PROJECT NAME:- FOOD ORDERING SYSTEM

MEMBERS :-

Arghya Mukherjee	(MT21014)
Jeet Tewatia	(MT21036)
Mitali Gupta	(MT21051)
Shrey Rustagi	(MT21145)

HOW TO RUN:-

Step 1 : Open the folder Database and it contains a SQL file of tables(with dummy data) .

Step 2: Run the code (master.java) on any IDE.

INTRODUCTION:-

This project starts with giving customers the choice to sign up (using a new email id) if he/she is a new user or login using email and password. Customer also enters his/her address (city and area also) The system checks for the correctness of these credentials. After this he is given several options.

If we choses he wants to order some food then he is shown restaurants(along with menus) of his city only . He can choose to order from any restaurant and enter the name of the food item and quantity required. He is given a choice to enter items directly in the cart (from here he will be directed to the order page directly) or save items in the wishlist for future use .

On the order page: He is given a choice if he still wants to do any modification in his cart items , Total amount is calculated for the items using the menu table and delivery charges are calculated according to the distance between the area of customer and the restaurant chosen . After his final amount is calculated he is given promo codes: To save 50% , if already used this then he is given another promo code to save 20%. After this he is directed to the payment gateway where he is given several payment modes , he can choose any of them and place his order .

He can then rate the restaurant and the app also . After this he is given a real time tracking feature for his order . He is even provided the choice to cancel the order if the time has exceeded 10% more than the total expected delivery time .

SOFTWARE REQUIREMENTS:-

- Jdk 8 or above
- SQL Workbench
- Eclipse or any java based IDE

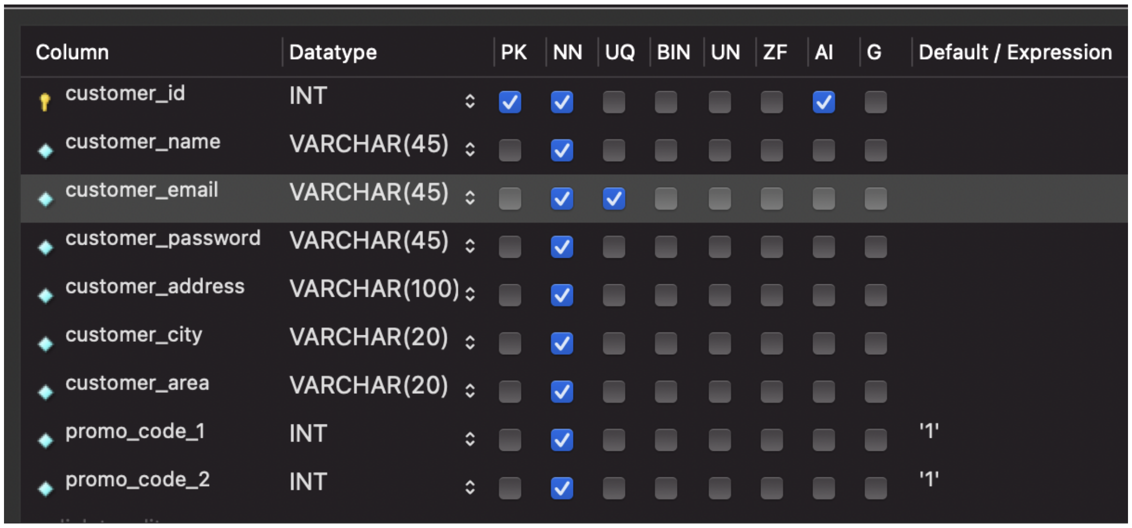
HARDWARE REQUIREMENTS:-

- RAM : At least 128 MB
- Disk Space : 124 MB for JRE, 2 MB for Java Update
- Processor : Minimum Pentium 2 266 MHz processor

DATABASE:-

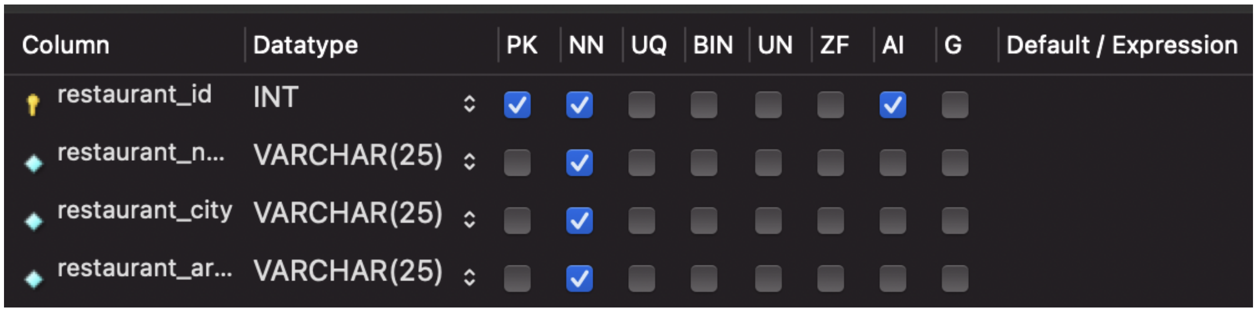
Schema of the different table which we used:-

Customer Table



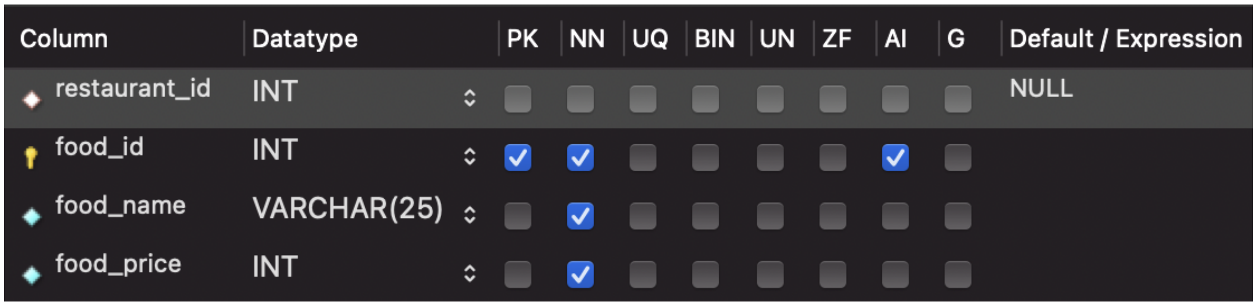
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
customer_id	INT	✓	✓					✓		
customer_name	VARCHAR(45)		✓							
customer_email	VARCHAR(45)		✓	✓						
customer_password	VARCHAR(45)		✓							
customer_address	VARCHAR(100)		✓							
customer_city	VARCHAR(20)		✓							
customer_area	VARCHAR(20)		✓							
promo_code_1	INT		✓							'1'
promo_code_2	INT		✓							'1'

Restaurant Table



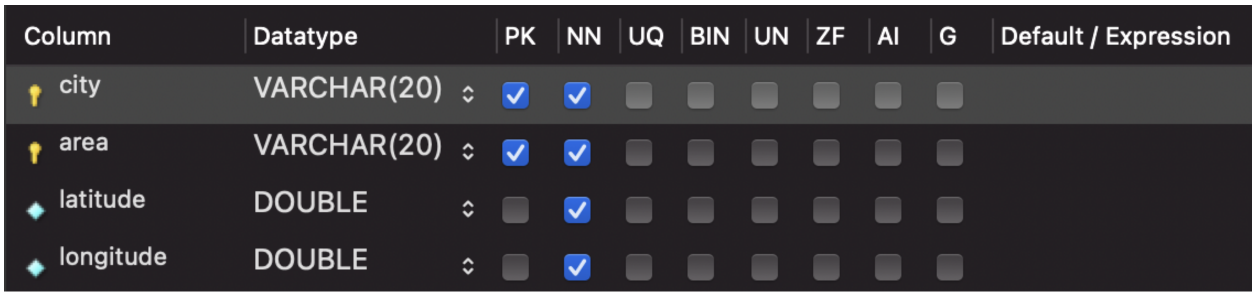
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
restaurant_id	INT	✓	✓					✓		
restaurant_n...	VARCHAR(25)		✓							
restaurant_city	VARCHAR(25)		✓							
restaurant_ar...	VARCHAR(25)		✓							

Menu Table



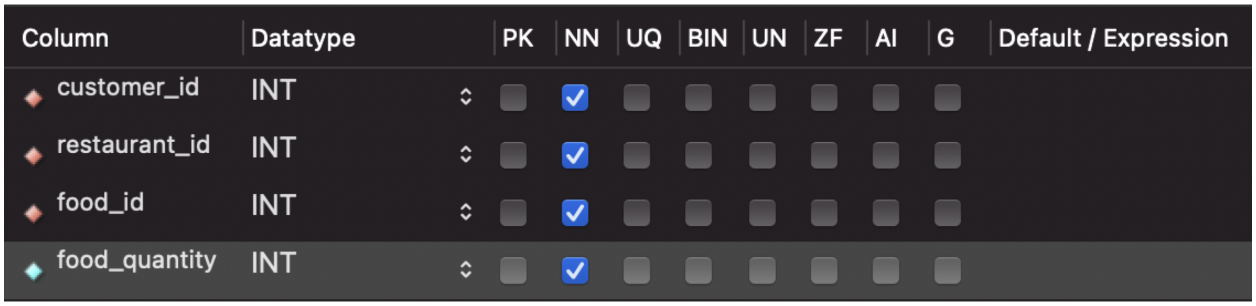
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
restaurant_id	INT									NULL
food_id	INT	✓	✓					✓		
food_name	VARCHAR(25)		✓							
food_price	INT		✓							

Area Coordinates Table



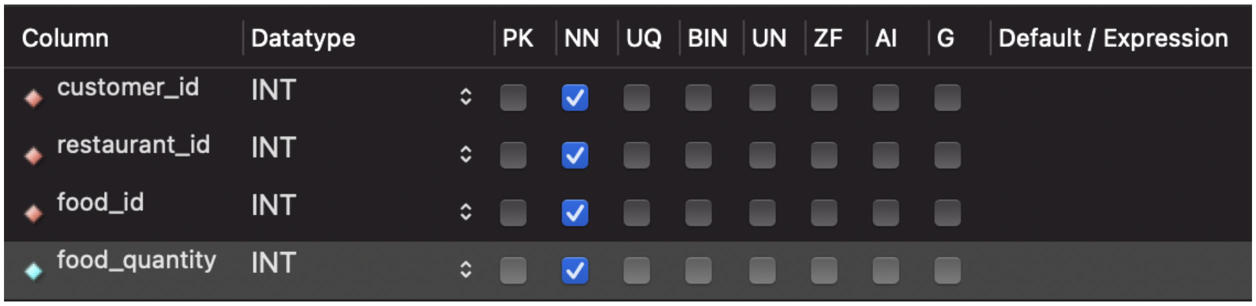
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
city	VARCHAR(20)	✓	✓							
area	VARCHAR(20)	✓	✓							
latitude	DOUBLE		✓							
longitude	DOUBLE		✓							

Cart Table



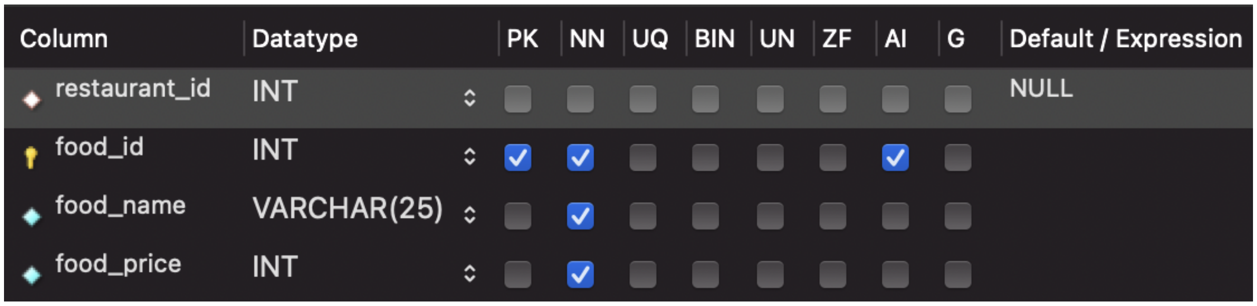
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
customer_id	INT		✓							
restaurant_id	INT		✓							
food_id	INT		✓							
food_quantity	INT		✓							

Wishlist Table



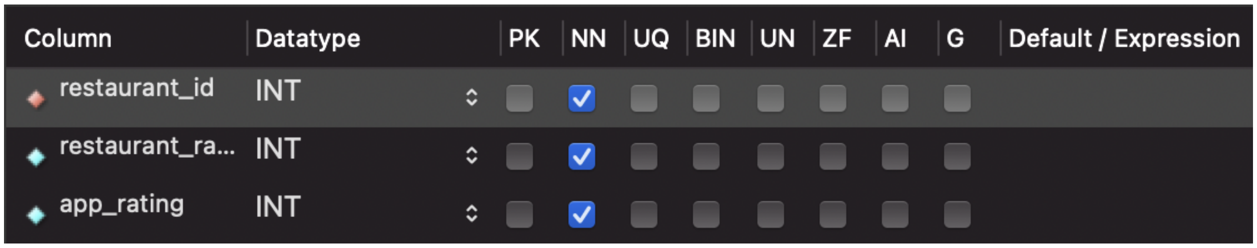
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
customer_id	INT		✓							
restaurant_id	INT		✓							
food_id	INT		✓							
food_quantity	INT		✓							

Order Table



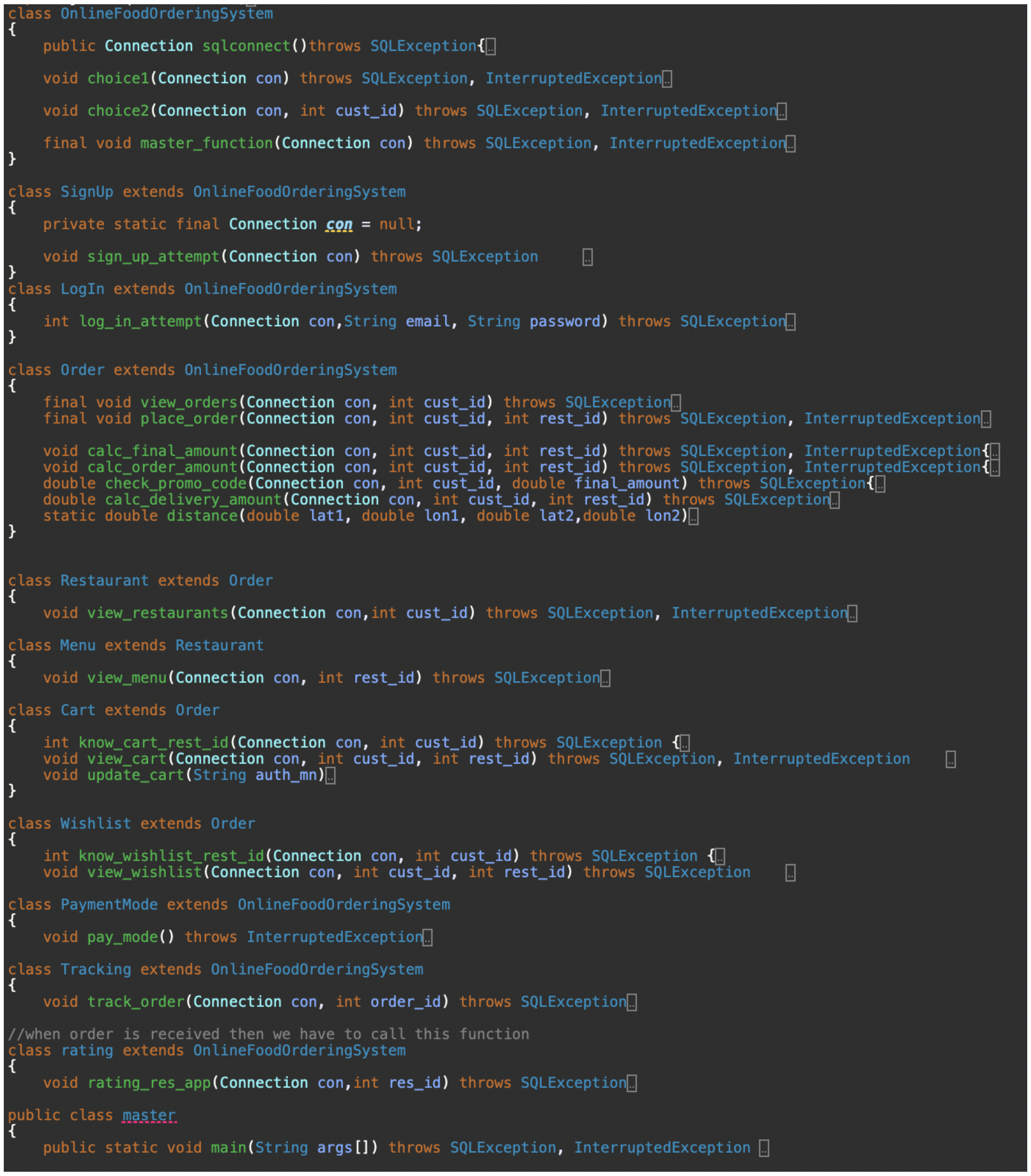
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
restaurant_id	INT									NULL
food_id	INT	✓	✓					✓		
food_name	VARCHAR(25)		✓							
food_price	INT		✓							

Rating Table



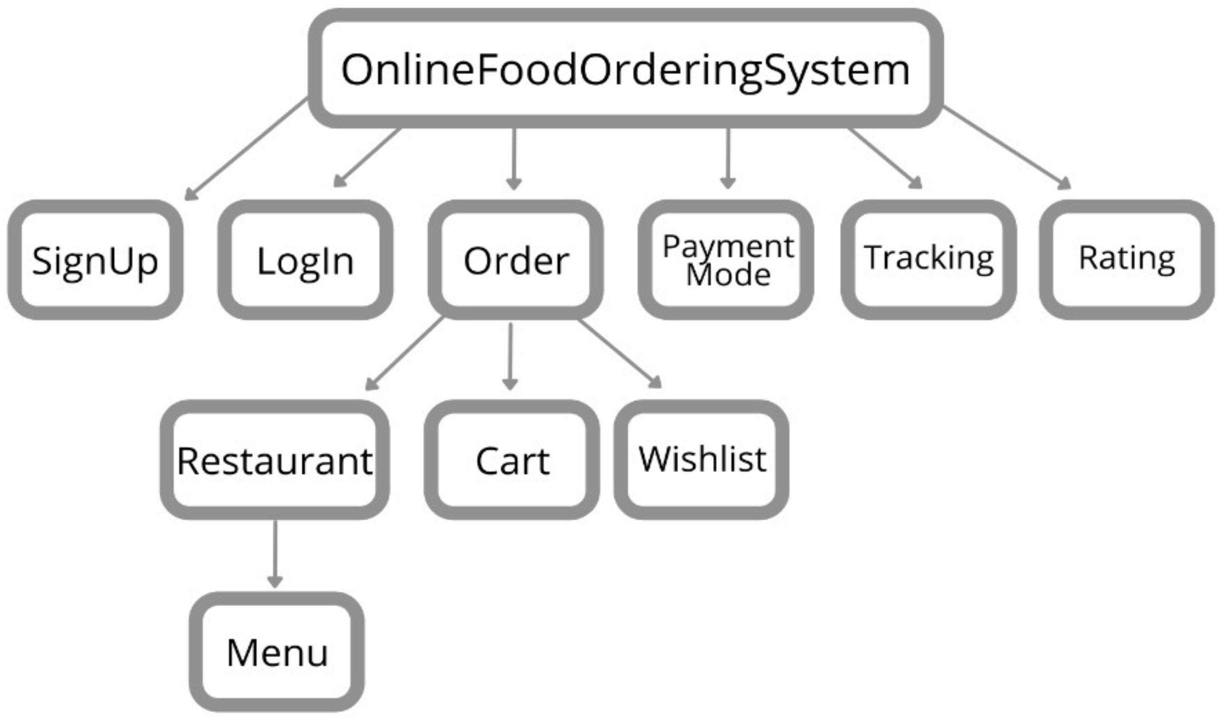
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
restaurant_id	INT		✓							
restaurant_ra...	INT		✓							
app_rating	INT		✓							

OVERVIEW OF ALL CLASSES :-



```
class OnlineFoodOrderingSystem
{
    public Connection sqlconnect()throws SQLException{

    void choice1(Connection con) throws SQLException, InterruptedException

    void choice2(Connection con, int cust_id) throws SQLException, InterruptedException

    final void master_function(Connection con) throws SQLException, InterruptedException
}

class SignUp extends OnlineFoodOrderingSystem
{
    private static final Connection con = null;

    void sign_up_attempt(Connection con) throws SQLException
}
class LogIn extends OnlineFoodOrderingSystem
{
    int log_in_attempt(Connection con,String email, String password) throws SQLException
}

class Order extends OnlineFoodOrderingSystem
{
    final void view_orders(Connection con, int cust_id) throws SQLException
    final void place_order(Connection con, int cust_id, int rest_id) throws SQLException, InterruptedException

    void calc_final_amount(Connection con, int cust_id, int rest_id) throws SQLException, InterruptedException{
    void calc_order_amount(Connection con, int cust_id, int rest_id) throws SQLException, InterruptedException{
    double check_promo_code(Connection con, int cust_id, double final_amount) throws SQLException{
    double calc_delivery_amount(Connection con, int cust_id, int rest_id) throws SQLException
    static double distance(double lat1, double lon1, double lat2,double lon2)
}


class Restaurant extends Order
{
    void view_restaurants(Connection con,int cust_id) throws SQLException, InterruptedException

class Menu extends Restaurant
{
    void view_menu(Connection con, int rest_id) throws SQLException

class Cart extends Order
{
    int know_cart_rest_id(Connection con, int cust_id) throws SQLException {
    void view_cart(Connection con, int cust_id, int rest_id) throws SQLException, InterruptedException
    void update_cart(String auth_mn)
}

class Wishlist extends Order
{
    int know_wishlist_rest_id(Connection con, int cust_id) throws SQLException {
    void view_wishlist(Connection con, int cust_id, int rest_id) throws SQLException

class PaymentMode extends OnlineFoodOrderingSystem
{
    void pay_mode() throws InterruptedException

class Tracking extends OnlineFoodOrderingSystem
{
    void track_order(Connection con, int order_id) throws SQLException

//when order is received then we have to call this function
class rating extends OnlineFoodOrderingSystem
{
    void rating_res_app(Connection con,int res_id) throws SQLException

public class master
{
    public static void main(String args[]) throws SQLException, InterruptedException
```

INHERITANCE OF CLASSES :-



Class OnlineFoodOrderingSystem is superclass for :-

- SignUp
- Login
- Order
- PaymentMode
- Tracking
- Rating

Class Order is superclass for:-

- Restaurant
- Cart
- Wishlist

Class Restaurant is superclass for:-

- Menu

ENCAPSULATION :-

Order class have various variables like final_amount, delivery_amount and order_amount .

It have 4 major functions :-

- calc_final_amount() :-
  sums up the value of order_amount and delivery_amount.
- calc_order_amount() :-
  Calculate the total value of items in the cart by using the menu table to get prices and then multiply by quantity and return order_amount.
- calc_delivery_amount() :-
  Calculate the delivery_amount using the area coordinates of user and the area coordinates of restaurant.
- check_promo_code():-
  The user profile is checked if he is left with his promo codes (using customer_id to refer customer table) and then he can have choice to use any one offer among promo_code_50 or promo_code_20 (this saves 50% and 20% respectively) . So final_amount can be modified in this function.

Here we see we have many different tasks to be done on these four variables and instead of doing them in different classes and making all the variables global , we made class Order which have different functions to perform all the required tasks on these variables .

ABSTRACTION :-

An interface is a completely abstract class for grouping related methods with empty bodies.

As we are using interface's methods , we only know the classes that implement it and their methods and therefore all knowledge about the implementation is concealed from the user, resulting in 100 percent abstraction.

We have used the ‘Connection’ interface to establish database connection in our program. Which helps to perform data abstraction in our program.

DATABASE CONNECTIVITY :-

We performed database connectivity using a public function namely sqlconnect() whose return type is ‘Connection’. It is a public function which uses try catch exception handling to perform database connection. We established a connection using ‘Connection’ class object as follows:-

Connection con = DriverManager.getConnection(url,UserName,Password);

Where;
con: It is a reference to the ‘Connection’ interface.
url : Uniform Resource Locator which is created as shown below:
String url = “jdbc:mysql://localhost:3306/oopd”;
UserName: username from which our SQL command prompt can be accessed.
Password: password from which our SQL command prompt can be accessed.