Structural Design Patterns

Social Media and Restaurants

Bridge

You are working on a software system that allows users to send messages using different messaging apps such as WhatsApp, Telegram, and Facebook Messenger. The system needs to be flexible enough to accommodate new messaging apps in the future. Additionally, the system should be able to support different message types such as text, image, and video.

Design a messaging system that uses the Bridge design pattern to handle the different messaging apps and message types.

To solve this problem, you can use the Bridge design pattern to decouple the messaging app implementations from the message type implementations. This will allow you to add new messaging apps and message types independently without affecting the other components of the system.

You can start by creating an abstraction for the messaging app interface and an abstraction for the message type interface. Then, you can create concrete implementations of the messaging app interface for each messaging app (e.g., WhatsAppMessagingApp, TelegramMessagingApp, FacebookMessagingApp) and concrete implementations of the message type interface for each message type (e.g., TextMessage, ImageMessage, VideoMessage).

Finally, you can use the Bridge pattern to link the messaging app and message type abstractions together, allowing the messaging system to support different combinations of messaging apps and message types. This will enable the system to handle different scenarios such as sending a text message via WhatsApp or sending a video message via Telegram.

To help you get started here is the interface for the messageType. There should be three: Text, Image and Video

```
public interface MessageType {
    public void setMessageApp(MessagingApp messagingApp);
    public void setContent(Media content);
    public String getMediaType();
    <MediaType> MediaType getContent();
    public void sendMessage();
}
```

NOTE: You do not have to send actual images or video's just print out "Image sent" and "Video sent"

Create a driver program where you send multiple message types in different messaging Apps.

Decorator

Suppose 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.

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.