

Mini Food Ordering Console App - Project Documentation (Beta)

Project Overview

The **Mini Food Ordering Console App** is a Java-based application that allows two types of users:

- **Admin:** To manage food menus, discounts, and delivery partners
- **Customer:** To browse menus, place orders, make payments, and receive an invoice

This project follows **SOLID principles** and demonstrates strong **object-oriented design** using Java **interfaces, abstractions, and encapsulation**.

Interfaces and Their Roles

| Interface | Purpose |
|--------------------------------|---|
| <code>ICheckUser</code> | Used for user authentication (<code>Admin</code> and <code>Customer</code>) |
| <code>IMenuManager</code> | Manages creation, deletion of menus and menu items |
| <code>IMenuViewer</code> | Allows viewing of available menus |
| <code>IOrderManager</code> | Handles cart operations like add/remove items and calculate total |
| <code>IDiscountStrategy</code> | Allows dynamic discount policies (Percentage based) |
| <code>IPaymentService</code> | Allows multiple payment strategies (Cash/UPI) |
| <code>IDeliveryService</code> | Manages and assigns delivery partners |
| <code>IInvoiceGenerator</code> | Handles invoice generation with proper formatting |

Project Folder Structure

```
com.aurionpro
├── 🗂️ delivery
│   ├── 📄 DeliveryPartner.java
│   ├── 📄 DeliveryService.java
│   └── 📄 IDeliveryService.java
├── 🗂️ discount
```

```

|   | } 🚫 IDiscountStrategy.java
|   | L 🚫 PercentageDiscount.java
|   |
|   | 🗝️ food
|   | | } 🚫 Menu.java
|   | | } 🚫 MenuItem.java
|   | | } 🚫 MenuManager.java
|   | | } 🚫 IMenuManager.java
|   | | L 🚫 IMenuViewer.java
|   |
|   | 🗝️ invoice
|   | | } 🚫 InvoiceGenerator.java
|   | | L 🚫 IInvoiceGenerator.java
|   |
|   | 🗝️ orders
|   | | } 🚫 CartItem.java
|   | | } 🚫 OrderManager.java
|   | | L 🚫 IOrderManager.java
|   |
|   | 🗝️ payments
|   | | } 🚫 IPaymentService.java
|   | | } 🚫 CashPayment.java
|   | | L 🚫 UpiPayment.java
|   |
|   | 🗝️ users
|   | | } 🚫 Admin.java
|   | | } 🚫 Customer.java
|   | | L 🚫 ICheckUser.java
|   |
|   | 🗝️ tests
|   | | L 🚫 FoodOrderingDriver.java

```

SOLID Principles Applied

| Principle | Application in the Project |
|----------------------------------|---|
| S - Single Responsibility | Each class handles one specific task only (e.g., <code>MenuManager</code> for menu logic, <code>InvoiceGenerator</code> for billing). |
| O - Open/Closed | New payment modes or discounts can be added using interfaces without changing existing logic. |
| L - Liskov Substitution | You can use <code>CashPayment</code> or <code>UpiPayment</code> wherever <code>IPaymentService</code> is expected. |
| I - Interface Segregation | Interfaces are small and specific (e.g., <code>IMenuViewer</code> , <code>IDeliveryService</code>). |
| D - Dependency Inversion | High-level classes depend on abstractions, not on concrete implementations. |

Java Concepts Used

- **Encapsulation** via private fields and public methods
- **Abstraction** through interface-based design
- **Polymorphism** via dynamic dispatch using interfaces
- **Collections Framework** (`List` , `Map`)
- **Scanner Input** for console interaction
- **Formatted Output** with `System.out.printf`
- **Exception Handling** for robust user inputs
- **Enums and Booleans** to handle course types and Veg/Non-Veg items

Features Explained

Admin



- Add/Delete **menus**
- Add/Delete **menu items** (categorized by course type + veg/non-veg)
- View menu structure
- Add/Delete **delivery partners**

Customer

- View **all menus**
- Add/Delete items to **cart**
- Calculate **total + discount**
- Make payment (Cash/UPI)
- Get assigned delivery partner
- **Generate a beautiful invoice**

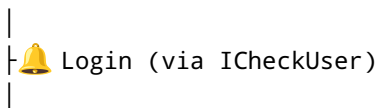
Menu Structure

Each menu has items tagged with:

-  Course Type: `Starters` , `Main Course` , `Desserts`
-  Veg/Non-Veg marker
- ₹ Price

Flow Diagram

Customer/Admin



```
|🔔 If Admin:
| |🔔 Manage Menus (IMenuManager)
| |🔔 Manage Delivery Partners (IDeliveryService)
|
|🔔 If Customer:
| |🔔 View Menus (IMenuViewer)
| |🔔 Add/Remove Items (IOrderManager)
| |🔔 Calculate Total (DiscountStrategy)
| |🔔 Make Payment (IPaymentService)
| |🔔 Assign Delivery (IDeliveryService)
| |🔔 Generate Invoice (IInvoiceGenerator)
```

Advantages

- 🕒 Fully **modular** and **scalable**
- 📦 Easy to **extend** new menu items, courses, delivery services
- 🕒 Adheres to **best coding practices** (SOLID, Clean Code)
- 🔧 Simple to test individual components
- 📊 Ideal for beginners learning object-oriented Java design



How to Run

```
git clone https://github.com/yourusername/food-ordering-console-app.git
cd food-ordering-console-app
```

Open in IntelliJ/Eclipse and run `FoodOrderingDriver.java` from `com.aurionpro.tests` package.

Final Note

This project is a great demonstration of Java principles applied to a real-world domain (food ordering). It can be further extended into a GUI or web-based app using frameworks like JavaFX or Spring Boot.

Happy Coding 🍷!