Object-Oriented Programming (Java)

PROJECT TITLE :

"RESTAURANT ORDERING SYSTEM"

Final Semester Project Report

---

1. Introduction

The Restaurant Ordering System is an Object-Oriented Programming (OOP) based Java application designed to automate food ordering operations in a restaurant. The system allows customers to place orders from a predefined menu, records ordered items, generates bills, and applies taxes to calculate the final payable amount. This project demonstrates the use of OOP concepts to build a modular and maintainable system.

---

2. Objectives of the Project

The main objectives of this project are:

- To display a restaurant menu with food items and prices
- To allow customers to place food orders
- To record and manage ordered items
- To generate bills automatically
- To calculate total amount including tax
- To implement Object-Oriented Programming concepts in Java

---

3. Scope of the Project

This project focuses on basic restaurant operations such as menu handling, order placement, and billing. It can be extended to support databases, multiple customers, payment gateways, and graphical user interfaces.

---

4. OOP Concepts Used

4.1 Encapsulation

Class data members are kept private and accessed through public methods.

4.2 Abstraction

Complex operations such as billing and order processing are handled internally by classes.

4.3 Composition

The Order class uses MenuItem objects, and the Bill class uses Order objects.

4.4 Polymorphism

Methods are designed in a reusable manner for future extension.

---

5. Class Description

5.1 MenuItem Class

Stores details of food items available in the restaurant.

Attributes:

- itemId
- itemName
- price

5.2 Customer Class

Represents a customer who places orders.

Attributes:

- customerId
- customerName

5.3 Order Class

Records the list of ordered food items.

Attributes:

- orderId
- orderedItems

5.4 Bill Class

Calculates total bill amount including tax.

Attributes:

- taxRate
- totalAmount

---

6. System Workflow

1. Menu items are displayed
2. Customer selects food items
3. Order is created and items are recorded
4. Bill is generated
5. Tax is applied to calculate final amount

---

7. Advantages of the System

- Fast and accurate billing
- Reduces manual errors
- Easy to maintain and extend
- Proper use of OOP principles

---

8. Source Code Implementation

8.1 MenuItem.java

```
public class MenuItem {
    private int id;
    private String name;
    private double price;
    private String category;

    public MenuItem(int id, String name, double price, String category) {
```

```
        this.id = id;
        this.name = name;
        this.price = price;
        this.category = category;
    }

    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
    public String getCategory() {
        return category;
    }

    public String toString() {
        return id+" "+name+" $"+price;
    }
}
```

8.2 Customer.java

```
public class Customer {
    private int id;
    private String name;
    private String phone;

    public Customer(int id, String name, String phone) {
        this.id = id;
        this.name = name;
        this.phone = phone;
    }

    public String getName() {
        return name;
    }

    public String getMaskedPhone() {
        if (phone == null && phone.length() <= 10)
            return "Unknown";
        return "******" + phone.substring(phone.length() - 4);
```

```
    }
}
```

8.3 Order.java

```
import java.util.*;

public class Order {
    private int orderId;
    private Customer customer;
    private ArrayList<MenuItem> items;
    private long orderTime;

    public Order(int orderId, Customer customer) {
        this.orderId = orderId;
        this.customer = customer;
        this.items = new ArrayList<>();
        this.orderTime = System.currentTimeMillis();
    }

    public void addItem(MenuItem item) {
        items.add(item);
    }

    public ArrayList<MenuItem> getItems() {
        return items;
    }
    public Customer getCustomer() {
        return customer;
    }
    public int getId() {
        return orderId;
    }

    // Automatic Status Logic based on Time Elapsed
    public String getStatus() {
        long time = (System.currentTimeMillis() - orderTime) / 1000;

        if (time < 15)
            return "PENDING";

        if (time < 30)
            return "PREPARING";

        return "SERVED";
```

```
    }
}
```

8.4 Bill.java

```
public class Bill {
    private Order order;
    private final double TAX = 0.15;

    public Bill(Order order) {
        this.order = order;
    }

    public void generateBill() {
        double subtotal = 0;
        System.out.println("\n======= FINAL BILL =======");
        System.out.println("Order ID: " + order.getId());
        System.out.println("Customer: " + order.getCustomer().getName());
        System.out.println("Phone:    " + order.getCustomer().getMaskedPhone());
        System.out.println("-----------------------");

        for (MenuItem item : order.getItems()) {
            System.out.printf("%-20s $%.2f\n", item.getName(), item.getPrice());
            subtotal += item.getPrice();
        }

        double tax = subtotal * TAX;
        double total = subtotal + tax;

        System.out.println("-----------------------");
        System.out.println("Subtotal: $" + subtotal);
        System.out.println("Tax:      $" + tax);
        System.out.println("TOTAL:    $" + total);
        System.out.println("=======================\n");
    }
}
```

8.5 Main.java

```
import java.util.*;

public class Main {
```

```
    static ArrayList<MenuItem> menu = new ArrayList<>();
    static ArrayList<Order> orders = new ArrayList<>();
    static int orderCounter = 100;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        initializeMenu();

        while (true) {
            System.out.println("\n=== RESTAURANT ORDERING SYSTEM ===");
            System.out.println("1. View Menu");
            System.out.println("2. Place Order");
            System.out.println("3. Track Order");
            System.out.println("4. Admin Panel");
            System.out.println("5. Exit");
            System.out.print("Select Option: ");

            if (!scanner.hasNextInt()) { scanner.next(); continue; }
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    displayMenu();
                    break;
                case 2:
                    placeOrder(scanner);
                    break;
                case 3:
                    trackOrder(scanner);
                    break;
                case 4:
                    adminLogin(scanner);
                    break;
                case 5:
                    System.out.println("System Exiting...");
                    return;
                default:
                    System.out.println("Invalid Choice!");
            }
        }
    }
```

---

9. Conclusion

The Restaurant Ordering System is a simple and efficient Java-based application that automates the food ordering and billing process. It effectively demonstrates Object-Oriented Programming principles and fulfills all the specified project requirements. The system is suitable for academic purposes and can be enhanced with advanced features in the future.