## 一句话概括：

一个类的行为或算法可以在运行时更改。

## 补充介绍：

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。

*Strategy的意思是“策略”，指的是与敌军对垒时行军作战的方法。*

在策略模式中，我们**创建各种表示策略的对象**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。

定义一系列的算法,把策略对象一个个封装起来, 并且使它们可相互替换。

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。

## 参与角色:

1）策略类的抽象基类（也可以是接口）拥有不同策略类的共有方法

2）各种策略实现类

3）Context类（持有策略基类的对象，可以动态随着策略对象的改变而改变行为）

## 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。

## 使用案例或场景：

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。

3）JAVA AWT 中的 LayoutManager。

## 示例程序

需要源码的朋友可以前往github下载：

<https://github.com/aharddreamer/chendong/tree/master/design-patterns/demo-code/design-patterns>

**程序简介：**

下面这个程序的功能是让电脑玩猜拳游戏。我们考虑了两种猜拳策略，第一种策略是“如果这局获胜，那么下局也出一样的手势” （VinningStrategy），这是一种稍微有些笨的策略；另一种策略是“根据上一局的手势从概率上计算出下一局的手势” （ProbStrategy）。

**示例程序类/接口一览：**

Hand 表示猜拳游戏中的“手势”的类（工具）

Strategy 表示猜拳游戏中的策略的类 （策略类的基类）

WinningStrategy 表示“如果这局获胜，那么下局也出一样的手势” 这一策略的类 （具体策略对象1）

ProbStrategy 表示“根据上一局的手势从概率上计算出下一局的手势” 这一策略的类 （具体策略对象2）

Player 表示进行猜拳游戏的类 （Context环境类）

StrategyPatternTest 测试程序的类

**代码：**

**public class** Hand {  
 **public static final int *SHI\_TOU*** = 0; *//表示石头的值* **public static final int *JIAN\_DAO*** = 1; *//表示石头的值* **public static final int *BU*** = 2; *//表示石头的值* **public static final** Hand[] ***hand*** = {  
 **new** Hand(***SHI\_TOU***), **new** Hand(***JIAN\_DAO***), **new** Hand(***BU***)  
 };  
 **public static final** String[] ***name*** = {  
 **"石头"**, **"剪刀"**, **"布"** };  
 **private int handValue**;  
 **private** Hand (**int** handValue) {  
 **this**.**handValue** = handValue;  
 }  
 **public boolean** isStrongerThan(Hand hand) {  
 **return** fight(hand) == -1;  
 }  
 **public int** fight(Hand hand) {  
 **if** (**this** == hand) {  
 **return** 0;  
 }**else if** ((**this**.**handValue** + 1) % 3 == hand.**handValue**) {  
 **return** 1;  
 }**else** {  
 **return** -1;  
 }  
 }  
 **public** String toString() {  
 **return *name***[**this**.**handValue**];  
 }  
 **public static** Hand getHand(**int** handValue) {  
 **return *hand***[handValue];  
 }  
 **public boolean** isWeakerThan(Hand hand) {  
 **return** fight(hand) == -1;  
 }  
}

**public interface** Strategy {  
 Hand nextHand();  
 **void** study(**boolean** vin);  
}

**public class** WinningStrategy **implements** Strategy {  
 **private** Random **random**;  
 **private boolean won** = **false**;  
 **private** Hand **preHand**;  
 **public** WinningStrategy(**int** seed) {  
 **random** = **new** Random(seed);  
 }  
 **public** Hand nextHand() {  
 **if** (!**won**) {  
 **preHand** = Hand.*getHand*(**random**.nextInt(3));  
 }  
 **return preHand**;  
 }  
 **public void** study(**boolean** win) {  
 **won** = win;  
 }  
}

**public class** Player {  
 **private** String **name**;  
 **private** Strategy **strategy**;  
 **private int wincount**;  
 **private int losecount**;  
 **private int gamecount**;  
 *//赋予姓名和策略* **public** Player(String name, Strategy strategy) {  
 **this**.**name** = name;  
 **this**.**strategy** = strategy;  
 }  
  
 *//策略决定下一句要出的手势* **public** Hand nextHand() {  
 **return strategy**.nextHand();  
 }  
 **public void** win() {  
 **strategy**.study(**true**);  
 **wincount**++;  
 **gamecount**++;  
 }  
 **public void** lose() {  
 **strategy**.study(**false**);  
 **losecount**++;  
 **gamecount**++;  
 }  
 **public void** even() {  
 **gamecount**++;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Player{"** +  
 **"name='"** + **name** + **'\''** +  
 **", strategy="** + **strategy** +  
 **", wincount="** + **wincount** +  
 **", losecount="** + **losecount** +  
 **", gamecount="** + **gamecount** +  
 **'}'**;  
 }  
}

**public class** ProbStrategy **implements** Strategy {  
 **private** Random **random**;  
 **private int preHandValue** = 0;  
 **private int currentHandValue** = 0;  
 **private int**[][] **history** = {  
 {1, 1, 1, },  
 {1, 1, 1, },  
 {1, 1, 1, },  
 };  
 **public** ProbStrategy(**int** seed) {  
 **random** = **new** Random(seed);  
 }  
 **public** Hand nextHand() {  
 **int** bet = **random**.nextInt(getSum(**currentHandValue**));  
 **int** handValue = 0;  
 **if** (bet < **history**[**currentHandValue**][0]) {  
 handValue = 0;  
 }**else if** (bet < **history**[**currentHandValue**][0] + **history**[**currentHandValue**][1]) {  
 handValue = 1;  
 }**else** {  
 handValue = 2;  
 }  
 **preHandValue** = **currentHandValue**;  
 **currentHandValue** = handValue;  
 **return** Hand.*getHand*(handValue);  
 }  
 **private int** getSum(**int** hv) {  
 **int** sum = 0;  
 **for** (**int** i = 0; i < 3 ; i++) {  
 sum += **history**[hv][i];  
 }  
 **return** sum;  
 }  
 **public void** study(**boolean** win) {  
 **if** (win) {  
 **history**[**preHandValue**][**currentHandValue**] ++;  
 }**else** {  
 **history**[**preHandValue**][(**currentHandValue** + 1) % 3] ++;  
 **history**[**preHandValue**][(**currentHandValue** + 2) % 3] ++;  
 }  
 }  
}

**public class** StrategyPatternTest {  
 **public static void** main(String[] args) {  
 *//可以选择动态确定seed  
 /\*if (args.length != 2) {  
 System.out.println("Usage: java Main randomseed1 randomseed2");  
 System.out.println("Example: java Main 314 15");  
 System.exit(0);  
 }  
 int seed1 = Integer.parseInt(args[0]);  
 int seed2 = Integer.parseInt(args[1]);\*/* **int** seed1 = 314;  
 **int** seed2 = 15;  
 Player player1 = **new** Player(**"WinningStrategyPlayer"**, **new** WinningStrategy(seed1));  
 Player player2 = **new** Player(**"ProbStrategyPlayer"**, **new** ProbStrategy(seed2));  
 **for** (**int** i = 0; i < 1000 ; i++) {  
 Hand nextHand1 = player1.nextHand();  
 Hand nextHand2 = player2.nextHand();  
 **if** (nextHand1.isStrongerThan(nextHand2)) {  
 System.***out***.println(**"Winner: "** + player1);  
 player1.win();  
 player2.lose();  
 }**else if** (nextHand2.isStrongerThan(nextHand1)) {  
 System.***out***.println(**"Winner: "** + nextHand2);  
 player1.lose();  
 player2.win();  
 }**else** {  
 System.***out***.println(**"Even..."**);  
 player1.even();  
 player2.even();  
 }  
 }  
 System.***out***.println(**"Total Result: "**);  
 System.***out***.println(player1.toString());  
 System.***out***.println(player2.toString());  
 }  
}

**运行结果：**

Winner: Player{name='WinningStrategyPlayer', strategy=org.cd.designpatterns.strategy.WinningStrategy@2ff4acd0, wincount=310, losecount=330, gamecount=993}

Winner: 石头

Winner: Player{name='WinningStrategyPlayer', strategy=org.cd.designpatterns.strategy.WinningStrategy@2ff4acd0, wincount=311, losecount=331, gamecount=995}

Even...

Winner: 石头

Even...

Even...

-------（省略若干行）---------

Total Result:

Player{name='WinningStrategyPlayer', strategy=org.cd.designpatterns.strategy.WinningStrategy@2ff4acd0, wincount=312, losecount=332, gamecount=1000}

Player{name='ProbStrategyPlayer', strategy=org.cd.designpatterns.strategy.ProbStrategy@54bedef2, wincount=332, losecount=312, gamecount=1000}

参考：

《图解设计模式》【日】结城浩著

《装饰器模式》菜鸟教程网站