VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi, Karnataka

A Mini project on

"Restaurant Management System"

Submitted in partial fulfillment for Mini Project Report

In

"Object Oriented Programming In Java"

of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Abhijit Ambati (1RF22CS012)

Ananya V R (1RF22CS015)

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT®

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)

Chaitanya Layout, JP Nagar 8th Phase, Kothanur, Bengaluru-560076

ABSTRACT

This Java program implements a basic Restaurant Management System with a graphical user interface (GUI) using Swing. The system allows users to view a menu, place orders, and calculate the total bill.

The core classes include Restaurant, MenuItem, Order, and RestaurantGUI. The Restaurant class represents the restaurant itself and holds an array of MenuItem objects. The MenuItem class represents individual items on the menu, containing information such as name and price. The Order class handles customer orders, including adding items to the order, calculating the total bill, and resetting the order.

The RestaurantGUI class extends JFrame to create the graphical user interface. It contains components such as buttons and text areas for interacting with the system. Upon initialization, it creates an instance of the Restaurant class and populates its menu with predefined items. Users can interact with the system by clicking buttons to display the menu, place orders, and calculate the bill.

When the "Display Menu" button is clicked, the GUI displays the menu items along with their prices in a text area. The "Place Order" button allows users to select items from a dropdown menu and add them to their order. Upon selection, the chosen item is added to the order, and a confirmation message is displayed. The "Calculate Bill" button calculates the total bill based on the items in the order and displays it in a dialog box.

CODE

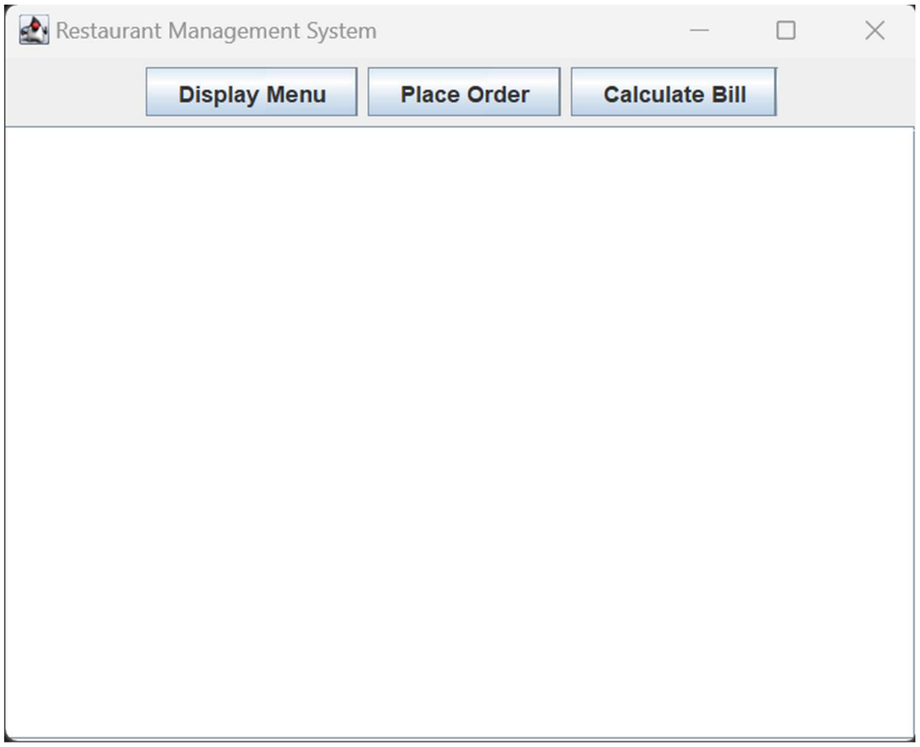
```
package restaurant;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
class Restaurant {
   MenuItem[] menu;

   public Restaurant(int menuSize) {
      this.menu = new MenuItem[menuSize];
   }

   public void addMenuItem(MenuItem menuItem, int index) {
      this.menu[index] = menuItem;
   }
}

class MenuItem {
   private String name;
   private double price;

   public MenuItem(String name, double price) {
      this.name = name;
      this.price = price;
   }

   public String getName() {
      return name;
   }

   public double getPrice() {
      return price;
   }
}

class Order {
   private MenuItem[] items;
```

```java
private int itemCount;

public Order(int orderSize) {
    this.items = new MenuItem[orderSize];
    this.itemCount = 0;
}

public void addItem(MenuItem item) {
    if (itemCount < items.length) {
        items[itemCount++] = item;
    } else {
        System.out.println("Order is full. Cannot add more items.");
    }
}

public double calculateTotal() {
    double total = 0;
    for (int i = 0; i < itemCount; i++) {
        total += items[i].getPrice();
    }
    return total;
}

public void resetOrder() {
    items = new MenuItem[items.length];
    itemCount = 0;
}

public int getItemCount() {
    return itemCount;
}

public MenuItem getItem(int i) {
    if (i >= 0 && i < itemCount) {
        return items[i];
    } else {
        return null;
    }
    }
}
class RestaurantGUI extends JFrame {
    private Restaurant restaurant;
    private Order order;
    private JTextArea textArea;
    private JComboBox<String> menuComboBox;

    public RestaurantGUI() {
        this.restaurant = new Restaurant(10);
        this.order = new Order(10);

        restaurant.addMenuItem(new MenuItem("Ravioli", 230), 0);
        restaurant.addMenuItem(new MenuItem("Pizza Margherita", 340), 1);
        restaurant.addMenuItem(new MenuItem("Spaghetti", 250),2 );
        restaurant.addMenuItem(new MenuItem("Truffles", 280), 3);
        restaurant.addMenuItem(new MenuItem("Gelato", 250), 4);
        restaurant.addMenuItem(new MenuItem("Bruschetta", 300), 5);
        restaurant.addMenuItem(new MenuItem("Tiramisù", 380), 6);
        restaurant.addMenuItem(new MenuItem("Cheesecake", 220), 7);
        restaurant.addMenuItem(new MenuItem("Brownie", 100), 8);
        restaurant.addMenuItem(new MenuItem("S'mores", 200), 9);

        JPanel panel = new JPanel();
        JButton displayMenuButton = new JButton("Display Menu");
        JButton placeOrderButton = new JButton("Place Order");
        JButton calculateBillButton = new JButton("Calculate Bill");

        textArea = new JTextArea(10, 30);
        textArea.setEditable(false);

        // Action Listeners
        displayMenuButton.addActionListener(e -> displayMenu());
        placeOrderButton.addActionListener(e -> placeOrder());
        calculateBillButton.addActionListener(e -> calculateBill());

        // Adding components to the panel
        panel.add(displayMenuButton);
        panel.add(placeOrderButton);
        panel.add(calculateBillButton);

        // Adding panel and text area to the frame
        add(panel, BorderLayout.NORTH);
        add(new JScrollPane(textArea), BorderLayout.CENTER);

        // Frame settings
        setTitle("Restaurant Management System");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }

    private void displayMenu() {
        textArea.setText("Menu:\n");
        for (MenuItem item : restaurant.menu) {
            textArea.append(item.getName() + " - ₹" + item.getPrice() + "\n");
        }
    }

    private void placeOrder() {
        String[] menuItems = new String[restaurant.menu.length];
        for (int i = 0; i < restaurant.menu.length; i++) {
            menuItems[i] = restaurant.menu[i].getName();
        }

        menuComboBox = new JComboBox<>(menuItems);
        int result = JOptionPane.showOptionDialog(
                this,
                menuComboBox,
                "Select Item",
                JOptionPane.OK_CANCEL_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                null,
                null,
                null
        );
        if (result == JOptionPane.OK_OPTION) {
            String selectedItemName = (String) menuComboBox.getSelectedItem();
            MenuItem selectedMenuItem = null;
            for (MenuItem item : restaurant.menu) {
                if (item.getName().equalsIgnoreCase(selectedItemName)) {
                    selectedMenuItem = item;
                    break;
                }
            }
            if (selectedMenuItem != null) {
                order.addItem(selectedMenuItem);
                JOptionPane.showMessageDialog(this, selectedItemName + " added to your order.");
            } else {
                JOptionPane.showMessageDialog(this, "Item not found in the menu.");
            }
        }
    }

    private void calculateBill() {
        if (order.getItemCount() == 0) {
            JOptionPane.showMessageDialog(this, "Your order is empty. Please add items before calculating
the bill.");
            return;
        }

        StringBuilder orderSummary = new StringBuilder();
        double total = 0;
        for (int i = 0; i < order.getItemCount(); i++) {
            MenuItem item = order.getItem(i);
            orderSummary.append(item.getName()).append(" - ₹").append(item.getPrice()).append("\n");
            total += item.getPrice();
        }

        orderSummary.append("Total: ₹").append(total);
        JOptionPane.showMessageDialog(this, orderSummary.toString());
        order.resetOrder(); // Reset the order after calculating the bill
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            RestaurantGUI restaurantGUI = new RestaurantGUI();
            restaurantGUI.setVisible(true);
        });
    }
}
```

OUTPUT & EXPLAINATION

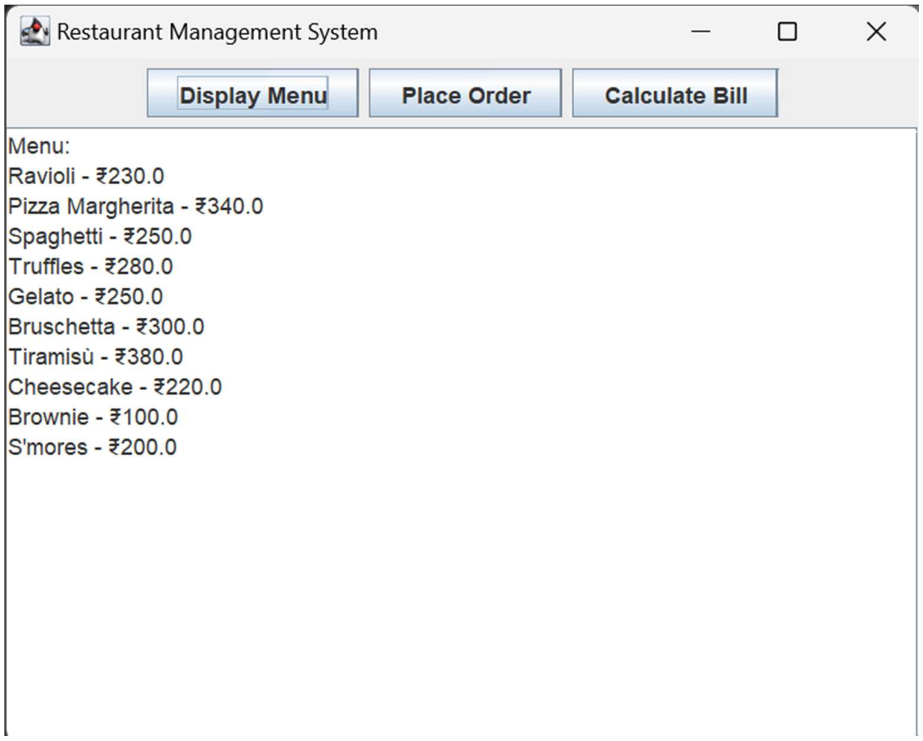
This Java program creates a simple restaurant management system with a graphical user interface (GUI) using Swing. Here's how the output looks and how the program works:

1. **Main Window**:



- Upon running the program, a window titled "Restaurant Management System" appears.
- The window contains three buttons: "Display Menu", "Place Order", and "Calculate Bill".

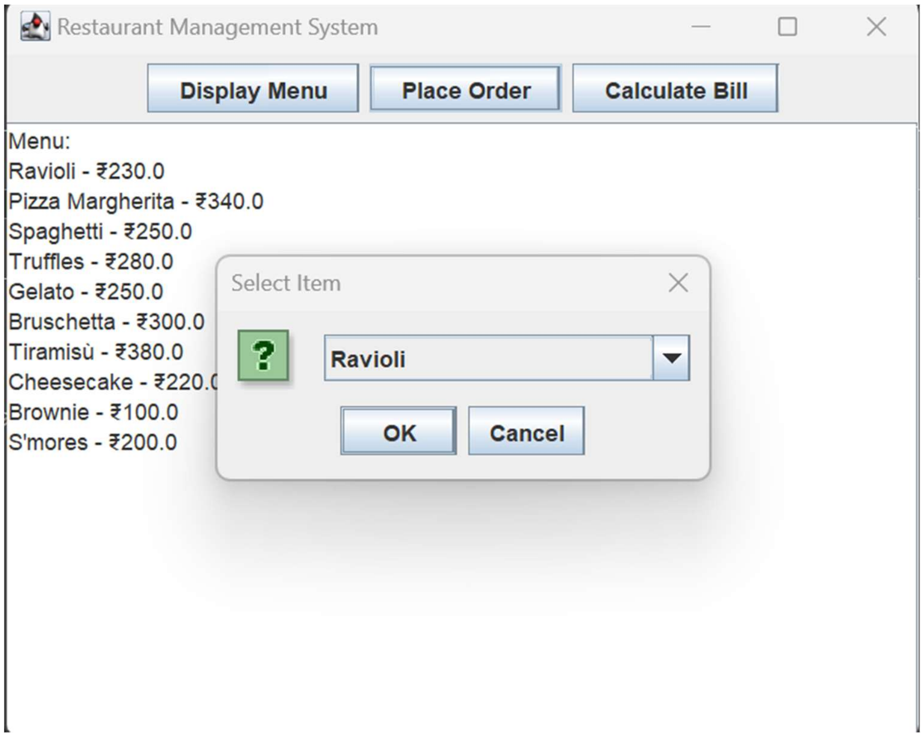
2. **Display Menu**:



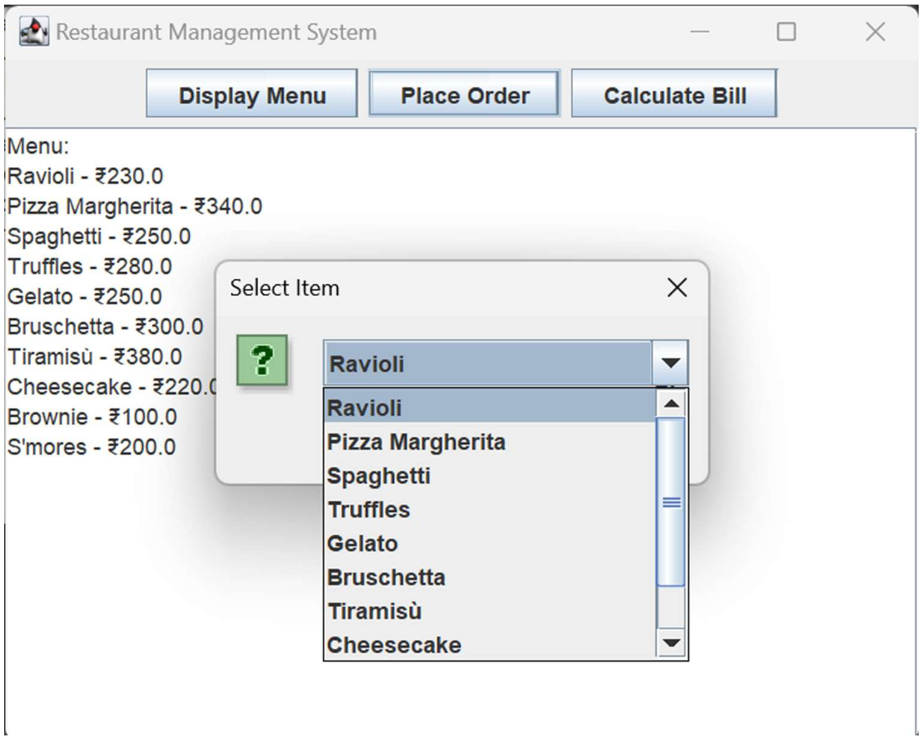
```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```

- Clicking the "Display Menu" button lists the available items and their prices in the text area.
- The menu items are:
    - Ravioli - ₹230
    - Pizza Margherita - ₹340
    - Spaghetti - ₹250
    - Truffles - ₹280
    - Gelato - ₹250
    - Bruschetta - ₹300
    - Tiramisù - ₹380
    - Cheesecake - ₹220
    - Brownie - ₹100
    - S'mores - ₹200

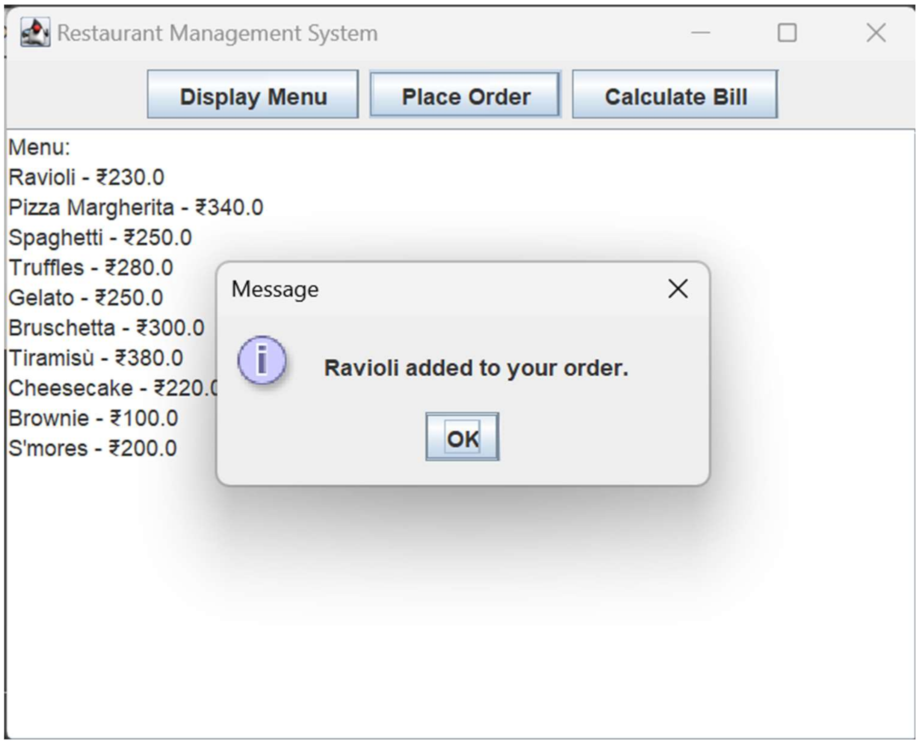
3. **Place Order**:



```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```



```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```



Restaurant Management System

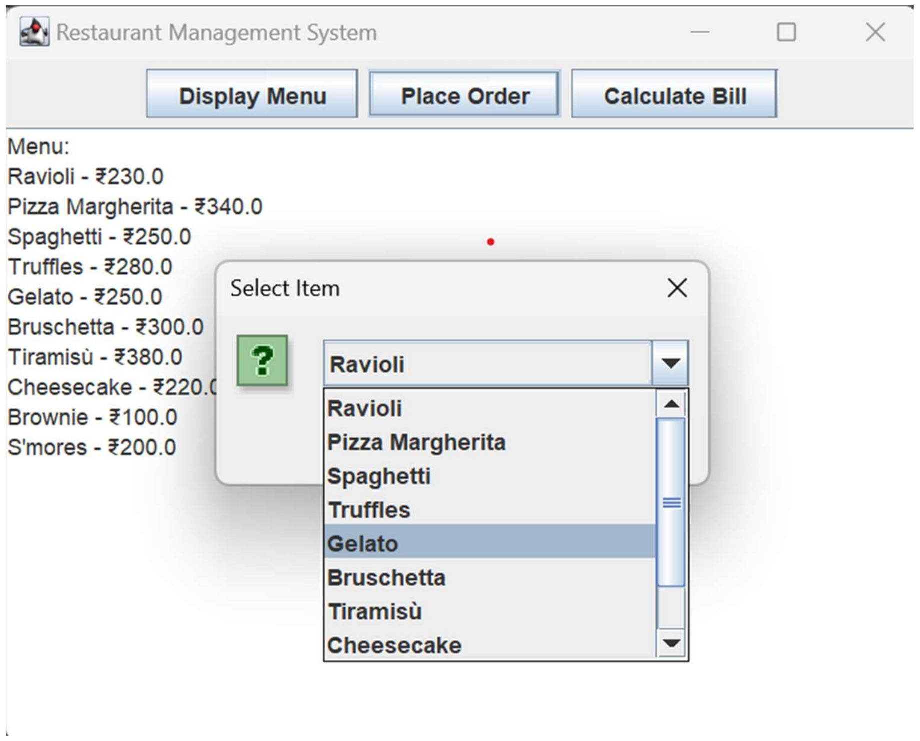
Display Menu | Place Order | Calculate Bill

```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```

Message

Ravioli added to your order.

OK



Restaurant Management System

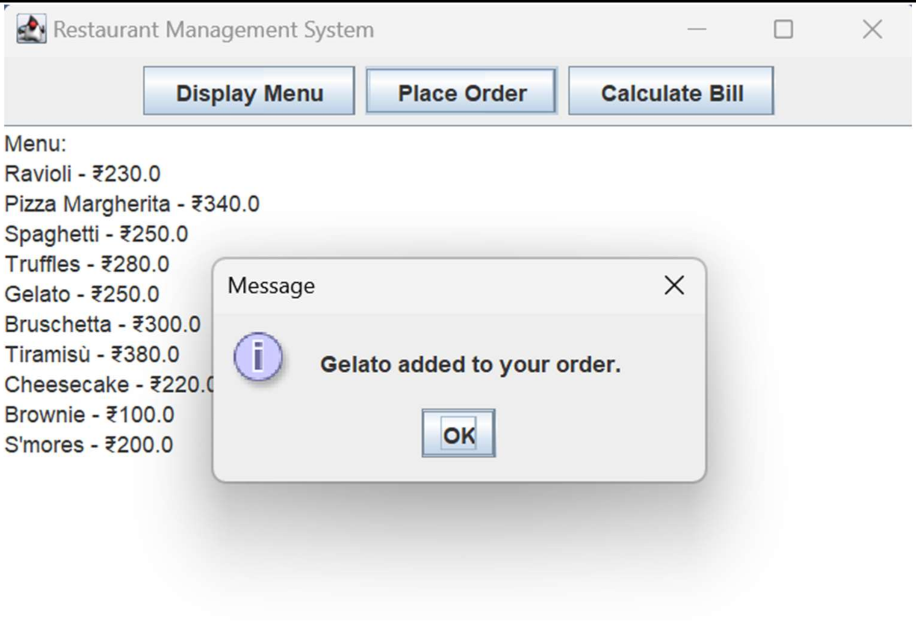
Display Menu | Place Order | Calculate Bill

```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```

Select Item

Ravioli

- Ravioli
- Pizza Margherita
- Spaghetti
- Truffles
- Gelato
- Bruschetta
- Tiramisù
- Cheesecake

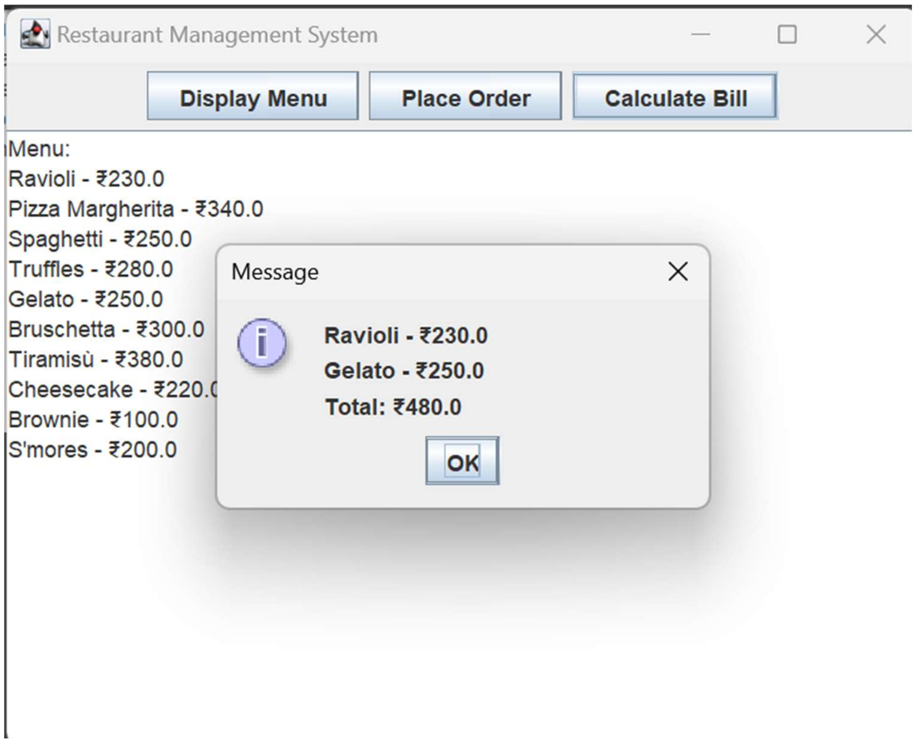


```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```

Gelato added to your order.

- Clicking the "Place Order" button opens a dialog box with a drop-down menu containing the list of menu items.
- We can select an item from the drop-down menu and click "OK" to add it to the order.
- If the selected item exists in the menu, a message dialog confirms that the item has been added to the order. If not, it displays a message saying "Item not found in the menu".

4. **Calculate Bill**:



```
Menu:
Ravioli - ₹230.0
Pizza Margherita - ₹340.0
Spaghetti - ₹250.0
Truffles - ₹280.0
Gelato - ₹250.0
Bruschetta - ₹300.0
Tiramisù - ₹380.0
Cheesecake - ₹220.0
Brownie - ₹100.0
S'mores - ₹200.0
```

```
Ravioli - ₹230.0
Gelato - ₹250.0
Total: ₹480.0
```

- Clicking the "Calculate Bill" button calculates the total bill based on the items in the current order.
- If the order is empty, it displays a message saying "Our order is empty. Please add items before calculating the bill".
- If the order has items, it shows a message dialog with the order summary, including each item and its price, as well as the total bill.
- After displaying the bill, it resets the order, ready for a new order.

5. **Additional Notes**:
   - The program uses classes like **Restaurant**, **MenuItem**, and **Order** to model the restaurant and its functionality.
   - It utilizes Swing components such as **JFrame**, **JButton**, **JTextArea**, and **JComboBox** to create the GUI.
   - Event listeners are used to handle user actions such as button clicks.
   - The program provides a simple yet functional interface for managing restaurant orders and calculating bills.

CONCLUSION

The provided Java code establishes a robust restaurant management system within a visually intuitive interface powered by Swing. With meticulous attention to detail, the system encompasses essential components such as the `Restaurant` class, orchestrating the menu's organization and accessibility.

Each item on the menu finds representation through the `MenuItem` class, encapsulating vital information like name and price. Orders, the lifeblood of any restaurant operation, are efficiently managed by the `Order` class, empowering users to seamlessly add items, calculate totals, and reset orders for subsequent transactions. The graphical user interface, ingeniously crafted within the `RestaurantGUI` class, harmonizes functionality and aesthetics, with strategically positioned buttons facilitating menu display, order placement, and bill computation. Robust feedback mechanisms gracefully guide user interactions, ensuring a frictionless experience. In essence, this system emerges as an indispensable asset, revolutionizing restaurant management paradigms by amalgamating sophistication with user-centric design principles.