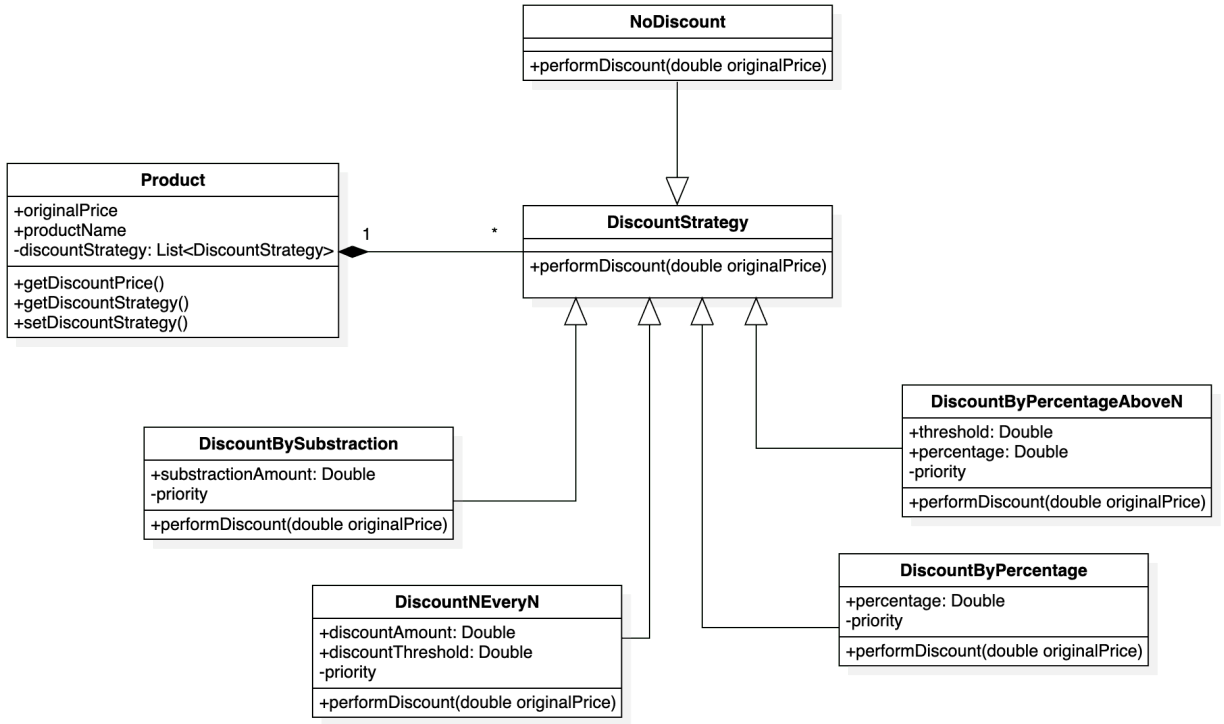
OOAD Assignment 8

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：

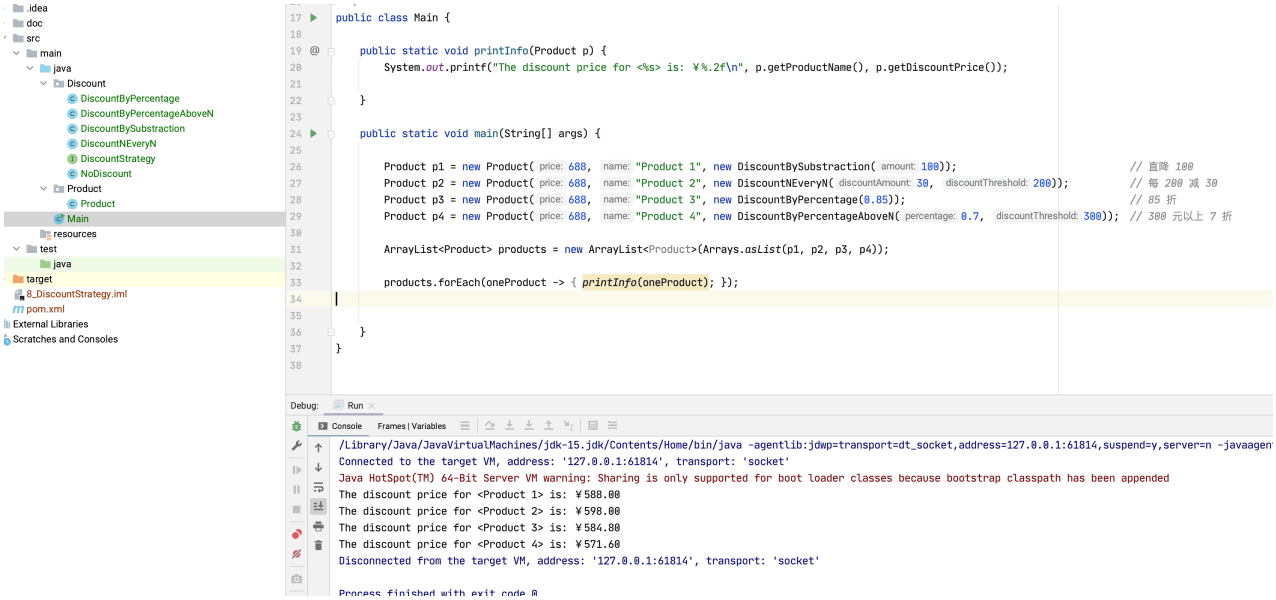


这里给出了 4 种折扣方案：

- `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：



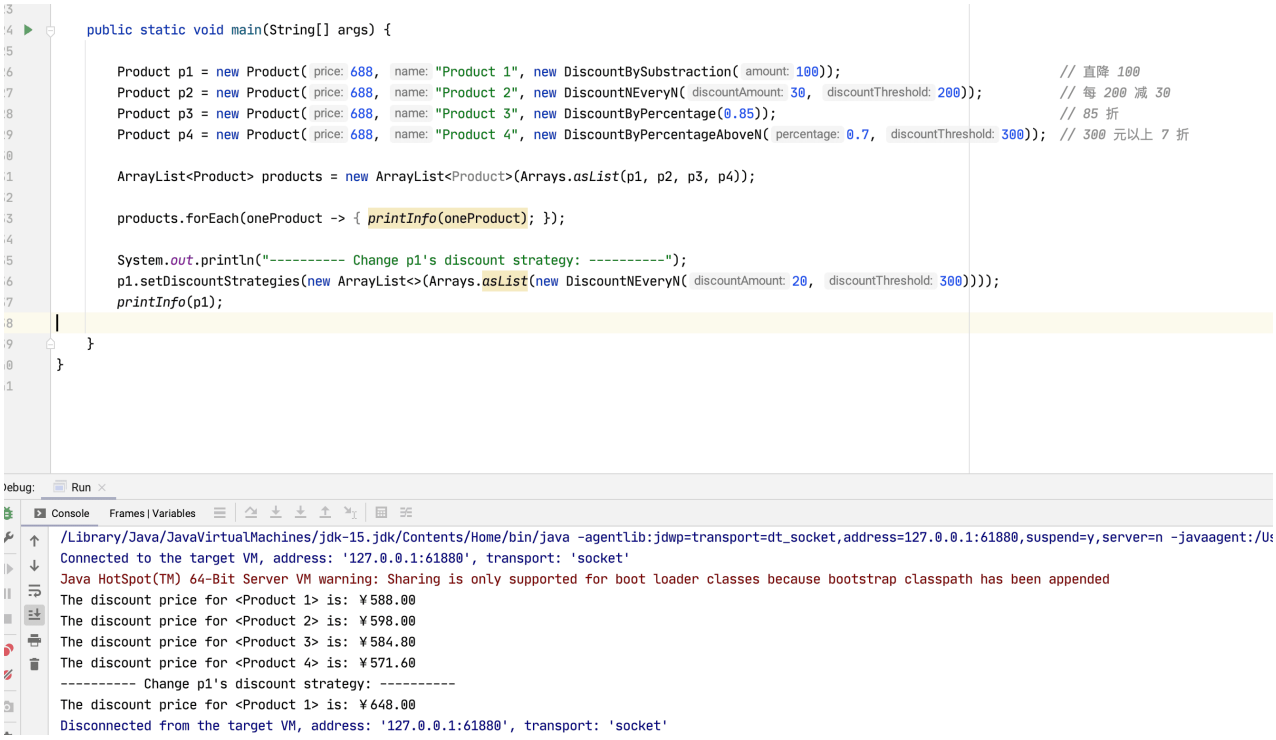
源代码见另一个压缩包内。

其他问题

1. 折扣方式改掉怎么办?

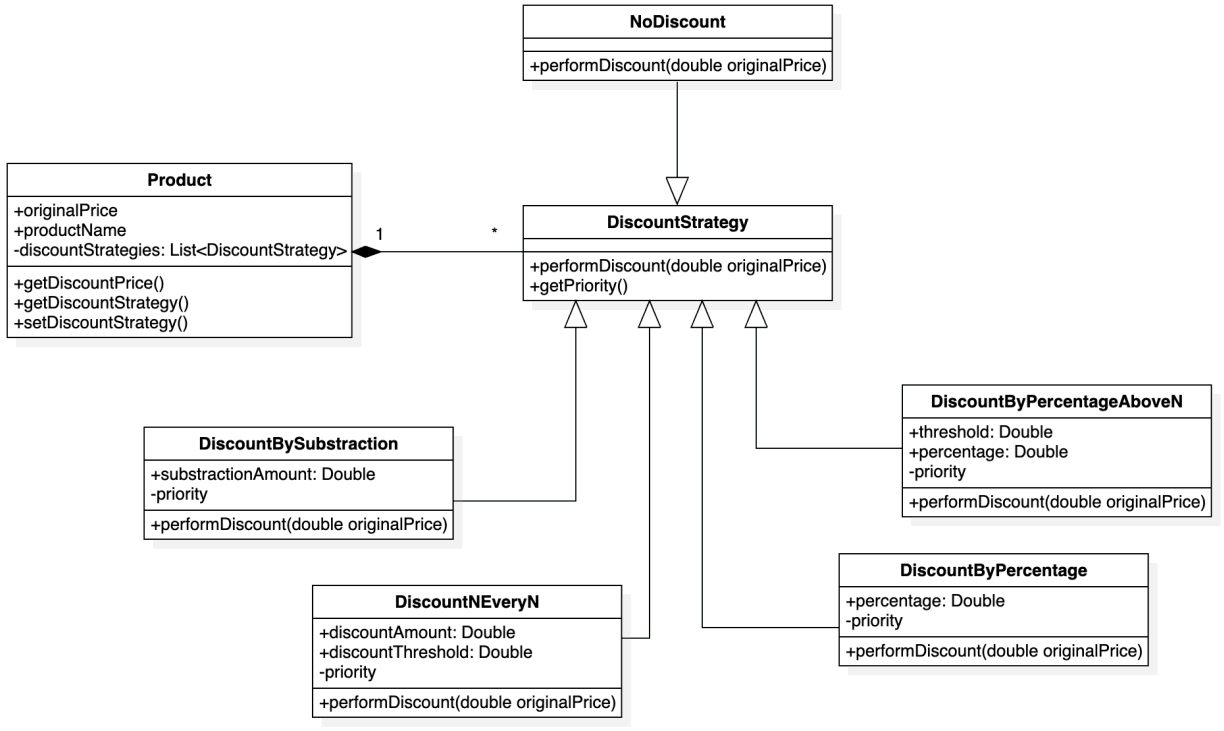
使用 `Product` 的 `setDiscountStrategy()` 方法即可。如:

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

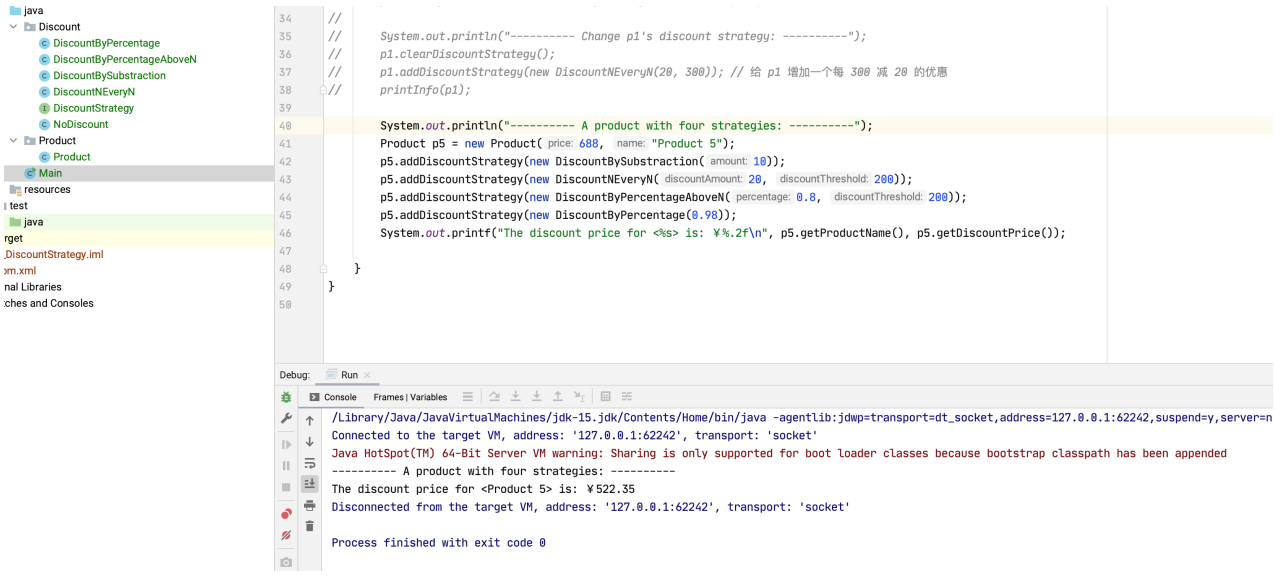


2. 多种折扣方式同时使用怎么办?

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：



完整代码

Main.java

```
import Discount.DiscountByPercentage;
import Discount.DiscountByPercentageAboveN;
import Discount.DiscountBySubstraction;
import Discount.DiscountNEveryN;
import Product.Product;

import java.util.ArrayList;
import java.util.Arrays;

/**
 * Main Class
 *
 * @author billchen
 * @version 1.0
 * @create 2020-12-13 21:58
 **/
public class Main {

    public static void main(String[] args) {

        System.out.println("---------- Four strategies for four products: ----------
");
        Product p1 = new Product(688, "Product 1", new DiscountBySubstraction(100));
// 直降 100
        Product p2 = new Product(688, "Product 2", new DiscountNEveryN(30, 200));
// 每 200 减 30
        Product p3 = new Product(688, "Product 3", new DiscountByPercentage(0.85));
// 85 折
        Product p4 = new Product(688, "Product 4", new
DiscountByPercentageAboveN(0.7, 300));   // 300 元以上 7 折

        ArrayList<Product> products = new ArrayList<Product>(Arrays.asList(p1, p2,
p3, p4));

        products.forEach(oneProduct -> {
            System.out.printf("The discount price for <%s> is: ¥%.2f\n",
oneProduct.getProductName(), oneProduct.getDiscountPrice());
        });

        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
        System.out.printf("The discount price for <%s> is: ￥%.2f\n", p1.getProductName(), p1.getDiscountPrice());

        System.out.println("---------- A product with four strategies: ----------");
        Product p5 = new Product(688, "Product 5");
        p5.addDiscountStrategy(new DiscountBySubstraction(10));
        p5.addDiscountStrategy(new DiscountNEveryN(20, 200));
        p5.addDiscountStrategy(new DiscountByPercentageAboveN(0.8, 200));
        p5.addDiscountStrategy(new DiscountByPercentage(0.98));
        System.out.printf("The discount price for <%s> is: ￥%.2f\n", p5.getProductName(), p5.getDiscountPrice());

    }
}
```

Product.java

```
package Product;

import Discount.DiscountStrategy;
import Discount.NoDiscount;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;

/**
 * @author billchen
 * @version 1.0
 * @create 2020-12-13 22:53
 **/
public class Product {

    private double originalPrice;
    private String productName;

    private ArrayList<DiscountStrategy> discountStrategies;

```

```
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();
```

```
}

```

NoDiscount.java

```
package Discount;

/**
 * No any discount
 *
 * @author billchen
 * @version 1.0
 * @create 2020-12-13 23:08
 **/
public class NoDiscount implements DiscountStrategy{

    @Override
    public double performDiscount(double originalPrice) {
        return originalPrice;
    }

    @Override
    public int getPriority() {
        return 100;
    }
}
```

DiscountByPercentage.java

```
package Discount;

/**
 * Discount by percentage.
 *
 * @author billchen
 * @version 1.0
 * @create 2020-12-13 23:14
 **/
public class DiscountByPercentage implements DiscountStrategy{

    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 :
```

```
                discountThreshold + percentage * (originalPrice -
discountThreshold);
    }

    @Override
    public int getPriority() {
        return 35;
    }
}
```

DiscountBySubtraction.java

```
package Discount;

/**
 * Directly substract certain amount for the original product.
 *
 * @author billchen
 * @version 1.0
 * @create 2020-12-13 23:09
 **/
public class DiscountBySubstraction implements DiscountStrategy{

    private double substractionAmount;

    public DiscountBySubstraction(double amount) {
        substractionAmount = amount;
    }

    @Override
    public double performDiscount(double originalPrice) {
        return Math.max(0, originalPrice - substractionAmount);
    }

    @Override
    public int getPriority() {
        return 80;
    }

}
```

DiscountNEveryN.java

```java
package Discount;

/**
 * Discount certain amount of money for every N amount of money.
 *
 * @author billchen
 * @version 1.0
 * @create 2020-12-13 23:12
 **/
public class DiscountNEveryN implements DiscountStrategy{

    private double discountAmount;
    private double discountThreshold;

    public DiscountNEveryN(double discountAmount, double discountThreshold) {
        this.discountAmount = discountAmount;
        this.discountThreshold = discountThreshold;
        if(discountAmount > discountThreshold) {
            System.out.println("Discount amount larger than discount threshold. Unexpected low price may occur.");
        }
    }

    @Override
    public double performDiscount(double originalPrice) {
        return originalPrice - ((int) (originalPrice / discountThreshold)) * discountAmount;
    }

    @Override
    public int getPriority() {
        return 10;
    }
}
```