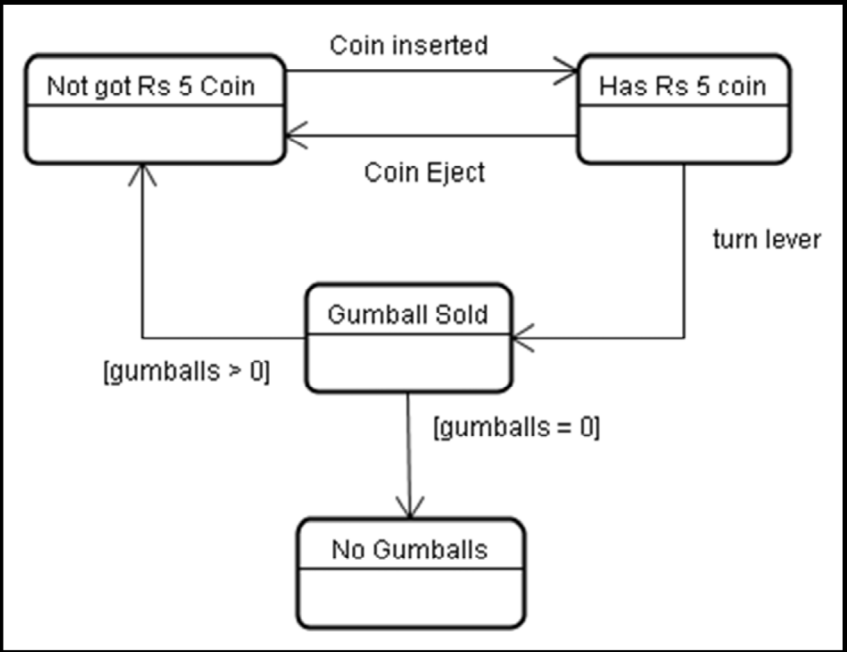
# State DP

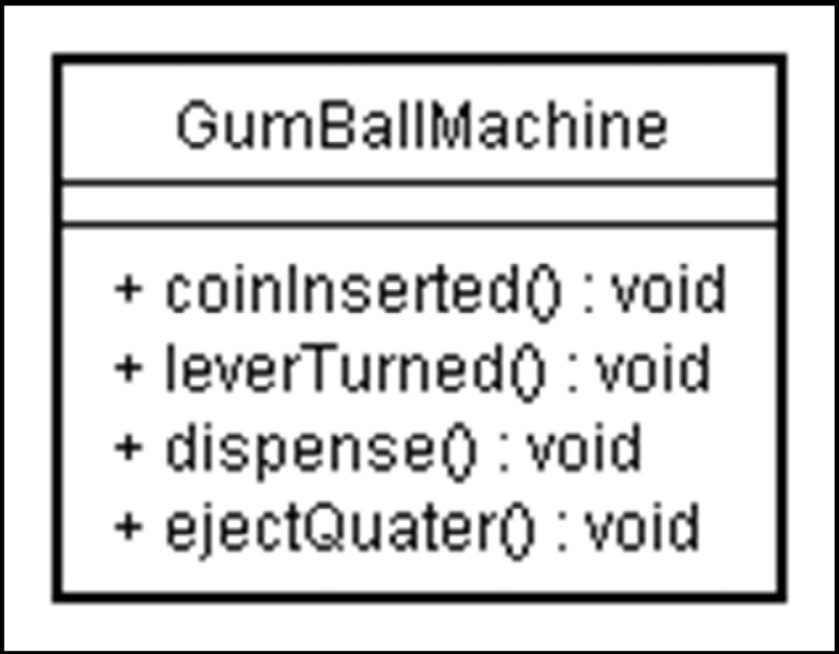
# Problem

- We have a gumball machine.
  - People put a coin in it to get a big ball of chewing-gum.
  - Simulate the function of this machine.

# State diagram

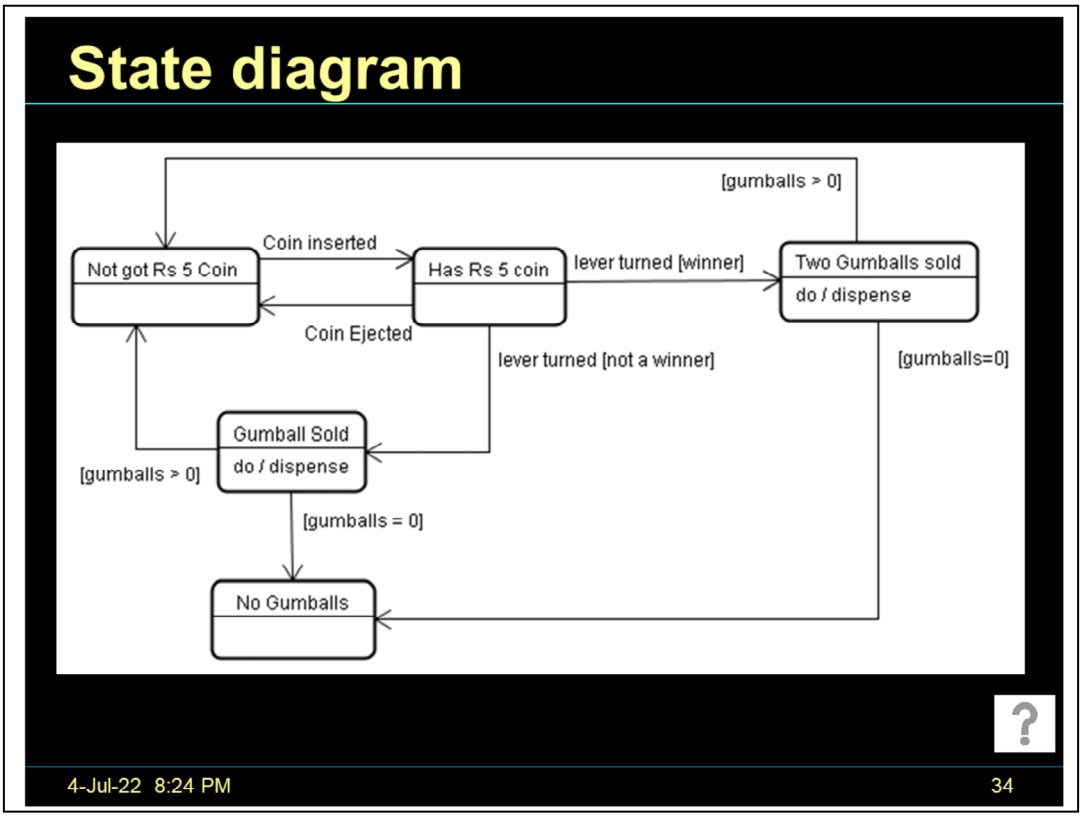


# Code for this Class?



## Problem

- Now, we have a change request.
  - One out of the ten customers get 2 gumballs instead of 1.These 10% customers are the lucky ones.
  - How does the state diagram change?
  - How does the code change?



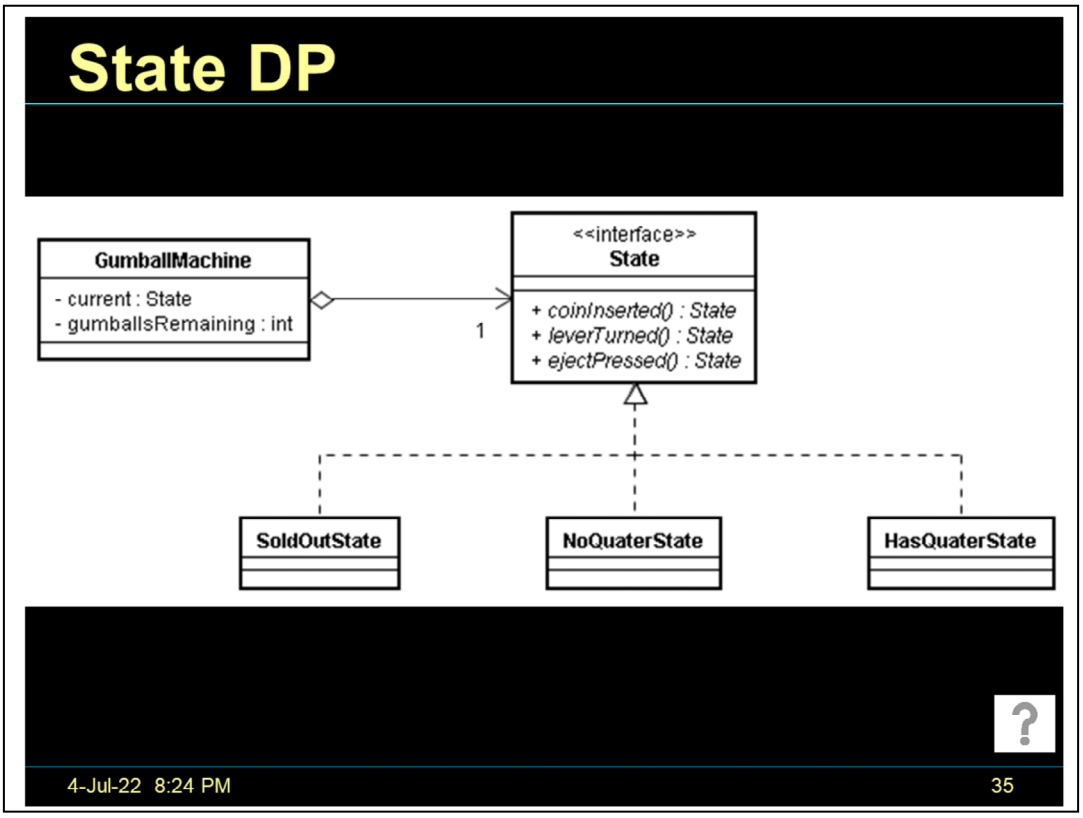

The code does not adhere to Open Closed principle.

State transitions are not explicit. They are buried in the middle of conditional statements.

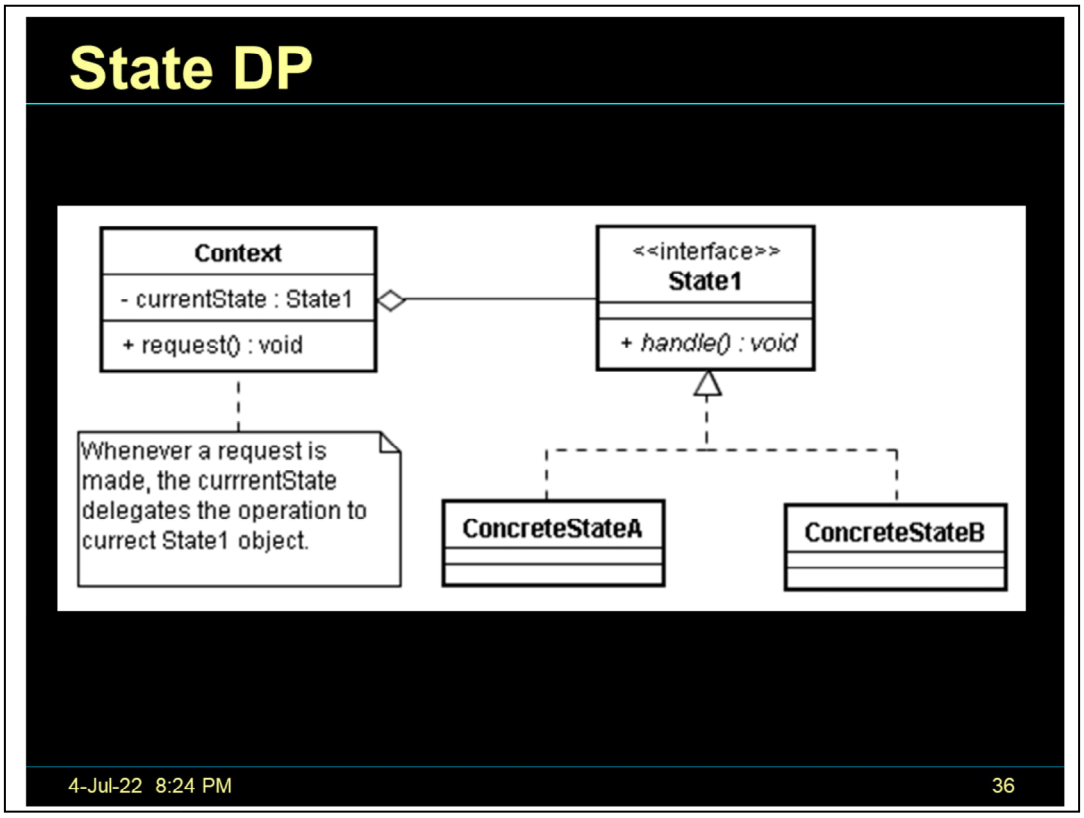
We have not encapsulated anything that varies.

Further changes are probable. They will need existing code to be changed.

The design is not object oriented.




Code example




The context

is of interest to Clients

maintains an instance of a concreteState subclass that defines the current state.

It switches from one state to another during its lifetime, in order to produce different behavior from the same method call(s).

The State

defines an interface for encapsulating the behavior associated with a particular state of the Context.

The ConcreteState subclasses

Each subclass implements a behavior associated with a state of the context.

==========

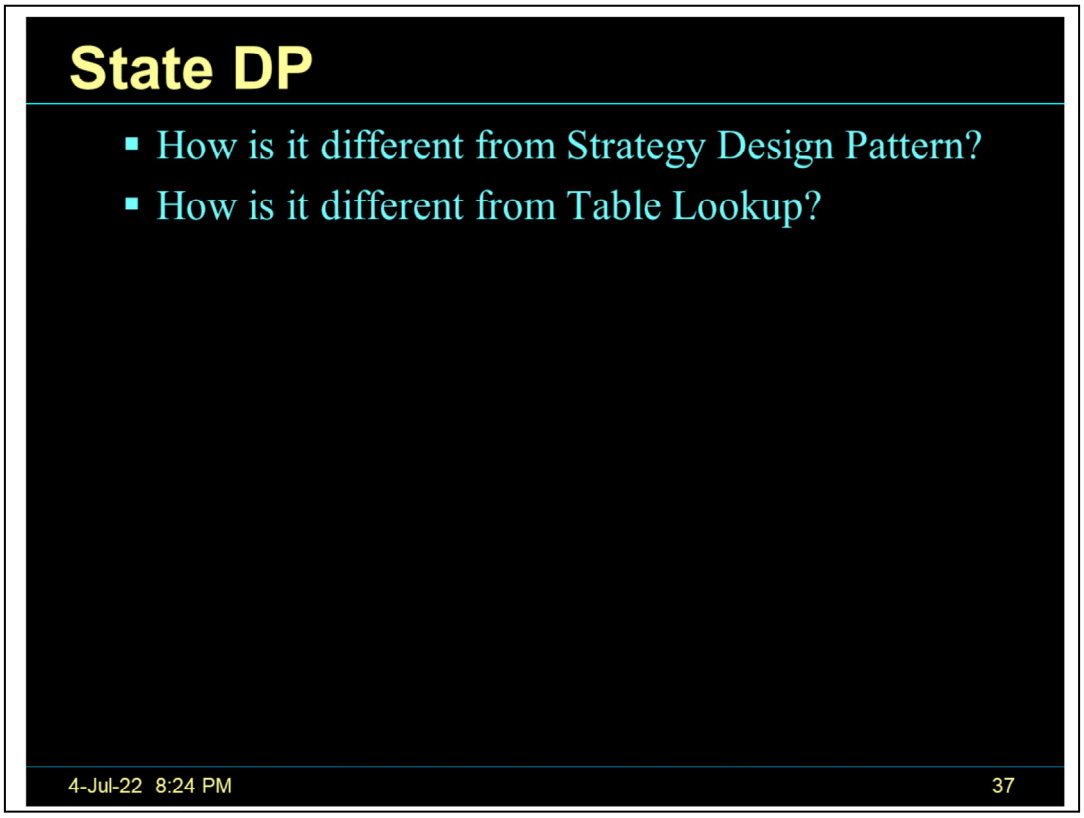
It allows an object to have many different behaviors that are based on its current internal state.

Clients never interact directly with the State objects.

It eliminates the necessity for a set of long, look-alike conditional statements scattered through the program's code.

It makes the state transitions explicit.

Without this DP, some variable value implicitly determines the state of object.




State DP:

The current state of the context changes over time. So the context's behavior also changes.

The client knows little about state objects.

Strategy DP:

The choice of the object is fairly stable but can be changed at runtime.

The client usually specifies the strategy object to be used.

==========

We can use a table to map state transitions.

The table maps every possible input to the succeeding state.

It converts a conditional code / virtual functions into table lookup.

The State DP focuses on state-specific behavior whereas the table-driven approach focuses on defining state transitions.

==========

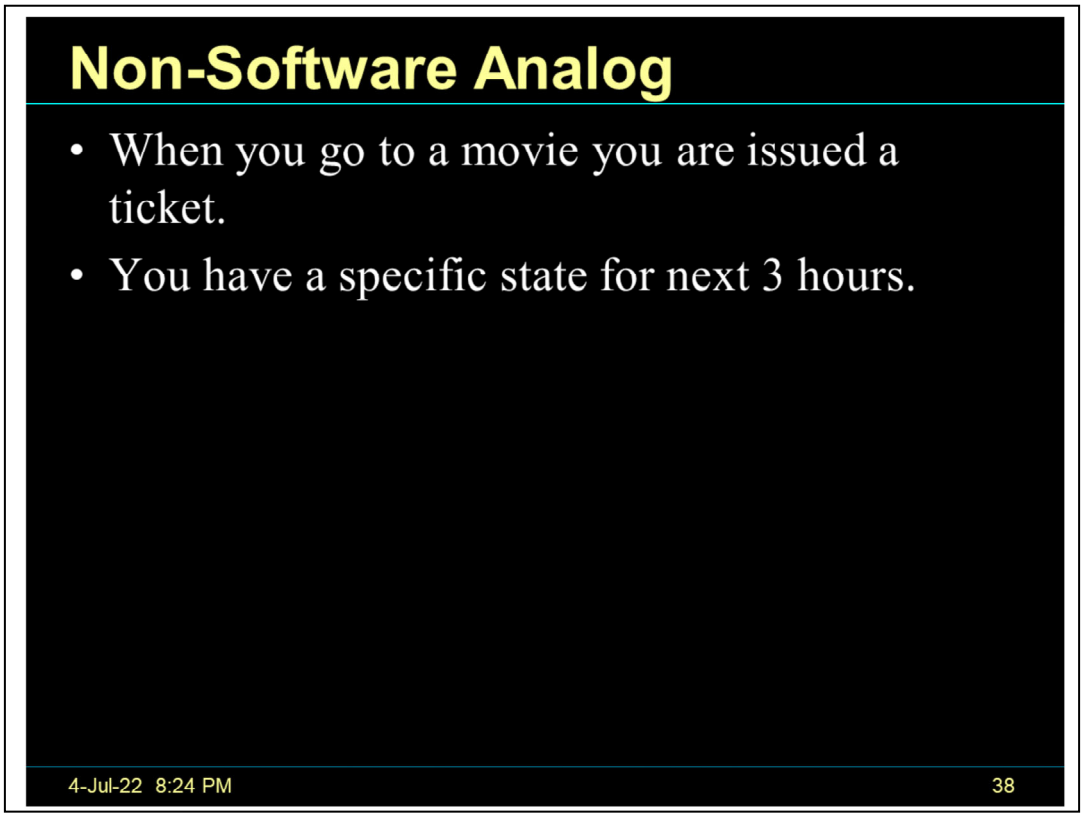
We use the State DP when

An object's behavior depends upon its state and it must change its behavior at runtime depending on that state.

Operations have large, multipart conditional statements that depend on the object's state.

==========

Differences between State and Strategy: The State pattern is useful for a class that must easily transition between instances of a family of state classes, while the Strategy pattern is useful for allowing a class to delegate execution of an algorithm to an instance of a family of Strategy classes. Because of these differences, the motivation and mechanics for refactoring to these two patterns differs

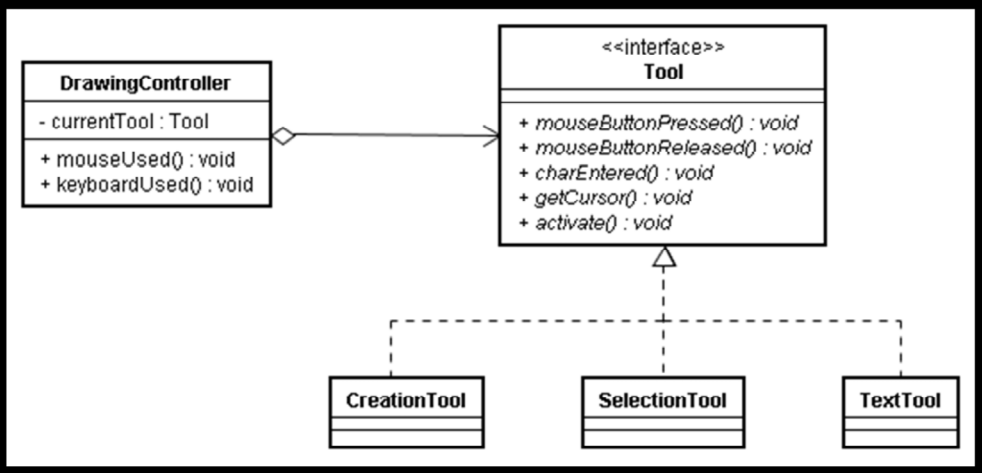



This ticket - or the absence thereof - encapsulates your privileges to see the movie for which you paid. It grants to entry into the theater and in so doing becomes torn. Once you have a torn ticket you are able to enter and leave the theater freely for the duration of the movie. Any time you are inside the theater, you may be subjected to a test: an usher may check your ticket to make sure you are properly authorized for the movie you are viewing.

# Problem

- Interactive drawing programs provide tools like: drawing tool, selection tool, text tool.
  - The editor's behavior changes depending upon the tool selected.
  - How would we use the State DP here?

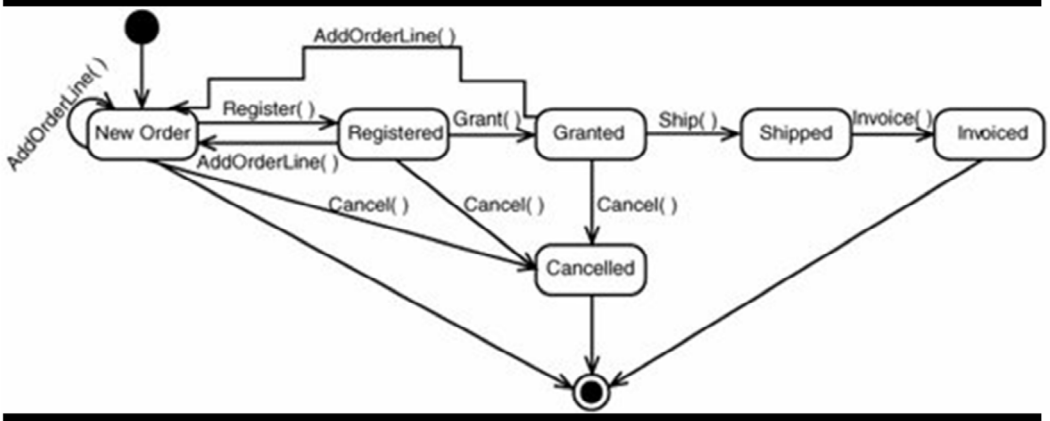
# State DP


DrawingController
- currentTool : Tool
+ mouseUsed() : void
+ keyboardUsed() : void
<<interface>>
Tool
+ mouseButtonPressed() : void
+ mouseButtonReleased() : void
+ charEntered() : void
+ getCursor() : void
+ activate() : void
CreationTool
SelectionTool
TextTool

# Problem

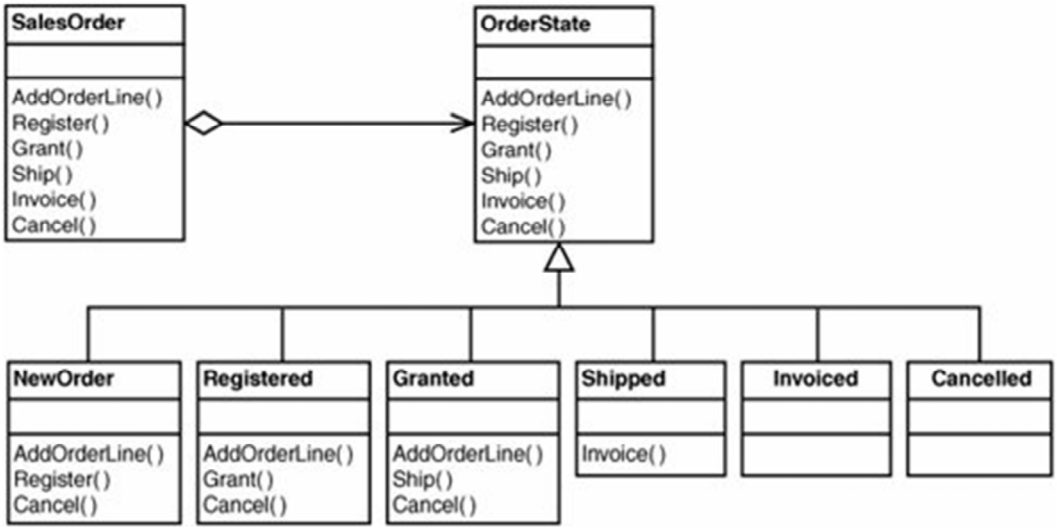
- A sales order can be in different states, such as "NewOrder," "Registered," "Granted," "Shipped," "Invoiced," and "Cancelled."
  - There are strict rules concerning to which states the order can "go" from which states. For example, it's not allowed to go directly from Registered to Shipped.
  - There are also differences in behavior depending upon the states.
    - For example, when Cancelled, you can't call AddOrderLine() for adding more items to the order. (That also goes for Shipped and Invoiced, by the way.)
  - One more thing to remember is that certain behavior will lead to state transformation. For example, when AddOrderLine() is called, the state transforms from Granted back to New Order.

# Draw Class Diagram




?

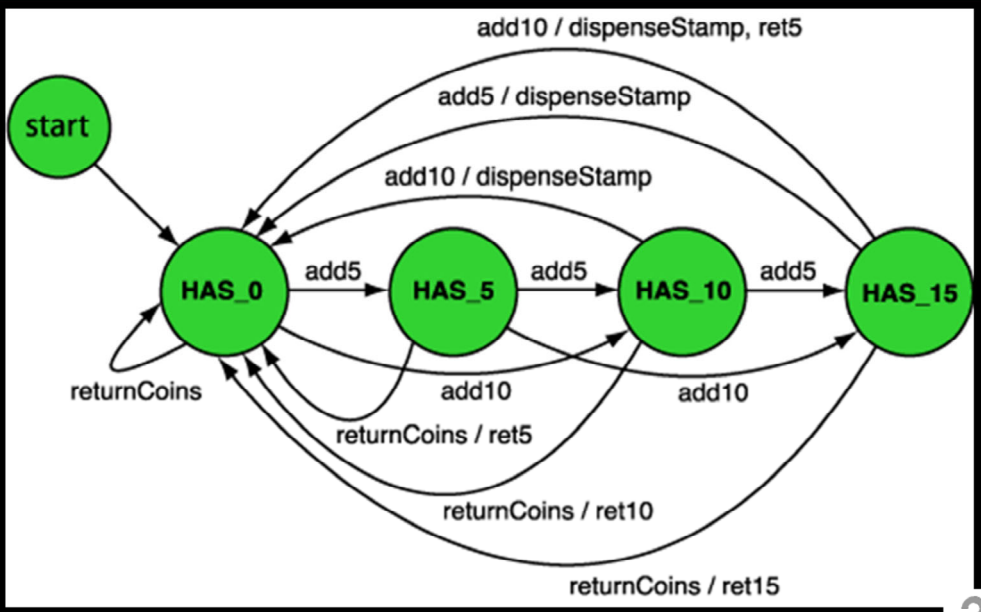
# Solution


SalesOrder
AddOrderLine()
Register()
Grant()
Ship()
Invoice()
Cancel()
OrderState
AddOrderLine()
Register()
Grant()
Ship()
Invoice()
Cancel()
NewOrder
AddOrderLine()
Register()
Cancel()
Registered
AddOrderLine()
Grant()
Cancel()
Granted
AddOrderLine()
Ship()
Cancel()
Shipped
Invoice()
Invoiced
Cancelled

# Stamp Dispenser

- Write control software for an automated stamp dispenser.
  - The stamp dispenser accepts only nickels (5 cents) and dimes (10 cents) and dispenses only 20-cent stamps.
    - The stamp dispenser's LED display indicates to the user the total amount of money inserted so far. As soon as 20 or more cents is inserted, a 20-cent stamp automatically dispenses along with any change. The only amounts the display shows, therefore, are 0, 5, 10, and 15 cents.
    - If a dime and a nickel have been inserted, the display indicates 15 cents. If the user then inserts another dime, the stamp dispenser dispenses a 20-cent stamp, returns a nickel, and changes the display to show 0 cents.
  - In addition to a coin slot, an opening for dispensing stamps and change, and an LED display, the stamp dispenser also has a coin return lever. When the user presses coin return, the stamp dispenser returns the amount of money indicated on the display, and changes the display to show 0 cents.

# State diagram




Draw the Class diagram using State DP

?