我将带大家一起去看一看在面向对象编程中出现的23大设计模式和9大设计原则。设计原则是我们开发研究设计模式的目的，而设计模式则是达到设计原则的手段和方法。在记录笔记过程中，会在不同的设计模式中穿插讲解设计原则。
本笔记知识点主要来自于《Head First 设计模式》

目前为止学习到的设计原则：

- 尽量将变化的和不变化的分离开来，把变化的包装成类；
- 针对接口编程，而不是针对实现编程
- 多用组合，少用继承
- 尽量保持交互对象之间的松耦合(loose coupling)

结构型模式一：装饰模式(Decorator Pattern)

定义：
装饰者模式动态的将责任添加到对象上，相比于继承来说，装饰者模式提供了一种更加具有扩展性的方式.

根据装饰者模式的定义，我们可以看出装饰者模式的特性：

- 装饰者和被装饰者的类型必须一致，也就是说二者之间可以相互替换；
- 装饰者模式的代码形式就像是层层包裹那样，把装饰者和被装饰者一个个包裹起来

设计原则：

- 对扩展开放，对修改关闭

装饰者模式的组成：

- 基本组件类(component): 这个类定义了所有装饰者和被装饰者的超类，所有归属这个模式的类都应该是这个类的子类；
- 具体实现被装饰者类：这个类继承了基本组件类，往往是作为被装饰者；
- 装饰者基本类：这个类也是继承基本组件类，但是又是作为其余装饰者类的基类；
- 装饰者实现类：这些类是继承装饰者基本类，进而也继承了基本组件类，因此和被装饰者类在本质上类型一致，这满足了装饰者模式的基本特性，因而也保证了被装饰者在装饰者装饰之后还能保持原有数据类型

> 在这些继承中，我们希望的是继承类型，而并非继承超类的行为

看下面这个例子：

```
//基本组件类
public abstract class Coffee {
    String description = "Unknown description";
    public String getDescription() {
        return description;
    }
    public abstract double getCost();
}
//具体实现被装饰者类
public class RoastCoffee extends {
    public RoastCoffee() {
        description = "RoastCoffee";
    }
    public double getCost() {
        return 1.99;
    }
}
//装饰者基本类
public abstract class Decorator extends Coffee {
    public abstract String getDescription();
}
//具体实现的装饰者类
public Mocha extends Decorator {
    Coffee coffee;
    public Mocha(Coffee coffee) {
        this.coffee = coffee;
    }
    public String getDescription() {
        return coffee.getDescription ()+ ", Mocha";
    }
    public double getCost() {
        return coffee.getCost() + 0.99;
    }
}
//测试类
public class Test {
    public static void main(String[] args) {
        Coffee coffee = new RoastCoffee();
        coffee = new Mocha(coffee);
        System.out.println(coffee.getDescription() + coffee.getCos
t());
    }
}
从上面的代码例子可以看到，Mocha并不是coffee，但是这了却继承类coffee这个超类，其原因在于在装饰者模式中，继承的主要目的在于继承类型，而非继承行为；
```

这就是结构型模式一：装饰者模式。