COSC 436 Dierbach
Spring 2019

Assignment 4 - Automated Restaurant System (Command Design Pattern)

50 pts.

Due Sunday, April 7th by 11:59pm

PROBLEM

You are to implement an automated restaurant system utilizing the Command design pattern, minimally consisting of the following three commands. You will then extend this basic system in any way that you choose to demonstrate the use of additional design patterns.

- Display menu
- Submit Order
- Display tab

SCENARIO (of the basic system)

In the basic system, we assume that all orders are from the same table (i.e., there is only one table in the restaurant). Therefore, a tab is generated by simply totaling all of the ordered items, and tabs are not stored.

The menu of the basic system will consist of just main entrees. It will not include appetizers, desserts, drinks, etc. The information for each entrée will just be the name of the dish ("Roast Beef", "Chicken Korma", "Jiaozi", etc.) and the price.

APPROACH

You should implement the Command design pattern. This includes the following:

- Text-based user interface
- SystemInterface class
- **Invoker class**
- **Command interface**
- **Command classes (one for each command)**
- **Aggregator class**
- Menu class
- MenuItem Class
- Orders class
- OrderItem Class
- Tab class

interfaces/classes of the Command design pattern

The user interface should just be a text-based numbered list of options, implemented in the main method. (It can be a GUI if you desire and are familiar with the development of GUIs, but no extra points will be given for this).

The SystemInterface can be a class of all static methods (one for each of the commands of the user interface) if it does not have any state in your extension of the program.

The Aggregator class maintains references to the Menu object and the Orders object. It should provide a getter method for retrieving the Menu and Orders objects (no setters are needed). The Menu and Orders classes store a collection of MenuItem and OrderItem objects, respectively. A MenuItem object will store the menu item #, the description, and its cost. An OrderItem object will store an order by its item number only (not its description).

The Invoker class has methods cooresponding to the methods of the system interface. Each method creates a Command object of the appropriate Command class (constructed with a reference to the Aggregator object, and any other parameters providing needed information for the Command object), calls its execute method, and returns the single object results it gets back to the system interface. (Execute methods should not be passed any parameter values - any needed values are passed to theconstructor.)

A tab will be constructed from the Tab class containing all of the ordered items, returned as an array of strings for the user interface to display. Note that a tab needs information from both the Menu and the Orders objects. (The Orders object indicates what menu items were ordered, and the Menu class has the description of each item to include in the Tab.)