Food Ordering and Delivery System

Heer Soni, Joey Myers, Sindhur Bansal, Dylan Drotts

Project Overview

- Java-based food ordering and delivery system
- Users customize pizzas, salads, drinks
- Delivery options: Bike, Car, Drone
- Order tracking from Confirmed to Delivered
- Java Swing GUI implemented

Design Patterns Used

- **Command Pattern**: Manage, place, cancel, reorder, undo
- **Strategy Pattern**: Flexible delivery options
- **Decorator Pattern**: Customizable food items
- **Factory Pattern**: Food creation and decoration
- **State Pattern** (Bonus): Order lifecycle management

Factory Pattern

- **Purpose**: Centralized creation of base food items and their customizations.
- **Key Class**: FoodFactory

How It Works:

- Parses user input string (e.g., "Pizza Flat Bacon ExtraCheese")
- Creates base item (Pizza, Salad, Drink)
- Applies appropriate decorators (toppings, sizes)

Example:

- Input: "Drink Coke Large No Ice"
- Output: Coke Large No Ice ($2.75)

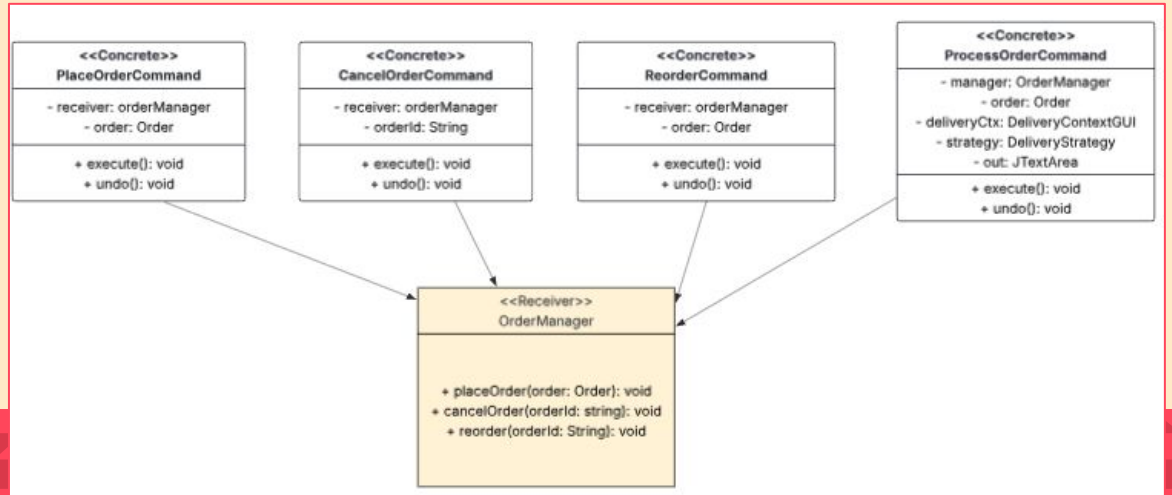
Decorator Pattern

- Wraps each food item in various toppings/modifications
- PizzaDecorator, SaladDecorator, DrinkDecorator classes inherit from Pizza, Salad, and Drink respectively
- Each wrapped object potentially adds an upcharge and adds on to the item’s description.
- Combines inheritance and composition. E.g. types of pizza/PizzaDecorator inherit from pizza, toppings inherit from PizzaDecorator, and each topping wraps a Pizza.

Command Pattern OverView

Command Pattern: Order Actions as Commands:

- Encapsulates actions like **Place**, **Cancel**, **Reorder**, and **Undo** into objects
- Promotes **loose coupling** between UI and backend logic
- Enables an **undo** system using a command history (Stack)
- Easy to extend (e.g., future: DeliverOrderCommand)



How it works in our project

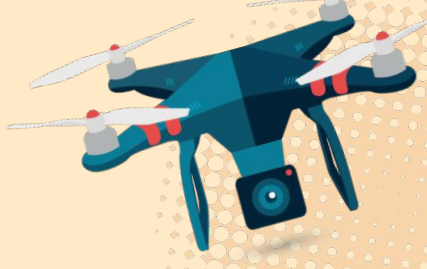
🎮 How We Use Command Pattern

- Each button in the **GUI** creates and sends a command to the OrderInvoker
- The invoker stores command history for **undo** functionality
- Example:
  - Place Order -> PlaceOrderCommand.execute()
  - Undo -> commandHistory.pop().undo()

System Architecture

- **Food Factory ->** Builds items (Pizza, Salad, Drink)
- **Decorators ->** Add toppings, size, modifications
- **Command Invoker ->** Executes and undoes actions
- **Delivery Context ->** Executes selected delivery strategy
- **Order State ->** Manages status progression

Delivery Segment

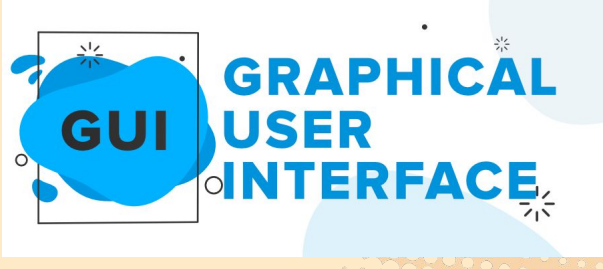


1. Uses Strategy Pattern to use Delivery Content to set the desired delivery strategy.
2. Every Strategy: Sets the desired route, estimates time, calculates cost of delivery, and finally delivers
3. Options:
   a. Bike Delivery: Low cost, longer ETA
   b. Car Delivery: Balanced cost and speed
   c. Drone Delivery: Fast but expensive
4. Easily extendable for newer delivery options, Runtime flexibility and clean separation of delivery logic

Order Lifecycle

1. Manages the lifecycle of an order using State pattern. Each state: defines its name and sets the next state, except the Delivery State which is terminal.
2. States:
   a. Confirmed State -> Picking Up State
   b. Picking-Up State -> EnRoute State
   c. EnRoute State -> Delivered State
3. Clear and maintainable state transitions
4. Avoids complex conditional logic
5. Easy to extend states and add more complex logic in each state

GUI



- Allows User to specify exact specification of the food they want to order.
- User sets their own drop off address.
- A dropdown menu can be used to select the delivery method which will use strategy pattern to help select the appropriate method.
- Menu option shows all possible combinations for our app
- Order button carries out the query and Undo last undos the last action.
- A text box under shows the details, status of the order.

Challenges Faced

- Complex interaction between multiple design patterns
- Ensuring loose coupling and maintainability
- Implementing undo for composite operations
- GUI integration with backend logic

Conclusion

- Met project goals using five design patterns
- GUI bonus achieved
- Built scalable, flexible architecture
- Strengthened understanding of design principles

Thank you!

Questions?