SE 3313 Software Design and Architecture

Facade Pattern and State Pattern

Lab 10

1 Facade Pattern

Facade design pattern provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.

2 Problem Description (Facade Pattern)

You are designing a travel management system in Java, and the system includes subsystems for flight booking FlightBookingSystem, hotel reservation HotelReservationSystem, car rental CarRentalSystem, activity booking ActivityBookingSystem, travel insurance InsuranceSystem, and a user notification service NotificationService. The TravelFacade class acts as a facade to simplify interactions with these subsystems.

Your task is to implement the TravelFacade class with the following functionalities:

- Book a flight from a departure city to a destination city on a specified departure date.
- Reserve a hotel in a city for a specified check-in and check-out date.
- Rent a car in a city starting from a specified date for a given number of days.
- Book an activity in a destination city on a specified date.
- Purchase travel insurance for a trip to a destination city from a start date to an end date.
- Notify the user that their trip is booked successfully.

Additionally, write a simple Main class that demonstrates the usage of the TravelFacade by planning a trip with the following output

3 Output (Facade Pattern)

```
Flight booked from Izmir to Istanbul on 2023-12-11
Hotel reserved in Istanbul Hotel from 2023-12-11 to 2023-12-17
Car rented in Istanbul from 2023-12-11 for 6 days
Activity booked in Istanbul: Sightseeing Tour on 2023-12-15
Insurance purchased for the trip to Istanbul from 2023-12-11 to 2023-12-17
Notification: Your trip is booked successfully!
```

Figure 1:

4 State Pattern

The state pattern is a behavioral software design pattern that allows an object to alter its behavior when its internal state changes. This pattern is close to the concept of finite-state machines. The state pattern can be interpreted as a strategy pattern, which is able to switch a strategy through invocations of methods defined in the pattern's interface.

5 Problem Description (State Pattern)

Your task is to implement an elevator system using the State Pattern in Java. The elevator has three states: StationaryState, MovingUpState, and MovingDownState. The elevator should transition between these states based on button presses. Your goal is to implement the Elevator, ElevatorState, and concrete state classes.

- ElevatorState Interface: Define an interface ElevatorState with a method pressButton(int destinationFloor)
- Concrete States: Implement concrete classes StationaryState, MovingUpState, and MovingDownState that represent the elevator's behavior in different states.
- Elevator Class: Implement the Elevator class as the context for the state pattern. It should have methods to set the current state, get and set the current floor, and handle button presses by transitioning between states.
- Client Code: In the client code (ElevatorSystemExample), instantiate the Elevator class and demonstrate the elevator's behavior by pressing buttons for different floors.

Additionally, write a simple Main class that demonstrates the usage of the State Pattern with the following output (Assume that elevator is at StationaryState at floor 0.)

6 Output (State Pattern)

```
Elevator is stationary. Moving to floor 3
Elevator is moving up to floor 5
Elevator is moving down to floor 2
```

Figure 2:

7 Measure of Success

Your task is to implement the above patterns as desired. Please zip separate project folders then you can upload the zipped file.