**Structural Design Patterns**

**E-Commerce and Restaurants**

**Bridge**

**Part 1**) You are working for a small e-commerce company. You want to create a payment notification system when customers make a payment. Notifications can be sent through different channels (Email, SMS, or Push messages). Additionally, customers can make payments online(credit card or paypal) or cash on delivery each requiring a different message.

You must design and build this using the **Bridge design pattern**, so that:

* New payment types can be added without changing notification channels (**EmailChannel**, **SmsChannel**, **PushChannel**)
* New channels can be added without changing payment types (**OnlinePaymentNotification**, **CashOnDeliveryPayment, BitcoinPayment**)

Create a driver program where you show different scenarios such as online and email, online and sms, cash and email and cash and sms.

**Part 2**) Add a new payment option BitcoinPayment and add two scenarios: one with email and one with sms to the driver program

**Part 3**) Add a new notification channel PushChannel and and two scenarios with pushChannel

**Decorator**

You are building a software system for a restaurant that needs to keep track of customer orders. Customers can order various food items, such as burgers, fries, and hot dogs. Each food item has a base price and can have additional toppings or add-ons, which increase the price of the item.

Toppings: Ketchup, Cheese, Onions…etc

Your task is to design a system that allows the restaurant to:

1. Create new food items with different base prices and toppings.
2. Calculate the total cost of a customer's order, including the cost of each item and any toppings.
3. Apply discounts to the total order cost based on the customer's loyalty status.

To accomplish this task, use the decorator design pattern to create a hierarchy of classes that represent the different food items and toppings. Each class should implement a common interface that allows the system to calculate the cost of the item.

Then, create a class that represents the customer's order. This class should have a list of food items and toppings, and it should be able to calculate the total cost of the order by iterating over the list and summing up the cost of each item.

Finally, create a class that represents the customer's loyalty status. This class should have a method that applies a discount to the total order cost based on the customer's status.

Implement this system using the decorator design pattern, and test it by creating some food items, adding toppings, creating an order, and applying a loyalty discount in the driver program.