Food Ordering App - Domain and Class Design

Part 1: Understanding the Domain

1. Main entities in a food ordering app:

- FoodItem: Represents each food item available for purchase.

- Cart: Holds all selected food items the user wants to buy.

- Payment: Handles different payment methods (Cash, Card, UPI).

- Order: Represents a confirmed order, potentially used to store history.

- User (optional): Represents the customer, useful for login, order history, etc.

2. Entities grouped using inheritance:

- All types of food like Pizza, Burger, and Fries can inherit from a base class FoodItem.

- The base class defines common properties such as name and price.

- Subclasses add specific behavior or characteristics unique to each food type.

3. Private data and how to access it:

- Encapsulation restricts access to internal data like name, price, quantity, cart item list, and payment details.

- Use getters and setters (e.g., getName(), getPrice()) to access private data.

Part 2: Class Design

4. Generic FoodItem class:

- Properties: String name, double price

- Should be abstract as it represents a concept, not a concrete item.

- Abstract method: displayInfo()

5. Specific food item classes (Pizza, Burger, Fries):

- Inherit from FoodItem

- Pizza: Attributes like size and toppings

- Burger: Options like patty type, extra cheese

- Fries: Size-based pricing

- Override displayInfo() to show relevant details

6. Cart class:

- Internally uses ArrayList<FoodItem>

- Key methods: addItem(FoodItem), removeItem(int), displayCart(), getTotalPrice()

- Item list is private and accessed only via methods

Part 3: Behavior & Logic

7. Multiple payment options using interface:

- Create Payment interface with method void pay(double amount)

- Implement in: CashPayment, CardPayment, UPIPayment

8. Polymorphism in payment:

- Each payment class implements pay() differently

- Cash: prompts "Please pay at delivery"

- Card: collects card number and validates

- UPI: asks for UPI ID and confirms

9. MainApp logic:

- Console-based interface

- Menu options: View items, Add to cart, View cart, Choose payment, Confirm order

- Performs actions based on user input

10. Input validations:

- Handle invalid menu selections, negative quantities, invalid indexes

- Ensure valid payment selection

- Catch exceptions (e.g., InputMismatchException)

Part 4: Extension Ideas

11. Choose quantities for items:

- Modify FoodItem to include int quantity

- Prompt user for quantity on add

- Total = price * quantity

12. Apply tax or discounts:

- Cart class methods: applyTax(double), applyDiscount(double)

- Final = subtotal + tax - discount

13. Save order history to file:

- Use FileWriter, BufferedWriter, or PrintWriter

- Store item names, quantities, prices, total, payment method, timestamp

14. GUI version with JavaFX:

Food Ordering App - Domain and Class Design

- Use JavaFX components: ListView, ComboBox, TextField, TableView, Button

- Use Scene and Stage classes for windows