设计模式：

软件领域中的设计模式为开发人员提供了一种使用专家设计经验的有效途径。设计模式中运用了面向对象编程语言的重要特性：封装、继承、多态，真正领悟设计模式的精髓是可能一个漫长的过程，需要大量实践经验的积累。最近看设计模式的书，对于每个模式，用C++写了个小例子，加深一下理解。主要参考《大话设计模式》和《设计模式:可复用面向对象软件的基础》两本书。本文介绍策略模式的实现。

下面介绍一下八大设计模式的原则：

一：依赖倒置原则

高层模块（稳定不应该依赖于底层模块（变化），二者都应该依赖于抽象（稳定）

抽象（稳定）不应该依赖于实现细节（变化），实现细节应该依赖于抽象（稳定）

二：开闭原则

对扩展开放，对更改关闭

类模块是可扩展的，但是不可更改

三：单一职责原则

一个类应该仅有一个引起它变化的原因

变化的方向隐含着类的责任

四：替换原则

子类必须能够替换他们的基类

继承表达类型抽象

五：接口隔离原则

不应该强迫客户程序依赖于他们不用的方法

接口应该小而完备

六：优先使用对象组合而不是类继承

类继承通常为“白箱复用”，而对象组合为“黑箱复用”

继承在某种程度上破坏了类的封装性，子类父类耦合度很高

而对像组合耦合度低

七：封装变化点

使用封装来创建对象之间的分界层，让设计者可以再分界层的一侧进行修改，而不会对另一侧产生影响，从而实现层次见的松耦合

八：针对接口编程，而不是针对实现

策略模式

定义：

策略模式定义了一系列的算法，并将每一个算法封装起来，使每个算法可以相互替代，使算法本身和使用算法的客户端分割开来，相互独立。

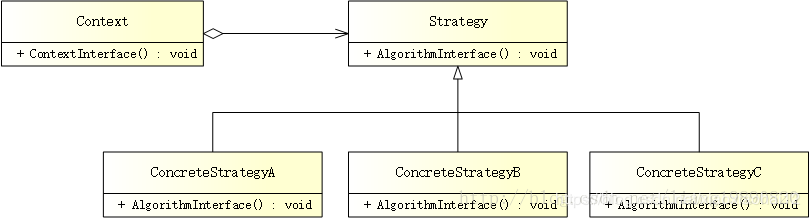
结构：

1.策略接口角色IStrategy:用来约束一系列具体的策略算法，策略上下文角色ConcreteStrategy使用此策略接口来调用具体的策略所实现的算法。

2.具体策略实现角色ConcreteStrategy:具体的策略实现，即具体的算法实现。

3.策略上下文角色StrategyContext:策略上下文，负责和具体的策略实现交互，通常策略上下文对象会持有一个真正的策略实现对象，策略上下文还可以让具体的策略实现从其中获取相关数据，回调策略上下文对象的方法。

UML图例



其中starategy就是一个接口类（稳定），concretestrategy就是具体类，根据不同进行扩展，而不是修改基类，context就想一个“工厂”（稳定），其中的具体调用是稳定的，不会变化

下面是一个小例

