题目 1 策略模式和单一实例模式的运用

1. SalesFormatter.java
2. PlainTextSalesFormatter.java
3. HTMLSalesFormatter.java
4. XMLSalesFormatter.java
5. GourmetCoffee.java

题目：Using Design Patterns in the Gourmet Coffee System

背景

In this assignment, you will create another version of the *Gourmet Coffee System*. This version will present the user with four choices:

```
[0] Quit
[1] Display sales (Plain Text)
[2] Display sales (HTML)
[3] Display sales (XML)
choice>
```

The user will be able to display the sales information in three formats: plain text, HTML, or XML. Part of the work has been done for you and is provided in the student archive. You will implement the code that formats the sales information. This code will use the singleton and strategy patterns.

描述

实现下述类图

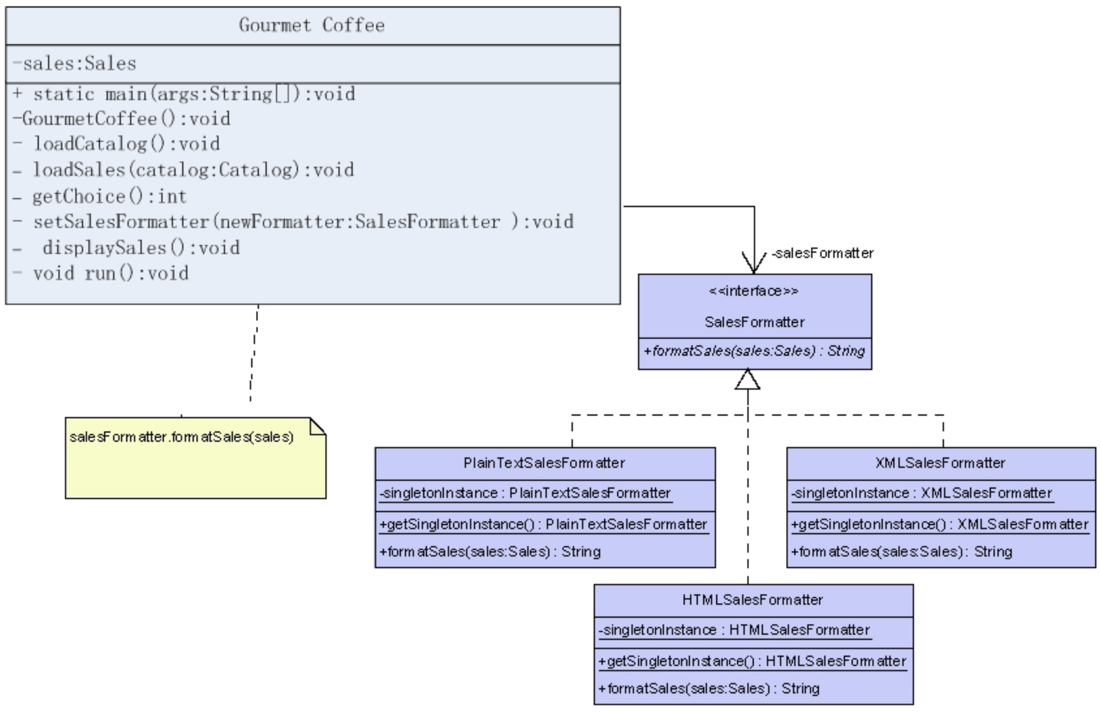
Gourmet Coffee
-sales:Sales
+ static main(args:String[]):void
-GourmetCoffee():void
- loadCatalog():void
- loadSales(catalog:Catalog):void
- getChoice():int
- setSalesFormatter(newFormatter:SalesFormatter):void
- displaySales():void
- void run():void
-salesFormatter
<<interface>>
SalesFormatter
+formatSales(sales:Sales) : String
salesFormatter.formatSales(sales)
PlainTextSalesFormatter
-singletonInstance : PlainTextSalesFormatter
+getSingletonInstance() : PlainTextSalesFormatter
+formatSales(sales:Sales) : String
XMLSalesFormatter
-singletonInstance : XMLSalesFormatter
+getSingletonInstance() : XMLSalesFormatter
+formatSales(sales:Sales): String
HTMLSalesFormatter
-singletonInstance : HTMLSalesFormatter
+getSingletonInstance() : HTMLSalesFormatter
+formatSales(sales:Sales) : String

Figure 1 *Portion of Gourmet Coffee System class diagram*

对于 Class PlainTextSalesFormatter

public String formatSales(Sales sales). Produces a string that contains the specified sales information in a plain-text format. Each order in the sales information has the following format:

Order *number*

quantity1 code1 price1
quantity2 code2 price2
...
quantityN codeN priceN

Total = *totalCost*

where

number is the order number.

quantityX is the quantity of the product.

codeX is the code of the product.

priceX is the price of the product.

totalCost is the total cost of the order.

Each order should begin with a dashed line. The first order in the sales information should be given an order number of 1, the second should be given an order number of 2, and so on.

对于 Class HTMLSalesFormatter

- *public String formatSales(Sales sales)*. Produces a string that contains the specified sales information in an HTML format.
  - The string should begin with the following HTML:
  - <html>
  - <body>
    <center><h2>Orders</h2></center>

  - Each order in the sales information should begin with horizontal line, that is, an <hr> tag.
  - Each order in the sales information should have the following format:

```
<hr>

<h4>Total = totalCost</h4>
  <p>
    <b>code:</b> code1<br>

    <b>quantity:</b> quantity1<br>
    <b>price:</b> price1
 </p>


     ...
 <p>
    <b>code:</b> codeN<br>
    <b>quantity:</b> quantityN<br>

    <b>price:</b> priceN
 </p>
```

where:

- *quantityX* is the quantity of the product.
- *codeX* is the code of the product.

        - *priceX* is the price of the product.
        - *totalCost* is the total cost of the order.

    - The string should end with the following HTML:

```
  </body>
</html>
```

对于 Class XMLSalesFormatter

- *public String formatSales(Sales sales)*. Produces a string that contains the specified sales information in an XML format.

    - The string should begin with the following XML:

```
<Sales>
```

    - Each order in the sales information should have the following format:
    - <Order total="*totalCost*">
        <OrderItem quantity="*quantity1*" price="*price1*">*code1*</OrderItem>
    -
        ...
        <OrderItem quantity="*quantityN*" price="*priceN*">*codeN*</OrderItem>
      </Order>

      where:

        - *quantityX* is the quantity of the product.
        - *codeX* is the code of the product.
        - *priceX* is the price of the product.
        - *totalCost* is the total cost of the order.

    - The string should end with the following XML:

```
</Sales>
```

对于 Class GourmetCoffee

Class GourmetCoffee lets the user display the sales information in one of three formats: plain text, HTML, or XML. A partial implementation of this class is provided in the student archive.

Instance variables:

- *private Sales sales*. A list of the orders that have been paid for.
- *private SalesFormatter salesFormatter*. A reference variable that refers to the current formatter: a PlainTextSalesFormatter, HTMLSalesFormatter, or XMLSalesFormatter object.

Constructor and methods:

The following methods and constructor are complete and require no modification:

- *public static void main(String[] args) throws IOException*. Starts the application.
- *private GourmetCoffee()*. Initialize instance variables sales and salesFormatter.
- *private Catalog loadCatalog()*.Populates the product catalog.
- *private void loadSales(Catalog catalog)*. Populates the sales object.
- *private int getChoice() throws IOException*. Displays a menu of options and verifies the user's choice.
- *private void setSalesFormatter(SalesFormatter newFormatter)*. Changes the current formatter by updating the instance variable salesFormatter with the object specified in the parameter newFormatter.
- *private void displaySales()*. Displays the sales information in the standard output using the method salesFormatter.formatSales to obtain the sales information in the current format.
- *private void run() throws IOException*. Presents the user with a menu of options and executes the selected task
  - If the user chooses option 1, run calls method setSalesFormatter with the singleton instance of class PlainTextSalesFormatter, and calls method displaySales to display the sales information in the standard output.
  - If the user chooses option 2, run calls method setSalesFormatter with the singleton instance of class HTMLSalesFormatter, and calls method displaySales to display the sales information in the standard output.

    - If the user chooses option 3, run calls method setSalesFormatter with the singleton instance of class XMLTextSalesFormatter, and calls method displaySales to display the sales information in the standard output.

所需文件

The following files are needed to complete this assignment:

- *student-files.zip* — Download this file. This archive contains the following:
    - Class files
        - *Coffee.class*
        - *CoffeeBrewer.class*
        - *Product.class*
        - *Catalog.class*
        - *OrderItem.class*
        - *Order.class*
        - *Sales.class*
    - Documentation
        - *Coffee.html*
        - *CoffeeBrewer.html*
        - *Product.html*
        - *Catalog.html*
        - *OrderItem.html*
        - *Order.html*
        - *Sales.html*

任务:

Implement the interface SalesFormatter and the classes PlainTextSalesFormatter, HTMLSalesFormatter, XMLSalesFormatter. Finish the implementation of class GourmetCoffee. Document using Javadoc and follow Sun's code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. implement interface SalesFormatter from scratch.

3. implement class PlainTextSalesFormatter from scratch.

4. implement class HTMLSalesFormatter from scratch.

5. implement class XMLSalesFormatter from scratch.

6. complete the method GourmetCoffee.setSalesFormatter.

7. complete the method GourmetCoffee.displaySales.

8. complete the method GourmetCoffee.run.

9. compile and execute the class GourmetCoffee. Sales information has been hard-coded in the GourmetCoffee template provided by iCarnegie.

   If the user chooses to display the sales information in plain text, the output should be:

```
------------------------
Order 1

5 C001 17.99

Total = 89.94999999999999
------------------------
Order 2

2 C002 18.75
2 A001 9.0

Total = 55.5
------------------------
Order 3

1 B002 200.0

   Total = 200.0
```

   If the user chooses to display the sales information in HTML, the output should be:

```
<html>
  <body>
    <center><h2>Orders</h2></center>
    <hr>
    <h4>Total = 89.94999999999999</h4>
      <p>
        <b>code:</b> C001<br>
        <b>quantity:</b> 5<br>
        <b>price:</b> 17.99
      </p>
    <hr>
    <h4>Total = 55.5</h4>
      <p>
        <b>code:</b> C002<br>
        <b>quantity:</b> 2<br>
        <b>price:</b> 18.75
      </p>
      <p>
        <b>code:</b> A001<br>
        <b>quantity:</b> 2<br>
        <b>price:</b> 9.0
      </p>
    <hr>
    <h4>Total = 200.0</h4>
      <p>
        <b>code:</b> B002<br>
        <b>quantity:</b> 1<br>
        <b>price:</b> 200.0
      </p>
  </body>
</html>
```

If the user chooses to display the sales information in XML, the output should be:

```
<Sales>
  <Order total="89.94999999999999">
    <OrderItem quantity="5" price="17.99">C001</OrderItem>
  </Order>
  <Order total="55.5">
    <OrderItem quantity="2" price="18.75">C002</OrderItem>
    <OrderItem quantity="2" price="9.0">A001</OrderItem>
  </Order>
  <Order total="200.0">
    <OrderItem quantity="1" price="200.0">B002</OrderItem>
  </Order>
</Sales>
```