SE 3313 Software Design and Architecture

Strategy Pattern

Lab 2

1 Objective

In this lab experiment, we will be exploring the Strategy design pattern in Java. This pattern is a behavioral design pattern that allows us to define a family of algorithms, encapsulate each one of them, and make them interchangeable. This enables the algorithm to vary independently from the context that uses it.

2 Background

Before we dive into the code, let's understand the concept of the Strategy pattern. Imagine a scenario where a company needs to handle different strategies for delivering packages. Some packages may have standard delivery with a fixed price, while others might require express delivery with variable pricing. The Strategy pattern helps us switch between these strategies at runtime.

3 Components of the Experiment:

- DeliveryStrategy Interface:
- Methods
  - printDeliveryType(): Displays the type of delivery.
  - calculateDeliveryTime(): Calculates the estimated delivery time.
- PricingStrategy Interface:
  - Defines the method for calculating the delivery price - calculateDeliveryPrice() .
- Concrete Delivery Strategies:
  - StandardDelivery: Implements DeliveryStrategy for standard deliveries.
  - ExpressDelivery: Implements DeliveryStrategy for express deliveries.
- Concrete Pricing Strategies:
  - FlatRatePricing: Implements PricingStrategy with a flat rate pricing model.
  - VariableRatePricing: Implements PricingStrategy with variable rate pricing.

- CargoCompany Class:
  - Acts as the context class that uses both a delivery strategy and a pricing strategy.
  - Allows us to set different strategies.
- Subclasses of CargoCompany:
  - CargoCompA, CargoCompB, and CargoCompC.
  - Inherit from CargoCompany and set default strategies for their specific scenarios.

4 Your Task

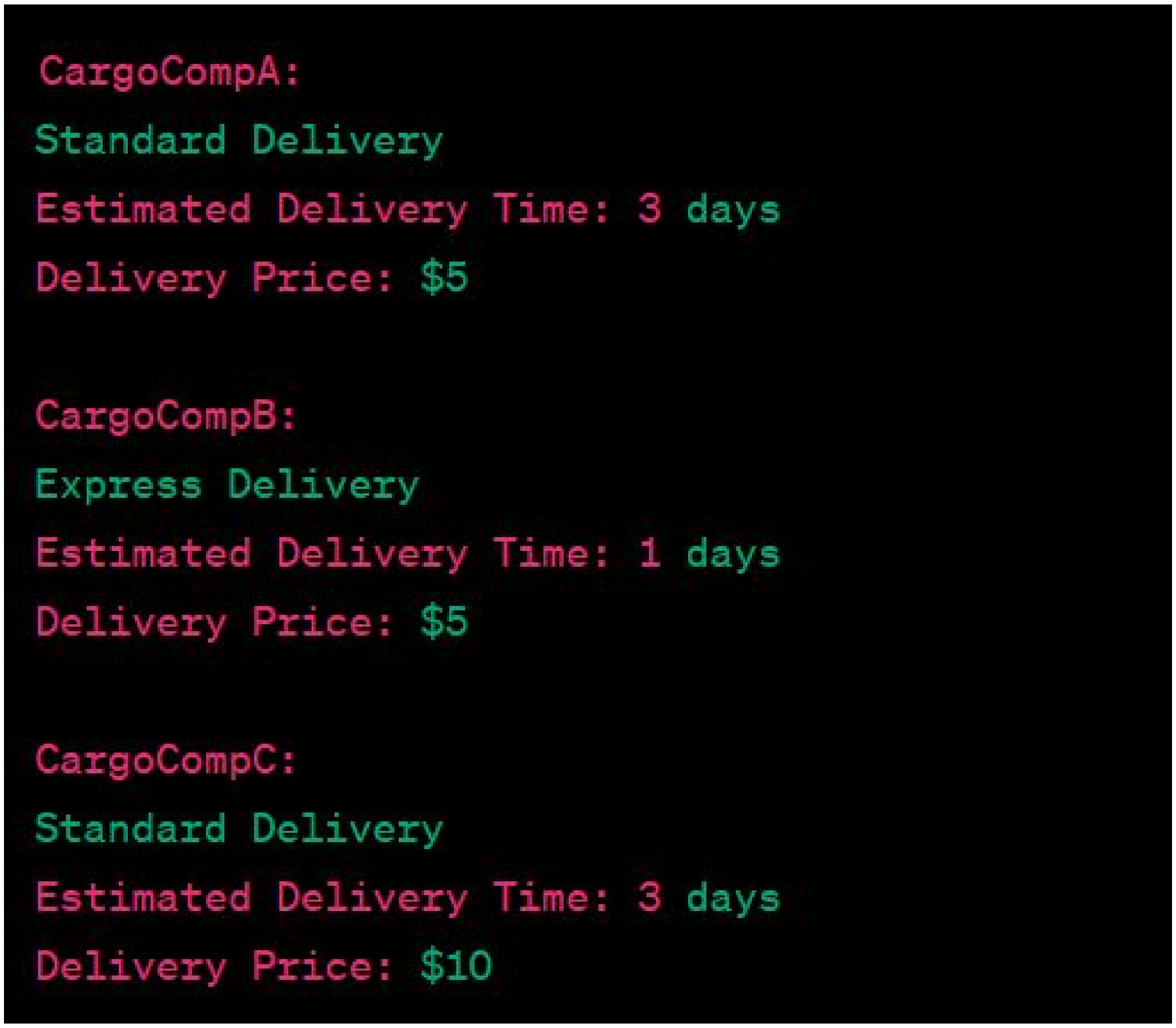
- Experiment with Different Strategies:
  - Create instances of CargoCompA, CargoCompB, and CargoCompC.
  - Observe how changing the default strategies in each subclass affects the output.
  - If you want to further explore, try creating your own custom delivery and pricing strategies.

5 Conclusion

This lab experiment provides hands-on experience with the Strategy pattern, allowing you to dynamically switch between different algorithms at runtime. This flexibility is invaluable in real-world applications where behaviors may need to change based on different contexts or requirements.

Enjoy the experiment, and feel free to reach out if you have any questions!

6 Output



```
CargoCompA:
Standard Delivery
Estimated Delivery Time: 3 days
Delivery Price: $5

CargoCompB:
Express Delivery
Estimated Delivery Time: 1 days
Delivery Price: $5

CargoCompC:
Standard Delivery
Estimated Delivery Time: 3 days
Delivery Price: $10
```

Figure 1:

In this example, we have three instances: CargoCompA, CargoCompB, and CargoCompC, each with its own default delivery and pricing strategies.

- CargoCompA uses Standard Delivery and Flat Rate Pricing.
- CargoCompB uses Express Delivery and Flat Rate Pricing.
- CargoCompC uses Standard Delivery and Variable Rate Pricing.

As a result, the output displays the type of delivery, estimated delivery time, and the associated delivery price for each instance. This demonstrates how different strategies can be applied dynamically to achieve varying outcomes.