COMP62542 Pattern-Based Software Development
Coursework Specification
Semester 2, 2019-2020

General Information

Lecturer: Liping Zhao

Coursework: This is the only coursework for this course unit.

Marks: The coursework is marked out of 100.

Requirement: *To be completed independently by individual students; no groupwork is allowed.*

Submission Method: Blackboard, detailed to be announced

Submission Deadline: 17:00 British Summer Time, Monday 11 May 2020.

Late Submission: Extension will only be granted as a result of formally processed Mitigating Circumstances (http://documents.manchester.ac.uk/display.aspx?DocID=24561). Marks for late submissions will be reduced in line with the following university policy (http://documents.manchester.ac.uk/display.aspx?DocID=24561)

Specification

This coursework should be completed independently by individual students; no groupwork is allowed. The coursework requires using some design patterns to design and implement a simple checkout system for an online grocer. The implementation should be realized by Java programming language. The checkout system is described as follows.

A small online grocer currently only sells three types of product: milk (skimmed, semi-skimmed and whole milk), bread (whole-wheat, sourdough, rye, and pita) and fruit (apple, banana and orange). You are asked to build a checkout system for this grocer. Your system should support the following scenarios:

1. *Check and display the price for each scanned product*. Note: you can assume the price for each product, e.g., apple - £2 per kilo; bread - £1.5 per pack; milk: 50p per pint.
2. *Display the price for each product from each presented shopping cart and show the total price for all the products in the shopping cart.* Note: you can assume the products in each shopping cart yourself, but they need to be different; you can also assume the price for each product.
3. *Apply special offer to the products in a given shopping cart.* Note: assume special offers include buy one get one free and buy three for the price of two; you can also assume the price for each product.
4. *Simulate the payment process.* Note: the payment process must consider at least three different payment methods. Examples are Paypal, card payment and Google Pay. Each method requires a different set of information, similar to the real-world situation. The payment process should display different states of payment: payment pending, payment accepted, and payment rejected.

To design this checkout system, you are required to use **at least four design patterns** that you have learned from Week 1 to Week 4 in this course. You can add additional patterns if they make your design more flexible. **NB. You should implement each pattern yourself and you are not allowed to use any Java inbuilt patterns in your design**. You are NOT required to use MVC; but if you do decide to use it, you can use Java Spring MVC Framework. However, the use of MVC does not count towards the four patterns required. Use the following questions to help you assess and determine if a particular pattern should be applied to your design:

- Does this pattern help to solve a particular problem in your design? Would the problem warrant the use of this pattern in that it cannot be solved by a simpler solution?
- What is the relationship between this pattern and the patterns you have already applied to your design? Is the use of this pattern really necessary? Why?
- Will this pattern help improve the cohesion and coupling of the overall system? If so, how is the improvement being made?

- Will this pattern make your overall design easy to change? If so, which aspect of the design can be changed? How is the change being enabled? Is it the design time change or the runtime change?
- Is your choice of this pattern guided by the design pattern principles? If so, which of these principles is your main motivation for using the pattern: Is it "program to an interface, not to an implementation", "favour object composition over class inheritance" or "design for change"?

Submission Requirements

You are required to submit the following documents for assessment:

- **A written report** containing the following sections:
  1. Your name and email address.
  2. Describe each pattern in your system by answering the questions stated above.
  3. Show the overall design of your system by using a UML Class Diagram
  4. Describe briefly the main classes in each pattern, how these classes interact and the main implementation techniques that you feel most proud of.
  5. Describe how to run your system and how to execute each of the above-mentioned scenarios.
- The **source code** of your program.

The submission method and venue will be announced in due course.