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》

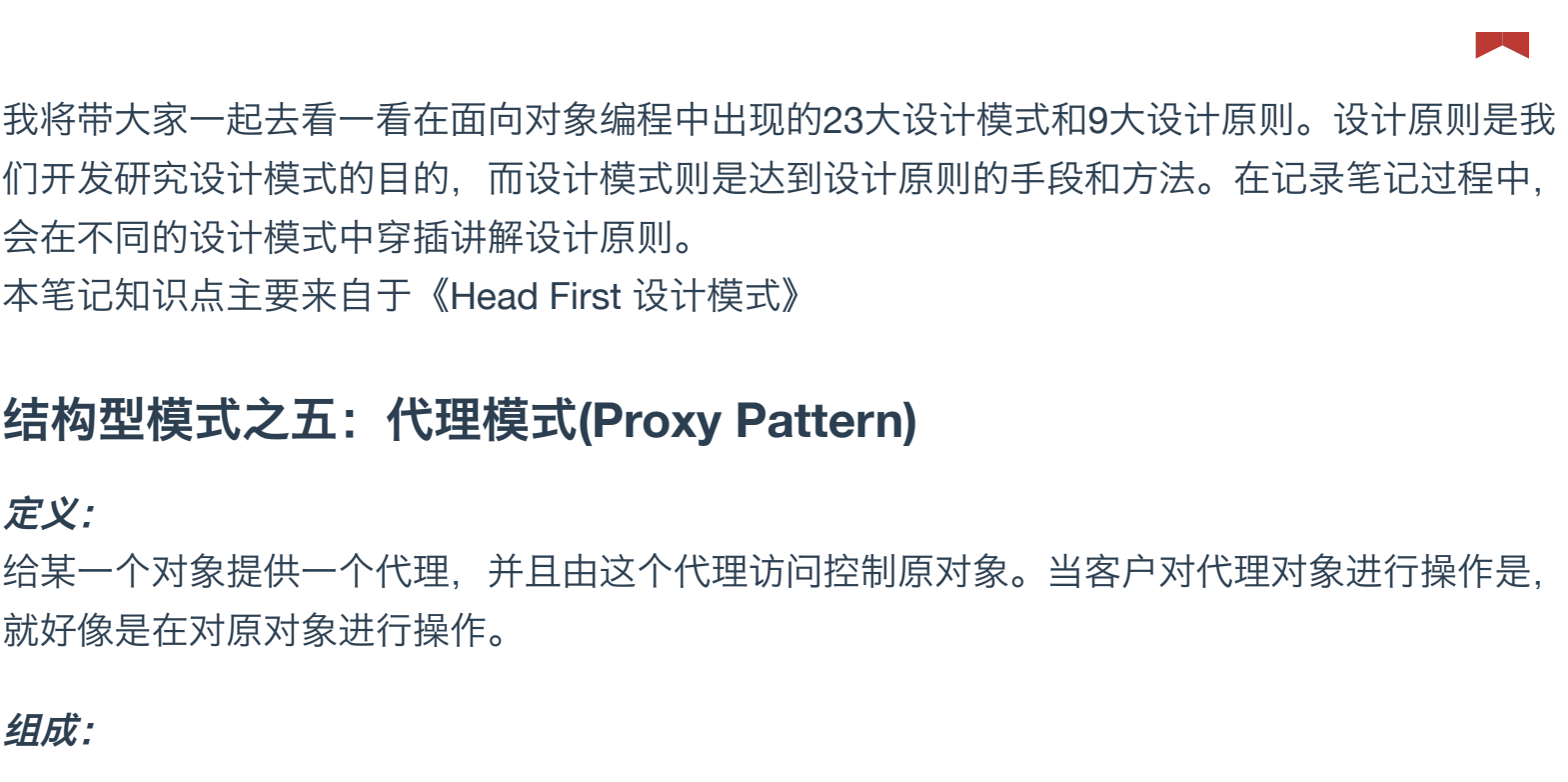
结构型模式之五：代理模式(Proxy Pattern)

定义：
给某一个对象提供一个代理，并且由这个代理访问控制原对象。当客户对代理对象进行操作是，就好像是在对原对象进行操作。

组成：

- 抽象主题角色
- 代理主题角色
- 真实主题角色

代理主题角色和真实主题角色都实现类抽象主题角色的接口，因此两者的方法以及调用方法并无明显差别。在代理主题角色当中含有一个真实主题角色变量以便用来真正调用在真实主题角色中的逻辑。



看代码：

```
1. 在本例子中定义了买车这一个抽象的主题角色，然后真实主题角色是人，代理主题角色是车行代理商。
2. //首先是抽象主题角色--买车
3. public interface Buy_car {
4.     public void buy_car();
5. }
6.
7. //然后是真实主题角色
8. public class People implements Buy_car {
9.     private int cash;
10.     private String name;
11.     private boolean isVip;
12.
13.     public People(boolean isVip, int cash, String name) {
14.         this.isVip  = isVip;
15.         this.cash = cash;
16.         this.name = name;
17.     }
18.
19.     public boolean getVip() {
20.         return isVip;
21.     }
22.
23.     public int getCash() {
24.         return cash;
25.     }
26.
27.     public String getName() {
28.         return name;
29.     }
30.     public void buy_car() {
31.         System.out.println(name + " is VIP, get a car...");
32.     }
33. }
34.
35. //代理主题角色
36. public Proxy implements Buy_car {
37.     private People people;
38.     public Proxy(People people) {
39.         this.people = people;
40.     }
41.
42.     public void buy_car() {
43.         if (people.getVip()) {
44.             people.buy_car();
45.             return;
46.         }
47.         else if (people.getCash() > 5000) {
48.             System.out.println(people.getName() + " buy a car.");
49.         }
50.         else {
51.             System.out.println(people.getName() + " can't afford a car...");
52.         }
53.     }
54. }
55. 在eclipse测试通过
```