# Java OOP Task Project – Online Food Ordering System

📁 Package: com.foodorder

🎯 Difficulty: Beginner–Intermediate

🧑‍🎓 Ideal For: Learning OOP concepts via real-world application

## 🔥 Project Scenario:

You are designing a simple online food ordering system. A user will interact through a console menu to:  
1. View food options  
2. Add food items to their cart  
3. View the total price  
4. Choose a payment method  
5. Complete the order  
  
Your goal is to use Java OOP concepts to structure the project cleanly and logically.

## 📌 Core Java Concepts to Apply:

|  |  |
| --- | --- |
| Concept | What to Focus On |
| Encapsulation | Hide internal details (like cart contents, prices, etc.) |
| Inheritance | Share common features among different food items |
| Abstraction | Use abstract classes or interfaces for generic behaviors |
| Polymorphism | Let the same method behave differently for different types |

## 📋 Questions & Tasks for Students

### 🟠 Part 1: Understanding the Domain

1. 1. What are the main entities in a food ordering app (e.g., food item, cart, payment)?
2. 2. Which entities can be grouped using inheritance (e.g., all food items)?
3. 3. What data should be private, and how will you access it?

### 🟡 Part 2: Class Design

1. 4. Create a class for a generic food item: What common properties do all food items have? Should this class be abstract? Why?
2. 5. Create 2–3 specific food item classes (e.g., Pizza, Burger, Fries). What makes each of these different? How will they override behavior?
3. 6. How would you build a Cart class that stores multiple food items? What methods should it have? Should the internal list be directly accessible?

### 🟢 Part 3: Behavior & Logic

1. 7. You want to offer multiple payment options (Cash, Card, UPI). How can you use an interface for payment?
2. 8. How will each payment method handle the payment differently using polymorphism?
3. 9. How will the MainApp allow users to: view menu, add items, choose payment, and complete order?
4. 10. What user input validations do you need (e.g., wrong choices, negative inputs)?

### 🔵 Part 4: Extension Ideas

1. 11. How would you allow users to choose quantities for each item?
2. 12. How would you apply tax or discounts to the final bill?
3. 13. How could you save the order history to a file?
4. 14. How would you build a GUI version later using JavaFX?

## 🎯 Final Output Expectations

- A working Java console app  
- Uses all 4 pillars of OOP correctly  
- Logical menu-based flow  
- Clean code using meaningful class and method names  
- Bonus features if attempted (tax, quantity, save file, etc.)

## 📝 Submission Instructions

- Project folder name: YourName\_OnlineFoodOrder  
- Include all .java files inside a src/com/foodorder/ package structure  
- Add a short README file explaining how to run it  
- Include 1–2 screenshots of your working program  
- Deadline: [Set by teacher]