**TRƯỜNG CAO ĐẲNG THỰC HÀNH**

**FPT POLYTECHNIC - TP. HCM**

🞟🞟🕮🞟🞟

****

**DỰ ÁN 1**

**NGÀNH CNTT (ỨNG DỤNG PHẦN MỀM)**

**PHẦN MỀM QUẢN LÝ SIÊU THỊ MINI**

***“SuperSALES”***

***Phần 02***

SVTH: 1. Nguyễn Trần Duy – PS09687 (Trưởng nhóm)

2. Phan Văn Phú – PS08761

3. Nguyễn Quốc Khánh – PS08765

4. Nguyễn Nhật Thông – PS08608

LỚP: UD14306

GVHD: ThS. Lê Anh Tú

**TPHCM –2019**

**LỜI NÓI ĐẦU**

Với mọi ngành nghề công việc, kỹ năng cứng và kỹ năng mềm là hai yêu cầu luôn đi đôi với nhau không tách rời. Đặc biệt đối với lập trình viên, kỹ thuật phân tích thiết kế hệ thống và kỹ năng năng lập kế hoạch, làm việc nhóm là rất quan trọng. Vì thế, chúng em tiếp cận môn *“Dự án 1”* với hai mục tiêu cơ bản: 1 – ôn tập và đào sâu những kiến thức đã học về lập trình Java, 2 – vận dụng kỹ năng làm việc nhóm vào công việc thực tế.

Chúng em lựa chọn xây dựng phầm mềm quản lý siêu thị mini, bao gồm các yêu cầu nghiệp vụ dàn trải từ thu ngân đến quản lý kho hàng, quản lý khách hàng/đối tác và quản lý tài chính. Để thực hiện được những yêu cầu đó, chúng em đã dành thời gian tìm hiểu, nghiên cứu quy trình bán hàng thực tế ở các cửa hàng bán lẻ và các chuỗi bán hàng thương hiệu lớn như Bách Hoá Xanh, FamilyMart, VinMart+, … Ngoài ra chúng em cũng tham khảo các phần mềm bán hàng hiện có trên thị trường, và kết hợp đọc một số tài liệu về quản lý thu/chi, kiểm kê kho hàng và lập báo cáo tài chính để nắm các yêu cầu nghiệp vụ cơ bản.

Qua quá trình tìm hiểu và xây dựng phần mềm, chúng em đã cơ bản hoàn thành những yêu cầu trên và nắm được quy trình thiết kế một phần mềm. Bên cạnh đó, chúng em cũng học hỏi thêm những kiến thức mới khi tích hợp các thư viện Java vào phần mềm. Tuy nhiên, vì thời gian của môn học giới hạn và sự dàn trải nghiệp vụ đã nêu trên, một số chức năng chúng em thiết kế tượng trưng và để dành sự hoàn thiện đó cho tương lai triển khai phần mềm vào sử dụng thực tế.

Đề tài tuy đã hoàn thành nhưng không tránh khỏi còn những sơ sót, chúng em rất mong nhận được những lời góp ý, chỉ dẫn của quý thầy cô để đề tài hoàn thiện tốt hơn.

Nhóm thực hiện đề tài

**MỤC LỤC**

[**3. THỰC HIỆN VIẾT MÃ** 1](#_Toc27673350)

[3. 1. VIẾT MÃ CHO ỨNG DỤNG 1](#_Toc27673351)

[3.3.1. Các lớp tiện ích 1](#_Toc27673352)

[3.3.2. Viết mã cho các chức năng 5](#_Toc27673353)

[**4. KIỂM THỬ** 182](#_Toc27673354)

[4.1. BanHangJFrame 182](#_Toc27673355)

[4.2. ThemMoiKhachHangJFrame 182](#_Toc27673356)

[4.3. CapNhatKhachHangJFrame 183](#_Toc27673357)

[4.4. ThemMoiNhaCungCapJFrame 183](#_Toc27673358)

[4.5. CapNhatNhaCungCapJFrame 184](#_Toc27673359)

[4.6. ThemMoiNhanVienJFrame 184](#_Toc27673360)

[4.7. CapNhatNhanVienJFrame 185](#_Toc27673361)

[4.8. ThemMoiSanPhamJFrame 185](#_Toc27673362)

[4.9. CapNhatSanPhamJFrame 186](#_Toc27673363)

[4.10. ThemMoiPhieuKiemJFrame 186](#_Toc27673364)

[4.11. CapNhatPhieuKiemJFrame 187](#_Toc27673365)

[4.12. ThemMoiPhieuNhapJFrame 187](#_Toc27673366)

[4.13. ThemMoiPhieuHuyJFrame 187](#_Toc27673367)

[**5. ĐÓNG GÓI VÀ TRIỂN KHAI** 188](#_Toc27673368)

[5.1. Hướng dẫn chuyển đổi jar thành exe 188](#_Toc27673369)

[5.2. Hướng dẫn cài đặt triển khai 188](#_Toc27673370)

[5.3. Hướng dẫn sử dụng phần mềm 188](#_Toc27673371)

# 

# THỰC HIỆN VIẾT MÃ

## VIẾT MÃ CHO ỨNG DỤNG

### Các lớp tiện ích

* + - 1. **DialogHelper**

package helper;

import java.awt.Component;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author Administrator

\*/

public class DialogHelper {

/\*\*

\* Hiển thị thông báo cho người dùng

\*

\* @param parent là cửa sổ chứa thông báo

\* @param message là thông báo

\*/

public static void alert(Component parent, String message) {

JOptionPane.showMessageDialog(parent, message, "SuperSALES - Quản lý siêu thị mini", JOptionPane.INFORMATION\_MESSAGE);

}

/\*\*

\* Hiển thị thông báo và yêu cầu người dùng xác nhận

\*

\* @param parent là cửa sổ chứa thông báo

\* @param message là câu hỏi yes/no

\* @return là kết quả nhận được true/false

\*/

public static boolean confirm(Component parent, String message) {

int result = JOptionPane.showConfirmDialog(parent, message,

"SuperSALES - Quản lý siêu thị mini",

JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

return result == JOptionPane.YES\_OPTION;

}

/\*\*

\* Hiển thị thông báo yêu cầu nhập dữ liệu

\*

\* @param parent là cửa sổ chứa thông báo

\* @param message là thông báo nhắc nhở nhập

\* @return là kết quả nhận được từ người sử dụng nhập vào

\*/

public static String prompt(Component parent, String message) {

return JOptionPane.showInputDialog(parent, message,

"SuperSALES - Quản lý siêu thị mini", JOptionPane.INFORMATION\_MESSAGE);

}

}

* + - 1. **ShareHelper**

package helper;

import java.io.File;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

import java.util.ArrayList;

import java.util.List;

import javax.swing.ImageIcon;

import model.HoaDon;

import model.KhachHang;

import model.NguoiDung;

import model.SanPham;

/\*\*

\*

\* @author Administrator

\*/

public class ShareHelper {

/\*\*

\* Đối tượng này chứa thông tin khách hàng đang giao dịch tại quầy

\* Sử dụng đối tượng để lấy các thông tin cho frame banhang, timkhachhang, chietkhau

\* Sau khi click THANH TOÁN thì đối tượng này được set null

\*/

public static KhachHang KhachGiaoDich = null;

/\*\*

\* Đối tượng này chứa thông tin hoá đơn đang giao dịch

\* Sử dụng đối tượng để lưu các thông tin của hoá đơn và tính chiết khấu

\* Sau khi click THANH TOÁN thì đối tượng này được set null

\*/

public static HoaDon HoaDonGiaoDich = null;

/\*\*

\* Đối tượng này chứa điểm thưởng tạm tính của khách đang giao dịch

\* sử dụng để cập nhật điểm thưởng giữa frame bán hàng và chiết khấu

\*/

public static int DiemThuongMoi = 0;

/\*\*

\* Đối tượng này chứa thông tin người dùng đăng nhập

\*/

public static NguoiDung NguoiDung = null;

/\*\*

\* Đối tượng này chứa thông tin các sản phẩm đề nghị xuất huỷ khi hết HSD

\*/

public static List<SanPham> listDeNghiHuy = new ArrayList<>();

/\*\*

\* Sao chép file ảnh sản phẩm vào thư mục images/products

\* @param file là đối tượng file ảnh

\* @param filename là tên file ảnh

\* @return chép được hay không

\*/

public static boolean luuAnh(File file, String filename) {

File dir = null;

dir = new File("images/products/");

if(!dir.exists()){

dir.mkdirs();

}

File newFile = new File(dir, filename);

try {

// Copy vào thư mục (đè nếu đã tồn tại)

Path source = Paths.get(file.getAbsolutePath());

Path destination = Paths.get(newFile.getAbsolutePath());

Files.copy(source, destination, StandardCopyOption.REPLACE\_EXISTING);

return true;

} catch (Exception ex) {

return false;

}

}

/\*\*

\* Đọc hình ảnh sản phẩm

\* @param fileName là tên file

\* @return ảnh đọc được

\*/

public static ImageIcon docAnh(String fileName) {

File path = new File("images/products/", fileName);

return new ImageIcon(path.getAbsolutePath());

}}

### Viết mã cho các chức năng

* + - 1. **DangNhapJFrame**

**MÔ TẢ HOẠT ĐỘNG**

|  |  |
| --- | --- |
| **Sự kiện** | **Công việc** |
| Constructor | inIt() |
| Quản lý | dangNhap() |
| Bán hàng | dangNhap() |
| Thoát | Exit() |

**CÀI ĐẶT MÃ NGUỒN**

AnimationClass AC = new AnimationClass();

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

pnl\_DangNhap.setBackground(new Color(255,255,255,160));

setResizable(false);

setLocationRelativeTo(null);

sildeshow();

}

/\*

\* Hàm tạo slideshow ảnh nền

\*/

public void sildeshow() {

new Thread() {

int count;

@Override

public void run() {

try {

while (true) {

switch (count) {

case 0:

ImageIcon img = new ImageIcon(getClass().getResource("/icons/BackgroundSlideshow1.jpeg"));

lbl\_AnhSlide1.setIcon(img);

Thread.sleep(3000);

AC.jLabelXLeft(0, -1366, 60, 10, lbl\_AnhSlide1);

AC.jLabelXLeft(1366, 0, 60, 10, lbl\_AnhSlide2);

count = 1;

break;

case 1:

ImageIcon img1 = new ImageIcon(getClass().getResource("/icons/BackgroundSlideshow3.jpeg"));

lbl\_AnhSlide1.setIcon(img1);

Thread.sleep(3000);

AC.jLabelXRight(-1366, 0, 60, 10, lbl\_AnhSlide1);

AC.jLabelXRight(0, 1366, 60, 10, lbl\_AnhSlide2);

count = 2;

break;

case 2:

ImageIcon img2 = new ImageIcon(getClass().getResource("/icons/BackgroundSlideshow3.jpeg"));

lbl\_AnhSlide1.setIcon(img2);

Thread.sleep(3000);

AC.jLabelXLeft(0, -1366, 60, 10, lbl\_AnhSlide1);

AC.jLabelXLeft(1366, 0, 60, 10, lbl\_AnhSlide2);

count = 3;

break;

}

}

} catch (Exception e) {

DialogHelper.alert(rootPane, "Có lỗi: " + e.toString());

}

}

}.start();

}

/\*

\* Hàm xét điều kiện và thông tin đăng nhập

\*/

public void dangNhap(int chon) {

String TenDangNhap = txtTenDangNhap.getText();

String MatKhau = new String(txt\_MatKhau.getPassword());

try {

NguoiDung ND = NguoiDungDAO.timTenDangNhap(TenDangNhap);

if (ND != null) {

if (ND.getTrangThai()) {

String matKhau2 = ND.getMatKhau();

if (MatKhau.equals(matKhau2)) {

int vaiTro = ND.getVaiTro();

if (chon == 1 && vaiTro == 1) {

lbl\_ThongBao.setText("");

ShareHelper.NguoiDung = ND;

DialogHelper.alert(this, "Đăng nhập thành công!");

this.dispose();

new MainQuanLyJFrame().setVisible(true);

} else if (chon == 1 && vaiTro == 0) {

lbl\_ThongBao.setText("Bạn không được cấp quyền sử dụng chức năng này");

} else if (chon == 2) {

lbl\_ThongBao.setText("");

ShareHelper.NguoiDung = ND;

DialogHelper.alert(this, "Đăng nhập thành công!");

this.dispose();

new BanHangJFrame().setVisible(true);

}

} else {

DialogHelper.alert(this, "Sai mật khẩu");

}

}

else {

DialogHelper.alert(this, "Người dùng đang bị khoá tài khoản");

}

} else {

DialogHelper.alert(this, "Sai tên đăng nhập");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **BanHangJFrame**

**CÀI ĐẶT MÃ NGUỒN**

KhachHangDAO KhachHangDAO = new KhachHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

NhomHangDAO NhomHangDAO = new NhomHangDAO();

HoaDonDAO HoaDonDAO = new HoaDonDAO();

HoaDonChiTietDAO HDCTDAO = new HoaDonChiTietDAO();

// Hàm định dạng hiển thị số tiền

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm này thiết lập các thuộc tính của giao diện

\*/

void inIt() {

// start code setSize cố định không cho co giãn lung tung:

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

// end code setSize

setResizable(false);

setLocationRelativeTo(null);

startDongHo();

doDanhSachSanPham();

doNhomHang();

hienKhachHang();

hienNguoiDung();

taoHoaDon();

btn\_ThanhToan.setMnemonic(KeyEvent.VK\_F1);

btn\_XoaDanhSach.setMnemonic(KeyEvent.VK\_F2);

btn\_CapNhatDanhSach.setMnemonic(KeyEvent.VK\_F5);

txt\_NhapThongTin.requestFocus();

}

/\*

\* Hàm khởi tạo đồng hồ

\*/

void startDongHo(){

lbl\_NgayThangNam.setText(DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

new Timer(1000, new ActionListener() {

SimpleDateFormat format = new SimpleDateFormat("hh:mm:ss a");

@Override

public void actionPerformed(ActionEvent e) {

lbl\_DongHo.setText(format.format(new Date()));

}

}).start();

}

/\*

\* Hàm này đổ danh sách sản phẩm vào tab Danh Sách Sản Phẩm

\*/

void doDanhSachSanPham() {

try {

List<SanPham> ListSP = SanPhamDAO.locSanPhamDangBan();

doDuLieuSanPham(ListSP);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm thao tác đổ dữ liệu lên bảng Sản Phẩm

\*/

void doDuLieuSanPham(List<SanPham> list) {

String[] TenCot = {"#", "Mã sản phẩm", "Tên sản phẩm", "Đơn giá", "Hình ảnh"};

DefaultTableModel dtm = new DefaultTableModel(null, TenCot) {

// ghi đè kiểu của column để chuyển sang kiểu ImageIcon

@Override

public Class<?> getColumnClass(int column) {

if (column == 4) {

return ImageIcon.class;

}

return Object.class;

}

// ghi đè không cho edit cell khi click vào dòng

@Override

public boolean isCellEditable(int row, int column) {

return false;

}

};

dtm.setRowCount(0);

int STT = 0;

for (SanPham sp : list) {

STT++;

ImageIcon AnhSanPham = null;

if (sp.getHinhAnh() != null) {

AnhSanPham = new ImageIcon(new ImageIcon(System.getProperty("user.dir") + "/images/products/" + sp.getHinhAnh()).getImage().getScaledInstance(100, 100, java.awt.Image.SCALE\_SMOOTH));

}

Object[] row = {

STT,

sp.getMaSanPham(),

sp.getTenSanPham(),

dcf.format(sp.getGiaBan()),

AnhSanPham

};

dtm.addRow(row);

}

tbl\_DanhSachSanPham.setModel(dtm);

}

/\*

\* Hàm này đổ danh sách nhóm hàng vào combobox ở tab DanhSachSanPham

\*/

void doNhomHang() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

try {

List<NhomHang> ListNH = NhomHangDAO.select();

dcm.addElement("Tất cả");

for (NhomHang NH : ListNH) {

dcm.addElement(NH);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn ở đây: " + e.toString());

}

}

/\*

\* Hàm tìm kiếm khách hàng theo SDT hoặc họ tên

\* Nếu có kết quả thì đổ listKH đó vào frame Kết quả

\* Nếu không có kết quả thì hiện dialog thông báo không tìm thấy

\*/

void timKhachHang() {

String ChuoiTimKiem = txt\_TimKhachHang.getText();

if (ChuoiTimKiem.isEmpty()) {

txt\_TimKhachHang.requestFocus();

} else {

try {

List<KhachHang> ListKH = KhachHangDAO.timKhachHang(ChuoiTimKiem);

if (ListKH.isEmpty()) {

DialogHelper.alert(this, "Không tìm thấy khách hàng");

} else {

new KetQuaTimKHJFrame(ListKH).setVisible(true);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi truy vấn!!!: " + e.toString());

}

}

}

/\*

\* Hàm này để hiển thị thông tin khách giao dịch khi có kết quả truy vấn

\*/

void hienKhachHang() {

if (!(ShareHelper.KhachGiaoDich == null)) {

String TenKhach = ShareHelper.KhachGiaoDich.getHoTen();

txt\_TimKhachHang.setForeground(new Color(0, 0, 0));

txt\_TimKhachHang.setText(TenKhach);

}

}

/\*

\* Hàm này để hiển thị frame thêm khách hàng khi khách có yêu cầu

\* Nếu đang tồn tại KhachGiaoDich thì không được sử dụng chức năng này

\*/

void themKhachHang() {

if (ShareHelper.KhachGiaoDich == null) {

new ThemMoiKhachHangJFrame().setVisible(true);

} else {

DialogHelper.alert(this, "Đang trong phiên giao dịch, không được thêm mới Khách hàng");

}

}

/\*

\* Hàm này tạo mới Hoá Đơn giao dịch và nhận các thuộc tính truyền vào

\* Mục đích: - lưu giữ thông tin hoá đơn khi load lại frame

- đẩy dữ liệu lên database

\* Hoá đơn được clear và tạo mới instance khi bấm THANH TOÁN

\*/

void taoHoaDon() {

if (ShareHelper.HoaDonGiaoDich == null) {

ShareHelper.HoaDonGiaoDich = new HoaDon();

ShareHelper.HoaDonGiaoDich.setMaNhanVien(txt\_TenNhanVien.getText());

} else {

capnhatDonHang();

capnhatChietKhau();

}

}

/\*

\* Hàm này tính giá trị tiền thừa trả lại cho khách

\* Sau đó truyền giá trị tiền khách trả vào hoá đơn đang giao dịch

\* Nếu tiền thừa < 0 thì phát ra thông báo kiểm tra lại thông tin

\*/

void tinhTienThua() {

float TienHoaDon = Float.parseFloat(txt\_TongCong.getText());

float TienKhachTra = Float.parseFloat(txt\_TienKhachTra.getText());

float TienThua = TienKhachTra - TienHoaDon;

if (TienThua < 0) {

DialogHelper.alert(this, "Kiểm tra lại tiền khách trả.");

txt\_TienKhachTra.requestFocus();

} else {

txt\_TienThua.setText(TienThua + "");

// truyền giá trị tiền khách trả vào hoá đơn giao dịch để đưa lên database

ShareHelper.HoaDonGiaoDich.setTienKhachTra(TienKhachTra);

}

}

/\*

\* Hàm này tính điểm thưởng của hoá đơn theo quy định đổi điểm thưởng

\* Biến ThanhTienHoaDon là giá trị hoá đơn chưa tính VAT và chiết khấu

\*/

void tinhDiemThuong() {

if (!txt\_TongThanhTien.getText().isEmpty()) {

float ThanhTienHoaDon = Float.parseFloat(txt\_TongThanhTien.getText());

if (ThanhTienHoaDon >= 100000) {

int DiemThuong = (int) Math.floor(ThanhTienHoaDon / 100000);

txt\_DiemThuong.setText(DiemThuong + "");

} else {

txt\_DiemThuong.setText("0");

}

}

}

/\*

\* Hàm này tìm sản phẩm sau khi nhận text từ ô tìm kiếm

nếu là sản phẩm đang ngưng bán thì báo lỗi

\*/

void quetMaVachSanPham() {

String ChuoiTimKiem = txt\_NhapThongTin.getText();

if (!ChuoiTimKiem.isEmpty()) {

try {

SanPham SP = SanPhamDAO.timTheoMaVach(ChuoiTimKiem);

if (SP != null) {

if (SP.getTrangThai()) {

nhapSanPhamVaoDon(SP);

}

else {

DialogHelper.alert(this, "Sản phẩm này đang ngừng kinh doanh. Thu hồi ngay");

}

} else {

txt\_NhapThongTin.setText("");

txt\_NhapThongTin.requestFocus();

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

}

/\*

\* Hàm này nhập sản phẩm vào hoá đơn

\* - Thêm SP vào Hoá Đơn Giao Dịch để lưu và đẩy lên database

- Đổ SP vào bảng danh sách đơn hàng tbl\_ChiTietDonHang

\*/

void nhapSanPhamVaoDon(SanPham SP) {

ShareHelper.HoaDonGiaoDich.themSP(SP, 1);

duyetDanhSach(SP);

txt\_NhapThongTin.setText("");

txt\_NhapThongTin.requestFocus();

}

/\*

\* Hàm này duyệt danh sách sản phẩm trong table trước khi add row

\* - nếu sản phẩm đã có thì get số lượng và cộng 1; cập nhật lại thành tiền

- nếu sản phẩm chưa có thì add row mới

\*/

void duyetDanhSach(SanPham SP) {

boolean check = true; // biến kiểm tra để xét thêm mới hay cập nhật

// chạy vòng lặp duyệt từng dòng của bảng

for (int row = 0; row <= tbl\_ChiTietDonHang.getRowCount() - 1; row++) {

// xét mã sản phẩm nếu đã tồn tại thì cập nhật số lượng và thoát vòng lặp

if (SP.getMaSanPham().equals(tbl\_ChiTietDonHang.getValueAt(row, 1))) {

int SoLuongMoi = Integer.parseInt(tbl\_ChiTietDonHang.getValueAt(row, 4).toString()) + 1;

// cập nhật lại số lượng

tbl\_ChiTietDonHang.setValueAt(SoLuongMoi, row, 4);

// cập nhật lại thành tiền

tbl\_ChiTietDonHang.setValueAt(SoLuongMoi \* SP.getGiaBan(), row, 5);

check = false;

return;

}

}

if (check) {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietDonHang.getModel();

int STT = tbl\_ChiTietDonHang.getRowCount() + 1;

double ThanhTien = ShareHelper.HoaDonGiaoDich.getDSSP().get(SP) \* SP.getGiaBan();

Object[] addrow = {

STT,

SP.getMaSanPham(),

SP.getTenSanPham(),

dcf.format(SP.getGiaBan()),

ShareHelper.HoaDonGiaoDich.getDSSP().get(SP),

ThanhTien,

"Xoá"

};

taoButtonXoa();

dtm.addRow(addrow);

}

}

/\*

\* Hàm này tạo button XOÁ trong danh sách sản phẩm của đơn hàng

\*/

void taoButtonXoa() {

tbl\_ChiTietDonHang.getColumn("Thao tác").setCellRenderer(new ButtonRenderer());

tbl\_ChiTietDonHang.getColumn("Thao tác").setCellEditor(new ButtonEditor(new JCheckBox()));

tbl\_ChiTietDonHang.setPreferredScrollableViewportSize(tbl\_ChiTietDonHang.getPreferredSize());

tbl\_ChiTietDonHang.getColumnModel().getColumn(6).setPreferredWidth(100);

}

/\*

\* Hàm này xoá sản phẩm trong đơn hàng và trong danh sách TreeMap<SanPham,Integer>

\*/

void xoaSanPhamDonHang() {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietDonHang.getModel();

dtm.setRowCount(0);

ShareHelper.HoaDonGiaoDich.getDSSP().clear();

}

/\*

\* Hàm này tìm sản phẩm ở tab DANH SÁCH SẢN PHẨM

\*/

void timSanPham() {

List<SanPham> list;

// get tên nhóm hàng và mã nhóm hàng

String TuyChonNhomHang = cmb\_NhomHang.getSelectedItem().toString();

// get chuỗi tìm kiếm

String ChuoiTimKiem = txt\_TimKiemSanPham.getText();

// nếu chuỗi rỗng thì đổ tất cả sản phẩm theo mã nhóm

if (ChuoiTimKiem.isEmpty()) {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.select();

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timTheoMaNhom(MaNhom);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

} else {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.timSanPhamTheoTen(ChuoiTimKiem);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timKetHopNhomHang(ChuoiTimKiem, MaNhom);

if (list != null) {

doDuLieuSanPham(list);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

}

}

/\*

\* Hàm này đếm tổng số lượng sản phẩm trong đơn hàng

\* và tính tổng tiền để điền vào textfield

\*/

void demSoLuong() {

int demSoLuong = 0;

float tongThanhTien = 0;

for (int i = 0; i < tbl\_ChiTietDonHang.getRowCount(); i++) {

int SoLuong = Integer.parseInt(tbl\_ChiTietDonHang.getValueAt(i, 4).toString());

float ThanhTien = Float.parseFloat(tbl\_ChiTietDonHang.getValueAt(i, 5).toString());

tongThanhTien += ThanhTien;

demSoLuong += SoLuong;

}

txt\_TongSoLuong.setText(demSoLuong + "");

txt\_TongThanhTien.setText(tongThanhTien + "");

}

/\*

\* Hàm cập nhật thông tin sản phẩm trong đơn hàng

\* 1. Cập nhật Key & Value trong DSSP trước

\* 2. Duyệt TreeMap DSSP và đổ dữ liệu lên lại tbl\_ChiTietDonHang

\* 3. Cập nhật lại số lượng và thành tiền

\*/

void capnhatDonHang() {

if(!ShareHelper.HoaDonGiaoDich.getDSSP().isEmpty()){

// duyệt các dòng trong table để put lại số lượng của SP trong TreeMap

for (int i = 0; i < tbl\_ChiTietDonHang.getRowCount(); i++) {

String MaSP = tbl\_ChiTietDonHang.getValueAt(i, 1).toString();

int SoLuong = Integer.parseInt(tbl\_ChiTietDonHang.getValueAt(i, 4).toString());

SanPham putSP = SanPhamDAO.timTheoMaSP(MaSP);

if (SoLuong == 0) {

// nếu số lượng nhập lại là 0 thì xoá sản phẩm khỏi TreeMap

ShareHelper.HoaDonGiaoDich.getDSSP().remove(putSP);

} else {

ShareHelper.HoaDonGiaoDich.getDSSP().put(putSP, SoLuong);

}

}

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietDonHang.getModel();

dtm.setRowCount(0);

// duyệt TreeMap và đổ thông tin lên table

for (SanPham key : ShareHelper.HoaDonGiaoDich.getDSSP().keySet()) {

int STT = tbl\_ChiTietDonHang.getRowCount() + 1;

Object[] addrow = {

STT,

key.getMaSanPham(),

key.getTenSanPham(),

key.getGiaBan(),

ShareHelper.HoaDonGiaoDich.getDSSP().get(key),

ShareHelper.HoaDonGiaoDich.getDSSP().get(key) \* key.getGiaBan(),

"Xoá"

};

taoButtonXoa();

dtm.addRow(addrow);

}

demSoLuong();

}

}

/\*

\* Hàm xoá thông tin khách giao dịch (để cập nhật lại thông tin nếu có sai sót)

và xoá các thông tin: chiết khấu, tiền khách trả

\*/

void xoaThongTinKhachGiaoDich() {

ShareHelper.KhachGiaoDich = null;

txt\_TimKhachHang.setText("Tìm khách hàng");

txt\_TimKhachHang.setForeground(new Color(204, 204, 204));

ShareHelper.HoaDonGiaoDich.setChietKhau(0);

txt\_ChietKhau.setText("0");

txt\_TienKhachTra.setText("0");

}

/\*

\* Hàm chọn và thêm sản phẩm thủ công từ tab DanhSachSanPham

\* sử dụng khi sản phẩm không có mã vạch

\*/

void themSanPhamThuCong() {

int index = tbl\_DanhSachSanPham.getSelectedRow();

String MaSP = tbl\_DanhSachSanPham.getValueAt(index, 1).toString();

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

nhapSanPhamVaoDon(SP);

demSoLuong();

tab\_ThongTinSanPham.setSelectedIndex(0);

}

/\*

\* Hàm xử lý khi click button chiết khấu

\* - nếu khách lẻ thì không sử dụng được chức năng này

\* - nếu chưa chọn khách hàng thì không sử dụng được chức năng này

\*/

void chonChietKhau() {

if (ShareHelper.KhachGiaoDich == null) {

DialogHelper.alert(this, "Vui lòng điền thông tin khách đang giao dịch");

} else {

if (ShareHelper.KhachGiaoDich.getMaKhachHang().equals("KH00000")) {

DialogHelper.alert(this, "Khách lẻ không được áp dụng chiết khấu");

} else {

new ChietKhauJFrame(txt\_TongThanhTien.getText()).setVisible(true);

}

}

}

/\*

\* Hàm cập nhật chiết khấu của hoá đơn tạm đang giao dịch

\*/

void capnhatChietKhau(){

if(ShareHelper.HoaDonGiaoDich.getChietKhau() > 0) {

txt\_ChietKhau.setText(ShareHelper.HoaDonGiaoDich.getChietKhau()+"");

}

else {

txt\_ChietKhau.setText("0");

}

}

/\*

\* Hàm tính tổng tiền thanh toán của hoá đơn

đã bao gồm chiết khấu và VAT

\*/

void tinhTongTien(){

float ThanhTien = 0;

float ChietKhau = 0;

if(!txt\_TongThanhTien.getText().isEmpty()){

ThanhTien = Float.parseFloat(txt\_TongThanhTien.getText());

}

if(!txt\_ChietKhau.getText().isEmpty()){

ChietKhau = Float.parseFloat(txt\_ChietKhau.getText());

}

float VAT = 0;

if(cmb\_VAT.getSelectedIndex() != 0){

VAT = 0.1f;

}

float TongTien = (ThanhTien - ChietKhau) + (ThanhTien - ChietKhau)\*VAT;

txt\_TongCong.setText(TongTien+"");

}

/\*

\* Hàm đẩy dữ liệu lên database

và có thể kết nối máy in để in hoá đơn

\*/

void xuatHoaDon(){

// kiem tra cac thong tin can thiet

if(xetDieuKien()){

ShareHelper.HoaDonGiaoDich.setMaNhanVien(ShareHelper.NguoiDung.getTenDangNhap());

// đưa thông tin hoá đơn lên database: hoá đơn -> hoá đơn chi tiết

try {

ShareHelper.HoaDonGiaoDich.setTongSoLuong(Integer.parseInt(txt\_TongSoLuong.getText()));

ShareHelper.HoaDonGiaoDich.setChietKhau(Float.parseFloat(txt\_ChietKhau.getText()));

if(cmb\_VAT.getSelectedIndex() == 1){

ShareHelper.HoaDonGiaoDich.setVAT(10);

}

else {

ShareHelper.HoaDonGiaoDich.setVAT(0);

}

ShareHelper.HoaDonGiaoDich.setThanhTien(Float.parseFloat(txt\_TongCong.getText()));

ShareHelper.HoaDonGiaoDich.setTienKhachTra(Float.parseFloat(txt\_TienKhachTra.getText()));

int rs = HoaDonDAO.themHoaDon(ShareHelper.HoaDonGiaoDich);

// nếu hoá đơn bị lỗi không thêm được thì không thêm HĐCT

// và không trừ tồn kho

if(rs > 0){

themHDCT();

capnhatTonKho();

}

// cập nhật điểm thưởng khách hàng

capnhatDiemThuong();

// sau khi hoàn tất thao tác với database thì reload lại các thông tin

DialogHelper.alert(this, "Đã hoàn tất thanh toán");

// in hoá đơn cho khách

inHoaDon();

ShareHelper.HoaDonGiaoDich.getDSSP().clear();

ShareHelper.HoaDonGiaoDich = null;

ShareHelper.KhachGiaoDich = null;

this.dispose();

new BanHangJFrame().setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm này xét các điều kiện đảm bảo đúng trước khi thanh toán

\*/

boolean xetDieuKien(){

boolean ketqua = true;

// kiem tra co dien thong tin khach giao dich hay khong

if(ShareHelper.KhachGiaoDich == null){

ketqua = false;

}

// kiem tra nhan vien login chua

if(ShareHelper.NguoiDung == null){

ketqua = false;

}

// kiem tra da cap nhat lai don hang hay chua (De phong co thay doi)

if(btn\_CapNhatDanhSach.getBackground() == Color.red){

ketqua = false;

}

// kiem tra trong list hoa don tam co san pham hay khong

if(ShareHelper.HoaDonGiaoDich.getDSSP().isEmpty()){

ketqua = false;

}

// kiem tra tien khach tra

float TienKhachTra = Float.parseFloat(txt\_TienKhachTra.getText());

float TongCong = Float.parseFloat(txt\_TongCong.getText());

if(TienKhachTra < TongCong){

ketqua = false;

}

return ketqua;

}

/\*

\* Hàm đưa hoá đơn chi tiết lên database

\*/

static String MaHoaDon;

void themHDCT() throws SQLException{

// get mã hoá đơn vừa đưa lên database (tức mã mới nhất)

int dem = HoaDonDAO.demHoaDon();

MaHoaDon = "HD" + dem;

for(SanPham SP : ShareHelper.HoaDonGiaoDich.getDSSP().keySet()){

HoaDonChiTiet HDCT = new HoaDonChiTiet();

HDCT.setMaHoaDon(MaHoaDon);

HDCT.setMaSanPham(SP.getMaSanPham());

HDCT.setDonGia(SP.getGiaBan());

HDCT.setSoLuong(ShareHelper.HoaDonGiaoDich.getDSSP().get(SP));

HDCT.setThanhTien(SP.getGiaBan() \* ShareHelper.HoaDonGiaoDich.getDSSP().get(SP));

HDCTDAO.themHDCT(HDCT);

}

}

/\*

\* Hàm lấy thông tin người dùng đăng nhập để hiển thị

\*/

void hienNguoiDung(){

if(ShareHelper.NguoiDung != null){

txt\_TenNhanVien.setText(ShareHelper.NguoiDung.getHoTen());

}

}

/\*

\* Hàm cập nhật điểm thưởng khách hàng lên database nếu có thay đổi

\*/

void capnhatDiemThuong() throws SQLException{

int DiemThuong = 0;

if(ShareHelper.DiemThuongMoi > 0){

DiemThuong = ShareHelper.DiemThuongMoi + Integer.parseInt(txt\_DiemThuong.getText());

}

else {

DiemThuong = ShareHelper.KhachGiaoDich.getDiemThuong() + Integer.parseInt(txt\_DiemThuong.getText());

}

ShareHelper.KhachGiaoDich.setDiemThuong(DiemThuong);

KhachHangDAO.capnhatDiemThuong(ShareHelper.KhachGiaoDich);

}

/\*

\* Hàm exit khỏi chương trình

\*/

void exit(){

if(DialogHelper.confirm(this, "Bạn muốn thoát chương trình?")){

System.exit(0);

}

}

/\*

\* Hàm logout ra màn hình đăng nhập

- xoá hết thông tin: người dùng, khách hàng giao dịch, hoá đơn tạm

\*/

void logout(){

if(DialogHelper.confirm(this, "Bạn muốn đăng xuất?")){

ShareHelper.HoaDonGiaoDich = null;

ShareHelper.KhachGiaoDich = null;

ShareHelper.NguoiDung = null;

this.dispose();

new DangNhapJFrame().setVisible(true);

}

}

/\*

\* Hàm kết nối máy in và in hoá đơn

\*/

void inHoaDon() {

try {

HashMap HoaDon = new HashMap();

HoaDon.put("MaInHoaDon", MaHoaDon);

jdbcHelper.inHoaDon(HoaDon);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm cập nhật tồn kho khi xuất hoá đơn

\*/

void capnhatTonKho() {

try {

for (SanPham SP : ShareHelper.HoaDonGiaoDich.getDSSP().keySet()) {

int SoLuongMoi = SP.getTonKho() - ShareHelper.HoaDonGiaoDich.getDSSP().get(SP);

SanPhamDAO.capnhatTonKho(SP.getMaSanPham(), SoLuongMoi);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **MainQuanLyJFrame**

ThongKeDAO TKDAO = new ThongKeDAO();

HoaDonDAO HDDAO = new HoaDonDAO();

SanPhamDAO SPDAO = new SanPhamDAO();

DecimalFormat dcf = new DecimalFormat("###,###,###");

void inIt() {

// start code setSize cố định không cho co giãn lung tung:

this.setSize(1366,768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

// end code setSize

setResizable(false);

setLocationRelativeTo(null);

this.getContentPane().setBackground(Color.white);

startDongHo();

doThongTin();

hienBieuDo();

}

/\*

\* Hàm khởi tạo đồng hồ

\*/

void startDongHo(){

lbl\_NgayThangNam.setText(DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

new Timer(1000, new ActionListener() {

SimpleDateFormat format = new SimpleDateFormat("hh:mm:ss a");

@Override

public void actionPerformed(ActionEvent e) {

lbl\_DongHo.setText(format.format(new Date()));

}

}).start();

}

void doThongTin() {

try {

double TongDoanhThu = TKDAO.tinhTongDoanhThu();

int TongDonHang = HDDAO.demHoaDon();

int SPHetHan = TKDAO.demSanPhamHetHSD();

lbl\_NumTongDoanhThu.setText(dcf.format(TongDoanhThu));

lbl\_NumTongDonHang.setText(TongDonHang+"");

lbl\_NumCanhBao.setText(SPHetHan+"");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm exit khỏi chương trình

\*/

void exit(){

if(DialogHelper.confirm(this, "Bạn muốn thoát chương trình?")){

System.exit(0);

}

}

/\*

\* Hàm logout ra màn hình đăng nhập

- xoá hết thông tin: người dùng, khách hàng giao dịch, hoá đơn tạm

\*/

void logout(){

if(DialogHelper.confirm(this, "Bạn muốn đăng xuất?")){

ShareHelper.HoaDonGiaoDich = null;

ShareHelper.KhachGiaoDich = null;

ShareHelper.NguoiDung = null;

this.dispose();

new DangNhapJFrame().setVisible(true);

}

}

/\*

\* Hàm hiển thị biểu đồ thống kê

\*/

void hienBieuDo() {

DefaultCategoryDataset barChartdata = new DefaultCategoryDataset();

barChartdata.setValue(4000000, "Doanh thu", "Tháng 10");

barChartdata.setValue(5000000, "Doanh thu", "Tháng 11");

barChartdata.setValue(7000000, "Doanh thu", "Tháng 12");

JFreeChart barChart = ChartFactory.createBarChart("", "", "Doanh thu", barChartdata, PlotOrientation.VERTICAL, false, false, false);

CategoryPlot plot = barChart.getCategoryPlot();

NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();

rangeAxis.setTickUnit(new NumberTickUnit(1000000));

ChartPanel barPanel = new ChartPanel(barChart);

pnl\_BarChart.removeAll();

pnl\_BarChart.add(barPanel, BorderLayout.CENTER);

}

}

* + - 1. **ThongTinTaiKhoanJFrame**

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

/\*

\* Hàm hiện thông tin người dùng lên các textfield

\*/

void hienThongTin(){

txt\_TenDangNhap.setText(ShareHelper.NguoiDung.getTenDangNhap());

txt\_HoTen.setText(ShareHelper.NguoiDung.getHoTen());

txt\_DienThoai.setText(ShareHelper.NguoiDung.getDienThoai());

if(ShareHelper.NguoiDung.getGioiTinh()){

rad\_Nam.setSelected(true);

}

else {

rad\_Nu.setSelected(true);

}

dat\_NgaySinh.setDate(ShareHelper.NguoiDung.getNgaySinh());

txt\_Email.setText(ShareHelper.NguoiDung.getEmail());

txa\_DiaChi.setText(ShareHelper.NguoiDung.getDiaChi());

}

/\*

\* Hàm này để enable/disable textfield mật khẩu mới

khi check/uncheck vào checkbox cập nhật mật khẩu

\*/

void checkCapNhat(){

if(chk\_CapNhatMatKhau.isSelected()){

txt\_MatKhauMoi.setEnabled(true);

}

else {

txt\_MatKhauMoi.setEnabled(false);

}

}

/\*

\* Hàm cập nhật thông tin người dùng lên database

sau khi cập nhật thì close frame này, quay về frame bán hàng

\*/

void capnhatNguoiDung(){

try {

// kiem loi neu true thi moi cap nhat vao database

if(kiemLoi()){

int rs = 0;

// cap nhat model:

capnhatModel();

// kiem tra co cap nhat mat khau moi hay khong

// neu co thi cap nhat vao model

if(chk\_CapNhatMatKhau.isSelected()){

String MatKhauMoi = new String(txt\_MatKhauMoi.getPassword());

if(checkMatKhauMoi()){

ShareHelper.NguoiDung.setMatKhau(MatKhauMoi);

rs = NguoiDungDAO.capnhatNguoiDung(ShareHelper.NguoiDung, true);

}

}

else {

rs = NguoiDungDAO.capnhatNguoiDung(ShareHelper.NguoiDung, false);

}

if(rs != 0){

DialogHelper.alert(this, "Cập nhật thành công");

this.dispose();

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Cập nhật thất bại: " + e.toString());

}

}

/\*

\* Hàm kiểm lỗi trước khi cập nhật người dùng

\*/

boolean kiemLoi(){

boolean ketqua = true;

if(!checkEmail()){

ketqua = false;

}

if(!checkDienThoai()){

ketqua = false;

}

if(!checkNgaySinh()){

ketqua = false;

}

if(!checkDiaChi()){

ketqua = false;

}

if(!checkMatKhau()){

ketqua = false;

}

return ketqua;

}

/\*

\* Hàm check email

\*/

boolean checkEmail(){

String email = txt\_Email.getText();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if(txt\_Email.getText().length() <= 0 | (!email.matches(pattern))){

return false;

}

return true;

}

/\*

\* Hàm check ngày sinh: tính đến thời gian hiện tại phải từ 18t trở lên

\*/

boolean checkNgaySinh(){

if(dat\_NgaySinh.getDate() != null){

Date NgayHienTai = DateHelper.now();

Date NgaySinh = dat\_NgaySinh.getDate();

long kiemtra = DateHelper.getDateDiff(NgaySinh, NgayHienTai, TimeUnit.DAYS);

if(kiemtra/365 < 18){

DialogHelper.alert(this, "Tuổi nhân viên phải từ 18 trở lên");

return false;

}

}

return true;

}

/\*

\* Hàm check thông tin số điện thoại

chỉ chứa số, 10 hoặc 11 ký tự

\*/

boolean checkDienThoai(){

if(txt\_DienThoai.getText().length() < 10 || txt\_DienThoai.getText().length() > 11){

return false;

}

else

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi(){

if(txa\_DiaChi.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm check mật khẩu (tối thiểu 3 ký tự), không có khoảng trắng

\*/

boolean checkMatKhau(){

String MatKhauHienTai = ShareHelper.NguoiDung.getMatKhau();

if (txt\_MatKhauHienTai.getPassword().length >= 3) {

if (MatKhauHienTai.equals(new String(txt\_MatKhauHienTai.getPassword()))) {

return true;

} else {

return false;

}

}

else

return false;

}

/\*

\* Hàm check mật khẩu mới (tối thiểu 3 ký tự), không khoảng trắng

\*/

boolean checkMatKhauMoi(){

if (new String(txt\_MatKhauMoi.getPassword()).length() >= 3) {

return true;

}

else

return false;

}

/\*

\* Hàm cập nhật model

\*/

void capnhatModel(){

ShareHelper.NguoiDung.setDienThoai(txt\_DienThoai.getText());

ShareHelper.NguoiDung.setGioiTinh(rad\_Nam.isSelected());

ShareHelper.NguoiDung.setNgaySinh(dat\_NgaySinh.getDate());

ShareHelper.NguoiDung.setEmail(txt\_Email.getText());

ShareHelper.NguoiDung.setDiaChi(txa\_DiaChi.getText());

}

}

* + - 1. **DanhSachNhanVienJFrame**

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

ThongKeDAO ThongKeDAO = new ThongKeDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

}

/\*

\* Hàm cập nhật dữ liệu NguoiDung

\*/

void capnhatDuLieu() {

try {

List<NguoiDung> listND = NguoiDungDAO.select();

doDuLieuVaoBang(listND);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<NguoiDung> listND) throws SQLException {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachNhanVien.getModel();

dtm.setRowCount(0);

for (NguoiDung ND : listND) {

double DoanhThu = ThongKeDAO.tinhDoanhThuNV(ND.getTenDangNhap());

Object[] row = {ND.getTenDangNhap(),

ND.getHoTen(),

ND.getVaiTro() == 1 ? "Quản lý" : (ND.getVaiTro() == 0 ? "Nhân viên" : "Thủ kho"),

ND.getDienThoai(),

dcf.format(DoanhThu),

ND.getTrangThai() ? "Đang hoạt động" : "Ngưng hoạt động"

};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm tổng hợp theo mã, tên, SĐT

\*/

void timTongHop() {

String ChuoiTimKiem = txt\_TimTheoMaTenSDT.getText();

if (!ChuoiTimKiem.isEmpty()) {

try {

List<NguoiDung> listND = NguoiDungDAO.timTheoMaTenSDT(ChuoiTimKiem);

doDuLieuVaoBang(listND);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

} else {

capnhatDuLieu();

}

}

/\*

\* Hàm lọc theo trạng thái

\*/

void locTheoTrangThai(int i) {

try {

List<NguoiDung> listND;

if (i == 1) {

listND = NguoiDungDAO.locTheoTrangThai(1);

} else {

listND = NguoiDungDAO.locTheoTrangThai(0);

}

doDuLieuVaoBang(listND);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm gọi frame cập nhật nhân viên

\*/

void capnhatNhanVien() {

int index = tbl\_DanhSachNhanVien.getSelectedRow();

String TenDangNhap = tbl\_DanhSachNhanVien.getValueAt(index, 0).toString();

try {

NguoiDung ND = NguoiDungDAO.timTenDangNhap(TenDangNhap);

new CapNhatNhanVienJFrame(ND).setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **ThemMoiNhanVienJFrame**

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho Frame

\*/

void inIt() {

this.setSize(530, 600);

this.setMinimumSize(new Dimension(530, 600));

this.setMaximumSize(new Dimension(530, 600));

setResizable(false);

setLocationRelativeTo(null);

}

/\*

\* Hàm xoá thông tin trên textfield để nhập lại

\*/

void xoaThongTin() {

txt\_TenDangNhap.setText("");

txt\_TenDangNhap.setBackground(Color.white);

txt\_HoTen.setText("");

txt\_HoTen.setBackground(Color.white);

txt\_DienThoai.setText("");

txt\_DienThoai.setBackground(Color.white);

((JTextField) dat\_NgaySinh.getDateEditor().getUiComponent()).setText("");

((JTextField) dat\_NgaySinh.getDateEditor().getUiComponent()).setBackground(Color.white);

rad\_Nam.setSelected(true);

cmb\_VaiTro.setSelectedIndex(0);

txt\_Email.setText("");

txt\_Email.setBackground(Color.white);

txa\_DiaChi.setText("");

txa\_DiaChi.setBackground(Color.white);

cmb\_TrangThai.setSelectedIndex(0);

txt\_MatKhau.setText("");

txt\_MatKhau.setBackground(Color.white);

txt\_NhapLaiMatKhau.setText("");

txt\_NhapLaiMatKhau.setBackground(Color.white);

}

/\*

\* Hàm kiểm tra tên đăng nhập có tồn tại chưa

\*/

boolean checkTenDangNhap() {

boolean ketqua = true;

try {

String TenDangNhap = txt\_TenDangNhap.getText();

if (TenDangNhap.isEmpty() || TenDangNhap.length() < 3 || TenDangNhap.length() > 20) {

ketqua = false;

txt\_TenDangNhap.setBackground(Color.yellow);

} else if (!NguoiDungDAO.kiemTenDangNhap(TenDangNhap)) {

ketqua = false;

DialogHelper.alert(this, "Tên đăng nhập đã tồn tại");

txt\_TenDangNhap.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra sdt có tồn tại trong database hay không

\*/

boolean checkDienThoai() {

boolean ketqua = true;

try {

String SDT = txt\_DienThoai.getText();

if (SDT.isEmpty() || SDT.length() < 10 || SDT.length() > 11) {

ketqua = false;

txt\_DienThoai.setBackground(Color.yellow);

} else if (!NguoiDungDAO.kiemSDT(SDT)) {

ketqua = false;

DialogHelper.alert(this, "Số điện thoại đã tồn tại");

txt\_DienThoai.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra định dạng email

\*/

boolean checkEmail() {

try {

String email = txt\_Email.getText();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if (email.length() <= 0 | (!email.matches(pattern))) {

return false;

}

if (!NguoiDungDAO.kiemEmail(email)) {

DialogHelper.alert(this, "Email đã tồn tại");

return false;

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return true;

}

/\*

\* Hàm check ngày sinh người dùng: tính đến thời gian hiện tại phải từ 18t trở lên

\*/

boolean checkNgaySinh() {

if (dat\_NgaySinh.getDate() != null) {

Date NgayHienTai = DateHelper.now();

Date NgaySinh = dat\_NgaySinh.getDate();

long kiemtra = DateHelper.getDateDiff(NgaySinh, NgayHienTai, TimeUnit.DAYS);

if (kiemtra / 365 < 18) {

DialogHelper.alert(this, "Nhân viên phải đủ 18 tuổi");

return false;

}

}

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi() {

if (txa\_DiaChi.getText().isEmpty()) {

return false;

} else {

return true;

}

}

/\*

\* Hàm kiểm tra họ tên

\*/

boolean checkHoTen() {

String pattern = "^[A-Za-zÀÁÂÃÈÉÊÌÍÒÓÔÕÙÚÝàáâãèéêìíòóôõùúýĂăĐđĨĩŨũƠơƯưẠ-ỹ\\s]+$";

if (!txt\_HoTen.getText().matches(pattern)) {

return false;

}

return true;

}

/\*

\* Hàm kiểm tra mật khẩu

- không có khoảng trắng

- độ dài: 3 <= MK <= 20

- phải trùng với txt\_NhapLaiMatKhau

\*/

boolean checkMatKhau() {

boolean ketqua = true;

String MatKhau = new String(txt\_MatKhau.getPassword());

String NhapLaiMatKhau = new String(txt\_NhapLaiMatKhau.getPassword());

if (MatKhau.length() < 3 || MatKhau.length() > 20) {

ketqua = false;

}

if (!NhapLaiMatKhau.equals(MatKhau)) {

ketqua = false;

}

return ketqua;

}

/\*

\* Hàm kiểm lỗi trước khi thêm Nhanvien

\*/

boolean kiemLoi() {

if (!checkTenDangNhap() || !checkDiaChi() || !checkEmail() || !checkNgaySinh() || !checkDienThoai() || !checkHoTen()) {

return false;

}

return true;

}

/\*

\* Hàm tạo model NguoiDung để đưa lên database

\*/

NguoiDung taoModel() {

NguoiDung model = new NguoiDung();

model.setTenDangNhap(txt\_TenDangNhap.getText());

model.setMatKhau(new String(txt\_MatKhau.getPassword()));

model.setHoTen(txt\_HoTen.getText());

model.setDienThoai(txt\_DienThoai.getText());

model.setGioiTinh(rad\_Nam.isSelected());

model.setNgaySinh(dat\_NgaySinh.getDate());

model.setEmail(txt\_Email.getText());

model.setDiaChi(txa\_DiaChi.getText());

model.setVaiTro(cmb\_VaiTro.getSelectedIndex());

model.setTrangThai(true);

return model;

}

/\*

\* Hàm thêm Nguoi Dung vào database

sau khi thêm thì close frame này

\*/

void themNguoiDung() {

try {

if (kiemLoi()) {

NguoiDung model = taoModel();

int rs = NguoiDungDAO.themNguoiDung(model);

if (rs != 0) {

DialogHelper.alert(this, "Thêm mới thành công");

this.dispose();

if (capnhatFrameDSNV()) {

new DanhSachNhanVienJFrame().setVisible(true);

}

} else {

DialogHelper.alert(this, "Thêm mới thất bại");

}

} else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm reload lại frame DANH SÁCH NHAN VIEN để cập nhật thong tin

hàm chỉ thực thi khi frame DSNV đang mở

\*/

boolean capnhatFrameDSNV() {

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.nhanvien.DanhSachNhanVienJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **CapNhatNhanVienJFrame**

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho Frame

\*/

void inIt() {

this.setSize(530, 600);

this.setMinimumSize(new Dimension(530, 600));

this.setMaximumSize(new Dimension(530, 600));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm đổ thông tin người dùng lên textfield

\*/

void doThongTin() {

txt\_TenDangNhap.setText(model.getTenDangNhap());

txt\_HoTen.setText(model.getHoTen());

txt\_DienThoai.setText(model.getDienThoai());

if (model.getGioiTinh()) {

rad\_Nam.setSelected(true);

} else {

rad\_Nu.setSelected(true);

}

dat\_NgaySinh.setDate(model.getNgaySinh());

cmb\_VaiTro.setSelectedIndex(model.getVaiTro());

txt\_Email.setText(model.getEmail());

txa\_DiaChi.setText(model.getDiaChi());

if (model.getTrangThai()) {

cmb\_TrangThai.setSelectedIndex(0);

} else {

cmb\_TrangThai.setSelectedIndex(1);

}

}

/\*

\* Hàm xoá thông tin trên textfield để nhập lại

\*/

void xoaThongTin() {

txt\_HoTen.setBackground(Color.white);

txt\_DienThoai.setBackground(Color.white);

((JTextField) dat\_NgaySinh.getDateEditor().getUiComponent()).setBackground(Color.white);

txt\_Email.setBackground(Color.white);

txa\_DiaChi.setBackground(Color.white);

txt\_MatKhau.setBackground(Color.white);

txt\_NhapLaiMatKhau.setBackground(Color.white);

chk\_CapNhatMatKhau.setSelected(false);

txt\_MatKhau.setText("");

txt\_NhapLaiMatKhau.setText("");

doThongTin();

}

/\*

\* Hàm kiểm tra sdt có tồn tại trong database hay không

\*/

boolean checkDienThoai() {

boolean ketqua = true;

try {

String SDT = txt\_DienThoai.getText();

if (SDT.isEmpty() || SDT.length() < 10 || SDT.length() > 11) {

ketqua = false;

txt\_DienThoai.setBackground(Color.yellow);

} else if (!NguoiDungDAO.kiemSDT(SDT) && !SDT.equals(model.getDienThoai())) {

ketqua = false;

DialogHelper.alert(this, "Số điện thoại đã tồn tại");

txt\_DienThoai.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra định dạng email

\*/

boolean checkEmail() {

try {

String email = txt\_Email.getText();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if (email.length() <= 0 | (!email.matches(pattern))) {

return false;

}

if (!NguoiDungDAO.kiemEmail(email) && !email.equals(model.getEmail())) {

DialogHelper.alert(this, "Email đã tồn tại");

return false;

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return true;

}

/\*

\* Hàm check ngày sinh người dùng: tính đến thời gian hiện tại phải từ 18t trở lên

\*/

boolean checkNgaySinh() {

if (dat\_NgaySinh.getDate() != null) {

Date NgayHienTai = DateHelper.now();

Date NgaySinh = dat\_NgaySinh.getDate();

long kiemtra = DateHelper.getDateDiff(NgaySinh, NgayHienTai, TimeUnit.DAYS);

if (kiemtra / 365 < 18) {

DialogHelper.alert(this, "Nhân viên phải đủ 18 tuổi");

return false;

}

}

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi() {

if (txa\_DiaChi.getText().isEmpty()) {

return false;

} else {

return true;

}

}

/\*

\* Hàm kiểm tra họ tên

\*/

boolean checkHoTen() {

String pattern = "^[A-Za-zÀÁÂÃÈÉÊÌÍÒÓÔÕÙÚÝàáâãèéêìíòóôõùúýĂăĐđĨĩŨũƠơƯưẠ-ỹ\\s]+$";

if (!txt\_HoTen.getText().matches(pattern)) {

return false;

}

return true;

}

/\*

\* Hàm kiểm tra mật khẩu

- không có khoảng trắng

- độ dài: 3 <= MK <= 20

- phải trùng với txt\_NhapLaiMatKhau

\*/

boolean checkMatKhau() {

boolean ketqua = true;

String MatKhau = new String(txt\_MatKhau.getPassword());

String NhapLaiMatKhau = new String(txt\_NhapLaiMatKhau.getPassword());

if (MatKhau.length() < 3 || MatKhau.length() > 20) {

ketqua = false;

}

if (!NhapLaiMatKhau.equals(MatKhau)) {

ketqua = false;

}

return ketqua;

}

/\*

\* Hàm kiểm lỗi trước khi cập nhật Nhanvien

\*/

boolean kiemLoi() {

if (!checkDiaChi() || !checkEmail() || !checkNgaySinh() || !checkDienThoai() || !checkHoTen()) {

return false;

}

return true;

}

/\*

\* Hàm tạo model NguoiDung để đưa lên database

\*/

NguoiDung taoModel() {

NguoiDung ND = new NguoiDung();

ND.setTenDangNhap(txt\_TenDangNhap.getText().trim());

ND.setHoTen(txt\_HoTen.getText());

ND.setDienThoai(txt\_DienThoai.getText());

ND.setGioiTinh(rad\_Nam.isSelected());

ND.setNgaySinh(dat\_NgaySinh.getDate());

ND.setEmail(txt\_Email.getText().trim());

ND.setDiaChi(txa\_DiaChi.getText());

ND.setVaiTro(cmb\_VaiTro.getSelectedIndex());

ND.setTrangThai(cmb\_TrangThai.getSelectedIndex() == 0);

return ND;

}

/\*

\* Hàm cập nhật thông tin người dùng lên database

sau khi cập nhật thì close frame này

\*/

void capnhatNguoiDung() {

try {

// kiem loi neu true thi moi cap nhat vao database

if (kiemLoi()) {

int rs = 0;

// tạo model:

NguoiDung ND = taoModel();

// kiem tra co cap nhat mat khau moi hay khong

// neu co thi cap nhat vao model

if (chk\_CapNhatMatKhau.isSelected()) {

String MatKhauMoi = new String(txt\_MatKhau.getPassword());

if (checkMatKhau()) {

ND.setMatKhau(MatKhauMoi);

rs = NguoiDungDAO.capnhatNguoiDung(ND, true);

}

} else {

rs = NguoiDungDAO.capnhatNguoiDung(ND, false);

}

if (rs != 0) {

DialogHelper.alert(this, "Cập nhật thành công");

this.dispose();

if (capnhatFrameDSNV()) {

new DanhSachNhanVienJFrame().setVisible(true);

}

} else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

} else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Cập nhật thất bại: " + e.toString());

}

}

/\*

\* Hàm reload lại frame DANH SÁCH NHAN VIEN để cập nhật thong tin

hàm chỉ thực thi khi frame DSNV đang mở

\*/

boolean capnhatFrameDSNV() {

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.nhanvien.DanhSachNhanVienJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **DanhSachKhachHangJFrame**

KhachHangDAO KhachHangDAO = new KhachHangDAO();

ThongKeDAO ThongKeDAO = new ThongKeDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu của Frame

\*/

void inIt(){

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

btn\_NgungGiaoDich.setVisible(false);

}

/\*

\* Hàm cập nhật dữ liệu khách hàng

\*/

void capnhatDuLieu(){

try {

List<KhachHang> listKH = KhachHangDAO.select();

doDuLieuVaoBang(listKH);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm khách hàng có sinh nhật trong tháng

lọc khi select checkbox 'Sinh nhật trong tháng'

\*/

void timTheoSinhNhat(){

try {

List<KhachHang> listKH = KhachHangDAO.timTheoSinhNhat();

doDuLieuVaoBang(listKH);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm khách hàng có trạng thái 'Ngưng giao dịch'

\*/

void locTheoTrangThai(){

try {

List<KhachHang> listKH = KhachHangDAO.timKHNgungHoatDong();

doDuLieuVaoBang(listKH);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng tbl\_DanhSachKhachHang

\*/

void doDuLieuVaoBang(List<KhachHang> listKH) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachKhachHang.getModel();

dtm.setRowCount(0);

for(KhachHang KH : listKH){

float TongGiaoDich = ThongKeDAO.tongGiaoDichKH(KH.getMaKhachHang());

Object[] row = {false,

KH.getMaKhachHang(),

KH.getHoTen(),

KH.getDienThoai(),

TongGiaoDich,

KH.getDiemThuong(),

KH.getTrangThai() ? "Đang giao dịch" : "Ngưng giao dịch"

};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm khách hàng theo ngày giao dịch cuối cùng

\*/

void timTheoGDCuoi() {

String NgayTim = DateHelper.toString(dat\_GiaoDichCuoi.getDate(), "yyyy-MM-dd");

if (!NgayTim.isEmpty()) {

try {

List<KhachHang> listKH = KhachHangDAO.timTheoGDCuoi(NgayTim);

doDuLieuVaoBang(listKH);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm tổng hợp theo mã, tên, SĐT

\*/

void timTongHop() {

String ChuoiTimKiem = txt\_TimTheoMaTenSDT.getText();

if (!ChuoiTimKiem.isEmpty()) {

try {

List<KhachHang> listKH = KhachHangDAO.timTheoMaTenSDT(ChuoiTimKiem);

doDuLieuVaoBang(listKH);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

capnhatDuLieu();

}

}

// list chua ma khach hang duoc chon de ngung giao dich

static List<String> listKH = new ArrayList<>();

/\*

\* Hàm tạo danh sách các khách hàng đang được chọn trên tbl\_DanhSachKhachHang

\*/

void taoDanhSachDangChon() {

int i = tbl\_DanhSachKhachHang.getSelectedRow();

if(i >= 0){

String MaKH = tbl\_DanhSachKhachHang.getValueAt(i, 1).toString();

if((Boolean)tbl\_DanhSachKhachHang.getValueAt(i, 0) && !MaKH.equalsIgnoreCase("KH0")){

if(!(listKH.indexOf(MaKH) >= 0)) {

listKH.add(MaKH);

}

}

else {

if((listKH.indexOf(MaKH) >= 0)){

listKH.remove(MaKH);

}

}

}

if (!listKH.isEmpty()) {

btn\_NgungGiaoDich.setVisible(true);

}

else {

btn\_NgungGiaoDich.setVisible(false);

}

}

/\*

\* Hàm cập nhật trạng thái ngưng giao dịch cho khách hàng

\*/

void ngungGiaoDich() {

if(DialogHelper.confirm(this, "Chuyển trạng thái ngưng giao dịch cho các khách hàng này?")) {

try {

int i = 0;

for(String MaKH : listKH) {

i = KhachHangDAO.ngungGiaoDich(MaKH);

}

if(i > 0) {

DialogHelper.alert(this, "Cập nhật thành công");

listKH.clear();

btn\_NgungGiaoDich.setVisible(false);

capnhatDuLieu();

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm mở frame cập nhật khách hàng

và truyền model KhachHang đang được chọn qua frame cập nhật

\*/

void capnhatKhachHang() {

int index = tbl\_DanhSachKhachHang.getSelectedRow();

String MaKH = tbl\_DanhSachKhachHang.getValueAt(index, 1).toString();

if (!MaKH.equalsIgnoreCase("KH0")) {

KhachHang KH = KhachHangDAO.timTheoMa(MaKH);

new CapNhatKhachHangJFrame(KH).setVisible(true);

}

else {

DialogHelper.alert(this, "Không được quyền sửa khách mặc định"); }}

* + - 1. **ThemMoiKhachHangJFrame**

KhachHangDAO KhachHangDAO = new KhachHangDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt(){

this.setSize(530,600);

this.setMinimumSize(new Dimension(530, 600));

this.setMaximumSize(new Dimension(530, 600));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm tự động đổ mã khách hàng và NVQL lên textfield

\*/

void doThongTin(){

String MaNhanVien = ShareHelper.NguoiDung.getTenDangNhap();

txt\_NhanVienQuanLy.setText(MaNhanVien);

try {

int dem = KhachHangDAO.tongKhachHang();

String MaKhachHang = "KH" + dem;

txt\_MaKhachHang.setText(MaKhachHang);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm xoá thông tin trên textfield để nhập lại

\*/

void xoaThongTin(){

txt\_HoTen.setText("");

txt\_HoTen.setBackground(Color.white);

txt\_DienThoai.setText("");

txt\_DienThoai.setBackground(Color.white);

((JTextField)dat\_NgaySinh.getDateEditor().getUiComponent()).setText("");

((JTextField)dat\_NgaySinh.getDateEditor().getUiComponent()).setBackground(Color.white);

txt\_Email.setText("");

txt\_Email.setBackground(Color.white);

txa\_DiaChi.setText("");

txa\_DiaChi.setBackground(Color.white);

txa\_GhiChu.setText("");

txa\_GhiChu.setBackground(Color.white);

rad\_Nam.setSelected(true);

cmb\_TrangThai.setSelectedIndex(0);

}

/\*

\* Hàm kiểm tra sdt có tồn tại trong database hay không

\*/

boolean checkDienThoai(){

boolean ketqua = true;

try {

String SDT = txt\_DienThoai.getText();

if(SDT.isEmpty() || SDT.length() < 10 || SDT.length() > 11){

ketqua = false;

txt\_DienThoai.setBackground(Color.yellow);

}

else if(!KhachHangDAO.kiemSDT(SDT)){

ketqua = false;

DialogHelper.alert(this, "Số điện thoại đã tồn tại");

txt\_DienThoai.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra định dạng email

\*/

boolean checkEmail(){

try {

String email = txt\_Email.getText();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if(email.length() <= 0 | (!email.matches(pattern))){

return false;

}

if(!KhachHangDAO.kiemEmail(email)){

DialogHelper.alert(this, "Email đã tồn tại");

return false;

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return true;

}

/\*

\* Hàm check ngày sinh khách hàng: tính đến thời gian hiện tại phải từ 13t trở lên

\*/

boolean checkNgaySinh(){

if(dat\_NgaySinh.getDate() != null){

Date NgayHienTai = DateHelper.now();

Date NgaySinh = dat\_NgaySinh.getDate();

long kiemtra = DateHelper.getDateDiff(NgaySinh, NgayHienTai, TimeUnit.DAYS);

if(kiemtra/365 < 13){

DialogHelper.alert(this, "Khách hàng phải đủ 13 tuổi");

return false;

}

}

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi(){

if(txa\_DiaChi.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm thêm khách hàng vào database

sau khi thêm thì close frame này

\*/

void themKhachHang(){

try {

if(kiemLoi()){

KhachHang model = taoModel();

int rs = KhachHangDAO.themKhachHang(model);

if(rs != 0){

DialogHelper.alert(this, "Thêm mới thành công");

this.dispose();

if(capnhatFrameDSKH()){

new DanhSachKhachHangJFrame().setVisible(true);

}

}

else {

DialogHelper.alert(this, "Thêm mới thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm kiểm tra họ tên

\*/

boolean checkHoTen(){

String pattern = "^[A-Za-zÀÁÂÃÈÉÊÌÍÒÓÔÕÙÚÝàáâãèéêìíòóôõùúýĂăĐđĨĩŨũƠơƯưẠ-ỹ\\s]+$";

if(!txt\_HoTen.getText().matches(pattern)){

return false;

}

return true;

}

/\*

\* Hàm kiểm lỗi trước khi thêm Khách hàng

\*/

boolean kiemLoi(){

if(!checkDiaChi() || !checkEmail() || !checkNgaySinh() || !checkDienThoai() || !checkHoTen())

return false;

return true;

}

/\*

\* Hàm tạo model khách hàng để đưa lên database

\*/

KhachHang taoModel(){

KhachHang model = new KhachHang();

model.setMaKhachHang(txt\_MaKhachHang.getText());

model.setHoTen(txt\_HoTen.getText());

model.setDiemThuong(0);

model.setDienThoai(txt\_DienThoai.getText());

model.setGioiTinh(rad\_Nam.isSelected());

model.setNgaySinh(dat\_NgaySinh.getDate());

model.setNVQuanLy(txt\_NhanVienQuanLy.getText());

model.setEmail(txt\_Email.getText());

model.setDiaChi(txa\_DiaChi.getText());

model.setGhiChu(txa\_GhiChu.getText());

model.setTrangThai(true);

return model;

}

/\*

\* Hàm reload lại frame DANH SÁCH KHÁCH HÀNG để cập nhật khách hàng mới

hàm chỉ thực thi khi frame DSKH đang mở

\*/

boolean capnhatFrameDSKH(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.doitac.khachhang.DanhSachKhachHangJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **CapNhatKhachHangJFrame**

KhachHangDAO KhachHangDAO = new KhachHangDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt(){

this.setSize(530,600);

this.setMinimumSize(new Dimension(530, 600));

this.setMaximumSize(new Dimension(530, 600));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm tự động đổ thông tin khách hàng lên textfield

\*/

void doThongTin(){

txt\_MaKhachHang.setText(model.getMaKhachHang().trim());

txt\_HoTen.setText(model.getHoTen().trim());

txt\_DienThoai.setText(model.getDienThoai().trim());

if(model.getGioiTinh())

rad\_Nam.setSelected(true);

else

rad\_Nu.setSelected(true);

dat\_NgaySinh.setDate(model.getNgaySinh());

txt\_NhanVienQuanLy.setText(model.getNVQuanLy().trim());

txt\_Email.setText(model.getEmail().trim());

txa\_DiaChi.setText(model.getDiaChi());

if(model.getTrangThai())

cmb\_TrangThai.setSelectedIndex(0);

else

cmb\_TrangThai.setSelectedIndex(1);

txa\_GhiChu.setText(model.getGhiChu());

txt\_DiemThuong.setText(model.getDiemThuong()+"");

}

/\*

\* Hàm xoá thông tin trên textfield để nhập lại

\*/

void xoaThongTin(){

txt\_HoTen.setBackground(Color.white);

txt\_DienThoai.setBackground(Color.white);

((JTextField)dat\_NgaySinh.getDateEditor().getUiComponent()).setBackground(Color.white);

txt\_Email.setBackground(Color.white);

txa\_DiaChi.setBackground(Color.white);

txa\_GhiChu.setBackground(Color.white);

doThongTin();

}

/\*

\* Hàm kiểm tra sdt có tồn tại trong database hay không

\*/

boolean checkDienThoai(){

boolean ketqua = true;

try {

String SDT = txt\_DienThoai.getText();

if(SDT.isEmpty() || SDT.length() < 10 || SDT.length() > 11){

ketqua = false;

txt\_DienThoai.setBackground(Color.yellow);

}

else if(!SDT.equalsIgnoreCase(model.getDienThoai()) && !KhachHangDAO.kiemSDT(SDT)){

ketqua = false;

DialogHelper.alert(this, "Số điện thoại đã tồn tại");

txt\_DienThoai.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra định dạng email

\*/

boolean checkEmail(){

try {

String email = txt\_Email.getText().trim();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if(txt\_Email.getText().length() <= 0 | (!email.matches(pattern))){

return false;

}

if(!email.equalsIgnoreCase(model.getEmail().trim()) && !KhachHangDAO.kiemEmail(email)){

DialogHelper.alert(this, "Email đã tồn tại");

return false;

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return true;

}

/\*

\* Hàm check ngày sinh khách hàng: tính đến thời gian hiện tại phải từ 13t trở lên

\*/

boolean checkNgaySinh(){

if(dat\_NgaySinh.getDate() != null){

Date NgayHienTai = DateHelper.now();

Date NgaySinh = dat\_NgaySinh.getDate();

long kiemtra = DateHelper.getDateDiff(NgaySinh, NgayHienTai, TimeUnit.DAYS);

if(kiemtra/365 < 13){

DialogHelper.alert(this, "Khách hàng phải đủ 13 tuổi");

return false;

}

}

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi(){

if(txa\_DiaChi.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm cập nhật thông tin khách hàng vào database

sau khi thêm thì close frame này

\*/

void capnhatKhachHang(){

try {

if(kiemLoi()){

KhachHang KH = taoModel();

int rs = KhachHangDAO.capnhatKhachHang(KH);

if(rs != 0){

DialogHelper.alert(this, "Cập nhật thành công");

this.dispose();

if(capnhatFrameDSKH()){

new DanhSachKhachHangJFrame().setVisible(true);

}

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm kiểm tra họ tên

\*/

boolean checkHoTen(){

String pattern = "^[A-Za-zÀÁÂÃÈÉÊÌÍÒÓÔÕÙÚÝàáâãèéêìíòóôõùúýĂăĐđĨĩŨũƠơƯưẠ-ỹ\\s]+$";

if(!txt\_HoTen.getText().matches(pattern)){

return false;

}

return true;

}

/\*

\* Hàm kiểm lỗi trước khi cập nhật Khách hàng

\*/

boolean kiemLoi(){

if(!checkDiaChi() || !checkEmail() || !checkNgaySinh() || !checkDienThoai() || !checkHoTen())

return false;

return true;

}

/\*

\* Hàm tạo model khách hàng để đưa lên database

\*/

KhachHang taoModel(){

KhachHang KH = new KhachHang();

KH.setMaKhachHang(txt\_MaKhachHang.getText());

KH.setHoTen(txt\_HoTen.getText());

KH.setDiemThuong(Integer.parseInt(txt\_DiemThuong.getText()));

KH.setDienThoai(txt\_DienThoai.getText());

KH.setGioiTinh(rad\_Nam.isSelected());

KH.setNgaySinh(dat\_NgaySinh.getDate());

KH.setNVQuanLy(txt\_NhanVienQuanLy.getText());

KH.setEmail(txt\_Email.getText());

KH.setDiaChi(txa\_DiaChi.getText());

KH.setGhiChu(txa\_GhiChu.getText());

KH.setTrangThai(true);

return KH;

}

/\*

\* Hàm reload lại frame DANH SÁCH KHÁCH HÀNG để cập nhật khách hàng mới

hàm chỉ thực thi khi frame DSKH đang mở

\*/

boolean capnhatFrameDSKH(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.doitac.khachhang.DanhSachKhachHangJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **DanhSachNhaCungCapJFrame**

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

ThongKeDAO ThongKeDAO = new ThongKeDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// list chua maNCC duoc chon de ngung giao dich

static List<String> listNgungGD = new ArrayList<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu của Frame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

btn\_NgungGiaoDich.setVisible(false);

}

/\*

\* Hàm cập nhật dữ liệu NCC

\*/

void capnhatDuLieu(){

try {

List<NhaCungCap> listNCC = NCCDAO.select();

doDuLieuVaoBang(listNCC);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng tbl\_DanhSachNCC

\*/

void doDuLieuVaoBang(List<NhaCungCap> listNCC) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachNhaCungCap.getModel();

dtm.setRowCount(0);

for(NhaCungCap NCC : listNCC){

double TongNhap = NCCDAO.tongTienNhap(NCC.getMaNCC());

double TongTra = NCCDAO.tongTienTra(NCC.getMaNCC());

Object[] row = {false,

NCC.getMaNCC(),

NCC.getTenNCC(),

NCC.getDienThoai(),

dcf.format(TongNhap),

dcf.format(TongTra),

dcf.format(TongNhap - TongTra)

};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm tổng hợp theo mã, tên, SĐT

\*/

void timTongHop() {

String ChuoiTimKiem = txt\_TimTheoMaTenSDT.getText();

if (!ChuoiTimKiem.isEmpty()) {

try {

List<NhaCungCap> listNCC = NCCDAO.timTheoMaTenSDT(ChuoiTimKiem);

doDuLieuVaoBang(listNCC);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

capnhatDuLieu();

}

}

/\*

\* Hàm lọc NCC theo trạng thái

\*/

void locTheoTrangThai(int i) {

try {

List<NhaCungCap> listNCC;

if (i == 1) {

listNCC = NCCDAO.locTheoTrangThai(1);

}

else

listNCC = NCCDAO.locTheoTrangThai(0);

doDuLieuVaoBang(listNCC);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tạo danh sách các NCC đang được chọn trên table

\*/

void taoDanhSachDangChon() {

int i = tbl\_DanhSachNhaCungCap.getSelectedRow();

if(i >= 0){

String MaNCC = tbl\_DanhSachNhaCungCap.getValueAt(i, 1).toString();

if((Boolean)tbl\_DanhSachNhaCungCap.getValueAt(i, 0)){

if(!(listNgungGD.indexOf(MaNCC) >= 0)) {

listNgungGD.add(MaNCC);

}

}

else {

if((listNgungGD.indexOf(MaNCC) >= 0)){

listNgungGD.remove(MaNCC);

}

}

}

if (!listNgungGD.isEmpty()) {

btn\_NgungGiaoDich.setVisible(true);

}

else {

btn\_NgungGiaoDich.setVisible(false);

}

}

/\*

\* Hàm cập nhật trạng thái ngưng giao dịch cho NCC

\*/

void ngungGiaoDich() {

if(DialogHelper.confirm(this, "Chuyển trạng thái ngưng giao dịch cho các NCC này?")) {

try {

int i = 0;

for(String MaNCC : listNgungGD) {

i = NCCDAO.ngungGiaoDich(MaNCC);

}

if(i > 0) {

DialogHelper.alert(this, "Cập nhật thành công");

listNgungGD.clear();

btn\_NgungGiaoDich.setVisible(false);

capnhatDuLieu();

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm mở frame cập nhật NCC

và truyền model NCC đang được chọn qua frame cập nhật

\*/

void capnhatNCC() {

try {

int index = tbl\_DanhSachNhaCungCap.getSelectedRow();

String MaNCC = tbl\_DanhSachNhaCungCap.getValueAt(index, 1).toString();

NhaCungCap NCC = NCCDAO.timTheoMa(MaNCC);

new CapNhatNhaCungCapJFrame(NCC).setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **ThemMoiNhaCungCapJFrame**

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(530,520);

this.setMinimumSize(new Dimension(530, 520));

this.setMaximumSize(new Dimension(530, 520));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm tự động đổ mã NCC và NVQL lên textfield

\*/

void doThongTin(){

try {

int dem = NCCDAO.tongNCC() + 1;

String MaNCC = "NCC" + dem;

txt\_MaNhaCungCap.setText(MaNCC);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm xoá thông tin trên textfield để nhập lại

\*/

void xoaThongTin(){

txt\_TenNhaCungCap.setText("");

txt\_TenNhaCungCap.setBackground(Color.white);

txt\_DienThoai.setText("");

txt\_DienThoai.setBackground(Color.white);

txt\_Email.setText("");

txt\_Email.setBackground(Color.white);

txa\_DiaChi.setText("");

txa\_DiaChi.setBackground(Color.white);

txa\_GhiChu.setText("");

txa\_GhiChu.setBackground(Color.white);

cmb\_TrangThai.setSelectedIndex(0);

}

/\*

\* Hàm kiểm tra sdt có tồn tại trong database hay không

\*/

boolean checkDienThoai(){

boolean ketqua = true;

try {

String SDT = txt\_DienThoai.getText();

if(SDT.isEmpty() || SDT.length() < 10 || SDT.length() > 11){

ketqua = false;

txt\_DienThoai.setBackground(Color.yellow);

}

else if(!NCCDAO.kiemSDT(SDT)){

ketqua = false;

DialogHelper.alert(this, "Số điện thoại đã tồn tại");

txt\_DienThoai.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra định dạng email

\*/

boolean checkEmail(){

try {

String email = txt\_Email.getText();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if(email.length() <= 0 | (!email.matches(pattern))){

return false;

}

if(!NCCDAO.kiemEmail(email)){

DialogHelper.alert(this, "Email đã tồn tại");

return false;

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi(){

if(txa\_DiaChi.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm check ghi chú

\*/

boolean checkGhiChu(){

if(txa\_GhiChu.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm thêm NCC vào database

sau khi thêm thì close frame này

\*/

void themNCC(){

try {

if(kiemLoi()){

NhaCungCap model = taoModel();

int rs = NCCDAO.themNCC(model);

if(rs != 0){

DialogHelper.alert(this, "Thêm mới thành công");

this.dispose();

if(capnhatFrameDSNCC()){

new DanhSachNhaCungCapJFrame().setVisible(true);

}

}

else {

DialogHelper.alert(this, "Thêm mới thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm kiểm tra tên NCC

\*/

boolean checkTenNCC(){

String pattern = "^[A-Za-zÀÁÂÃÈÉÊÌÍÒÓÔÕÙÚÝàáâãèéêìíòóôõùúýĂăĐđĨĩŨũƠơƯưẠ-ỹ\\s]+$";

if(!txt\_TenNhaCungCap.getText().matches(pattern)){

return false;

}

return true;

}

/\*

\* Hàm kiểm lỗi trước khi thêm NCC

\*/

boolean kiemLoi(){

if(!checkDiaChi() || !checkEmail() || !checkDienThoai() || !checkTenNCC() || !checkGhiChu())

return false;

return true;

}

/\*

\* Hàm tạo model NCC để đưa lên database

\*/

NhaCungCap taoModel(){

NhaCungCap model = new NhaCungCap();

model.setMaNCC(txt\_MaNhaCungCap.getText());

model.setTenNCC(txt\_TenNhaCungCap.getText());

model.setDienThoai(txt\_DienThoai.getText());

model.setEmail(txt\_Email.getText());

model.setDiaChi(txa\_DiaChi.getText());

model.setGhiChu(txa\_GhiChu.getText());

model.setTrangThai(true);

return model;

}

/\*

\* Hàm reload lại frame DANH SÁCH NCC để cập nhật NCC mới

\*/

boolean capnhatFrameDSNCC(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.doitac.nhacungcap.DanhSachNhaCungCapJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **CapNhatNhaCungCapJFrame**

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(530,520);

this.setMinimumSize(new Dimension(530, 520));

this.setMaximumSize(new Dimension(530, 520));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm tự động đổ thông tin NCC lên textfield

\*/

void doThongTin(){

txt\_MaNhaCungCap.setText(model.getMaNCC().trim());

txt\_TenNhaCungCap.setText(model.getTenNCC().trim());

txt\_DienThoai.setText(model.getDienThoai().trim());

txt\_Email.setText(model.getEmail().trim());

txa\_DiaChi.setText(model.getDiaChi());

if(model.getTrangThai())

cmb\_TrangThai.setSelectedIndex(0);

else

cmb\_TrangThai.setSelectedIndex(1);

txa\_GhiChu.setText(model.getGhiChu());

}

/\*

\* Hàm xoá thông tin trên textfield để nhập lại

\*/

void xoaThongTin(){

txt\_TenNhaCungCap.setBackground(Color.white);

txt\_DienThoai.setBackground(Color.white);

txt\_Email.setBackground(Color.white);

txa\_DiaChi.setBackground(Color.white);

txa\_GhiChu.setBackground(Color.white);

doThongTin();

}

/\*

\* Hàm kiểm tra sdt có tồn tại trong database hay không

\*/

boolean checkDienThoai(){

boolean ketqua = true;

try {

String SDT = txt\_DienThoai.getText();

if(SDT.isEmpty() || SDT.length() < 10 || SDT.length() > 11){

ketqua = false;

txt\_DienThoai.setBackground(Color.yellow);

}

else if(!NCCDAO.kiemSDT(SDT) && !SDT.equalsIgnoreCase(model.getDienThoai())){

ketqua = false;

DialogHelper.alert(this, "Số điện thoại đã tồn tại");

txt\_DienThoai.setBackground(Color.yellow);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return ketqua;

}

/\*

\* Hàm kiểm tra định dạng email

\*/

boolean checkEmail(){

try {

String email = txt\_Email.getText();

String pattern = "([A-Za-z0-9-\_.]+@[A-Za-z0-9-\_]+(?:\\.[A-Za-z0-9]+)+)";

if(email.length() <= 0 | (!email.matches(pattern))){

return false;

}

if(!NCCDAO.kiemEmail(email) && !email.equalsIgnoreCase(model.getEmail())){

DialogHelper.alert(this, "Email đã tồn tại");

return false;

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

return true;

}

/\*

\* Hàm check địa chỉ

\*/

boolean checkDiaChi(){

if(txa\_DiaChi.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm check ghi chú

\*/

boolean checkGhiChu(){

if(txa\_GhiChu.getText().isEmpty()){

return false;

}

else

return true;

}

/\*

\* Hàm cập nhật NCC vào database

sau khi thêm thì close frame này

\*/

void capnhatNCC(){

try {

if(kiemLoi()){

NhaCungCap model = taoModel();

int rs = NCCDAO.capnhatNCC(model);

if(rs != 0){

DialogHelper.alert(this, "Cập nhật thành công");

this.dispose();

if(capnhatFrameDSNCC()){

new DanhSachNhaCungCapJFrame().setVisible(true);

}

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm kiểm tra tên NCC

\*/

boolean checkTenNCC(){

String pattern = "^[A-Za-zÀÁÂÃÈÉÊÌÍÒÓÔÕÙÚÝàáâãèéêìíòóôõùúýĂăĐđĨĩŨũƠơƯưẠ-ỹ\\s]+$";

if(!txt\_TenNhaCungCap.getText().matches(pattern)){

return false;

}

return true;

}

/\*

\* Hàm kiểm lỗi trước khi thêm NCC

\*/

boolean kiemLoi(){

if(!checkDiaChi() || !checkEmail() || !checkDienThoai() || !checkTenNCC() || !checkGhiChu())

return false;

return true;

}

/\*

\* Hàm tạo model NCC để đưa lên database

\*/

NhaCungCap taoModel(){

NhaCungCap model = new NhaCungCap();

model.setMaNCC(txt\_MaNhaCungCap.getText());

model.setTenNCC(txt\_TenNhaCungCap.getText());

model.setDienThoai(txt\_DienThoai.getText());

model.setEmail(txt\_Email.getText());

model.setDiaChi(txa\_DiaChi.getText());

model.setGhiChu(txa\_GhiChu.getText());

model.setTrangThai(true);

return model;

}

/\*

\* Hàm reload lại frame DANH SÁCH NCC để cập nhật NCC mới

\*/

boolean capnhatFrameDSNCC(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.doitac.nhacungcap.DanhSachNhaCungCapJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **DanhSachSanPhamJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// list chua maSP duoc chon de ngung kinh doanh

static List<String> listNgungKD = new ArrayList<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

doNhomHang();

doNCC();

capnhatDuLieu();

btn\_NgungKinhDoanh.setVisible(false);

}

/\*

\* Hàm này đổ danh sách nhóm hàng vào combobox ở tab DanhSachSanPham

\*/

void doNhomHang() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

try {

List<NhomHang> ListNH = NhomHangDAO.select();

dcm.addElement("Tất cả");

for (NhomHang NH : ListNH) {

dcm.addElement(NH);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách Nhà cung cấp vào combobox

\*/

void doNCC() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhaCungCap.getModel();

dcm.removeAllElements();

try {

List<NhaCungCap> ListNCC = NCCDAO.select();

dcm.addElement("Tất cả");

for (NhaCungCap NCC : ListNCC) {

dcm.addElement(NCC);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm cập nhật dữ liệu Sản Phẩm

\*/

void capnhatDuLieu(){

try {

List<SanPham> listSP = SanPhamDAO.select();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<SanPham> listSP) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachSanPham.getModel();

dtm.setRowCount(0);

for(SanPham SP : listSP){

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

Object[] row = {false,

SP.getMaSanPham(),

SP.getTenSanPham(),

DVT.getMoTa(),

dcf.format(SP.getGiaVon()),

dcf.format(SP.getGiaBan()),

SP.getTonKho()

};

dtm.addRow(row);

}

}

/\*

\* Hàm lọc sản phẩm dưới định mức tồn kho

\*/

void locDuoiDinhMuc() {

try {

List<SanPham> listSP = SanPhamDAO.locDuoiDinhMuc();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm lọc sản phẩm vượt định mức tồn kho

\*/

void locVuotDinhMuc() {

try {

List<SanPham> listSP = SanPhamDAO.locVuotDinhMuc();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm lọc sản phẩm còn hàng

\*/

void locConHang() {

try {

List<SanPham> listSP = SanPhamDAO.locConHang();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm lọc sản phẩm hết hàng

\*/

void locHetHang() {

try {

List<SanPham> listSP = SanPhamDAO.locHetHang();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm lọc sản phẩm đang kinh doanh

\*/

void locDangKinhDoanh() {

try {

List<SanPham> listSP = SanPhamDAO.locDangKinhDoanh();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm lọc sản phẩm ngừng kinh doanh

\*/

void locNgungKinhDoanh() {

try {

List<SanPham> listSP = SanPhamDAO.locNgungKinhDoanh();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm sản phẩm theo mã hoặc tên

\*/

void timTheoMaHoacTen() {

String ChuoiTimKiem = txt\_TimTheoTenHoacMa.getText();

try {

List<SanPham> listSP = SanPhamDAO.timTheoMaHoacTen(ChuoiTimKiem);

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm lọc SP theo nhà cung cấp (chọn từ combobox)

\*/

void locTheoNhaCungCap() {

if (cmb\_NhaCungCap.getSelectedIndex() == 0) {

capnhatDuLieu();

}

else {

NhaCungCap NCC = (NhaCungCap) cmb\_NhaCungCap.getSelectedItem();

try {

List<SanPham> listSP = SanPhamDAO.timTheoNCC(NCC.getMaNCC());

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm lọc SP theo nhóm hàng (chọn từ combobox)

\*/

void locTheoNhomHang() {

if (cmb\_NhomHang.getSelectedIndex() == 0) {

capnhatDuLieu();

}

else {

try {

NhomHang NH = (NhomHang) cmb\_NhomHang.getSelectedItem();

List<SanPham> listSP = SanPhamDAO.timTheoMaNhom(NH.getMaNhomHang());

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tạo danh sách các sản phẩm đang được chọn trên table

\*/

void taoDanhSachDangChon() {

int i = tbl\_DanhSachSanPham.getSelectedRow();

if(i >= 0){

String MaSP = tbl\_DanhSachSanPham.getValueAt(i, 1).toString();

if((Boolean)tbl\_DanhSachSanPham.getValueAt(i, 0)){

if(!(listNgungKD.indexOf(MaSP) >= 0)) {

listNgungKD.add(MaSP);

}

}

else {

if((listNgungKD.indexOf(MaSP) >= 0)){

listNgungKD.remove(MaSP);

}

}

}

if (!listNgungKD.isEmpty()) {

btn\_NgungKinhDoanh.setVisible(true);

}

else {

btn\_NgungKinhDoanh.setVisible(false);

}

}

/\*

\* Hàm cập nhật trạng thái ngưng kinh doanh cho SP

\*/

void ngungKinhDoanh() {

if(DialogHelper.confirm(this, "Chuyển trạng thái ngưng kinh doanh cho các sản phẩm này?")) {

try {

int i = 0;

for(String MaSP : listNgungKD) {

i = SanPhamDAO.ngungKinhDoanh(MaSP);

}

if(i > 0) {

DialogHelper.alert(this, "Cập nhật thành công");

listNgungKD.clear();

btn\_NgungKinhDoanh.setVisible(false);

capnhatDuLieu();

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm truyền model SanPham vào frame Cập nhật và mở fram cập nhật

\*/

void capnhatSanPham() {

try {

int index = tbl\_DanhSachSanPham.getSelectedRow();

String MaSP = tbl\_DanhSachSanPham.getValueAt(index, 1).toString();

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

new CapNhatSanPhamJFrame(SP).setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **ThemMoiSanPhamJFrame**

SanPhamDAO SanPhamDAO = new SanPhamDAO();

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 635);

this.setMinimumSize(new Dimension(1366, 635));

this.setMaximumSize(new Dimension(1366, 635));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm tự động đổ các ttin maSP, nhóm hàng, NCC, ĐVT lên textfield/combobox

\*/

void doThongTin(){

try {

doMaSP();

doNhomHang();

doNCC();

doDVT();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

void doMaSP() throws SQLException {

int dem = SanPhamDAO.demSanPham() + 1;

String MaSP = "SP" + dem;

txt\_MaSanPham.setText(MaSP);

}

void doNhomHang() throws SQLException {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

List<NhomHang> listNH = NhomHangDAO.select();

for (NhomHang NH : listNH) {

dcm.addElement(NH);

}

}

void doNCC() throws SQLException {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhaCungCap.getModel();

dcm.removeAllElements();

List<NhaCungCap> listNCC = NCCDAO.select();

for (NhaCungCap NCC : listNCC) {

// chỉ hiện những NCC có trạng thái 'đang giao dịch'

if (NCC.getTrangThai()) {

dcm.addElement(NCC);

}

}

}

void doDVT() throws SQLException {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_DonViTinh.getModel();

dcm.removeAllElements();

List<DonViTinh> listDVT = DVTDAO.select();

for (DonViTinh DVT : listDVT) {

dcm.addElement(DVT);

}

}

/\*

\* Hàm xoá thông tin để nhập lại

\*/

void xoaThongTin() {

doThongTin();

txt\_TenSanPham.setText("");

txt\_TenSanPham.setBackground(Color.white);

txt\_MaVach.setText("");

txt\_MaVach.setBackground(Color.white);

txt\_DinhMucTonLonNhat.setText("");

txt\_DinhMucTonLonNhat.setBackground(Color.white);

txt\_DinhMucTonNhoNhat.setText("");

txt\_DinhMucTonNhoNhat.setBackground(Color.white);

txt\_GiaVon.setText("");

txt\_GiaVon.setBackground(Color.white);

txt\_GiaBan.setText("");

txt\_GiaBan.setBackground(Color.white);

lbl\_Hinh.setIcon(null);

lbl\_Hinh.setToolTipText("");

((JTextField)dat\_HSDMoi.getDateEditor().getUiComponent()).setBackground(Color.white);

((JTextField)dat\_HSDMoi.getDateEditor().getUiComponent()).setText("");

((JTextField)dat\_HSDGanNhat.getDateEditor().getUiComponent()).setBackground(Color.white);

((JTextField)dat\_HSDGanNhat.getDateEditor().getUiComponent()).setText("");

}

/\*

\* Hàm check tên sản phẩm

- ký tự tối thiểu: 5

- ký tự tối đa: 50 (đã bắt lỗi trực tiếp event KeyTyped)

\*/

boolean checkTenSanPham() {

String TenSanPham = txt\_TenSanPham.getText();

if (TenSanPham.length() < 5)

return false;

else

return true;

}

/\*

\* Hàm check định mức tồn nhỏ nhất

- không được để trống

\*/

boolean checkDinhMucNho() {

if (txt\_DinhMucTonNhoNhat.getText().length() < 1)

return false;

else

return true;

}

/\*

\* Hàm check định mức tồn lớn nhất

- không được để trống

\*/

boolean checkDinhMucLon() {

if (txt\_DinhMucTonLonNhat.getText().length() < 1)

return false;

else

return true;

}

/\*

\* Hàm check giá vốn

- không được để trống

- tối thiểu 3 chữ số

\*/

boolean checkGiaVon() {

if (txt\_GiaVon.getText().length() < 3)

return false;

else

return true;

}

/\*

\* Hàm check giá bán

- không được để trống

- tối thiểu 4 chữ số

\*/

boolean checkGiaBan() {

if (txt\_GiaBan.getText().length() < 4)

return false;

else

return true;

}

/\*

\* Hàm kiểm tổng lỗi trước khi thêm sản phẩm

\*/

boolean kiemLoi() {

if (checkTenSanPham() && checkDinhMucNho() && checkDinhMucLon() && checkGiaBan() && checkGiaVon())

return true;

else

return false;

}

/\*

\* Hàm tạo model Sản Phẩm để thêm vào database

\*/

SanPham taoModel() {

SanPham SP = new SanPham();

String MaNhomHang = ((NhomHang) cmb\_NhomHang.getSelectedItem()).getMaNhomHang();

String MaNCC = ((NhaCungCap) cmb\_NhaCungCap.getSelectedItem()).getMaNCC();

int MaDVT = ((DonViTinh) cmb\_DonViTinh.getSelectedItem()).getMaDVT();

SP.setMaSanPham(txt\_MaSanPham.getText().trim());

SP.setTenSanPham(txt\_TenSanPham.getText());

SP.setMaVach(txt\_MaVach.getText());

SP.setMaNhomHang(MaNhomHang);

SP.setMaNCC(MaNCC);

SP.setMaDVT(MaDVT);

SP.setTonKho(0);

SP.setDinhMucNhoNhat(Integer.parseInt(txt\_DinhMucTonNhoNhat.getText()));

SP.setDinhMucLonNhat(Integer.parseInt(txt\_DinhMucTonLonNhat.getText()));

SP.setGiaVon(Float.parseFloat(txt\_GiaVon.getText()));

SP.setGiaBan(Float.parseFloat(txt\_GiaBan.getText()));

SP.setTrangThai(false);

// kiểm tra ảnh sản phẩm trước khi thêm thuộc tính ảnh:

if (lbl\_Hinh.getIcon() != null)

SP.setHinhAnh(txt\_MaSanPham.getText().toLowerCase() + ".jpg");

else

SP.setHinhAnh("default.jpg");

return SP;

}

/\*

\* Hàm thêm sản phẩm lên database và cập nhật lại frame DanhSachSanPham

\*/

void themSanPham() {

try {

if(kiemLoi()){

SanPham model = taoModel();

int rs = SanPhamDAO.themSanPham(model);

if(rs != 0){

DialogHelper.alert(this, "Thêm mới thành công");

this.dispose();

if(capnhatFrameDSSP()){

new DanhSachSanPhamJFrame().setVisible(true);

}

}

else {

DialogHelper.alert(this, "Thêm mới thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm reload lại frame DANH SÁCH SP để cập nhật SP mới

\*/

boolean capnhatFrameDSSP(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.sanpham.dssanpham.DanhSachSanPhamJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

/\*

\* Hàm chọn ảnh cho sản phẩm

\*/

void chonAnh() {

String tenAnh = txt\_MaSanPham.getText();

if (!tenAnh.isEmpty()) {

JFileChooser fileChooser = new JFileChooser();

if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

String fileType = file.getName().substring(file.getName().lastIndexOf("."), file.getName().length());

if (fileType.equalsIgnoreCase(".jpg")) {

String newFileName = (tenAnh + fileType).toLowerCase();

if (ShareHelper.luuAnh(file, newFileName)) {

lbl\_Hinh.setIcon(new ImageIcon(ShareHelper.docAnh(newFileName).getImage().getScaledInstance(270, 220, java.awt.Image.SCALE\_SMOOTH)));

lbl\_Hinh.setToolTipText(newFileName);

}

} else {

DialogHelper.alert(this, "Chỉ được chọn ảnh có địng dạng .jpg");

}

}

} else {

DialogHelper.alert(this, "Kiểm tra lại mã sản phẩm");

}

}

}

* + - 1. **CapNhatSanPhamJFrame**

SanPhamDAO SanPhamDAO = new SanPhamDAO();

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 635);

this.setMinimumSize(new Dimension(1366, 635));

this.setMaximumSize(new Dimension(1366, 635));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

}

/\*

\* Hàm tự động đổ các ttin maSP, nhóm hàng, NCC, ĐVT lên textfield/combobox

\*/

void doThongTin(){

try {

doNhomHang();

doNCC();

doDVT();

doThongTinSP();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

void doThongTinSP() throws SQLException {

txt\_MaSanPham.setText(model.getMaSanPham());

txt\_TenSanPham.setText(model.getTenSanPham());

txt\_MaVach.setText(model.getMaVach());

txt\_TonKho.setText(model.getTonKho()+"");

txt\_DinhMucTonLonNhat.setText(model.getDinhMucLonNhat()+"");

txt\_DinhMucTonNhoNhat.setText(model.getDinhMucNhoNhat()+"");

txt\_GiaBan.setText(model.getGiaBan()+"");

txt\_GiaVon.setText(model.getGiaVon()+"");

dat\_HSDMoi.setDate(model.getHSDMoiNhat());

dat\_HSDGanNhat.setDate(model.getHSDTruoc());

chk\_KhongDuocPhepBan.setSelected(!model.getTrangThai());

// đổ hình ảnh lên lbl\_Hinh:

String FileAnh = model.getHinhAnh();

lbl\_Hinh.setIcon(new ImageIcon(ShareHelper.docAnh(FileAnh).getImage().getScaledInstance(270, 220, java.awt.Image.SCALE\_SMOOTH)));

lbl\_Hinh.setToolTipText(FileAnh);

// Đổ thông tin nhóm hàng:

String MaNH = model.getMaNhomHang();

NhomHang NH = NhomHangDAO.timTheoMa(MaNH);

cmb\_NhomHang.getModel().setSelectedItem(NH);

// Đổ thông tin NCC:

String MaNCC = model.getMaNCC();

NhaCungCap NCC = NCCDAO.timTheoMa(MaNCC);

cmb\_NhaCungCap.getModel().setSelectedItem(NCC);

// Đổ thông tin ĐVT:

int MaDVT = model.getMaDVT();

DonViTinh DVT = DVTDAO.timTheoMa(MaDVT);

cmb\_DonViTinh.getModel().setSelectedItem(DVT);

}

void doNhomHang() throws SQLException {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

List<NhomHang> listNH = NhomHangDAO.select();

for (NhomHang NH : listNH) {

dcm.addElement(NH);

}

}

void doNCC() throws SQLException {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhaCungCap.getModel();

dcm.removeAllElements();

List<NhaCungCap> listNCC = NCCDAO.select();

for (NhaCungCap NCC : listNCC) {

dcm.addElement(NCC);

}

}

void doDVT() throws SQLException {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_DonViTinh.getModel();

dcm.removeAllElements();

List<DonViTinh> listDVT = DVTDAO.select();

for (DonViTinh DVT : listDVT) {

dcm.addElement(DVT);

}

}

/\*

\* Hàm xoá thông tin để nhập lại

\*/

void xoaThongTin() {

doThongTin();

txt\_TenSanPham.setBackground(Color.white);

txt\_MaVach.setBackground(Color.white);

txt\_DinhMucTonLonNhat.setBackground(Color.white);

txt\_DinhMucTonNhoNhat.setBackground(Color.white);

txt\_GiaBan.setBackground(Color.white);

((JTextField)dat\_HSDMoi.getDateEditor().getUiComponent()).setBackground(Color.white);

((JTextField)dat\_HSDGanNhat.getDateEditor().getUiComponent()).setBackground(Color.white);

}

/\*

\* Hàm check tên sản phẩm

- ký tự tối thiểu: 5

- ký tự tối đa: 50 (đã bắt lỗi trực tiếp event KeyTyped)

\*/

boolean checkTenSanPham() {

String TenSanPham = txt\_TenSanPham.getText();

if (TenSanPham.length() < 5)

return false;

else

return true;

}

/\*

\* Hàm check định mức tồn nhỏ nhất

- không được để trống

\*/

boolean checkDinhMucNho() {

if (txt\_DinhMucTonNhoNhat.getText().length() < 1)

return false;

else

return true;

}

/\*

\* Hàm check định mức tồn lớn nhất

- không được để trống

\*/

boolean checkDinhMucLon() {

if (txt\_DinhMucTonLonNhat.getText().length() < 1)

return false;

else

return true;

}

/\*

\* Hàm check giá vốn

- không được để trống

- tối thiểu 3 chữ số

\*/

boolean checkGiaVon() {

if (txt\_GiaVon.getText().length() < 3)

return false;

else

return true;

}

/\*

\* Hàm check giá bán

- không được để trống

- tối thiểu 4 chữ số

\*/

boolean checkGiaBan() {

if (txt\_GiaBan.getText().length() < 4)

return false;

else

return true;

}

/\*

\* Hàm kiểm tổng lỗi trước khi thêm sản phẩm

\*/

boolean kiemLoi() {

if (checkTenSanPham() && checkDinhMucNho() && checkDinhMucLon() && checkGiaBan() && checkGiaVon())

return true;

else

return false;

}

/\*

\* Hàm tạo model Sản Phẩm để thêm vào database

\*/

SanPham taoModel() {

SanPham SP = new SanPham();

String MaNhomHang = ((NhomHang) cmb\_NhomHang.getSelectedItem()).getMaNhomHang();

String MaNCC = ((NhaCungCap) cmb\_NhaCungCap.getSelectedItem()).getMaNCC();

int MaDVT = ((DonViTinh) cmb\_DonViTinh.getSelectedItem()).getMaDVT();

SP.setMaSanPham(txt\_MaSanPham.getText().trim());

SP.setTenSanPham(txt\_TenSanPham.getText());

SP.setMaVach(txt\_MaVach.getText());

SP.setMaNhomHang(MaNhomHang);

SP.setMaNCC(MaNCC);

SP.setMaDVT(MaDVT);

// SP.setTonKho(0);

SP.setDinhMucNhoNhat(Integer.parseInt(txt\_DinhMucTonNhoNhat.getText()));

SP.setDinhMucLonNhat(Integer.parseInt(txt\_DinhMucTonLonNhat.getText()));

// SP.setGiaVon(Float.parseFloat(txt\_GiaVon.getText()));

SP.setGiaBan(Float.parseFloat(txt\_GiaBan.getText()));

SP.setTrangThai(!chk\_KhongDuocPhepBan.isSelected());

SP.setHSDMoiNhat(dat\_HSDMoi.getDate());

SP.setHSDTruoc(dat\_HSDGanNhat.getDate());

// kiểm tra ảnh sản phẩm trước khi thêm thuộc tính ảnh:

if (lbl\_Hinh.getIcon() != null)

SP.setHinhAnh(txt\_MaSanPham.getText().toLowerCase() + ".jpg");

else

SP.setHinhAnh("default.jpg");

return SP;

}

/\*

\* Hàm cập nhật sản phẩm và cập nhật lại frame DanhSachSanPham

\*/

void capnhatSanPham() {

try {

if(kiemLoi()){

SanPham model = taoModel();

int rs = SanPhamDAO.capnhatSP(model);

if(rs != 0){

DialogHelper.alert(this, "Cập nhật thành công");

this.dispose();

if(capnhatFrameDSSP()){

new DanhSachSanPhamJFrame().setVisible(true);

}

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại thông tin");

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi cập nhật sản phẩm: " + e.toString());

}

}

/\*

\* Hàm reload lại frame DANH SÁCH SP để cập nhật SP mới

\*/

boolean capnhatFrameDSSP(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.sanpham.dssanpham.DanhSachSanPhamJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

/\*

\* Hàm chọn ảnh cho sản phẩm

\*/

void chonAnh() {

String tenAnh = txt\_MaSanPham.getText();

if (!tenAnh.isEmpty()) {

JFileChooser fileChooser = new JFileChooser();

if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

String fileType = file.getName().substring(file.getName().lastIndexOf("."), file.getName().length());

if (fileType.equalsIgnoreCase(".jpg")) {

String newFileName = (tenAnh + fileType).toLowerCase();

if (ShareHelper.luuAnh(file, newFileName)) {

lbl\_Hinh.setIcon(new ImageIcon(ShareHelper.docAnh(newFileName).getImage().getScaledInstance(270, 220, java.awt.Image.SCALE\_SMOOTH)));

lbl\_Hinh.setToolTipText(newFileName);

}

} else {

DialogHelper.alert(this, "Chỉ được chọn ảnh có địng dạng .jpg");

}

}

} else {

DialogHelper.alert(this, "Kiểm tra lại mã sản phẩm");

}

}

}

* + - 1. **KiemKeSanPhamJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuKiemDAO PhieuKiemDAO = new PhieuKiemDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// list chứa mã phiếu kiểm để huỷ

static List<String> listHuy = new ArrayList<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

btn\_HuyPhieu.setVisible(false);

}

/\*

\* Hàm cập nhật dữ liệu Sản Phẩm

\*/

void capnhatDuLieu(){

try {

rad\_TatCa.setSelected(true);

chk\_PhieuTam.setSelected(true);

chk\_DaCanBang.setSelected(true);

chk\_DaHuy.setSelected(true);

List<PhieuKiem> list = PhieuKiemDAO.select();

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<PhieuKiem> listPK) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuKiem.getModel();

dtm.setRowCount(0);

for(PhieuKiem PK : listPK){

String TrangThai = "";

switch (PK.getTrangThai()) {

case 0:

TrangThai = "Chưa cân bằng";

break;

case 1:

TrangThai = "Đã cân bằng";

break;

case 2:

TrangThai = "Đã huỷ";

break;

}

Object[] row = {false,

PK.getMaPhieuKiem(),

PK.getGhiChu(),

DateHelper.toString(PK.getNgayTao(), "dd-MM-yyyy"),

PK.getNgayCanBang() != null ? DateHelper.toString(PK.getNgayCanBang(), "dd-MM-yyyy") : "",

dcf.format(PK.getGiaTriLech()),

TrangThai

};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm phiếu kiểm theo mã

\*/

void timTheoMa() {

String ChuoiTimKiem = txt\_TimTheoMaChungTu.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

PhieuKiem PK = PhieuKiemDAO.timTheoMa(ChuoiTimKiem);

List<PhieuKiem> list = new ArrayList<>();

if (PK != null)

list.add(PK);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuKiem.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm phiếu kiểm theo mã sản phẩm hoặc tên sản phẩm

\*/

void timTheoSanPham() {

String ChuoiTimKiem = txt\_TimTheoSanPham.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

List<PhieuKiem> list = PhieuKiemDAO.timTheoSanPham(ChuoiTimKiem);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm lọc phiếu kiểm theo các tuỳ chọn

\*/

void locPhieuKiem() {

String TrangThai = "";

if(chk\_PhieuTam.isSelected())

TrangThai += "0";

if(chk\_DaCanBang.isSelected())

TrangThai += "1";

if(chk\_DaHuy.isSelected())

TrangThai += "2";

if (TrangThai.isEmpty())

TrangThai += "012";

if (rad\_TatCa.isSelected())

timTatCa(TrangThai);

if (rad\_HomNay.isSelected())

timTrongNgay(TrangThai);

if (rad\_HomQua.isSelected())

timNgayHomQua(TrangThai);

if (rad\_7NgayTruoc.isSelected())

tim7NgayTruoc(TrangThai);

if (rad\_Khac.isSelected())

timTheoKhoangNgay(TrangThai);

}

/\*

\* Hàm tìm tất cả phiếu kiểm theo trạng thái

\*/

void timTatCa(String TrangThai) {

try {

List<PhieuKiem> list = PhieuKiemDAO.timTheoTrangThai(TrangThai);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu kiểm trong 7 ngày trước (không tính hôm nay)

\*/

void tim7NgayTruoc(String TrangThai) {

try {

List<PhieuKiem> list = PhieuKiemDAO.tim7NgayTruoc(TrangThai);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu kiểm trong ngày

\*/

void timTrongNgay(String TrangThai) {

try {

List<PhieuKiem> list = PhieuKiemDAO.timTrongNgay(TrangThai);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu kiểm tạo ngày hôm qua

\*/

void timNgayHomQua(String TrangThai) {

try {

List<PhieuKiem> list = PhieuKiemDAO.timNgayHomQua(TrangThai);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu kiểm theo khoảng ngày tuỳ chọn

\*/

void timTheoKhoangNgay(String TrangThai) {

String TuNgay = DateHelper.toString(dat\_TuNgay.getDate(),"yyyy-MM-dd");

String DenNgay = DateHelper.toString(dat\_DenNgay.getDate(),"yyyy-MM-dd");

if (dat\_TuNgay.getDate().after(dat\_DenNgay.getDate())) {

DialogHelper.alert(this, "Chọn lại ngày");

}

else {

try {

List<PhieuKiem> list = PhieuKiemDAO.timTheoKhoangNgay(TrangThai, TuNgay, DenNgay);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm mở jframe cập nhật phiếu kiểm

(truyền các thông tin qua frame cập nhật)

\*/

void capnhatPhieuKiem() {

int index = tbl\_DanhSachPhieuKiem.getSelectedRow();

String MaPK = tbl\_DanhSachPhieuKiem.getValueAt(index, 1).toString();

try {

PhieuKiem PK = PhieuKiemDAO.timTheoMa(MaPK);

new CapNhatPhieuKiemJFrame(PK).setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tạo danh sách các phiếu kiểm đang chọn trên table

\*/

void taoDanhSachDangChon() {

int i = tbl\_DanhSachPhieuKiem.getSelectedRow();

if(i >= 0){

String MaPK = tbl\_DanhSachPhieuKiem.getValueAt(i, 1).toString();

if((Boolean)tbl\_DanhSachPhieuKiem.getValueAt(i, 0)){

if(!(listHuy.indexOf(MaPK) >= 0)) {

listHuy.add(MaPK);

}

}

else {

if((listHuy.indexOf(MaPK) >= 0)){

listHuy.remove(MaPK);

}

}

}

if (!listHuy.isEmpty()) {

btn\_HuyPhieu.setVisible(true);

}

else {

btn\_HuyPhieu.setVisible(false);

}

}

/\*

\* Hàm huỷ phiếu kiểm

\*/

void huyPhieuKiem() {

if(DialogHelper.confirm(this, "Huỷ các phiếu kiểm này?")) {

try {

int i = 0;

for(String MaPK : listHuy) {

PhieuKiem PK = PhieuKiemDAO.timTheoMa(MaPK);

i = PhieuKiemDAO.huyPhieuKiem(PK);

}

if(i > 0) {

DialogHelper.alert(this, "Cập nhật thành công");

listHuy.clear();

btn\_HuyPhieu.setVisible(false);

capnhatDuLieu();

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

}

* + - 1. **ThemMoiPhieuKiemJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuKiemDAO PhieuKiemDAO = new PhieuKiemDAO();

PhieuKiemChiTietDAO PKCTDAO = new PhieuKiemChiTietDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// Tạo treemap chứa sản phẩm và số lượng kiểm đếm thực tế

static TreeMap<SanPham, Integer> DSSP = new TreeMap<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

doNhomHang();

doDanhSachSanPham();

doThongTin();

}

/\*

\* Hàm đổ các thông tin mặc định

\*/

void doThongTin() {

try {

// đổ mã nhân viên đang thao tác vào textfield

txt\_MaNhanVien.setText(ShareHelper.NguoiDung.getTenDangNhap());

// khởi tạo mã phiếu kiểm

int demPhieu = PhieuKiemDAO.demPhieuKiem();

String MaPhieuKiem = "PK" + (demPhieu+1);

txt\_MaChungTu.setText(MaPhieuKiem);

// khởi tạo ngày tạo phiếu

dat\_NgayTao.setDate(DateHelper.now());

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách nhóm hàng vào combobox ở tab DanhSachSanPham

\*/

void doNhomHang() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

try {

List<NhomHang> ListNH = NhomHangDAO.select();

dcm.addElement("Tất cả");

for (NhomHang NH : ListNH) {

dcm.addElement(NH);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách sản phẩm vào tab Danh Sách Sản Phẩm

\*/

void doDanhSachSanPham() {

try {

List<SanPham> ListSP = SanPhamDAO.select();

doDuLieuSanPham(ListSP);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này tìm sản phẩm ở tab DANH SÁCH SẢN PHẨM

\*/

void timSanPham() {

List<SanPham> list;

// get tên nhóm hàng và mã nhóm hàng

String TuyChonNhomHang = cmb\_NhomHang.getSelectedItem().toString();

// get chuỗi tìm kiếm

String ChuoiTimKiem = txt\_TimKiemSanPham.getText();

// nếu chuỗi rỗng thì đổ tất cả sản phẩm theo mã nhóm

if (ChuoiTimKiem.isEmpty()) {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.select();

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timTheoMaNhom(MaNhom);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

} else {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.timSanPhamTheoTen(ChuoiTimKiem);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timKetHopNhomHang(ChuoiTimKiem, MaNhom);

if (list != null) {

doDuLieuSanPham(list);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

}

}

/\*

\* Hàm thao tác đổ dữ liệu lên bảng Sản Phẩm

\*/

void doDuLieuSanPham(List<SanPham> list) {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachSanPham.getModel();

dtm.setRowCount(0);

int STT = 0;

for (SanPham sp : list) {

STT++;

Object[] row = {

STT,

sp.getMaSanPham(),

sp.getTenSanPham(),

sp.getTonKho(),

dcf.format(sp.getGiaVon()),

dcf.format(sp.getGiaBan())

};

dtm.addRow(row);

}

}

/\*

\* Hàm thêm sản phẩm ở tbl\_DSSP vào phiếu kiểm ở tbl\_ChiTietPhieuKiem

\*/

void themSanPham() {

int index = tbl\_DanhSachSanPham.getSelectedRow();

String MaSP = tbl\_DanhSachSanPham.getValueAt(index, 1).toString();

try {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

// check trong TreeMap có SP chưa, nếu có thì báo lỗi, chưa thì add vào

if (DSSP.containsKey(SP))

DialogHelper.alert(this, "Sản phẩm đã có trong danh sách kiểm kê");

else {

DSSP.put(SP, -1);

capnhatDanhSachKiemKe();

tab\_ChiTietPhieu.setSelectedIndex(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm cập nhật danh sách kiểm kê

\*/

void capnhatDanhSachKiemKe() {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietPhieuKiem.getModel();

dtm.setRowCount(0);

int STT = 1;

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

Object[] row = {STT++,

SP.getMaSanPham(),

SP.getTenSanPham(),

DVT.getMoTa(),

dcf.format(SP.getGiaVon()),

SP.getTonKho(),

DSSP.get(SP),

DSSP.get(SP) > -1 ? (DSSP.get(SP) - SP.getTonKho()) : -1,

DSSP.get(SP) > -1 ? dcf.format((DSSP.get(SP) - SP.getTonKho())\*SP.getGiaVon()) : -1

};

dtm.addRow(row);

}

}

/\*

\* Hàm nhập số lượng thực tế kiểm kê được vào bảng

sau khi nhập thì cập nhật lại bảng

\*/

void nhapSoLuongThucTe() {

try {

int index = tbl\_ChiTietPhieuKiem.getEditingRow();

if (index >= 0) {

String MaSP = tbl\_ChiTietPhieuKiem.getValueAt(index, 1).toString();

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

int SLThucTe = Integer.parseInt(tbl\_ChiTietPhieuKiem.getValueAt(index, 6)+"");

DSSP.put(SP, SLThucTe);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi nhập liệu");

}

finally {

capnhatDanhSachKiemKe();

tinhTongGiaTriLech();

}

}

/\*

\* Hàm cập nhật tổng giá trị lệch

\*/

void tinhTongGiaTriLech() {

float TongGiaTriLech = 0;

for (SanPham SP : DSSP.keySet()) {

if (DSSP.get(SP) > -1)

TongGiaTriLech += (DSSP.get(SP) - SP.getTonKho())\*SP.getGiaVon();

}

txt\_GiaTriLech.setText(TongGiaTriLech+"");

}

/\*

\* Hàm xoá hết sản phẩm trong danh sách kiểm kê

\*/

void xoaDanhSachKiemKe() {

DSSP.clear();

capnhatDanhSachKiemKe();

tinhTongGiaTriLech();

}

/\*

\* Hàm check ghi chú: không được để trống, tối thiểu phải đủ 3 ký tự

\*/

boolean kiemGhiChu() {

if(txa\_GhiChu.getText().length() < 3)

return false;

return true;

}

/\*

\* Hàm kiểm tra dssp có null hay không, null thì không được phép lưu tạm/cân bằng

\*/

boolean kiemDSSP() {

if (DSSP.isEmpty())

return false;

return true;

}

/\*

\* Hàm lưu tạm phiếu kiểm

không thực hiện sửa tồn kho

\*/

void luuTam() {

if (xetDangNhap() && kiemGhiChu() && kiemDSSP()) {

PhieuKiem PK = taoPhieuKiem();

try {

int rs = PhieuKiemDAO.themPhieuTam(PK);

themPKCT(PK);

if (rs > 0) {

DialogHelper.alert(this, "Thêm thành công");

if (capnhatFrameKiemKeSP()) {

this.dispose();

new KiemKeSanPhamJFrame().setVisible(true);

}

}

else

DialogHelper.alert(this, "Thêm thất bại");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm lưu phiếu kiểm và cân bằng kho

nếu SL thực < 0 thì không cho cân bằng

\*/

void themPhieuCanBang() {

if (xetDangNhap() && kiemGhiChu() && kiemDSSP() && kiemSoLuongThucTe()) {

PhieuKiem PK = taoPhieuKiem();

PK.setNgayCanBang(DateHelper.now());

try {

int rs = PhieuKiemDAO.themPhieuCanBang(PK);

if (rs > 0) {

themPKCT(PK);

canbangTonKho();

DialogHelper.alert(this, "Đã thêm phiếu kiểm và cân bằng kho");

xoaDanhSachKiemKe();

if (capnhatFrameKiemKeSP()) {

this.dispose();

new KiemKeSanPhamJFrame().setVisible(true);

}

}

else

DialogHelper.alert(this, "Cân bằng thất bại");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm kiểm tra số lượng thực tế của SP trước khi cân bằng

nếu có SP có SL '-1' (tức chưa kiểm kê) thì trả về false

\*/

boolean kiemSoLuongThucTe() {

for (SanPham SP : DSSP.keySet()) {

if (DSSP.get(SP) == -1)

return false;

}

return true;

}

/\*

\* Hàm cập nhật lại tồn kho của SP (cân bằng kho hàng)

\*/

void canbangTonKho() {

for (SanPham SP : DSSP.keySet()) {

SanPhamDAO.capnhatTonKho(SP.getMaSanPham(), DSSP.get(SP));

}

}

/\*

\* Hàm thêm PKCT

\*/

void themPKCT(PhieuKiem PK) {

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

PhieuKiemChiTiet PKCT = new PhieuKiemChiTiet();

PKCT.setMaPhieuKiem(PK.getMaPhieuKiem());

PKCT.setMaSanPham(SP.getMaSanPham());

PKCT.setDonGia(SP.getGiaVon());

PKCT.setDVT(DVT.getMoTa());

PKCT.setTonKho(SP.getTonKho());

PKCT.setThucTe(DSSP.get(SP));

if (DSSP.get(SP) > -1) {

PKCT.setSoLuongLech(DSSP.get(SP) - SP.getTonKho());

PKCT.setGiaTriLech((DSSP.get(SP) - SP.getTonKho()) \* SP.getGiaVon());

} else {

PKCT.setSoLuongLech(-1);

PKCT.setGiaTriLech(-1);

}

PKCTDAO.themPKCT(PKCT);

}

}

/\*

\* Hàm kiểm tra đăng nhập

\*/

boolean xetDangNhap() {

if (ShareHelper.NguoiDung != null)

return true;

return false;

}

/\*

\* Hàm tạo model Phiếu Kiểm Kê

\*/

PhieuKiem taoPhieuKiem() {

PhieuKiem PK = new PhieuKiem();

PK.setMaPhieuKiem(txt\_MaChungTu.getText().trim());

PK.setMaNhanVien(txt\_MaNhanVien.getText().trim());

PK.setNgayTao(dat\_NgayTao.getDate());

PK.setGiaTriLech(Double.parseDouble(txt\_GiaTriLech.getText()));

PK.setGhiChu(txa\_GhiChu.getText());

PK.setDSSP(DSSP);

return PK;

}

/\*

\* Hàm thoát frame thêm mới và cập nhật lại frame kiểm kê sản phẩm

\*/

boolean capnhatFrameKiemKeSP(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.sanpham.kiemke.KiemKeSanPhamJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

/\*

\* Hàm xoá sản phẩm trong danh sách kiểm kê khi click chọn

\*/

void xoaSanPham() {

int index = tbl\_ChiTietPhieuKiem.getSelectedRow();

String MaSP = tbl\_ChiTietPhieuKiem.getValueAt(index, 1).toString();

try {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

if (DialogHelper.confirm(this, "Xoá sản phẩm " + MaSP + "khỏi danh sách kiểm kê?")) {

DSSP.remove(SP);

capnhatDanhSachKiemKe();

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **CapNhatPhieuKiemJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuKiemDAO PhieuKiemDAO = new PhieuKiemDAO();

PhieuKiemChiTietDAO PKCTDAO = new PhieuKiemChiTietDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// Tạo treemap chứa sản phẩm và số lượng kiểm đếm thực tế

static TreeMap<SanPham, Integer> DSSP = new TreeMap<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

tab\_ChiTietPhieu.setEnabledAt(1, false);

xetTrangThai();

doThongTin();

}

/\*

\* Hàm xét trạng thái của phiếu kiểm:

- nếu là phiếu TẠM (0): cho phép cập nhật số lượng thực tế

- nếu là phiếu ĐÃ CÂN BẰNG (1): chỉ được xem, disable hết mọi chức năng

- nếu là phiếu HUỶ (2): giống phiếu ĐÃ CÂN BẰNG

\*/

void xetTrangThai() {

btn\_XoaDanhSach.setEnabled(false);

int TrangThai = PK.getTrangThai();

if (TrangThai == 1 || TrangThai == 2) {

btn\_CanBang.setEnabled(false);

btn\_LuuTam.setEnabled(false);

btn\_CapNhatDanhSach.setEnabled(false);

txa\_GhiChu.setEnabled(false);

tbl\_ChiTietPhieuKiem.setEnabled(false);

}

}

/\*

\* Hàm đổ các thông tin mặc định

\*/

void doThongTin() {

try {

// đổ mã nhân viên

txt\_MaNhanVien.setText(PK.getMaNhanVien());

// mã phiếu kiểm

txt\_MaChungTu.setText(PK.getMaPhieuKiem());

// ngày tạo phiếu

dat\_NgayTao.setDate(PK.getNgayTao());

// ngày cân bằng

if (PK.getTrangThai() == 1) {

dat\_NgayCanBang.setDate(PK.getNgayCanBang());

}

// giá trị lệch

txt\_GiaTriLech.setText(PK.getGiaTriLech()+"");

// ghi chú

txa\_GhiChu.setText(PK.getGhiChu());

// đổ thông tin dssp vào treemap

List<PhieuKiemChiTiet> listPKCT = PKCTDAO.timDanhSachPKCT(PK.getMaPhieuKiem());

for (PhieuKiemChiTiet PKCT : listPKCT) {

SanPham SP = SanPhamDAO.timTheoMaSP(PKCT.getMaSanPham());

DSSP.put(SP, PKCT.getThucTe());

}

capnhatDanhSachKiemKe();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm cập nhật danh sách kiểm kê

\*/

void capnhatDanhSachKiemKe() {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietPhieuKiem.getModel();

dtm.setRowCount(0);

int STT = 1;

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

Object[] row = {STT++,

SP.getMaSanPham(),

SP.getTenSanPham(),

DVT.getMoTa(),

dcf.format(SP.getGiaVon()),

SP.getTonKho(),

DSSP.get(SP),

DSSP.get(SP) > -1 ? (DSSP.get(SP) - SP.getTonKho()) : -1,

DSSP.get(SP) > -1 ? dcf.format((DSSP.get(SP) - SP.getTonKho())\*SP.getGiaVon()) : -1

};

dtm.addRow(row);

}

}

/\*

\* Hàm nhập số lượng thực tế kiểm kê được vào bảng

sau khi nhập thì cập nhật lại bảng

\*/

void nhapSoLuongThucTe() {

try {

int index = tbl\_ChiTietPhieuKiem.getEditingRow();

if (index >= 0) {

String MaSP = tbl\_ChiTietPhieuKiem.getValueAt(index, 1).toString();

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

int SLThucTe = Integer.parseInt(tbl\_ChiTietPhieuKiem.getValueAt(index, 6)+"");

DSSP.put(SP, SLThucTe);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi nhập liệu");

}

finally {

capnhatDanhSachKiemKe();

tinhTongGiaTriLech();

}

}

/\*

\* Hàm cập nhật tổng giá trị lệch

\*/

void tinhTongGiaTriLech() {

float TongGiaTriLech = 0;

for (SanPham SP : DSSP.keySet()) {

if (DSSP.get(SP) > -1)

TongGiaTriLech += (DSSP.get(SP) - SP.getTonKho())\*SP.getGiaVon();

}

txt\_GiaTriLech.setText(TongGiaTriLech+"");

}

/\*

\* Hàm xoá hết sản phẩm trong danh sách kiểm kê

\*/

void xoaDanhSachKiemKe() {

DSSP.clear();

capnhatDanhSachKiemKe();

tinhTongGiaTriLech();

}

/\*

\* Hàm check ghi chú: không được để trống, tối thiểu phải đủ 3 ký tự

\*/

boolean kiemGhiChu() {

if(txa\_GhiChu.getText().length() < 3)

return false;

return true;

}

/\*

\* Hàm kiểm tra dssp có null hay không, null thì không được phép lưu tạm/cân bằng

\*/

boolean kiemDSSP() {

if (DSSP.isEmpty())

return false;

return true;

}

/\*

\* Hàm lưu tạm phiếu kiểm

không thực hiện sửa tồn kho

\*/

void luuTam() {

if (xetDangNhap() && kiemGhiChu() && kiemDSSP()) {

PhieuKiem PK = taoPhieuKiem();

PK.setTrangThai(0);

try {

int rs = PhieuKiemDAO.capnhatPhieuKiem(PK);

capnhatPKCT(PK);

if (rs > 0) {

DialogHelper.alert(this, "Cập nhật thành công");

if (capnhatFrameKiemKeSP()) {

this.dispose();

new KiemKeSanPhamJFrame().setVisible(true);

}

}

else

DialogHelper.alert(this, "Cập nhật thất bại");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm lưu phiếu kiểm và cân bằng kho

nếu SL thực < 0 thì không cho cân bằng

\*/

void capnhatPhieuCanBang() {

if (xetDangNhap() && kiemGhiChu() && kiemDSSP() && kiemSoLuongThucTe()) {

PhieuKiem PK = taoPhieuKiem();

PK.setTrangThai(1);

PK.setNgayCanBang(DateHelper.now());

try {

int rs = PhieuKiemDAO.capnhatPhieuKiem(PK);

if (rs > 0) {

capnhatPKCT(PK);

canbangTonKho();

DialogHelper.alert(this, "Đã thêm phiếu kiểm và cân bằng kho");

xoaDanhSachKiemKe();

if (capnhatFrameKiemKeSP()) {

this.dispose();

new KiemKeSanPhamJFrame().setVisible(true);

}

}

else

DialogHelper.alert(this, "Cân bằng thất bại");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm kiểm tra số lượng thực tế của SP trước khi cân bằng

nếu có SP có SL '-1' (tức chưa kiểm kê) thì trả về false

\*/

boolean kiemSoLuongThucTe() {

for (SanPham SP : DSSP.keySet()) {

if (DSSP.get(SP) == -1)

return false;

}

return true;

}

/\*

\* Hàm cập nhật lại tồn kho của SP (cân bằng kho hàng)

\*/

void canbangTonKho() {

for (SanPham SP : DSSP.keySet()) {

SanPhamDAO.capnhatTonKho(SP.getMaSanPham(), DSSP.get(SP));

}

}

/\*

\* Hàm cập nhật PKCT

\*/

void capnhatPKCT(PhieuKiem PK) {

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

PhieuKiemChiTiet PKCT = new PhieuKiemChiTiet();

PKCT.setMaPhieuKiem(PK.getMaPhieuKiem());

PKCT.setMaSanPham(SP.getMaSanPham());

PKCT.setDonGia(SP.getGiaVon());

PKCT.setDVT(DVT.getMoTa());

PKCT.setTonKho(SP.getTonKho());

PKCT.setThucTe(DSSP.get(SP));

if (DSSP.get(SP) > -1) {

PKCT.setSoLuongLech(DSSP.get(SP) - SP.getTonKho());

PKCT.setGiaTriLech((DSSP.get(SP) - SP.getTonKho()) \* SP.getGiaVon());

} else {

PKCT.setSoLuongLech(-1);

PKCT.setGiaTriLech(-1);

}

PKCTDAO.themPKCT(PKCT);

}

}

/\*

\* Hàm kiểm tra đăng nhập

\*/

boolean xetDangNhap() {

if (ShareHelper.NguoiDung != null)

return true;

return false;

}

/\*

\* Hàm tạo model Phiếu Kiểm Kê

\*/

PhieuKiem taoPhieuKiem() {

PhieuKiem PK = new PhieuKiem();

PK.setMaPhieuKiem(txt\_MaChungTu.getText().trim());

PK.setMaNhanVien(txt\_MaNhanVien.getText().trim());

PK.setNgayTao(dat\_NgayTao.getDate());

PK.setGiaTriLech(Double.parseDouble(txt\_GiaTriLech.getText()));

PK.setGhiChu(txa\_GhiChu.getText());

PK.setDSSP(DSSP);

return PK;

}

/\*

\* Hàm thoát frame thêm mới và cập nhật lại frame kiểm kê sản phẩm

\*/

boolean capnhatFrameKiemKeSP(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.sanpham.kiemke.KiemKeSanPhamJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

}

* + - 1. **CanhBaoHSDJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// list chua maSP duoc chon de ngung kinh doanh hoac xuat huy

static List<String> listTam = new ArrayList<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

btn\_NgungBan.setVisible(false);

btn\_XuatHuy.setVisible(false);

doNhomHang();

capnhatDuLieu();

}

/\*

\* Hàm này đổ danh sách nhóm hàng vào combobox ở tab DanhSachSanPham

\*/

void doNhomHang() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

try {

List<NhomHang> ListNH = NhomHangDAO.select();

dcm.addElement("Tất cả");

for (NhomHang NH : ListNH) {

dcm.addElement(NH);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm cập nhật dữ liệu Sản Phẩm

\*/

void capnhatDuLieu(){

try {

List<SanPham> listSP = SanPhamDAO.select();

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<SanPham> listSP) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachSanPham.getModel();

dtm.setRowCount(0);

for(SanPham SP : listSP){

String HSDTruoc = "";

String HSDMoiNhat = "";

if (SP.getHSDTruoc() != null) {

HSDTruoc = DateHelper.toString(SP.getHSDTruoc(), "dd-MM-yyyy");

}

if (SP.getHSDMoiNhat() != null) {

HSDMoiNhat = DateHelper.toString(SP.getHSDMoiNhat(), "dd-MM-yyyy");

}

Object[] row = {false,

SP.getMaSanPham(),

SP.getTenSanPham(),

HSDTruoc,

HSDMoiNhat,

SP.getTonKho(),

SP.getTrangThai() ? "Đang bán" : "Ngưng bán"

};

dtm.addRow(row);

}

}

/\*

\* Hàm lọc sản phẩm theo các tuỳ chỉnh

\*/

void locSanPham() {

String MaNH;

// lấy mã nhóm hàng:

if (cmb\_NhomHang.getSelectedIndex() != 0)

MaNH = ((NhomHang) cmb\_NhomHang.getSelectedItem()).getMaNhomHang();

else

MaNH = "0";

// lấy trạng thái:

boolean TrangThai = !chk\_SPNgungBan.isSelected();

// lấy tuỳ chọn ngày hết hạn:

int chonNgayHetHan = cmb\_NgayHetHan.getSelectedIndex();

// 0: 3 ngày || 1: 7 ngày || 2: tháng này || 3: tháng tới || 4: đã hết hạn

try {

List<SanPham> listSP;

switch (chonNgayHetHan) {

case 0:

listSP = SanPhamDAO.locHSD3NgayToi(MaNH, TrangThai);

break;

case 1:

listSP = SanPhamDAO.locHSD7NgayToi(MaNH, TrangThai);

break;

case 2:

listSP = SanPhamDAO.locHSDThangNay(MaNH, TrangThai);

break;

case 3:

listSP = SanPhamDAO.locHSDThangToi(MaNH, TrangThai);

break;

case 4:

listSP = SanPhamDAO.locHetHSD(MaNH, TrangThai);

break;

default:

listSP = SanPhamDAO.select();

}

doDuLieuVaoBang(listSP);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tạo danh sách các sản phẩm đang được chọn trên table

\*/

void taoDanhSachDangChon() {

int i = tbl\_DanhSachSanPham.getSelectedRow();

if(i >= 0){

String MaSP = tbl\_DanhSachSanPham.getValueAt(i, 1).toString();

if((Boolean)tbl\_DanhSachSanPham.getValueAt(i, 0)){

if(!(listTam.indexOf(MaSP) >= 0)) {

listTam.add(MaSP);

}

}

else {

if((listTam.indexOf(MaSP) >= 0)){

listTam.remove(MaSP);

}

}

}

if (!listTam.isEmpty()) {

btn\_NgungBan.setVisible(true);

btn\_XuatHuy.setVisible(true);

}

else {

btn\_NgungBan.setVisible(false);

btn\_XuatHuy.setVisible(false);

}

}

/\*

\* Hàm cập nhật trạng thái ngưng kinh doanh cho SP

\*/

void ngungKinhDoanh() {

if(DialogHelper.confirm(this, "Chuyển trạng thái ngưng kinh doanh cho các sản phẩm này?")) {

try {

int i = 0;

for(String MaSP : listTam) {

i = SanPhamDAO.ngungKinhDoanh(MaSP);

}

if(i > 0) {

DialogHelper.alert(this, "Cập nhật thành công");

listTam.clear();

btn\_NgungBan.setVisible(false);

btn\_XuatHuy.setVisible(false);

capnhatDuLieu();

}

else {

DialogHelper.alert(this, "Cập nhật thất bại");

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm đổ DSSP vào list đề nghị huỷ và mở frame thêm mới phiếu huỷ

\*/

void huySanPham() {

try {

for (String MaSP : listTam) {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

ShareHelper.listDeNghiHuy.add(SP);

}

new ThemMoiPhieuHuyJFrame().setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **DanhSachHoaDonJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

HoaDonDAO HoaDonDAO = new HoaDonDAO();

HoaDonChiTietDAO HDCTDAO = new HoaDonChiTietDAO();

KhachHangDAO KhachHangDAO = new KhachHangDAO();

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

}

/\*

\* Hàm cập nhật dữ liệu Sản Phẩm

\*/

void capnhatDuLieu(){

try {

rad\_TatCa.setSelected(true);

List<HoaDon> list = HoaDonDAO.select();

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<HoaDon> listHD) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachHoaDon.getModel();

dtm.setRowCount(0);

for(HoaDon HD : listHD){

String MaKH = HD.getMaKhachHang();

KhachHang KH = KhachHangDAO.timTheoMa(MaKH);

Object[] row = {HD.getMaHoaDon(),

KH.getHoTen(),

HD.getMaNhanVien(),

HD.getNgayBan(),

HD.getTongSoLuong(),

dcf.format(HD.getThanhTien())

};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm hoá đơn theo mã

\*/

void timTheoMa() {

String ChuoiTimKiem = txt\_TimTheoMaHoaDon.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

HoaDon HD = HoaDonDAO.timHoaDonTheoMa(ChuoiTimKiem);

List<HoaDon> list = new ArrayList<>();

if (HD != null)

list.add(HD);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachHoaDon.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm hoá đơn theo tên khách hàng

\*/

void timTheoTenKhachHang() {

String ChuoiTimKiem = txt\_TimTheoKhachHang.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

List<HoaDon> list = HoaDonDAO.timHoaDonTheoTenKH(ChuoiTimKiem);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachHoaDon.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm hoá đơn theo tên mã người bán

\*/

void timTheoMaNguoiBan() {

String ChuoiTimKiem = txt\_TimTheoNguoiBan.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

List<HoaDon> list = HoaDonDAO.timHoaDonTheoMaNguoiBan(ChuoiTimKiem);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachHoaDon.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm lọc hoá đơn theo các tuỳ chọn

\*/

void locHoaDon() {

List<HoaDon> list = new ArrayList<>();

if (rad\_TatCa.isSelected())

list = HoaDonDAO.select();

if (rad\_HomNay.isSelected())

list = HoaDonDAO.timTrongNgay();

if (rad\_HomQua.isSelected())

list = HoaDonDAO.timTrongHomQua();

if (rad\_7NgayTruoc.isSelected())

list = HoaDonDAO.timTrong7NgayTruoc();

if (rad\_Khac.isSelected())

list = timTheoKhoangNgay();

try {

if (list.size() > 0) {

doDuLieuVaoBang(list);

}

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachHoaDon.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm hoá đơn theo khoảng ngày tuỳ chọn

\*/

List<HoaDon> timTheoKhoangNgay() {

List<HoaDon> list = new ArrayList<>();

String TuNgay = DateHelper.toString(dat\_TuNgay.getDate(),"yyyy-MM-dd");

String DenNgay = DateHelper.toString(dat\_DenNgay.getDate(),"yyyy-MM-dd");

if (dat\_TuNgay.getDate().after(dat\_DenNgay.getDate())) {

DialogHelper.alert(this, "Chọn lại ngày");

}

else {

try {

list = HoaDonDAO.timTheoKhoangNgay(TuNgay, DenNgay);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

return list;

}

/\*

\* Hàm mở jframe chi tiêt hoá đơn

\*/

void xemHDCT() {

int index = tbl\_DanhSachHoaDon.getSelectedRow();

String MaHD = tbl\_DanhSachHoaDon.getValueAt(index, 0).toString();

try {

HoaDon HD = HoaDonDAO.timHoaDonTheoMa(MaHD);

new ChiTietHoaDonJFrame(HD).setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **ChiTietHoaDonJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

HoaDonDAO HoaDonDAO = new HoaDonDAO();

HoaDonChiTietDAO HDCTDAO = new HoaDonChiTietDAO();

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

KhachHangDAO KhachHangDAO = new KhachHangDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

doThongTin();

doDanhSachSP();

}

/\*

\* Hàm đổ các thông tin mặc định

\*/

void doThongTin() {

try {

// đổ mã hoá đơn

txt\_MaHoaDon.setText(HD.getMaHoaDon());

// khởi tạo tên khách hàng

KhachHang KH = KhachHangDAO.timTheoMa(HD.getMaKhachHang());

txt\_TenKhachHang.setText(KH.getHoTen());

// đổ mã nhân viên

txt\_NhanVien.setText(HD.getMaNhanVien());

// đổ ngày bán

txt\_NgayBan.setText(DateHelper.toString(HD.getNgayBan(), "dd-MM-yyyy"));

// đổ tổng số lượng

txt\_TongSoLuong.setText(HD.getTongSoLuong()+"");

// đổ thành tiền

float ThanhTien = HoaDonDAO.tinhThanhTien(HD.getMaHoaDon());

txt\_ThanhTien.setText(dcf.format(ThanhTien));

// đổ chiết khấu

txt\_ChietKhau.setText(dcf.format(HD.getChietKhau()));

// VAT

txt\_VAT.setText(HD.getVAT()+"");

// tổng cộng

txt\_TongCong.setText(dcf.format(HD.getThanhTien()));

// khách trả

txt\_TienKhachTra.setText(dcf.format(HD.getTienKhachTra()));

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ danh sách sản phẩm trong hoá đơn lên table

\*/

void doDanhSachSP() {

try {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachSanPham.getModel();

dtm.setRowCount(0);

List<HoaDonChiTiet> listHDCT = HDCTDAO.timDanhSachHDCT(HD.getMaHoaDon());

int STT = 1;

for (HoaDonChiTiet HDCT : listHDCT) {

SanPham SP = SanPhamDAO.timTheoMaSP(HDCT.getMaSanPham());

Object[] row = {STT++,

HDCT.getMaSanPham(),

SP.getTenSanPham(),

dcf.format(HDCT.getDonGia()),

HDCT.getSoLuong(),

dcf.format(HDCT.getThanhTien())};

dtm.addRow(row);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm kết nối máy in và in hoá đơn

\*/

void inHoaDon() {

try {

HashMap HoaDon = new HashMap();

HoaDon.put("MaInHoaDon", HD.getMaHoaDon());

jdbcHelper.inHoaDon(HoaDon);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **DanhSachPhieuNhapJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuNhapDAO PhieuNhapDAO = new PhieuNhapDAO();

PhieuNhapChiTietDAO PNCTDAO = new PhieuNhapChiTietDAO();

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

}

/\*

\* Hàm cập nhật dữ liệu phiếu nhập

\*/

void capnhatDuLieu(){

try {

rad\_TatCa.setSelected(true);

List<PhieuNhap> list = PhieuNhapDAO.select();

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<PhieuNhap> listPN) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuNhap.getModel();

dtm.setRowCount(0);

for(PhieuNhap PN : listPN){

NhaCungCap NCC = NCCDAO.timTheoMa(PN.getMaNCC());

Object[] row = {PN.getMaPhieuNhap(),

NCC.getTenNCC(),

PN.getMaNhanVien(),

DateHelper.toString(PN.getNgayNhap(), "dd-MM-yyyy"),

PN.getTongSoLuong(),

PN.getChietKhau(),

PN.getVAT(),

dcf.format(PN.getThanhTien())

};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm phiếu nhập theo mã

\*/

void timTheoMa() {

String ChuoiTimKiem = txt\_TimTheoMaChungTu.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

PhieuNhap PN = PhieuNhapDAO.timTheoMa(ChuoiTimKiem);

List<PhieuNhap> list = new ArrayList<>();

if (PN != null)

list.add(PN);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuNhap.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm phiếu nhập theo mã/tên sản phẩm

\*/

void timTheoSanPham() {

String ChuoiTimKiem = txt\_TimTheoSanPham.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

List<PhieuNhap> list = PhieuNhapDAO.timTheoSanPham(ChuoiTimKiem);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuNhap.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm phiếu nhập theo mã NCC

\*/

void timTheoMaNCC() {

String ChuoiTimKiem = txt\_TimTheoMaNCC.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

List<PhieuNhap> list = PhieuNhapDAO.timTheoMaNCC(ChuoiTimKiem);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuNhap.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm lọc phiếu nhập theo các tuỳ chọn

\*/

void locPhieuNhap() {

List<PhieuNhap> list = new ArrayList<>();

if (rad\_TatCa.isSelected())

list = PhieuNhapDAO.select();

if (rad\_HomNay.isSelected())

list = PhieuNhapDAO.timTrongNgay();

if (rad\_HomQua.isSelected())

list = PhieuNhapDAO.timNgayHomQua();

if (rad\_7NgayTruoc.isSelected())

list = PhieuNhapDAO.tim7NgayTruoc();

if (rad\_Khac.isSelected())

list = timTheoKhoangNgay();

try {

if (list.size() > 0) {

doDuLieuVaoBang(list);

}

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuNhap.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu nhập theo khoảng ngày tuỳ chọn

\*/

List<PhieuNhap> timTheoKhoangNgay() {

List<PhieuNhap> list = new ArrayList<>();

String TuNgay = DateHelper.toString(dat\_TuNgay.getDate(),"yyyy-MM-dd");

String DenNgay = DateHelper.toString(dat\_DenNgay.getDate(),"yyyy-MM-dd");

if (dat\_TuNgay.getDate().after(dat\_DenNgay.getDate())) {

DialogHelper.alert(this, "Chọn lại ngày");

}

else {

try {

list = PhieuNhapDAO.timTheoKhoangNgay(TuNgay, DenNgay);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

return list;

}

}

* + - 1. **ThemMoiPhieuNhapJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuNhapDAO PhieuNhapDAO = new PhieuNhapDAO();

PhieuNhapChiTietDAO PNCTDAO = new PhieuNhapChiTietDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

NguoiDungDAO NguoiDungDAO = new NguoiDungDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// Tạo treemap chứa sản phẩm, số lượng và giá nhập

static TreeMap<SanPham, List<String>> DSSP = new TreeMap<>();

// TreeMap chứa sản phẩm và vốn sau điều chỉnh

static TreeMap<SanPham, Float> DSDieuChinhVon = new TreeMap<>();

// TreeMap chứa sản phẩm và HSD Mới

static TreeMap<SanPham, String> DSDieuChinhHSDMoi = new TreeMap<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

doNhomHang();

doDanhSachSanPham();

doThongTin();

}

/\*

\* Hàm đổ các thông tin mặc định

\*/

void doThongTin() {

try {

// mã nhân viên

txt\_MaNhanVien.setText(ShareHelper.NguoiDung.getTenDangNhap());

// mã phiếu nhập

int demPhieuNhap = PhieuNhapDAO.demPhieuNhap();

String MaPN = "PN" + (demPhieuNhap + 1);

txt\_MaChungTu.setText(MaPN);

// ngay nhap

dat\_NgayNhap.setDate(DateHelper.now());

// NCC: chi hien thi nhung NCC co trang thai dang giao dich

doNhaCungCap();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ danh sách các NCC vào combobox (trạng thái đang giao dịch)

\*/

void doNhaCungCap() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhaCungCap.getModel();

dcm.removeAllElements();

try {

List<NhaCungCap> listNCC = NCCDAO.locTheoTrangThai(1);

for (NhaCungCap NCC : listNCC) {

dcm.addElement(NCC);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách nhóm hàng vào combobox ở tab DanhSachSanPham

\*/

void doNhomHang() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

try {

List<NhomHang> ListNH = NhomHangDAO.select();

dcm.addElement("Tất cả");

for (NhomHang NH : ListNH) {

dcm.addElement(NH);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách sản phẩm vào tab Danh Sách Sản Phẩm

\*/

void doDanhSachSanPham() {

try {

List<SanPham> ListSP = SanPhamDAO.select();

doDuLieuSanPham(ListSP);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này tìm sản phẩm ở tab DANH SÁCH SẢN PHẨM

\*/

void timSanPham() {

List<SanPham> list;

// get tên nhóm hàng và mã nhóm hàng

String TuyChonNhomHang = cmb\_NhomHang.getSelectedItem().toString();

// get chuỗi tìm kiếm

String ChuoiTimKiem = txt\_TimKiemSanPham.getText();

// nếu chuỗi rỗng thì đổ tất cả sản phẩm theo mã nhóm

if (ChuoiTimKiem.isEmpty()) {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.select();

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timTheoMaNhom(MaNhom);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

} else {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.timSanPhamTheoTen(ChuoiTimKiem);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timKetHopNhomHang(ChuoiTimKiem, MaNhom);

if (list != null) {

doDuLieuSanPham(list);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

}

}

/\*

\* Hàm thao tác đổ dữ liệu lên bảng Sản Phẩm

\*/

void doDuLieuSanPham(List<SanPham> list) {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachSanPham.getModel();

dtm.setRowCount(0);

int STT = 0;

for (SanPham sp : list) {

STT++;

NhaCungCap NCC = NCCDAO.timTheoMa(sp.getMaNCC());

Object[] row = {

STT,

sp.getMaSanPham(),

sp.getTenSanPham(),

NCC.getTenNCC(),

sp.getTonKho(),

dcf.format(sp.getGiaVon()),

dcf.format(sp.getGiaBan())

};

dtm.addRow(row);

}

}

/\*

\* Hàm thêm sản phẩm ở tbl\_DSSP vào tab chi tiết phiếu nhập

\*/

void themSanPham() {

int index = tbl\_DanhSachSanPham.getSelectedRow();

String MaSP = tbl\_DanhSachSanPham.getValueAt(index, 1).toString();

try {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

// check trong TreeMap có SP chưa, nếu có thì hiện thông báo, chưa thì add vào

if (DSSP.containsKey(SP))

DialogHelper.alert(this, "Sản phẩm đã có trong danh sách nhập");

else {

List<String> list = new ArrayList<>();

list.add("0");

list.add("0.0");

DSSP.put(SP, list);

capnhatDanhSachNhap();

tab\_ChiTietPhieu.setSelectedIndex(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm cập nhật danh sách hàng nhập

\*/

void capnhatDanhSachNhap() {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietPhieuNhap.getModel();

dtm.setRowCount(0);

DSDieuChinhVon.clear();

int STT = 1;

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

int SoLuong = Integer.parseInt(DSSP.get(SP).get(0));

float GiaNhap = Float.parseFloat(DSSP.get(SP).get(1));

String HSDMoiNhat = "";

// cập nhật danh sách SP cần điều chỉnh HSD

if (DSSP.get(SP).size() == 3) {

HSDMoiNhat = DSSP.get(SP).get(2);

if (SP.getHSDMoiNhat() != null || SP.getHSDMoiNhat().before(DateHelper.toDate(HSDMoiNhat, "yyyy-MM-dd"))) {

DSDieuChinhHSDMoi.put(SP, HSDMoiNhat);

}

}

float VonSauDieuChinh = 0;

/\*

\* Tính giá vốn sau điều chỉnh: MAC = (A+B)/C

\* MAC: giá vốn sản phẩm theo bình quân tức thời (giá vốn hiện tại)

\* A: Giá trị kho trước nhập = Tồn kho trước nhập \* MAC trước nhập

\* B: Giá trị kho nhập mới = Tồn nhập mới \* giá nhập mới

\* C: tổng tồn = tồn trước nhập + tồn sau nhập

\*/

if (GiaNhap > 0 && SoLuong > 0) {

float A = SP.getTonKho() \* SP.getGiaVon();

float B = GiaNhap \* SoLuong;

int C = SP.getTonKho() + SoLuong;

VonSauDieuChinh = (A+B)/C;

// cập nhật danh sách sản phẩm cần điều chỉnh giá vốn

DSDieuChinhVon.put(SP, VonSauDieuChinh);

}

Object[] row = {STT++,

SP.getMaSanPham(),

SP.getTenSanPham(),

DVT.getMoTa(),

HSDMoiNhat,

SoLuong,

GiaNhap,

dcf.format(SoLuong\*GiaNhap),

dcf.format(SP.getGiaVon()),

dcf.format(VonSauDieuChinh)};

dtm.addRow(row);

}

}

/\*

\* Hàm nhập số lượng cần nhập và giá nhập vào bảng

sau khi nhập thì cập nhật lại bảng

\*/

void nhapThongTinNhapHang() {

try {

int index = tbl\_ChiTietPhieuNhap.getEditingRow();

if (index >= 0) {

String MaSP = tbl\_ChiTietPhieuNhap.getValueAt(index, 1).toString();

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

String SLNhap = tbl\_ChiTietPhieuNhap.getValueAt(index, 5).toString();

String GiaNhap = tbl\_ChiTietPhieuNhap.getValueAt(index, 6).toString();

String HSDMoi = "";

if (!tbl\_ChiTietPhieuNhap.getValueAt(index, 4).toString().isEmpty()) {

HSDMoi = tbl\_ChiTietPhieuNhap.getValueAt(index, 4).toString();

}

List<String> listThongTin = new ArrayList<>();

listThongTin.add(SLNhap);

listThongTin.add(GiaNhap);

if (!HSDMoi.isEmpty())

listThongTin.add(HSDMoi);

DSSP.put(SP, listThongTin);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi nhập liệu");

}

finally {

capnhatDanhSachNhap();

tinhTongSoLuongNhap();

tinhTongGiaTriNhap();

tinhTongTien();

}

}

/\*

\* Hàm cập nhật tổng số lượng nhập

\*/

void tinhTongSoLuongNhap() {

int TongSLNhap = 0;

for (SanPham SP : DSSP.keySet()) {

TongSLNhap += Integer.parseInt(DSSP.get(SP).get(0));

}

txt\_TongSoLuong.setText(TongSLNhap+"");

}

/\*

\* Hàm cập nhật tổng giá trị nhập

\*/

void tinhTongGiaTriNhap() {

float TongGTNhap = 0;

for (SanPham SP : DSSP.keySet()) {

int SL = Integer.parseInt(DSSP.get(SP).get(0));

float GiaNhap = Float.parseFloat(DSSP.get(SP).get(1));

TongGTNhap += SL \* GiaNhap;

}

txt\_ThanhTien.setText(TongGTNhap+"");

}

/\*

\* Hàm tính tổng tiền cuối cùng (bao gồm chiết khấu + VAT)

\*/

void tinhTongTien() {

float TienTamTinh = Float.parseFloat(txt\_ThanhTien.getText());

float ChietKhau = 0f;

if (!txt\_ChietKhau.getText().isEmpty() && !txt\_ChietKhau.getText().equals("0"))

ChietKhau = Float.parseFloat(txt\_ChietKhau.getText());

float VAT = (chk\_VAT.isSelected() ? 0.1f : 0);

float TongTien = 0f;

if (TienTamTinh > 0) {

// TongTien = (TienTamTinh - GiaTriChietKhau) + (TienTamTinh - GiaTriChietKhau)\*VAT

TongTien = (TienTamTinh - TienTamTinh\*ChietKhau/100) + (TienTamTinh - TienTamTinh\*ChietKhau/100)\*VAT;

}

txt\_TongCong.setText(TongTien+"");

}

/\*

\* Hàm xoá hết sản phẩm trong danh sách nhập

\*/

void xoaDanhSachNhap() {

DSSP.clear();

capnhatDanhSachNhap();

tinhTongGiaTriNhap();

tinhTongSoLuongNhap();

tinhTongTien();

}

/\*

\* Hàm check ghi chú: không được để trống, tối thiểu phải đủ 3 ký tự

\*/

boolean kiemGhiChu() {

return txa\_GhiChu.getText().length() >= 3;

}

/\*

\* Hàm kiểm tra dssp

- có null hay không, null thì không được phép hoàn tất

\*/

boolean kiemDSSP() {

return !DSSP.isEmpty();

}

/\*

\* Hàm kiểm tra số lượng hàng nhập và giá nhập

nếu có SP có SL = 0 hoặc giá nhập = 0 thì trả về false

\*/

boolean kiemSoLuongThucTe() {

for (SanPham SP : DSSP.keySet()) {

if (Integer.parseInt(DSSP.get(SP).get(0)) == 0

|| Float.parseFloat(DSSP.get(SP).get(1)) == 0)

return false;

}

return true;

}

/\*

\* Hàm lưu phiếu nhập, cân bằng kho và cập nhật lại giá vốn

\*/

void themPhieuNhap() {

if (xetDangNhap() && kiemGhiChu() && kiemDSSP() && kiemSoLuongThucTe()) {

PhieuNhap PN = taoPhieuNhap();

try {

int rs = PhieuNhapDAO.themPhieuNhap(PN);

if (rs > 0) {

themPNCT(PN);

canbangTonKho();

capnhatGiaVon();

capnhatHSDMoi();

DialogHelper.alert(this, "Đã hoàn tất quá trình nhập hàng");

xoaDanhSachNhap();

if (capnhatFrameDSPhieuNhap()) {

this.dispose();

new DanhSachPhieuNhapJFrame().setVisible(true);

}

else

this.dispose();

}

else

DialogHelper.alert(this, "Thêm mới thất bại");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm cập nhật lại tồn kho của SP (cân bằng kho hàng)

\*/

void canbangTonKho() throws SQLException {

for (SanPham SP : DSSP.keySet()) {

int TonKhoMoi = SP.getTonKho() + Integer.parseInt(DSSP.get(SP).get(0));

SanPhamDAO.capnhatTonKho(SP.getMaSanPham(), TonKhoMoi);

}

}

/\*

\* Hàm cập nhật lại giá vốn của SP

\*/

void capnhatGiaVon() throws SQLException {

for (SanPham SP : DSDieuChinhVon.keySet()) {

float GiaVonMoi = DSDieuChinhVon.get(SP);

SanPhamDAO.capnhatGiaVon(SP.getMaSanPham(), GiaVonMoi);

}

}

/\*

\* Hàm cập nhật lại HSD mới của SP

\*/

void capnhatHSDMoi() throws SQLException {

for (SanPham SP : DSDieuChinhHSDMoi.keySet()) {

String HSDMoi = DSDieuChinhHSDMoi.get(SP);

SanPhamDAO.capnhatHSDMoi(SP.getMaSanPham(), DateHelper.toDate(HSDMoi, "yyyy-MM-dd"));

}

}

/\*

\* Hàm thêm PNCT

\*/

void themPNCT(PhieuNhap PN) {

for (SanPham SP : DSSP.keySet()) {

PhieuNhapChiTiet PNCT = new PhieuNhapChiTiet();

PNCT.setMaPhieuNhap(PN.getMaPhieuNhap());

PNCT.setMaSanPham(SP.getMaSanPham());

PNCT.setSoLuong(Integer.parseInt(DSSP.get(SP).get(0)));

PNCT.setGiaNhap(Float.parseFloat(DSSP.get(SP).get(1)));

float ThanhTien = Integer.parseInt(DSSP.get(SP).get(0))\*Float.parseFloat(DSSP.get(SP).get(1));

PNCT.setThanhTien(ThanhTien);

if (DSDieuChinhHSDMoi.containsKey(SP)) {

PNCT.setHSDMoiNhat(DateHelper.toDate(DSDieuChinhHSDMoi.get(SP), "yyyy-MM-dd"));

}

PNCTDAO.themPNCT(PNCT);

}

}

/\*

\* Hàm kiểm tra đăng nhập

\*/

boolean xetDangNhap() {

return ShareHelper.NguoiDung != null;

}

/\*

\* Hàm tạo model Phiếu Nhập Hàng

\*/

PhieuNhap taoPhieuNhap() {

PhieuNhap PN = new PhieuNhap();

PN.setMaPhieuNhap(txt\_MaChungTu.getText().trim());

PN.setMaNhanVien(txt\_MaNhanVien.getText().trim());

PN.setNgayNhap(dat\_NgayNhap.getDate());

PN.setMaNCC(((NhaCungCap)cmb\_NhaCungCap.getSelectedItem()).getMaNCC());

PN.setTongSoLuong(Integer.parseInt(txt\_TongSoLuong.getText()));

PN.setChietKhau(Float.parseFloat(txt\_ChietKhau.getText()));

float VAT = (chk\_VAT.isSelected() ? 10f : 0f);

PN.setVAT(VAT);

PN.setThanhTien(Float.parseFloat(txt\_TongCong.getText()));

PN.setGhiChu(txa\_GhiChu.getText());

return PN;

}

/\*

\* Hàm thoát frame thêm mới và cập nhật lại frame xuất huỷ hàng

\*/

boolean capnhatFrameDSPhieuNhap(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.giaodich.nhaphang.DanhSachPhieuNhapJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

/\*

\* Hàm xoá sản phẩm trong danh sách nhập hàng khi click chọn

\*/

void xoaSanPham() {

int index = tbl\_ChiTietPhieuNhap.getSelectedRow();

String MaSP = tbl\_ChiTietPhieuNhap.getValueAt(index, 1).toString();

try {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

if (DialogHelper.confirm(this, "Xoá sản phẩm " + MaSP + " khỏi danh sách nhập hàng?")) {

DSSP.remove(SP);

capnhatDanhSachNhap();

tinhTongSoLuongNhap();

tinhTongGiaTriNhap();

tinhTongTien();

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

void xuatFile() {

String[] columns = {"STT", "Mã SP", "Tên SP", "ĐVT", "HSD", "Số lượng", "Giá nhập", "Thành tiền"};

JFileChooser excelFileChooser = new JFileChooser();

excelFileChooser.setDialogTitle("Save as");

FileNameExtensionFilter fnef = new FileNameExtensionFilter("EXCEL FILES", "xls", "xlsx", "xlxs");

excelFileChooser.setFileFilter(fnef);

int excelChooser = excelFileChooser.showSaveDialog(null);

if (excelChooser == JFileChooser.APPROVE\_OPTION) {

FileOutputStream excelFOU = null;

BufferedOutputStream excelBOU = null;

XSSFWorkbook excelJTableExporter = null;

Row row = null;

Cell cell = null;

try {

excelJTableExporter = new XSSFWorkbook();

XSSFSheet excelSheet = excelJTableExporter.createSheet("JTable Sheet");

// xét font cho header

XSSFFont headerFont = excelJTableExporter.createFont();

headerFont.setBold(true); // in đậm

headerFont.setFontHeightInPoints((short) 14);

// xét font cho title chính

XSSFFont titleFont = excelJTableExporter.createFont();

titleFont.setBold(true);

titleFont.setFontHeightInPoints((short) 18);

titleFont.setColor(IndexedColors.BLUE.getIndex());

//

XSSFFont tieuDePhuFont = excelJTableExporter.createFont();

tieuDePhuFont.setBold(true);

tieuDePhuFont.setFontHeightInPoints((short) 12);

tieuDePhuFont.setColor(IndexedColors.BLUE.getIndex());

// cellStyle cua header bảng

CellStyle headerCellStyle = excelJTableExporter.createCellStyle();

headerCellStyle.setFont(headerFont);

headerCellStyle.setFillForegroundColor(IndexedColors.AQUA.getIndex());

headerCellStyle.setFillPattern(FillPatternType.SOLID\_FOREGROUND);

headerCellStyle.setBorderBottom(BorderStyle.MEDIUM);

headerCellStyle.setBorderTop(BorderStyle.MEDIUM);

headerCellStyle.setBorderLeft(BorderStyle.MEDIUM);

headerCellStyle.setBorderRight(BorderStyle.MEDIUM);

// cellStyle của bảng

CellStyle tableCellStyle = excelJTableExporter.createCellStyle();

tableCellStyle.setBorderBottom(BorderStyle.MEDIUM);

tableCellStyle.setBorderTop(BorderStyle.MEDIUM);

tableCellStyle.setBorderLeft(BorderStyle.MEDIUM);

tableCellStyle.setBorderRight(BorderStyle.MEDIUM);

// test for

for (int i = 0; i < tbl\_ChiTietPhieuNhap.getRowCount() + 10; i++) {

row = excelSheet.createRow(i);

for (int j = 0; j < tbl\_ChiTietPhieuNhap.getColumnCount(); j++) {

cell = row.createCell(j);

if (i == 0 && j == 0) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue("PHIẾU NHẬP KHO HÀNG");

titleCellStyle.setFont(titleFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.CENTER); // xét nằm giữa

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(0, 0, 0, columns.length - 1)); // gộp dòng

}

if (i == 1 && j == 0) {

CellStyle NgayThongKeCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

String NgayThangNam = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String Ngay = NgayThangNam.substring(0, 2);

String Thang = NgayThangNam.substring(3, 5);

String Nam = NgayThangNam.substring(6, 10);

cell.setCellValue("Ngày " + Ngay + " tháng " + Thang + " năm " + Nam);

NgayThongKeCellStyle.setAlignment(HorizontalAlignment.CENTER); // xét nằm giữa

NgayThongKeCellStyle.setFont(tieuDePhuFont);

cell.setCellStyle(NgayThongKeCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(1, 1, 0, columns.length - 1)); // gộp dòng

}

if (i == 2 && j == 0) {

CellStyle SoPhieuCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

String MaPhieuNhap = txt\_MaChungTu.getText();

cell.setCellValue("Số phiếu: " + MaPhieuNhap);

SoPhieuCellStyle.setAlignment(HorizontalAlignment.CENTER); // xét nằm giữa

SoPhieuCellStyle.setFont(tieuDePhuFont);

cell.setCellStyle(SoPhieuCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(2, 2, 0, columns.length - 1)); // gộp dòng

}

if (i == 4 && j == 5) {

CellStyle TongSoLuongCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

TongSoLuongCellStyle.setFont(tieuDePhuFont);

TongSoLuongCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(TongSoLuongCellStyle);

int TongSL = Integer.parseInt(txt\_TongSoLuong.getText());

cell.setCellValue("Tổng số lượng: " + TongSL);

excelSheet.addMergedRegion(new CellRangeAddress(4, 4, 5, columns.length - 1)); // gộp dòng

}

}

}

// xét vị trí giá trị cho header bảng

row = excelSheet.createRow(5);

for (int i = 0; i < columns.length; i++) {

cell = row.createCell(i);

cell.setCellValue(columns[i]);

cell.setCellStyle(headerCellStyle);

}

// xet vị trí giá trị cho bảng

for (int i = 0; i < tbl\_ChiTietPhieuNhap.getRowCount(); i++) {

row = excelSheet.createRow(i + 6);

for (int j = 0; j < tbl\_ChiTietPhieuNhap.getColumnCount() - 2; j++) {

cell = row.createCell(j);

cell.setCellValue(tbl\_ChiTietPhieuNhap.getValueAt(i, j).toString());

cell.setCellStyle(tableCellStyle);

// xet size cho cot

excelSheet.autoSizeColumn(i);

excelSheet.autoSizeColumn(i + 1);

}

}

// các giá trị thống kê

for (int i = 0; i < tbl\_ChiTietPhieuNhap.getRowCount()+30; i++) {

row = excelSheet.createRow(i + 6 + tbl\_ChiTietPhieuNhap.getRowCount());

for (int j = 0; j < tbl\_ChiTietPhieuNhap.getColumnCount() - 2; j++) {

cell = row.createCell(j);

if (i == 0 && j == 4) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue("Tạm tính");

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(6 + tbl\_ChiTietPhieuNhap.getRowCount(), 6 + tbl\_ChiTietPhieuNhap.getRowCount(), 4, 5)); // gộp dòng

}

if (i == 0 && j == 6) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue(txt\_ThanhTien.getText());

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(6 + tbl\_ChiTietPhieuNhap.getRowCount(), 6 + tbl\_ChiTietPhieuNhap.getRowCount(), 6, 7)); // gộp dòng

}

if (i == 1 && j == 4) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue("Chiết khấu (%)");

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(7 + tbl\_ChiTietPhieuNhap.getRowCount(), 7 + tbl\_ChiTietPhieuNhap.getRowCount(), 4, 5)); // gộp dòng

}

if (i == 1 && j == 6) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue(txt\_ChietKhau.getText());

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(7 + tbl\_ChiTietPhieuNhap.getRowCount(), 7 + tbl\_ChiTietPhieuNhap.getRowCount(), 6, 7)); // gộp dòng

}

if (i == 2 && j == 4) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue("VAT (%)");

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(8 + tbl\_ChiTietPhieuNhap.getRowCount(), 8 + tbl\_ChiTietPhieuNhap.getRowCount(), 4, 5)); // gộp dòng

}

if (i == 2 && j == 6) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

String VAT = (chk\_VAT.isSelected() ? "10" : "0");

cell.setCellValue(VAT);

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(8 + tbl\_ChiTietPhieuNhap.getRowCount(), 8 + tbl\_ChiTietPhieuNhap.getRowCount(), 6, 7)); // gộp dòng

}

if (i == 3 && j == 4) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue("Tổng thanh toán");

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(9 + tbl\_ChiTietPhieuNhap.getRowCount(), 9 + tbl\_ChiTietPhieuNhap.getRowCount(), 4, 5)); // gộp dòng

}

if (i == 3 && j == 6) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue(txt\_TongCong.getText());

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(9 + tbl\_ChiTietPhieuNhap.getRowCount(), 9 + tbl\_ChiTietPhieuNhap.getRowCount(), 6, 7)); // gộp dòng

}

if (i == 5 && j == 3) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

NguoiDung ND = NguoiDungDAO.timTenDangNhap(txt\_MaNhanVien.getText());

cell.setCellValue("Người lập phiếu: " + ND.getHoTen());

titleCellStyle.setFont(tieuDePhuFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(11 + tbl\_ChiTietPhieuNhap.getRowCount(), 11 + tbl\_ChiTietPhieuNhap.getRowCount(), 4, 7)); // gộp dòng

}

}

}

excelFOU = new FileOutputStream(excelFileChooser.getSelectedFile() + ".xlsx");

excelBOU = new BufferedOutputStream(excelFOU);

excelJTableExporter.write(excelBOU);

JOptionPane.showMessageDialog(this, "Đã xuất file");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

} finally {

try {

if (excelBOU != null) {

excelBOU.close();

}

if (excelFOU != null) {

excelFOU.close();

}

if (excelJTableExporter != null) {

excelJTableExporter.close();

}

excelFOU.close();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

}

}

* + - 1. **XuatHuyHangJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

NhaCungCapDAO NCCDAO = new NhaCungCapDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuHuyDAO PhieuHuyDAO = new PhieuHuyDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

capnhatDuLieu();

}

/\*

\* Hàm cập nhật dữ liệu Sản Phẩm

\*/

void capnhatDuLieu(){

try {

rad\_TatCa.setSelected(true);

chk\_HuyHetHSD.setSelected(true);

chk\_HuyLoi.setSelected(true);

List<PhieuHuyHang> list = PhieuHuyDAO.select();

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ dữ liệu vào bảng

\*/

void doDuLieuVaoBang(List<PhieuHuyHang> listPH) throws SQLException{

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuHuy.getModel();

dtm.setRowCount(0);

for(PhieuHuyHang PH : listPH){

String LoaiPhieu = "";

switch (PH.getLoaiPhieuHuy()) {

case 0:

LoaiPhieu = "Huỷ hết HSD";

break;

case 1:

LoaiPhieu = "Huỷ lỗi";

break;

}

Object[] row = {PH.getMaPhieuHuy(),

LoaiPhieu,

PH.getNgayHuy(),

PH.getSoLuongHuy(),

dcf.format(PH.getGiaTriHuy()),

PH.getGhiChu()};

dtm.addRow(row);

}

}

/\*

\* Hàm tìm phiếu huỷ theo mã

\*/

void timTheoMa() {

String ChuoiTimKiem = txt\_TimTheoMaChungTu.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

PhieuHuyHang PH = PhieuHuyDAO.timTheoMa(ChuoiTimKiem);

List<PhieuHuyHang> list = new ArrayList<>();

if (PH != null)

list.add(PH);

if (list.size() > 0)

doDuLieuVaoBang(list);

else {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachPhieuHuy.getModel();

dtm.setRowCount(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm tìm phiếu huỷ theo mã sản phẩm hoặc tên sản phẩm

\*/

void timTheoSanPham() {

String ChuoiTimKiem = txt\_TimTheoSanPham.getText();

if (ChuoiTimKiem.isEmpty()) {

capnhatDuLieu();

}

else {

try {

List<PhieuHuyHang> list = PhieuHuyDAO.timTheoSanPham(ChuoiTimKiem);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm lọc phiếu huỷ theo các tuỳ chọn

\*/

void locPhieuHuy() {

String LoaiPhieu = "";

if(chk\_HuyHetHSD.isSelected())

LoaiPhieu += "0";

if(chk\_HuyLoi.isSelected())

LoaiPhieu += "1";

if (LoaiPhieu.isEmpty())

LoaiPhieu += "01";

if (rad\_TatCa.isSelected())

timTatCa(LoaiPhieu);

if (rad\_HomNay.isSelected())

timTrongNgay(LoaiPhieu);

if (rad\_HomQua.isSelected())

timNgayHomQua(LoaiPhieu);

if (rad\_7NgayTruoc.isSelected())

tim7NgayTruoc(LoaiPhieu);

if (rad\_Khac.isSelected())

timTheoKhoangNgay(LoaiPhieu);

}

/\*

\* Hàm tìm tất cả phiếu huỷ theo loại phiếu

\*/

void timTatCa(String LoaiPhieu) {

try {

List<PhieuHuyHang> list = PhieuHuyDAO.timTheoLoaiPhieu(LoaiPhieu);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu huỷ trong 7 ngày trước (không tính hôm nay)

\*/

void tim7NgayTruoc(String LoaiPhieu) {

try {

List<PhieuHuyHang> list = PhieuHuyDAO.tim7NgayTruoc(LoaiPhieu);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu huỷ trong ngày

\*/

void timTrongNgay(String LoaiPhieu) {

try {

List<PhieuHuyHang> list = PhieuHuyDAO.timTrongNgay(LoaiPhieu);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu huỷ tạo ngày hôm qua

\*/

void timNgayHomQua(String LoaiPhieu) {

try {

List<PhieuHuyHang> list = PhieuHuyDAO.timNgayHomQua(LoaiPhieu);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tìm phiếu huỷ theo khoảng ngày tuỳ chọn

\*/

void timTheoKhoangNgay(String LoaiPhieu) {

String TuNgay = DateHelper.toString(dat\_TuNgay.getDate(),"yyyy-MM-dd");

String DenNgay = DateHelper.toString(dat\_DenNgay.getDate(),"yyyy-MM-dd");

if (dat\_TuNgay.getDate().after(dat\_DenNgay.getDate())) {

DialogHelper.alert(this, "Chọn lại ngày");

}

else {

try {

List<PhieuHuyHang> list = PhieuHuyDAO.timTheoKhoangNgay(LoaiPhieu, TuNgay, DenNgay);

doDuLieuVaoBang(list);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

/\*

\* Hàm mở jframe chi tiêt phiếu huỷ

(truyền các thông tin qua frame cập nhật)

\*/

void xemPhieuHuy() {

int index = tbl\_DanhSachPhieuHuy.getSelectedRow();

String MaPH = tbl\_DanhSachPhieuHuy.getValueAt(index, 0).toString();

try {

PhieuHuyHang PH = PhieuHuyDAO.timTheoMa(MaPH);

new ChiTietPhieuHuyJFrame(PH).setVisible(true);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}}

* + - 1. **ThemMoiPhieuHuyJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuHuyDAO PhieuHuyDAO = new PhieuHuyDAO();

PhieuHuyChiTietDAO PHCTDAO = new PhieuHuyChiTietDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// Tạo treemap chứa sản phẩm và số lượng sản phẩm trong danh sách huỷ

static TreeMap<SanPham, Integer> DSSP = new TreeMap<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

doNhomHang();

doDanhSachSanPham();

doThongTin();

kiemDanhSachDeNghiHuy();

}

/\*

\* Hàm kiểm tra danh sách đề nghị huỷ,

nếu có sản phẩm thì đổ sản phẩm vào TreeMap DSSP

\*/

void kiemDanhSachDeNghiHuy(){

if (ShareHelper.listDeNghiHuy != null) {

for (SanPham SP : ShareHelper.listDeNghiHuy) {

DSSP.put(SP, 0);

}

capnhatDanhSachHuy();

}

}

/\*

\* Hàm đổ các thông tin mặc định

\*/

void doThongTin() {

try {

// khởi tạo ngày huỷ (là ngày hiện tại)

dat\_NgayHuy.setDate(DateHelper.now());

// đổ mã nhân viên đang thao tác vào textfield

txt\_MaNhanVien.setText(ShareHelper.NguoiDung.getTenDangNhap());

// khởi tạo mã phiếu kiểm

int demPhieu = PhieuHuyDAO.demTongPhieu();

String MaPhieuHuy = "PH" + (demPhieu+1);

txt\_MaChungTu.setText(MaPhieuHuy);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách nhóm hàng vào combobox ở tab DanhSachSanPham

\*/

void doNhomHang() {

DefaultComboBoxModel dcm = (DefaultComboBoxModel) cmb\_NhomHang.getModel();

dcm.removeAllElements();

try {

List<NhomHang> ListNH = NhomHangDAO.select();

dcm.addElement("Tất cả");

for (NhomHang NH : ListNH) {

dcm.addElement(NH);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này đổ danh sách sản phẩm vào tab Danh Sách Sản Phẩm

\*/

void doDanhSachSanPham() {

try {

List<SanPham> ListSP = SanPhamDAO.select();

doDuLieuSanPham(ListSP);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn: " + e.toString());

}

}

/\*

\* Hàm này tìm sản phẩm ở tab DANH SÁCH SẢN PHẨM

\*/

void timSanPham() {

List<SanPham> list;

// get tên nhóm hàng và mã nhóm hàng

String TuyChonNhomHang = cmb\_NhomHang.getSelectedItem().toString();

// get chuỗi tìm kiếm

String ChuoiTimKiem = txt\_TimKiemSanPham.getText();

// nếu chuỗi rỗng thì đổ tất cả sản phẩm theo mã nhóm

if (ChuoiTimKiem.isEmpty()) {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.select();

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timTheoMaNhom(MaNhom);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

} else {

// nếu tên nhóm hàng là TẤT CẢ thì truy vấn tất cả

if (TuyChonNhomHang.equalsIgnoreCase("Tất cả")) {

try {

list = SanPhamDAO.timSanPhamTheoTen(ChuoiTimKiem);

doDuLieuSanPham(list);

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

} else {

try {

NhomHang NhomHang = NhomHangDAO.timMaNhom(TuyChonNhomHang);

String MaNhom = NhomHang.getMaNhomHang();

list = SanPhamDAO.timKetHopNhomHang(ChuoiTimKiem, MaNhom);

if (list != null) {

doDuLieuSanPham(list);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi truy vấn:" + e.toString());

}

}

}

}

/\*

\* Hàm thao tác đổ dữ liệu lên bảng Sản Phẩm

\*/

void doDuLieuSanPham(List<SanPham> list) {

DefaultTableModel dtm = (DefaultTableModel) tbl\_DanhSachSanPham.getModel();

dtm.setRowCount(0);

int STT = 0;

for (SanPham sp : list) {

STT++;

Object[] row = {

STT,

sp.getMaSanPham(),

sp.getTenSanPham(),

sp.getTonKho(),

dcf.format(sp.getGiaVon()),

dcf.format(sp.getGiaBan())

};

dtm.addRow(row);

}

}

/\*

\* Hàm thêm sản phẩm ở tbl\_DSSP vào tab chi tiết phiếu huỷ

\*/

void themSanPham() {

int index = tbl\_DanhSachSanPham.getSelectedRow();

String MaSP = tbl\_DanhSachSanPham.getValueAt(index, 1).toString();

try {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

// check trong TreeMap có SP chưa, nếu có thì báo lỗi, chưa thì add vào

if (DSSP.containsKey(SP))

DialogHelper.alert(this, "Sản phẩm đã có trong danh sách huỷ");

else {

DSSP.put(SP, 0);

capnhatDanhSachHuy();

tab\_ChiTietPhieu.setSelectedIndex(0);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm cập nhật danh sách huỷ

\*/

void capnhatDanhSachHuy() {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietPhieuHuy.getModel();

dtm.setRowCount(0);

int STT = 1;

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

Object[] row = {STT++,

SP.getMaSanPham(),

SP.getTenSanPham(),

DVT.getMoTa(),

dcf.format(SP.getGiaVon()),

DSSP.get(SP),

dcf.format(DSSP.get(SP)\*SP.getGiaVon())

};

dtm.addRow(row);

}

}

/\*

\* Hàm nhập số lượng cần huỷ vào bảng

sau khi nhập thì cập nhật lại bảng

\*/

void nhapSoLuongHuy() {

try {

int index = tbl\_ChiTietPhieuHuy.getEditingRow();

if (index >= 0) {

String MaSP = tbl\_ChiTietPhieuHuy.getValueAt(index, 1).toString();

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

int SLHuy = Integer.parseInt(tbl\_ChiTietPhieuHuy.getValueAt(index, 5)+"");

DSSP.put(SP, SLHuy);

}

} catch (Exception e) {

DialogHelper.alert(this, "Lỗi nhập liệu");

}

finally {

capnhatDanhSachHuy();

tinhTongSoLuongHuy();

tinhTongGiaTriHuy();

}

}

/\*

\* Hàm cập nhật tổng số lượng huỷ

\*/

void tinhTongSoLuongHuy() {

int TongSLHuy = 0;

for (SanPham SP : DSSP.keySet()) {

TongSLHuy += DSSP.get(SP);

}

txt\_TongSoLuong.setText(TongSLHuy+"");

}

/\*

\* Hàm cập nhật tổng giá trị huỷ

\*/

void tinhTongGiaTriHuy() {

float TongGTHuy = 0;

for (SanPham SP : DSSP.keySet()) {

TongGTHuy += DSSP.get(SP)\*SP.getGiaVon();

}

txt\_TongGiaTri.setText(TongGTHuy+"");

}

/\*

\* Hàm xoá hết sản phẩm trong danh sách huỷ

\*/

void xoaDanhSachHuy() {

DSSP.clear();

capnhatDanhSachHuy();

tinhTongGiaTriHuy();

tinhTongSoLuongHuy();

}

/\*

\* Hàm check ghi chú: không được để trống, tối thiểu phải đủ 3 ký tự

\*/

boolean kiemGhiChu() {

if(txa\_GhiChu.getText().length() < 3)

return false;

return true;

}

/\*

\* Hàm kiểm tra dssp

- có null hay không, null thì không được phép hoàn tất

\*/

boolean kiemDSSP() {

return !DSSP.isEmpty();

}

/\*

\* Hàm lưu phiếu huỷ và cân bằng kho

\*/

void themPhieuHuy() {

if (xetDangNhap() && kiemGhiChu() && kiemDSSP() && kiemSoLuongThucTe()) {

PhieuHuyHang PH = taoPhieuHuy();

try {

int rs = PhieuHuyDAO.themPhieuHuy(PH);

if (rs > 0) {

themPHCT(PH);

canbangTonKho();

DialogHelper.alert(this, "Đã thêm phiếu huỷ và cân bằng kho");

xoaDanhSachHuy();

if (capnhatFrameXuatHuy()) {

this.dispose();

new XuatHuyHangJFrame().setVisible(true);

}

else

this.dispose();

}

else

DialogHelper.alert(this, "Thêm mới thất bại");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

else {

DialogHelper.alert(this, "Kiểm tra lại các thông tin");

}

}

/\*

\* Hàm kiểm tra số lượng thực tế của SP trước khi huỷ

nếu có SP có SL < 1 thì không cho huỷ

\*/

boolean kiemSoLuongThucTe() {

for (SanPham SP : DSSP.keySet()) {

if (DSSP.get(SP) < 1)

return false;

}

return true;

}

/\*

\* Hàm cập nhật lại tồn kho của SP (cân bằng kho hàng)

\*/

void canbangTonKho() {

for (SanPham SP : DSSP.keySet()) {

int TonKhoMoi = SP.getTonKho() - DSSP.get(SP);

SanPhamDAO.capnhatTonKho(SP.getMaSanPham(), TonKhoMoi);

}

}

/\*

\* Hàm thêm PHCT

\*/

void themPHCT(PhieuHuyHang PH) {

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

PhieuHuyChiTiet PHCT = new PhieuHuyChiTiet();

PHCT.setMaPhieuHuy(PH.getMaPhieuHuy());

PHCT.setMaSanPham(SP.getMaSanPham());

PHCT.setDVT(DVT.getMoTa());

PHCT.setSoLuong(DSSP.get(SP));

PHCT.setGiaVon(SP.getGiaVon());

PHCT.setThanhTien(SP.getGiaVon()\*DSSP.get(SP));

PHCTDAO.themPHCT(PHCT);

}

}

/\*

\* Hàm kiểm tra đăng nhập

\*/

boolean xetDangNhap() {

if (ShareHelper.NguoiDung != null)

return true;

return false;

}

/\*

\* Hàm tạo model Phiếu Huỷ Hàng

\*/

PhieuHuyHang taoPhieuHuy() {

PhieuHuyHang PH = new PhieuHuyHang();

PH.setMaPhieuHuy(txt\_MaChungTu.getText().trim());

PH.setLoaiPhieuHuy(rad\_HuyHetHSD.isSelected() ? 0 : 1);

PH.setMaNhanVien(txt\_MaNhanVien.getText().trim());

PH.setNgayHuy(dat\_NgayHuy.getDate());

PH.setSoLuongHuy(Integer.parseInt(txt\_TongSoLuong.getText()));

PH.setGiaTriHuy(Float.parseFloat(txt\_TongGiaTri.getText()));

PH.setGhiChu(txa\_GhiChu.getText());

PH.setDSSP(DSSP);

return PH;

}

/\*

\* Hàm thoát frame thêm mới và cập nhật lại frame xuất huỷ hàng

\*/

boolean capnhatFrameXuatHuy(){

Frame[] listFrames = Frame.getFrames();

for (Frame fr : listFrames) {

String tenFrame = fr.getClass().getName();

if (tenFrame.equalsIgnoreCase("ui.quanly.giaodich.xuathuyhang.XuatHuyHangJFrame")) {

fr.dispose();

return true;

}

}

return false;

}

/\*

\* Hàm xoá sản phẩm trong danh sách huỷ khi click chọn

\*/

void xoaSanPham() {

int index = tbl\_ChiTietPhieuHuy.getSelectedRow();

String MaSP = tbl\_ChiTietPhieuHuy.getValueAt(index, 1).toString();

try {

SanPham SP = SanPhamDAO.timTheoMaSP(MaSP);

if (DialogHelper.confirm(this, "Xoá sản phẩm " + MaSP + "khỏi danh sách huỷ?")) {

DSSP.remove(SP);

capnhatDanhSachHuy();

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **ChiTietPhieuHuyJFrame**

NhomHangDAO NhomHangDAO = new NhomHangDAO();

SanPhamDAO SanPhamDAO = new SanPhamDAO();

DonViTinhDAO DVTDAO = new DonViTinhDAO();

PhieuHuyDAO PhieuHuyDAO = new PhieuHuyDAO();

PhieuHuyChiTietDAO PHCTDAO = new PhieuHuyChiTietDAO();

// Định dạng hiển thị số tiền:

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

// Tạo treemap chứa sản phẩm và số lượng sản phẩm trong danh sách huỷ

static TreeMap<SanPham, Integer> DSSP = new TreeMap<>();

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

this.setSize(1366, 768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

setResizable(false);

setLocationRelativeTo(null);

tab\_ChiTietPhieu.setEnabledAt(1, false);

btn\_XuatHuy.setEnabled(false);

btn\_CapNhatDanhSach.setEnabled(false);

btn\_XoaDanhSach.setEnabled(false);

doThongTin();

}

/\*

\* Hàm đổ các thông tin mặc định

\*/

void doThongTin() {

try {

// khởi tạo ngày huỷ

dat\_NgayHuy.setDate(PH.getNgayHuy());

// đổ mã nhân viên đang thao tác vào textfield

txt\_MaNhanVien.setText(PH.getMaNhanVien());

// đổ mã phiếu kiểm

txt\_MaChungTu.setText(PH.getMaPhieuHuy());

// đổ loại phiếu huỷ

if (PH.getLoaiPhieuHuy() == 0)

rad\_HuyHetHSD.setSelected(true);

else

rad\_HuyHangLoi.setSelected(true);

// số lượng huỷ

txt\_TongSoLuong.setText(PH.getSoLuongHuy()+"");

// giá trị huỷ

txt\_TongGiaTri.setText(dcf.format(PH.getGiaTriHuy()));

// ghi chú

txa\_GhiChu.setText(PH.getGhiChu());

// đổ thông tin dssp vào treemap

List<PhieuHuyChiTiet> listPHCT = PHCTDAO.timDanhSachPHCT(PH.getMaPhieuHuy());

for (PhieuHuyChiTiet PHCT : listPHCT) {

SanPham SP = SanPhamDAO.timTheoMaSP(PHCT.getMaSanPham());

DSSP.put(SP, PHCT.getSoLuong());

}

capnhatDanhSachHuy();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm cập nhật danh sách huỷ

\*/

void capnhatDanhSachHuy() {

DefaultTableModel dtm = (DefaultTableModel) tbl\_ChiTietPhieuHuy.getModel();

dtm.setRowCount(0);

int STT = 1;

for (SanPham SP : DSSP.keySet()) {

DonViTinh DVT = DVTDAO.timTheoMa(SP.getMaDVT());

Object[] row = {STT++,

SP.getMaSanPham(),

SP.getTenSanPham(),

DVT.getMoTa(),

dcf.format(SP.getGiaVon()),

DSSP.get(SP),

dcf.format(DSSP.get(SP)\*SP.getGiaVon())

};

dtm.addRow(row);

}

}

}

* + - 1. **BaoCaoBanHangJFrame**

ThongKeDAO TKDAO = new ThongKeDAO();

HoaDonDAO HoaDonDAO = new HoaDonDAO();

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\*

\*/

void inIt() {

// start code setSize cố định không cho co giãn lung tung:

this.setSize(1366,768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

// end code setSize

setResizable(false);

setLocationRelativeTo(null);

locToanThoiGian();

}

/\*

\* Hàm báo cáo toàn thời gian

\*/

void locToanThoiGian() {

try {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

lbl\_NgayThongKe.setText("Từ ngày 01-10-2019 đến ngày " + NgayHienTai);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

double TongDoanhThu = TKDAO.tinhTongDoanhThu();

lbl\_DoanhThu.setText("Doanh thu: " + dcf.format(TongDoanhThu));

int TongHoaDon = HoaDonDAO.demHoaDon();

lbl\_SoDonHang.setText("Số đơn hàng: " + TongHoaDon);

List<HoaDon> listHD = HoaDonDAO.select();

doThongTinHoaDon(listHD);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo doanh thu hôm nay

\*/

void locHomNay() {

try {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

lbl\_NgayThongKe.setText("Từ ngày " + NgayHienTai + " đến ngày " + NgayHienTai);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

double DoanhThu = TKDAO.tinhDoanhThuTrongNgay();

lbl\_DoanhThu.setText("Doanh thu: " + dcf.format(DoanhThu));

int SoHoaDon = TKDAO.demSoDonTrongNgay();

lbl\_SoDonHang.setText("Số đơn hàng: " + SoHoaDon);

List<HoaDon> listHD = HoaDonDAO.timTrongNgay();

doThongTinHoaDon(listHD);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo doanh thu trong tuần

\*/

void locTrongTuan() {

try {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String NgayDauTuan = DateHelper.tinhNgayDauTuan(0);

lbl\_NgayThongKe.setText("Từ ngày " + NgayDauTuan + " đến ngày " + NgayHienTai);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

double DoanhThu = TKDAO.tinhDoanhThuTrongTuan();

lbl\_DoanhThu.setText("Doanh thu: " + dcf.format(DoanhThu));

int SoHoaDon = TKDAO.demSoDonTrongTuan();

lbl\_SoDonHang.setText("Số đơn hàng: " + SoHoaDon);

List<HoaDon> listHD = HoaDonDAO.locHoaDonTrongTuan();

doThongTinHoaDon(listHD);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo doanh thu trong tháng

\*/

void locTrongThang() {

try {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String ThangHienTai = NgayHienTai.substring(3, 5);

String NamHienTai = NgayHienTai.substring(6, 10);

lbl\_NgayThongKe.setText("Thống kê: Tháng " + ThangHienTai + "-" + NamHienTai );

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

double DoanhThu = TKDAO.tinhDoanhThuTrongThang();

lbl\_DoanhThu.setText("Doanh thu: " + dcf.format(DoanhThu));

int SoHoaDon = TKDAO.demSoDonTrongThang();

lbl\_SoDonHang.setText("Số đơn hàng: " + SoHoaDon);

List<HoaDon> listHD = HoaDonDAO.locHoaDonTrongThang();

doThongTinHoaDon(listHD);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo doanh thu trong quý

\*/

void locTrongQuy() {

try {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

int Quy = DateHelper.tinhSoQuyTrongNam();

String NamHienTai = NgayHienTai.substring(6,10);

lbl\_NgayThongKe.setText("Thống kê: Quý " + Quy + "-" + NamHienTai);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

double DoanhThu = TKDAO.tinhDoanhThuTrongQuy();

lbl\_DoanhThu.setText("Doanh thu: " + dcf.format(DoanhThu));

int SoHoaDon = TKDAO.demSoDonTrongQuy();

lbl\_SoDonHang.setText("Số đơn hàng: " + SoHoaDon);

List<HoaDon> listHD = HoaDonDAO.locHoaDonTrongQuy();

doThongTinHoaDon(listHD);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo doanh thu trong năm

\*/

void locTrongNam() {

try {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String NamHienTai = NgayHienTai.substring(6,10);

lbl\_NgayThongKe.setText("Thống kê năm: " + NamHienTai);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

double DoanhThu = TKDAO.tinhDoanhThuTrongNam();

lbl\_DoanhThu.setText("Doanh thu: " + dcf.format(DoanhThu));

int SoHoaDon = TKDAO.demSoDonTrongNam();

lbl\_SoDonHang.setText("Số đơn hàng: " + SoHoaDon);

List<HoaDon> listHD = HoaDonDAO.locHoaDonTrongNam();

doThongTinHoaDon(listHD);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm đổ thông tin vào table

\*/

void doThongTinHoaDon(List<HoaDon> listHD) {

DefaultTableModel dtm = (DefaultTableModel) tbl\_BangBaoCao.getModel();

dtm.setRowCount(0);

int STT = 1;

for (HoaDon HD : listHD) {

Object[] row = {STT++,

HD.getMaHoaDon(),

DateHelper.toString(HD.getNgayBan(), "dd-MM-yyyy"),

HD.getMaNhanVien(),

HD.getTongSoLuong(),

dcf.format(HD.getChietKhau()),

dcf.format(HD.getThanhTien())};

dtm.addRow(row);

}

}

void xuatFile() {

String[] columns = {"STT", "Mã HĐ", "Ngày bán", "Người bán", "Số lượng SP", "Chiết khấu", "Giá Trị"};

JFileChooser excelFileChooser = new JFileChooser();

excelFileChooser.setDialogTitle("Save as");

FileNameExtensionFilter fnef = new FileNameExtensionFilter("EXCEL FILES", "xls", "xlsx", "xlxs");

excelFileChooser.setFileFilter(fnef);

int excelChooser = excelFileChooser.showSaveDialog(null);

if (excelChooser == JFileChooser.APPROVE\_OPTION) {

FileOutputStream excelFOU = null;

BufferedOutputStream excelBOU = null;

XSSFWorkbook excelJTableExporter = null;

Row row = null;

Cell cell = null;

try {

excelJTableExporter = new XSSFWorkbook();

XSSFSheet excelSheet = excelJTableExporter.createSheet("JTable Sheet");

excelSheet.setDefaultColumnWidth(15);

// xét font cho header

XSSFFont headerFont = excelJTableExporter.createFont();

headerFont.setBold(true); // in đậm

headerFont.setFontHeightInPoints((short) 14);

// xét font cho title chính

XSSFFont titleFont = excelJTableExporter.createFont();

titleFont.setBold(true);

titleFont.setFontHeightInPoints((short) 18);

titleFont.setColor(IndexedColors.BLUE.getIndex());

//

XSSFFont tieuDePhuFont = excelJTableExporter.createFont();

tieuDePhuFont.setBold(true);

tieuDePhuFont.setFontHeightInPoints((short) 12);

tieuDePhuFont.setColor(IndexedColors.BLUE.getIndex());

// cellStyle cua header bảng

CellStyle headerCellStyle = excelJTableExporter.createCellStyle();

headerCellStyle.setFont(headerFont);

headerCellStyle.setFillForegroundColor(IndexedColors.AQUA.getIndex());

headerCellStyle.setFillPattern(FillPatternType.SOLID\_FOREGROUND);

headerCellStyle.setBorderBottom(BorderStyle.MEDIUM);

headerCellStyle.setBorderTop(BorderStyle.MEDIUM);

headerCellStyle.setBorderLeft(BorderStyle.MEDIUM);

headerCellStyle.setBorderRight(BorderStyle.MEDIUM);

// cellStyle của bảng

CellStyle tableCellStyle = excelJTableExporter.createCellStyle();

tableCellStyle.setBorderBottom(BorderStyle.MEDIUM);

tableCellStyle.setBorderTop(BorderStyle.MEDIUM);

tableCellStyle.setBorderLeft(BorderStyle.MEDIUM);

tableCellStyle.setBorderRight(BorderStyle.MEDIUM);

// test for

for (int i = 0; i < tbl\_BangBaoCao.getRowCount() + 10; i++) {

row = excelSheet.createRow(i);

for (int j = 0; j < tbl\_BangBaoCao.getColumnCount(); j++) {

cell = row.createCell(j);

if (i == 0 && j == 0) {

CellStyle titleCellStyle = excelJTableExporter.createCellStyle();

cell.setCellValue(lbl\_TenBaoCao.getText());

titleCellStyle.setFont(titleFont); // add font vào cell

titleCellStyle.setAlignment(HorizontalAlignment.CENTER); // xét nằm giữa

cell.setCellStyle(titleCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(0, 0, 0, columns.length - 1)); // gộp dòng

}

if (i == 1 && j == 0) {

CellStyle NgayThongKeCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

cell.setCellValue(lbl\_NgayThongKe.getText());

NgayThongKeCellStyle.setAlignment(HorizontalAlignment.CENTER); // xét nằm giữa

NgayThongKeCellStyle.setFont(tieuDePhuFont);

cell.setCellStyle(NgayThongKeCellStyle);

excelSheet.addMergedRegion(new CellRangeAddress(1, 1, 0, columns.length - 1)); // gộp dòng

}

if (i == 4 && j == 0) {

CellStyle TongSoLuongCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

TongSoLuongCellStyle.setFont(tieuDePhuFont);

TongSoLuongCellStyle.setAlignment(HorizontalAlignment.LEFT);

cell.setCellStyle(TongSoLuongCellStyle);

cell.setCellValue(lbl\_NgayTao.getText());

excelSheet.addMergedRegion(new CellRangeAddress(4, 4, 0, 3)); // gộp dòng

}

if (i == 5 && j == 0) {

CellStyle TongSoLuongCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

TongSoLuongCellStyle.setFont(tieuDePhuFont);

TongSoLuongCellStyle.setAlignment(HorizontalAlignment.LEFT);

cell.setCellStyle(TongSoLuongCellStyle);

cell.setCellValue(lbl\_SoDonHang.getText());

excelSheet.addMergedRegion(new CellRangeAddress(5, 5, 0, 3)); // gộp dòng

}

if (i == 5 && j == 4) {

CellStyle TongSoLuongCellStyle = excelJTableExporter.createCellStyle(); // tạo cellStyle để định dạng

TongSoLuongCellStyle.setFont(tieuDePhuFont);

TongSoLuongCellStyle.setAlignment(HorizontalAlignment.RIGHT);

cell.setCellStyle(TongSoLuongCellStyle);

cell.setCellValue(lbl\_DoanhThu.getText());

excelSheet.addMergedRegion(new CellRangeAddress(5, 5, 4, 6)); // gộp dòng

}

}

}

// xét vị trí giá trị cho header bảng

row = excelSheet.createRow(6);

for (int i = 0; i < columns.length; i++) {

cell = row.createCell(i);

cell.setCellValue(columns[i]);

cell.setCellStyle(headerCellStyle);

}

// xet vị trí giá trị cho bảng

for (int i = 0; i < tbl\_BangBaoCao.getRowCount(); i++) {

row = excelSheet.createRow(i + 7);

for (int j = 0; j < tbl\_BangBaoCao.getColumnCount(); j++) {

cell = row.createCell(j);

cell.setCellValue(tbl\_BangBaoCao.getValueAt(i, j).toString());

cell.setCellStyle(tableCellStyle);

// xet size cho cot

excelSheet.autoSizeColumn(i);

excelSheet.autoSizeColumn(i + 1);

}

}

excelFOU = new FileOutputStream(excelFileChooser.getSelectedFile() + ".xlsx");

excelBOU = new BufferedOutputStream(excelFOU);

excelJTableExporter.write(excelBOU);

JOptionPane.showMessageDialog(this, "Đã xuất file");

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

} finally {

try {

if (excelBOU != null) {

excelBOU.close();

}

if (excelFOU != null) {

excelFOU.close();

}

if (excelJTableExporter != null) {

excelJTableExporter.close();

}

excelFOU.close();

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

}

}

* + - 1. **BaoCaoKhoHangJFrame**

ThongKeDAO TKDAO = new ThongKeDAO();

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho jframe

\*/

void inIt() {

// start code setSize cố định không cho co giãn lung tung:

this.setSize(1366,768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

// end code setSize

setResizable(false);

setLocationRelativeTo(null);

bcGiaTriKhoHang();

}

/\*

\* Hàm báo cáo giá trị kho hàng

\*/

void bcGiaTriKhoHang() {

lbl\_TenBaoCao.setText("Báo cáo giá trị kho hàng");

lbl\_NgayThongKe.setText("Ngày thống kê: " + DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

lbl\_NgayTao.setVisible(false);

rad\_ToanThoiGian.setEnabled(false);

rad\_HomNay.setEnabled(false);

rad\_TuanNay.setEnabled(false);

rad\_ThangNay.setEnabled(false);

rad\_QuyNay.setEnabled(false);

rad\_NamNay.setEnabled(false);

rad\_Khac.setEnabled(false);

DefaultTableModel dtm = (DefaultTableModel) tbl\_BangBaoCao.getModel();

dtm.setRowCount(0);

String[] header = {"STT", "Nhóm hàng", "Mã hàng", "Tên hàng", "DVT", "Số lượng", "Đơn giá", "Giá trị"};

dtm.setColumnIdentifiers(header);

try {

List<Object[]> list = TKDAO.thongkeKhoHang();

int STT = 1;

for (int i = 0; i < list.size(); i++) {

Object[] row = {STT++,

list.get(i)[0],

list.get(i)[1],

list.get(i)[2],

list.get(i)[3],

list.get(i)[4],

dcf.format(list.get(i)[5]),

dcf.format(list.get(i)[6])};

dtm.addRow(row);

}

int TongSL = TKDAO.demTongSanPham();

double TongGT = TKDAO.tongGiaTriKho();

lbl\_TongSoLuong.setText("Tổng số lượng: " + TongSL);

lbl\_TongGiaTri.setText("Tổng giá trị: " + dcf.format(TongGT));

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo hàng bán chạy

\*/

void bcHangBanChay () {

rad\_ToanThoiGian.setEnabled(true);

rad\_TuanNay.setEnabled(true);

if (rad\_ToanThoiGian.isSelected())

bcHangBanChayToanTG();

if (rad\_TuanNay.isSelected())

bcHangBanChayTuanNay();

}

/\*

\* Hàm báo cáo hàng bán chạy toàn thời gian

\*/

void bcHangBanChayToanTG() {

lbl\_TenBaoCao.setText("Danh sách sản phẩm bán chạy");

lbl\_NgayThongKe.setText("Ngày thống kê: " + DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

lbl\_NgayTao.setVisible(false);

lbl\_TongGiaTri.setVisible(false);

lbl\_TongSoLuong.setVisible(false);

DefaultTableModel dtm = (DefaultTableModel) tbl\_BangBaoCao.getModel();

dtm.setRowCount(0);

String[] header = {"Mã SP", "Tên sản phẩm", "ĐVT", "Số lượng đã bán"};

dtm.setColumnIdentifiers(header);

try {

List<Object[]> list = TKDAO.locSPBanChayToanThoiGian();

for (int i = 0; i < list.size(); i++) {

Object[] row = list.get(i);

dtm.addRow(row);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo hàng bán chạy tuần này

\*/

void bcHangBanChayTuanNay() {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String NgayDauTuan = DateHelper.tinhNgayDauTuan(0);

lbl\_TenBaoCao.setText("Danh sách sản phẩm bán chạy");

lbl\_NgayThongKe.setText("Ngày thống kê: Từ " + NgayDauTuan + " đến " + NgayHienTai);

lbl\_NgayTao.setVisible(true);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

lbl\_TongGiaTri.setVisible(false);

lbl\_TongSoLuong.setVisible(false);

DefaultTableModel dtm = (DefaultTableModel) tbl\_BangBaoCao.getModel();

dtm.setRowCount(0);

String[] header = {"Mã SP", "Tên sản phẩm", "ĐVT", "Số lượng đã bán"};

dtm.setColumnIdentifiers(header);

try {

List<Object[]> list = TKDAO.locSPBanChay(DateHelper.tinhNgayDauTuan(1), DateHelper.toString(DateHelper.now(), "yyyy-MM-dd"));

for (int i = 0; i < list.size(); i++) {

Object[] row = list.get(i);

dtm.addRow(row);

}

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **BaoCaoNhapHangJFrame**

ThongKeDAO TKDAO = new ThongKeDAO();

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

// start code setSize cố định không cho co giãn lung tung:

this.setSize(1366,768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

// end code setSize

setResizable(false);

setLocationRelativeTo(null);

bcTongHopHangNhap();

}

/\*

\* Hàm báo cáo tổng hợp hàng nhập

\*/

void bcTongHopHangNhap() {

lbl\_TenBaoCao.setText("Báo cáo tổng hợp hàng nhập");

lbl\_NgayTao.setText("Ngày tạo: " + DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

if (rad\_ToanThoiGian.isSelected())

bcTongHopHangNhapToanTG();

if (rad\_TuanNay.isSelected())

bcTongHopHangNhapTrongTuan();

}

/\*

\* Hàm báo cáo tổng hợp hàng nhập toàn thời gian

\*/

void bcTongHopHangNhapToanTG() {

lbl\_NgayThongKe.setText("Từ ngày 01-10-2019 đến " + DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

DefaultTableModel dtm = (DefaultTableModel) tbl\_BangBaoCao.getModel();

dtm.setRowCount(0);

String[] header = {"STT", "Mã SP", "Tên SP", "ĐVT", "Tổng số lượng", "Tổng giá trị nhập"};

dtm.setColumnIdentifiers(header);

try {

List<Object[]> list = TKDAO.thongkeHangNhapToanTG();

int STT = 1;

for (int i = 0; i < list.size(); i++) {

Object[] row = {STT++,

list.get(i)[0],

list.get(i)[1],

list.get(i)[2],

list.get(i)[3],

dcf.format(list.get(i)[4])};

dtm.addRow(row);

}

double TongGT = TKDAO.tinhTongGiaTriHangNhap();

int TongSL = TKDAO.tinhTongSLHangNhap();

lbl\_TongGiaTri.setText("Tổng giá trị (bao gồm VAT & chiết khấu): " + dcf.format(TongGT));

lbl\_TongSoLuong.setText("Tổng số lượng: " + TongSL);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm tổng hợp hàng nhập trong tuần

\*/

void bcTongHopHangNhapTrongTuan() {

String NgayDauTuan = DateHelper.tinhNgayDauTuan(0);

lbl\_NgayThongKe.setText("Từ ngày " + NgayDauTuan + " đến " + DateHelper.toString(DateHelper.now(), "dd-MM-yyyy"));

DefaultTableModel dtm = (DefaultTableModel) tbl\_BangBaoCao.getModel();

dtm.setRowCount(0);

String[] header = {"STT", "Mã SP", "Tên SP", "ĐVT", "Tổng số lượng", "Tổng giá trị nhập"};

dtm.setColumnIdentifiers(header);

try {

List<Object[]> list = TKDAO.thongkeHangNhap(DateHelper.tinhNgayDauTuan(1), DateHelper.toString(DateHelper.now(), "yyyy-MM-dd"));

int STT = 1;

for (int i = 0; i < list.size(); i++) {

Object[] row = {STT++,

list.get(i)[0],

list.get(i)[1],

list.get(i)[2],

list.get(i)[3],

dcf.format(list.get(i)[4])};

dtm.addRow(row);

}

double GTNhap = TKDAO.tinhTongGiaTriHangNhap(DateHelper.tinhNgayDauTuan(1), DateHelper.toString(DateHelper.now(), "yyyy-MM-dd"));

int SLNhap = TKDAO.tinhTongSLHangNhap(DateHelper.tinhNgayDauTuan(1), DateHelper.toString(DateHelper.now(), "yyyy-MM-dd"));

lbl\_TongSoLuong.setText("Tổng số lượng: " + SLNhap);

lbl\_TongGiaTri.setText("Tổng giá trị (bao gồm VAT & chiết khấu): " + dcf.format(GTNhap));

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

* + - 1. **BaoCaoTaiChinhJFrame**

ThongKeDAO TKDAO = new ThongKeDAO();

DecimalFormat dcf = new DecimalFormat("###,###,###.###");

/\*

\* Hàm khởi tạo các giá trị ban đầu cho JFrame

\*/

void inIt() {

// start code setSize cố định không cho co giãn lung tung:

this.setSize(1366,768);

this.setMinimumSize(new Dimension(1366, 768));

this.setMaximumSize(new Dimension(1366, 768));

// end code setSize

setResizable(false);

setLocationRelativeTo(null);

bcKetQuaKDThang();

}

/\*

\* Hàm báo cáo kết quả kinh doanh tháng

\*/

void bcKetQuaKDThang() {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String Thang = NgayHienTai.substring(3, 5);

String Nam = NgayHienTai.substring(6, 10);

String NgayDauThang = Nam + "-" + Thang + "-01";

String NgayCuoiThang = "";

switch (Thang) {

case "01":

case "03":

case "05":

case "07":

case "08":

case "10":

case "12":

NgayCuoiThang = Nam + "-" + Thang + "-31";

break;

case "04":

case "06":

case "09":

case "11":

NgayCuoiThang = Nam + "-" + Thang + "-30";

break;

case "02":

if (DateHelper.xetNamNhuan(Integer.parseInt(Nam))) {

NgayCuoiThang = Nam + "-" + Thang + "-29";

}

else

NgayCuoiThang = Nam + "-" + Thang + "-28";

break;

}

lbl\_NgayThongKe.setText("Thống kê tháng: " + Thang + "-" + Nam);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

try {

double DoanhThu = TKDAO.tinhDoanhThuTrongThang();

double GiaVonHangBan = TKDAO.tinhGiaVonHangBan(NgayDauThang, NgayCuoiThang);

double LoiNhuanGop = DoanhThu - GiaVonHangBan;

tbl\_BangBaoCao.setValueAt(dcf.format(DoanhThu), 0, 1);

tbl\_BangBaoCao.setValueAt(0, 2, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(DoanhThu), 3, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(GiaVonHangBan), 4, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(LoiNhuanGop), 5, 1);

tbl\_BangBaoCao.setValueAt(0, 6, 1);

tbl\_BangBaoCao.setValueAt(0, 7, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(LoiNhuanGop), 8, 1);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo kết quả kinh doanh năm

\*/

void bcKetQuaKDNam() {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

String Nam = NgayHienTai.substring(6, 10);

String NgayDauNam = Nam + "-01-01";

String NgayCuoiNam = Nam + "-12-31";

lbl\_NgayThongKe.setText("Thống kê năm: " + Nam);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

try {

double DoanhThu = TKDAO.tinhDoanhThuTrongNam();

double GiaVonHangBan = TKDAO.tinhGiaVonHangBan(NgayDauNam, NgayCuoiNam);

double LoiNhuanGop = DoanhThu - GiaVonHangBan;

tbl\_BangBaoCao.setValueAt(dcf.format(DoanhThu), 0, 1);

tbl\_BangBaoCao.setValueAt(0, 2, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(DoanhThu), 3, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(GiaVonHangBan), 4, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(LoiNhuanGop), 5, 1);

tbl\_BangBaoCao.setValueAt(0, 6, 1);

tbl\_BangBaoCao.setValueAt(0, 7, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(LoiNhuanGop), 8, 1);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

/\*

\* Hàm báo cáo kết quả kinh hôm nay

\*/

void bcKetQuaKDNgay() {

String NgayHienTai = DateHelper.toString(DateHelper.now(), "dd-MM-yyyy");

lbl\_NgayThongKe.setText("Thống kê ngày: " + NgayHienTai);

lbl\_NgayTao.setText("Ngày tạo: " + NgayHienTai);

try {

double DoanhThu = TKDAO.tinhDoanhThuTrongNgay();

double GiaVonHangBan = TKDAO.tinhGiaVonHangBan(DateHelper.toString(DateHelper.now(), "yyyy-MM-dd"), DateHelper.toString(DateHelper.now(), "yyyy-MM-dd"));

double LoiNhuanGop = DoanhThu - GiaVonHangBan;

tbl\_BangBaoCao.setValueAt(dcf.format(DoanhThu), 0, 1);

tbl\_BangBaoCao.setValueAt(0, 2, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(DoanhThu), 3, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(GiaVonHangBan), 4, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(LoiNhuanGop), 5, 1);

tbl\_BangBaoCao.setValueAt(0, 6, 1);

tbl\_BangBaoCao.setValueAt(0, 7, 1);

tbl\_BangBaoCao.setValueAt(dcf.format(LoiNhuanGop), 8, 1);

} catch (Exception e) {

DialogHelper.alert(this, "Có lỗi: " + e.toString());

}

}

}

# KIỂM THỬ

## BanHangJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Xuất Hoá Đơn** |
| Tên khách hàng | Không được bỏ trống |
| Tiền khách trả | Không được bỏ trống |
| Danh sách mua | Không được rỗng |

## ThemMoiKhachHangJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Họ tên | Chỉ chưa alphabet, nguyên âm có dấu và ký tự trắng |
| Điện thoại | Chỉ chứa số từ 10 đến 11 ký tự  Không được trùng với khách hàng khác |
| Ngày sinh | Đủ 13 tuổi trở lên |
| Email | Đúng định dạng, không được trùng khách hàng khác |
| Địa chỉ | Không được bỏ trống |

## CapNhatKhachHangJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Cập nhật** |
| Họ tên | Chỉ chưa alphabet, nguyên âm có dấu và ký tự trắng |
| Điện thoại | Chỉ chứa số từ 10 đến 11 ký tự  Không được trùng khách hàng khác |
| Ngày sinh | Đủ 13 tuổi trở lên |
| Email | Đúng định dạng, không được trùng khách hàng khác |
| Địa chỉ | Không được bỏ trống |
| Điểm thưởng | Chỉ được nhập số |

## ThemMoiNhaCungCapJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Tên NCC | Chỉ chưa alphabet, nguyên âm có dấu và ký tự trắng |
| Điện thoại | Chỉ chứa số từ 10 đến 11 ký tự  Không được trùng NCC khác |
| Email | Đúng định dạng, không được trùng NCC khác |
| Địa chỉ | Không được bỏ trống |

## CapNhatNhaCungCapJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Cập nhật** |
| Tên NCC | Chỉ chưa alphabet, nguyên âm có dấu và ký tự trắng |
| Điện thoại | Chỉ chứa số từ 10 đến 11 ký tự  Không được trùng NCC khác |
| Email | Đúng định dạng, không được trùng NCC khác |
| Địa chỉ | Không được bỏ trống |

## ThemMoiNhanVienJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Tên đăng nhập | Không được trùng nhân viên khác  Không chứa ký tự trắng, số ký tự trong khoảng 3 đến 20 |
| Họ tên | Chỉ chứa alphabet, nguyên âm có dấu và khoảng trắng |
| Điện thoại | Chỉ chứa số từ 10 đến 11 ký tự  Không được trùng nhân viên khác |
| Ngày sinh | Không dưới 18 tuổi |
| Email | Đúng định dạng, không được trùng nhân viên khác |
| Địa chỉ | Không được bỏ trống |
| Mật khẩu | Không chứa khoảng trắng |
| Nhập lại mật khẩu | Đúng chuỗi đã nhập ở ô Mật khẩu |

## CapNhatNhanVienJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Cập nhật** |
| Tên đăng nhập | Không được trùng nhân viên khác  Không chứa ký tự trắng  Số ký tự trong khoảng 3 đến 20 |
| Họ tên | Chỉ chứa alphabet, nguyên âm có dấu và khoảng trắng |
| Điện thoại | Chỉ chứa số từ 10 đến 11 ký tự  Không được trùng nhân viên khác |
| Ngày sinh | Không dưới 18 tuổi |
| Email | Đúng định dạng, không được trùng nhân viên khác |
| Địa chỉ | Không được bỏ trống |
| Mật khẩu | Không chứa khoảng trắng |
| Nhập lại mật khẩu | Đúng chuỗi đã nhập ở ô Mật khẩu |

## ThemMoiSanPhamJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Tên sản phẩm | Không được để trống, tối đa 50 ký tự  Không trùng tên sản phẩm khác |
| Mã vạch | Chỉ chứa số, không trùng với sản phẩm khác |
| Định mức tồn lớn nhất | Chỉ chứa số |
| Định mức tồn nhỏ nhất | Chỉ chứa số |
| Giá vốn | Chỉ chứa số, tối đa 9 ký tự |
| Giá bán | Chỉ chứa số, tối đa 9 ký tự |

## CapNhatSanPhamJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Cập nhật** |
| Tên sản phẩm | Không được để trống, tối đa 50 ký tự  Không trùng tên sản phẩm khác |
| Mã vạch | Chỉ chứa số, không trùng với sản phẩm khác |
| Định mức tồn lớn nhất | Chỉ chứa số |
| Định mức tồn nhỏ nhất | Chỉ chứa số |
| Giá bán | Chỉ chứa số, tối đa 9 ký tự |
| Hạn sử dụng mới nhất | Không được trước hạn sử dụng gần nhất |

## ThemMoiPhieuKiemJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Danh sách sản phẩm kiểm kê | Không được để trống |
| Ghi chú | Không được để trống |

## CapNhatPhieuKiemJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Cập nhật** |
| Danh sách sản phẩm kiểm kê | Không được để trống |
| Ghi chú | Không được để trống |

## ThemMoiPhieuNhapJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Danh sách sản phẩm nhập | Không được để trống  Phải điền số lượng nhập và giá nhập |
| Ghi chú | Không được để trống |

## ThemMoiPhieuHuyJFrame

|  |  |
| --- | --- |
| **Thuộc tính** | **Thao tác** |
| **Thêm mới** |
| Danh sách sản phẩm cần huỷ | Không được để trống  Phải điền số lượng huỷ |
| Ghi chú | Không được để trống |

# ĐÓNG GÓI VÀ TRIỂN KHAI

## Hướng dẫn chuyển đổi jar thành exe

* Sử dụng phần mềm EXE4J để chuyển đổi jar sang exe
* Sử dụng phần mềm InnoSetup để đóng gói

## Hướng dẫn cài đặt triển khai

* Cài SQL Server 2008 trở lên
* Tạo CSDL quanlysieuthimini bằng cách chạy file quanlysieuthimini.sql
* Cài đặt JDK 1.8 trở lên
* Chạy file setup

## Hướng dẫn sử dụng phần mềm

* Đăng nhập với tài khoản **admin** và mật khẩu là ***123***
* Cấp các tài khoản mới có vai trò tương ứng (nhân viên/thủ kho)

♦♦♦

**LỜI CẢM ƠN**

Trong quãng thời gian hơn một tháng qua, chúng em đã nhận được sự hỗ trợ tận tình của thầy **ThS. Lê Anh Tú –** giảng viên đứng lớp môn *“Dự án 1” –* để chúng em hoàn thiện phần mềm này.

Chúng em xin gởi lời cảm ơn đến thầy và chúc thầy nhiều sức khoẻ để tiếp tục hướng dẫn, hỗ trợ cho sinh viên Fpoly đạt được kết quả học tập tốt nhất.

Trân trọng

*Nhóm thực hiện đề tài*