HD in Software Engineering -

Contemporary Topics in Software Engineering

(ITP4507)

Report

POON Ngai Kuen (180091780)

Contents

[Assumption 3](#_Toc25320896)

[Analysis 3](#_Toc25320897)

[Design 4](#_Toc25320898)

[Class diagram 4](#_Toc25320899)

[Discussion 5](#_Toc25320900)

[User guide 7](#_Toc25320901)

[Test plan 8](#_Toc25320902)

# Assumption

1. Users always input the right format data.

# Analysis

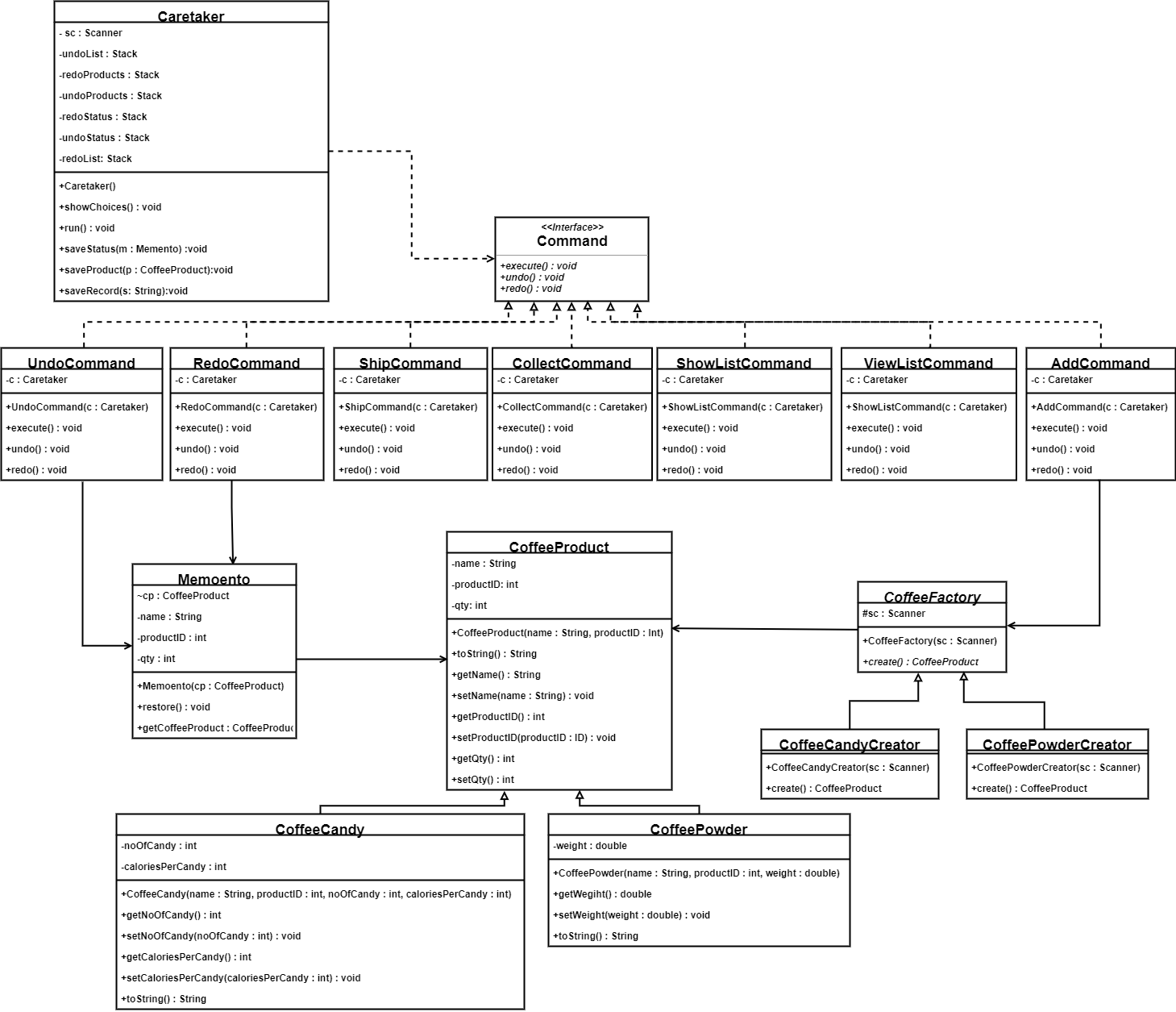
Consideration of future extending function, the system needs to suit the Open Close Principle. I will use the command pattern to design the action of classes so I will create an Interface. In the future, if the programmer wants to add new action, he just needs to add a new class and implements the super class Command.

Also, I will use the Abstract Factory pattern to design the creation of CoffeeProduct object. If the programmer wants to add a new type of coffee product, he just need to add a new class and extends the super class CoffeeProduct and add the related creator class that extends the super class CoffeeFactory that help how the product to be added by users.

Lastly, I will use Memento pattern to design the restore of the status of CoffeeProduct.

# Design

## Class diagram

The diagram is created by draw.io.

# Discussion

The class Caretake is the main class that will provide the main methods, including **run**, **saveStatus**, **saveProduct**, **saveRecord**, and **showChoices**. The description of the methods of Caretake is shown below.

1. **saveStatus**: Used to save the status of the products. For example, the quantity of the product.
2. **saveProduct**: Used to save the product objects.
3. **saveRecord**: Used to save the message that what user did.
4. **run:** Used to run the system and receive the input from the user.
5. **showChoices:** Used to show the command list message to the user.

Also, there are seven member variables on Caretaker

|  |
| --- |
| private Stack<String> undoList;  private Stack<String> redoList;  private Stack<Memento> undoStatus;  private Stack<Memento> redoStatus;  private Stack<CoffeeProduct> undoProducts;  private Stack<CoffeeProduct> redoProducts; |

**undoList:** for saving the text that shows what the user did before.

**redoList:** for saving the text that shows what the user undo the step(s) before.

**undoStatus**: for saving the status of products. It will be used to undo the status of products.

**redoStatus:** for saving the status of products. It will be used to redo the status of products.

**undoProducts:** for saving the new product. It will be used to undo the creation of the new product.

**redoProducts:** for saving the new product. It will be used to redo the creation of the new product.

Command patterns are used for doing the actions such as adding a new product, shipping products, etc. I design seven classes which implement the Interface Command that provides three main method including execute, undo, redo.The name and description of the classes are show below:

1. **AddCommand:** provides the adding products function.
2. **CollectCommand:** provides the adding the quantity of products function that can increase the quantity of the product by product ID.
3. **ShipCommand:** provides the shipping products function that can reduce the quantity of the product by product ID.
4. **ViewCommand:** provides the showing of products list function that shows the specified product by product ID or all products.
5. **UndoCommand:** provides the undo function that returns to the previous step and the current step will pass to the Redo stack. If the current step involves changing the status of products, it will find the top same product of the undoStatus stack and restore the status of the product back while if the current step involves creation of a new product, the product will be deleted from the undoProduct stack, then saved to the redoProduct stack.
6. **RedoCommand:** provides the redo function that can restore the undo function. The top step on the redo stack will be passed to the undo stack. If the step involves changing the status of products, it will call the restore function of memento class while it will add the new product back.
7. **ShowListCommand:** provides the showing list function that shows all the step saved to undo list or redo list.

Memento pattern is used for undo or redo the quantity of products. The class **Memento** is created for save the status of products. When calling the undo or the redo function about the status of products, the object **Memento** returns the status of the product back.

Factory pattern is used for how the coffee product objects to be create by users. CoffeeFactory is an abstract class that provides a creation of CoffeeProduct. The creation of different coffee product class will extend CoffeeFactory to create the object.

# User guide

|  |  |
| --- | --- |
| Actions | Way |
| Adding a new product | 1. Input a 2. Input the type of product according the format. 3. Input product information according the format. |
| Collecting products. | 1. Input c 2. Input the product ID that will be collected. 3. Input the quantity of collecting products. |
| Shipping products | 1. Input s 2. Input the product ID that will be shipped. 3. Input the quantity of shipping products. |
| View product(s) | 1. Input v 2. Input the product ID that you want to query. (input \* will show information of all products. |
| Undo last action | 1. Input u |
| Redo last undo action | 1. Input r |
| Show Undo List & Redo List | 1. Input sl |

# Test plan

I will use **Black Box Testing** to test the system because the system is not complex.

Below are the test cases:

|  |  |
| --- | --- |
| Test case | Add a new product |
| Case ID | T01 |
| User | Normal user |
| Test Flows | 1. Input a 2. Input cc 3. Input 1001, Premium Coffee Candy, 50, 15 |
| Input Data | 1. The command of adding a new product. 2. Type of the new product 3. Information of the new product |
| Expected Results | Display “New product record created.”  The new product has added to the system. |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Add a new product |
| Case ID | T02 |
| User | Normal user |
| Test Flows | 1. Input a 2. Input cp 3. Input 2001, Colombia Coffee, 250 |
| Input Data | 1. The command of adding a new product. 2. Type of the new product 3. Information of the new product |
| Expected Results | Display “New product record created.”  The new product has added to the system. |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Collect products |
| Case ID | T03 |
| User | Normal user |
| Test Flows | 1. Input c 2. Input 1001 3. Input 50 4. Input c 5. Input 1001 6. Input 150 |
| Input Data | 1. The command of collecting products. 2. Product ID 3. Quantity of the shipping products. |
| Expected Results | Display “Received 50 packs of Premium Coffee Candy. Current quantity is 50.”  Display “Received 150 packs of Premium Coffee Candy. Current quantity is 200.” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | View information of a product |
| Case ID | T04 |
| User | Normal user |
| Test Flows | 1. Input v 2. Input 1001 |
| Input Data | 1. The command of querying products. 2. Product ID |
| Expected Results | Display “  Product information:  ID: 1001  Name: Premium Coffee Candy  Quantity: 200  Number of candies per package: 50  Calories Per candy: 15” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Ship products |
| Case ID | T05 |
| User | Normal user |
| Test Flows | 1. Input s 2. Input 1001 3. Input 100 |
| Input Data | 1. The command of shipping products. 2. The product ID 3. The quantity of shipping products |
| Expected Results | Display “Shipped 100 packs of Premium Coffee Candy. Current quantity is 100.” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | View information of all products |
| Case ID | T06 |
| User | Normal user |
| Test Flows | 1. Input v 2. Input \* |
| Input Data | 1. The command of querying products. 2. The command of show all products |
| Expected Results | Display “  Coffee Product information:  ID Name Quantity Other Info  1001 Premium Coffee Candy 100 50 candy per package (15 calories each)  2001 Colombia Coffee 0 250g” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Show Undo List and Redo List |
| Case ID | T07 |
| User | Normal user |
| Test Flows | 1. Input sl |
| Input Data | 1. The command of show Undo List and Redo List. |
| Expected Results | Display “  Undo List  Added 1001 Premium Coffee Candy  Added 2001 Colombia Coffee  Received 50 Premium Coffee Candy (1001)  Received 150 Premium Coffee Candy (1001)  Shipped 100 Premium Coffee Candy (1001)  Redo List  Empty” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Undo |
| Case ID | T08 |
| User | Normal user |
| Test Flows | 1. Input u 2. Input u |
| Input Data | 1. The command of Undo |
| Expected Results | Shipped 100 Premium Coffee Candy (1001) will be cancelled and saved to Redo List  Received 150 Premium Coffee Candy (1001) will be cancelled and saved to Read List |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Show Undo List and Redo List |
| Case ID | T09 |
| User | Normal user |
| Test Flows | 1. Input sl |
| Input Data | 1. The command of show Undo List and Redo List. |
| Expected Results | Display “  Undo List  Added 1001 Premium Coffee Candy  Added 2001 Colombia Coffee  Received 50 Premium Coffee Candy (1001)  Redo List  Shipped 100 Premium Coffee Candy (1001)  Received 150 Premium Coffee Candy (1001)” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | View information of all products |
| Case ID | T10 |
| User | Normal user |
| Test Flows | 1. Input v 2. Input \* |
| Input Data | 1. The command of querying products. 2. The command of show all products |
| Expected Results | Display “  Coffee Product information:  ID Name Quantity Other Info  1001 Premium Coffee Candy 50 50 candy per package (15 calories each)  2001 Colombia Coffee 0 250g” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | Redo |
| Case ID | T11 |
| User | Normal user |
| Test Flows | 1. Input r 2. Input r |
| Input Data | 1. The command of Redo. |
| Expected Results | Shipped 100 Premium Coffee Candy (1001) will be restored and saved to Undo List  Received 150 Premium Coffee Candy (1001) will be restored and saved to Undo List |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | View information of all products |
| Case ID | T12 |
| User | Normal user |
| Test Flows | 1. Input v 2. Input \* |
| Input Data | 1. The command of querying products. 2. The command of show all products |
| Expected Results | Display “  Coffee Product information:  ID Name Quantity Other Info  1001 Premium Coffee Candy 100 50 candy per package (15 calories each)  2001 Colombia Coffee 0 250g” |
| Pass/Fail |  |

|  |  |
| --- | --- |
| Test case | View information of all products |
| Case ID | T13 |
| User | Normal user |
| Test Flows | 1. Input x |
| Input Data | 1. The command of exit the system. |
| Expected Results | The system is end.  Display “Thanks for using Coffee Inventory Management System!!” |
| Pass/Fail |  |