Online Food Ordering System

Object-Oriented Programming Project Report

Arman Kumar Aditya	Pulkit Saurastri	Eshan Karia	Amogh Tarun
2022A8PS0790G	2022A8PS1529G	2022A3PS0433G	2022A3PS0047G

System Overview

This project models an **Online Food Ordering System** that allows users to browse restaurants, add dishes to cart, place orders, make payments, and track deliveries. The system is fully based on object-oriented principles in Java, featuring abstraction, inheritance, interfaces, exception handling and file handling concepts.

The system consists of the following key classes:

- User
- Restaurant
- Dish
- Address
- Delivery (Abstract)
- RegularDelivery
- EcoSaverDelivery
- Bill
- Payment (Interface)
- CODPayment
- CreditCardPayment
- UPIPayment
- Driver

Class Descriptions

Class: Address

Fields:

- `String name` - Name associated with the address.
- `int x, y` - Coordinates representing location.

Constructor:

- Initializes `name`, `x` and `y`.

Methods:

- `getName()`, `getX()`, `getY()` - Standard getters.

Class: Dish

Fields:

- `String name` - Name of the dish.
- `int rate` - Price of the dish.
- `boolean isVeg`
- `String type` - Cuisine type (e.g., Indian, Chinese).

Constructor:

- Initializes dish attributes.

Methods:

- `getName()`, `getType()`, `getRate()`, `isVeg()` - Standard getters.
- `getDishDetails()` - Returns dish details.

Class: Restaurant

Fields:

- `Address address`
- `String name`
- `Dish[] dishes` - Menu dishes.
- `int dishCount` - Count of dishes currently in cart.
- `final int MAX_DISHES = 10` - Maximum number of dishes that can be added to the cart.

Constructors:

- Default constructor initializes an empty restaurant.
- Parameterized constructor initializes with dishes and address.

Methods:

- `getAddress()`, `getDishes()` - Standard getters.
- `addDish(Dish)` - Adds the dish to the cart, if cart is not full.
- `isDishPresent(Dish)` - If dish is already added to cart.

Class: User

Fields:

- `String name, contact_number, email_id` - User information.
- `Address[] address` - Up to 5 saved addresses.
- `int number_of_addresses`
- `Cart cart` - Nested static class representing user's cart.

Constructors:

- `public User(String name, String number, String email_id)` - If user enters email_id.
- `public User(String name, String number)`

Methods:

- Address management: `add_address()`
- Order placement: `order()` (overloaded for payment types: COD, UPI, Credit Card)
- `getName()`, `getContactNumber()`, `getEmailId()` - Standard getters.

- `setContactNumber(String number)`, `setEmailId(String email_id)` - Standard setters.

Nested Class: Cart

- `Restaurant restaurant`
- `Dish[] dishes`
- `int number_of_dishes`

Constructors:

- Default constructor initializes an empty cart.

Methods:

- `update_restaurant(Restaurant restaurant)` - Updates restaurant and cart.
- `add_dish(Dish dish)` - Adds dish to cart.

Abstract Class: Delivery

Methods:

- `abstract void deliveryTime(Address source, Address destination)`

Class: RegularDelivery (extends Delivery class)

Fields:

- `int speed = 40` - Delivery speed for standard orders.

Methods:

- `deliveryTime(Address source, Address destination)` - Calculates delivery time based on euclidean-distance from the restuarant.

Class: EcoSaverDelivery (extends Delivery class)

Fields:

- `int speed = 20` - Delivery speed for Eco-Saver orders.

Methods:

- `deliveryTime(Address source, Address destination)` - Calculates delivery time based on euclidean-distance from the restuarant.

Class: Bill (extends User class)

Fields:

- `Cart cart` - Nested static class representing user's cart.
- `double totalAmount` - Total amount including GST.
- `double finalAmount` - Total amount including payment portal taxes.
- `int orderNumber`

Constructors:

- `public Bill(String userName, int orderNumber, User.Cart cart, String phoneNumber, String email_id)` - If user enters email_id.

- `public Bill(String userName, int orderNumber, User.Cart cart, String phoneNumber)`

Methods:

- `calculateBill()` - If Cash on delivery option is selected.
- `calculateBill(String upiId)` - If UPI payment option is selected.
- `calculateBill(long cardNumber)` - If Credit card payment is selected.

Interface: Payment

Methods:

- `pay(double amount)`
- `getPaymentMethod()`

Classes Implementing Payment

- CODPayment
- CreditCardPayment
- UPIPayment

Each payment class implements customized `pay()` method logic.

Class: Driver

Fields:

- `static Restaurant[] restaurants` - Array to store restaurant objects (max 100).
- `static User[] users` - Array to store user objects (max 100).
- `static int number_of_restaurants` - Current number of restaurants.
- `static int number_of_users` - Current number of users.

Methods:

- `static void add_restaurant(String... words)`
  Adds a new restaurant using name and coordinates.
- `static void add_dish(String... words)`
  Adds a new dish to an existing restaurant's menu.
- `static void add_user(String... words)`
  Adds a new user. Can handle both with and without email ID.
- `static void add_to_cart(String... words)`
  Adds a specified dish from a restaurant into a user's cart.
- `static void place_order(String... words)`
  Places an order for a user, handling multiple payment modes (COD, UPI, Card).
- `static void add_address(String... words)`
  Adds an address for a user with given coordinates.
- `public static void main(String[] args)`
  Reads and executes commands from `input.txt` file line-by-line using BufferedReader. Handles different actions like user creation, restaurant addition, cart management, order placement, and address updates.

Input File Structure:

- Each line in `input.txt` specifies a command.

- Examples:
  - `User Shivansh 7905488913 f20220140@goa.bits-pilani.ac.in`
  - `Restaurant 7th_Heaven 12 20`
  - `Dish 7th_Heaven Burger 120 true FastFood`
  - `Item Shivansh 7th_Heaven Burger`
  - `Order Shivansh 0 true`
  - `Address Shivansh Home 10 15`

Rubric Table: Features Used

Sr. No.	Feature	Usage Count
I	Overloaded Methods	6 (order(), calculateBill())
II	Overloaded Constructors	4 (User, Bill)
III	Varargs Overloading	6 (add_user, add_restaurant, add_dish, add_to_cart, place_order, add_address)
IV	Nested Class	1 (Cart inside User)
V	Abstract Class	1 (Delivery)
VI	Interface	1 (Payment)
VII	Hierarchical Inheritance	1 (Delivery → RegularDelivery, EcoSaverDelivery)
VIII	Multiple Inheritance	3 (CODPayment, CreditCardPayment, UPIPayment implement Payment)
IX	Wrapper Classes	2 (long CardNumber, String UPIId)
X	Package	1 (Mentioned via comment, OOPS.Final_Presentation)
XI	Exception Handling	2 (IOException, ArrayIndexOutOfBoundsException)
XII	I/O (File Handling and BufferedReader)	1 (Driver file reading)