从本次笔记开始，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》

行为型模式一： 策略模式（Strategy Pattern）

策略模式的正式定义：***策略模式定义了算法族，分别封装起来，让他们之间可以相互替换，此模式让算法的变化独立于算法的客户。***
本章涉及到的设计原则：
设计原则一：***找出应用中需要变化的部分，然后把它们和那些不需要变化的部分独立开来，也就是封装变化；***
设计原则二：***针对接口编程，而不是针对实现编程；***
设计原则三：***多用组合，少用继承；***

策略模式适用场景：
当客户需要许多具有相关作用当类的时候，而这些类仅仅是在效果上会有不同；更加具体的就是客户可能会在这几种不同的类中进行切换。在这种情况下，我们可以将相关性较强的类抽离出来作为一个类接口，然后具有相关性的类实现这个接口但是具有各自特有的性质。

策略模式优缺点：

- 优点：在上面的适用场景已经提到，客户可以在不同的具体策略之间实现策略切换；
- 缺点：
    - 客户必须了解各种具体策略
    - 策略模式会存在众多的具体策略类

策略模式的组成：

- 环境类(Context)：这个环境类就是我们所说的客户，它一般拥有抽离出来的那个策略接口类，然后它可以根据环境以及需求的改变，在这些众多的策略中实现切换；
- 抽象策略类(Strategy)：这个类就是我们在许多相关类中抽离出来的那个策略类接口，它可能不止一个，而那些具有相关性的类实现这个接口并且可以各自定义自己的具体实现策略。这个抽象策略类的实例必须被客户也就是环境类拥有，然后根据环境类的切换进行接口实现类的随意切换；
- 具体策略类：这个类肯定不止一个，如果只有一个我们就没必要抽离出来一个抽象类进行接口化编程。这些类根据其作用选定上述的抽象策略类进行实现，并且给出自己特定的策略；

看下面的一个代码案例；该案例来自网络上博客:

```
1. //抽象策略接口类
2. public interface Strategy {
3.     public void operate();//定义一个操作，不给出具体实现
4. }
5. //三个具体策略实现类
6. public class Strategy1 implements Strategy {
7.     public void operate() {
8.         System.out.println("This is strategy 1.");
9.     }
10. }
11. public class Strategy2 implements Strategy {
12.     public void operate() {
13.         System.out.println("This is strategy 2.");
14.     }
15. }
16. public class Strategy3 implements Strategy {
17.     public void operate() {
18.         System.out.println("This is strategy 3.");
19.     }
20. }
21. //环境类
22. public class Context() {
23.     Strategy strategy;
24.     public Context(Strategy strategy) {
25.             this.strategy = strategy;
26.     }
27.     public void setStrategy(Strategy strategy) {
28.         this.strategy = strategy;
29.     }
30.     public void operate() {
31.         this.strategy.operate();
32.     }
33. }
34. //测试类
35. public class Test {
36.     public static void main(String[] args) {
37.         Context context;
38.         context = new Context(new Strategy1());
39.         context.operate();
40.         context.setStrategy(new Strategy2());//切换策略
41.         context.operate();
42.         context.setStrategy(new Strategy3());//切换策略
43.         context.operate();
44.     }
45. }
46. 上述代码分别输出:
47. This is strategy 1.
48. This is strategy 2.
49. This is strategy 3.
```

这就是**行为型模式**中提到的第一个模式：***策略模式.***