```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

// Product Class - Represents individual product details
class Product {
    private String productId;
    private String name;
    private String category;
    private int quantity;
    private double price;
    private int reorderLevel;

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

    // Method to update stock
    public void updateStock(int change) {
        this.quantity += change;
    }
```

```
    // Check if stock is low
    public boolean isLowStock() {
        return quantity <= reorderLevel;
    }

    // Getters and Setters
    public String getProductId() { return productId; }
    public String getName() { return name; }
    public String getCategory() { return category; }
    public int getQuantity() { return quantity; }
    public double getPrice() { return price; }
    public int getReorderLevel() { return reorderLevel; }

    public void setName(String name) { this.name = name; }
    public void setCategory(String category) { this.category = category; }
    public void setQuantity(int quantity) { this.quantity = quantity; }
    public void setPrice(double price) { this.price = price; }
    public void setReorderLevel(int reorderLevel) { this.reorderLevel = reorderLevel; }

    // Get product details as array for table display
    public Object[] getProductDetails() {
        return new Object[]{
            productId, name, category, quantity, price, reorderLevel
        };
    }
}

// Inventory Management Class
class Inventory {
    private List<Product> productList;
```

```
public Inventory() {
    productList = new ArrayList<>();
}

// Add a new product
public void addProduct(Product product) {
    productList.add(product);
}

// Remove a product by ID
public boolean removeProduct(String productId) {
    return productList.removeIf(p -> p.getProductId().equals(productId));
}

// Update an existing product
public boolean updateProduct(Product updatedProduct) {
    for (int i = 0; i < productList.size(); i++) {
        if (productList.get(i).getProductId().equals(updatedProduct.getProductId())) {
            productList.set(i, updatedProduct);
            return true;
        }
    }
    return false;
}

// Get all products
public List<Product> getAllProducts() {
    return new ArrayList<>(productList);
}
```

```
    // Get low stock products
    public List<Product> getLowStockProducts() {
        return productList.stream()
                .filter(Product::isLowStock)
                .collect(Collectors.toList());
    }

    // Get products by category
    public List<Product> getProductsByCategory(String category) {
        return productList.stream()
                .filter(p -> p.getCategory().equalsIgnoreCase(category))
                .collect(Collectors.toList());
    }
}

// Alert System (Observer Pattern)
class AlertSystem {
    private List<String> alertList;
    private Inventory inventory;

    public AlertSystem(Inventory inventory) {
        this.inventory = inventory;
        this.alertList = new ArrayList<>();
    }

    // Check stock levels and generate alerts
    public List<String> checkStockLevels() {
        alertList.clear();
        List<Product> lowStockProducts = inventory.getLowStockProducts();

        for (Product product : lowStockProducts) {
```

```
            String alert = String.format(
                "LOW STOCK ALERT: %s (ID: %s) - Current Quantity: %d, Reorder Level: %d",
                product.getName(),
                product.getProductId(),
                product.getQuantity(),
                product.getReorderLevel()
            );
            alertList.add(alert);
        }

        return alertList;
    }

    // Get current alerts
    public List<String> getAlerts() {
        return new ArrayList<>(alertList);
    }
}

// Main User Interface Class
public class InventoryManagementSystem extends JFrame {
    private Inventory inventory;
    private AlertSystem alertSystem;
    private JTable productTable;
    private DefaultTableModel tableModel;
    private JTextArea alertArea;

    // Fix the constructor name to match the class name
    public InventoryManagementSystem() {  // Changed from InventoryManagementUI
        // Initialize inventory and alert system
        inventory = new Inventory();
```

```
        alertSystem = new AlertSystem(inventory);

        // Set up the main frame
        setTitle("Inventory Management System");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Create and setup components
        setupProductTable();
        setupControlPanel();
        setupAlertPanel();

        // Add sample products
        addSampleProducts();
    }

    private void setupProductTable() {
        // Table columns
        String[] columnNames = {"Product ID", "Name", "Category", "Quantity", "Price", "Reorder
Level"};
        tableModel = new DefaultTableModel(columnNames, 0);
        productTable = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(productTable);
        add(scrollPane, BorderLayout.CENTER);
    }

    private void setupControlPanel() {
        JPanel controlPanel = new JPanel(new FlowLayout());

        // Add Product Button
```

```
    JButton addButton = new JButton("Add Product");
    addButton.addActionListener(e -> addProductDialog());
    controlPanel.add(addButton);

    // Edit Product Button
    JButton editButton = new JButton("Edit Product");
    editButton.addActionListener(e -> editProductDialog());
    controlPanel.add(editButton);

    // Delete Product Button
    JButton deleteButton = new JButton("Delete Product");
    deleteButton.addActionListener(e -> deleteProduct());
    controlPanel.add(deleteButton);

    // Check Alerts Button
    JButton alertButton = new JButton("Check Alerts");
    alertButton.addActionListener(e -> displayAlerts());
    controlPanel.add(alertButton);

    add(controlPanel, BorderLayout.NORTH);
}

private void setupAlertPanel() {
    alertArea = new JTextArea(5, 50);
    alertArea.setEditable(false);
    JScrollPane alertScrollPane = new JScrollPane(alertArea);
    alertScrollPane.setBorder(BorderFactory.createTitledBorder("Low Stock Alerts"));
    add(alertScrollPane, BorderLayout.SOUTH);
}

private void addSampleProducts() {
```

```
    Product p1 = new Product("P001", "Laptop", "Electronics", 50, 999.99, 10);
    Product p2 = new Product("P002", "Smartphone", "Electronics", 5, 599.99, 20);
    Product p3 = new Product("P003", "Headphones", "Accessories", 15, 99.99, 25);

    inventory.addProduct(p1);
    inventory.addProduct(p2);
    inventory.addProduct(p3);

    refreshProductTable();
}

private void refreshProductTable() {
    // Clear existing rows
    tableModel.setRowCount(0);

    // Add products to table
    for (Product product : inventory.getAllProducts()) {
        tableModel.addRow(product.getProductDetails());
    }
}

private void addProductDialog() {
    JTextField idField = new JTextField(10);
    JTextField nameField = new JTextField(10);
    JTextField categoryField = new JTextField(10);
    JTextField quantityField = new JTextField(10);
    JTextField priceField = new JTextField(10);
    JTextField reorderLevelField = new JTextField(10);

    JPanel panel = new JPanel(new GridLayout(6, 2));
    panel.add(new JLabel("Product ID:"));
```

```
panel.add(idField);
panel.add(new JLabel("Name:"));
panel.add(nameField);
panel.add(new JLabel("Category:"));
panel.add(categoryField);
panel.add(new JLabel("Quantity:"));
panel.add(quantityField);
panel.add(new JLabel("Price:"));
panel.add(priceField);
panel.add(new JLabel("Reorder Level:"));
panel.add(reorderLevelField);

int result = JOptionPane.showConfirmDialog(
    this, panel, "Add New Product", JOptionPane.OK_CANCEL_OPTION
);

if (result == JOptionPane.OK_OPTION) {
    try {
        Product newProduct = new Product(
            idField.getText(),
            nameField.getText(),
            categoryField.getText(),
            Integer.parseInt(quantityField.getText()),
            Double.parseDouble(priceField.getText()),
            Integer.parseInt(reorderLevelField.getText())
        );
        inventory.addProduct(newProduct);
        refreshProductTable();
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(
            this,
```

```
                "Invalid input. Please check your numbers.",
                "Error",
                JOptionPane.ERROR_MESSAGE
            );
        }
    }
}

private void editProductDialog() {
    int selectedRow = productTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(
            this,
            "Please select a product to edit.",
            "No Product Selected",
            JOptionPane.WARNING_MESSAGE
        );
        return;
    }

    String productId = (String) tableModel.getValueAt(selectedRow, 0);
    Product existingProduct = inventory.getAllProducts().stream()
        .filter(p -> p.getProductId().equals(productId))
        .findFirst()
        .orElse(null);

    if (existingProduct == null) return;

    JTextField nameField = new JTextField(existingProduct.getName(), 10);
    JTextField categoryField = new JTextField(existingProduct.getCategory(), 10);
    JTextField quantityField = new JTextField(String.valueOf(existingProduct.getQuantity()), 10);
```

```
    JTextField priceField = new JTextField(String.valueOf(existingProduct.getPrice()), 10);
    JTextField reorderLevelField = new JTextField(String.valueOf(existingProduct.getReorderLevel()),
10);

    JPanel panel = new JPanel(new GridLayout(5, 2));
    panel.add(new JLabel("Name:"));
    panel.add(nameField);
    panel.add(new JLabel("Category:"));
    panel.add(categoryField);
    panel.add(new JLabel("Quantity:"));
    panel.add(quantityField);
    panel.add(new JLabel("Price:"));
    panel.add(priceField);
    panel.add(new JLabel("Reorder Level:"));
    panel.add(reorderLevelField);

    int result = JOptionPane.showConfirmDialog(
       this, panel, "Edit Product", JOptionPane.OK_CANCEL_OPTION
    );

    if (result == JOptionPane.OK_OPTION) {
       try {
          existingProduct.setName(nameField.getText());
          existingProduct.setCategory(categoryField.getText());
          existingProduct.setQuantity(Integer.parseInt(quantityField.getText()));
          existingProduct.setPrice(Double.parseDouble(priceField.getText()));
          existingProduct.setReorderLevel(Integer.parseInt(reorderLevelField.getText()));

          inventory.updateProduct(existingProduct);
          refreshProductTable();
       } catch (NumberFormatException ex) {
```

```
            JOptionPane.showMessageDialog(
                this,
                "Invalid input. Please check your numbers.",
                "Error",
                JOptionPane.ERROR_MESSAGE
            );
        }
    }
}

private void deleteProduct() {
    int selectedRow = productTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(
            this,
            "Please select a product to delete.",
            "No Product Selected",
            JOptionPane.WARNING_MESSAGE
        );
        return;
    }

    String productId = (String) tableModel.getValueAt(selectedRow, 0);

    int confirmDelete = JOptionPane.showConfirmDialog(
        this,
        "Are you sure you want to delete this product?",
        "Confirm Deletion",
        JOptionPane.YES_NO_OPTION
    );
```

```
        if (confirmDelete == JOptionPane.YES_OPTION) {
            inventory.removeProduct(productId);
            refreshProductTable();
        }
    }

    private void displayAlerts() {
        List<String> alerts = alertSystem.checkStockLevels();

        alertArea.setText("");
        if (alerts.isEmpty()) {
            alertArea.append("No low stock alerts at the moment.");
        } else {
            for (String alert : alerts) {
                alertArea.append(alert + "\n");
            }
        }
    }

    // Main method to run the application
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            InventoryManagementSystem app = new InventoryManagementSystem();  // Changed from InventoryManagementUI
            app.setVisible(true);
        });
    }
}
```