Assignment2
Name: Kaichen Zhang ID: 40000160

Question #1
Which architectural style would you select – justify your decision briefly (1 sentence).

a.) You were hired to develop a new application, which read some data from a database, sorts the data then does some calculations and some grouping on the newly calculated data, stores then and prints different reports based on this data. The program should run automatically at the end of the month.

Answer:

Data flow architecture (Batch Sequential system).

Because the uni-directional data flow is components reading from database, components are independent programs, also there is a periodic report.

b.) An organization would like to develop a new application that allows easy sharing of data among their employees in their organization. As part of this architecture, user should be able to provide data to other people in the organization and vice versa.

Answer:

Blackboard style (active repository)

Because this application needs concurrent data sharing among people in the organization, users need to be informed with changes of data or access of data, and such data can be considered as knowledge base.

Question #5 (55 points)
Your are developing a software system to support a gas station. The owner of a gas station has given you the following description.

Before being able to pump gas, a user has to insert his/her credit card in the card reader and select the pre-payment option to authorize the fill-up amount of gas. The pre-payment option performs several operations in the background, such as verifying that the card is not damaged, connecting to the credit card processing system, to ensure that the card is not reported stolen, the available balance on the card is sufficient to cover the gas purchase and to charge the card for the pre-payment amount.

Once the payment is approved, the user picks up the nozzle from the holster and starts fueling. The pump shows in different displays, the amount of fuel has been already pumped, the total $ amount and the number of liters left until the prepaid amount is reached. Once the prepaid amount is reached, fueling stops, the user returns the nozzle and a receipt is printed for the prepaid amount and the fueling is completed. In a case of a problem during the fueling, the system will stop the fueling and refund the credit card for the non-used amount.

1. Which architectural style would you suggest for the above problem. Justify briefly your decision.

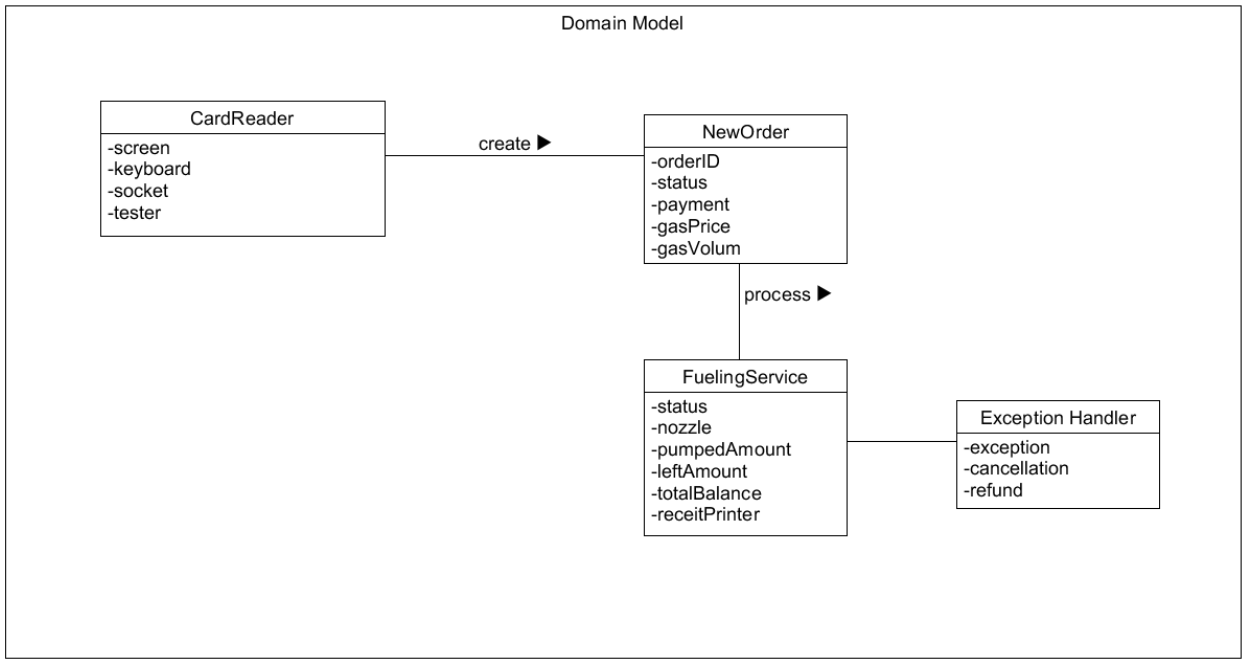
   Answer:

   Process control architecture(closed-loop)

   Because it is a sustaining system for controlling correct gas and payment values, and errors can be handled by feedback. It's a closed-loop system because it uses process information for compensating variations.

2. Create a domain model for the above gas station problem

Answer:

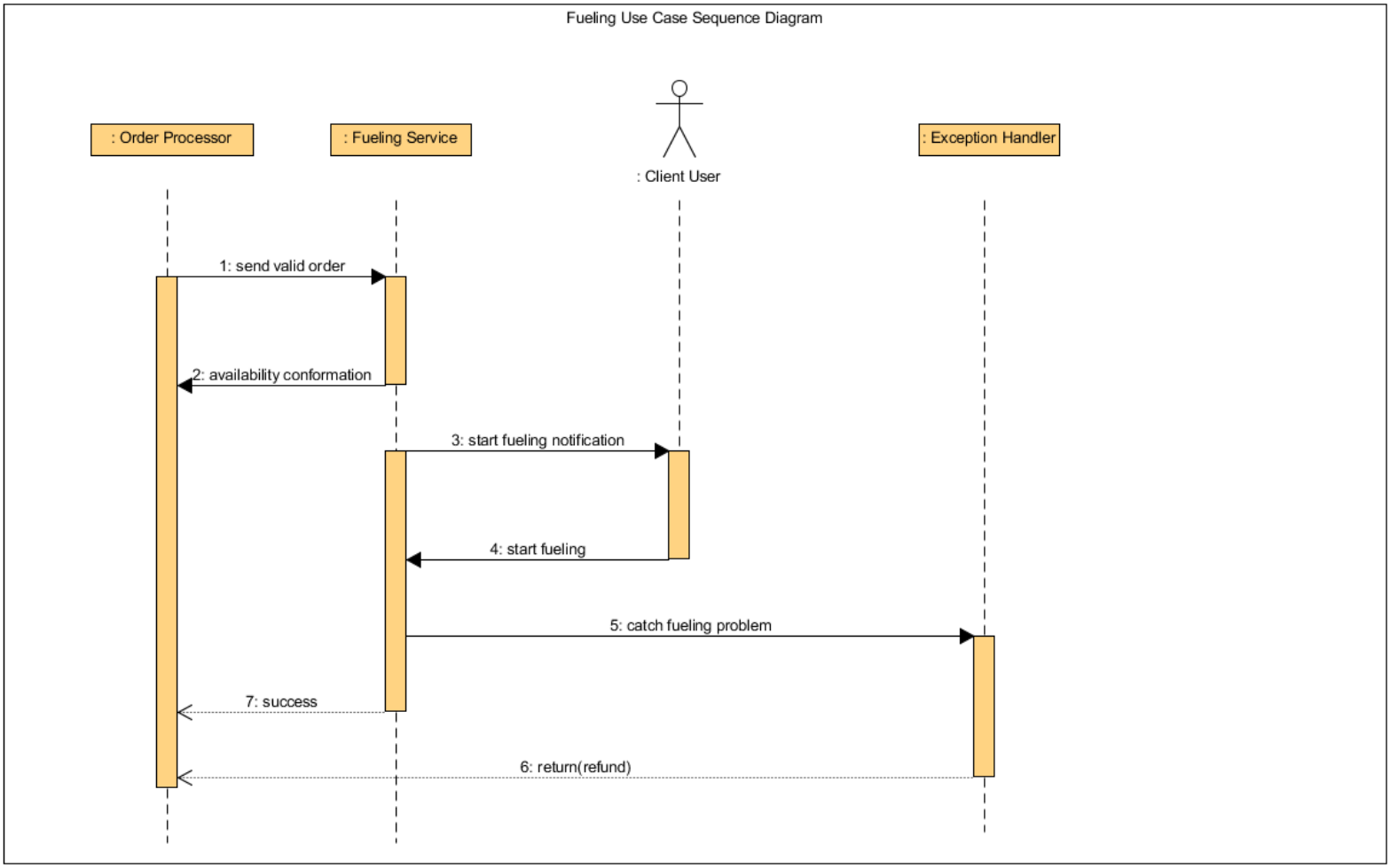


Create a full-dressed scenario for the fueling use case

Use Case:	**Fueling**
Initiating Actor:	**Order Processor**
Actor's Goal:	**Pump out total left amount of gas**
Participating Actors:	**Nozzle, Displayer, Exception Handler**
Preconditions:	**A valid order is created, Fueling Service is available**
Post conditions:	**Total left amount of gas is zero or exit the system with exception.**
Flow of Events for Main Success Scenario:	
——>	**1. Order Processor send a valid order to Fueling Service**
<——	**2. Fueling Service is available**
——>	**3. User picks up the nozzle starts fueling**
<——	**4. If the left amount of gas is zero, stop fueling and print receipt.**
<——	**5. If there is an exception, stop fueling and refund unused money.**

3. Create a sequence diagram for the above scenario (indicate clearly any patterns used and justify briefly their use

Answer:



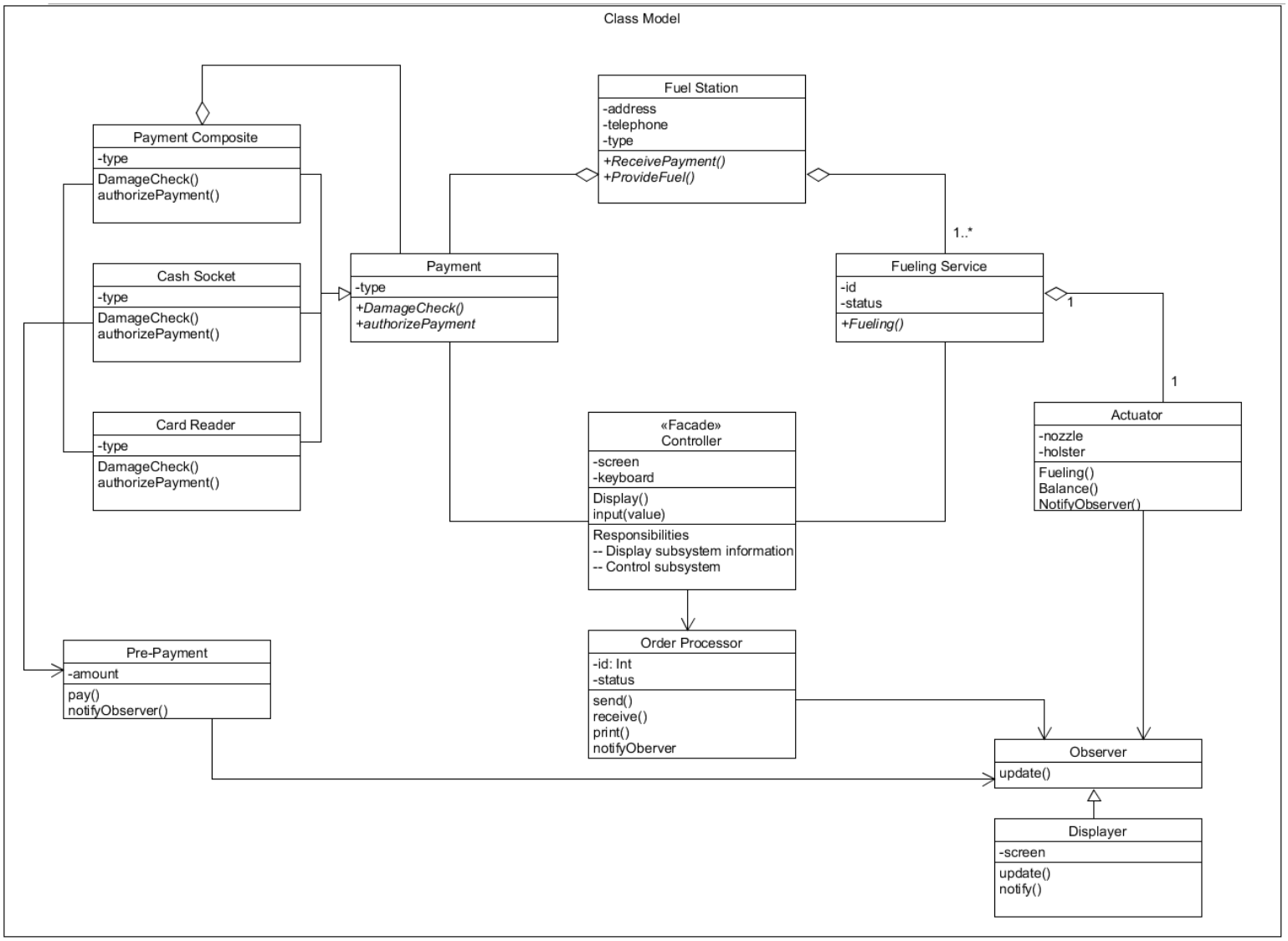
Patterns Used: Composite pattern and Façade pattern.

Because the fueling system can be composed of a whole-part hierarchies, there are relationships "is a", "has a" between container and containee. So it's a composite pattern.

Also the subsystems are decoupled to the users, users have only single entry point fuel tank displayer) to the subsystems. So it's a façade pattern.

4. Provide for the above gas station problem a UML class model. Justify briefly the use of any pattern(s). Please note that the gas station owner has added some additional requirements, which have to be considered when designing the software system. The system should support in the future a more flexible pre-payment option where a user can pay any portion of the pre-paid fueling amount using different credit cards (e.g., Visa, Master), Debit cards, or cash and the remain portion again with any of these payment options.

Answer:

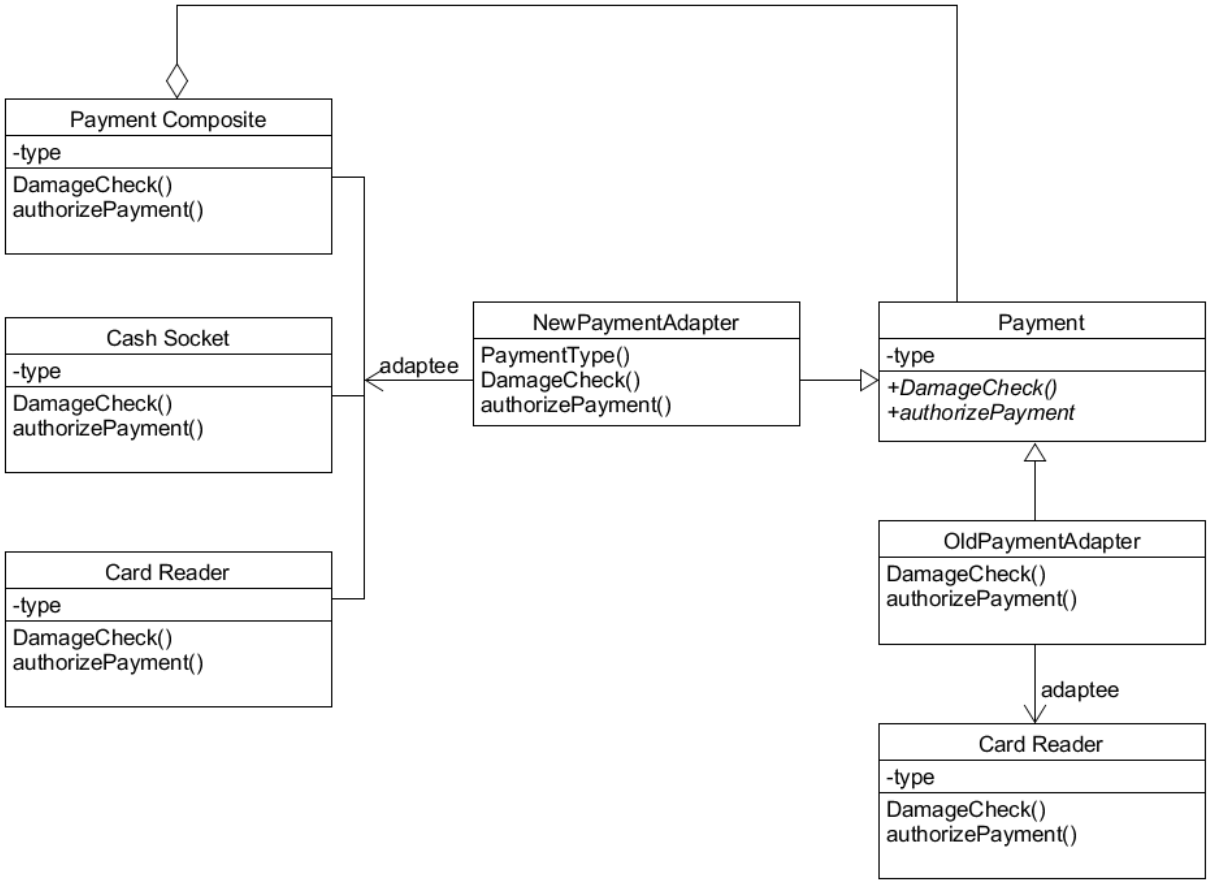


Used patterns: Composite pattern, Façade pattern, Observer Pattern

Composite pattern is used for payment, because there will be more options added and this can be presented as a whole-part hierarchy.
Façade pattern is used to control order processor, because we need to simplify the interaction between user and subsystems.
Observer pattern is used to display information from subsystems.

5. In order to add an additional new requirement to your system, the design not only should be able to connect to the current card reader but also be able to take advantage of an old card reader as a backup reader. Provide a UML class model (based on the previous class model) – **draw only those parts being affected by this change**. Justify briefly the use of any pattern(s) for the changed part.

Answer:



Changed Pattern: Adapter Pattern.

By using adapter pattern we can make both old payment system and new payment system compatible, with the least cost.

Question #4

The designer of an adventure game wants a player to be able take (and drop) various items found in the rooms of the game. All items can be sold or traded by the player. Two of the items found in the game are bags and boxes. Both bags and boxes can contain individual items (money, tokens and food) as well as other bags and boxes. Bags and boxes can be opened and closed and items can be consumed.

a. Which GoF design pattern might be a good choice for the above problem? Justify briefly your decision.

Answer: Composite pattern might be a good choice.
Because the bags and boxes can contain themselves or individual items, it's a perfect example of part-whole hierarchy. By using such pattern it would be easier and simpler to manage the items.

b. Please provide a UML class model for the above file system problem and based on your answer in a.). Include operations based on the problem description

Answer: