裝饰器模式(Decorator Pattern)允许向一个现有的对象添加新的功能,同时又不改变其结构。这种类型的设计模式属于结构型模式,它是作为现有的类的一个包装。这种模式创建了一个装饰类,用来包装原有的类,并在保持类方法签名完整性的前提下,提供了额外的功能。

目的

动态地给一个对象添加一些额外的职责。就增加功能来说,装饰器模式相比生成子类更为灵活。

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。装饰器模式可以在不想增加很多子类的情况下扩展类。

被装饰抽象接口 (Component)

被装饰抽象接口提供一个接口, 定义被装饰的行为。

被装饰对象 (Concrete Component)

被装饰对象提供基本操作,装饰者会在这个基本操作上面进行扩展,被装饰对象不用关注其他的内容,只需要聚焦实现自己的业务即可。

抽象装饰器 (Decorator)

抽象装饰器提供一个装饰的通用模式,不需要实现具体的装饰操作,只需要定义好被装饰的行为即可。

具体装饰器 (Concrete Decorator)

实现抽象装饰器所要求的装饰接口,以便扩展被装饰接口。

优点

角色

装饰类和被装饰类可以独立发展,不会相互耦合,装饰模式是继承的一个替代模式,装饰模式 可以动态扩展一个实现类的功能。

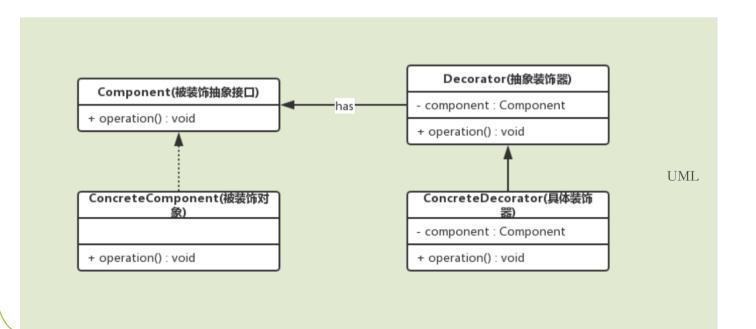
缺点

多层装饰比较复杂。

使用场景

扩展一个类的功能。

动态增加功能, 动态撤销。



装饰器模式