Software Engineering
Homework 3 – Design Patterns

Homework topic(s): Design Patterns
Homework objective(s): Practicing of the:

- Identification of the different classes from the requirements
- Definition the class diagram
- Use design patterns to design improved software solutions
- Java implementation of the designed solutions
- Encourage team work. Each team should be composed of 2 students
- Individual submissions are also possible.

System description:
The word has spread about your successful implementation of the order management system (OMS) for VIPResto Inc. You received an email from the manager of PizzaStar Inc. who heard about your success. He wants you to implement a new OMS for his pizza delivery shop. For the first milestone, he has defined several requirements that need to be fulfilled by your OMS:

- The shop offers pizza, several beverages (lemonade, water and wine) as well as some franchise items (shirts, mugs). The system needs to be able to represent these products.
- All products have a price and pizza and beverages have specific nutrition facts.
- Every pizza contains at least cheese and tomato sauce (Pizza Margherita). It can be combined with additional toppings (ham, onions, etc.), which can be ordered multiple times.
- Pizzas with certain topping combinations are named, e.g., Hawaiian Pizza for the base pizza with ham and pineapple. Those names should be contained in the bill. You do not need to auto-detect names for such combinations if they are not explicitly requested in the order.
- In addition to the normal size, pizza can be ordered in family size.

The manager sent you the menu card of PizzaStar for a better understanding of their available items:

Pizzas	**Calories**	**Price**
Pizza Margherita (tomato, cheese)	1104	4.95
Hawaiian Pizza (tomato, cheese, ham, pineapple)	1024	6.45
Salami Pizza (tomato, cheese, salami)	1160	5.95
Family Size for Pizza	x 1.95	+ 4.15
Toppings	**Calories**	**Price**
Cheese	92	0.65
Ham	35	0.95
Onions	22	0.65
Pineapple	24	0.75
Salami	86	0.95
Drinks	**Calories**	**Price**
Lemonade (0.33l)	128	1.25
Water (0.5l)	0	1.25
Wine (0.75l, 13%)	608	7.45

Franchise		**Price**
PizzaStar Shirt		21.95
PizzaStar Mug		4.95

Students are required to respond to the below 3 tasks:

Task1: Solution 1, no design pattern is used:

- Provide a UML class diagram that illustrates your design for this solution.
- Implement an OMS library in Java for PizzaStar, Inc that provides the requested features. Add a client class to document the usage of your library.
- Your library design should use immutable objects, i.e., orders and pizzas are never changed in-place. For example, adding a topping to a pizza creates a new pizza object.
- The implementation should be in line with (i.e. 100% conform to) the class diagram.

Task2: Solution 2, use the Decorator design pattern:

- Use the Decorator pattern to implement the different variations of pizza (additional toppings, family sized). Add a client class to document the usage of your library
- Provide a UML class diagram that illustrates your design.
- The implementation should be in line with (i.e. 100% conform to) the class diagram.

Note that there is one single PizzaStar shop on earth, make sure that your solutions are well designed so that they prevent the proliferation of fake PizzaStar shops by allowing the creation of only one PizzaStar shop. Also note that your programs should allow the client to display: the name, the size, the total cost, the total calories, and the toppings (name, quantity, calories, cost) of each pizza he/she orders.

Task3:

1. Discuss the main differences between the two solutions (compare them and give the pros and cons of each one).
2. Would it be possible to use the Abstract Factory Design pattern to implement a 3rd solution that is better than those suggested above? Give a detailed justification.
3. Could you think about any other pattern that you can use to further improve solution 2. Give the details of your suggestions

Submission instructions

- **Format:** A compressed file composed of the Eclipse project ready to be tested and a PDF file containing (i) the two class diagrams, (ii) your answers to the questions of task3, (iii) and any additional clarification you want to add.
  Make sure that you submit a clean and fully working java code. A null grade will be given to any program that is not working due to syntax or logic errors.
- **Deadline:** Sat. May 4, 2019 at 11:55pm
- The system will be configured to allow two submissions, but only the last one will be graded.
- The system will be configured to allow late submissions, but these will be subject to 5 points penalty per 24h (even if you submit 1 minute after the deadline).
- Email submissions will not be graded unless there is a major technical problem.