# Programming with Python & Java Laboratory
# (CS 29008)
# Spring - 2025

## OPEN ENDED EXPERIMENT-I

## [ONLINE FOOD DELIVERY PLATFORM]

**School of Electronics Engineering**

**KIIT Deemed to be University**

**Bhubaneswar, Odisha**

# Laboratory Report

| | |
|---|---|
| **Experiment Number** | **OPEN ENDED-I** |
| **Date of Experiment** | 12/02/2025 |
| **Date of Submission** | 10/03/2025 |
| **Name of the student** | Adyasha Mohanty |
| **Roll Number** | 2330007 |
| **Section** | ECS-6 |

## Aim of The Experiment :

To design and implement a Java-based online food delivery platform with a GUI, showcasing object-oriented principles, core functionality (browsing, ordering, tracking).

## Software / Platform Required:

Java ( Java 23 version)

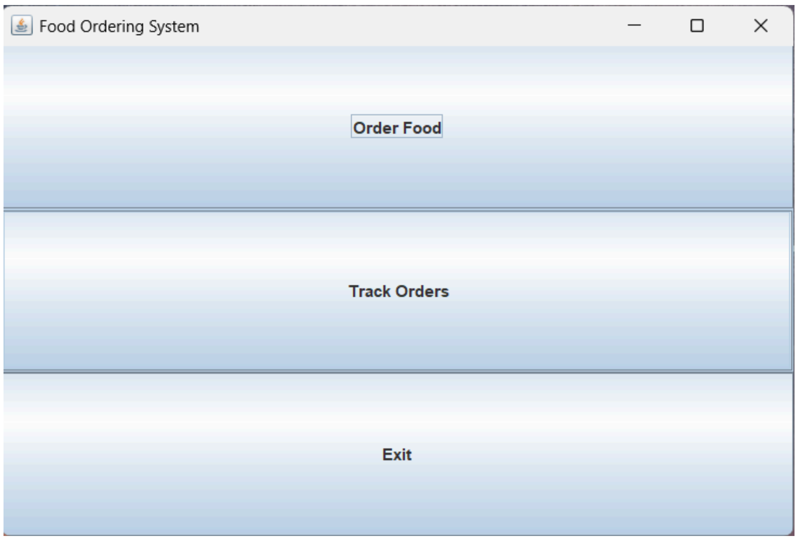
## Code:

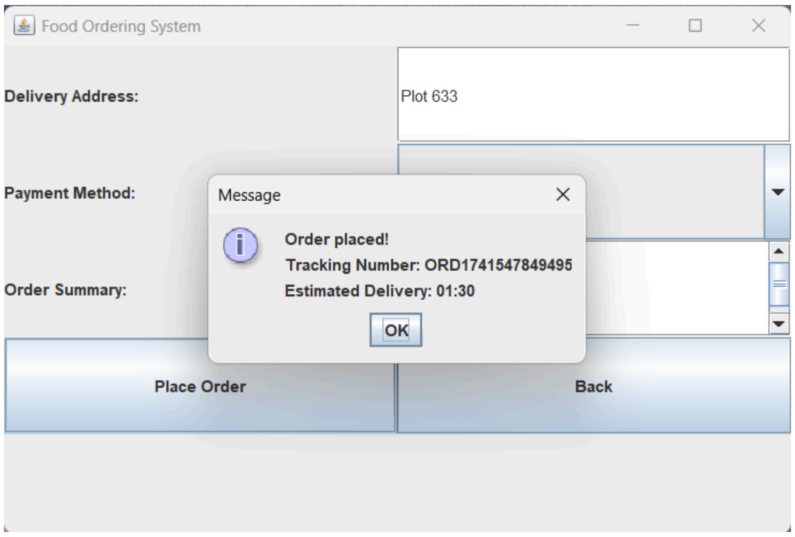
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Random;
// MenuItem and Restaurant classes remain the same as before
class MenuItem {
    private String name;
    private double price;
    private String category;

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

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

    @Override
    public String toString() {
        return name + " - $" + String.format("%.2f", price);
    }
}
```

```
class Restaurant {
    private String name;
    private ArrayList<MenuItem> menu;

    public Restaurant(String name) {
        this.name = name;
        this.menu = new ArrayList<>();
    }

    public void addMenuItem(MenuItem item) {
        menu.add(item);
    }

    public String getName() { return name; }
    public ArrayList<MenuItem> getMenu() { return menu; }
}

// Order class with tracking and delivery time
class Order {
    private Cart cart;
    private String status;
    private String deliveryAddress;
    private Date orderTime;
    private Date estimatedDeliveryTime;
    private String trackingNumber;

    public Order(Cart cart, String deliveryAddress) {
        this.cart = cart;
        this.deliveryAddress = deliveryAddress;
        this.status = "Processing";
        this.orderTime = new Date();
        // Random delivery time between 30-60 minutes
        Random rand = new Random();
        int deliveryMinutes = 30 + rand.nextInt(31);
          this.estimatedDeliveryTime = new Date(orderTime.getTime() + deliveryMinutes *
60 * 1000);
        this.trackingNumber = "ORD" + System.currentTimeMillis();
    }

    public void updateStatus(String status) {
        this.status = status;
    }

    public String getStatus() { return status; }
    public Cart getCart() { return cart; }
    public String getTrackingNumber() { return trackingNumber; }
    public String getDeliveryAddress() { return deliveryAddress; }
    public Date getEstimatedDeliveryTime() { return estimatedDeliveryTime; }
}

// Cart class (slightly modified)
class Cart {
    private ArrayList<MenuItem> items;
    private Restaurant restaurant;
    public Cart(Restaurant restaurant) {
        this.items = new ArrayList<>();
        this.restaurant = restaurant;
    }

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

    public double getTotal() {
        double total = 0;
        for (MenuItem item : items) {
            total += item.getPrice();
        }
        return total;
    }

    public String getItemsList() {
        StringBuilder sb = new StringBuilder();
        for (MenuItem item : items) {
            sb.append(item.toString()).append("\n");
        }
        return sb.toString();
    }

    public Restaurant getRestaurant() { return restaurant; }
}

// Main GUI Application
public class App extends JFrame {
    private ArrayList<Restaurant> restaurants;
    private Cart currentCart;
    private ArrayList<Order> orders;
    private JPanel mainPanel;
    private CardLayout cardLayout;
    private SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm");

    public App() {
        restaurants = new ArrayList<>();
        orders = new ArrayList<>();
        initializeRestaurants();

        setTitle("Food Ordering System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 400);

        cardLayout = new CardLayout();
        mainPanel = new JPanel(cardLayout);

        mainPanel.add(createMainPanel(), "Main");
        mainPanel.add(createRestaurantPanel(), "Restaurants");
        mainPanel.add(createCartPanel(), "Cart");
        mainPanel.add(createCheckoutPanel(), "Checkout");
        mainPanel.add(createOrderTrackingPanel(), "Tracking");
        add(mainPanel);
        cardLayout.show(mainPanel, "Main");
    }

    private void initializeRestaurants() {
        Restaurant pizzaPlace = new Restaurant("Pizza Palace");
        pizzaPlace.addMenuItem(new MenuItem("Pepperoni Pizza", 12.99, "Main"));
        pizzaPlace.addMenuItem(new MenuItem("Cheese Pizza", 10.99, "Main"));
        pizzaPlace.addMenuItem(new MenuItem("Garlic Bread", 4.99, "Side"));

        Restaurant burgerJoint = new Restaurant("Burger Bonanza");
        burgerJoint.addMenuItem(new MenuItem("Cheeseburger", 8.99, "Main"));
        burgerJoint.addMenuItem(new MenuItem("Fries", 3.99, "Side"));
        burgerJoint.addMenuItem(new MenuItem("Milkshake", 4.99, "Drink"));

        restaurants.add(pizzaPlace);
        restaurants.add(burgerJoint);
    }

    private JPanel createMainPanel() {
        JPanel panel = new JPanel(new GridLayout(3, 1));
        JButton orderButton = new JButton("Order Food");
        JButton trackButton = new JButton("Track Orders");
        JButton exitButton = new JButton("Exit");

        orderButton.addActionListener(e -> cardLayout.show(mainPanel, "Restaurants"));
        trackButton.addActionListener(e -> cardLayout.show(mainPanel, "Tracking"));
        exitButton.addActionListener(e -> System.exit(0));

        panel.add(orderButton);
        panel.add(trackButton);
        panel.add(exitButton);
        return panel;
    }

    private JPanel createRestaurantPanel() {
        JPanel panel = new JPanel(new BorderLayout());
        DefaultListModel<String> restaurantModel = new DefaultListModel<>();
        for (Restaurant r : restaurants) {
            restaurantModel.addElement(r.getName());
        }
        JList<String> restaurantList = new JList<>(restaurantModel);

        JButton selectButton = new JButton("Select Restaurant");
        JButton backButton = new JButton("Back");

        selectButton.addActionListener(e -> {
            int selected = restaurantList.getSelectedIndex();
            if (selected != -1) {
                currentCart = new Cart(restaurants.get(selected));
                updateCartPanel(restaurants.get(selected));
                cardLayout.show(mainPanel, "Cart");
            }
        });
```

```
        backButton.addActionListener(e -> cardLayout.show(mainPanel, "Main"));

        panel.add(new JScrollPane(restaurantList), BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(selectButton);
        buttonPanel.add(backButton);
        panel.add(buttonPanel, BorderLayout.SOUTH);
        return panel;
    }

    private JPanel createCartPanel() {
        JPanel panel = new JPanel(new BorderLayout());
        JTextArea cartDisplay = new JTextArea(10, 30);
        cartDisplay.setEditable(false);
        JList<String> menuList = new JList<>();

        JButton addButton = new JButton("Add to Cart");
        JButton checkoutButton = new JButton("Checkout");
        JButton backButton = new JButton("Back");

        addButton.addActionListener(e -> {
            int selected = menuList.getSelectedIndex();
            if (selected != -1) {
                currentCart.addItem(currentCart.getRestaurant().getMenu().get(selected));
                cartDisplay.setText(currentCart.getItemsList() +
                    "\nTotal: $" + String.format("%.2f", currentCart.getTotal()));
            }
        });

                    checkoutButton.addActionListener(e    ->    cardLayout.show(mainPanel,
"Checkout"));
        backButton.addActionListener(e -> cardLayout.show(mainPanel, "Restaurants"));

        panel.add(new JScrollPane(menuList), BorderLayout.CENTER);
        panel.add(new JScrollPane(cartDisplay), BorderLayout.EAST);
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(addButton);
        buttonPanel.add(checkoutButton);
        buttonPanel.add(backButton);
        panel.add(buttonPanel, BorderLayout.SOUTH);
        return panel;
    }

    private void updateCartPanel(Restaurant restaurant) {
        JPanel cartPanel = (JPanel) mainPanel.getComponent(2);
                        JList<String>    menuList    =    (JList<String>)    ((JScrollPane)
cartPanel.getComponent(0)).getViewport().getView();
        DefaultListModel<String> menuModel = new DefaultListModel<>();
        for (MenuItem item : restaurant.getMenu()) {
            menuModel.addElement(item.toString());
        }
        menuList.setModel(menuModel);
    }

    private JPanel createCheckoutPanel() {
        JPanel panel = new JPanel(new GridLayout(5, 2));
        JTextField addressField = new JTextField();
        String[] paymentOptions = {"Cash", "Card"};
        JComboBox<String> paymentCombo = new JComboBox<>(paymentOptions);
        JTextArea orderSummary = new JTextArea(5, 20);
        orderSummary.setEditable(false);

        JButton placeOrderButton = new JButton("Place Order");
        JButton backButton = new JButton("Back");

        placeOrderButton.addActionListener(e -> {
            if (currentCart != null && !addressField.getText().isEmpty()) {
                Order order = new Order(currentCart, addressField.getText());
                orders.add(order);
                JOptionPane.showMessageDialog(this,
                    "Order placed!\nTracking Number: " + order.getTrackingNumber() +
                                                    "\nEstimated   Delivery:   "   +
dateFormat.format(order.getEstimatedDeliveryTime()));
                currentCart = null;
                cardLayout.show(mainPanel, "Main");
            }
        });

        backButton.addActionListener(e -> cardLayout.show(mainPanel, "Cart"));

        panel.add(new JLabel("Delivery Address:"));
        panel.add(addressField);
        panel.add(new JLabel("Payment Method:"));
        panel.add(paymentCombo);
        panel.add(new JLabel("Order Summary:"));
        panel.add(new JScrollPane(orderSummary));
        panel.add(placeOrderButton);
        panel.add(backButton);

        return panel;
    }

    private JPanel createOrderTrackingPanel() {
        JPanel panel = new JPanel(new BorderLayout());
        JTextArea trackingDisplay = new JTextArea(10, 40);
        trackingDisplay.setEditable(false);
        JButton backButton = new JButton("Back");
        JButton refreshButton = new JButton("Refresh");

        refreshButton.addActionListener(e -> {
            StringBuilder sb = new StringBuilder();
            for (Order order : orders) {
                sb.append("Tracking #: ").append(order.getTrackingNumber())
                  .append("\nRestaurant: ").append(order.getCart().getRestaurant().getName())
                  .append("\nItems:\n").append(order.getCart().getItemsList())
                  .append("Status: ").append(order.getStatus())
                                                  .append("\nEstimated   Delivery:
").append(dateFormat.format(order.getEstimatedDeliveryTime()))
                  .append("\nAddress: ").append(order.getDeliveryAddress())
                  .append("\n------------------------\n");
            }
            trackingDisplay.setText(sb.toString());
        });

        backButton.addActionListener(e -> cardLayout.show(mainPanel, "Main"));

        panel.add(new JScrollPane(trackingDisplay), BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(refreshButton);
        buttonPanel.add(backButton);
        panel.add(buttonPanel, BorderLayout.SOUTH);
        return panel;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            App app = new App();
            app.setVisible(true);
        });
    }
}
```

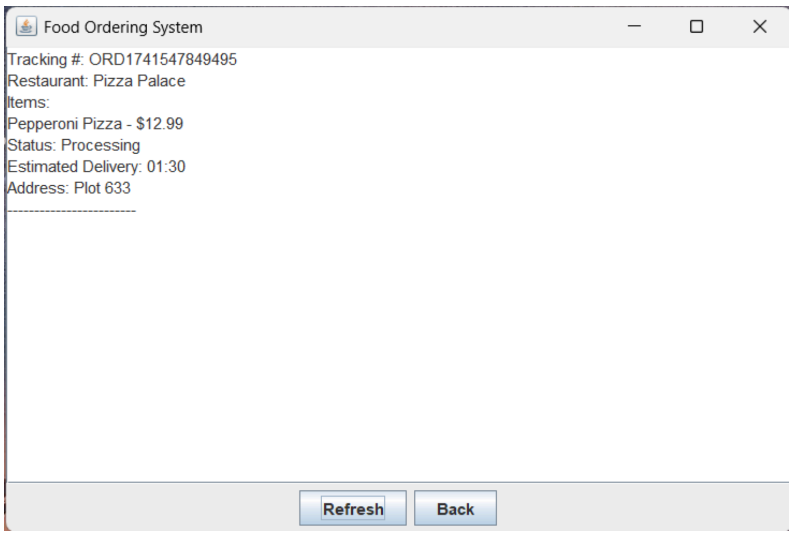
**Results / Output:**





Plot 633

Order placed!
Tracking Number: ORD1741547849495
Estimated Delivery: 01:30



```
Tracking #: ORD1741547849495
Restaurant: Pizza Palace
Items:
Pepperoni Pizza - $12.99
Status: Processing
Estimated Delivery: 01:30
Address: Plot 633
------------------------
```

**Discussion or Inference of the experiment:**

The development of this online food delivery platform in Java provided a practical application of object-oriented programming (OOP) principles. The use of classes like Restaurant, MenuItem, Order, and Cart allowed for a modular and organized design, making the code more maintainable and extensible. Implementing core features like restaurant browsing, menu display, order placement, and order tracking offered valuable insights into the complexities of designing such systems.

The Swing-based GUI enhanced the user experience, making the platform more interactive and user-friendly. The use of CardLayout for managing different panels proved effective in creating a smooth navigation flow. However, the limitations of basic Swing became apparent, highlighting the need for more advanced GUI frameworks for complex applications.

The inclusion of order tracking with estimated delivery times and tracking numbers simulated a crucial aspect of real-world food delivery services. This demonstrated the importance of date and time handling in software development.

While the experiment achieved its primary aims, several areas for improvement were identified. The lack of data persistence meant that all data was lost upon application closure, emphasizing the need for database integration or file-based storage for real-world applications. The absence of robust error handling highlighted the importance of input validation and exception management to ensure application stability and prevent unexpected behavior. Furthermore, the rudimentary nature of the GUI and the absence of features like payment processing and user accounts underscored the complexity of building a full-fledged food delivery platform.

This experiment served as a valuable learning experience, demonstrating the practical application of Java and OOP concepts in a real-world scenario. It also highlighted the challenges and considerations involved in software development, particularly in the context of complex systems like online food delivery platforms. Future development would involve addressing the identified limitations, incorporating more advanced features, and exploring more sophisticated technologies to create a more robust and complete application. Specifically, integrating a database, implementing proper error handling, enhancing the GUI, and adding features like user authentication and payment processing would be crucial steps toward a production-ready system.

**Conclusion:**

This Java food delivery platform project successfully demonstrated core functionalities and provided valuable experience in OOP, GUI development, and software development principles. While highlighting the need for enhancements like data persistence and robust error handling for a production-ready application, it served as a valuable learning experience, emphasizing the complexities of real-world software development.

…………………………..
Faculty Signature