Universitatea Tehnica din Cluj-Napoca
Departament Calculatoare
Tehnici de Programare, 2019

Prof. Ioan Salomie, Conf. Tudor Cioara, S.L. Cristina Pop
{ioan.salomie, tudor.cioara, cristina.pop}@cs.utcluj.ro

TP Lab–Homework 4

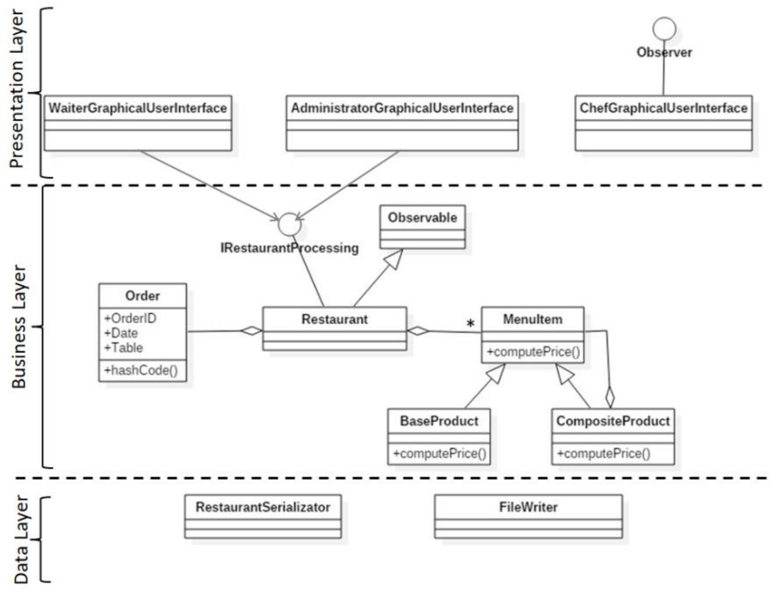
Objectives

- Design by Contract Programming Techniques
- Polymorphism
- Design Patterns: Observer, Composite
- JCF HashMap and HashSet implementations
- Serialization

Description

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 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 RestaurantProcessing containing the main operations that can be executed by the waiter or the 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 which uses 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 a structure of type Collection<MenuItem> which will save the menu of the restaurant. Choose the appropriate collection type for your implementation.
   - Define a method of type “well formed” for the class Restaurant.
   - Implement the class 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.