一、策略模式定义

在软件开发中常常遇到这种情况，实现某一个功能有多种算法或者策略，我们可以根据应用场景的不同选择不同的算法或者策略来完成该功能。

把一个类(A)中经常改变或者将来可能改变的部分提取出来，作为一个接口(B)，然后在类(A)中包含这个接口(B)，这样类(A)的实例在运行时就可以随意调用实现了这个接口的类(C)的行为。比如定义一系列的算法,把每一个算法封装起来, 并且使它们可相互替换，使得算法可独立于使用它的客户而变化。这就是策略模式。

二、策略模式的结构

•抽象策略：通常情况下为一个接口，当各个实现类中存在着重复的逻辑时，则使用抽象类来封装这部分公共的代码，此时，策略模式看上去更像是模版方法模式。

•具体策略：具体策略角色通常由一组封装了算法的类来担任，这些类之间可以根据需要自由替换。

•封装类：也叫上下文，对策略进行二次封装，目的是避免高层模块对策略的直接调用。

三、策略模式的特点

•策略模式提供了可以替换继承关系的办法。

•使用策略模式可以避免使用多重条件转移语句。

四、策略模式的缺点

1. 客户端必须知道所有的策略类，并自行决定使用哪一个策略类。这就意味着客户端必须理解这些算法的区别，以便适时选择恰当的算法类。换言之，策略模式只适用于客户端知道所有的算法或行为的情况。

2. 策略模式造成很多的策略类。有时候可以通过把依赖于环境的状态保存到客户端里面，而将策略类设计成可共享的，这样策略类实例可以被不同客户端使用。换言之，可以使用享元模式来减少对象的数量。