İzmir University of Economics

2021-2022 Spring

SE311 Project

Pizza Company

Project #3

1.Thought process

The patterns to be applied are chosen separately for each task specifically. We discussed the requirements of each task and tried to match it with the corresponding design pattern.

The first task requires the distributor center to observe all restaurants inventory status and when it falls below the threshold, the distributor center is notified. The notion of observing and notifying resembles the **observer design pattern**.

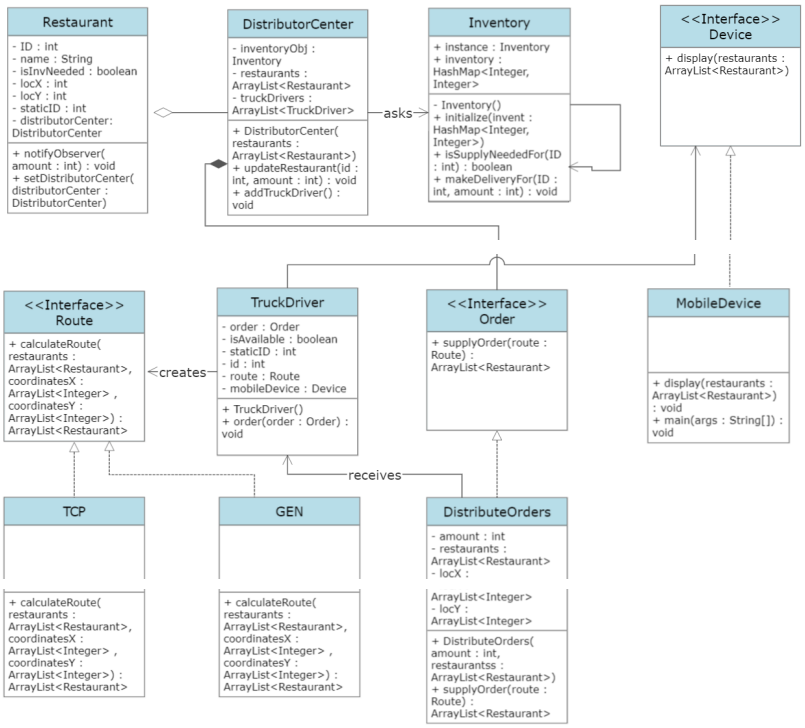
The second task requires a single database to be used. By the word choice and to maintain the data only in one object, we have decided to choose the **singleton design pattern** for it. By using the singleton pattern, that is called “Inventory” in our project, we managed to maintain the inventory information for each restaurant on a single database in the distributor center.

The third task explains the delivery system. Each driver takes an order from the distributor center and applies the order. This is like taking a command from the higher power and applying it to the smaller selections. Thus, we believe that the design pattern to be used in this task is the **command pattern**.

Fourth task works as a factory, that is calculating routes on different algorithms. The same method (calculateRoute()) will be used, but it will call either TSP or GEN. Therefore, we decided to use an interface that has the method calculateRoute() and depending on the choice the same method can call TSP or GEN. The two different choices for the same task reminded us of the **factory design pattern**, which we implemented.

Fifth task is the wrap-up for all the things we have done. Additionally, we have to display our work on another user interface, so we have to connect our system with another system, which reminded us of the **adapter design pattern**.

2.UML Diagram



3.Class Explanation

1. **Restaurant**
   It plays roles of,
   - Subject in the Observer Design Pattern → It is the subject that is being observed by the DistributorCenter. It is being observed and when it signals the notification to the DistributorCenter, its inventory is updated.

2. **DistributorCenter**
   DistributionCenter is the manager class of the system. It becomes the bridge. It plays roles of,
   - Observer in the Observer Design Pattern → It observes the restaurants and gets notified from them. It has the update method to update the restaurant that sends the notification. When there is a pizza delivery, the restaurant notifies the observer (DistributionCenter) and the update method is called, the status (inventory) of the restaurant is updated.
   - Client in the Singleton Design Pattern → It communicates with the Singleton, that is the Inventory. It uses the Inventory, and gets hold of it. DistributionCenter is the only accessor of the single object of the inventory.

- Invoker in the Command Design Pattern → It invokes the Order interface, being the creator of it. Therefore, the relation between them is the composition, orders cannot exist without the invoke of the DistributionCenter.

3. **Inventory**
   Inventory is the database of the system. Its instance is kept within the class and accessed by the DistributorCenter, which makes the DistributerCenter the holder of the database. Inventory plays roles of,
   - Singleton in the Singleton Design Pattern → Inventory has only one object, that is within itself. Because we restrict the program to create only 1 object, and it is already within the class, we restrict the construct as private, eliminating the possibility of other classes to create an object of the Invoker. Only Invoker can create an Invoker object, that is the only object. We give access to this object, including the possibility of changing and altering it, the only restriction is the amount. This creates the Singleton of the pattern.

4. **TruckDriver**
   It plays the role of,
   - Receiver in the Command Design Pattern → It calls the method supplyOrder() of DistributeOrders and gets the result of it, making it the receiver part of the command design pattern.
   - Factory in the Factory Design Pattern → It gives the order to create routes via a Route object. Either TSP or GEN route is calculated and received, making the TruckDriver the factory of the pattern.
   - Cient in the Adapter Design Pattern → It requests a new device to screen the the results; thus, it become the client of this pattern.

5. **DistributeOrders**
   It plays the role of,
   - ConcreteCommand in the Command Design Pattern → The order interface is implemented by DistributeOrders. The execute method (supplyOrder() within this project) returns the list of restaurants to visit. This list is received by the TruckDriver, making it the Receiver class of the pattern.

6. **MobileDevice**
   It plays the role of,
   - Adaptee in the Adapter Design Pattern → The version of a device, MobileDevice plays the role of adaptee as a new way of being the user interface / new device.

7. **TCP / GEN**
   They play the role of,

- Product in Factory Design Pattern → The factory can create various types of products with the same execution order, which is given through the Route interface in this project. They both implement this interface and produce different routes (products).

4. Outputs