**观察者模式**：

|  |
| --- |
| 一对多的对应关系，当某个对象状态发生变化时，同时给不同的观察者发送状态  Subject: 主题的接口，通知给observer的对象，例如前台的服务员，其内部包括所有要通知的对象，其中主要方法有：  registerObserver(),  removeObserver(),  notifyAll() => 通知所有的观察者对象，调用其update()方法  Observer: 所有观察者的接口，具有update()方法 |

**策略模式**：

|  |
| --- |
| 策略模式是一种定义一系列算法的方法，从概念上看，所有的算法都是完成相同的工作，只是实现不同，可以通过相同的方式调用，**减少了各种算法类与使用算法类之间的耦合**。  两个接口：1: Context类，内部含有一个Strategy对象  2: Strategy,所有的算法类的公用接口，定义一个方法getResult()  策略模式用来封装算法，但是它可以封装几乎任何类型的规则，只要在分析的过程中需要不同时间应用不同的业务规则，就可以考虑策略模式。  但是对于客户端，需要实现具体的职责，这样的话用switch语句来创建相应的策略对象，比较麻烦。 可以使用工厂模式的方式，将switch判断语句移到Context的子类中。 |

**职责链模式**：

|  |
| --- |
| 职责链模式：多个对象都有机会处理请求，避免请求的发送者与接受者之间的耦合性，将这些对象练成一条链，沿着链传递该请求，知道有一个对象处理为止。  一个接口：Handler类：两个方法=>handleRequest() 和 setSuccessor(Handler h)  关键是：当客户提交一个请求时，请求是沿着链直至一个ConcreteHandler对象负责.每个链中的对象不知道整个链的结构，他们尽需要保持一个指向其后继者的引用。  具体的例子：公司职员要求加薪，需要通过 项目经历=>总监=>总经理的每一步处理。 |