```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// Product class remains unchanged
class Product {
    private String productId;
    private String name;
    private int quantity;
    private double price;

    public Product(String productId, String name, int quantity, double price) {
        this.productId = productId;
        this.name = name;
        this.quantity = quantity;
        this.price = price;
    }

    public String getProductId() {
        return productId;
    }

    public String getName() {
        return name;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public double getPrice() {
        return price;
    }

    public void updateStock(int newQuantity) {
        this.quantity += newQuantity;
```

```
    }

    @Override
    public String toString() {
        return String.format("ID: %s, Name: %s, Quantity: %d, Price: %.2f", productId, name,
quantity, price);
    }
}

// Order class remains unchanged
class Order {
    private String orderId;
    private Map<Product, Integer> orderedProducts;
    private double totalAmount;

    public Order(String orderId) {
        this.orderId = orderId;
        this.orderedProducts = new HashMap<>();
        this.totalAmount = 0.0;
    }

    public void addProductToOrder(Product product, int quantity) {
        if (product.getQuantity() >= quantity) {
            orderedProducts.put(product, quantity);
            product.setQuantity(product.getQuantity() - quantity);
            totalAmount += product.getPrice() * quantity;
        } else {
            System.out.println("Insufficient stock for product: " + product.getName());
        }
    }

    public double getTotalAmount() {
        return totalAmount;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Order ID: ").append(orderId).append("\n");
        for (Map.Entry<Product, Integer> entry : orderedProducts.entrySet()) {
            sb.append(entry.getKey().getName()).append(" - Quantity:
").append(entry.getValue()).append("\n");
        }
        sb.append("Total Amount: ").append(totalAmount).append("\n");
```

```
        return sb.toString();
    }
}


public class Main extends JFrame {
    private List<Product> products;
    private List<Order> orders;

    public Main() {
        products = new ArrayList<>();
        orders = new ArrayList<>();
        initializeUI();
    }



    private void initializeUI() {
        setTitle("Stock Management System");
        setSize(600, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        Font largeFont = new Font("Arial", Font.BOLD, 25);
        // Menu bar
        JMenuBar menuBar = new JMenuBar();

        JMenu menu = new JMenu("Menu"){
            @Override
            public Dimension getPreferredSize() {
                return new Dimension(100, 40); // Set width and height for the menu button

            }
        };
        menu.setFont(largeFont); // Apply larger font to the menu

        // Set the menu bar for the frame
        menuBar.add(menu);
        setJMenuBar(menuBar);
        JMenuItem addProduct = new JMenuItem("Add Product");
        JMenuItem viewProducts = new JMenuItem("View Products");
        JMenuItem updateStock = new JMenuItem("Update Stock");
        JMenuItem createOrder = new JMenuItem("Create Order");
        JMenuItem viewOrders = new JMenuItem("View Orders");
        JMenuItem exit = new JMenuItem("Exit");
```

```
        addProduct.setBackground(Color.CYAN);
        addProduct.setForeground(Color.BLACK);
        viewProducts.setBackground(Color.GREEN);
        viewProducts.setForeground(Color.BLACK);
        updateStock.setBackground(Color.YELLOW);
        updateStock.setForeground(Color.BLACK);
        createOrder.setBackground(Color.ORANGE);
        createOrder.setForeground(Color.BLACK);
        viewOrders.setBackground(Color.PINK);
        viewOrders.setForeground(Color.BLACK);
        exit.setBackground(Color.RED);
        exit.setForeground(Color.WHITE);




        addProduct.setFont(largeFont);
        viewProducts.setFont(largeFont);
        updateStock.setFont(largeFont);
        createOrder.setFont(largeFont);
        viewOrders.setFont(largeFont);
        exit.setFont(largeFont);

        menu.add(addProduct);
        menu.add(viewProducts);
        menu.add(updateStock);
        menu.add(createOrder);
        menu.add(viewOrders);
        menu.addSeparator();
        menu.add(exit);
        menuBar.add(menu);
        setJMenuBar(menuBar);

        // Event Listeners
        addProduct.addActionListener(e -> addProductDialog());
        viewProducts.addActionListener(e -> displayProducts());
        updateStock.addActionListener(e -> updateStockDialog());
        createOrder.addActionListener(e -> createOrderDialog());
        viewOrders.addActionListener(e -> displayOrders());
        exit.addActionListener(e -> System.exit(0));
    }
```

```
    private void addProductDialog() {
        JTextField idField = new JTextField();
        JTextField nameField = new JTextField();
        JTextField qtyField = new JTextField();
        JTextField priceField = new JTextField();

        Object[] message = {
                "Product ID:", idField,
                "Name:", nameField,
                "Quantity:", qtyField,
                "Price:", priceField,
        };

        int option = JOptionPane.showConfirmDialog(this, message, "Add Product",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION) {
            String id = idField.getText();
            String name = nameField.getText();
            int quantity = Integer.parseInt(qtyField.getText());
            double price = Double.parseDouble(priceField.getText());
            products.add(new Product(id, name, quantity, price));
            JOptionPane.showMessageDialog(this, "Product added successfully!");
        }
    }

    private void displayProducts() {
        StringBuilder productList = new StringBuilder("Available Products:\n");
        for (Product product : products) {
            productList.append(product).append("\n");
        }
        JOptionPane.showMessageDialog(this, productList.toString(), "Products",
JOptionPane.INFORMATION_MESSAGE);
    }

    private void updateStockDialog() {
        JTextField idField = new JTextField();
        JTextField qtyField = new JTextField();

        Object[] message = {
                "Product ID:", idField,
                "Quantity to Add:", qtyField,
        };
```

```
        int option = JOptionPane.showConfirmDialog(this, message, "Update Stock",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION) {
            String id = idField.getText();
            int quantity = Integer.parseInt(qtyField.getText());
            Product product = findProductById(id);
            if (product != null) {
                product.updateStock(quantity);
                JOptionPane.showMessageDialog(this, "Stock updated successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Product not found!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    private void createOrderDialog() {
        String orderId = JOptionPane.showInputDialog(this, "Enter Order ID:");
        if (orderId != null && !orderId.isEmpty()) {
            Order order = new Order(orderId);

            while (true) {
                String productId = JOptionPane.showInputDialog(this, "Enter Product ID (or type
'done' to finish):");
                if (productId == null || productId.equalsIgnoreCase("done")) {
                    break;
                }

                Product product = findProductById(productId);
                if (product != null) {
                    String quantityStr = JOptionPane.showInputDialog(this, "Enter Quantity:");
                    int quantity = Integer.parseInt(quantityStr);
                    order.addProductToOrder(product, quantity);
                } else {
                    JOptionPane.showMessageDialog(this, "Product not found!", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }

            if (order.getTotalAmount() > 0) {
                orders.add(order);
                JOptionPane.showMessageDialog(this, "Order created successfully!\n" + order,
"Order", JOptionPane.INFORMATION_MESSAGE);
            } else {
```

```
                JOptionPane.showMessageDialog(this, "No products added to the order.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    private void displayOrders() {
        StringBuilder orderList = new StringBuilder("Orders:\n");
        for (Order order : orders) {
            orderList.append(order).append("\n");
        }
        JOptionPane.showMessageDialog(this, orderList.toString(), "Orders",
JOptionPane.INFORMATION_MESSAGE);
    }

    private Product findProductById(String productId) {
        for (Product product : products) {
            if (product.getProductId().equals(productId)) {
                return product;
            }
        }
        return null;
    }

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