**BYTE ME!!!**

**FOOD DELIVERY APPLICATION**

The following project is my implementation of a basic food delivery service providing application. It provides access to two kinds of users Customers and Admin. The functionalities have been implemented with respect to the basic needs of the 2 users for the required system. The project has helped me a lot to improve my understanding of OOPS concents and learn their practical use cases and application.

**FEATURES IMPLEMENTED**

**CUSTOMERS**

--Browse Menu, Cart operations(add, remove, update, checkout), order tracking, reviews and comments.

**ADMIN**

--Menu Management(add, remove, update), order management(process pending orders, update status etc.), sales report generator

**Assumptions**

--Availability of items has been handled by admin in the form of quantity of an item in the menu

--The customers can only cancel the order if it has not been delivered already.

--Admin has the choice to handle or not to handle a particular order’s special request.

-- Please enter the correct data type for the input else program will get terminated(All the choices to be entered are int or string (what has to be entered will be obvious))

**OOPS Concepts Applied**

**1.Abstraction**

Each user interacts only with the parts of the system they are concerned with, and unnecessary details are hidden. By providing methods (like browsing a menu or placing an order) without exposing the complexities of the operations, the program effectively uses abstraction

**2.Encapsulation**

The system uses getter and setter methods to access or modify private data fields while preventing unauthorized access to sensitive data. For example, the Customer class has several private attributes, such as MyOrders, CustomerID, MyCart, and isVIP. These attributes cannot be accessed directly from outside the class, which restricts the visibility and protects the data from unintended modifications.

**3.Inheritance**

The Customer class and Admin class extends the User class, which is an abstract representation of a user in the system. By doing this, they inherits properties and behaviors from User without revealing how the User class is implemented internally.

**4.Polymorphism**

Polymorphism allows objects to be treated as instances of their parent class, enabling multiple implementations of the same method. For example, the toString() method is a method of an object class which is the parent class of all class. So through overriding it, I have implemented it differently in fooditem and order class.

**HOW TO RUN THE CODE**

--Open the project inside a text editor that can handle java code (preferably IntelliJ)

--Run the main.java

**PROCESS FLOW**

**Example case scenario --> A customer wants to place an order**

--Run main.java

--Select login as admin

--Signup as a new admin and the admin menu will appear

--Select Menu management then add a new item

--Enter the required details for the corresponding item (name, item id, price ,quantity)

--Logout from admin

--Select Customer and signup a new Customer

-- Select the Add a new order

--Select add an item to cart

--Select the item id and the quantity required

--Select the view cart option to view your current selected order

--Select checkout

--Enter the required amount

--The order has been placed !!!