

**TRƯỜNG ĐẠI HỌC**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO BÀI TẬP LỚN**

**CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI:**

**QUẢN LÝ BÁN HÀNG CỦA SIÊU THỊ**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: |  |
| Sinh viên thực hiện: |  |
| Thành viên nhóm: |  |
| Lớp: |  |

Hà Nội, năm 2024

**LẬP TRÌNH GIAO DỊCH**

|  |  |
| --- | --- |
| Tên chương trình: QUẢN LÝ BÁN HÀNG CỦA SIÊU THỊ | Mã chương trình:  QLBHCST |
| Tên module: Lập trình giao dịch | Mã tài liệu:  LT2 |
| Tên người làm: | Mã người làm: N3 |
| Thời hạn: Ngày bắt đầu: 10/04/2024 | Ngày kết thúc:  14/05/2024 |
| Ngày nộp: 29/05/2024 | |
| Phiên bản/ File: LT2-240526.docx | |

**NGƯỜI LÀM KÝ XÁC NHẬN**

**MỤC LỤC**

I. GIAO DIỆN 3

1. FormDangNhap 3

2. FormGiaoDich 14

II.THỰC THỂ 42

1. Product 42

2. SalesReceipt 42

3. SalesReceiptDetail 43

III. CONTOLLER 44

1. ProductController 44

2. SalesReceiptController 57

3. SalesReceiptDetailController 59

IV. CONNECT 62

MyConnect 62

V. KẾT QUẢ 63

1. Đăng nhập 63

2. Thêm giao dịch 63

3. Biên lai giao dịch 64

4. Chi tiết biên lai giao dịch 64

# I. GIAO DIỆN

## 1. FormDangNhap

1. /\*  
    \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
    \* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
    \*/  
   package org.example.views;  
     
   import org.example.controllers.UserController;  
   import org.example.models.Role;  
     
   import javax.swing.\*;  
   import java.awt.\*;  
   import java.awt.event.ActionEvent;  
   import java.awt.event.KeyAdapter;  
   import java.awt.event.KeyEvent;  
   import java.awt.event.KeyListener;  
   import java.util.List;  
     
   */\*\*  
    \*  
    \* @author   
    \*/*public class FormDangNhap extends javax.swing.JFrame {  
    private UserController userController; // Thêm biến để truy cập vào controller  
    private final int loggedInUserID; // Declare as final  
     
    public int getLoggedInUserID() {  
    return loggedInUserID;  
    }  
     
    */\*\*  
    \* Creates new form FormDangNhap  
    \*/* public FormDangNhap() {  
    initComponents();  
    userController = new UserController(); // Khởi tạo UserController  
    loggedInUserID = 0; // Initialize the final variable  
    loadRoles(); // Load danh sách quyền khi form được tạo  
    addKeyListeners();  
    addEscapeKeyBinding();  
    }  
     
    private void addEscapeKeyBinding() {  
    String escapeAction = "escapeAction";  
    KeyStroke escapeKeyStroke = KeyStroke.*getKeyStroke*(KeyEvent.*VK\_ESCAPE*, 0, false);  
     
    Action escapeActionHandler = new AbstractAction() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
    Object[] options = {"Thoát", "Không"};  
    int response = JOptionPane.*showOptionDialog*(  
    FormDangNhap.this,  
    "Bạn có muốn thoát chương trình?",  
    "Xác nhận",  
    JOptionPane.*YES\_NO\_OPTION*,  
    JOptionPane.*QUESTION\_MESSAGE*,  
    null,  
    options,  
    options[0]  
    );  
    if (response == JOptionPane.*NO\_OPTION*) {  
    System.*out*.println("không có gì");  
    } else if (response == JOptionPane.*YES\_OPTION*) {  
    // Thoát chương trình  
    System.*out*.println("đã thoát chương trình đăng nhập");  
    System.*exit*(0);  
    }  
    }  
    };  
     
    JRootPane rootPane = this.getRootPane();  
    rootPane.getInputMap(JComponent.*WHEN\_IN\_FOCUSED\_WINDOW*).put(escapeKeyStroke, escapeAction);  
    rootPane.getActionMap().put(escapeAction, escapeActionHandler);  
    }  
     
     
    private void addKeyListeners() {  
    KeyListener keyListener = new KeyAdapter() {  
    @Override  
    public void keyPressed(KeyEvent e) {  
    int keyCode = e.getKeyCode();  
    if (keyCode == KeyEvent.*VK\_ENTER*) {  
    btnDangNhap.doClick(); // Giả lập sự kiện nhấn nút Đăng Nhập  
    }  
   // else if (keyCode == KeyEvent.VK\_ESCAPE) {  
   // dispose(); // Đóng form đăng nhập  
   // }  
    }  
    };  
     
    tbUsername.addKeyListener(keyListener);  
    tbPassword.addKeyListener(keyListener);  
    cbRole.addKeyListener(keyListener);  
    btnDangNhap.addKeyListener(keyListener);  
    }  
    private void loadRoles() {  
    List<Role> roles = userController.getAllRoles();  
    for (Role role : roles) {  
    cbRole.addItem(role.getRoleName());  
    }  
    }  
    */\*\*  
    \* This method is called from within the constructor to initialize the form.  
    \* WARNING: Do NOT modify this code. The content of this method is always  
    \* regenerated by the Form Editor.  
    \*/* @SuppressWarnings("unchecked")  
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
    private void initComponents() {  
     
    jPanel1 = new javax.swing.JPanel();  
    btnDangNhap = new javax.swing.JButton();  
    jLabel4 = new javax.swing.JLabel();  
    jLabel1 = new javax.swing.JLabel();  
    jLabel2 = new javax.swing.JLabel();  
    jLabel3 = new javax.swing.JLabel();  
    cbRole = new javax.swing.JComboBox<>();  
    tbUsername = new javax.swing.JTextField();  
    tbPassword = new javax.swing.JTextField();  
     
    setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
    setBackground(new java.awt.Color(102, 153, 255));  
     
    jPanel1.setBackground(new java.awt.Color(204, 204, 255));  
     
    btnDangNhap.setBackground(new java.awt.Color(102, 102, 255));  
    btnDangNhap.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N  
    btnDangNhap.setForeground(new java.awt.Color(255, 255, 255));  
    btnDangNhap.setText("Đăng nhập");  
    btnDangNhap.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    btnDangNhapActionPerformed(evt);  
    }  
    });  
     
    jLabel4.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N  
    jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.*CENTER*);  
    jLabel4.setText("Đăng Nhập");  
     
    jLabel1.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  
    jLabel1.setText("Tên dăng nhập");  
     
    jLabel2.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  
    jLabel2.setText("Mật khẩu");  
     
    jLabel3.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  
    jLabel3.setText("Quyền");  
     
    cbRole.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    cbRoleActionPerformed(evt);  
    }  
    });  
     
    tbUsername.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    tbUsernameActionPerformed(evt);  
    }  
    });  
     
    tbPassword.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    tbPasswordActionPerformed(evt);  
    }  
    });  
     
    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
    jPanel1.setLayout(jPanel1Layout);  
    jPanel1Layout.setHorizontalGroup(  
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addContainerGap()  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addComponent(jLabel1)  
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
    .addComponent(tbUsername))  
    .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.*TRAILING*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addGap(62, 62, 62)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addGap(31, 31, 31)  
    .addComponent(jLabel3)  
    .addGap(18, 18, 18)  
    .addComponent(cbRole, 0, 454, Short.*MAX\_VALUE*))  
    .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
    .addComponent(jLabel2)  
    .addGap(18, 18, 18)  
    .addComponent(tbPassword))))  
    .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
    .addGap(0, 0, Short.*MAX\_VALUE*)  
    .addComponent(btnDangNhap, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 177, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
    .addContainerGap())  
    );  
    jPanel1Layout.setVerticalGroup(  
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
    .addContainerGap()  
    .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 74, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addGap(66, 66, 66)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addComponent(jLabel1)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addGap(3, 3, 3)  
    .addComponent(tbUsername, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
    .addGap(12, 12, 12)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
    .addComponent(jLabel2)  
    .addComponent(tbPassword, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
    .addComponent(cbRole, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addComponent(jLabel3))  
    .addGap(86, 86, 86)  
    .addComponent(btnDangNhap, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 48, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addContainerGap())  
    );  
     
    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
    getContentPane().setLayout(layout);  
    layout.setHorizontalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(layout.createSequentialGroup()  
    .addContainerGap()  
    .addComponent(jPanel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
    );  
    layout.setVerticalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(layout.createSequentialGroup()  
    .addContainerGap()  
    .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
    .addContainerGap())  
    );  
     
    pack();  
    }// </editor-fold>//GEN-END:initComponents  
     
    private void tbUsernameActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbUsernameActionPerformed  
    // *TODO add your handling code here:* }//GEN-LAST:event\_tbUsernameActionPerformed  
     
    private void tbPasswordActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbPasswordActionPerformed  
    // *TODO add your handling code here:* }//GEN-LAST:event\_tbPasswordActionPerformed  
     
    private void cbRoleActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_cbRoleActionPerformed  
    // *TODO add your handling code here:* }//GEN-LAST:event\_cbRoleActionPerformed  
     
    private void btnDangNhapActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnDangNhapActionPerformed  
    // *TODO add your handling code here:* String username = tbUsername.getText();  
    String password = tbPassword.getText();  
    String selectedRole = cbRole.getSelectedItem().toString();  
     
    // Attempt login  
    boolean loggedIn = userController.loginWithRole(username, password, selectedRole);  
     
    if (loggedIn) {  
    // Retrieve the user's ID after successful login  
    int userID = userController.getUserID(username);  
     
    // Save the user ID in the session  
    userController.setUserIDInSession(userID);  
     
    // Update the loggedInUserID  
     
    JOptionPane.*showMessageDialog*(this, "Đăng nhập thành công với quyền: " + selectedRole);  
    System.*out*.println("Đăng nhập thành công với các thông tin");  
    // Redirect to the appropriate page based on the user's role  
    switch(selectedRole) {  
    case "bộ phận nhân sự":  
    FormNhanSu formNhanSu = new FormNhanSu();  
    formNhanSu.setVisible(true);  
    System.*out*.println("Chức Vụ: bộ phận nhân sự");  
    break;  
    case "bộ phận giao dịch":  
    FormGiaoDich formCungUng = new FormGiaoDich(userID);  
    formCungUng.setVisible(true);  
    System.*out*.println("Chức Vụ: bộ phận giao dịch");  
    break;  
    case "quản lý hàng":  
    FormQuanLyHang formQuanLyHang = new FormQuanLyHang(userID);  
    formQuanLyHang.setVisible(true);  
    System.*out*.println("Chức Vụ: quản lý hàng");  
    break;  
    case "kế toán":  
    FormKeToan formKeToan = new FormKeToan(userID);  
    formKeToan.setVisible(true);  
    System.*out*.println("Chức Vụ: kế toán");  
    break;  
    case "Admin":  
    FormAdmin formAdmin = new FormAdmin();  
    formAdmin.setVisible(true);  
    break;  
    default:  
    JOptionPane.*showMessageDialog*(this, "Không tìm thấy trang tương ứng cho quyền này");  
    }  
    System.*out*.println("UserName: " + username);  
    System.*out*.println("UserID: " + userID);  
     
    this.dispose(); // Close the login form  
    } else {  
    JOptionPane.*showMessageDialog*(this, "Đăng nhập thất bại");  
    }  
    }//GEN-LAST:event\_btnDangNhapActionPerformed  
     
    */\*\*  
    \* @param args the command line arguments  
    \*/* public static void main(String args[]) {  
    /\* Set the Nimbus look and feel \*/  
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
    /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
    \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
    \*/  
    try {  
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
    if ("Nimbus".equals(info.getName())) {  
    javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
    break;  
    }  
    }  
    } catch (ClassNotFoundException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    } catch (InstantiationException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    } catch (IllegalAccessException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    }  
    //</editor-fold>  
     
    /\* Create and display the form \*/  
    java.awt.EventQueue.*invokeLater*(new Runnable() {  
    public void run() {  
    new FormDangNhap().setVisible(true);  
    }  
    });  
    }  
     
    // Variables declaration - do not modify//GEN-BEGIN:variables  
    private javax.swing.JButton btnDangNhap;  
    private javax.swing.JComboBox<String> cbRole;  
    private javax.swing.JLabel jLabel1;  
    private javax.swing.JLabel jLabel2;  
    private javax.swing.JLabel jLabel3;  
    private javax.swing.JLabel jLabel4;  
    private javax.swing.JPanel jPanel1;  
    private javax.swing.JTextField tbPassword;  
    private javax.swing.JTextField tbUsername;  
    // End of variables declaration//GEN-END:variables  
   }

## 2. FormGiaoDich

/\*  
 \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
 \* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
 \*/  
package org.example.views;  
  
import org.example.controllers.\*;  
import org.example.models.Product;  
import org.example.models.SalesReceipt;  
import org.example.models.SalesReceiptDetail;  
import org.example.util.ImageRenderer;  
  
import javax.swing.\*;  
import javax.swing.event.DocumentEvent;  
import javax.swing.event.DocumentListener;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.KeyEvent;  
import java.sql.SQLException;  
import java.util.Date;  
  
*/\*\*  
 \* @author   
 \*/*public class FormGiaoDich extends javax.swing.JFrame {  
 private DefaultTableModel tableModel;  
 private InventoryReceiptController inventoryReceiptController;  
 private SalesReceiptController salesReceiptController;  
 private SalesReceiptDetailController salesReceiptDetailController;  
 private ProductController productController;  
 private SupplierController supplierController;  
 private int loggedInUserID;  
 private int employeeID; // Thêm biến để lưu trữ EmployeeID  
  
 */\*\*  
 \* Creates new form FormCungUng  
 \*/* public FormGiaoDich(int loggedInUserID) {  
 initComponents();  
 this.loggedInUserID = loggedInUserID;  
  
 inventoryReceiptController = new InventoryReceiptController();  
 salesReceiptController = new SalesReceiptController();  
 salesReceiptDetailController = new SalesReceiptDetailController();  
 productController = new ProductController();  
 supplierController = new SupplierController();  
// showPhieuBanHang();  
// showPhieuNhapHang();  
 // Lấy thông tin EmployeeID từ UserController  
 UserController userController = new UserController();  
 employeeID = userController.getEmployeeIDByUserID(loggedInUserID);  
 System.*out*.println("EmployeeID: " + employeeID);  
 showChiTietBienLaiBanHang();  
 showSanPham();  
 showBienLaiBanHang();  
 addEscapeKeyBinding();  
 System.*out*.println("Đang Quản Lý Giao Dịch Với UserID: " + loggedInUserID); // In giá trị loggedInUserID  
  
 tbSoLuong.getDocument().addDocumentListener(new DocumentListener() {  
 @Override  
 public void insertUpdate(DocumentEvent e) {  
 updateTotalPrice();  
 }  
  
 @Override  
 public void removeUpdate(DocumentEvent e) {  
 updateTotalPrice();  
 }  
  
 @Override  
 public void changedUpdate(DocumentEvent e) {  
 updateTotalPrice();  
 }  
 });  
 }  
  
 private void showBienLaiBanHang() {  
 salesReceiptController = new SalesReceiptController();  
 tableModel = salesReceiptController.getAllPhieuBanHang();  
 TableBienLaiBanHang.setModel(tableModel);  
 }  
  
 private void showChiTietBienLaiBanHang() {  
 salesReceiptDetailController = new SalesReceiptDetailController();  
 tableModel = salesReceiptDetailController.getAllSalesReceiptDetail();  
 TableChiTietBienLaiBanHang.setModel(tableModel);  
 }  
  
 private void addEscapeKeyBinding() {  
 String escapeAction = "escapeAction";  
 KeyStroke escapeKeyStroke = KeyStroke.*getKeyStroke*(KeyEvent.*VK\_ESCAPE*, 0, false);  
  
 Action escapeActionHandler = new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Object[] options = {"Đăng xuất", "Thoát", "Không"};  
 int response = JOptionPane.*showOptionDialog*(  
 FormGiaoDich.this,  
 "Bạn có muốn đăng xuất hay thoát chương trình?",  
 "Xác nhận",  
 JOptionPane.*YES\_NO\_CANCEL\_OPTION*,  
 JOptionPane.*QUESTION\_MESSAGE*,  
 null,  
 options,  
 options[0]  
 );  
  
 if (response == JOptionPane.*YES\_OPTION*) {  
 System.*out*.println("Bộ phận quản lý giao dịch đã đăng xuất");  
 // Xử lý đăng xuất  
 btnDangXuatActionPerformed(null);  
 } else if (response == JOptionPane.*NO\_OPTION*) {  
 // Thoát chương trình  
 System.*out*.println("Đã thoát chương trình");  
 System.*exit*(0);  
 } else if (response == JOptionPane.*CANCEL\_OPTION*) {  
 System.*out*.println("Không có gì");  
 }  
 }  
 };  
  
 JRootPane rootPane = this.getRootPane();  
 rootPane.getInputMap(JComponent.*WHEN\_IN\_FOCUSED\_WINDOW*).put(escapeKeyStroke, escapeAction);  
 rootPane.getActionMap().put(escapeAction, escapeActionHandler);  
 }  
  
 private void showSanPham() {  
 productController = new ProductController();  
 tableModel = productController.getAllProductGiaoDichDetails();  
 TableSanPham.setModel(tableModel);  
 ImageRenderer imageRenderer = new ImageRenderer();  
 TableSanPham.getColumnModel().getColumn(11).setCellRenderer(imageRenderer);  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 private void initComponents() {  
  
 jPanel1 = new javax.swing.JPanel();  
 jTabbedPane1 = new javax.swing.JTabbedPane();  
 jPanel3 = new javax.swing.JPanel();  
 jScrollPane1 = new javax.swing.JScrollPane();  
 TableSanPham = new javax.swing.JTable();  
 jPanel2 = new javax.swing.JPanel();  
 jScrollPane2 = new javax.swing.JScrollPane();  
 TableChiTietBienLaiBanHang = new javax.swing.JTable();  
 jPanel4 = new javax.swing.JPanel();  
 jScrollPane3 = new javax.swing.JScrollPane();  
 TableBienLaiBanHang = new javax.swing.JTable();  
 jLabel1 = new javax.swing.JLabel();  
 btnDangXuat = new javax.swing.JButton();  
 jPanel5 = new javax.swing.JPanel();  
 jLabel4 = new javax.swing.JLabel();  
 tbTongGiaTien = new javax.swing.JTextField();  
 jLabel5 = new javax.swing.JLabel();  
 tbSoLuong = new javax.swing.JTextField();  
 ComboBoxEmployeeID = new javax.swing.JComboBox<>();  
 jPanel6 = new javax.swing.JPanel();  
 tbTenSanPham = new javax.swing.JTextField();  
 jLabel2 = new javax.swing.JLabel();  
 jLabel6 = new javax.swing.JLabel();  
 tbMaSanPham = new javax.swing.JTextField();  
 tbGiaTien = new javax.swing.JTextField();  
 jLabel3 = new javax.swing.JLabel();  
 btnGiaoDich = new javax.swing.JButton();  
 jPanel7 = new javax.swing.JPanel();  
 jLabel8 = new javax.swing.JLabel();  
 tbPhuongThucThanhToan = new javax.swing.JTextField();  
 tbTenKhachHang = new javax.swing.JTextField();  
 jLabel7 = new javax.swing.JLabel();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 TableSanPham.setFont(new java.awt.Font("Arial", 1, 20)); // NOI18N  
 TableSanPham.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
 {},  
 {},  
 {},  
 {}  
 },  
 new String [] {  
  
 }  
 ));  
 TableSanPham.setRowHeight(75);  
 TableSanPham.setSelectionBackground(new java.awt.Color(51, 153, 255));  
 TableSanPham.setShowHorizontalLines(true);  
 TableSanPham.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseClicked(java.awt.event.MouseEvent evt) {  
 TableSanPhamMouseClicked(evt);  
 }  
 });  
 jScrollPane1.setViewportView(TableSanPham);  
  
 javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);  
 jPanel3.setLayout(jPanel3Layout);  
 jPanel3Layout.setHorizontalGroup(  
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jScrollPane1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 769, Short.*MAX\_VALUE*)  
 );  
 jPanel3Layout.setVerticalGroup(  
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel3Layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 359, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
  
 jTabbedPane1.addTab("Sản Phẩm", jPanel3);  
  
 TableChiTietBienLaiBanHang.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
 {},  
 {},  
 {},  
 {}  
 },  
 new String [] {  
  
 }  
 ));  
 jScrollPane2.setViewportView(TableChiTietBienLaiBanHang);  
  
 javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);  
 jPanel2.setLayout(jPanel2Layout);  
 jPanel2Layout.setHorizontalGroup(  
 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel2Layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane2, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 757, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
 jPanel2Layout.setVerticalGroup(  
 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel2Layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane2, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 359, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
  
 jTabbedPane1.addTab("Chi tiết biên lai bán hàng", jPanel2);  
  
 TableBienLaiBanHang.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
 {},  
 {},  
 {},  
 {}  
 },  
 new String [] {  
  
 }  
 ));  
 jScrollPane3.setViewportView(TableBienLaiBanHang);  
  
 javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);  
 jPanel4.setLayout(jPanel4Layout);  
 jPanel4Layout.setHorizontalGroup(  
 jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel4Layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane3, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 757, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
 jPanel4Layout.setVerticalGroup(  
 jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel4Layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 355, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(10, Short.*MAX\_VALUE*))  
 );  
  
 jTabbedPane1.addTab("Biên lai bán hàng", jPanel4);  
  
 jLabel1.setFont(new java.awt.Font("Arial", 0, 36)); // NOI18N  
 jLabel1.setText("Bộ Phận Giao dịch");  
  
 btnDangXuat.setBackground(new java.awt.Color(220, 53, 69));  
 btnDangXuat.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnDangXuat.setForeground(new java.awt.Color(255, 255, 255));  
 btnDangXuat.setText("Đăng Xuất");  
 btnDangXuat.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnDangXuatActionPerformed(evt);  
 }  
 });  
  
 jLabel4.setText("Tổng giá tiền");  
  
 tbTongGiaTien.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbTongGiaTienActionPerformed(evt);  
 }  
 });  
  
 jLabel5.setText("Số lượng");  
  
 tbSoLuong.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbSoLuongActionPerformed(evt);  
 }  
 });  
  
 ComboBoxEmployeeID.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 ComboBoxEmployeeIDActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);  
 jPanel5.setLayout(jPanel5Layout);  
 jPanel5Layout.setHorizontalGroup(  
 jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel5Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel5Layout.createSequentialGroup()  
 .addComponent(jLabel4, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 78, Short.*MAX\_VALUE*)  
 .addGap(12, 12, 12)  
 .addComponent(tbTongGiaTien, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 131, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel5Layout.createSequentialGroup()  
 .addComponent(jLabel5)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(ComboBoxEmployeeID, javax.swing.GroupLayout.Alignment.*TRAILING*, 0, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(tbSoLuong))))  
 .addContainerGap())  
 );  
 jPanel5Layout.setVerticalGroup(  
 jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel5Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(tbSoLuong, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel5))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addComponent(ComboBoxEmployeeID, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(8, 8, 8)  
 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel4)  
 .addComponent(tbTongGiaTien, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addContainerGap(11, Short.*MAX\_VALUE*))  
 );  
  
 tbTenSanPham.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbTenSanPhamActionPerformed(evt);  
 }  
 });  
  
 jLabel2.setText("Tên sản phẩm");  
  
 jLabel6.setText("Mã sản phẩm");  
  
 tbMaSanPham.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbMaSanPhamActionPerformed(evt);  
 }  
 });  
  
 tbGiaTien.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbGiaTienActionPerformed(evt);  
 }  
 });  
  
 jLabel3.setText("Giá tiền");  
  
 javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);  
 jPanel6.setLayout(jPanel6Layout);  
 jPanel6Layout.setHorizontalGroup(  
 jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel6Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(jLabel6)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addGap(18, 49, Short.*MAX\_VALUE*)  
 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(tbGiaTien, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 95, Short.*MAX\_VALUE*)  
 .addComponent(tbMaSanPham)  
 .addComponent(tbTenSanPham))  
 .addContainerGap())  
 );  
 jPanel6Layout.setVerticalGroup(  
 jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel6Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel6Layout.createSequentialGroup()  
 .addGap(51, 51, 51)  
 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel2)  
 .addComponent(tbTenSanPham, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel6)  
 .addComponent(tbMaSanPham, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addGap(7, 7, 7)  
 .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(tbGiaTien, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel3))  
 .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 );  
  
 btnGiaoDich.setBackground(new java.awt.Color(0, 123, 255));  
 btnGiaoDich.setFont(new java.awt.Font("Arial", 1, 18)); // NOI18N  
 btnGiaoDich.setForeground(new java.awt.Color(255, 255, 255));  
 btnGiaoDich.setText("Giao dịch");  
 btnGiaoDich.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnGiaoDichActionPerformed(evt);  
 }  
 });  
  
 jLabel8.setText("Phương thức thanh toán");  
  
 tbPhuongThucThanhToan.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbPhuongThucThanhToanActionPerformed(evt);  
 }  
 });  
  
 tbTenKhachHang.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbTenKhachHangActionPerformed(evt);  
 }  
 });  
  
 jLabel7.setText("Tên khách hàng");  
  
 javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);  
 jPanel7.setLayout(jPanel7Layout);  
 jPanel7Layout.setHorizontalGroup(  
 jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel7Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel7Layout.createSequentialGroup()  
 .addComponent(jLabel7, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 130, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(tbTenKhachHang, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 126, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel7Layout.createSequentialGroup()  
 .addComponent(jLabel8)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(tbPhuongThucThanhToan, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 126, javax.swing.GroupLayout.*PREFERRED\_SIZE*))))  
 );  
 jPanel7Layout.setVerticalGroup(  
 jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel7Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel7Layout.createSequentialGroup()  
 .addComponent(jLabel7)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, 17, Short.*MAX\_VALUE*)  
 .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel8)  
 .addComponent(tbPhuongThucThanhToan, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(37, 37, 37))  
 .addGroup(jPanel7Layout.createSequentialGroup()  
 .addComponent(tbTenKhachHang, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))))  
 );  
  
 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
 jPanel1.setLayout(jPanel1Layout);  
 jPanel1Layout.setHorizontalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jLabel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addGap(18, 18, 18)  
 .addComponent(btnDangXuat)  
 .addGap(62, 62, 62))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jTabbedPane1)  
 .addContainerGap())  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(btnGiaoDich, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 183, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(0, 0, Short.*MAX\_VALUE*))  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
 .addGap(0, 0, Short.*MAX\_VALUE*)  
 .addComponent(jPanel6, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addComponent(jPanel7, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addComponent(jPanel5, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap())))  
 );  
 jPanel1Layout.setVerticalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addComponent(btnDangXuat)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 47, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addComponent(jTabbedPane1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 406, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jPanel6, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(35, 35, 35)  
 .addComponent(btnGiaoDich, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addComponent(jPanel5, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jPanel7, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addContainerGap())  
 );  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 getContentPane().setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jPanel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 );  
  
 pack();  
 }// </editor-fold>//GEN-END:initComponents  
  
 private void TableSanPhamMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_TableSanPhamMouseClicked  
 // *TODO add your handling code here:* int selectedRow = TableSanPham.getSelectedRow();  
 if (selectedRow >= 0) {  
 String maSanPham = TableSanPham.getValueAt(selectedRow, 0).toString();  
 String tenSanPham = TableSanPham.getValueAt(selectedRow, 1).toString();  
 String giaTien = TableSanPham.getValueAt(selectedRow, 2).toString(); // Adjust the column index as needed  
// String soLuong = TableSanPham.getValueAt(selectedRow, 3).toString(); // Adjust the column index as needed  
  
 tbMaSanPham.setText(maSanPham);  
 tbTenSanPham.setText(tenSanPham);  
 tbGiaTien.setText(giaTien);  
// tbSoLuong.setText(soLuong); // Set the quantity  
  
 // Thêm dữ liệu vào bảng chi tiết biên lai bán hàng  
 DefaultTableModel chiTietModel = (DefaultTableModel) TableChiTietBienLaiBanHang.getModel();  
 chiTietModel.addRow(new Object[]{maSanPham, tenSanPham, giaTien}); // Add quantity to details table  
 }  
 }//GEN-LAST:event\_TableSanPhamMouseClicked  
  
 private void tbSoLuongActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbSoLuongActionPerformed  
 try {  
 int productID = Integer.*parseInt*(tbMaSanPham.getText()); // Lấy ProductID từ giao diện  
 int soLuong = Integer.*parseInt*(tbSoLuong.getText());  
  
 if (soLuong < 0) {  
 throw new NumberFormatException("Số lượng không thể âm");  
 }  
  
 // Kiểm tra số lượng tồn kho  
 int stockQuantity = productController.getStockQuantity(productID);  
 if (soLuong > stockQuantity) {  
 JOptionPane.*showMessageDialog*(this, "Số lượng nhập vào lớn hơn số lượng tồn kho hiện tại.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 tbSoLuong.setText("");  
 return;  
 }  
  
 // Lấy đơn giá từ cơ sở dữ liệu thông qua ProductController  
 double unitPrice = productController.getUnitPrice(productID);  
 double tongGiaTien = unitPrice \* soLuong;  
 System.*out*.println(unitPrice);  
 System.*out*.println(tongGiaTien);  
  
 // Hiển thị tổng giá tiền  
 tbTongGiaTien.setText(String.*valueOf*(tongGiaTien));  
  
 // Xử lý logic khi số lượng hợp lệ  
 System.*out*.println("Số lượng hợp lệ: " + soLuong);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(this, "Vui lòng nhập một số nguyên hợp lệ cho số lượng.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 tbSoLuong.setText("");  
 } catch (SQLException e) {  
 JOptionPane.*showMessageDialog*(this, "Lỗi khi truy xuất dữ liệu từ cơ sở dữ liệu.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 e.printStackTrace();  
 }  
 }//GEN-LAST:event\_tbSoLuongActionPerformed  
  
 private void updateTotalPrice() {  
 try {  
 String productIdText = tbMaSanPham.getText().trim();  
 String quantityText = tbSoLuong.getText().trim();  
  
 if (productIdText.isEmpty() || quantityText.isEmpty()) {  
 tbTongGiaTien.setText("0"); // Set total price to 0 if either product ID or quantity is empty  
 return;  
 }  
  
 int productID = Integer.*parseInt*(productIdText);  
 int soLuong = Integer.*parseInt*(quantityText);  
  
 if (soLuong < 0) {  
 throw new NumberFormatException("Số lượng không thể âm");  
 }  
  
 // Kiểm tra số lượng tồn kho  
 int stockQuantity = productController.getStockQuantity(productID);  
 if (soLuong > stockQuantity) {  
 SwingUtilities.*invokeLater*(() -> {  
 JOptionPane.*showMessageDialog*(this, "Số lượng nhập vào lớn hơn số lượng tồn kho hiện tại.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 tbSoLuong.setText("0"); // Set quantity to 0 if quantity is greater than stock quantity  
 tbTongGiaTien.setText("0"); // Set total price to 0 if quantity is greater than stock quantity  
 });  
 return;  
 }  
  
 // Lấy đơn giá từ cơ sở dữ liệu thông qua ProductController  
 double unitPrice = productController.getUnitPrice(productID);  
 double tongGiaTien = unitPrice \* soLuong;  
  
 // Hiển thị tổng giá tiền  
 tbTongGiaTien.setText(String.*valueOf*(tongGiaTien));  
  
 // Xử lý logic khi số lượng hợp lệ  
 System.*out*.println("Số lượng hợp lệ: " + soLuong);  
 } catch (NumberFormatException e) {  
 SwingUtilities.*invokeLater*(() -> {  
 JOptionPane.*showMessageDialog*(this, "Vui lòng nhập một số nguyên hợp lệ cho số lượng.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 tbSoLuong.setText("0"); // Set quantity to 0 if there's a number format exception  
 tbTongGiaTien.setText("0"); // Set total price to 0 if there's a number format exception  
 });  
 } catch (SQLException e) {  
 SwingUtilities.*invokeLater*(() -> {  
 JOptionPane.*showMessageDialog*(this, "Lỗi khi truy xuất dữ liệu từ cơ sở dữ liệu.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 e.printStackTrace();  
 tbSoLuong.setText("0"); // Set quantity to 0 if there's an SQL exception  
 tbTongGiaTien.setText("0"); // Set total price to 0 if there's an SQL exception  
 });  
 }  
 }  
  
  
 private void tbTongGiaTienActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbTongGiaTienActionPerformed  
 // *TODO add your handling code here:* // Đặt TextBox tbExample thành không thể chỉnh sửa  
 tbTongGiaTien.setEditable(false);  
  
 // Hoặc, nếu bạn muốn hoàn toàn vô hiệu hóa TextBox để ngăn người dùng tương tác với nó  
 tbTongGiaTien.setEnabled(false);  
  
 try {  
 double tongGiaTien = Double.*parseDouble*(tbTongGiaTien.getText());  
 if (tongGiaTien < 0) {  
 throw new NumberFormatException("Tổng giá tiền không thể âm");  
 }  
 // Xử lý logic khi tổng giá tiền hợp lệ  
 System.*out*.println("Tổng giá tiền hợp lệ: " + tongGiaTien);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(this, "Vui lòng nhập một số thập phân hợp lệ cho tổng giá tiền.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 tbTongGiaTien.setText("");  
 }  
 }//GEN-LAST:event\_tbTongGiaTienActionPerformed  
  
 private void tbGiaTienActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbGiaTienActionPerformed  
 // *TODO add your handling code here:* // *TODO add your handling code here:* // Đặt TextBox tbExample thành không thể chỉnh sửa  
  
 // Hoặc, nếu bạn muốn hoàn toàn vô hiệu hóa TextBox để ngăn người dùng tương tác với nó  
  
 }//GEN-LAST:event\_tbGiaTienActionPerformed  
  
 private void tbTenSanPhamActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbTenSanPhamActionPerformed  
 // *TODO add your handling code here:* // Hoặc, nếu bạn muốn hoàn toàn vô hiệu hóa TextBox để ngăn người dùng tương tác với nó  
 }//GEN-LAST:event\_tbTenSanPhamActionPerformed  
  
 private void tbMaSanPhamActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbMaSanPhamActionPerformed  
 // *TODO add your handling code here:* // Hoặc, nếu bạn muốn hoàn toàn vô hiệu hóa TextBox để ngăn người dùng tương tác với nó  
 }//GEN-LAST:event\_tbMaSanPhamActionPerformed  
  
 private void btnGiaoDichActionPerformed(java.awt.event.ActionEvent evt) {  
 try {  
 String productIdText = tbMaSanPham.getText().trim();  
 String quantityText = tbSoLuong.getText().trim();  
 String tenKhachHang = tbTenKhachHang.getText().trim(); // Lấy tên khách hàng từ giao diện  
 String phuongThucThanhToan = tbPhuongThucThanhToan.getText().trim(); // Lấy phương thức thanh toán từ giao diện  
  
 if (productIdText.isEmpty() || quantityText.isEmpty() || tenKhachHang.isEmpty() || phuongThucThanhToan.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "Vui lòng nhập đầy đủ thông tin sản phẩm, số lượng, tên khách hàng và phương thức thanh toán.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 int productID = Integer.*parseInt*(productIdText);  
 int soLuong = Integer.*parseInt*(quantityText);  
  
 if (soLuong < 0) {  
 JOptionPane.*showMessageDialog*(this, "Số lượng không thể âm.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 // Kiểm tra số lượng tồn kho  
 int stockQuantity = productController.getStockQuantity(productID);  
 if (soLuong > stockQuantity) {  
 JOptionPane.*showMessageDialog*(this, "Số lượng nhập vào lớn hơn số lượng tồn kho hiện tại.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 // Lấy đơn giá từ cơ sở dữ liệu thông qua ProductController  
 double unitPrice = productController.getUnitPrice(productID);  
 double tongGiaTien = unitPrice \* soLuong;  
  
 // Tạo chi tiết phiếu bán hàng  
 SalesReceiptDetail salesReceiptDetail = new SalesReceiptDetail();  
 salesReceiptDetail.setProductID(productID);  
 salesReceiptDetail.setQuantity(soLuong);  
 salesReceiptDetail.setUnitPrice(unitPrice);  
 salesReceiptDetail.setTotalPrice(tongGiaTien);  
  
 // Thực hiện giao dịch  
 boolean result = productController.giaoDichSanPham(salesReceiptDetail, tenKhachHang, phuongThucThanhToan, employeeID, loggedInUserID);  
  
 if (result) {  
 JOptionPane.*showMessageDialog*(this, "Giao dịch thành công.", "Thông báo", JOptionPane.*INFORMATION\_MESSAGE*);  
 // Cập nhật lại tổng giá tiền hoặc các thông tin khác nếu cần  
 showChiTietBienLaiBanHang();  
 showBienLaiBanHang();  
 showSanPham();  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Giao dịch thất bại.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(this, "Vui lòng nhập một số nguyên hợp lệ cho số lượng.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 } catch (SQLException e) {  
 JOptionPane.*showMessageDialog*(this, "Lỗi khi truy xuất dữ liệu từ cơ sở dữ liệu.", "Lỗi", JOptionPane.*ERROR\_MESSAGE*);  
 e.printStackTrace();  
 }  
 }   
  
 private void ComboBoxEmployeeIDActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_ComboBoxEmployeeIDActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_ComboBoxEmployeeIDActionPerformed  
  
 private void tbTenKhachHangActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbTenKhachHangActionPerformed  
 // *TODO add your handling code here:* String tenKhachHang = tbTenKhachHang.getText().trim();  
 // You can perform actions based on the customer name entered by the user  
 System.*out*.println("Customer name entered: " + tenKhachHang);  
 }//GEN-LAST:event\_tbTenKhachHangActionPerformed  
  
 private void tbPhuongThucThanhToanActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbPhuongThucThanhToanActionPerformed  
 // *TODO add your handling code here:* String phuongThucThanhToan = tbPhuongThucThanhToan.getText().trim();  
 // You can perform actions based on the payment method entered by the user  
 System.*out*.println("Payment method entered: " + phuongThucThanhToan);  
 }//GEN-LAST:event\_tbPhuongThucThanhToanActionPerformed  
  
 private void btnDangXuatActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnDangXuatActionPerformed  
 // *TODO add your handling code here:* FormDangNhap loginForm = new FormDangNhap(); // Assuming LoginForm is your login screen  
 loginForm.setVisible(true);  
 this.dispose(); // Close the current window  
 }//GEN-LAST:event\_btnDangXuatActionPerformed  
  
 */\*\*  
 \* @param args the command line arguments  
 \*/* public static void main(String args[]) {  
 final int[] loggedInUserID = {0}; // Khởi tạo với một giá trị mặc định  
  
 // Thử lấy ID của người dùng đã đăng nhập từ FormDangNhap  
 try {  
 FormDangNhap form = new FormDangNhap();  
 loggedInUserID[0] = form.getLoggedInUserID(); // Gọi phương thức không tĩnh trên một thể hiện  
 } catch (Exception e) {  
 // Xử lý trường hợp người dùng chưa đăng nhập  
 System.*out*.println("Người dùng chưa đăng nhập. Vui lòng đăng nhập trước.");  
 // Có thể chuyển hướng người dùng đến form đăng nhập hoặc xử lý theo logic của ứng dụng  
 return; // Kết thúc chương trình hoặc thực hiện bất kỳ hành động cần thiết khác  
 }  
  
 // Kiểm tra giá trị của loggedInUserID trước khi sử dụng  
 if (loggedInUserID[0] == 0) {  
 System.*out*.println("Người dùng chưa đăng nhập. Vui lòng đăng nhập trước.");  
 return;  
 }  
 /\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
 \*/  
 try {  
 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
 if ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
 break;  
 }  
 }  
 } catch (ClassNotFoundException ex) {  
 java.util.logging.Logger.*getLogger*(FormGiaoDich.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (InstantiationException ex) {  
 java.util.logging.Logger.*getLogger*(FormGiaoDich.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (IllegalAccessException ex) {  
 java.util.logging.Logger.*getLogger*(FormGiaoDich.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(FormGiaoDich.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 //</editor-fold>  
 //</editor-fold>  
  
 /\* Create and display the form \*/  
 java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new FormGiaoDich(loggedInUserID[0]).setVisible(true);  
 System.*out*.println(loggedInUserID[0]);  
 }  
 });  
 }  
  
 // Variables declaration - do not modify//GEN-BEGIN:variables  
 private javax.swing.JComboBox<String> ComboBoxEmployeeID;  
 private javax.swing.JTable TableBienLaiBanHang;  
 private javax.swing.JTable TableChiTietBienLaiBanHang;  
 private javax.swing.JTable TableSanPham;  
 private javax.swing.JButton btnDangXuat;  
 private javax.swing.JButton btnGiaoDich;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel3;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JLabel jLabel5;  
 private javax.swing.JLabel jLabel6;  
 private javax.swing.JLabel jLabel7;  
 private javax.swing.JLabel jLabel8;  
 private javax.swing.JPanel jPanel1;  
 private javax.swing.JPanel jPanel2;  
 private javax.swing.JPanel jPanel3;  
 private javax.swing.JPanel jPanel4;  
 private javax.swing.JPanel jPanel5;  
 private javax.swing.JPanel jPanel6;  
 private javax.swing.JPanel jPanel7;  
 private javax.swing.JScrollPane jScrollPane1;  
 private javax.swing.JScrollPane jScrollPane2;  
 private javax.swing.JScrollPane jScrollPane3;  
 private javax.swing.JTabbedPane jTabbedPane1;  
 private javax.swing.JTextField tbGiaTien;  
 private javax.swing.JTextField tbMaSanPham;  
 private javax.swing.JTextField tbPhuongThucThanhToan;  
 private javax.swing.JTextField tbSoLuong;  
 private javax.swing.JTextField tbTenKhachHang;  
 private javax.swing.JTextField tbTenSanPham;  
 private javax.swing.JTextField tbTongGiaTien;  
 // End of variables declaration//GEN-END:variables  
}

# II.THỰC THỂ

## 1. Product

package org.example.models;  
  
import lombok.Data;  
  
import java.util.Date;  
@Data  
public class Product {  
 private int productID;  
 private String productName;  
 private String category;  
 private int stockQuantity;  
 private double unitPrice;  
 private int supplierID;  
 private int createdBy;  
 private int updatedBy;  
 private Date updatedAt;  
  
 // Constructors, getters, and setters  
}

## 2. SalesReceipt

package org.example.models;  
  
import lombok.Data;  
  
import java.sql.Date;  
@Data  
public class SalesReceipt {  
 private int salesReceiptID;  
 private int employeeID;  
 private Date saleDate;  
 private double totalAmount;  
 private String customerName;  
 private String paymentMethod;  
 private int createdBy;  
 private int updatedBy;  
 private Date updatedAt;  
  
 // Constructors, getters, and setters  
}

## 3. SalesReceiptDetail

package org.example.models;  
  
import lombok.Data;  
  
import java.util.Date;  
@Data  
public class SalesReceiptDetail {  
 private int detailID;  
 private int salesReceiptID;  
 private int productID;  
 private int quantity;  
 private double unitPrice;  
 private double totalPrice;  
 private int createdBy;  
 private int updatedBy;  
 private Date updatedAt;  
  
 // Constructors, getters, and setters  
}

# III. CONTOLLER

## 1. ProductController

package org.example.controllers;  
  
import org.example.connect.MyConnection;  
import org.example.models.\*;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.math.BigDecimal;  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
public class ProductController extends Component {  
  
 private Connection connection;  
  
 public ProductController() {  
 // Initialize database connection  
 this.connection = MyConnection.*getConnection*();  
 }  
  
 public DefaultTableModel getAllProducts() {  
 String[] columnNames = {"Mã sản phẩm", "Tên sản phẩm", "Danh mục", "Số lượng tồn kho", "Đơn giá", "Mã nhà cung cấp", "Tạo bởi", "Cập nhật bởi", "Thời gian cập nhật"};  
 DefaultTableModel model = new DefaultTableModel(columnNames, 0);  
  
 String sql = "SELECT \* FROM Products";  
  
 try (Connection conn = MyConnection.*getConnection*();  
 PreparedStatement preparedStatement = conn.prepareStatement(sql);  
 ResultSet resultSet = preparedStatement.executeQuery()) {  
  
 while (resultSet.next()) {  
 int productID = resultSet.getInt("ProductID");  
 String productName = resultSet.getString("ProductName");  
 String category = resultSet.getString("Category");  
 int stockQuantity = resultSet.getInt("StockQuantity");  
 double unitPrice = resultSet.getDouble("UnitPrice");  
 int supplierID = resultSet.getInt("SupplierID");  
 int createdBy = resultSet.getInt("CreatedBy");  
 int updatedBy = resultSet.getInt("UpdatedBy");  
 Date updatedAt = resultSet.getTimestamp("UpdatedAt");  
  
 model.addRow(new Object[]{productID, productName, category, stockQuantity, unitPrice, supplierID, createdBy, updatedBy, updatedAt});  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return model;  
 }  
  
  
 public DefaultTableModel getAllProductDetails() {  
 String[] columnNames = {"Mã sản phẩm", "Tên sản phẩm", "Danh mục", "Số lượng tồn kho", "Đơn giá", "Nhà cung cấp", "Tạo bởi", "Người tạo", "Cập nhật bởi", "Người cập nhật", "Thời gian cập nhật", "Hình ảnh"};  
 DefaultTableModel model = new DefaultTableModel(columnNames, 0);  
  
 String sql = "SELECT " +  
 "p.ProductID AS 'Mã sản phẩm', " +  
 "p.ProductName AS 'Tên sản phẩm', " +  
 "p.Category AS 'Danh mục', " +  
 "p.StockQuantity AS 'Số lượng tồn kho', " +  
 "p.UnitPrice AS 'Đơn giá', " +  
 "s.SupplierName AS 'Nhà cung cấp', " +  
 "p.CreatedBy AS 'Tạo bởi', " +  
 "uc.Username AS 'Người tạo', " +  
 "p.UpdatedBy AS 'Cập nhật bởi', " +  
 "uu.Username AS 'Người cập nhật', " +  
 "p.UpdatedAt AS 'Thời gian cập nhật', " +  
 "pi.ImageUrl AS 'Hình ảnh' " +  
 "FROM Products p " +  
 "LEFT JOIN Suppliers s ON p.SupplierID = s.SupplierID " +  
 "LEFT JOIN Users uc ON p.CreatedBy = uc.UserID " +  
 "LEFT JOIN Users uu ON p.UpdatedBy = uu.UserID " +  
 "LEFT JOIN ProductImages pi ON p.ProductID = pi.ProductID";  
  
 try (Connection conn = MyConnection.*getConnection*();  
 PreparedStatement preparedStatement = conn.prepareStatement(sql);  
 ResultSet resultSet = preparedStatement.executeQuery()) {  
  
 while (resultSet.next()) {  
 int productID = resultSet.getInt("Mã sản phẩm");  
 String productName = resultSet.getString("Tên sản phẩm");  
 String category = resultSet.getString("Danh mục");  
 int stockQuantity = resultSet.getInt("Số lượng tồn kho");  
 double unitPrice = resultSet.getDouble("Đơn giá");  
 String supplierName = resultSet.getString("Nhà cung cấp");  
 int createdBy = resultSet.getInt("Tạo bởi");  
 String creatorUsername = resultSet.getString("Người tạo");  
 int updatedBy = resultSet.getInt("Cập nhật bởi");  
 String updaterUsername = resultSet.getString("Người cập nhật");  
 java.util.Date updatedAt = resultSet.getTimestamp("Thời gian cập nhật");  
 String imageUrl = resultSet.getString("Hình ảnh");  
  
 model.addRow(new Object[]{productID, productName, category, stockQuantity, unitPrice, supplierName, createdBy, creatorUsername, updatedBy, updaterUsername, updatedAt, imageUrl});  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return model;  
 }  
  
  
 public boolean addProduct(Product product, ProductImage productImage, int supplierID, int userID) {  
 String insertProductQuery = "INSERT INTO Products (ProductName, Category, StockQuantity, UnitPrice, SupplierID, CreatedBy, UpdatedBy, UpdatedAt) " +  
 "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";  
  
 String insertProductImageQuery = "INSERT INTO ProductImages (ProductID, ImageUrl, CreatedAt) VALUES (?, ?, ?)";  
  
 try {  
 // Thêm sản phẩm  
 PreparedStatement insertProductStmt = connection.prepareStatement(insertProductQuery, Statement.*RETURN\_GENERATED\_KEYS*);  
 insertProductStmt.setString(1, product.getProductName());  
 insertProductStmt.setString(2, product.getCategory());  
 insertProductStmt.setInt(3, product.getStockQuantity());  
 insertProductStmt.setDouble(4, product.getUnitPrice());  
 insertProductStmt.setInt(5, supplierID);  
 insertProductStmt.setInt(6, userID); // Sử dụng userID cho trường CreatedBy  
 insertProductStmt.setInt(7, userID); // Sử dụng userID cho trường UpdatedBy  
 insertProductStmt.setTimestamp(8, new Timestamp(new Date().getTime()));  
 insertProductStmt.executeUpdate();  
  
 // Lấy ID của sản phẩm vừa thêm  
 ResultSet generatedKeys = insertProductStmt.getGeneratedKeys();  
 int productID = 0;  
 if (generatedKeys.next()) {  
 productID = generatedKeys.getInt(1);  
 } else {  
 throw new SQLException("Creating product failed, no ID obtained.");  
 }  
  
 // Thêm hình ảnh sản phẩm  
 PreparedStatement insertProductImageStmt = connection.prepareStatement(insertProductImageQuery);  
 insertProductImageStmt.setInt(1, productID);  
 insertProductImageStmt.setString(2, productImage.getImageUrl());  
 insertProductImageStmt.setTimestamp(3, new Timestamp(new Date().getTime()));  
 insertProductImageStmt.executeUpdate();  
  
 return true;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
  
 public boolean giaoDichSanPham(SalesReceiptDetail salesReceiptDetail, String customerName, String paymentMethod, int employeeID, int userID) {  
 String insertSalesReceiptSQL = "INSERT INTO SalesReceipts (EmployeeID, SaleDate, TotalAmount, CustomerName, PaymentMethod, CreatedBy, UpdatedBy) VALUES (?, ?, ?, ?, ?, ?, ?)";  
 String insertSalesReceiptDetailSQL = "INSERT INTO SalesReceiptDetails (SalesReceiptID, ProductID, Quantity, UnitPrice, TotalPrice, CreatedBy, UpdatedBy) VALUES (?, ?, ?, ?, ?, ?, ?)";  
 String updateProductStockSQL = "UPDATE Products SET StockQuantity = StockQuantity - ? WHERE ProductID = ?";  
  
 try (Connection conn = MyConnection.*getConnection*();  
 PreparedStatement insertSalesReceiptStmt = conn.prepareStatement(insertSalesReceiptSQL, Statement.*RETURN\_GENERATED\_KEYS*);  
 PreparedStatement insertSalesReceiptDetailStmt = conn.prepareStatement(insertSalesReceiptDetailSQL);  
 PreparedStatement updateStmt = conn.prepareStatement(updateProductStockSQL)) {  
  
 // Thực hiện giao dịch trong một transaction  
 conn.setAutoCommit(false);  
  
 // Thêm thông tin phiếu bán hàng vào bảng SalesReceipts  
 insertSalesReceiptStmt.setInt(1, employeeID);  
 // Thay đổi SaleDate thành ngày hiện tại hoặc ngày của giao dịch  
 insertSalesReceiptStmt.setDate(2, new java.sql.Date(System.*currentTimeMillis*()));  
 // Thay đổi TotalAmount thành tổng giá tiền của chi tiết phiếu bán hàng  
 insertSalesReceiptStmt.setBigDecimal(3, BigDecimal.*valueOf*(salesReceiptDetail.getTotalPrice()));  
 insertSalesReceiptStmt.setString(4, customerName); // Sử dụng tên khách hàng từ tham số  
 insertSalesReceiptStmt.setString(5, paymentMethod); // Sử dụng phương thức thanh toán từ tham số  
 insertSalesReceiptStmt.setInt(6, userID);  
 insertSalesReceiptStmt.setInt(7, userID);  
 insertSalesReceiptStmt.executeUpdate();  
  
 // Lấy SalesReceiptID vừa được tạo  
 ResultSet generatedKeys = insertSalesReceiptStmt.getGeneratedKeys();  
 int salesReceiptID = -1;  
 if (generatedKeys.next()) {  
 salesReceiptID = generatedKeys.getInt(1);  
 }  
  
 // Thêm thông tin chi tiết phiếu bán hàng vào bảng SalesReceiptDetails  
 insertSalesReceiptDetailStmt.setInt(1, salesReceiptID);  
 insertSalesReceiptDetailStmt.setInt(2, salesReceiptDetail.getProductID());  
 insertSalesReceiptDetailStmt.setInt(3, salesReceiptDetail.getQuantity());  
 insertSalesReceiptDetailStmt.setBigDecimal(4, BigDecimal.*valueOf*(salesReceiptDetail.getUnitPrice()));  
 insertSalesReceiptDetailStmt.setBigDecimal(5, BigDecimal.*valueOf*(salesReceiptDetail.getTotalPrice()));  
 insertSalesReceiptDetailStmt.setInt(6, userID);  
 insertSalesReceiptDetailStmt.setInt(7, userID);  
 insertSalesReceiptDetailStmt.executeUpdate();  
  
 // Cập nhật số lượng sản phẩm trong bảng Products  
 updateStmt.setInt(1, salesReceiptDetail.getQuantity());  
 updateStmt.setInt(2, salesReceiptDetail.getProductID());  
 updateStmt.executeUpdate();  
  
 // Commit transaction  
 conn.commit();  
  
 return true; // Giao dịch thành công  
  
 } catch (SQLException e) {  
 // Rollback transaction nếu có lỗi  
 try {  
 connection.rollback();  
 } catch (SQLException rollbackEx) {  
 rollbackEx.printStackTrace();  
 }  
 e.printStackTrace();  
 return false; // Giao dịch không thành công  
 }  
 }  
  
  
 private int getInsertedSupplierID() throws SQLException {  
 // Lấy ID của nhà cung cấp vừa được thêm vào  
 String selectLastSupplierIDQuery = "SELECT LAST\_INSERT\_ID()";  
 PreparedStatement selectLastSupplierIDStmt = connection.prepareStatement(selectLastSupplierIDQuery);  
 ResultSet resultSet = selectLastSupplierIDStmt.executeQuery();  
 resultSet.next();  
 return resultSet.getInt(1);  
 }  
  
 private int getInsertedProductID() throws SQLException {  
 // Lấy ID của sản phẩm vừa được thêm vào  
 String selectLastProductIDQuery = "SELECT LAST\_INSERT\_ID()";  
 PreparedStatement selectLastProductIDStmt = connection.prepareStatement(selectLastProductIDQuery);  
 ResultSet resultSet = selectLastProductIDStmt.executeQuery();  
 resultSet.next();  
 return resultSet.getInt(1);  
 }  
  
 public boolean updateProduct(Product product, ProductImage productImage, int supplierID, int userID) {  
 String updateProductQuery = "UPDATE Products SET ProductName = ?, Category = ?, StockQuantity = ?, UnitPrice = ?, SupplierID = ?, UpdatedBy = ?, UpdatedAt = ? WHERE ProductID = ?";  
 String updateProductImageQuery = "UPDATE ProductImages SET ImageUrl = ? WHERE ProductID = ?";  
  
 try {  
 // Bắt đầu giao dịch  
 connection.setAutoCommit(false);  
  
 // Cập nhật thông tin sản phẩm  
 PreparedStatement updateProductStmt = connection.prepareStatement(updateProductQuery);  
 updateProductStmt.setString(1, product.getProductName());  
 updateProductStmt.setString(2, product.getCategory());  
 updateProductStmt.setInt(3, product.getStockQuantity());  
 updateProductStmt.setDouble(4, product.getUnitPrice());  
 updateProductStmt.setInt(5, supplierID);  
 updateProductStmt.setInt(6, userID);  
 updateProductStmt.setTimestamp(7, new Timestamp(new Date().getTime()));  
 updateProductStmt.setInt(8, product.getProductID());  
 updateProductStmt.executeUpdate();  
  
 // Cập nhật thông tin hình ảnh sản phẩm  
 PreparedStatement updateProductImageStmt = connection.prepareStatement(updateProductImageQuery);  
 updateProductImageStmt.setString(1, productImage.getImageUrl());  
 updateProductImageStmt.setInt(2, product.getProductID());  
 updateProductImageStmt.executeUpdate();  
  
 // Hoàn thành giao dịch  
 connection.commit();  
 return true;  
 } catch (SQLException e) {  
 try {  
 // Rollback nếu có lỗi  
 connection.rollback();  
 } catch (SQLException rollbackEx) {  
 rollbackEx.printStackTrace();  
 }  
 e.printStackTrace();  
 return false;  
 } finally {  
 try {  
 // Đặt lại trạng thái tự động commit  
 connection.setAutoCommit(true);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
  
 public boolean deleteProduct(int productID) {  
 String deleteProductImageQuery = "DELETE FROM ProductImages WHERE ProductID = ?";  
 String deletePricingQuery = "DELETE FROM Pricing WHERE ProductID = ?";  
 String deleteProductQuery = "DELETE FROM Products WHERE ProductID = ?";  
 String deleteSalesReceiptDetailsQuery = "DELETE FROM salesreceiptdetails WHERE ProductID = ?";  
  
 try {  
 // Begin transaction  
 connection.setAutoCommit(false);  
  
 // Delete product images  
 PreparedStatement deleteProductImageStmt = connection.prepareStatement(deleteProductImageQuery);  
 deleteProductImageStmt.setInt(1, productID);  
 deleteProductImageStmt.executeUpdate();  
  
 // Delete related pricing rows  
 PreparedStatement deletePricingStmt = connection.prepareStatement(deletePricingQuery);  
 deletePricingStmt.setInt(1, productID);  
 deletePricingStmt.executeUpdate();  
  
 // Delete related sales receipt details  
 PreparedStatement deleteSalesReceiptDetailsStmt = connection.prepareStatement(deleteSalesReceiptDetailsQuery);  
 deleteSalesReceiptDetailsStmt.setInt(1, productID);  
 deleteSalesReceiptDetailsStmt.executeUpdate();  
  
 // Delete product  
 PreparedStatement deleteProductStmt = connection.prepareStatement(deleteProductQuery);  
 deleteProductStmt.setInt(1, productID);  
 deleteProductStmt.executeUpdate();  
  
 // Commit transaction  
 connection.commit();  
 return true;  
 } catch (SQLException e) {  
 try {  
 // Rollback if there's an error  
 connection.rollback();  
 } catch (SQLException rollbackEx) {  
 rollbackEx.printStackTrace();  
 }  
  
 // Handle the specific SQLIntegrityConstraintViolationException  
 if (e instanceof SQLIntegrityConstraintViolationException) {  
 SQLIntegrityConstraintViolationException sqlEx = (SQLIntegrityConstraintViolationException) e;  
 if (sqlEx.getErrorCode() == 1451 || sqlEx.getErrorCode() == 1452) {  
 // Display a user-friendly error message for foreign key constraint violation  
 JOptionPane.*showMessageDialog*(null, "Không thể xóa sản phẩm vì có dữ liệu liên quan đến sản phẩm này.");  
 return false;  
 }  
 }  
  
 e.printStackTrace();  
 return false;  
 } finally {  
 try {  
 // Reset auto-commit mode  
 connection.setAutoCommit(true);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
  
 public boolean checkFileNameExists(String fileName) {  
 try {  
 // Prepare the SQL statement to check if the file name exists  
 String query = "SELECT COUNT(\*) AS count FROM ProductImages WHERE ImageUrl = ?";  
 try (PreparedStatement statement = connection.prepareStatement(query)) {  
 // Set the file name parameter in the prepared statement  
 statement.setString(1, fileName);  
  
 // Execute the query  
 try (ResultSet resultSet = statement.executeQuery()) {  
 // Check if any row exists with the given file name  
 if (resultSet.next()) {  
 int count = resultSet.getInt("count");  
 return count > 0;  
 }  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
  
 public int getStockQuantity(int productID) {  
 String query = "SELECT StockQuantity FROM Products WHERE ProductID = ?";  
 try (PreparedStatement stmt = connection.prepareStatement(query)) {  
 stmt.setInt(1, productID);  
 try (ResultSet rs = stmt.executeQuery()) {  
 if (rs.next()) {  
 return rs.getInt("StockQuantity");  
 } else {  
 throw new SQLException("Product not found.");  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return -1; // Trả về -1 nếu có lỗi xảy ra  
 }  
 }  
  
 public double getUnitPrice(int productID) throws SQLException {  
 String query = "SELECT UnitPrice FROM Products WHERE ProductID = ?";  
 try (PreparedStatement stmt = connection.prepareStatement(query)) {  
 stmt.setInt(1, productID);  
 try (ResultSet rs = stmt.executeQuery()) {  
 if (rs.next()) {  
 return rs.getDouble("UnitPrice");  
 } else {  
 throw new SQLException("Product not found.");  
 }  
 }  
 }  
 }  
  
 public DefaultTableModel getAllProductGiaoDichDetails() {  
 String[] columnNames = {"Mã sản phẩm", "Tên sản phẩm", "Danh mục", "Số lượng tồn kho", "Đơn giá", "Nhà cung cấp", "Tạo bởi", "Người tạo", "Cập nhật bởi", "Người cập nhật", "Thời gian cập nhật", "Hình ảnh"};  
 DefaultTableModel model = new DefaultTableModel(columnNames, 0);  
  
 String sql = "SELECT " +  
 "p.ProductID AS 'Mã sản phẩm', " +  
 "p.ProductName AS 'Tên sản phẩm', " +  
 "p.Category AS 'Danh mục', " +  
 "p.StockQuantity AS 'Số lượng tồn kho', " +  
 "p.UnitPrice AS 'Đơn giá', " +  
 "s.SupplierName AS 'Nhà cung cấp', " +  
 "p.CreatedBy AS 'Tạo bởi', " +  
 "uc.Username AS 'Người tạo', " +  
 "p.UpdatedBy AS 'Cập nhật bởi', " +  
 "uu.Username AS 'Người cập nhật', " +  
 "p.UpdatedAt AS 'Thời gian cập nhật', " +  
 "pi.ImageUrl AS 'Hình ảnh' " +  
 "FROM Products p " +  
 "LEFT JOIN Suppliers s ON p.SupplierID = s.SupplierID " +  
 "LEFT JOIN Users uc ON p.CreatedBy = uc.UserID " +  
 "LEFT JOIN Users uu ON p.UpdatedBy = uu.UserID " +  
 "LEFT JOIN ProductImages pi ON p.ProductID = pi.ProductID";  
  
 try (Connection conn = MyConnection.*getConnection*();  
 PreparedStatement preparedStatement = conn.prepareStatement(sql);  
 ResultSet resultSet = preparedStatement.executeQuery()) {  
  
 while (resultSet.next()) {  
 int productID = resultSet.getInt("Mã sản phẩm");  
 String productName = resultSet.getString("Tên sản phẩm");  
 String category = resultSet.getString("Danh mục");  
 int stockQuantity = resultSet.getInt("Số lượng tồn kho");  
 double unitPrice = resultSet.getDouble("Đơn giá");  
 String supplierName = resultSet.getString("Nhà cung cấp");  
 int createdBy = resultSet.getInt("Tạo bởi");  
 String creatorUsername = resultSet.getString("Người tạo");  
 int updatedBy = resultSet.getInt("Cập nhật bởi");  
 String updaterUsername = resultSet.getString("Người cập nhật");  
 java.util.Date updatedAt = resultSet.getTimestamp("Thời gian cập nhật");  
 String imageUrl = resultSet.getString("Hình ảnh");  
  
 model.addRow(new Object[]{productID, productName, category, stockQuantity, unitPrice, supplierName, createdBy, creatorUsername, updatedBy, updaterUsername, updatedAt, imageUrl});  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return model;  
 }  
  
 public boolean addSalesReceiptDetail(SalesReceiptDetail salesReceiptDetail) {  
 String insertSalesReceiptDetailQuery = "INSERT INTO SalesReceiptDetails (SalesReceiptID, ProductID, Quantity, UnitPrice, TotalPrice, CreatedBy, UpdatedBy) " +  
 "VALUES (?, ?, ?, ?, ?, ?, ?)";  
  
 try {  
 // Thực hiện thêm chi tiết giao dịch vào cơ sở dữ liệu  
 PreparedStatement insertSalesReceiptDetailStmt = connection.prepareStatement(insertSalesReceiptDetailQuery);  
 insertSalesReceiptDetailStmt.setInt(1, salesReceiptDetail.getSalesReceiptID());  
 insertSalesReceiptDetailStmt.setInt(2, salesReceiptDetail.getProductID());  
 insertSalesReceiptDetailStmt.setInt(3, salesReceiptDetail.getQuantity());  
 insertSalesReceiptDetailStmt.setDouble(4, salesReceiptDetail.getUnitPrice());  
 insertSalesReceiptDetailStmt.setDouble(5, salesReceiptDetail.getTotalPrice());  
 insertSalesReceiptDetailStmt.setInt(6, salesReceiptDetail.getCreatedBy());  
 insertSalesReceiptDetailStmt.setInt(7, salesReceiptDetail.getUpdatedBy());  
 insertSalesReceiptDetailStmt.executeUpdate();  
 return true;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
 public boolean updateStockQuantity(int productID, int newStockQuantity) {  
 String updateStockQuantityQuery = "UPDATE Products SET StockQuantity = ? WHERE ProductID = ?";  
  
 try {  
 // Thực hiện cập nhật số lượng hàng trong cơ sở dữ liệu  
 PreparedStatement updateStockQuantityStmt = connection.prepareStatement(updateStockQuantityQuery);  
 updateStockQuantityStmt.setInt(1, newStockQuantity);  
 updateStockQuantityStmt.setInt(2, productID);  
 updateStockQuantityStmt.executeUpdate();  
 return true;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
}

## 2. SalesReceiptController

package org.example.controllers;  
  
import org.example.connect.MyConnection;  
import org.example.models.SalesReceipt;  
  
import javax.swing.table.DefaultTableModel;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Date;  
import java.util.List;  
import java.util.Optional;  
  
public class SalesReceiptController {  
  
 private Connection connection;  
  
 public SalesReceiptController() {  
 // Initialize database connection  
 this.connection = MyConnection.*getConnection*();  
 }  
  
 public DefaultTableModel getAllPhieuBanHang() {  
 String query = "SELECT sr.SalesReceiptID, sr.EmployeeID, sr.SaleDate, sr.TotalAmount, sr.CustomerName, sr.PaymentMethod, sr.CreatedBy, sr.UpdatedBy, sr.UpdatedAt, " +  
 "e.EmployeeID AS EmployeeID, e.FirstName, e.LastName, e.Position, e.DateOfBirth, e.ContactInfo, e.HireDate, e.Salary, e.CreatedBy AS EmployeeCreatedBy, e.UpdatedBy AS EmployeeUpdatedBy, e.UpdatedAt AS EmployeeUpdatedAt " +  
 "FROM SalesReceipts sr " +  
 "INNER JOIN Employees e ON sr.EmployeeID = e.EmployeeID";  
  
 try (Connection connection = MyConnection.*getConnection*();  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 DefaultTableModel model = new DefaultTableModel(  
 new String[]{"SalesReceiptID", "EmployeeID", "SaleDate", "TotalAmount", "CustomerName", "PaymentMethod", "CreatedBy", "UpdatedBy", "UpdatedAt",  
 "EmployeeID", "FirstName", "LastName", "Position", "DateOfBirth", "ContactInfo", "HireDate", "Salary", "EmployeeCreatedBy", "EmployeeUpdatedBy", "EmployeeUpdatedAt"}, 0);  
  
 while (resultSet.next()) {  
 int salesReceiptID = resultSet.getInt("SalesReceiptID");  
 int employeeID = resultSet.getInt("EmployeeID");  
 Date saleDate = resultSet.getDate("SaleDate");  
 double totalAmount = resultSet.getDouble("TotalAmount");  
 String customerName = resultSet.getString("CustomerName");  
 String paymentMethod = resultSet.getString("PaymentMethod");  
 int createdBy = resultSet.getInt("CreatedBy");  
 int updatedBy = resultSet.getInt("UpdatedBy");  
 Date updatedAt = resultSet.getDate("UpdatedAt");  
  
 int employeeIDFromEmployee = resultSet.getInt("EmployeeID");  
 String firstName = resultSet.getString("FirstName");  
 String lastName = resultSet.getString("LastName");  
 String position = resultSet.getString("Position");  
 Date dateOfBirth = resultSet.getDate("DateOfBirth");  
 String contactInfo = resultSet.getString("ContactInfo");  
 Date hireDate = resultSet.getDate("HireDate");  
 double salary = resultSet.getDouble("Salary");  
 int employeeCreatedBy = resultSet.getInt("EmployeeCreatedBy");  
 int employeeUpdatedBy = resultSet.getInt("EmployeeUpdatedBy");  
 Date employeeUpdatedAt = resultSet.getDate("EmployeeUpdatedAt");  
  
 model.addRow(new Object[]{salesReceiptID, employeeID, saleDate, totalAmount, customerName, paymentMethod, createdBy, updatedBy, updatedAt,  
 employeeIDFromEmployee, firstName, lastName, position, dateOfBirth, contactInfo, hireDate, salary, employeeCreatedBy, employeeUpdatedBy, employeeUpdatedAt});  
 }  
  
 return model;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
}

## 3. SalesReceiptDetailController

package org.example.controllers;  
  
import org.example.connect.MyConnection;  
import org.example.models.SalesReceipt;  
  
import javax.swing.table.DefaultTableModel;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Date;  
import java.util.List;  
import java.util.Optional;  
  
public class SalesReceiptController {  
  
 private Connection connection;  
  
 public SalesReceiptController() {  
 // Initialize database connection  
 this.connection = MyConnection.*getConnection*();  
 }  
  
 public DefaultTableModel getAllPhieuBanHang() {  
 String query = "SELECT sr.SalesReceiptID, sr.EmployeeID, sr.SaleDate, sr.TotalAmount, sr.CustomerName, sr.PaymentMethod, sr.CreatedBy, sr.UpdatedBy, sr.UpdatedAt, " +  
 "e.EmployeeID AS EmployeeID, e.FirstName, e.LastName, e.Position, e.DateOfBirth, e.ContactInfo, e.HireDate, e.Salary, e.CreatedBy AS EmployeeCreatedBy, e.UpdatedBy AS EmployeeUpdatedBy, e.UpdatedAt AS EmployeeUpdatedAt " +  
 "FROM SalesReceipts sr " +  
 "INNER JOIN Employees e ON sr.EmployeeID = e.EmployeeID";  
  
 try (Connection connection = MyConnection.*getConnection*();  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 DefaultTableModel model = new DefaultTableModel(  
 new String[]{"SalesReceiptID", "EmployeeID", "SaleDate", "TotalAmount", "CustomerName", "PaymentMethod", "CreatedBy", "UpdatedBy", "UpdatedAt",  
 "EmployeeID", "FirstName", "LastName", "Position", "DateOfBirth", "ContactInfo", "HireDate", "Salary", "EmployeeCreatedBy", "EmployeeUpdatedBy", "EmployeeUpdatedAt"}, 0);  
  
 while (resultSet.next()) {  
 int salesReceiptID = resultSet.getInt("SalesReceiptID");  
 int employeeID = resultSet.getInt("EmployeeID");  
 Date saleDate = resultSet.getDate("SaleDate");  
 double totalAmount = resultSet.getDouble("TotalAmount");  
 String customerName = resultSet.getString("CustomerName");  
 String paymentMethod = resultSet.getString("PaymentMethod");  
 int createdBy = resultSet.getInt("CreatedBy");  
 int updatedBy = resultSet.getInt("UpdatedBy");  
 Date updatedAt = resultSet.getDate("UpdatedAt");  
  
 int employeeIDFromEmployee = resultSet.getInt("EmployeeID");  
 String firstName = resultSet.getString("FirstName");  
 String lastName = resultSet.getString("LastName");  
 String position = resultSet.getString("Position");  
 Date dateOfBirth = resultSet.getDate("DateOfBirth");  
 String contactInfo = resultSet.getString("ContactInfo");  
 Date hireDate = resultSet.getDate("HireDate");  
 double salary = resultSet.getDouble("Salary");  
 int employeeCreatedBy = resultSet.getInt("EmployeeCreatedBy");  
 int employeeUpdatedBy = resultSet.getInt("EmployeeUpdatedBy");  
 Date employeeUpdatedAt = resultSet.getDate("EmployeeUpdatedAt");  
  
 model.addRow(new Object[]{salesReceiptID, employeeID, saleDate, totalAmount, customerName, paymentMethod, createdBy, updatedBy, updatedAt,  
 employeeIDFromEmployee, firstName, lastName, position, dateOfBirth, contactInfo, hireDate, salary, employeeCreatedBy, employeeUpdatedBy, employeeUpdatedAt});  
 }  
  
 return model;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
}

# IV. CONNECT

## MyConnect

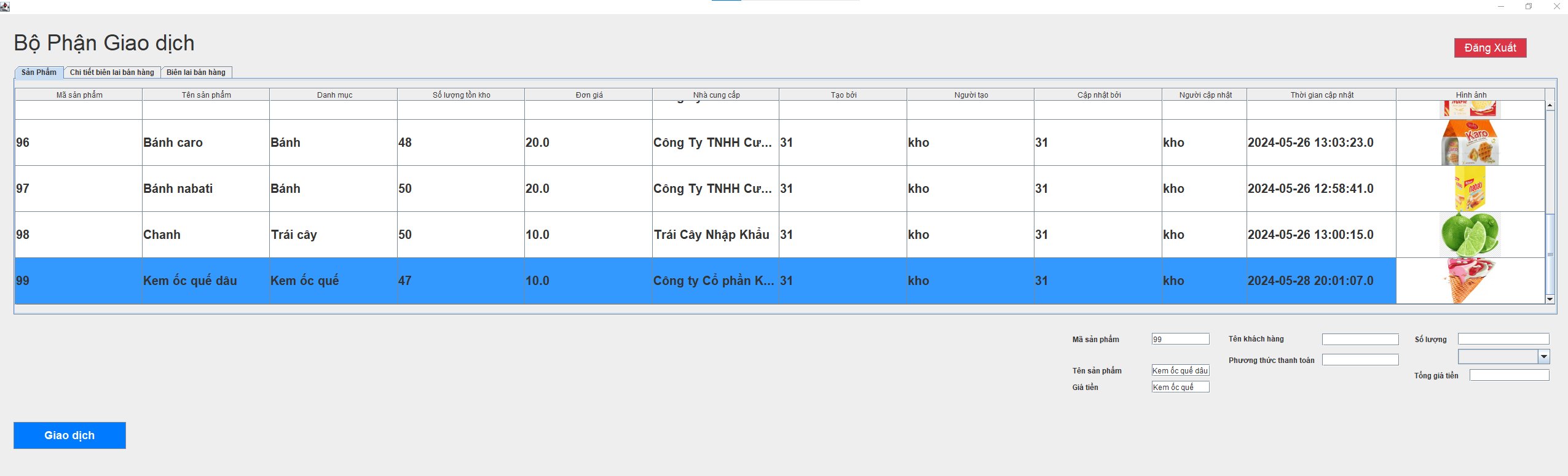
package org.example.connect;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class MyConnection {  
 private static final String *dbURL* = "jdbc:mysql://localhost:3306/mart";  
 private static final String *userName* = "root";  
 private static final String *password* = "";  
  
 public static Connection getConnection() {  
 Connection conn = null;  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 conn = DriverManager.*getConnection*(*dbURL*, *userName*, *password*);  
 } catch (ClassNotFoundException | SQLException e) {  
 e.printStackTrace();  
 }  
 return conn;  
 }  
}

# V. KẾT QUẢ

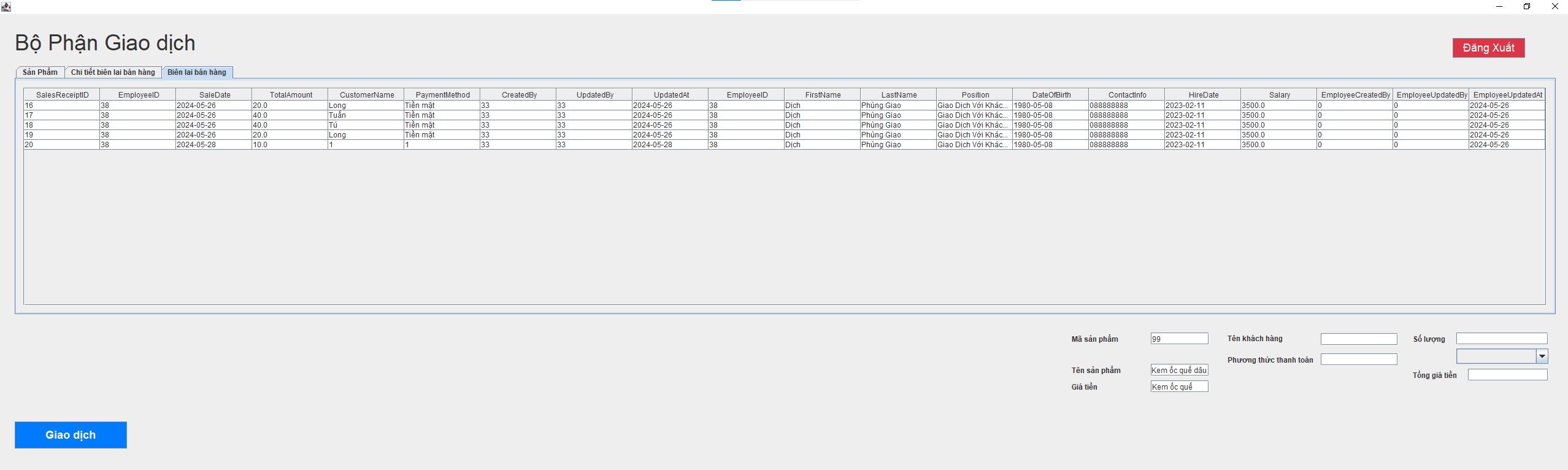
## 1. Đăng nhập



## 2. Thêm giao dịch



## 3. Biên lai giao dịch



## 4. Chi tiết biên lai giao dịch

