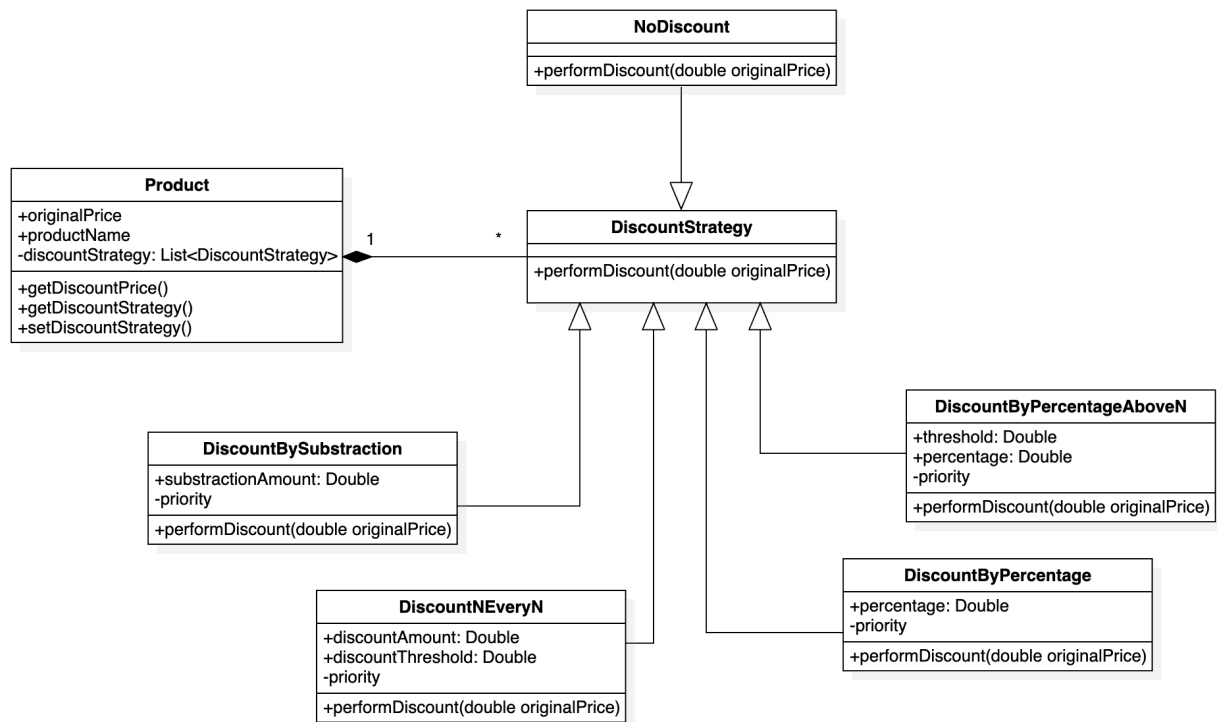


# OOAD Assignment 8

10185101210 陈俊潼

## 策略模式设计方案

商场打折的策略模式设计方案类图如下：

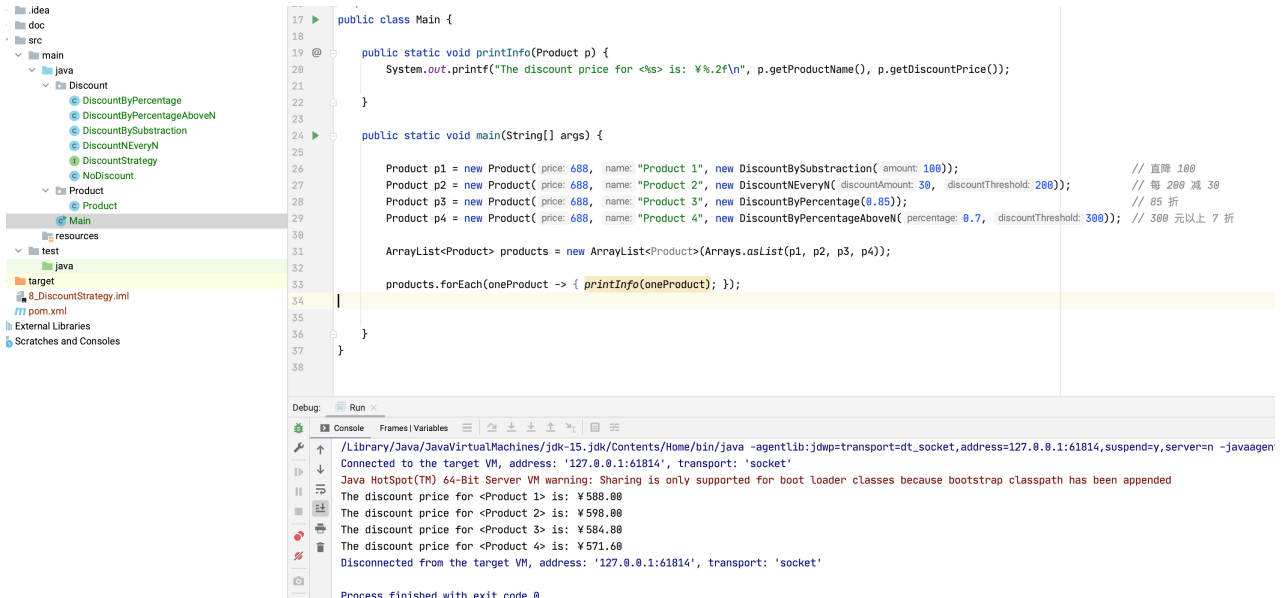


这里给出了 4 种折扣方案：

- **DiscountBySubstraction**：直接在价格上减去若干元
- **DiscountNEveryN**：每满若干元减去若干元
- **DiscountByPercentage**：在价格上打若干折
- **DiscountByPercentageAboveN**：超过若干元的部分打若干折

以及一个 **NoDiscount** 策略用于表示没有任何折扣。

例如，针对 4 个价格均为 688 元的商品，采用了 4 种折扣策略，运行结果如下图所示：



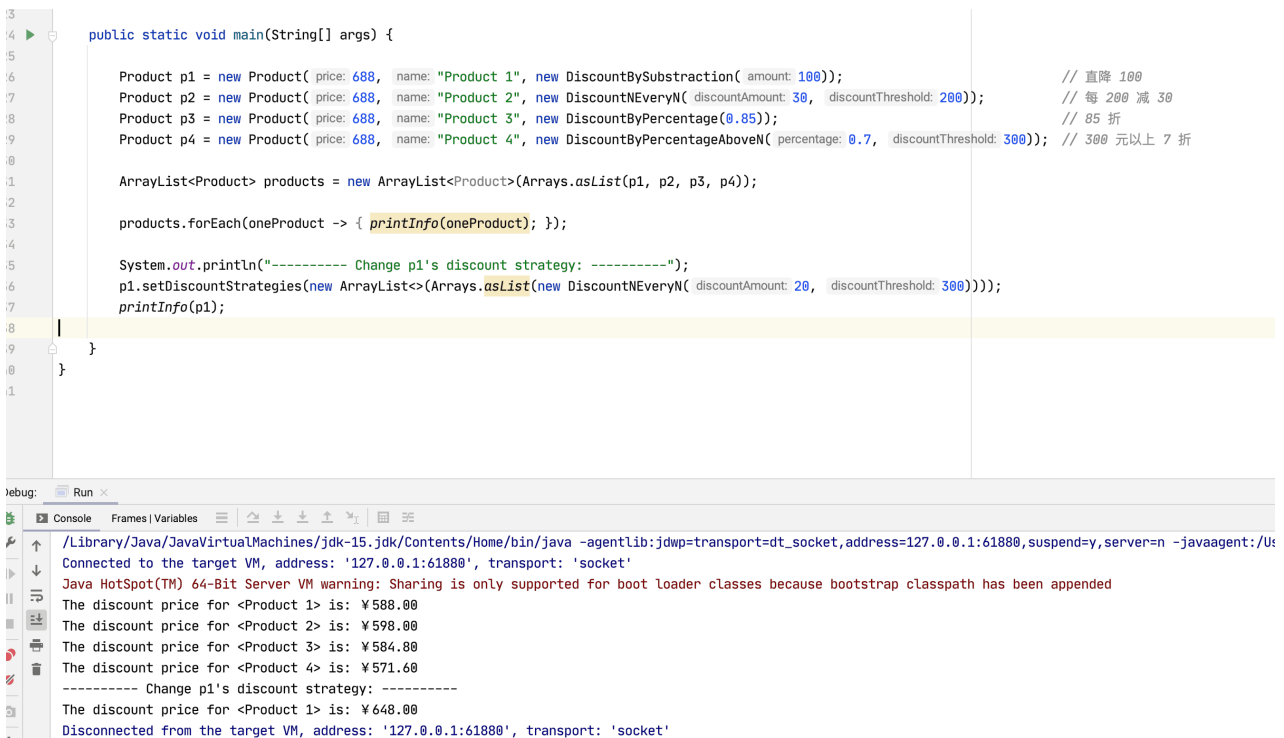
源代码见另一个压缩包内。

## 其他问题

### 1. 折扣方式改掉怎么办？

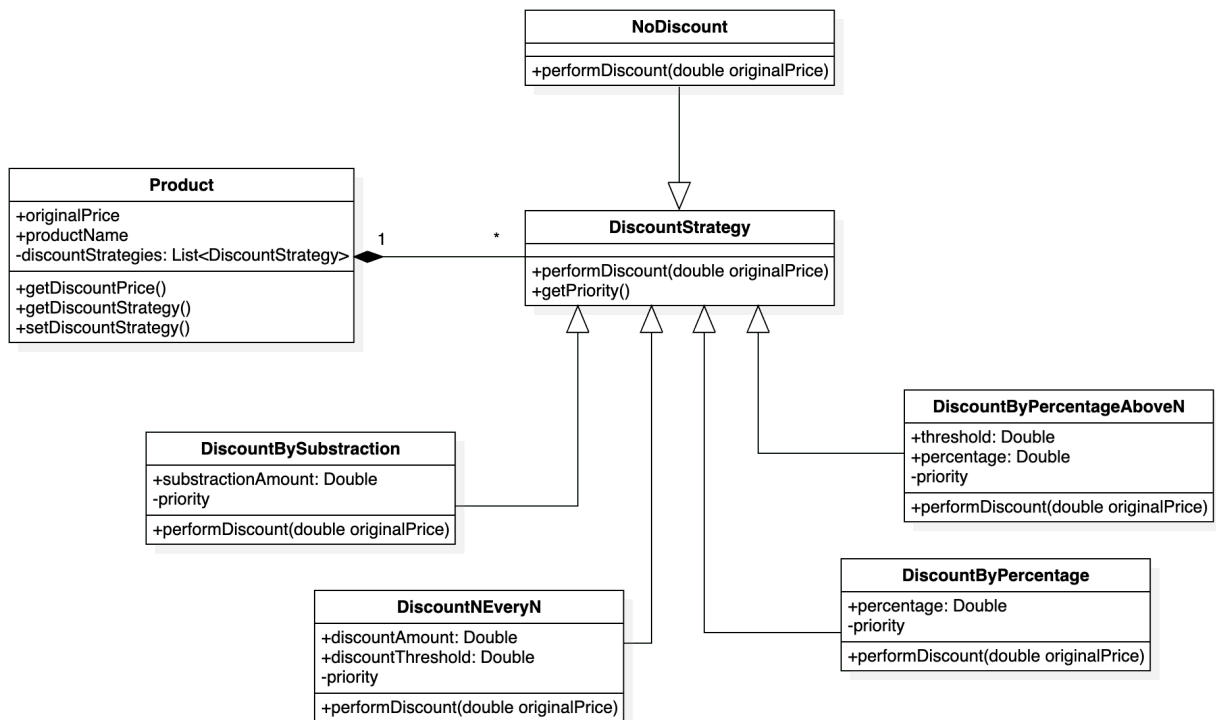
使用 `Product` 的 `setDiscountStrategy()` 方法即可。如：

```
1 p1.setDiscountStrategies(new ArrayList<>(Arrays.asList(new DiscountNEveryN(20, 300))));
```



## 2. 多种折扣方式同时使用怎么办？

首先将 `Product` 类中的 `discountStrategy` 从一个对象属性变成一个列表封装的属性，同时将属性名从 `discountStrategy` 改为 `discountStrategies`，如下图所示：



然后在各个折扣策略中指定优先级，数字小优先级越高。如果在 `Product` 的折扣策略列表中有多个折扣策略，则最后的价格会按照各个策略的优先级从低至高进行计算。例如，指定一个同时具有多个策略的产品，反复使用 `addDiscountStrategy` 来添加折扣策略，计算结果如下图所示：

```
java
└─ Discount
   ├─ DiscountByPercentage
   ├─ DiscountByPercentageAboveN
   ├─ DiscountBySubstraction
   ├─ DiscountNEveryN
   └─ NoDiscount
Product
└─ Product
Main
resources
test
java
rget
DiscountStrategyImpl
xml
nal Libraries
ches and Consoles

34 //
35 //
36 //
37 //
38 //
39 //
40 //
41 //
42 //
43 //
44 //
45 //
46 //
47 //
48 //
49 //
50 //

System.out.println("----- Change p1's discount strategy: -----");
p1.clearDiscountStrategy();
p1.addDiscountStrategy(new DiscountNEveryN(20, 300)); // 给 p1 增加一个每 300 减 20 的优惠
printInfo(p1);

System.out.println("----- A product with four strategies: -----");
Product p5 = new Product(price: 600, name: "Product 5");
p5.addDiscountStrategy(new DiscountBySubstraction(amount: 10));
p5.addDiscountStrategy(new DiscountNEveryN(discountAmount: 20, discountThreshold: 200));
p5.addDiscountStrategy(new DiscountByPercentageAboveN(percentage: 0.8, discountThreshold: 200));
p5.addDiscountStrategy(new DiscountByPercentage(0.98));
System.out.printf("The discount price for <%s> is: ¥%.2f\n", p5.getProductName(), p5.getDiscountPrice());

}

Debug: Run
Console Frames Variables
/Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:62242,suspend=y,server=n
Connected to the target VM, address: '127.0.0.1:62242', transport: 'socket'
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
----- A product with four strategies: -----
The discount price for <Product 5> is: ¥522.35
Disconnected from the target VM, address: '127.0.0.1:62242', transport: 'socket'

Process finished with exit code 0
```

## 完整代码

Main.java

```
1  import Discount.DiscountByPercentage;
2  import Discount.DiscountByPercentageAboveN;
3  import Discount.DiscountBySubstraction;
4  import Discount.DiscountNEveryN;
5  import Product.Product;
6
7  import java.util.ArrayList;
8  import java.util.Arrays;
9
10 /**
11  * Main Class
12  *
13  * @author billchen
14  * @version 1.0
15  * @create 2020-12-13 21:58
16  */
17 public class Main {
18
19     public static void main(String[] args) {
20
21         System.out.println("----- Four strategies for four products: -----
22         ");
23         Product p1 = new Product(688, "Product 1", new DiscountBySubstraction(100));
24         // 直降 100
25         Product p2 = new Product(688, "Product 2", new DiscountNEveryN(30, 200));
26         // 每 200 减 30
27         Product p3 = new Product(688, "Product 3", new DiscountByPercentage(0.85));
28         // 85 折
29         Product p4 = new Product(688, "Product 4", new
30         DiscountByPercentageAboveN(0.7, 300)); // 300 元以上 7 折
31
32         ArrayList<Product> products = new ArrayList<Product>(Arrays.asList(p1, p2,
33         p3, p4));
34
35         products.forEach(oneProduct -> {
36             System.out.printf("The discount price for <%s> is: ¥%.2f\n",
37             oneProduct.getProductName(), oneProduct.getDiscountPrice());
38         });
39
40         System.out.println("----- Change p1's discount strategy: -----");
```

```

34         p1.clearDiscountStrategy();
35         p1.addDiscountStrategy(new DiscountNEveryN(20, 300)); // 给 p1 增加一个每 300
    减 20 的优惠
36         System.out.printf("The discount price for <%s> is: ¥%.2f\n",
    p1.getProductName(), p1.getDiscountPrice());
37
38         System.out.println("----- A product with four strategies: -----");
39         Product p5 = new Product(688, "Product 5");
40         p5.addDiscountStrategy(new DiscountBySubstraction(10));
41         p5.addDiscountStrategy(new DiscountNEveryN(20, 200));
42         p5.addDiscountStrategy(new DiscountByPercentageAboveN(0.8, 200));
43         p5.addDiscountStrategy(new DiscountByPercentage(0.98));
44         System.out.printf("The discount price for <%s> is: ¥%.2f\n",
    p5.getProductName(), p5.getDiscountPrice());
45
46     }
47 }

```

## Product.java

```

1  package Product;
2
3  import Discount.DiscountStrategy;
4  import Discount.NoDiscount;
5
6  import java.util.ArrayList;
7  import java.util.Arrays;
8  import java.util.Comparator;
9
10 /**
11  * @author billchen
12  * @version 1.0
13  * @create 2020-12-13 22:53
14  */
15 public class Product {
16
17     private double originalPrice;
18     private String productName;
19
20     private ArrayList<DiscountStrategy> discountStrategies;
21

```

```

22     public Product(double price, String name, ArrayList<DiscountStrategy>
strategies) {
23         originalPrice = price;
24         productName = name;
25         discountStrategies = strategies;
26     }
27
28     public Product(double price, String name, DiscountStrategy strategy) {
29         originalPrice = price;
30         productName = name;
31         discountStrategies = new ArrayList<>() {{
32             add(strategy);
33         }};
34     }
35
36     public Product(double price, String name) {
37         this(price, name, new ArrayList<>());
38     }
39
40     public Product() {
41         this(0, "Unknown Product", new ArrayList<DiscountStrategy>(Arrays.asList(new
NoDiscount())) );
42     }
43
44     public double getOriginalPrice() {
45         return originalPrice;
46     }
47
48     public double getDiscountPrice() {
49         if (discountStrategies == null) {
50             return originalPrice;
51         }
52         discountStrategies.sort(new Comparator<DiscountStrategy>() {
53             @Override
54             public int compare(DiscountStrategy o1, DiscountStrategy o2) {
55                 return o1.getPriority() - o2.getPriority();
56             }
57         });
58         double discountPrice = originalPrice;
59         for (DiscountStrategy strategy : discountStrategies) {
60             discountPrice = strategy.performDiscount(discountPrice);
61         }
62         return discountPrice;
63     }

```

```

64
65     public void setDiscountStrategies(ArrayList<DiscountStrategy>
discountStrategies) {
66         this.discountStrategies = discountStrategies;
67     }
68
69     public void addDiscountStrategy(DiscountStrategy strategy) {
70         this.discountStrategies.add(strategy);
71     }
72
73     public void clearDiscountStrategy() {
74         this.discountStrategies.clear();
75         this.discountStrategies.add(new NoDiscount());
76     }
77
78     public ArrayList<DiscountStrategy> getDiscountStrategies() {
79         return discountStrategies;
80     }
81
82     public String getProductName() {
83         return productName;
84     }
85
86     public void setProductName(String productName) {
87         this.productName = productName;
88     }
89 }

```

## DiscountStrategy.java

```

1  package Discount;
2
3  /**
4   * The interface for discount strategy
5   *
6   * @author billchen
7   * @version 1.0
8   * @create 2020-12-13 22:58
9   */
10 public interface DiscountStrategy {
11     public abstract double performDiscount(double originalPrice);
12     public abstract int getPriority();

```

```
13 }  
14
```

## NoDiscount.java

```
1  package Discount;  
2  
3  /**  
4   * No any discount  
5   *  
6   * @author billchen  
7   * @version 1.0  
8   * @create 2020-12-13 23:08  
9   */  
10 public class NoDiscount implements DiscountStrategy{  
11  
12     @Override  
13     public double performDiscount(double originalPrice) {  
14         return originalPrice;  
15     }  
16  
17     @Override  
18     public int getPriority() {  
19         return 100;  
20     }  
21 }
```

## DiscountByPercentage.java

```
1  package Discount;  
2  
3  /**  
4   * Discount by percentage.  
5   *  
6   * @author billchen  
7   * @version 1.0  
8   * @create 2020-12-13 23:14  
9   */  
10 public class DiscountByPercentage implements DiscountStrategy{  
11  
12     private double percentage;
```



```

13
14     public DiscountByPercentage(double percentage) {
15         this.percentage = percentage;
16     }
17
18     @Override
19     public double performDiscount(double originalPrice) {
20         return originalPrice * percentage;
21     }
22
23     @Override
24     public int getPriority() {
25         return 30;
26     }
27 }
28

```

#### DiscountByPercentageAboveN.java

```

1  package Discount;
2
3  /**
4   * Discount the amount that is over N by a certain percentage.
5   *
6   * @author billchen
7   * @version 1.0
8   * @create 2020-12-13 23:15
9   */
10 public class DiscountByPercentageAboveN implements DiscountStrategy{
11
12     private double percentage;
13     private double discountThreshold;
14
15     public DiscountByPercentageAboveN(double percentage, double discountThreshold) {
16         this.percentage = percentage;
17         this.discountThreshold = discountThreshold;
18     }
19
20     @Override
21     public double performDiscount(double originalPrice) {
22         return originalPrice <= discountThreshold ?
23             originalPrice :

```

```

24         discountThreshold + percentage * (originalPrice -
discountThreshold);
25     }
26
27     @Override
28     public int getPriority() {
29         return 35;
30     }
31 }

```

## DiscountBySubtraction.java

```

1  package Discount;
2
3  /**
4   * Directly subtract certain amount for the original product.
5   *
6   * @author billchen
7   * @version 1.0
8   * @create 2020-12-13 23:09
9   */
10 public class DiscountBySubtraction implements DiscountStrategy{
11
12     private double subtractionAmount;
13
14     public DiscountBySubtraction(double amount) {
15         subtractionAmount = amount;
16     }
17
18     @Override
19     public double performDiscount(double originalPrice) {
20         return Math.max(0, originalPrice - subtractionAmount);
21     }
22
23     @Override
24     public int getPriority() {
25         return 80;
26     }
27
28 }

```

## DiscountNEveryN.java

```
1 package Discount;
2
3 /**
4  * Discount certain amount of money for every N amount of money.
5  *
6  * @author billchen
7  * @version 1.0
8  * @create 2020-12-13 23:12
9  */
10 public class DiscountNEveryN implements DiscountStrategy{
11
12     private double discountAmount;
13     private double discountThreshold;
14
15     public DiscountNEveryN(double discountAmount, double discountThreshold) {
16         this.discountAmount = discountAmount;
17         this.discountThreshold = discountThreshold;
18         if(discountAmount > discountThreshold) {
19             System.out.println("Discount amount larger than discount threshold.
20             Unexpected low price may occur.");
21         }
22     }
23
24     @Override
25     public double performDiscount(double originalPrice) {
26         return originalPrice - ((int) (originalPrice / discountThreshold)) *
27         discountAmount;
28     }
29
30     @Override
31     public int getPriority() {
32         return 10;
33     }
34 }
```