E-Commerce Management System

# AIM:

The aim of this project is to develop a user-friendly E-Commerce Management System that efficiently handles product listings, user cart management, and order processing. It utilizes various data structures to optimize performance for different functionalities, such as maintaining a product catalog, managing user orders, and prioritizing high-demand items. The system also features a visually appealing graphical user interface (GUI) to enhance user experience. Ultimately, this project seeks to streamline online shopping processes and improve customer satisfaction.

# CODING:

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

import java.util.List;

class Product {

    int id;

    String name;

    double price;

    int quantity;

    public Product(int id, String name, double price, int quantity) {

        this.id = id;

        this.name = name;

        this.price = price;

        this.quantity = quantity;

    }

    @Override

    public String toString() {

        return "ID: " + id + ", " + name + " ($" + price + ", Qty: " + quantity + ")";

    }

}

class User {

    int userId;

    String name;

    java.util.List<Product> cart;

    public User(int userId, String name) {

        this.userId = userId;

        this.name = name;

        this.cart = new LinkedList<>();

    }

    public void addToCart(Product product) {

        cart.add(product);

    }

    public void clearCart() {

        cart.clear();

    }

    @Override

    public String toString() {

        return "User: " + name + " (Cart: " + cart.size() + " items)";

    }

}

class Order {

    int orderId;

    User user;

    java.util.List<Product> products;

    boolean isHighPriority;

    String address;

    public Order(int orderId, User user, boolean isHighPriority, String address) {

        this.orderId = orderId;

        this.user = user;

        this.products = new ArrayList<>(user.cart);

        this.isHighPriority = isHighPriority;

        this.address = address;

    }

    @Override

    public String toString() {

        StringBuilder sb = new StringBuilder();

        sb.append("Order ID: ").append(orderId).append("\n")

          .append("User: ").append(user.name).append("\n")

          .append("Address: ").append(address).append("\n")

          .append("High Priority: ").append(isHighPriority).append("\n")

          .append("Products:\n");

        for (Product product : products) {

            sb.append(" - ").append(product).append("\n");

        }

        return sb.toString();

    }

}

public class ECommerceGUI extends JFrame {

    private Map<Integer, Product> productCatalog = new HashMap<>();

    private java.util.List<Order> orderHistory = new ArrayList<>();

    private LinkedList<Order> orderQueue = new LinkedList<>();

    private PriorityQueue<Order> priorityOrderQueue = new PriorityQueue<>(

        Comparator.comparingInt(order -> order.isHighPriority ? -1 : 1)

    );

    private int orderCounter = 1;

    private User user = new User(1, "John Doe");

    public ECommerceGUI() {

        productCatalog.put(1, new Product(1, "Laptop", 1000.0, 5));

        productCatalog.put(2, new Product(2, "Phone", 500.0, 10));

        productCatalog.put(3, new Product(3, "Headphones", 100.0, 15));

        productCatalog.put(4, new Product(4, "Smart Watch", 150.0, 20));

        productCatalog.put(5, new Product(5, "Tablet", 300.0, 12));

        productCatalog.put(6, new Product(6, "Gaming Console", 400.0, 8));

        productCatalog.put(7, new Product(7, "Smart TV", 800.0, 6));

        productCatalog.put(8, new Product(8, "Bluetooth Speaker", 80.0, 25));

        productCatalog.put(9, new Product(9, "Camera", 600.0, 7));

        productCatalog.put(10, new Product(10, "Keyboard", 50.0, 30));

        productCatalog.put(11, new Product(11, "Mouse", 25.0, 35));

        productCatalog.put(12, new Product(12, "Monitor", 200.0, 15));

        productCatalog.put(13, new Product(13, "Router", 60.0, 20));

        productCatalog.put(14, new Product(14, "External Hard Drive", 120.0, 18));

        productCatalog.put(15, new Product(15, "Flash Drive", 15.0, 40));

        setTitle("E-Commerce Management System");

        setSize(800, 600);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new BorderLayout());

        JPanel mainPanel = new JPanel(new BorderLayout());

        mainPanel.setBackground(new Color(230, 240, 255));

        add(mainPanel);

        JPanel productPanel = new JPanel(new GridLayout(0, 1, 5, 5));

        productPanel.setBackground(new Color(200, 220, 240));

        JScrollPane productScroll = new JScrollPane(productPanel);

        productScroll.setBorder(BorderFactory.createTitledBorder("Available Products"));

        mainPanel.add(productScroll, BorderLayout.CENTER);

        for (Product product : productCatalog.values()) {

            JButton productButton = new JButton(product.toString());

            productButton.setBackground(new Color(70, 130, 180));

            productButton.setForeground(Color.WHITE);

            productButton.setFocusPainted(false);

            productButton.setFont(new Font("Arial", Font.PLAIN, 14));

            productButton.addActionListener(e -> {

                user.addToCart(product);

                JOptionPane.showMessageDialog(this, product.name + " added to cart!");

            });

            productPanel.add(productButton);

        }

        JPanel cartPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));

        cartPanel.setBackground(new Color(230, 240, 255));

        JButton viewCartButton = new JButton("View Cart");

        JButton orderButton = new JButton("Order");

        styleButton(viewCartButton);

        styleButton(orderButton);

        viewCartButton.addActionListener(e -> viewCart());

        orderButton.addActionListener(e -> placeOrder());

        cartPanel.add(viewCartButton);

        cartPanel.add(orderButton);

        mainPanel.add(cartPanel, BorderLayout.SOUTH);

        JButton createHighPriorityOrderButton = new JButton("Create High-Priority Order");

        styleButton(createHighPriorityOrderButton);

        createHighPriorityOrderButton.addActionListener(e -> createOrder(true));

        cartPanel.add(createHighPriorityOrderButton);

        JButton processOrdersButton = new JButton("Process Orders");

        styleButton(processOrdersButton);

        processOrdersButton.addActionListener(e -> processOrders());

        cartPanel.add(processOrdersButton);

        setVisible(true);

    }

    private void viewCart() {

        if (user.cart.isEmpty()) {

            JOptionPane.showMessageDialog(this, "Your cart is empty.");

        } else {

            StringBuilder sb = new StringBuilder("Cart Contents:\n");

            for (Product product : user.cart) {

                sb.append(product).append("\n");

            }

            JOptionPane.showMessageDialog(this, sb.toString());

        }

    }

    private void placeOrder() {

        if (user.cart.isEmpty()) {

            JOptionPane.showMessageDialog(this, "Cart is empty. Add products to place an order.");

            return;

        }

        String address = JOptionPane.showInputDialog(this, "Enter delivery address:");

        if (address != null && !address.trim().isEmpty()) {

            Order order = new Order(orderCounter++, user, false, address);

            orderQueue.add(order);

            orderHistory.add(order);

            JOptionPane.showMessageDialog(this, "Order placed successfully!\n" + order);

            user.clearCart();

        } else {

            JOptionPane.showMessageDialog(this, "Address cannot be empty.");

        }

    }

    private void createOrder(boolean isHighPriority) {

        if (user.cart.isEmpty()) {

            JOptionPane.showMessageDialog(this, "Cart is empty. Add products to create an order.");

            return;

        }

        String address = JOptionPane.showInputDialog(this, "Enter delivery address:");

        if (address != null && !address.trim().isEmpty()) {

            Order order = new Order(orderCounter++, user, isHighPriority, address);

            if (isHighPriority) {

                priorityOrderQueue.add(order);

            } else {

                orderQueue.add(order);

            }

            orderHistory.add(order);

            JOptionPane.showMessageDialog(this, "High-priority order created successfully!\n" + order);

            user.clearCart();

        } else {

            JOptionPane.showMessageDialog(this, "Address cannot be empty.");

        }

    }

    private void processOrders() {

        StringBuilder sb = new StringBuilder("Processing Orders:\n");

        while (!priorityOrderQueue.isEmpty() || !orderQueue.isEmpty()) {

            Order order;

            if (!priorityOrderQueue.isEmpty()) {

                order = priorityOrderQueue.poll();

                sb.append("Processing High-Priority Order:\n").append(order).append("\n");

            } else {

                order = orderQueue.poll();

                sb.append("Processing Regular Order:\n").append(order).append("\n");

            }

        }

        JOptionPane.showMessageDialog(this, sb.toString());

    }

    private void styleButton(JButton button) {

        button.setBackground(new Color(70, 130, 180));

        button.setForeground(Color.WHITE);

        button.setFont(new Font("Arial", Font.BOLD, 14));

        button.setFocusPainted(false);

        button.setBorder(BorderFactory.createCompoundBorder(

                BorderFactory.createLineBorder(new Color(50, 100, 150), 1),

                BorderFactory.createEmptyBorder(5, 15, 5, 15)

        ));

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(ECommerceGUI::new);

    }

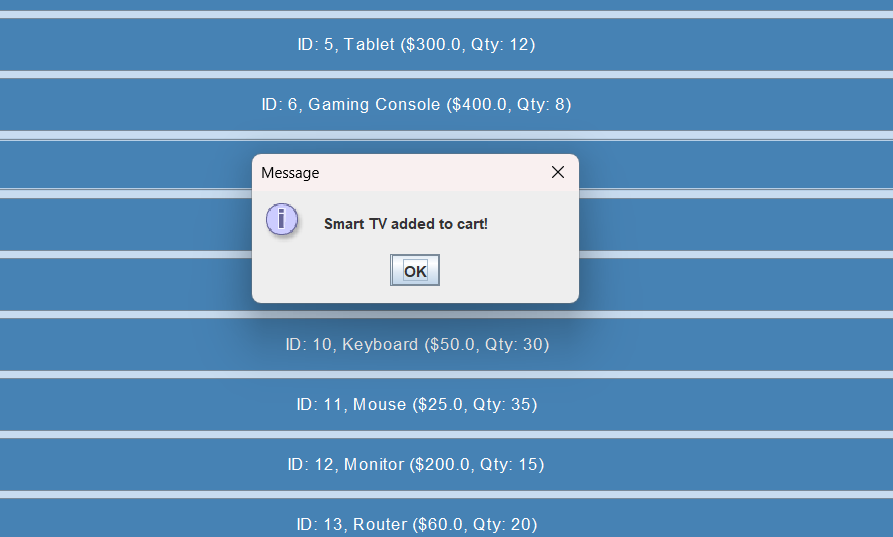
}

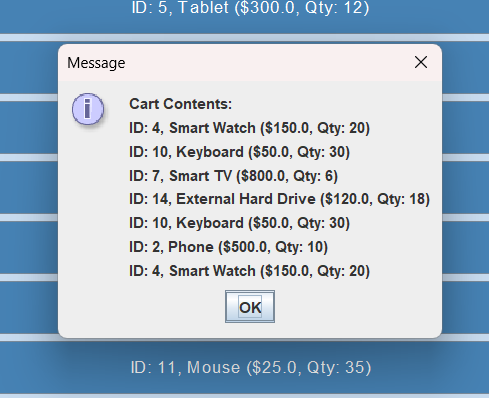
# OUTPUT:

Main GUI:

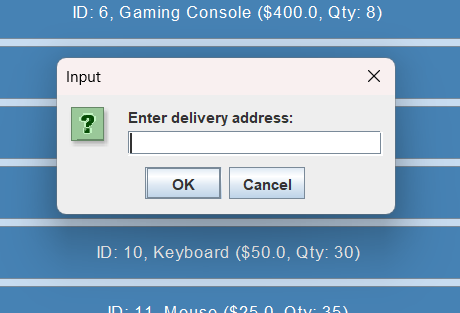


Cart Management:





ORDER PLACEMENT:



# 

# PROCESSING ORDERS:

# 

# RESULT:

Thus, creating a E-Commerce management system using data structure properties has done.