 1;
75      }
76
77      @Override
78      public int getRows() {
79          return display.getRows();
80      }
81
82      @Override
83      public String getRowText(int row) {
84          return borderChar + display.getRowText(row) + borderChar;
85      }
86
87  }
88
89  /**
90   * 装饰文字实现类：在文字前后加上边框
91   */
```

```
92 public class FullBorder extends Border{
93
94    public FullBorder(Display display) {
95       super(display);
96    }
97
98    @Override
99    public int getColumns() {
100       return 1 + display.getColumns() + 1;
101    }
102
103    @Override
104    public int getRows() {
105       return 1 + display.getRows() + 1;
106    }
107
108    @Override
109    public String getRowText(int row) {
110       if(row == 0){
111          return "+" + makeLine('-', display.getColumns()) + "+";
112       }else if(row == display.getRows() + 1){
113          return "+" + makeLine('-', display.getColumns()) + "+";
114       }
115       return "|" + display.getRowText(row - 1) + "|";
116    }
117
118    private String makeLine(char ch , int count){
119       StringBuffer buff = new StringBuffer();
120       for(int i = 0 ; i < count ; i ++){
121          buff.append(ch);
122       }
123       return buff.toString();
124    }
125
126 }
127
128 public class Main {
129
130
131    public static void main(String[] args) {
132
133       Display b1 = new StringDisplay("hello world");
134       Display b2 = new SideBorder(b1, '#');
135       Display b3 = new FullBorder(b2);
136
137       b1.show();
138       System.out.println();
139       b2.show();
140       System.out.println();
141       b3.show();
142       System.out.println();
143       Display b4 = new SideBorder(
144                       new FullBorder(
145                          new FullBorder(
```

```
                                new SideBorder(
                                      new FullBorder(new StringDisplay("你好世界"))
                                      , '*')
                                )
                            ),
                    '/');
        b4.show();
    }
}


、
/*

hello world

#hello world#

+--------------+
|#hello world#|
+--------------+

/+------------------+/
/|+----------------+|/
/||*+------------+*||/
/||*|你好世界|*||/
/||*+------------+*||/
/|+----------------+|/
/+------------------+/

*/
```

说明：在jdk中下面代码也是使用的此模式

```
Reader reader = new LineNumberReader(new BufferedReader(new FileReader("file")));
```

总结

优点：

1、一般的，我们为了扩展一个类经常使用继承方式实现，由于继承为类引入静态特征，并且随着扩展功能的增多，子类会很膨胀。装饰互耦合，装饰模式是继承的一个替代模式

缺点：

装饰层级太多就会比较复杂

使用场景

1、扩展一个类的功能

2、动态增加功能，动态撤销