定义

它定义了算法家族，分别封装起来，让他们之间可以互相替换，此模式让算法的变化，不会影响到使用算法的客户

UML

应用场景

* **出现有许多相关的类，仅仅是行为有差别的情况，可以使用策略模式来使用多个行为中的一个来配置一个类的方法，实现算法动态切换**
* **出现同一个算法，有很多不同的实现的情况，可以使用策略模式来把这些“不同的实现”实现成为一个算法的类层次**
* **出现抽象一个定义了很多行为的类，并且是通过多个if-else语句来选择这些行为的情况，可以使用策略模式来代替这些条件语句**

优点

可以以相同的方式调用所有的算法，减少了各种算法类与使用算法类之间的耦合

策略模式的Strategy类层为Context定义了一系列的可供重用的算法或行为。继承有助于析出析出算法中的公共功能。

简化了单元测试

将行为封装在一个个独立的Strategy类中，可以再使用这些行为的类中消除条件语句。

缺点

改客户端必须知道所有的策略类，增加对象的数目，只适合扁平的算法结构

本质

分离算法选择实现

代码