**策略模式**

在策略模式（Strategy Pattern）中，**一个类的行为或其算法可以在运行时更改**。这种类型的设计模式属于行为型模式。

**在策略模式中，我们创建表示各种策略的对象和一个行为随着策略对象改变而改变的 context 对象**。策略对象改变 context 对象的执行算法。

**介绍**

**意图：** 定义一系列的算法,把它们一个个封装起来, 并且使它们可相互替换(用到了组合)。

**主要解决：** 在有多种算法相似的情况下，使用 if...else 所带来的复杂和难以维护。

**何时使用：** 一个系统有许多许多类，而区分它们的只是他们直接的行为。

**如何解决：** 将这些算法封装成一个一个的类，任意地替换。

**关键代码：** 实现同一个接口。

**应用实例：** 1、诸葛亮的锦囊妙计，每一个锦囊就是一个策略。 2、旅行的出游方式，选择骑自行车、坐汽车，每一种旅行方式都是一个策略。

**优点：** 1、算法可以自由切换。 2、避免使用多重条件判断。 3、扩展性良好。

**缺点：** 1、策略类会增多。 2、所有策略类都需要对外暴露。

**使用场景：** 1、如果在一个系统里面有许多类，它们之间的区别仅在于它们的行为，那么使用策略模式可以动态地让一个对象在许多行为中选择一种行为。 2、一个系统需要动态地在几种算法中选择一种。 3、如果一个对象有很多的行为，如果不用恰当的模式，这些行为就只好使用多重的条件选择语句来实现。

**注意事项：**如果一个系统的策略多于四个，就需要考虑使用混合模式，解决策略类膨胀的问题。

**策略模式和状态模式的区别：**

**状态模式**：当对象的行为随对象的状态的改变而改变时，我们为了解耦多重判断条件，封装行为的变化，可以定义一个抽象的状态类，提供对象行为接口。具体与状态相关的行为，由它的子类去实现。**状态的转换不一定是放在Context中，有时候状态类本身也会自动切换状态。**

**策略模式**：“策略”二字等同于算法，当现实系统中需要对算法动态指定，并且可以互相替换，可以抽象出算法的调用接口，具体的算法实现由具体的策略角色去实现，根据里氏替换原则，任何出现父类的地方都可以使用它的子类去替换，这样符合我们的业务需求。