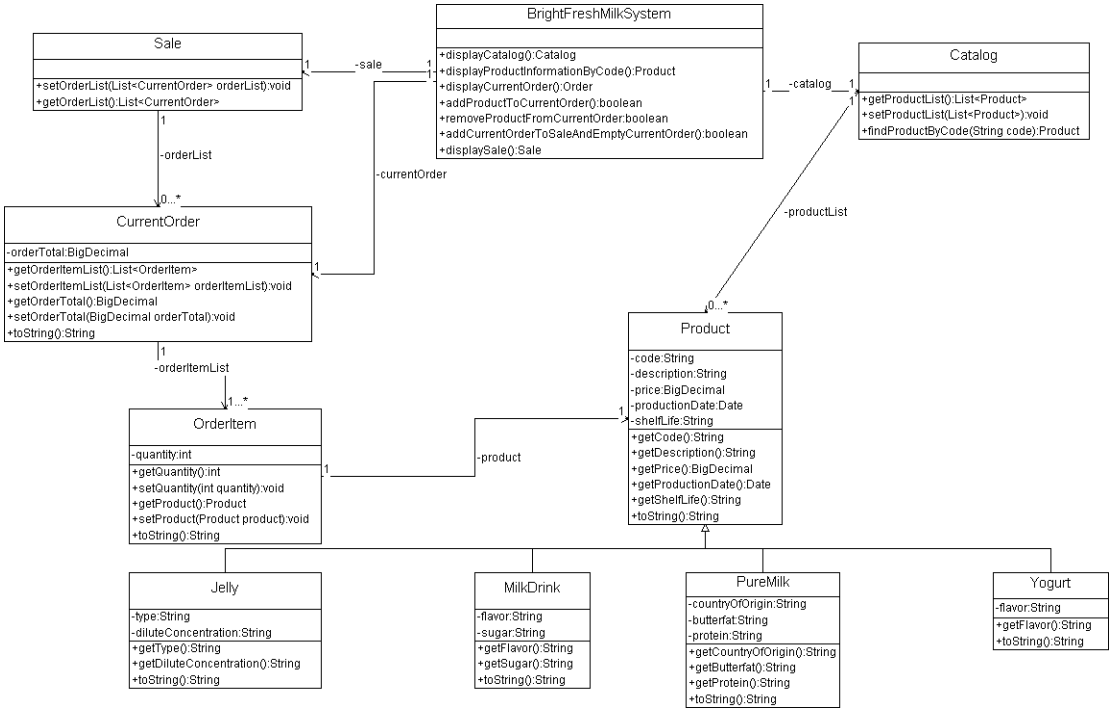
# 题目 1(综合性题目)

1．根据目前所学课堂内容，用 java 逐步编程实现下述类图，遵循 Java 编程规范，并为撰写的类提供相应的 Javadoc 注释。




2．在 BrightFreshMilkSystem.java 中已提供部分辅助函数，该类的其它方法，请按类图中的要求全部编程实现，最终保证程序在步骤 1-7 中的执行中，按要求完成功能。

**程序运行时可供用户选择要实现的功能，如下图。**

```
[0] Quit
[1] Display the catalog
[2] Display the information of a product by its code
[3] Display the current order
[4] Add a product to the current order
[5] Remove a product from the current order
[6] Adds the current order to the store's sales and empties the current order
[7] Display the sales including all the orders that have been sold
choice>
```


**步骤 1： 选择 1（displayCatalog 方法实现的功能）：**

显示目录中各产品的所有属性值，每个产品占 1 行，如下图所示：

```
choice> 1

PureMilk [code=A001, description=milk,400ml, price=10, productionDate=2017-09-14 10:47:26, shelfLife=25 days, countryOfOrigin=China, butterfa
PureMilk [code=A002, description=skim milk,800ml, price=18, productionDate=2017-09-14 10:47:26, shelfLife=30 days, countryOfOrigin=China, but
Jelly [code=B001, description=solid,250ml, price=8.5, productionDate=2017-09-14 10:47:26, shelfLife=15 days, type=solid, diluteConcentration=
Jelly [code=B002, description=solid,400ml, price=14, productionDate=2017-09-14 10:47:26, shelfLife=16 days, type=solid, diluteConcentration=8
Yogurt [code=C001, description=chocolate flavor,270ml, price=5, productionDate=2017-09-14 10:47:26, shelfLife=40 days, flavor=chocolate]
Yogurt [code=C002, description=strawberry flavor,400ml, price=10, productionDate=2017-09-14 10:47:26, shelfLife=40 days, flavor=strawberry]
MilkDrink [code=D001, description=taro flavor,250ml, price=13.1, productionDate=2017-09-14 10:47:26, shelfLife=25 days, flavor=taro, sugar=2.
MilkDrink [code=D002, description=apple flavor,400ml, price=16, productionDate=2017-09-14 10:47:26, shelfLife=30 days, flavor=apple, sugar=3.
```


## 步骤 2： 选择 2（displayProductInformationByCode 方法实现的功能）：

显示指定代码的产品信息，显示格式如下：

```
choice> 2

Product code> A001
PureMilk [code=A001, description=milk,400ml, price=10, productionDate=2017-09-14 10:47:26, shelfLife=25 days, countryOfOrigin=China, butterfa
```

## 步骤 3： 选择 3（displayCurrentOrder 方法实现的功能）：

```
choice> 3

[quantity=2, code=A002, description=skim milk,800ml, price=18]
[quantity=3, code=C001, description=chocolate flavor,270ml, price=5]
[quantity=1, code=D001, description=taro flavor,250ml, price=13.1]
[quantity=3, code=B001, description=solid,250ml, price=8.5]
orderTotal=89.6
```


## 步骤 4： 选择 4（addProductToCurrentOrder 方法实现的功能）：

```
choice> 4

Product code> A001
Add successfully!

[0] Quit
[1] Display the catalog
[2] Display the information of a product by its code
[3] Display the current order
[4] Add a product to the current order
[5] Remove a product from the current order
[6] Adds the current order to the store's sales and empties the current order
[7] Display the sales including all the orders that have been sold
choice> 3

[quantity=2, code=A002, description=skim milk,800ml, price=18]
[quantity=3, code=C001, description=chocolate flavor,270ml, price=5]
[quantity=1, code=D001, description=taro flavor,250ml, price=13.1]
[quantity=3, code=B001, description=solid,250ml, price=8.5]
[quantity=1, code=A001, description=milk,400ml, price=10]
orderTotal=99.6
```

## 步骤 5： 选择 5（removeProductFromCurrentOrder 方法实现的功能）：

```
choice> 5

Product code> A002
Remove successfully!

[0] Quit
[1] Display the catalog
[2] Display the information of a product by its code
[3] Display the current order
[4] Add a product to the current order
[5] Remove a product from the current order
[6] Adds the current order to the store's sales and empties the current order
[7] Display the sales including all the orders that have been sold
choice> 3

[quantity=3, code=C001, description=chocolate flavor,270ml, price=5]
[quantity=1, code=D001, description=taro flavor,250ml, price=13.1]
[quantity=3, code=B001, description=solid,250ml, price=8.5]
[quantity=1, code=A001, description=milk,400ml, price=10]
orderTotal=63.6
```

## 步骤 6：选择 6（addCurrentOrderToSaleAndEmptyCurrentOrder 方法实现的功能）：

```
choice> 6

Adds the current order to the sales and empties the current order successfully!

[0] Quit
[1] Display the catalog
[2] Display the information of a product by its code
[3] Display the current order
[4] Add a product to the current order
[5] Remove a product from the current order
[6] Adds the current order to the store's sales and empties the current order
[7] Display the sales including all the orders that have been sold
choice> 3

The current order is empty!
```

## 步骤 7： 选择 7（displaySale 方法实现的功能）：

```
choice> 7

[quantity=3, code=C001, description=chocolate flavor,270ml, price=5]
[quantity=1, code=D001, description=taro flavor,250ml, price=13.1]
[quantity=3, code=B001, description=solid,250ml, price=8.5]
[quantity=1, code=A001, description=milk,400ml, price=10]
orderTotal=63.6

[quantity=1, code=A001, description=milk,400ml, price=10]
orderTotal=10
```

# 题目 2（java 数组）

## Description

In this assignment, you will finish the implementation of ProductArray, a class with static methods for creating, analyzing, and manipulating arrays of Product objects. iCarnegie provides a test driver and the class Product.

### Class Product

A complete implementation of this class is included in the student archive *student-files.zip*. Stop *now* and review its documentation:

- *Product.html*. Documentation for class Product.

### Class ProductArray

A partial implementation of this class is included in the student archive *student-files.zip*. You should complete the implementation of every method in this class.

### Class TestProductArray

This class is a test driver for class ProductArray. It contains test cases for every method in the class. A complete implementation is included in the student archive *student-files.zip*. You should use this class to test your implementation of ProductArray.

## Files

The following files are needed to complete this assignment:

- *student-files.zip*. Download this file. This archive contains the following:
  - *Product.java*. A complete implementation
  - *ProductArray.java*. Use this template to complete your implementation.
  - *TestProductArray.java*. A complete implementation

## Tasks

Implement all methods in class ProductArray. Follow Sun's code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. **Extract** the files by issuing the following command at the command prompt:

   ```
   C:\>unzip student-files.zip
   ```

2. **Test** each method as soon as you finish writing it by issuing the following command at the command prompt:

   ```
   C:\>java TestProductArray
   ```

3. **Implement** the method makeArray.

- *public static Product[] makeArray(Product first, Product second, Product third)*. This method takes three Product objects and returns an Product array with three elements. The first element of the array contains a reference to the first argument; the second

element contains a reference to the second argument; and the third element contains a reference to the third argument.

For example, consider the following objects:

* Product[102,cruise,68250.0]
* Product[101,domestic,36000.0]
* Product[103,outbound,92175.0]

If these objects are passed to makeArray in the indicated order, makeArray will return the following array:

{ Product[102,cruise,68250.0],
Product[101,domestic,36000.0],
Product[103,outbound,92175.0]}

Note: Product[*ID*,*name*,*price*] is the string representation of an Product object.

4. **Implement** the method copyArray. Use indexes to implement this method.

- *public static Product[] copyArray(Product[] array)*. This method takes an Product array and returns a *new* array with the same elements in the same order.

For example, consider the following array:

{ Product[102,cruise,68250.0],
Product[101,domestic,36000.0],
Product[103,outbound,92175.0]}

If copyArray is passed this array, it will return the following array:

{ Product[102,cruise,68250.0],
Product[101,domestic,36000.0],
Product[103,outbound,92175.0]}

5. **Implement** the method getProduct. Use a for-each loop to implement this method.

- *public static Product getProduct(Product[] array, int id)*. This method takes two arguments, an Product array and an Product ID, and returns the Product object with the specified ID. This method returns null if there are no Products in the specified array with the specified ID.

For example, consider the following array:

{ Product[102,cruise,68250.0],

Product[101,domestic,36000.0],
Product[103,outbound,92175.0]}

If getProduct is passed this array and the integer 103, it will return the Product object for outbound. If getProduct is passed this array and the integer 222, it will return null.

6. **Implement** the method countLowerPrice. Use a for-each loop to implement this method.

- *public static int countLowerPrice(Product [] array, double amount)*. This method takes two arguments, an Product array and a dollar amount, and returns the number of Products in the specified array whose price is lower than the specified dollar amount.

For example, consider the following array:

{ Product[102,cruise,68250.0],
Product[101,domestic,36000.0],
Product[103,outbound,92175.0]}

If countLowerPrice is passed this array and the double 70000.0, it will return 2. If countLowerPrice is passed this array and the double 60000.0, it will return 1.

7. **Implement** the method sumPrice. Use a for-each loop to implement this method.

- *public static double sumPrice(Product[] array)*. This method takes an Product array and returns the sum of the price of the products in the specified array.

For example, consider the following array:

{ Product[102,cruise,60000.0],
Product[101,domestic,30000.0],
Product[103,outbound,90000.0]}

If sumPrice is passed this array, it will return 180000.0.

8. **Implement** the method getLowestPrice. Use indexes to implement this method.

- *public static double getLowestPrice(Product[] array)*. This method takes an Product array and returns the lowest price in the specified array. To simplify your code, you can assume that the specified array contains one or more elements.

For example, consider the following array:

{ Product[102,cruise,60000.0],

Product[101, domestic, 30000.0],
Product[103, outbound, 90000.0]}

If this array is passed to getLowestPrice, it will return 30000.0.

9. **Implement** the method `increaseAll`. Use a `for-each` loop to implement this method.

- *public static void increaseAll(Product[] array, double amount)*. This method takes two arguments, an Product array and a dollar amount, and increases the price of every Product in the specified array by the specified amount.

For example, consider the following array:

{ Product[102, cruise, 60000.0],
Product[101, domestic, 30000.0],
Product[103, outbound, 90000.0]}

If increaseAll is passed this array and the integer 1000, the array will be modified as follows:

{ Product[102, cruise, 61000.0],
Product[101, domestic, 31000.0],
Product[103, outbound, 91000.0]}

10. **Implement** the method `displayAll`. Use indexes to implement this method.

- *public static String displayAll( Product[] array)*. This method takes an Product array and returns a string representation of the specified array. To simplify your code, you can assume that every element in the specified array contains a valid reference to an Product object.

Use the method toString in the class Product to obtain the string representation of each object in the array. A new line character ( \n ) should separate the string representations of each Product object.

For example, consider the following array:

{Product[102, cruise, 61000.0],
Product[101, domestic, 31000.0]}

If this array is passed to displayAll, the following string will be returned:

"Product[102, cruise, 61000.0]\nProduct[101, domestic, 31000.0]"

Note that the string does *not* end with a new line character ( \n ).

# 题目 3（StringTokenizer and Exception）

## Coffee Shop Application

### 背景

本题目将实现一个咖啡店进货的程序。咖啡的代号从 0 到 50，每种咖啡有自己的名字和不同的口味，咖啡最低价格为 2 元，最高价格为 50 元。

Description

The application presents the user with a menu of options and prompts the user for a choice:

- Choice 0 terminates the program.
- Choice 1 adds a coffee to the coffee shop.
- Choice 2 displays the information of all the coffees in coffee shop.
- Choice 3 displays the total cost of all the coffee in the shop.

To add a coffee, the user enters a line with the following format:

*id_name_taste_cost*

Where:

- id is the id of the coffee.
- name is the name of the coffee.
- taste is the *taste* of the coffee.

The fields are delimited by an underscore ( _ ). If the user's input is invalid, the application displays an error message.

The application uses classes Coffee and CoffeeShop. CoffeeShop maintains a collection of coffee. Complete implementations of both are provided in the student archive *student-files.zip*. Review their documentation and become familiar with it.

- *Coffee*. Documentation for class `Coffee`
- *CoffeeShop*. Documentation for class `CoffeeShop`

A partial implement of `CoffeeShopApplication` is provided in the student archive *student-files.zip*. It contains some variables declarations and some methods that need no modification. You should complete method `readCoffee`, the method that reads coffee's information from the keyboard and returns a `Coffee` object.

## Files

The following files are needed to complete this assignment:

- *student-files.zip*. Download this file. This archive contains the following:
  - *Coffee.java*. A complete implementation
  - *CoffeeShop.java*. A complete implementation
  - *Coffee.html*. Documentation
  - *CoffeeShop.html*. Documentation
  - *CoffeeShopApplication.java* — Use this template to complete your implementation.

## Tasks

To complete this assignment, you will finish the implementation of class `CoffeeShopApplication`. The following steps will guide you through this assignment. Document using Javadoc and follow Sun's code conventions. Work incrementally and test each increment. Save often.

1. **Extract** the student archive by issuing the following command at the command prompt:

   ```
   C:\>unzip student-files.zip
   ```

2. Observe how the program responds to the following types of input:
   - Input with a number that is not a valid integer: `a_Mocha_chocolate_3.0`
   - Input with a cost that is not a valid double: `2_Mocha_chocolate_a`.
   - Input that contains negative numbers: `-1_Mocha_chocolate_150.0`, or `1_Mocha_chocolate_-150.0`.
   - Input that contains more than four values:`1_Mocha_chocolate_15.0_1`.
   - Input that contains fewer than four values: `Mocha_chocolate`.
   - Valid input:`1_Mocha_chocolate_3.0`.

2. **Then**, complete method `readCoffee`:

- *private Coffee readCoffee() throws IOException*. This method prompts the user for input, reads coffee information from the keyboard, and creates an instance of class `Coffee`. The coffee information should consist of four values, all entered on the same line, and delimited by an underscore ( _ ). The first value should be a non-empty string that represents the name of the coffee. The second value should be non-empty string that represents the taste of the coffee. The third value should be a positive integer that represents the id of the coffee. The fourth value should be a positive double that represents the cost of the coffee. Use java.util.StringTokenizer to extract the four values from the input.

`readCoffee` validates the user's input:

  - If the user enters more than four values, an error message is displayed.
  - If the user enters fewer than four values, an error message is displayed.
  - If the user enters a number that is not a valid integer, java.lang.NumberFormatException is caught and output.
  - If the user enters a cost that is not a valid double, java.lang.NumberFormatException is caught and output.
  - If the user enters a number that is negative or zero, an error message is displayed.
  - If the user enters a cost that is negative or zero, an error message is displayed.

The error messages displayed by your implementation should match the error messages displayed in the picture above.

If the input is invalid, `readCoffee` re-prompts the user for new input. Otherwise, it creates a new Coffee object using the specified name, taste, id and cost and returns a reference to the new object to the calling method.