Exercise 3

1 Objective

Practice UML and Design Patterns

2 Marking

This exercise will be graded of of 36 marks: 30 functionality and the rest quality.

It is worth 3% of your final grade.

The submission deadline is July 14, 2016, 11:50pm.

This is an individual exercise.

Late submission policy: No late submissions are accepted.

3 How to submit your work

1. Your individual svn repository now contains a new directory called E3. It contains the starter code for this exercise. Checkout and study the starter code.

2. Complete the code as indicated in this handout and in the starter code.

3. To submit your work, add and commit your changes to your repository.

Do not commit the files and directories generated by Eclipse, such as bin, doc, .project, etc. Marks will be deducted if you submit these.

4 The wallpaper manufacturing and distribution problem

The problem stated in this exercise is a simplified version of an actual software industry problem, solved and implemented in various commercial wallpaper manufacturers in North America.

An important specific of wallpaper manufacturing (shared with many other manufacturing processes) is that there is a minimum amount of wallpaper that can be printed in one production run (1000 yards for most common types of commercial wallpaper).

The whole process can be described as follows. For each Product, the warehouse maintains an Inventory item, which has two attributes: the product, the available quantity on the inventory, and the quantity required by the customers, also known as the backordered quantity. Also the warehouse keeps track or SalesOrders and ships them out as soon as sufficient quantity becomes available.

The manufacturing facility, maintains a ProductionOrder for each inventory item. A production order is used to manufacture a quantity greater or equal to the minimum quantity, and also sufficient to cover the quantity required by SalesOrders.

It is clear that an Inventory item is an Observable and SalesOrders and ProductionOrders are Observers. In addition, both SalesOrders and ProductionOrders implement a DisplayElement interface as shown in the UML diagram below.

So here is how the solution works. Initially, we set up a number of Customer objects, and a number of Product objects. For all the products that we intend to manufacture and sell, we set up and Inventory item that will be an Observable.
Who observes Inventory items? Well, in order to sell, we should manufacture some quantities, so we set up a ProductionOrder that is supposed to generate some quantity if there is a demand for that particular inventory item.

What does demand mean? Well, customers place SalesOrder items, which are supposed to be fulfilled by the warehouse, given that there is enough quantity on Inventory. So, at this point, please identify the other Observable.

Also both Observables implement DisplayElement interface that allows the display of observables.

The wallpaper manufacturing and distribution workflow is as follows:

1. Perform the setup (generate Customers, Products, Inventory items and ProductionOrders). Note that ProductionOrders are observables!

2. Customers place SalesOrders.

3. If there is enough quantity on inventory, the SalesOrder gets shipped immediately so it does not need to go to the list of observers. Otherwise it should be registered as an Observer.

4. When an SalesOrder gets registered as an observer, the backorderedQuantity should be updated accordingly.

5. Meanwhile, the ProductionOrder gets notified so if the backorderedQuantity is greater or equal than minQuantity, the ProductionOrder activates the manufacturing facility to make some quantity (sufficient to cover all SalesOrders!) and updates the availableQuantity.

6. The SalesOrders should get notified on the availability and ship out. The inventory should be maintained accordingly.

7. If a quantity is made by a ProductionOrder or if a quantity is shipped in behalf of a SalesOrder the display method should display the appropriate information on the console.

A complete example is shown below. The code (in the main method) that generates this output is provided in full in the starter code.

Shipping Order# 3 to Home Depot, Product: Flower Field, Quantity: 800.0
Shipping Order# 1 to Wal-Mart, Product: Flower Field, Quantity: 500.0
Production Order# 1, item Flower Field, produced 1300.0
Shipping Order# 4 to Home Depot, Product: Stormy Sea, Quantity: 1400.0
Shipping Order# 2 to Home Depot, Product: Stormy Sea, Quantity: 700.0
Production Order# 2, item Stormy Sea, produced 2100.0
321 Stormy Sea, Available: 0.0, Backorders: 0.0
[PO2 Stormy Sea 2000.0]
123 Flower Field, Available: 0.0, Backorders: 0.0
[PO1 Flower Field 1000.0]

5 What is to be done?

- Read the exercise carefully and study the provided UML diagram.
- Check our the starter code and using the UML diagram, write your solution.
- Please do use the same names for your classes, class variables and methods as indicated in the associated UML diagram.
- Throughout your coding process, follow the UML diagram carefully.
- Test your application thoroughly. You do not need to submit your test code.
- Submit only classes and interfaces shown on the UML diagram.
6  The UML Diagram

7 Checklist

Have you...

• tested your code on the lab computers using Java 1.8?
● committed the correct files in the correct directory?
● verified that your changes were committed using `svn list` and `svn status`?
● tested your solution, made any necessary changes, and re-committed if necessary?