Behavioral Design Patterns

Vending Machine

State and Chain of Responsibility

Design a vending machine that dispenses different types of snacks based on:

1) User selection of snack item
2) The amount of money inserted
3) Availability of snacks in the Vending Machine

You should create a class to represent the state of the vending machine such as "Idle", "Waiting for Money", "Dispensing Snack", etc. Additionally, you will use the chain of responsibility design pattern to handle the requests for dispensing different types of snacks. A snack should only be dispensed if there is enough quantity available and enough money has been inserted by the user.

Your implementation should include the following class's:

Snack– Represent all types of snacks that can be dispensed by the vending machine. Each Snack should have a name, price, and quantity.

VendingMachine - Should reference state of Vending Machine and SnackDispenser plus hold all snacks.

StateOfVendingMachine- Represents the different states of the vending machine. What are all the things vending machine can do?

SnackDispenseHandler- Represents the chain of responsibility for dispensing different types of snacks. Handlers should have a reference to the next handler. Cycle through snacks until the proper one is dispensed.

Vending machine starts in idle state. User should select a snack, selectSnack(), from Vending Machine. Vending machine should wait for user to insert money, insertMoney(). If money inserted is >= price then dispense item if enough quantity or return money. To dispense snack, the VendingMachine class should call the dispenseSnack() method on the current state object. The state object will handle the event and transition to the appropriate state based on the availability of snacks.

Create **driver** with 6 different snacks and Chain of Responsibility in this order ->Coke, Pepsi, Cheetos, Doritos, KitKat, and Snickers.

Your driver should include at least one case where the quantity hits 0 with snickers.

HINT: Draw a graph showing states of Vending Machine. Draw pointed arrows between states showing what must occur for the state to transition, move from one state to another.

UML

Class Diagram

Your assignment is to go back to the **text editor** in which you utilized the flyweight design pattern and create a UML Class Diagram.

NOTE: Use the solution provided to you in the lecture notes.

Object Diagram

Create the UML Object Diagram at two points.

1) After HelloWorldCS5800 is entered but, just before the Character Properties are modified.
2) And another Object Diagram just after the character properties H and W are modified.

Use any UML editor you prefer: PlantUML, LucidCharts, Visio…etc