

**实验报告**

|  |  |
| --- | --- |
| **课程名称**: | 设计模式 |
| **开课学期**: | 2020-2021学年 第2学期 |
| **专业**: | 软件工程 |
| **班级年级**: | 2018级 |
| **学生姓名**: | 宋行健 |
| **学号:** | 222018321062006 |
| **实验教师:** | 王晓蒙 |

**计算机与信息科学学院 软件学院**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 实验5 结构型模式2 | | | |
| 实验时间 | | 2021/4/20 | Type | \*验证性□设计性 □综合性 | |
| 1. 思路  首先对老师发的代码进行逐句研读，了解代码架构，并绘制类图。通过类图的展示，程序的逻辑逐渐明显。程序主要被封装在MainApp进行主流程，在这个类中进行了页面的布局和各种控件响应函数的创建，通过调试运行，这个程序可以通过点击主页面上的按钮，进行对图片不同元素进行添加和渲染。  整个代码中包含两个抽象类Entity和LookDecorator，分别定义了实体和装饰器。还有一个接口ILook，主要是规范了装饰器的渲染方法。另外有三个类Protection（保护罩）、Skin（皮肤）、Level（等级）分别是Player的三个附加元素。  2. 类图    3. 代码分析——装饰者模式  在程序中使用到了装饰者模式。其中包含了四大部分，抽象组件（Component）是Entity，具体组件也称为被装饰组件（ConcreteComponent）是Player，抽象装饰器（Decorator）是LookDecorator，具体装饰器（ConcreteDecorator）是Protection、Skin、Level，另外还有一个可动态扩展功能接口ILook。  Entity是一个抽象组件，用于定义可被动态扩展的实体类，其中包含了Player的基本属性，和通过各种方法构造Player的构造函数。    Player是被装饰组件的组件，能被附加额外功能的组件，它继承了Entity，同时实现了可动态扩展功能接口ILook。它在基础Entity实体类的基础上，增加了一个速度属性MAX\_SPEED。同时，它具体实现了ILook接口中的entityToDraw()方法。    LookDecorator是抽象装饰器类包含一个组件引用，定义了符合组件操作规范的接口，它继承了ILook可动态扩展功能接口，并被具体装饰器Protection、Skin、Level继承。它实现了entityToDraw()的公共代码，即图形之间的融合方式，并通过getImageForMerge() 获取要参与融合的装饰图片，由具体装饰类实现。    Protection、Skin、Level是具体装饰器，为具体组件Player扩展功能。它们也具体实现了getImageForMerge()方法，确定需要融合的图片，并根据每个具体装饰器不同的需求，构建了各自特有的属性。其中Level是最内层的装饰器，它的getImageForMerge()方法之间return了自己。   |  |  | | --- | --- | |  |  |     4. 运行结果  为了更好的理解代码，并观察元素的位置设置，我将Player类中的背景改成了红色，parameters.setFill(Color.RED)。    下图是未进行修改的原始代码程序。    5. 总结  该项目是装饰者模式的实现，其中包含了四大部分，抽象组件Entity，具体组件Player，抽象装饰器LookDecorator，具体装饰器Protection、Skin、Level，另外还有一个可动态扩展功能接口ILook。  装饰者能动态地给对象添加职责或功能，相比通过类扩展增加功能的方式更灵活。装饰者模式的**优点**是装饰类和被装饰类可以独立发展，不会相互耦合，装饰模式可以替代继承的代码模式，装饰模式可以动态扩展一个实现类的功能。它的**缺点**是多层装饰比较复杂，较难理解。 | | | | | |
| 教师评阅 | 实验内容（60%）： | | | |  |
| 实验思路（20%）： | | | |  |
| 实验总结（20%）： | | | |  |
| 实验成绩： | | | | |