KLE Society's

KLE Technological University



**Object Oriented Programming (22ECAC207)**

**Object Oriented Programming Lab (22ECAP206)**

**Course Project Report on**

**“Online Food Ordering System”**

*Submitted in partial fulfilment of the requirement for the award of*

**Degree of Bachelor of Engineering**

**in**

**Computer Science and Engineering**

*Submitted By*

**Devaj – 01FE21BCI055**

**Debanshu Behera – 01FE21BCI035**

*Submitted To*

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING,**

KLE TECHNOLOGICAL UNIVERSITY

HUBLI – 580 031 (India).

**Table of Content:**

1. **Problem definition ---------------------------------------------------------------------- 3**
2. **List of objects ----------------------------------------------------------------------------- 4**
3. **Class diagram ----------------------------------------------------------------------------- 6**
4. **Description of classes ------------------------------------------------------------------ 7**
5. **Description of main function -------------------------------------------------------- 8**
6. **Standard Design pattern -------------------------------------------------------------- 10**
7. **Results (output) -------------------------------------------------------------------------11**

**1.Problem definition:**

Develop an Online Food Ordering System that enables users to browse menus, place orders, and have food delivered to their location from various restaurants, providing a seamless and convenient online food ordering experience.

**2.List of objects:**

1. Customer

* c1(“Devaj”,”9384999999”, “M”, “Keshwapur Hubli”, 12344, 2000)

1. DeliveryPerson

* d1(“Raju”, “7564839475”, “M”, “Gokul Road”, 1)
* d2(“Diya”, “9874659900”, “F “Navnagar”, 2)
* d1(“Shyaam”, “7485009977”, “M”, “Naveen Park”, 3)

1. FoodItem

* f1(1,"Vada Pav",20,"Veg")
* f2(2,"Bhel Puri",30,"Veg")
* f3(3,"Chicken Biryani",130,"Non-Veg")
* f4(4,"Bread Omlette",150,"Non-Veg")
* f5(5,"Mango Lassi",60,"Veg")
* f6(1,"Veg Cheese Pizza",150,"Veg")
* f7(2,"Bruschetta",180,"Veg")
* f8(3,"Lasagna",200,"Non-Veg")
* f9(4,"Bombardino",120,"Non-Veg")
* f10(5,"Limoncello",110,"Veg")
* f11(1,"Veg Manchurian",80,"Veg")
* f12(2,"Chicken Noodles",150,"Non-Veg")
* f13(3,"Mutton Dumplings",170,"Non-Veg")
* f14(4,"Kung Pao Chicken",250,"Non-Veg")
* f15(5,"Bubble Tea",160,"Veg")

1. Menu

* menu1({f1,f2,f3,f4,f5})
* menu2({f6,f7,f8,f9,f10})
* menu3({f11,f12,f13,f14,f15})

1. Restaurant

* r1(1, “Spirit of India”, “Keshwapur”, menu1)
* r2(2, “The Sunday Sauce”, “Gokul Road”, menu2)
* r3(3, “Dragon Palace”, “Vidyanagar”, menu3)

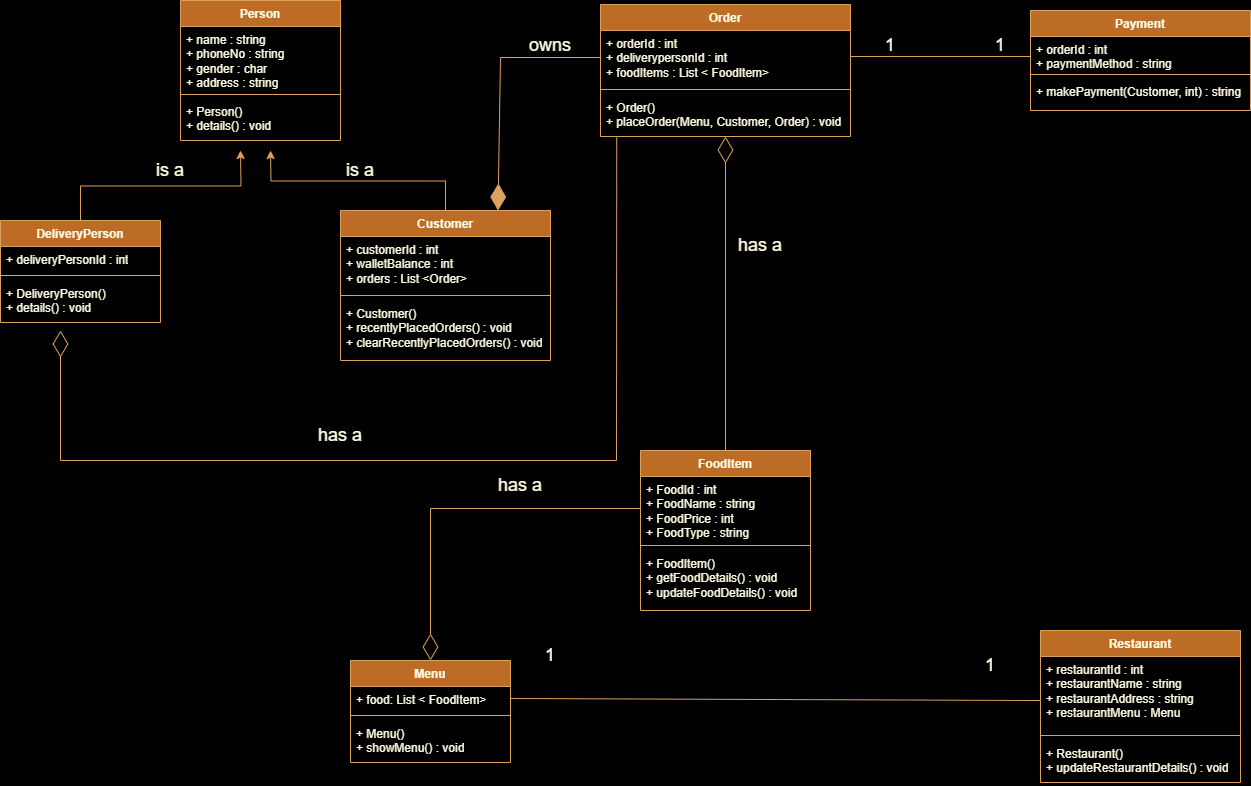
1. Order

* o1(1,d1)
* o2(2,d2)
* o3(3,d3)

1. Payment

* p(orderId, “Wallet Balance”)

**3.Class diagram:**

****

**4.Description of each class:**

Person: - This stores the personal information about the person whether it is delivery person or customer.

Delivery Person: - The Delivery Person class represents a delivery person responsible for delivering orders to customers.

Customer: - This stores information of the user of the application.

Food Items: - The Food Item class represents an individual item available on the restaurant's menu.

Menu: - The Menu class represents the collection of available food items at the restaurant.

Restaurant: - The Restaurant class represents a restaurant that offers food items for customers to order.

Orders: - The Order class represents a customer's order containing selected food items.

Payment: - The Payment class represents a payment made by the customer for an order.

**5.Main function description:**

The main function in the given C++ program serves as the entry point to the food ordering and delivery system. It orchestrates the interaction between the user (customer) and the system, allowing them to perform various actions such as selecting restaurants, placing orders, viewing customer details, and managing recently placed orders. Here's a detailed description of the main function:

1. It starts by creating instances of DeliveryPerson objects d1, d2, and d3, representing different delivery persons with their respective details.
2. Next, it creates instances of FoodItem objects f1 to f15, each representing a specific food item with attributes like ID, name, price, and type.
3. Then, it creates Menu objects menu1, menu2, and menu3, each containing a list of FoodItem objects. These menus represent the offerings of three different restaurants r1, r2, and r3.
4. Instances of Restaurant objects r1, r2, and r3 are created, each representing a restaurant with attributes like ID, name, address, and its specific menu.
5. A single instance of Customer object c1 is created, representing a customer with attributes such as name, phone number, gender, address, customerId, and walletBalance. The customer c1 is initialized with predefined values.
6. The program enters into an infinite loop, which acts as the main menu for the customer. It displays a list of available restaurants with their IDs and names, and provides options to the customer for various actions.
7. The customer is prompted to select a restaurant by its ID. Depending on the selected restaurant, the placeOrder function for the respective Order object (o1, o2, or o3) is called, allowing the customer to add food items to their cart and proceed with the ordering process.
8. Additionally, the main menu also offers the customer options to view their own details, view recently placed orders, clear recently placed orders history, or exit the application.
9. For each option chosen by the customer, the program executes the corresponding functions to display details, show recently placed orders, clear the order history, or exit the application.
10. The program uses exception handling to catch any invalid inputs made by the user and provides appropriate error messages.
11. The loop continues until the customer chooses to exit the application by selecting option 7.

Overall, the main function acts as the control center of the food ordering and delivery system, allowing customers to interact with the program and perform various actions related to ordering food from different restaurants and managing their orders.

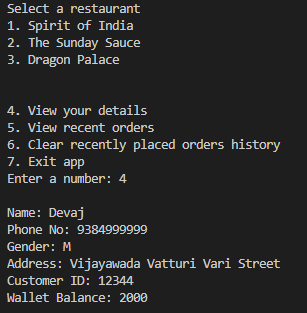
**6.Standard design pattern:**

The Singleton design pattern will be appropriate to ensure that there is only one instance of the customer class throughout the program’s execution. Here is an appropriate reason for using the Singleton pattern in this context:

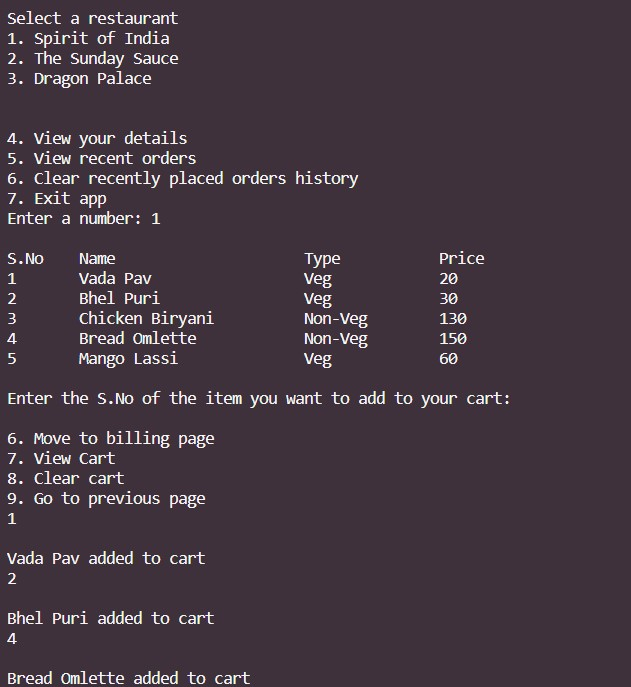
Unique Customer Instance: In this application, there should be only one representation of the current customer. By using the Singleton pattern, we can guarantee that there will be only one instance of customer class and all other parts of the code will interact with the single instance. This ensure consistency in customer’s data and behaviour.

**7.Results (output):**

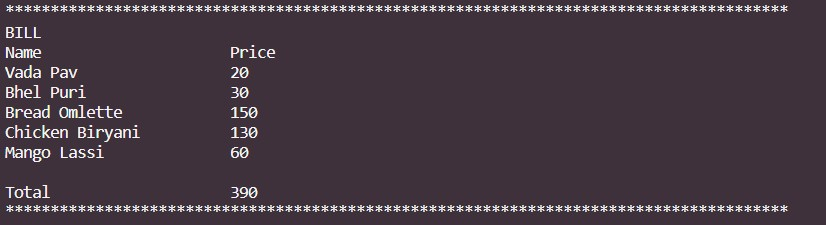
1.View your Details:



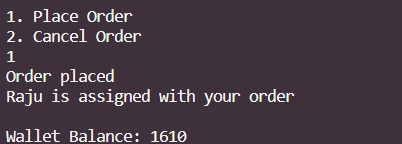
2.Selecting a restaurant and adding food items to your cart:



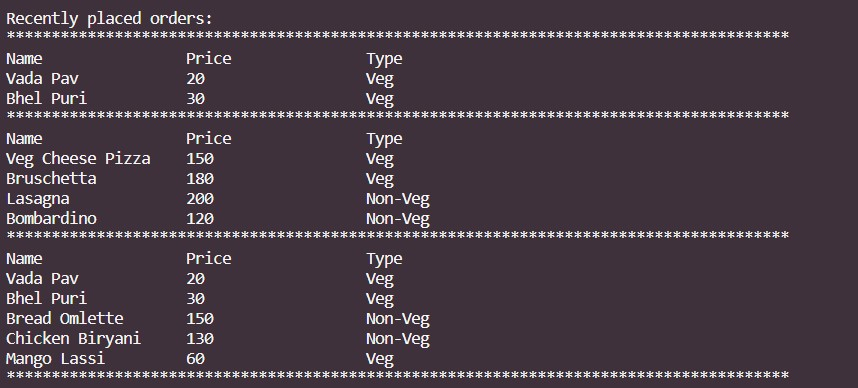
3.Move to the billing page to generate a bill for the items you have ordered:



4.Placing the order:



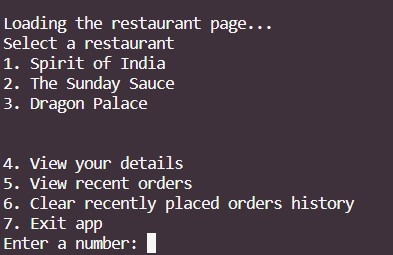
5.View recent orders history:



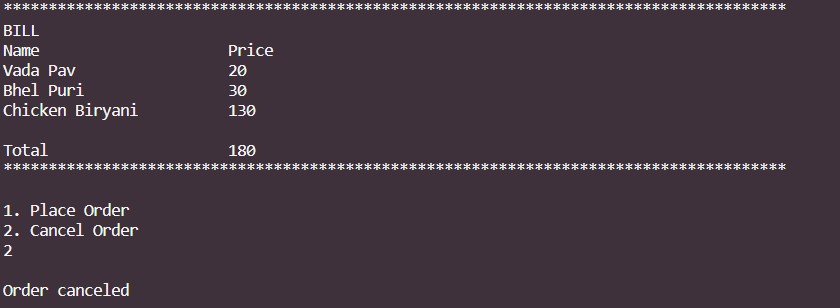
6.Clearing recently placed order:



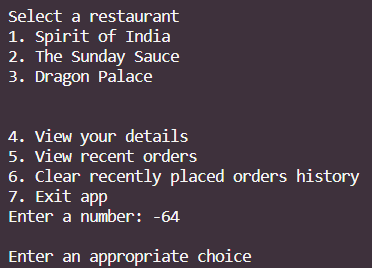
7.Going to previous page:



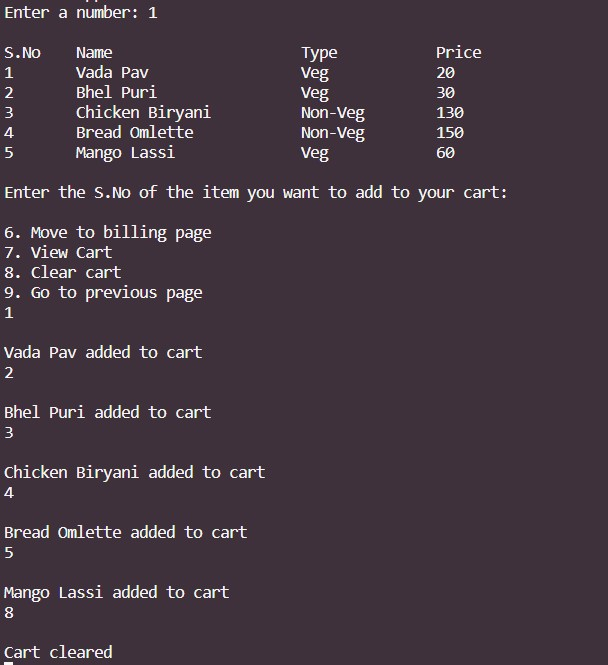
8.Cancelling Order:



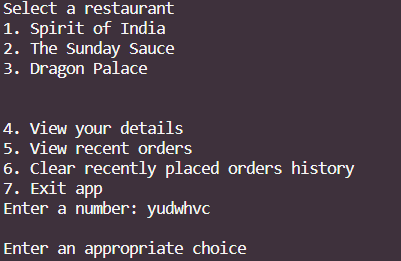
9.Exception Handling (If someone enters negative value):



10.Clearing cart:



11.Exception Handling (If someone enters string value):



12.Exit from the app:

