装饰模式p62

案例引入

设计一个可以给人搭配不同衣服的服饰的系统,最开始的代码就是每穿一种服装就新增加一个方法

问题提出

- 描述：现在要增加超人的装扮，如果这个时候在person类里面修改代码就违反了开闭原则
- 解决: 服装与人分开，建立一个服装抽象类，然后再让不同的装扮去继承抽象服装类，然后去实现抽象的方法

问题进一步提出

- 描述: 每一次调用服饰类的方法也意味着把穿服装的过程暴露出来了。需要隐藏起来。同时通过服饰的组合出一个人完全有多种方案，同时把所需要的功能按照正确的顺序串联起来

装饰模式

- 描述：动态得给一个对象添加一些额外的职责，就增加功能而言，装饰模式比生成子类更加灵活
- UML描述:
    - 一个对象接口类Component,可以给这些对象动态地添加职责
    - 一个ConcreteComponent,定义一个具体的对象，可以给这些对象添加职责
    - 一个Decorator,装饰的抽象类，它是继承自Component类的，同时他也持有一个父类的引用
    - 若干个具体的装饰类，用来给Component来添加职责

装饰模式总结

- 为已有的功能动态地添加更多功能的一种方式
- 当系统需要新功能的时候，是向旧类中添加新的代码。这些代码通常装饰了原有类的核心职责或者主要行为
- 新加入的东西仅仅是为了满足一些只在某种特定情况之下才会执行的特殊行为的需要。比如有可能需要超人装扮
- 装饰模式把需要装饰的功能单独放在一个勒种，让这个类去包装所要装饰的对象
- 因此，在需要执行特殊行为的时候，客户代码就可以根据需要有顺序地执行

优点

- 类中的装饰功能从类中移出，简化原来的类
- 把类的核心职责与装饰功能分开
- 去除了相关类中的重复的装饰逻辑