链滴

设计模式之装饰器模式

作者：sologxl

原文链接：https://ld246.com/article/1543288073427

来源网站：链滴

大家好，我闲来无事，又来说说模式了。

这次说一下装饰器模式吧。听名字，是不是感觉很简单，又不知道啥意思呢？嘻嘻，原谅我话痨，我是喜欢在正经的时候说一些不正经的话。

装饰的意思，就是外表打扮一下，不涉及到本体，这个是符合开闭原则的，即对扩展开放，对修改关。

举个不恰当的例子，如果我们去买一个花瓶，买来发现颜色不喜欢，这个时候我们第一时间肯定不是一个，因为这样太浪费钱了，更不是重新打造一个，伤筋动骨，不划算，很简单，在不影响本体的情下，买个颜料，涂上去就好了，是不是很简单呢？

而且这个颜料我们可以任意调，在这个过程中，丝毫不影响花瓶的状态。弄好涂上去就好了。

好了，说了那么多废话，我也满足了，接下来我弄个代码示例把。

弄个接口

```
public interface Shape {
  void draw();
}
```

弄个实体花瓶

```
public class Rectangle implements Shape {
  @Override
  public void draw() {
    System.out.println("Shape: Rectangle");
  }
}
```

弄个抽象装饰器，不要问我为啥要搞抽象装饰器，我也不知道，无非就是预防可能有多种装饰需求吧

其实在代码开发过程中，抽象和接口，可能已经变成了一种开发习惯和标准了。
你我都习惯就好，反正相信一下“古人”的经验吧，哈哈！

```
public abstract class ShapeDecorator implements Shape {
  protected Shape decoratedShape;
  public ShapeDecorator(Shape decoratedShape){
    this.decoratedShape = decoratedShape;
  }
  public void draw(){
    decoratedShape.draw();
  }
}
```

弄个实体装饰器

```
public class RedShapeDecorator extends ShapeDecorator {
  public RedShapeDecorator(Shape decoratedShape) {
    super(decoratedShape);
  }
  @Override
  public void draw() {
    decoratedShape.draw();
    setRedBorder(decoratedShape);
  }
```

```
  private void setRedBorder(Shape decoratedShape){
     System.out.println("Border Color: Red");
  }
}
```

现在我们来实现这个花瓶，然后在看看实现换颜色的花瓶

```
public class DecoratorPatternDemo {
  public static void main(String[] args) {
    Shape Rectangle = new Rectangle(); //创建一个长方形的花瓶。
    Rectangle.draw(); //创建成功
    /** 此时，如果喜欢花瓶是红色的，那么使用普通方式我们可能需要去Rectangle类中修改代码，
于已经成型的代码，修改往往会带来致命的错误，就是致命，爱听不听。
    所以我们就要使用其他思维方式，那就是装饰器的作用了，请看下面代码：**/
    //我们直接创建一个装饰器，将我们的花瓶对象传入进去。
    Shape redRectangle = new RedShapeDecorator(new Rectangle());
    redRectangle.draw(); //自然我们就得到了一个红色的花瓶了
  }
}
```

随着对java设计模式的学习，我们会慢慢发现，编码开发的很重要一点，就是开闭原则和依赖调用原。都是为了确保解耦，确保代码优美易懂，确保安全的很有价值的开发思维。

希望大家要懂得提高代码开发的思想认识，而不是一味的崇尚新技术。技术永远是为了业务服务滴。了，说了这么多了，我要去吃个猪脚饭了，混成这样我也是醉了