Delivery 1

By Cristian Racila & Florence K. Neflas

Scenario

Food delivery simulation platform (like DoorDash or UberEats)

The project will simulate the interaction between customers, restaurants, and drivers while managing tracking order status and delivery times.

In this project, the user will be able to live the Customer experience.

- Customer (Users) select items from restaurant menus, place orders, and track delivery status (if its processing, on the way, completed).
- Restaurant: Accepts orders from customers, prepares food, hands it to delivery drivers, manages their menu and updates the order status.
- Driver: Accepts delivery requests, picks up food from restaurants, and delivers it to customers while updating the order status to customers.

Design Paradigm

1. Classes:
   - Customer: Stores customer information (name, address, phone number, hasAllergy) and places orders.
   - {Our Delivery App Name}: Manages interactions between Customers and Restaurants, Drivers and Restaurants, and Drivers and Customers. Allows users to register, place orders, view statuses, view menus, etc.
   - Restaurant: Manages the menu, available food items, and order processing.
   - Driver: Handles driver details (licensePlate, licenseNumber, rating, availabilityStatus) and assigned orders.
   - Order: Represents the details of an order, including items, customer, restaurant, and driver.

2. Interfaces and Abstract Classes:
   - OrderProcessable (Interface): Defines common behaviours like placing orders, preparing food, or updating delivery status.
   - Person (Abstract class): A parent class for Customer and Driver to share attributes like name, phone number, etc.

Design Paradigm (cont.)

3. Design Patterns:
   - Singleton Design: For managing a centralized system that tracks all active orders and driver availability, ensuring only one instance manages these processes.
   - Factory Design: For creating different types of Orders based on delivery types (e.g., standard delivery, express delivery) and managing restaurant menus.

4. Functionalities:
   - Customers can browse restaurant menus, place orders (with at least one item required), and check order status.
   - Restaurants can accept or reject orders and manage their menu.
   - Drivers can accept or reject deliveries based on their current availability.
   - The system handles real-time updates to order status (e.g., when an order is prepared, when the driver is on the way).

Expected Output

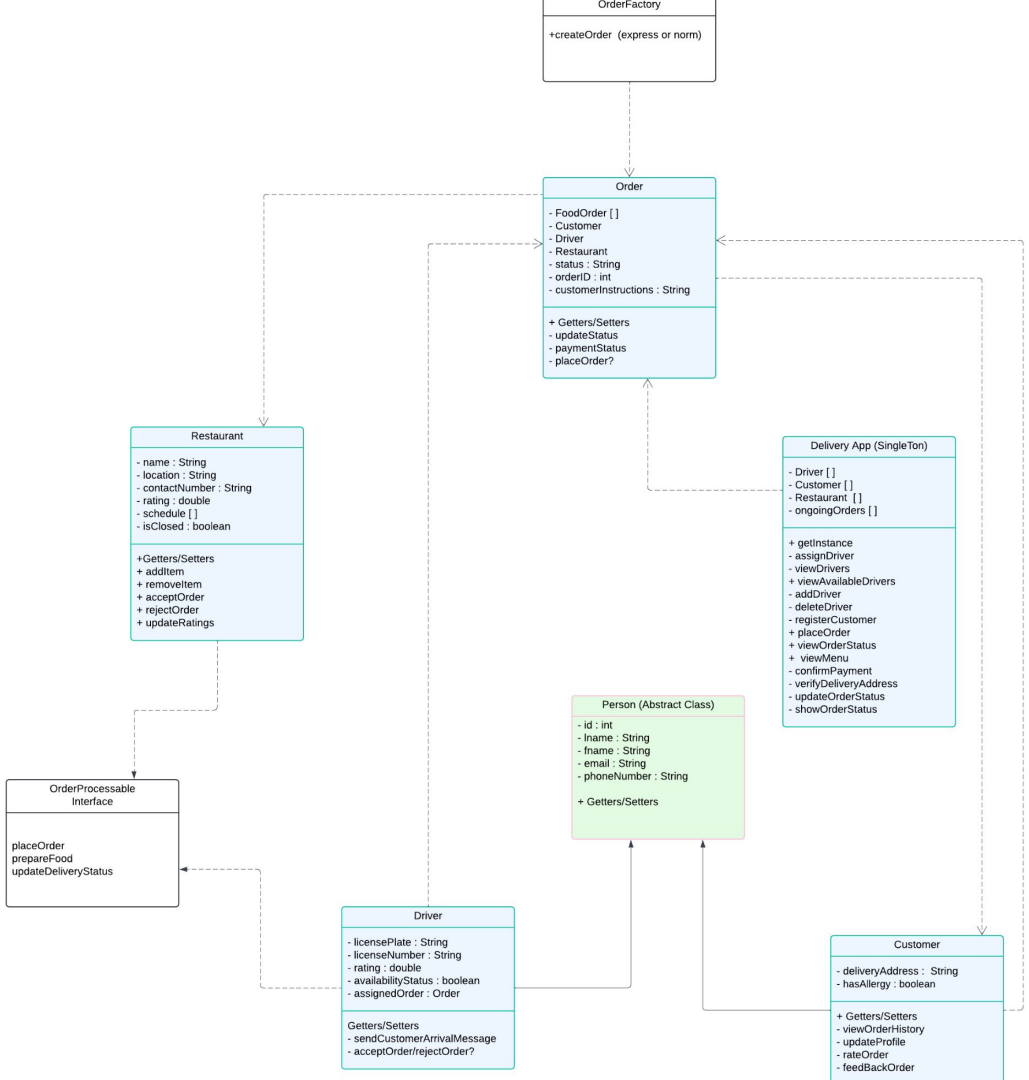
The expected output is a simulation where a user can act as a customer, place an order, and see that order go through the steps of being processed, accepted by a driver, and delivered. The user should be able to:

- Select a Restaurant: Browse a list of available restaurants.
- Choose Items: Select items from a restaurant's menu and place an order.
- Track Order: Receive status updates (e.g., "Order Confirmed," "Food Being Prepared," "Out for Delivery").
- Driver: The system will automatically assign an order to an available driver.
- Order Delivery: Once delivered, the user gets a notification, and the order is marked as complete.

Git Repository

https://github.com/Crisguts/FoodDeliveryProject.git

Diagram


OrderFactory
+createOrder (express or norm)
Order
- FoodOrder [ ]
- Customer
- Driver
- Restaurant
- status : String
- orderID : int
- customerInstructions : String
+ Getters/Setters
- updateStatus
- paymentStatus
- placeOrder?
Restaurant
- name : String
- location : String
- contactNumber : String
- rating : double
- schedule [ ]
- isClosed : boolean
+Getters/Setters
+ addItem
+ removeItem
+ acceptOrder
+ rejectOrder
+ updateRatings
Delivery App (SingleTon)
- Driver [ ]
- Customer [ ]
- Restaurant [ ]
- ongoingOrders [ ]
+ getInstance
- assignDriver
- viewDrivers
+ viewAvailableDrivers
- addDriver
- deleteDriver
- registerCustomer
+ placeOrder
+ viewOrderStatus
+ viewMenu
- confirmPayment
- verifyDeliveryAddress
- updateOrderStatus
- showOrderStatus
Person (Abstract Class)
- id : int
- lname : String
- fname : String
- email : String
- phoneNumber : String
+ Getters/Setters
OrderProcessable Interface
placeOrder
prepareFood
updateDeliveryStatus
Driver
- licensePlate : String
- licenseNumber : String
- rating : double
- availabilityStatus : boolean
- assignedOrder : Order
Getters/Setters
- sendCustomerArrivalMessage
- acceptOrder/rejectOrder?
Customer
- deliveryAddress : String
- hasAllergy : boolean
+ Getters/Setters
- viewOrderHistory
- updateProfile
- rateOrder
- feedBackOrder