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

Project Folder Structure

```
com.aurionpro
├🔐 delivery
│   ├🔖 DeliveryPartner.java
│   ├🔖 DeliveryService.java
│   └🔖 IDeliveryService.java
├🔐 discount
```

```
│   ├🔖 IDiscountStrategy.java
│   └🔖 PercentageDiscount.java
├🔐 food
│   ├🔖 Menu.java
│   ├🔖 MenuItem.java
│   ├🔖 MenuManager.java
│   ├🔖 IMenuManager.java
│   └🔖 IMenuViewer.java
├🔐 invoice
│   ├🔖 InvoiceGenerator.java
│   └🔖 IInvoiceGenerator.java
├🔐 orders
│   ├🔖 CartItem.java
│   ├🔖 OrderManager.java
│   └🔖 IOrderManager.java
├🔐 payments
│   ├🔖 IPaymentService.java
│   ├🔖 CashPayment.java
│   └🔖 UpiPayment.java
├🔐 users
│   ├🔖 Admin.java
│   ├🔖 Customer.java
│   └🔖 ICheckUser.java
└🔐 tests
    └🔖 FoodOrderingDriver.java
```

SOLID Principles Applied

Principle	Application in the Project
S - Single Responsibility	Each class handles one specific task only (e.g., `MenuManager` for menu logic, `InvoiceGenerator` 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 `CashPayment` or `UpiPayment` wherever `IPaymentService` is expected.
I - Interface Segregation	Interfaces are small and specific (e.g., `IMenuViewer`, `IDeliveryService`).
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
      │
      ├🔔 Login (via ICheckUser)
      │
```

```
    ├🔔 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 🌡️!