MINISTRY OF EDUCATION AND RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FUNDAMENTAL PROGRAMMING TECHNIQUES

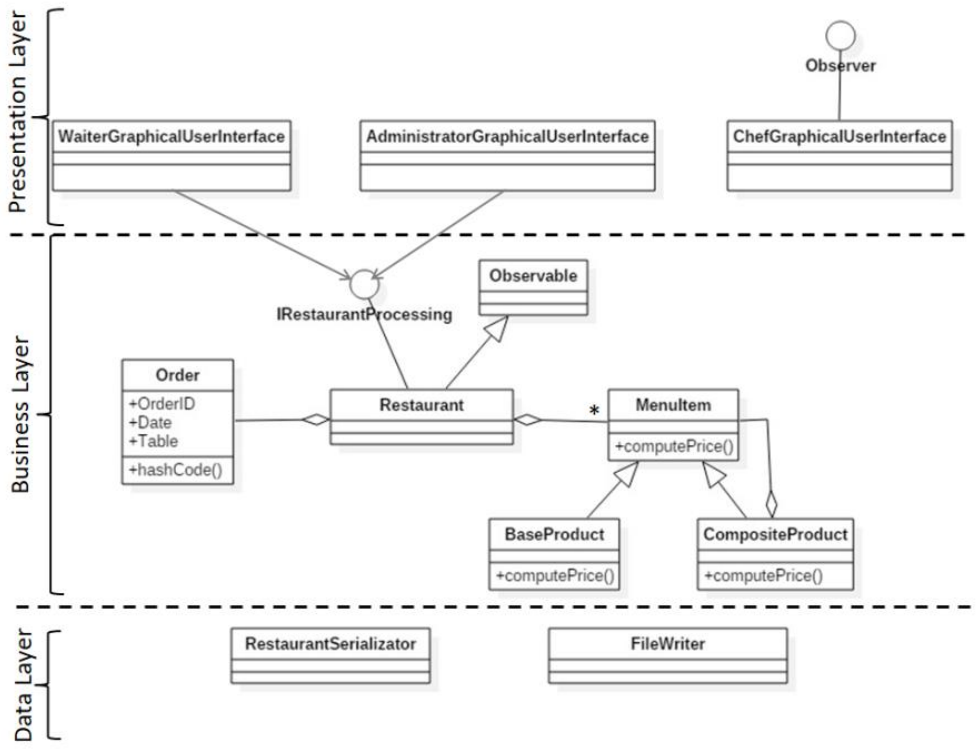
ASSIGNMENT 4

RESTAURANT MANAGEMENT SYSTEM

1. Requirements

Consider implementing a restaurant management system. The system should have three types of users: administrator, waiter and chef. The administrator can add, delete and modify existing products from the menu. The waiter can create a new order for a table, add elements from the menu, and compute the bill for an order. The chef is notified each time it must cook food that is ordered through a waiter.

Consider the system of classes in the diagram below.



To simplify the application, you may assume that the system is used by only one administrator, one waiter and one chef, and there is no need of a login process.

Solve the following:

1) Define the interface IRestaurantProcessing containing the main operations that can be executed by the waiter/administrator, as follows:
    - Administrator: create new menu item, delete menu item, edit menu item
    - Waiter: create new order; compute price for an order; generate bill in .txt format.
2) Define and implement the classes from the class diagram shown above:
    - Use the Composite Design Pattern for defining the classes MenuItem, BaseProduct and CompositeProduct

    - Use the Observer Design Pattern to notify the chef each time a new order containing a composite product is added.

3) Implement the class Restaurant using a predefined JCF collection that is based on a hashtable data structure. The hashtable key will be generated based on the class Order, which can have associated several MenuItems. Use JTable to display Restaurant related information.
    - Define a structure of type Map<Order, Collection<MenuItem>> for storing the order related information in the Restaurant class. The key of the Map will be formed of objects of type Order, for which the hashCode() method will be overwritten to compute the hash value within the Map from the attributes of the Order (OrderID, date, etc.).
    - Define an appropriate collection consisting of MenuItem objects to store the menu of the restaurant.
    - Define a method of type "well formed" for the class Restaurant.
    - Implement the class Restaurant using Design by Contract method (involving pre, post conditions, invariants, and assertions).
4) The menu items for populating the Restaurant object will be loaded/saved from/to a file using Serialization.

2. Deliverables

- A **solution description document** (minimum 2000 words, Times New Roman, 10pt, Single Spacing) with the structure specified in the **Lab Description** document.
- **Source files**
- **JavaDoc files** including the custom tags and descriptions associated to the defined pre, post conditions and invariants
- **jar file** required for executing the application
- **restaurant.ser** - the file in which the Restaurant object is saved through serialization

The deliverables will be **submitted** as follows:

- Create a repository on **gitlab** with the name: ***PT2020_Group_LastName_FirstName_Assignment_4***
- Push the following: **source code (push the code not an archive with the code), jar file, restaurant.ser file, documentation**
- Share the repository with the user **utcn_dsrl.**

3. Evaluation

The assignment will be graded as follows:

Requirement	Grading
Minimum to pass • Object-oriented programming design • Classes with maximum 300 lines • Methods with maximum 30 lines • Java naming conventions • Basic documentation • Implement the class diagram from the homework specification. Choose wisely the appropriate data structures for saving the Orders and the MenuItems • Graphical interface: ○ Window for Administrator operations: add new MenuItem, edit MenuItems, delete MenuItems, view all MenuItems in a table (JTable) ○ Window for Waiter operations: add new Order, view all Orders in a table (JTable), compute bill for an Order • jar file - the application should permit to be run with the following command: **java -jar PT2020_Group_LastName_FirstName_Assignment_4.jar**	5 points
Quality of the Documentation	1 point
Use Composite Design Pattern for modelling the classes MenuItem, BaseProduct, CompositeProduct.	1 point
Create bill in .txt format.	0.5 points
Design by contract: preconditions and postconditions in the IRestaurantProcessing interface. Implement them in the Restaurant class using the assert instruction. Define an invariant for the class Restaurant.	1 point
Window for Chef user: use Observer Design Pattern to notify each time a new Order is added	0.5 points
Save the information from the Restaurant class in a file (i.e. **restaurant.ser**) using serialization. Load the information when the application starts. Consequently, the application should permit to be run with the following command: **java -jar PT2020_Group_LastName_FirstName_Assignment_4.jar restaurant.ser**	1 point

4. Bibliography

- **Java serialization**
  - http://www.tutorialspoint.com/java/java_serialization.htm
  - https://www.baeldung.com/java-serialization
  - https://www.geeksforgeeks.org/serialization-in-java/
  - https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html
- **Java HashMap**
  - http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html
- **Java assert**
  - http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html
  - http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.html
  - http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise
  - https://intellij-support.jetbrains.com/hc/en-us/community/posts/207014815-How-to-enable-assert
- **Adding custom tags to javadoc**
  - https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag