


# 设计模式 状态模式 以自动售货机为例

原创 鸿洋\_ 于 2014-05-20 12:36:58 发布 26717 收藏 17 版权

分类专栏: [【Java 设计模式】](#) [设计模式融入生活](#) 文章标签: [设计模式](#) [状态模式](#)

 **【Java 设计模式】** 同时初

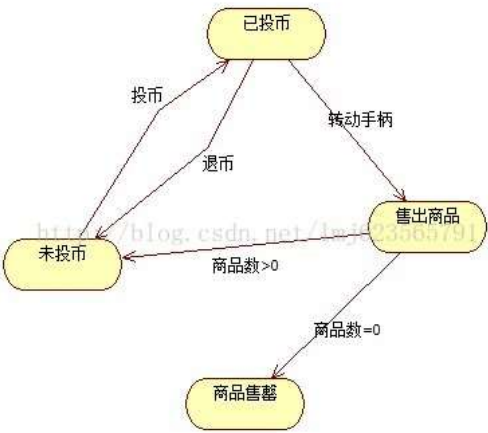
67 订阅 10 篇文章 [订阅专栏](#)

转载请标明出处: <http://blog.csdn.net/Imj623565791/article/details/26350617>

**状态模式** 给了我眼前一亮的感觉啊, 值得学习~

先看定义: 允许对象在内部状态改变时改变它的行为, 对象看起来好像修改了它的类。定义又开始模糊了, 理一下, 当对象的内部状态改变时, 它的行为跟随状态的变化而改变了, 看起来好像重新初始化了一个类似的。

下面使用个例子来说明状态模式的用法, 现在有个自动售货机的代码需要我们来写, 状态图如下:



分析一个这个状态图:

- a、包含4个状态 (我们使用4个int型常量来表示)
- b、包含3个暴露在外的方法 (投币、退币、转动手柄)
- c、我们需要处理每个状态下, 用户都可以触发这三个动作。

下面我们根据分析的结果, 写出代码:

```
1 package com.zhy.pattern.status;
2
3 /**
4  * 自动售货机
5  *
6  * @author zhy
7  *
8  */
9 public class VendingMachine
10 {
11
12     /**
13      * 已投币
14      */
15     private final static int HAS_MONEY = 0;
16     /**
17      * 未投币
18      */
19     private final static int NO_MONEY = 1;
20     /**
21      * 售出商品
22      */
```

```
23     private final static int SOLD = 2;
24     /**
25      * 商品售罄
26      */
27     private final static int SOLD_OUT = 3;
28
29     private int currentStatus = NO_MONEY;
30     /**
31      * 商品数量
32      */
33     private int count = 0;
34
35     public VendingMachine(int count)
36     {
37         this.count = count;
38         if (count > 0)
39         {
40             currentStatus = NO_MONEY;
41         }
42     }
43
44     /**
45      * 投入硬币, 任何状态用户都可能投币
46      */
47     public void insertMoney()
48     {
49         switch (currentStatus)
50         {
51             case NO_MONEY:
52                 currentStatus = HAS_MONEY;
53                 System.out.println("成功投入硬币");
54                 break;
55             case HAS_MONEY:
56                 System.out.println("已经有硬币, 无需投币");
57                 break;
58             case SOLD:
59                 System.out.println("请稍等...");
60                 break;
61             case SOLD_OUT:
62                 System.out.println("商品已经售罄, 请勿投币");
63                 break;
64         }
65     }
66 }
67
68 /**
69  * 退币, 任何状态用户都可能退币
70  */
71 public void backMoney()
72 {
73     switch (currentStatus)
74     {
75         case NO_MONEY:
76             System.out.println("您未投入硬币");
77             break;
78         case HAS_MONEY:
79             currentStatus = NO_MONEY;
80             System.out.println("退币成功");
81             break;
82         case SOLD:
83             System.out.println("您已经买了糖果...");
84             break;
85         case SOLD_OUT:
86             System.out.println("您未投币...");
87             break;
88     }
89 }
90
91 /**
92  * 转动手柄购买, 任何状态用户都可能转动手柄
93  */
```

```
94 | public void turnCrank()
    | 95 | {
96 |     switch (currentStatus)
97 |     {
98 |     case NO_MONEY:
99 |         System.out.println("请先投入硬币");
100 |         break;
101 |     case HAS_MONEY:
102 |         System.out.println("正在出商品...");
103 |         currentStatus = SOLD;
104 |         dispense();
105 |         break;
106 |     case SOLD:
107 |         System.out.println("连续转动也没用...");
108 |         break;
109 |     case SOLD_OUT:
110 |         System.out.println("商品已经售罄");
111 |         break;
112 |     }
113 | }
114 | }
115 |
116 | /**
117 |  * 发放商品
118 |  */
119 | private void dispense()
120 | {
121 |
122 |     switch (currentStatus)
123 |     {
124 |     case NO_MONEY:
125 |     case HAS_MONEY:
126 |     case SOLD_OUT:
127 |         throw new IllegalStateException("非法的状态...");
128 |     case SOLD:
129 |         count--;
130 |         System.out.println("发出商品...");
131 |         if (count == 0)
132 |         {
133 |             System.out.println("商品售罄");
134 |             currentStatus = SOLD_OUT;
135 |         } else
136 |         {
137 |             currentStatus = NO_MONEY;
138 |         }
139 |         break;
140 |
141 |     }
142 |
143 | }
144 | }
```

针对用户的每个动作，我们考虑了在任何状态下发生，并做了一定处理。下面进行一些测试：

```
1 | package com.zhy.pattern.status;
2 |
3 | public class TestTra
4 | {
5 |     public static void main(String[] args)
6 |     {
7 |         VendingMachine machine = new VendingMachine(10);
8 |         machine.insertMoney();
9 |         machine.backMoney();
10 |
11 |         System.out.println("-----");
12 |
13 |         machine.insertMoney();
```

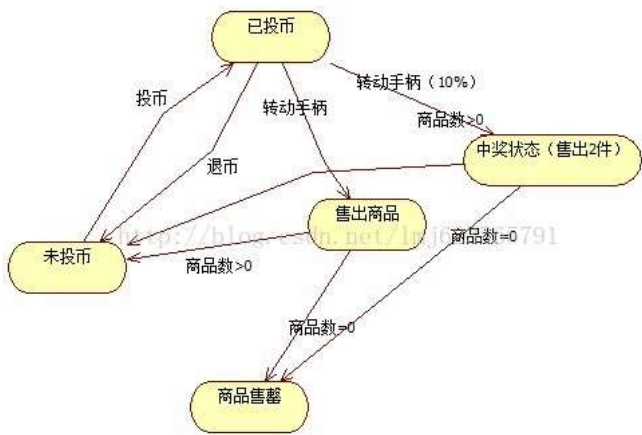
```
14 |         machine.turnCrank();
    |                                     15 |
16 |         System.out.println("-----压力测试-----");
17 |         machine.insertMoney();
18 |         machine.insertMoney();
19 |         machine.turnCrank();
20 |         machine.turnCrank();
21 |         machine.backMoney();
22 |         machine.turnCrank();
23 |
24 |     }
25 | }
```

输出结果：

```
1 | 成功投入硬币
2 | 退币成功
3 | -----
4 | 成功投入硬币
5 | 正在出商品....
6 | 发出商品...
7 | -----压力测试-----
8 | 成功投入硬币
9 | 已经有硬币，无需投币
10 | 正在出商品....
11 | 发出商品...
12 | 请先投入硬币
13 | 您未投入硬币
14 | 请先投入硬币
```

感觉还是不错的，基本实现了功能，但是有些事情是不可避免的，那就是需求的变化，现在为了提升销量，当用户每次转动手柄买商品的时候，有10%的几率赠送一瓶。

现在的状态图发生了变化，当用户转动手柄时，可能会达到一个中奖的状态：图如下：



如果在我们刚写的代码上直接添加，则需要每个动作的switch中添加判断条件，且非常容易出错。所以现在我们要考虑重新设计我们的代码，我们考虑把每个状态写状态类，负责实现在对应动作下的行为，然后自动售货机在不同的状态间切换：

下面开始重构，我们现在有5种状态，对应4个动作（投币、退币、转动手柄、发出商品），下面首先定义一个状态的超类型：

```
1 | package com.zhy.pattern.status.b;
2 |
3 | /**
4 |  * 状态的接口
5 |  * @author zhy
6 |  *
7 |  */
```

```
8 | public interface State
   | 9 | {
10 |
11 |     /**
12 |      * 放钱
13 |      */
14 |     public void insertMoney();
15 |     /**
16 |      * 退钱
17 |      */
18 |     public void backMoney();
19 |     /**
20 |      * 转动曲柄
21 |      */
22 |     public void turnCrank();
23 |     /**
24 |      * 出商品
25 |      */
26 |     public void dispense();
   | }
```

然后分别是每个状态的实现：

```
1 | package com.zhy.pattern.status.b;
2 |
3 | /**
4 |  * 没钱的状态
5 |  * @author zhy
6 |  *
7 |  */
8 | public class NoMoneyState implements State
9 | {
10 |
11 |     private VendingMachine machine;
12 |
13 |     public NoMoneyState(VendingMachine machine)
14 |     {
15 |         this.machine = machine;
16 |     }
17 |
18 |
19 |     @Override
20 |     public void insertMoney()
21 |     {
22 |         System.out.println("投币成功");
23 |         machine.setState(machine.getHasMoneyState());
24 |     }
25 |
26 |     @Override
27 |     public void backMoney()
28 |     {
29 |         System.out.println("您未投币，想退钱? ...");
30 |     }
31 |
32 |     @Override
33 |     public void turnCrank()
34 |     {
35 |         System.out.println("您未投币，想拿东西么? ...");
36 |     }
37 |
38 |     @Override
39 |     public void dispense()
40 |     {
41 |         throw new IllegalStateException("非法状态!");
42 |     }
43 |
44 | }
```

```
1 package com.zhy.pattern.status.b;
2
3 import java.util.Random;
4
5 /**
6  * 已投入钱的状态
7  *
8  * @author zhy
9  *
10 */
11 public class HasMoneyState implements State
12 {
13
14     private VendingMachine machine;
15     private Random random = new Random();
16
17     public HasMoneyState(VendingMachine machine)
18     {
19         this.machine = machine;
20     }
21
22     @Override
23     public void insertMoney()
24     {
25         System.out.println("您已经投过币了，无需再投...");
26     }
27
28     @Override
29     public void backMoney()
30     {
31         System.out.println("退币成功");
32
33         machine.setState(machine.getNoMoneyState());
34     }
35
36     @Override
37     public void turnCrank()
38     {
39         System.out.println("你转动了手柄");
40         int winner = random.nextInt(10);
41         if (winner == 0 && machine.getCount() > 1)
42         {
43             machine.setState(machine.getWinnerState());
44         } else
45         {
46             machine.setState(machine.getSoldState());
47         }
48     }
49
50     @Override
51     public void dispense()
52     {
53         throw new IllegalStateException("非法状态!");
54     }
55
56 }
```

```
1 package com.zhy.pattern.status.b;
2
3 /**
4  * 售罄的状态
5  *
6  * @author zhy
7  *
8  */
9 public class SoldOutState implements State
```

```
10 | {11 |
11 |
12 |     private VendingMachine machine;
13 |
14 |     public SoldOutState(VendingMachine machine)
15 |     {
16 |         this.machine = machine;
17 |     }
18 |
19 |     @Override
20 |     public void insertMoney()
21 |     {
22 |         System.out.println("投币失败, 商品已售罄");
23 |     }
24 |
25 |     @Override
26 |     public void backMoney()
27 |     {
28 |         System.out.println("您未投币, 想退钱么? ...");
29 |     }
30 |
31 |     @Override
32 |     public void turnCrank()
33 |     {
34 |         System.out.println("商品售罄, 转动手柄也木有用");
35 |     }
36 |
37 |     @Override
38 |     public void dispense()
39 |     {
40 |         throw new IllegalStateException("非法状态!");
41 |     }
42 |
43 | }
```

```
1 | package com.zhy.pattern.status.b;
2 |
3 | /**
4 |  * 准备出商品的状态, 该状态下, 不会有任何用户的操作
5 |  *
6 |  * @author zhy
7 |  *
8 |  */
9 | public class SoldState implements State
10 | {
11 |
12 |     private VendingMachine machine;
13 |
14 |     public SoldState(VendingMachine machine)
15 |     {
16 |         this.machine = machine;
17 |     }
18 |
19 |     @Override
20 |     public void insertMoney()
21 |     {
22 |         System.out.println("正在出货, 请勿投币");
23 |     }
24 |
25 |     @Override
26 |     public void backMoney()
27 |     {
28 |         System.out.println("正在出货, 没有可退的钱");
29 |     }
30 |
31 |     @Override
32 |     public void turnCrank()
33 |     {
34 |         System.out.println("正在出货, 请勿重复转动手柄");
35 |     }
36 | }
```

```
35 |     }
36 |
37 |     @Override
38 |     public void dispense()
39 |     {
40 |         machine.dispense();
41 |         if (machine.getCount() > 0)
42 |         {
43 |             machine.setState(machine.getNoMoneyState());
44 |         } else
45 |         {
46 |             System.out.println("商品已经售罄");
47 |             machine.setState(machine.getSoldOutState());
48 |         }
49 |     }
50 | }
```

```
1 | package com.zhy.pattern.status.b;
2 |
3 | /**
4 |  * 中奖的状态，该状态下不会有任何用户的操作
5 |  *
6 |  * @author zhy
7 |  *
8 |  */
9 | public class WinnerState implements State
10 | {
11 |
12 |     private VendingMachine machine;
13 |
14 |     public WinnerState(VendingMachine machine)
15 |     {
16 |         this.machine = machine;
17 |     }
18 |
19 |     @Override
20 |     public void insertMoney()
21 |     {
22 |         throw new IllegalStateException("非法状态");
23 |     }
24 |
25 |     @Override
26 |     public void backMoney()
27 |     {
28 |         throw new IllegalStateException("非法状态");
29 |     }
30 |
31 |     @Override
32 |     public void turnCrank()
33 |     {
34 |         throw new IllegalStateException("非法状态");
35 |     }
36 |
37 |     @Override
38 |     public void dispense()
39 |     {
40 |         System.out.println("你中奖了，恭喜你，将得到2件商品");
41 |         machine.dispense();
42 |
43 |         if (machine.getCount() == 0)
44 |         {
45 |             System.out.println("商品已经售罄");
46 |             machine.setState(machine.getSoldOutState());
47 |         } else
48 |         {
49 |             machine.dispense();
50 |             if (machine.getCount() > 0)
51 |             {
```



```

52 |         machine.setState(machine.getNoMoneyState());53 |
    |     } else54 |         {
55 |         System.out.println("商品已经售罄");
56 |         machine.setState(machine.getSoldOutState());
57 |     }
58 |
59 | }
60 |
61 | }
62 |
63 | }

```

最后是自动售货机的代码:

```

1 | package com.zhy.pattern.status.b;
2 |
3 | /**
4 |  * 自动售货机
5 |  *
6 |  * @author zhy
7 |  *
8 |  */
9 | public class VendingMachine
10 | {
11 |     private State noMoneyState;
12 |     private State hasMoneyState;
13 |     private State soldState;
14 |     private State soldOutState;
15 |     private State winnerState ;
16 |
17 |     private int count = 0;
18 |     private State currentState = noMoneyState;
19 |
20 |     public VendingMachine(int count)
21 |     {
22 |         noMoneyState = new NoMoneyState(this);
23 |         hasMoneyState = new HasMoneyState(this);
24 |         soldState = new SoldState(this);
25 |         soldOutState = new SoldOutState(this);
26 |         winnerState = new WinnerState(this);
27 |
28 |         if (count > 0)
29 |         {
30 |             this.count = count;
31 |             currentState = noMoneyState;
32 |         }
33 |     }
34 |
35 |     public void insertMoney()
36 |     {
37 |         currentState.insertMoney();
38 |     }
39 |
40 |     public void backMoney()
41 |     {
42 |         currentState.backMoney();
43 |     }
44 |
45 |     public void turnCrank()
46 |     {
47 |         currentState.turnCrank();
48 |
49 |         if (currentState == soldState || currentState == winnerSta
50 |             currentState.dispense();50 |     }
51 |
52 |     public void dispense()

```

```
53 | {
54 |     System.out.println("发出一件商品...");
55 |     if (count != 0)
56 |     {
57 |         count -= 1;
58 |     }
59 | }
60 |
61 | public void setState(State state)
62 | {
63 |     this.currentState = state;
64 | }
65 |
66 | //getter setter omitted ...
67 |
68 | }
```

可以看到，我们现在把每个状态对应于动作的行为局部化到了状态自己的类中实现，不仅增加了扩展性而且使代码的阅读性大幅度的提高。以后再添加状态，只需要针对新添加的状态的实现类，并在自动售货机中添加此状态即可。

下面进行一些测试：

```
1 | package com.zhy.pattern.status.b;
2 |
3 | public class Test
4 | {
5 |     public static void main(String[] args)
6 |     {
7 |         VendingMachine machine = new VendingMachine(10);
8 |         machine.insertMoney();
9 |         machine.backMoney();
10 |
11 |         System.out.println("----我要中奖----");
12 |
13 |         machine.insertMoney();
14 |         machine.turnCrank();
15 |         machine.insertMoney();
16 |         machine.turnCrank();
17 |         machine.insertMoney();
18 |         machine.turnCrank();
19 |         machine.insertMoney();
20 |         machine.turnCrank();
21 |         machine.insertMoney();
22 |         machine.turnCrank();
23 |         machine.insertMoney();
24 |         machine.turnCrank();
25 |         machine.insertMoney();
26 |         machine.turnCrank();
27 |
28 |         System.out.println("-----压力测试-----");
29 |
30 |         machine.insertMoney();
31 |         machine.backMoney();
32 |         machine.backMoney();
33 |         machine.turnCrank();// 无效操作
34 |         machine.turnCrank();// 无效操作
35 |         machine.backMoney();
36 |
37 |     }
38 | }
```

输出结果：

2 | 退币成功

3 | ----我要中奖----

4 | 投币成功

5 | 你转动了手柄

6 | 发出一件商品...

7 | 投币成功

8 | 你转动了手柄

9 | 发出一件商品...

10 | 投币成功

11 | 你转动了手柄

12 | 发出一件商品...

13 | 投币成功

14 | 你转动了手柄

15 | 发出一件商品...

16 | 投币成功

17 | 你转动了手柄

18 | 发出一件商品...

19 | 投币成功

20 | 你转动了手柄

21 | 发出一件商品...

22 | 投币成功

23 | 你转动了手柄

24 | 你中奖了，恭喜你，将得到2件商品

25 | 发出一件商品...

26 | 发出一件商品...

27 | -----压力测试-----

28 | 投币成功

29 | 退币成功

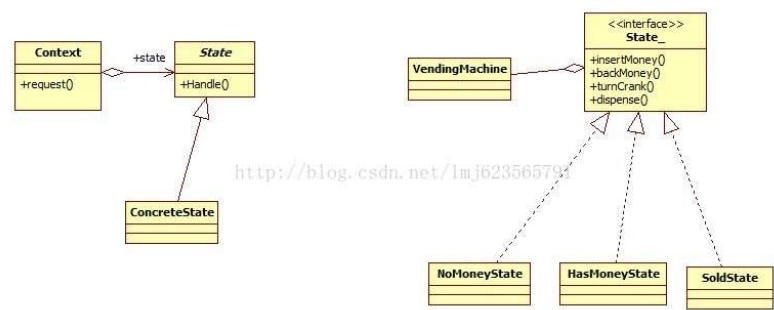
30 | 您未投币，想退钱？...

31 | 您未投币，想拿东西么？...

32 | 您未投币，想拿东西么？...

33 | 您未投币，想退钱？...

恭喜你，又学会了一个设计模式，状态模式。最后看下状态模式的类图：



最后，欢迎大家留言~

显示推荐内容

觉得还不错？[一键收藏](#)

鸿洋\_

关注

81

17

29

专栏目录