OBJECT ORIENTED PROGRAMING

MINI PROJECT

CAFE MANAGEMENT SYSTEM

- SUBMITTED BY

J.S.KATHYAEINI

II-CSE-B

REG NO: 2117230020099

- SUBMITTED TO

MR.M.ASHOK

ASSISTANT PROFESSOR

Reg no:2117230020099

CAFE MANAGEMENT SYSTEM :

Aim :

The aim of the CafeManagement program is to create an interactive console-based application that allows customers to view a cafe's menu, place orders, and calculate the total bill based on the ordered items and their quantities. It also involves storing order details in a MySQL database for record-keeping

Description:

The CafeManagement program consists of the following features:

1. **Database Connection**: Connects to a MySQL database named cafe which contains a menu table for items and a customer_orders table for storing customer orders.
2. **Menu Display**: Fetches and displays the menu items from the database, including item IDs, names, descriptions, and prices.
3. **Order Placement**: Prompts the user to enter their name, item IDs, quantities, and payment method. It calculates the total bill based on the ordered items.
4. **Database Insertion**: Inserts the order details, including the customer's name, ordered items, total bill amount, and payment method, into the customer_orders table in the database.
5. **Error Handling**: Includes error handling to manage database connection issues and invalid inputs.

Algorithm

1. **Initialize Database Connection**:
    - Establish a connection to the MySQL database using JDBC.
    - If the connection fails, print an error message and exit.
2. **Create Tables**:
    - Check if the customer_orders table exists. If not, create it to store order information.
3. **Display Menu**:
    - Execute a SQL query to retrieve and display the menu items from the menu table.
    - Print the item ID, name, description, and price in a formatted manner.
4. **Collect Customer Input**:
    - Prompt the customer for their name.
    - Ask for the item IDs of the ordered items (comma-separated).
    - Ask for the quantities of each item (comma-separated).

   - Prompt for the payment method.

5. **Calculate Total Bill**:
   - Initialize a variable totalBill to 0.
   - For each item ID provided:
     - Fetch the price of the item from the menu table using the item ID.
     - Multiply the price by the corresponding quantity and add it to totalBill.
6. **Insert Order into Database**:
   - Prepare a SQL insert statement for the customer_orders table.
   - Set the parameters for customer name, ordered items, total bill amount, and payment method.
   - Execute the insert statement.
7. **Display Confirmation**:
   - Print a message confirming that the order has been successfully inserted into the database.
8. **Close Resources**:
   - Close the database connection and any other resources used (e.g., Scanner).

Pseudocode Representation:

```
BEGIN
  CONNECT to MySQL database
  IF connection fails THEN
    PRINT error message
    EXIT
  CREATE customer_orders table IF NOT EXISTS
  DISPLAY cafe menu
  PROMPT customer for name
  PROMPT customer for order item IDs (comma-separated)
  PROMPT customer for quantities (comma-separated)
  PROMPT customer for payment method
  totalBill = 0.0
```

```
   FOR each item ID in order item IDs DO
      price = GET item price from menu using item ID
      quantity = GET corresponding quantity
      totalBill += price * quantity
   INSERT order into customer_orders with customer name, ordered items, totalBill, and payment
method
   PRINT confirmation message
   CLOSE database connection
END
```

PROGRAM:

MYSQL CODE:

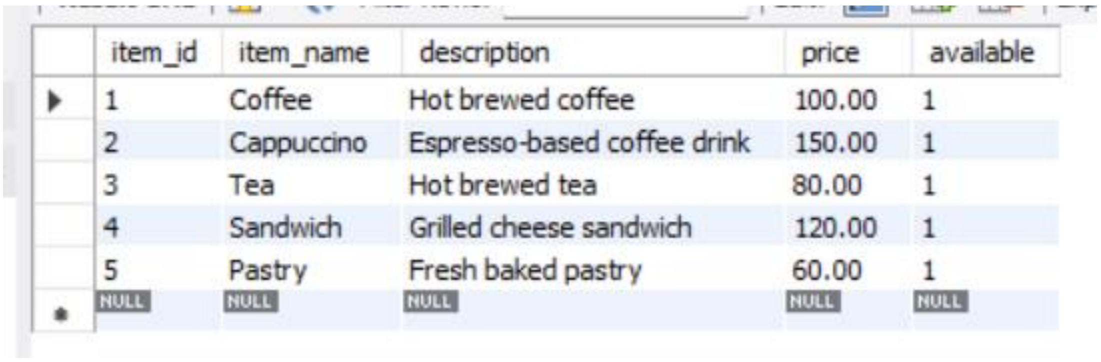
```
- Drop the database if it already exists (optional, use with caution)
DROP DATABASE IF EXISTS cafe;
-- Create the database
CREATE DATABASE cafe;
-- Use the created database
USE cafe;
-- Create the menu table
CREATE TABLE menu (
   item_id INT AUTO_INCREMENT PRIMARY KEY,
   item_name VARCHAR(100) NOT NULL,
   description TEXT,
   price DECIMAL(10, 2) NOT NULL,
   available BOOLEAN DEFAULT TRUE
);
-- Insert sample data into the menu table
INSERT INTO menu (item_name, description, price, available) VALUES
('Coffee', 'Hot brewed coffee', 100.00, TRUE),
('Cappuccino', 'Espresso-based coffee drink', 150.00, TRUE),
('Tea', 'Hot brewed tea', 80.00, TRUE),
('Sandwich', 'Grilled cheese sandwich', 120.00, TRUE),
('Pastry', 'Fresh baked pastry', 60.00, TRUE);
-- Check the inserted data
SELECT * FROM menu;
```

JAVA CODE:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;
public class CafeManagement {
  // MySQL database URL, username, and password
  private static final String URL = "jdbc:mysql://localhost:3306/cafe";
  private static final String USERNAME = "root";
  private static final String PASSWORD = "kathyaeini@1706";
  public static void main(String[] args) {
    // Establish connection and setup the table
    try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {
      // Create the table if it doesn't exist
      createTable(connection);
      // Display the cafe menu before taking order
      displayMenu(connection);
      // Collect user inputs
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter Customer Name: ");
      String customerName = scanner.nextLine();
```

```
            System.out.print("Enter Order Item IDs (comma-separated): ");
            String[] orderItems = scanner.nextLine().split(",");
            System.out.print("Enter Quantity for each item (comma-separated): ");
            String[] quantities = scanner.nextLine().split(",");
            double totalBillAmount = calculateTotalBill(connection, orderItems, quantities);
            System.out.printf("Total Bill Amount: %.2f%n", totalBillAmount);
            System.out.print("Enter Payment Method (Cash/Card/Online): ");
            String paymentMethod = scanner.nextLine();
            // Insert data into the table
            insertOrder(connection, customerName, String.join(",", orderItems), totalBillAmount,
paymentMethod);
            scanner.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Method to create the customer_orders table if it doesn't exist
    private static void createTable(Connection connection) throws SQLException {
        String createTableSQL = "CREATE TABLE IF NOT EXISTS customer_orders ("
                + "id INT AUTO_INCREMENT PRIMARY KEY, "
                + "customer_name VARCHAR(100), "
                + "order_items VARCHAR(255), "
                + "total_bill_amount DOUBLE, "
                + "payment_method VARCHAR(50))";
        try (Statement statement = connection.createStatement()) {
            statement.execute(createTableSQL);
            System.out.println("Table 'customer_orders' is ready (created if it didn't exist).");
        }
    }
    // Method to insert order details into the database
    private static void insertOrder(Connection connection, String customerName, String orderItems,
double totalBillAmount, String paymentMethod) {
        // SQL query to insert order into the database
        String query = "INSERT INTO customer_orders (customer_name, order_items, total_bill_amount,
payment_method) VALUES (?, ?, ?, ?)";
        try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            // Set parameters for the SQL query
            preparedStatement.setString(1, customerName);
            preparedStatement.setString(2, orderItems);
            preparedStatement.setDouble(3, totalBillAmount);
            preparedStatement.setString(4, paymentMethod);
            // Execute the query
            int rowsInserted = preparedStatement.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Order inserted successfully!");
            } else {
                System.out.println("Failed to insert order.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Method to display the cafe menu from the database
    private static void displayMenu(Connection connection) {
        String query = "SELECT item_id, item_name, description, price FROM menu";
        try (PreparedStatement statement = connection.prepareStatement(query);
             ResultSet resultSet = statement.executeQuery()) {
            System.out.println("Cafe Menu:");
            System.out.println("----------------------------------------------------");
            System.out.println("ID   | Name         | Description      | Price");
            System.out.println("----------------------------------------------------");
            // Iterate over the result set and display each menu item
            while (resultSet.next()) {
                int id = resultSet.getInt("item_id");
                String name = resultSet.getString("item_name");
                String description = resultSet.getString("description");
                double price = resultSet.getDouble("price");
                // Print each item only once
                System.out.printf("%-4d | %-12s | %-16s | %.2f%n", id, name, description, price);
            }
            System.out.println("----------------------------------------------------");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Method to calculate the total bill based on ordered items and their quantities
    private static double calculateTotalBill(Connection connection, String[] orderItems, String[] quantities)
{
        double totalBill = 0.0;
        for (int i = 0; i < orderItems.length; i++) {
            int itemId = Integer.parseInt(orderItems[i].trim());
            int quantity = Integer.parseInt(quantities[i].trim());
            double price = getItemPrice(connection, itemId);
            totalBill += price * quantity;
        }
        return totalBill;
    }
    // Method to get the price of a menu item by its ID
    private static double getItemPrice(Connection connection, int itemId) {
        double price = 0.0;
        String query = "SELECT price FROM menu WHERE item_id = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            preparedStatement.setInt(1, itemId);
            ResultSet resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                price = resultSet.getDouble("price");
            } else {
                System.out.println("Item ID " + itemId + " not found in the menu.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return price;
    }
}
```

MYSQL OUTPUT:



	item_id	item_name	description	price	available
▶	1	Coffee	Hot brewed coffee	100.00	1
	2	Cappuccino	Espresso-based coffee drink	150.00	1
	3	Tea	Hot brewed tea	80.00	1
	4	Sandwich	Grilled cheese sandwich	120.00	1
	5	Pastry	Fresh baked pastry	60.00	1
*	NULL	NULL	NULL	NULL	NULL

OUTPUT:

```
Table 'customer_orders' is ready (created if it didn't exist).

Cafe Menu:

----------------------------------------------------

ID   | Name        | Description      | Price

----------------------------------------------------

1    | Coffee      | Hot brewed coffee | 100.00

2    | Cappuccino   | Espresso-based coffee drink | 150.00

3    | Tea         | Hot brewed tea   | 80.00

4    | Sandwich     | Grilled cheese sandwich | 120.00

5    | Pastry       | Fresh baked pastry | 60.00

----------------------------------------------------

Enter Customer Name: Nandini

Enter Order Item IDs (comma-separated): 5,1

Enter Quantity for each item (comma-separated): 1,1

Total Bill Amount: 160.00

Enter Payment Method (Cash/Card/Online): online

Order inserted successfully!


Table 'customer_orders' is ready (created if it didn't exist).

Cafe Menu:

----------------------------------------------------

ID   | Name        | Description      | Price

----------------------------------------------------
```

```
1   | Coffee      | Hot brewed coffee | 100.00

2   | Cappuccino  | Espresso-based coffee drink | 150.00

3   | Tea         | Hot brewed tea  | 80.00

4   | Sandwich    | Grilled cheese sandwich | 120.00

5   | Pastry      | Fresh baked pastry | 60.00

----------------------------------------------------

Enter Customer Name: Neha

Enter Order Item IDs (comma-separated): 2,5

Enter Quantity for each item (comma-separated): 1,1

Total Bill Amount: 210.00

Enter Payment Method (Cash/Card/Online): card

Order inserted successfully!


Table 'customer_orders' is ready (created if it didn't exist).

Cafe Menu:

----------------------------------------------------

ID  | Name        | Description     | Price

----------------------------------------------------

1   | Coffee      | Hot brewed coffee | 100.00

2   | Cappuccino  | Espresso-based coffee drink | 150.00

3   | Tea         | Hot brewed tea  | 80.00

4   | Sandwich    | Grilled cheese sandwich | 120.00

5   | Pastry      | Fresh baked pastry | 60.00

----------------------------------------------------

Enter Customer Name: Poojitha

Enter Order Item IDs (comma-separated): 3,4

Enter Quantity for each item (comma-separated): 1,2

Total Bill Amount: 320.00

Enter Payment Method (Cash/Card/Online): online

Order inserted successfully!
```