**ỦY BAN NHÂN DÂN TP.HCM**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO XÂY DỰNG MÔ HÌNH PHÂN LỚP**

**QUẢN LÝ THÀNH VIÊN**

**HỌ VÀ TÊN THÀNH VIÊN NHÓM 2:**

Lâm Hào Kiệt 3121410286

Trần Xuân Danh 3121410104

Trần Khánh Duy 3121410124

Nguyễn Hoàng Kha 3120410228

Lê Trung Kiên 3121410284

Lê Hoài Nam 3121410330

Huỳnh Thanh Lộc 3121410301

**Giảng viên hướng dẫn: Thầy Cao Minh Thành và Thầy Nguyễn Minh Cảnh**

**TP.HỒ CHÍ MINH, THÁNG 4 NĂM 2024**

**MỤC LỤC**

[**BẢNG PHÂN CÔNG 2**](#_Toc26078)

[**Chức năng 1: Quản lý thành viên 3**](#_Toc17368)

[Xử lý 1: Hiển thị danh sách thành viên 3](#_Toc15708)

[Xử lý 2: Thêm thành viên : 4](#_Toc6121)

Xử lý 2.1: Nhập thông tin 1 thành viên................................................................4

Xử lý 2.2: Nhập theo file Excel...........................................................................9

[Xử lý 3: Xóa Thành viên 12](#_Toc68)

[Xử lý 4: Sửa Thông Tin Thành viên 18](#_Toc23318)

[Xử lý 5: Vào khu vực học tập 20](#_Toc5793)

[Xử lý 6: Mượn thiết bị 22](#_Toc30765)

[Xử lý 7: Trả thiết bị 25](#_Toc28268)

[Xử lý 8: Tìm kiếm thành viên 28](#_Toc29881)

[Xử lý 9: Hiển thị chi tiết 30](#_Toc14761)

[**Chức năng 2: Quản lý thiết bị 35**](#_Toc27263)

[Xử lý 1: Hiển thị danh sách thiết bị 35](#_Toc886)

[Xử lý 2: Thêm thiết bị 37](#_Toc30424)

[Xử lý 2.1. Nhập thông tin 1 thiết bị: 37](#_Toc24527)

[Xử lý 2.2. Nhập từ file excel : 39](#_Toc8647)

[Xử lý 3: Sửa thông tin thiết bị 43](#_Toc21178)

[Xử lý 4: Xóa thiết bị 47](#_Toc21250)

[Xử lý 5: Tìm kiếm thiết bị 51](#_Toc29108)

[Xử lý 6: Hiển thị chi tiết thông tin thiết bị 56](#_Toc27691)

[**Chức năng 3: Thống kê 60**](#_Toc6542)

[Xử lý 1: Thống kê thành viên 60](#_Toc6589)

[Xử lý 2: Thống kê sử dụng thiết bị 70](#_Toc9242)

[Xử lý 3: Thống kê vi phạm 78](#_Toc14461)

[**Chức năng 4: Xử lý vi phạm 81**](#_Toc9301)

[Xử lý 1: Hiển thị danh sách vi phạm 81](#_Toc23415)

[Xử lý 2: Thêm vi phạm 83](#_Toc9472)

[Xử lý 3: Xóa vi phạm 85](#_Toc7880)

[Xử lý 4: Chỉnh sửa vi phạm 87](#_Toc1209)

[Xử lý 5: Tìm kiếm vi phạm 89](#_Toc5567)

[Xử lý 6: Cập nhật trạng thái của vi phạm 91](#_Toc9230)

**BẢNG PHÂN CÔNG**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ Tên** | **Khối lượng công việc** | **Phần trăm công việc** |
| 1 | 3121410286 | Lâm Hào Kiệt | Thành viên | 16% |
| 2 | 3121410330 | Lê Hoài Nam | Thiết bị | 16% |
| 3 | 3121410284 | Lê Trung Kiên | Thiết bị | 16% |
| 4 | 3121410104 | Trần Xuân Danh | Thống kê | 16% |
| 5 | 3121410124 | Trần Khánh Duy | Thống kê | 16% |
| 6 | 3121410301 | Huỳnh Thanh Lộc | Xử lý vi phạm | 16% |
| 7 | 3120410228 | Nguyễn Hoàng Kha | Word | 4% |

**Chức năng 1: Quản lý thành viên**

**Xử lý 1: Hiển thị danh sách thành viên**

* Sequence diagram

A diagram of a line

Description automatically generated

* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL :**

public ArrayList<ThanhVienDTO> listThanhVien() {

        Session session = factory.openSession();

        ArrayList<ThanhVienDTO> List = new ArrayList<>();

        Transaction tx = null;

         try{

             tx = session.beginTransaction();

             List employess = session.createQuery("FROM ThanhVienDTO").list();

             for (Iterator iterator = employess.iterator(); iterator.hasNext();) {

                 ThanhVienDTO tv = (ThanhVienDTO) iterator.next();

                 List.add(tv);

             }

         } catch (HibernateException e){

             if (tx != null) tx.rollback();

             e.printStackTrace();

         }

         return List;

    }

* **BLL :**

public ArrayList<ThanhVienDTO> listThanhVien(){

        return tvDAL.listThanhVien();

    }

* **GUI :**

public void loadThanhVien(){

        arrThanhVien = tvBLL.listThanhVien();

        for(int i = modelTV.getRowCount()-1;i>=0;i--)

            modelTV.removeRow(i);

        for(int i = 0; i<arrThanhVien.size();i++){

            ThanhVienDTO em= arrThanhVien.get(i);

            int id= em.getMaTV();

            String name = em.getHoTen();

            String khoa = em.getKhoa();

            String nganh = em.getNganh();

            String sdt = em.getSDT();

            Object[] row = {id,name,khoa,nganh,sdt};

            modelTV.addRow(row);

        }

    }

**Xử lý 2: Thêm thành viên :**

***Xử lý 2.1: Nhập thông tin 1 thành viên :***

* Sequence diagram

A white sheet of paper with black lines

Description automatically generated

* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL :**

public void ThemThanhVien(ThanhVienDTO tv) {

        Session session = factory.openSession();

        Transaction tx = null;

        Integer tvienID = null;

        try {

            tx = session.beginTransaction();

             // Sử dụng Criteria để lấy số lượng bản ghi trong bảng ThanhVien

            String hql = "SELECT COUNT(\*) FROM ThongTinSuDungDTO";

            Query<Long> query = session.createQuery(hql, Long.class);

            Long count = query.uniqueResult();

            String id = "";

            String year = Year.now().toString().substring(2);

            String khoa = tv.getKhoa();

            String nganh = tv.getNganh();

            String khoaCode= "";

            switch (khoa.toUpperCase(Locale.ROOT)) {

                case "SP KHXH":

                    switch (nganh.toUpperCase(Locale.ROOT)) {

                        case "ĐỊA":

                            khoaCode = "11";

                            break;

                        case "SỬ":

                            khoaCode = "10";

                            break;

                        case "VĂN":

                            khoaCode = "09";

                            break;

                        default:

                            khoaCode = "00";

                            break;

                    }

                    break;

                case "SP KHTN":

                    switch (nganh.toUpperCase(Locale.ROOT)) {

                        case "LÍ":

                            khoaCode = "02";

                            break;

                        case "HÓA":

                            khoaCode = "03";

                            break;

                        case "SINH":

                            khoaCode = "04";

                            break;

                        default:

                            khoaCode = "00";

                            break;

                    }

                    break;

                case "NGOẠI NGỮ":

                    switch (nganh.toUpperCase(Locale.ROOT)) {

                        case "ANH":

                            khoaCode = "13";

                            break;

                        case "NNA":

                            khoaCode = "38";

                            break;

                        default:

                            khoaCode = "00";

                            break;

                    }

                    break;

                case "QTKD":

                    khoaCode = "55";

                    break;

                case "QLGD":

                    switch (nganh.toUpperCase(Locale.ROOT)) {

                        case "TLH":

                            khoaCode = "53";

                            break;

                        default:

                            khoaCode = "00";

                            break;

                    }

                    break;

                case "TOÁN UD":

                    switch (nganh.toUpperCase(Locale.ROOT)) {

                        case "TOÁN":

                            khoaCode = "48";

                            break;

                        default:

                            khoaCode = "00";

                            break;

                    }

                    break;

                case "CNTT":

                    switch (nganh.toUpperCase(Locale.ROOT)) {

                        case "CNTT":

                            khoaCode = "41";

                            break;

                        case "KTPM":

                            khoaCode = "42";

                            break;

                        case "HTTT":

                            khoaCode = "43";

                            break;

                        default:

                            khoaCode = "00";

                            break;

                    }

                    break;

                default:

                    khoaCode = "00";

                    break;

            }

            id = "11"+ year + khoaCode + String.format("%04d", count + 1);

            int idCode= Integer.parseInt(id);

            ThanhVienDTO tvien = new ThanhVienDTO(idCode,tv.getHoTen(), tv.getKhoa(), tv.getNganh(), tv.getSDT());

            session.save(tvien);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL :**

public void ThemThanhVien(ThanhVienDTO tv){

        tvDAL.ThemThanhVien(tv);

    }

* **GUI :**

private void jButton\_CreateActionPerformed(java.awt.event.ActionEvent evt) {

        System.out.println("---------------------------------ID:"+id);

        String Name = jTextField\_Name.getText();

        String Khoa = jComboBox\_Khoa.getSelectedItem().toString();

        String Nganh = jComboBox\_Nganh.getSelectedItem().toString();

        String SDT = jTextField\_SDT.getText();

        if (!"".equals(Name) && !"".equals(Khoa) && !"".equals(Nganh) && !"".equals(SDT)) {

            ThanhVienDTO sv = new ThanhVienDTO();

//            sv.setPersonID(id);

            sv.setMaTV(id);

            sv.setHoTen(Name);

            sv.setKhoa(Khoa);

            sv.setNganh(Nganh);

            sv.setSDT(SDT);

            tvBLL.ThemThanhVien(sv);

            JOptionPane.showMessageDialog(null, "Thêm thành công !!!", "Thêm sinh viên", JOptionPane.INFORMATION\_MESSAGE);

//            svGUI.Refresh();

            dispose();

        } else {

            JOptionPane.showMessageDialog(null, "Dữ liệu chưa được nhập!!", "Thông tin lỗi", JOptionPane.INFORMATION\_MESSAGE);

        }

    }

***Xử lý 2.2: Nhập theo file Excel :***

* Sequence diagram

A diagram with text and lines

Description automatically generated with medium confidence

* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL :**

public void ThemThanhVienExcel(String path) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            String hql = "SELECT COUNT(\*) FROM ThongTinSuDungDTO";

            Query<Long> query = session.createQuery(hql, Long.class);

            Long count = query.uniqueResult();

            FileInputStream inputStream = new FileInputStream(new File(path));

            Workbook workbook = WorkbookFactory.create(inputStream);

            Sheet sheet = workbook.getSheetAt(0); // Lấy sheet đầu tiên

            // Bắt đầu đọc từ hàng thứ hai (index 1)

            for (int i = 1; i <= sheet.getLastRowNum(); i++) {

                Row row = sheet.getRow(i);

                if (row != null) {

                    Cell idCell =row.getCell(0);

                    Cell nameCell = row.getCell(1);

                    Cell khoaCell = row.getCell(2);

                    Cell nganhCell = row.getCell(3);

                    Cell sdtCell = row.getCell(4);

                    Cell passCell = row.getCell(5);

                    Cell emailCell = row.getCell(6);

                    // Kiểm tra và lưu dữ liệu vào Hibernate

                    if (idCell != null && nameCell != null && khoaCell != null && nganhCell != null && sdtCell != null) {

                        int idCode=0;

                        if (idCell.getCellType() == CellType.NUMERIC) {

                            idCode = (int) idCell.getNumericCellValue();

                        } else if (idCell.getCellType() == CellType.STRING) {

                            String id = idCell.getStringCellValue();

                            idCode = Integer.parseInt(id);

                        }

                        String name = nameCell.getStringCellValue();

                        String khoa = khoaCell.getStringCellValue();

                        String nganh = nganhCell.getStringCellValue();

                        String sdt =  sdtCell.getStringCellValue();

                        String pass = passCell.getStringCellValue();

                        String email = emailCell.getStringCellValue();

                        // Tạo đối tượng ThanhVienDTO từ dữ liệu

                        ThanhVienDTO thanhVien = new ThanhVienDTO(idCode, name, khoa, nganh, sdt, pass,email);

                        // Lưu đối tượng vào cơ sở dữ liệu bằng Hibernate

                        session.save(thanhVien);

                        // Cập nhật giá trị của count

                        count++;

                    }

                }

            }

            // Commit transaction và đóng session

            tx.commit();

            session.close();

            // Đóng workbook và inputStream

            workbook.close();

            inputStream.close();

        } catch (IOException e) {

            e.printStackTrace();

        } catch (HibernateException ex) {

            if (tx != null) tx.rollback();

            ex.printStackTrace();

        }

    }

* **BLL :**

public void ThemThanhVienExcel(String path){

        tvDAL.ThemThanhVienExcel(path);

    }

* **GUI :**

private void jButton\_FileActionPerformed(java.awt.event.ActionEvent evt) {

        JFileChooser fileChooser = new JFileChooser();

        FileNameExtensionFilter filter = new FileNameExtensionFilter("Excel files", "xlsx", "xls");

        fileChooser.setFileFilter(filter);

        // Hiển thị hộp thoại và kiểm tra xem người dùng có chọn file không

        int returnValue = fileChooser.showOpenDialog(null);

        if (returnValue == JFileChooser.APPROVE\_OPTION) {

            // Lấy đường dẫn của file đã chọn

            File selectedFile = fileChooser.getSelectedFile();

            String filePath = selectedFile.getAbsolutePath();

            // Gọi phương thức để thêm thành viên từ file Excel

            tvBLL.ThemThanhVienExcel(filePath);

            JOptionPane.showMessageDialog(null, "Thêm thành công !!!", "Thêm thành viên", JOptionPane.INFORMATION\_MESSAGE);

            dispose();

        }

        else {

            JOptionPane.showMessageDialog(null, "Thêm thất bại !!!", "Thêm thành viên", JOptionPane.INFORMATION\_MESSAGE);

        }

    }

**Xử lý 3: Xóa Thành viên**

* Sequence diagram

A white sheet of paper with black lines

Description automatically generated

* Giao diện

+Xóa 1 thành viên:

A blue rectangle with black text

Description automatically generated

+Xóa theo khóa:

A screenshot of a computer

Description automatically generated

* Code
* **DAL:**

+ Xóa 1 thành viên :

public void deleteThanhVien(int tvID) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            ThanhVienDTO tbi = (ThanhVienDTO) session.get(ThanhVienDTO.class, tvID);

            session.delete(tbi);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

+ Xóa theo khóa:

public ArrayList<ThanhVienDTO> XoaTheoNam(int twoDigits) {

        Session session = factory.openSession();

        Transaction tx = null;

        ArrayList<ThanhVienDTO> danhSachCanXoa = new ArrayList<>();

        try {

            tx = session.beginTransaction();

            int thirdDigit = twoDigits / 10;

            int fourthDigit = twoDigits % 10;

            String thirdDigitStr = String.valueOf(thirdDigit);

            String fourthDigitStr = String.valueOf(fourthDigit);

            // Tạo câu lệnh HQL để truy vấn

            String hql = "FROM ThanhVienDTO WHERE SUBSTRING(MaTV, 3, 1) = :thirdDigit AND SUBSTRING(MaTV, 4, 1) = :fourthDigit";

            Query<ThanhVienDTO> query = session.createQuery(hql);

            query.setParameter("thirdDigit", thirdDigitStr);

            query.setParameter("fourthDigit", fourthDigitStr);

            danhSachCanXoa = (ArrayList<ThanhVienDTO>) query.getResultList();

            // Duyệt qua danh sách các đối tượng và kiểm tra dính khóa ngoại trước khi xóa

            // Kết thúc giao dịch

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) {

                tx.rollback();

            }

            e.printStackTrace();

        } finally {

            session.close();

        }

        return danhSachCanXoa;

    }

* **BLL:**

    public boolean CheckMuon(int ID){

        ArrayList<ThongTinSuDungDTO> list = new ThongTinSuDungDAL().listThongTinSuDung();

        for (ThongTinSuDungDTO s : list){

            if (s.getMaTV() != null){

                if (s.getMaTV().getMaTV() == ID){

                    return false;

                }

            }

        }

        return true;

    }

     public boolean checkVP(int id){

        ArrayList<XuLyViPhamDTO> arr = xlDAL.listXuLyViPham();

        for(XuLyViPhamDTO a:arr){

            if(a.getMaTV().getMaTV()==id && a.getTrangThaiXL()==0){

                return false;

            }

        }

        return true;

    }

+Xóa 1 thành viên:

public boolean deleteThanhVien(int id){

        if(CheckMuon(id) && checkVP(id)) {

            tvDAL.deleteThanhVien(id);

            return true;

        }

        else return false;

    }

+Xóa theo khóa:

public String XoaTheoKhoa(int khoa){

        ArrayList<ThanhVienDTO> list= tvDAL.XoaTheoNam(khoa);

        ArrayList<String> listtb = new ArrayList<>();

        for(ThanhVienDTO tvien : list){

            if(CheckMuon(tvien.getMaTV()) && checkVP(tvien.getMaTV())){

                tvDAL.deleteThanhVien(tvien.getMaTV());

            }

            else{

                listtb.add(tvien.getMaTV() + " " +  tvien.getHoTen());

            }

        }

        if (listtb.size() != 0){

            String result = "Thành viên đang mượn thiết bị hoặc đang bị xử lý : ";

            for (String i : listtb){

                result += "\n" + i;

            }

            return result;

        }

        else if(listtb.size() == 0){

            return "Không có thành viên để xóa!!!";

        }

        return "Xóa thất bại";

    }

* **GUI:**

+Xóa 1 thành viên:

private void jButton\_DeleteActionPerformed(java.awt.event.ActionEvent evt) {

        int selectedRow = jTable\_ThanhVien.getSelectedRow();

        if (selectedRow == -1) {

            XoaNhieuGUI atv = new XoaNhieuGUI();

            atv.setVisible(true);

        } else {

            id = (int) jTable\_ThanhVien.getValueAt(selectedRow, 0);

            if(tvBLL.deleteThanhVien(id)){

                JOptionPane.showMessageDialog(this, "Xóa thành công!!", "Thông báo", JOptionPane.WARNING\_MESSAGE);

                loadThanhVien();

            }

            else{

                JOptionPane.showMessageDialog(this, "Thành Viên hiện đang mượn thiết bị hoặc đang bị xử lý nên không thể xóa", "Thông báo", JOptionPane.WARNING\_MESSAGE);

            }

        }

    }

+Xóa theo khóa:

private void jButton\_DeleteActionPerformed(java.awt.event.ActionEvent evt) {

        if(check()){

            String Name = jTextField\_Khoa.getText();

            int so= Integer.parseInt(Name);

            JOptionPane.showMessageDialog(this, tvBLL.XoaTheoKhoa(so));

            dispose();

        }

        else{

            JOptionPane.showMessageDialog(null, "Thông tin phải dạng số !!!", "Xóa thành viên", JOptionPane.INFORMATION\_MESSAGE);

        }

    }

**Xử lý 4: Sửa Thông Tin Thành viên**

* Sequence diagram

A white sheet of paper with black lines

Description automatically generated

* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL:**

public void updateThanhVien(ThanhVienDTO tv) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            ThanhVienDTO tvien = (ThanhVienDTO) session.get(ThanhVienDTO.class, tv.getMaTV());

            tvien.setHoTen(tv.getHoTen());

            tvien.setKhoa(tv.getKhoa());

            tvien.setNganh(tv.getNganh());

            tvien.setSDT(tv.getSDT());

            session.update(tvien);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL:**

public void updateThanhVien(ThanhVienDTO tv){

        tvDAL.updateThanhVien(tv);

    }

* **GUI:**

private void jButton\_UpdateActionPerformed(java.awt.event.ActionEvent evt) {

        if (check()){

            String name = jTextField\_Name.getText().toString();

            String khoa = jComboBox\_Khoa.getSelectedItem().toString();

            String nganh = jComboBox\_Khoa1.getSelectedItem().toString();

            String sdt = jTextField\_SDT.getText();

            ThanhVienDTO tv= new ThanhVienDTO(pid,name,khoa,nganh,sdt);

            tvBLL.updateThanhVien(tv);

            tvGUI.load();

            JOptionPane.showMessageDialog(null, "Cập nhật thành công !!!", "Cập nhật thông tin sinh viên", JOptionPane.INFORMATION\_MESSAGE);

            dispose();

        }

        else {

            JOptionPane.showMessageDialog(null, "Thông tin không có thay đổi hoặc bị rỗng !!!", "ERROR !!!", JOptionPane.INFORMATION\_MESSAGE);

        }

    }

**Xử lý 5: Vào khu vực học tập**

* Sequence diagram

A white sheet of paper with black text

Description automatically generated

* Giao diện

A screenshot of a video chat

Description automatically generated

* Code
* **DAL:**

public void VaoKhuVucHocTap(int id){

        Session session = factory.openSession();

        Transaction tx=null;

        try {

            tx = session.beginTransaction();

            String hql = "SELECT COUNT(\*) FROM ThongTinSuDungDTO";

            Query<Long> query = session.createQuery(hql, Long.class);

            Long count = query.uniqueResult();

            ThanhVienDTO idtv = tvDAL.getThanhVienById(id);

            int idTT = count.intValue() + 1;

            Date date = new Date();

//            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

//            String TGVao = sdf.format(date);

            ThongTinSuDungDTO tt= new ThongTinSuDungDTO(idTT,idtv,date);

            session.save(tt);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL:**

public void VaoKhuVucHocTap(int id){

        ttDAL.VaoKhuVucHocTap(id);

    }

* **GUI:**

private void jButton\_VaoKhuVucActionPerformed(java.awt.event.ActionEvent evt) {

        int selectedRow = jTable\_ThanhVien.getSelectedRow();

        if (selectedRow == -1) {

            JOptionPane.showMessageDialog(this, "Vui lòng chọn thành viên vào khu vực học tập!!!!", "Thông báo", JOptionPane.WARNING\_MESSAGE);

        } else{

            id = (int) jTable\_ThanhVien.getValueAt(selectedRow, 0);

            if(xlBLL.checkVP(id)){

                ttBLL.VaoKhuVucHocTap(id);

                JOptionPane.showMessageDialog(this, "Thành viên đã checkin thành công!!", "Thông báo", JOptionPane.WARNING\_MESSAGE);

            }

            else{

                JOptionPane.showMessageDialog(this, "Thành viên đang thuộc trong biên chế xử lý . KHÔNG THỂ VÀO KHU VỰC HỌC TẬP!!!", "Thông báo", JOptionPane.WARNING\_MESSAGE);

            }

        }

    }

**Xử lý 6: Mượn thiết bị**

* Sequence diagram

A screenshot of a computer

Description automatically generated

* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL:**

public void MuonThietBi(ThongTinSuDungDTO tt){

        Session session = factory.openSession();

        Transaction tx=null;

        try {

            tx = session.beginTransaction();

            String hql = "SELECT COUNT(\*) FROM ThongTinSuDungDTO";

            Query<Long> query = session.createQuery(hql, Long.class);

            Long count = query.uniqueResult();

            int idTT = count.intValue() + 1;

            ThongTinSuDungDTO Thongtin= new ThongTinSuDungDTO(idTT,tt.getMaTV(),tt.getMaTB(),tt.getTGMuon(),tt.getTGTra());

            session.save(Thongtin);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL:**

public void MuonThietBi(ThongTinSuDungDTO tt){

        ttDAL.MuonThietBi(tt);

    }

* **GUI:**

public boolean checkDate() {

        Date muonDate = jDateChooser\_tgmuon.getDate();

        Date traDate = jDateChooser\_tgtra.getDate();

        return muonDate.before(traDate);

    }

    public boolean checkInit(){

        if(jComboBox\_ThanhVien.getSelectedItem()!=null && jComboBox\_ThietBi.getSelectedItem()!=null && jDateChooser\_tgmuon.getDate()!=null && jDateChooser\_tgtra.getDate()!=null){

            return true;

        }

        return false;

    }

    public boolean checkThietBi(int ID){

        for (ThongTinSuDungDTO s : list){

            if ( s.getMaTB() != null){

                if (s.getMaTB().getMaTB() == ID){

                    return false;

                }

            }

        }

        return true;

    }

**Xử lý 7: Trả thiết bị**

* Sequence diagram

A diagram of a diagram

Description automatically generated with medium confidence

* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL:**

public void xoaThongTinSuDung(int id) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            ThongTinSuDungDTO thongTin = (ThongTinSuDungDTO) session.get(ThongTinSuDungDTO.class, id);

            session.delete(thongTin);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL:**

public void xoaThongTinSuDung(int id) {

         ttDAL.xoaThongTinSuDung(id);

     }

* **GUI:**

private void jButton\_TraActionPerformed(java.awt.event.ActionEvent evt) {

        int selectedRow = jTable\_Muon.getSelectedRow();

        if(selectedRow == -1){

            JOptionPane.showMessageDialog(this, "Vui lòng chọn 1 dòng để trả thiết bị", "Thông báo", JOptionPane.WARNING\_MESSAGE);

        }

        else{

            int id = (int) jTable\_Muon.getValueAt(selectedRow, 0);

            ttBLL.xoaThongTinSuDung(id);

            JOptionPane.showMessageDialog(this, "Trả thành công!!", "Thông báo", JOptionPane.WARNING\_MESSAGE);

            dispose();

        }

    }

**Xử lý 8: Tìm kiếm thành viên**

* Sequence diagram A diagram of a diagram

  Description automatically generated
* Giao diện

A screenshot of a computer

Description automatically generated

* Code
* **DAL:**

public ArrayList<ThanhVienDTO> search(String s) {

        ArrayList<ThanhVienDTO> list = new ArrayList<>();

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            String hql = "FROM ThanhVienDTO WHERE CAST(MaTV AS string) LIKE :keyword OR HoTen LIKE :keyword OR Khoa LIKE :keyword OR Nganh LIKE :keyword OR SDT LIKE :keyword";

            Query<ThanhVienDTO> query = session.createQuery(hql, ThanhVienDTO.class);

            query.setParameter("keyword", "%" + s + "%");

            list = (ArrayList<ThanhVienDTO>) query.getResultList();

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) {

                tx.rollback();

            }

            e.printStackTrace();

        } finally {

            session.close();

        }

        return list;

    }

* **BLL:**

public ArrayList<ThanhVienDTO> search(String s){

        return tvDAL.search(s);

    }

* **GUI:**

private void jTextField\_SearchKeyReleased(java.awt.event.KeyEvent evt) {//GEN-FIRST:event\_jTextField\_SearchKeyReleased

        // TODO add your handling code here:

        String searchText = jTextField\_Search.getText().trim();

        // Gọi hàm search với nội dung tìm kiếm

        ArrayList<ThanhVienDTO> searchResult = tvBLL.search(searchText);

        for(int i = modelTV.getRowCount()-1;i>=0;i--)

        modelTV.removeRow(i);

        for(int i = 0; i<searchResult.size();i++){

            ThanhVienDTO em= searchResult.get(i);

            int id= em.getMaTV();

            String name= em.getHoTen();

            String khoa = em.getKhoa();

            String nganh = em.getNganh();

            String sdt = em.getSDT();

            Object[] row = {id,name,khoa,nganh,sdt};

            modelTV.addRow(row);

        }

    }

**Xử lý 9: Hiển thị chi tiết**

* Sequence diagram

A white sheet of paper with black text

Description automatically generated

* Giao diện

A screenshot of a chat

Description automatically generated

* Code
* **BLL:**

public ArrayList<ThongTinSuDungDTO> getMuon(int id) {

        ArrayList<ThongTinSuDungDTO> list = ttDAL.listThongTinSuDung();

        ArrayList<ThongTinSuDungDTO> result = new ArrayList<>();

        for (ThongTinSuDungDTO ttsd : list) {

            if (ttsd.getMaTB() != null && ttsd.getMaTV().getMaTV() == id) {

                result.add(ttsd);

            }

        }

        return result;

    }

    public ArrayList<ThongTinSuDungDTO> getKhuVuc(int id) {

        ArrayList<ThongTinSuDungDTO> list = ttDAL.listThongTinSuDung();

        ArrayList<ThongTinSuDungDTO> result = new ArrayList<>();

        for (ThongTinSuDungDTO ttsd : list) {

            if (ttsd.getTGVao() != null && ttsd.getMaTV().getMaTV() == id) {

                result.add(ttsd);

            }

        }

        return result;

    }

    public ArrayList<XuLyViPhamDTO> getViPham(int id){

        ArrayList<XuLyViPhamDTO> list = xlDAL.listXuLyViPham();

        ArrayList<XuLyViPhamDTO> result = new ArrayList<>();

        for(XuLyViPhamDTO xl:list){

            if(xl.getMaTV().getMaTV() == id) result.add(xl);

        }

        return result;

    }

    public String HienThiChiTiet(int id) {

        ThanhVienDTO s =  tvDAL.getThanhVienById(id);

        String message = "";

        // Thông tin thành viên

        message += "Mã thành viên: \t" + s.getMaTV() + "\n";

        message += "Họ Tên: \t" + s.getHoTen() + "\n";

        message += "Khoa: \t" + s.getKhoa() + "\n";

        message += "Ngành: \t" + s.getNganh() + "\n\n\n";

        // Thông tin mượn thiết bị

        message += "-Thông tin mượn thiết bị: \n";

        ArrayList<ThongTinSuDungDTO> listThietBi = getMuon(id);

        boolean coThongTinMuonThietBi = false;

        int i = 1;

        for (ThongTinSuDungDTO temp : listThietBi) {

            coThongTinMuonThietBi = true;

            message += "--------------" + i + "--------------\n";

            ThietBiDTO temp1 = tbDAL.getThietBiById(temp.getMaTB().getMaTB());

            message += "Mã thành viên: \t" + id + "\n";

            message += "Tên : \t" + s.getHoTen() + "\n";

            message += "Tên Thiết Bị: \t" + temp1.getTenTB() + "\n";

            message += "Ngày mượn: \t" + temp.getTGMuon().toString() + "\n";

            message += "Ngày trả: \t" + temp.getTGTra().toString() + "\n\n";

            i++;

        }

        if (!coThongTinMuonThietBi) {

            message += "Không có thông tin mượn thiết bị\n";

        }

        // Thông tin vào khu vực học tập

        message += "-Thông tin vào khu vực học tập: \n";

        ArrayList<ThongTinSuDungDTO> listKhuVuc = getKhuVuc(id);

        boolean coThongTinVaoKhuVuc = false;

        int j = 1;

        for (ThongTinSuDungDTO temp : listKhuVuc) {

            coThongTinVaoKhuVuc = true;

            message += "--------------" + j + "--------------\n";

            message += "Mã thành viên: \t" + id + "\n";

            message += "Tên : \t" + s.getHoTen() + "\n";

            message += "Thời gian vào khu vực học tập: \t" + temp.getTGVao().toString() + "\n\n";

            j++;

        }

        if (!coThongTinVaoKhuVuc) {

            message += "Không có thông tin vào khu vực học tập\n";

        }

        // Thông tin vi phạm

        message += "-Những vi phạm: \n";

        ArrayList<XuLyViPhamDTO> listVP = getViPham(id);

        boolean VIPHAM = false;

        int k = 1;

        for (XuLyViPhamDTO a : listVP) {

            VIPHAM = true;

            message += "--------------" + k + "--------------\n";

            message += "Mã thành viên: \t" + id + "\n";

            message += "Tên : \t" + s.getHoTen() + "\n";

            message += "Hình thức xử lý: \t" + a.getHinhThucXL() + "\n";

            String tien = Integer.toString(a.getSoTien());

            message += "Tiền trả: \t" + tien + "\n";

            message += "Ngày xử lý: \t" + a.getNgayXL().toString() + "\n";

            if (a.getTrangThaiXL() == 0) {

                message += "Tình trạng: \tĐang xử lý ....\n";

            } else {

                message += "Tình trạng: \tĐã xử lý\n";

            }

            k++;

        }

        if (!VIPHAM) {

            message += "Không có thông tin vi phạm\n";

        }

        return message;

    }

* **GUI:**

private void jTable\_ThanhVienMouseClicked(java.awt.event.MouseEvent evt) {

        int selectedRow = jTable\_ThanhVien.getSelectedRow();

        if (selectedRow != DoubleClick){

            DoubleClick = selectedRow;

        }

        else{

            int ID = (int) modelTV.getValueAt(selectedRow, 0);

            ImageIcon icon = new ImageIcon("src\\main\\java\\Image\\user.png");

            JOptionPane.showMessageDialog(this, tvBLL.HienThiChiTiet(ID),"Thông Tin Thành Viên", JOptionPane.INFORMATION\_MESSAGE, icon);

            DoubleClick = -1;

        }

    }

**Chức năng 2: Quản lý thiết bị**

**Xử lý 1: Hiển thị danh sách thiết bị**

* Sequence diagram:

A close-up of a white background

Description automatically generated

* Giao diện:

A screenshot of a computer

Description automatically generated

* Code:
* **DAL** :

public ArrayList<ThietBiDTO> listThietBi() {

        Session session = factory.openSession();

        ArrayList<ThietBiDTO> List = new ArrayList<>();

        Transaction tx = null;

         try{

             tx = session.beginTransaction();

             List employess = session.createQuery("FROM ThietBiDTO").list();

             for (Iterator iterator = employess.iterator(); iterator.hasNext();) {

                 ThietBiDTO tv = (ThietBiDTO) iterator.next();

                 List.add(tv);

             }

         } catch (HibernateException e){

             if (tx != null) tx.rollback();

             e.printStackTrace();

         }

         finally {

            session.close();

        }

         return List;

    }

* **BLL** :

public ArrayList<ThietBiDTO> getListTB(){

        list\_tb = tbDAL.listThietBi();

        return list\_tb;

    }

* **GUI** :

public void loadThietBi() {

        modelTB.setRowCount(0);

        list = thietBiBLL.getListTB();

        for (ThietBiDTO t : list) {

            Object[] row = {false,t.getMaTB(),t.getTenTB(),t.getMoTaTB()};

            modelTB.addRow(row);

        }

    }

**Xử lý 2: Thêm thiết bị**

***Xử lý 2.1. Nhập thông tin 1 thiết bị:***

* Sequence diagram :

A diagram of a diagram

Description automatically generated

* Giao diện:

A screenshot of a computer

Description automatically generated

* Code :
* **DAL** :

public void addThietBi(ThietBiDTO tb) {

        Session session = factory.openSession();

        Transaction tx = null;

        Integer tvienID = null;

        try {

            tx = session.beginTransaction();

            ThietBiDTO tbDTO = new ThietBiDTO(listThietBi().size()+1,tb.getTenTB(),tb.getMoTaTB());

            session.save(tb);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL** :

public void ThemTB(ThietBiDTO tb){

        tbDAL.addThietBi(tb);

    }

* **GUI** :

Calendar cal = Calendar.getInstance();

        // Lấy năm hiện tại

        int year = cal.get(Calendar.YEAR);

        int Selected = cbLoai.getSelectedIndex();

        //Nếu người dùng chưa chọn loại thì báo lỗi

        if(Selected==0){

            JOptionPane.showMessageDialog(rootPane, "Chưa chọn loại thiết bị!");

        }

        idTB = Integer.toString(Selected) + Integer.toString(year) + Integer.toString(idNext);

        String tenTB = txtenTB.getText();

        String motaTB = taMT.getText();

        if(tenTB.isEmpty()){

            JOptionPane.showMessageDialog(rootPane, "Chưa nhập tên thiết bị!");

            return;

        }

        ThietBiDTO tbDTO = new ThietBiDTO();

        tbDTO.setMaTB(Integer.parseInt(idTB));

        tbDTO.setTenTB(tenTB);

        tbDTO.setMoTaTB(motaTB);

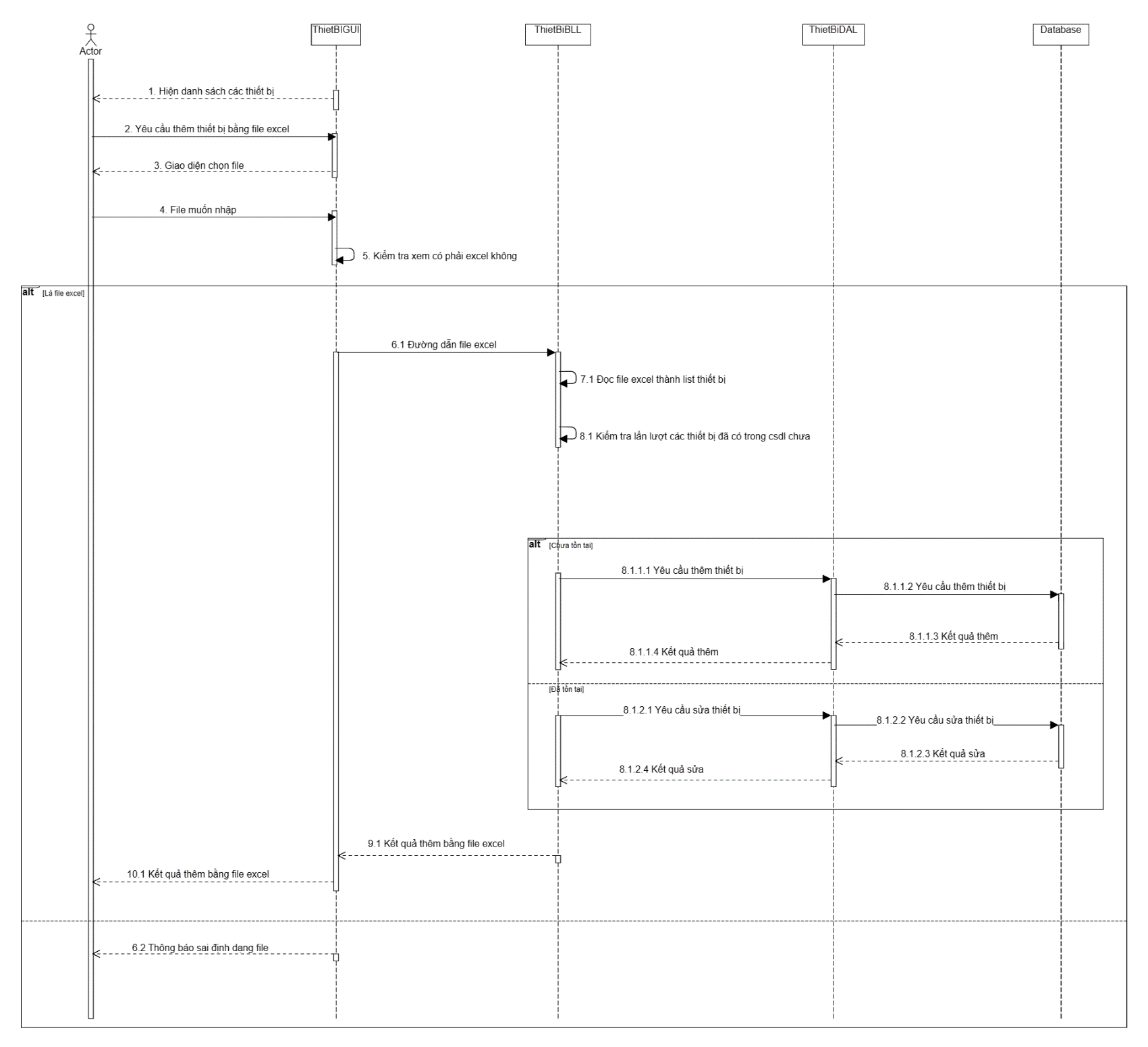
        tbBLL.ThemTB(tbDTO);

        JOptionPane.showMessageDialog(rootPane, "Thêm thành công!");

        this.dispose();

***Xử lý 2.2. Nhập từ file excel :***

* Sequence diagram :



* Giao diện :

A screenshot of a computer

Description automatically generated

* Code :
* **BLL** :

ArrayList<ThietBiDTO> list\_excel = new ArrayList<>();

        try{

            FileInputStream inputStream = new FileInputStream(new File(filePath));

            Workbook workbook = WorkbookFactory.create(inputStream);

            Sheet sheet = workbook.getSheetAt(0); // Lấy sheet đầu tiên

            Row Titlerow = sheet.getRow(0);

            Cell TitleMaCell = Titlerow.getCell(0);

            Cell TitleTenCell = Titlerow.getCell(1);

            Cell TitleMoTaCell = Titlerow.getCell(2);

            if (TitleMaCell != null && TitleTenCell != null && TitleMoTaCell != null){

                String ma = TitleMaCell.getStringCellValue();

                String ten = TitleTenCell.getStringCellValue();

                String mota = TitleMoTaCell.getStringCellValue();

                if (!ma.equals("MaTB") || !ten.equals("TenTB") || !mota.equals("MoTaTB")){

                    return "File không đúng cấu trúc cột";

                }

                // Bắt đầu đọc từ hàng thứ hai (index 1)

                for (int i = 1; i <= sheet.getLastRowNum(); i++) {

                    Row row = sheet.getRow(i);

                    if (row != null) {

                        Cell MaCell = row.getCell(0);

                        Cell TenCell = row.getCell(1);

                        Cell MoTaCell = row.getCell(2);

                        // Kiểm tra và lưu dữ liệu vào Hibernate

                        if (MaCell  != null && TenCell != null && MoTaCell != null) {

                            double Ma = 0;

                            String Ten = "";

                            String MoTa = "";

                            try{

                                Ma = MaCell.getNumericCellValue();

                                Ten = TenCell.getStringCellValue();

                                MoTa = MoTaCell.getStringCellValue();

                            }

                            catch(Exception e){

                                return "MaTB là số, TenTB và MoTaTB là chuỗi ký tự";

                            }

                            ThietBiDTO thietbi = new ThietBiDTO((int) Ma, Ten, MoTa);

                            list\_excel.add(thietbi);

                        }

                    }

                }

            }

            for (ThietBiDTO a : list\_excel){

                if (tbDAL.kiemTraMaThietBiTonTai(a.getMaTB())){

                    tbDAL.updateThietBi(a);

                }

                else{

                    tbDAL.addThietBi(a);

                }

            }

            return "Thêm thành công";

        }catch(Exception e){

            e.printStackTrace();

        }

        return "File rỗng rồi";

* **GUI** :

JFileChooser fileChooser = new JFileChooser();

        int returnValue = fileChooser.showOpenDialog(null);

        if (returnValue == JFileChooser.APPROVE\_OPTION) {

            String filePath = fileChooser.getSelectedFile().getPath();

            if (isExcelFile(filePath)){

                try {

                    JOptionPane.showMessageDialog(fileChooser, thietBiBLL.AddExcel(filePath));

                    loadThietBi();

                } catch (Exception ex) {

                    ex.printStackTrace();

                }

            }

            else{

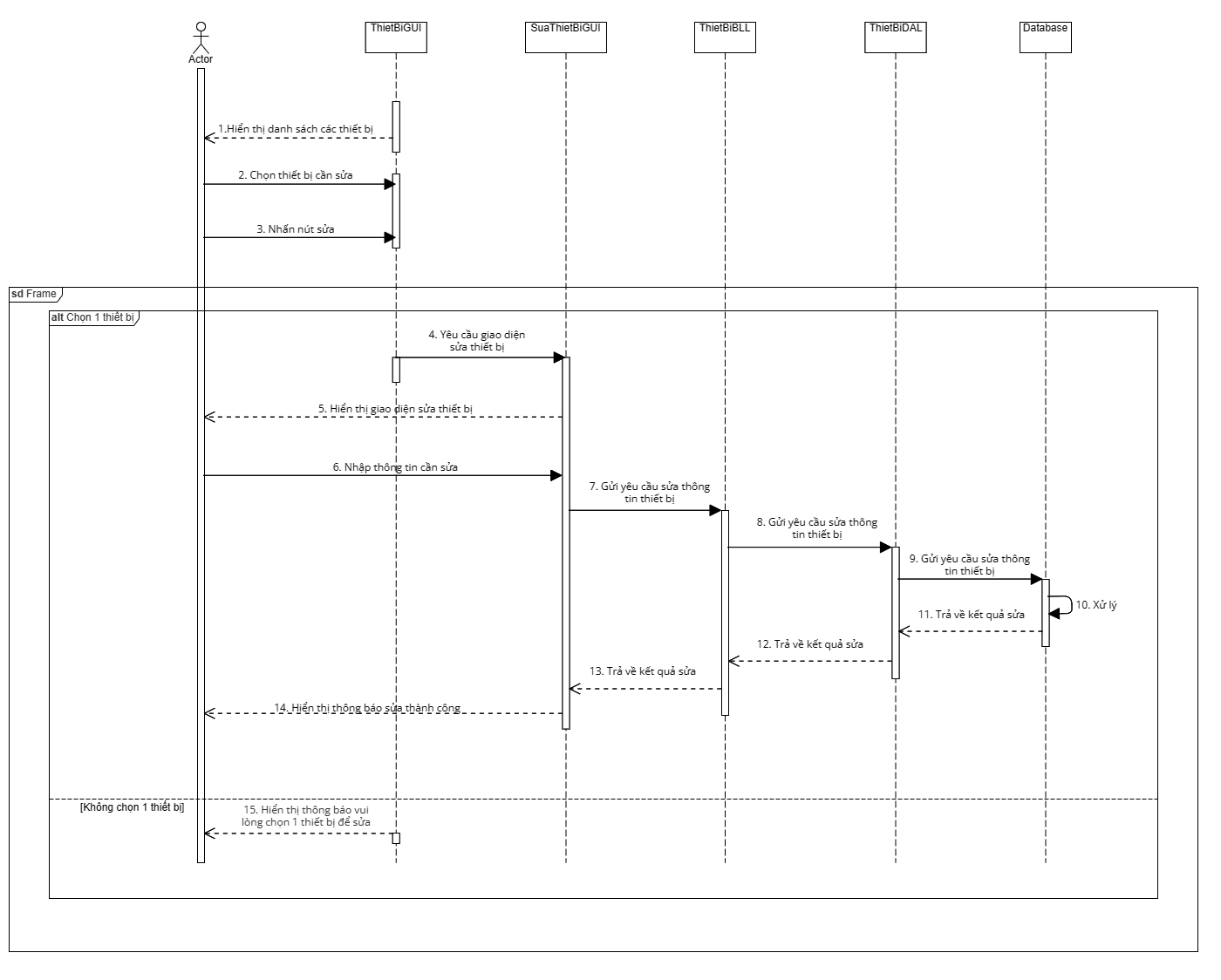
                JOptionPane.showMessageDialog(fileChooser, "File đã chọn không phải là file excel");

            }

        }

**Xử lý 3: Sửa thông tin thiết bị**

* **Sequence diagram :**



* **Giao diện :**

**A screenshot of a computer

Description automatically generated**

* **Code :**
* **DAL** :

public void updateThietBi(ThietBiDTO tb) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            ThietBiDTO tbi = (ThietBiDTO) session.get(ThietBiDTO.class, tb.getMaTB());

            tbi.setMoTaTB(tb.getMoTaTB());

            tbi.setTenTB(tb.getTenTB());

            session.update(tbi);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL** :

public void SuaTB(ThietBiDTO tb){

        tbDAL.updateThietBi(tb);

    }

* **GUI** :

-ThietBiGUI.java:

int selectedRow = jTableThietBi.getSelectedRow();

        if (selectedRow == -1) {

            JOptionPane.showMessageDialog(this, "Vui lòng chọn một dòng để sửa đổi.", "Thông báo", JOptionPane.WARNING\_MESSAGE);

        } else {

            idTB = (int) jTableThietBi.getValueAt(selectedRow, 1);

            TenTB = jTableThietBi.getValueAt(selectedRow, 2).toString();

            Object cellValue = jTableThietBi.getValueAt(selectedRow, 3);

            String MoTaTB = (cellValue != null) ? cellValue.toString() : "";

            SuaThietBi SuaTB = new SuaThietBi(idTB,TenTB,MoTaTB);

            SuaTB.setVisible(true);

        }

-SuaThietBi.java:

idTB = Integer.parseInt(txtMaTB.getText());

        TenTB = txtenTB.getText().toString();

        MoTaTB = taMT.getText().toString();

        ThietBiDTO tbDTO= new ThietBiDTO(idTB, TenTB, MoTaTB);

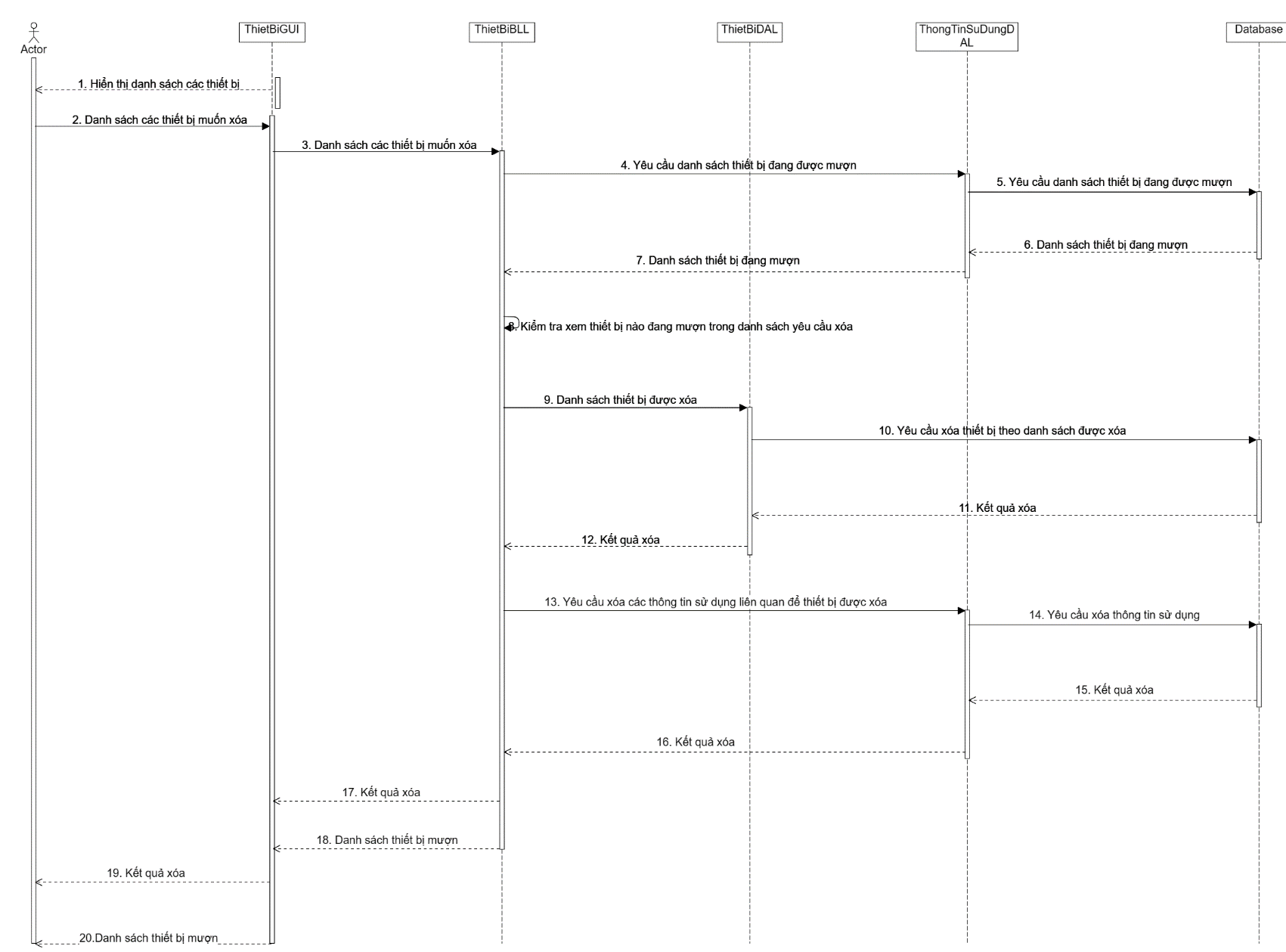
        tbBLL.SuaTB(tbDTO);

        JOptionPane.showMessageDialog(rootPane, "Sửa thiết bị thành công!");

        dispose();

**Xử lý 4: Xóa thiết bị**

* Sequence Diagram:



* Giao diện:

A screenshot of a computer

Description automatically generated

* Code:
* **DAL**

ThietBiDAL

public void deleteThietBi(int tbiID) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            ThietBiDTO tbi = (ThietBiDTO) session.get(ThietBiDTO.class, tbiID);

            session.delete(tbi);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

ThongTinSuDungDAL

public void xoaThongTinSuDung(int id) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            ThongTinSuDungDTO thongTin = (ThongTinSuDungDTO) session.get(ThongTinSuDungDTO.class, id);

            session.delete(thongTin);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL**

public int CheckMuon(int ID){

        ArrayList<ThongTinSuDungDTO> list = new ThongTinSuDungBLL().listThongTinSuDung();

        for (ThongTinSuDungDTO s : list){

            if (s.getMaTB() != null){

                if (s.getMaTB().getMaTB() == ID && s.getTGTra() != null){

                    return s.getMaTT();

                }

            }

        }

        return 0;

    }

public String KiemTraTruocKhiXoa(DefaultTableModel model){

        ArrayList<String> list = new ArrayList<>();

        ArrayList<ThongTinSuDungDTO> list\_ttsd = new ArrayList<>();

        int dem = 0;

        for (int i = 0; i < model.getRowCount();i++){

            boolean trangthai = (boolean) model.getValueAt(i, 0);

            if (trangthai){

                int ID = (int) model.getValueAt(i, 1);

                dem++;

                if (CheckMuon(ID)!=0){

                    list.add(ID + " " +  model.getValueAt(i, 2));

                }

                else{

                    list\_ttsd = new ThongTinSuDungDAL().listThongTinSuDung();

                    for (ThongTinSuDungDTO s : list\_ttsd){

                        if (s.getMaTB() != null){

                            if (s.getMaTB().getMaTB() == ID) {

                                new ThongTinSuDungDAL().xoaThongTinSuDung(s.getMaTT());

                            }

                        }

                    }

                    tbDAL.deleteThietBi(ID);

                }

            }

        }

        if (list.size() != 0){

            String result = "Mã thiết bị đang mượn: ";

            for (String i : list){

                result += "\n" + i;

            }

            return result;

        }

        else{

            if (dem == 0){

                return "Hãy chọn các thiết bị muốn xóa";

            }

            else{

                return "Xóa thành công";

            }

        }

    }

* **GUI**

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnDeleteActionPerformed

        // TODO add your handling code here:

        if (JOptionPane.showConfirmDialog(jScrollPane2, "Bạn có thực sự muốn xóa không") == JOptionPane.YES\_OPTION){

            JOptionPane.showMessageDialog(jScrollPane2, thietBiBLL.KiemTraTruocKhiXoa(modelTB));

            loadThietBi();

        }

    }//GEN-LAST:event\_btnDeleteActionPerformed

**Xử lý 5: Tìm kiếm thiết bị**

* Sequence Diagram:

A diagram of a diagram

Description automatically generated with medium confidence

* Giao diện:

A screenshot of a computer

Description automatically generated

* Code:

- **DAL**

public ArrayList<ThietBiDTO> listThietBi() {

        Session session = factory.openSession();

        ArrayList<ThietBiDTO> List = new ArrayList<>();

        Transaction tx = null;

         try{

             tx = session.beginTransaction();

             List employess = session.createQuery("FROM ThietBiDTO").list();

             for (Iterator iterator = employess.iterator(); iterator.hasNext();) {

                 ThietBiDTO tv = (ThietBiDTO) iterator.next();

                 List.add(tv);

             }

         } catch (HibernateException e){

             if (tx != null) tx.rollback();

             e.printStackTrace();

         }

         finally {

            session.close();

        }

         return List;

    }

 public ArrayList<ThietBiDTO> listThietBiComboBox(String id) {

        Session session = factory.openSession();

        ArrayList<ThietBiDTO> List = new ArrayList<>();

        Transaction tx = null;

         try{

                tx = session.beginTransaction();

                String hql = "FROM ThietBiDTO WHERE CAST(MaTB AS string) LIKE :value";

                Query query = session.createQuery(hql);

                query.setParameter("value", "" + id + "%");

                List result = query.list();

                for (Iterator iterator = result.iterator(); iterator.hasNext();) {

                 ThietBiDTO tv = (ThietBiDTO) iterator.next();

                 List.add(tv);

             }

         } catch (HibernateException e){

             if (tx != null) tx.rollback();

             e.printStackTrace();

         }

         finally {

            session.close();

        }

         return List;

    }

- **BLL**

public void search(ArrayList<ThietBiDTO> list\_tb,DefaultTableModel model, String txtMaTB, String txtTen){

        model.setRowCount(0);

        for (ThietBiDTO tb : list\_tb){

            if (!txtMaTB.equals("Nhập mã thiết bị") && !txtTen.equals("Nhập tên thiết bị")){

                if (Integer.toString(tb.getMaTB()).contains(txtMaTB) && tb.getTenTB().contains(txtTen)){

                    model.addRow(new Object[]{false,tb.getMaTB(),tb.getTenTB(),tb.getMoTaTB()});

                }

            }

            else if(txtMaTB.equals("Nhập mã thiết bị")){

                if (tb.getTenTB().contains(txtTen)){

                    model.addRow(new Object[]{false,tb.getMaTB(),tb.getTenTB(),tb.getMoTaTB()});

                }

            }

            else if(txtTen.equals("Nhập tên thiết bị")){

                if (Integer.toString(tb.getMaTB()).contains(txtMaTB)){

                    model.addRow(new Object[]{false,tb.getMaTB(),tb.getTenTB(),tb.getMoTaTB()});

                }

            }

        }

    }

    public ArrayList<ThietBiDTO> getListSearch(String id){

        return tbDAL.listThietBiComboBox(id);

    }

- **GUI**

Hàm bắt sự kiện của TextBox Mã thiết bị

private void txtFindMaKeyReleased(java.awt.event.KeyEvent evt) {//GEN-FIRST:event\_txtFindMaKeyReleased

        // TODO add your handling code here:

        int Selected = cbxLoai.getSelectedIndex();

        if (Selected != 0){

            thietBiBLL.search(thietBiBLL.getListSearch(Integer.toString(Selected)),modelTB, txtFindMa.getText(), txtFindTen.getText());

        }

        else{

            thietBiBLL.search(list,modelTB, txtFindMa.getText(), txtFindTen.getText());

        }

    }//GEN-LAST:event\_txtFindMaKeyReleased

Hàm bắt sự kiện của TextBox Tên thiết bị

    private void txtFindTenKeyReleased(java.awt.event.KeyEvent evt) {//GEN-FIRST:event\_txtFindTenKeyReleased

        // TODO add your handling code here:'

        int Selected = cbxLoai.getSelectedIndex();

        if (Selected != 0){

            thietBiBLL.search(thietBiBLL.getListSearch(Integer.toString(Selected)),modelTB, txtFindMa.getText(), txtFindTen.getText());

        }

        else{

            thietBiBLL.search(list,modelTB, txtFindMa.getText(), txtFindTen.getText());

        }

    }//GEN-LAST:event\_txtFindTenKeyReleased

Hàm bắt sự kiện của ComboBox Loại thiết bị

private void cbxLoaiActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_cbxLoaiActionPerformed

        // TODO add your handling code here:

        int Selected = cbxLoai.getSelectedIndex();

        txtFindMa.setText("Nhập mã thiết bị");

        txtFindTen.setText("Nhập tên thiết bị");

        if (Selected != 0){

            ArrayList<ThietBiDTO> list\_temp = thietBiBLL.getListSearch(Integer.toString(Selected));

            modelTB.setRowCount(0);

            for (ThietBiDTO t : list\_temp) {

                Object[] row = {false,t.getMaTB(),t.getTenTB(),t.getMoTaTB()};

                modelTB.addRow(row);

            }

        }

        else{

            loadThietBi();

        }

**Xử lý 6: Hiển thị chi tiết thông tin thiết bị**

* Sequence Diagram

A screenshot of a computer

Description automatically generated

* Giao diện

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Code
* **DAL**

public ArrayList<ThongTinSuDungDTO> listThongTinSuDung() {

        Session session = factory.openSession();

        ArrayList<ThongTinSuDungDTO> list = new ArrayList<>();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            list = (ArrayList<ThongTinSuDungDTO>) session.createQuery("FROM ThongTinSuDungDTO").list();

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

        return list;

    }

* **BLL**

public String HienChiTiet(int ID){

        ArrayList<ThongTinSuDungDTO> list = new ThongTinSuDungBLL().listThongTinSuDung();

        String message = "";

        for (ThongTinSuDungDTO s : list){

            if (s.getMaTB() != null){

                if (s.getMaTB().getMaTB() == ID && s.getTGTra() != null){

                    ThanhVienDTO temp = new ThanhVienBLL().getThanhVienByID(s.getMaTV().getMaTV());

                    message += "\nMã thành viên: " + s.getMaTV().getMaTV();

                    message += "\nTên người mượn: " + temp.getHoTen();

                    message += "\nNgày mượn: " + s.getTGMuon().toString();

                    message += "\nNgày trả: " + s.getTGTra().toString();

                    return message;

                }

            }

        }

        return "Thiết bị chưa được mượn";

    }

* **GUI**

private int DoubleClick = -1;

private void jTableThietBiMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jTableThietBiMouseClicked

        // TODO add your handling code here:

        int selectedRow = jTableThietBi.getSelectedRow();

        if (selectedRow != DoubleClick){

            DoubleClick = selectedRow;

        }

        else{

            int ID = (int) modelTB.getValueAt(selectedRow, 1);

            ImageIcon icon = new ImageIcon("src\\main\\java\\Image\\user.png");

            JOptionPane.showMessageDialog(jScrollPane2, thietBiBLL.HienChiTiet(ID),"Thông tin mượn", HEIGHT, icon);

            DoubleClick = -1;

        }

    }

**Chức năng 3: Thống kê**

**Xử lý 1: Thống kê thành viên**

* Sequence diagram:

A diagram of a diagram

Description automatically generated with medium confidence

* Giao diện

**A screenshot of a computer

Description automatically generated**

* **Thống kê theo thời gian**

**A screen shot of a blue screen

Description automatically generated**

* **Thống kê theo Khoa**

**A colorful pie chart with numbers

Description automatically generated**

* **Thống kê theo ngành**

**A pie chart with different colored circles

Description automatically generated**

* Code

**- DAL**

**- ThongTinSuDungDAL**

**+ Lấy danh sách thông tin sử dụng**

public ArrayList<ThongTinSuDungDTO> listThongTinSuDung() {

        Session session = factory.openSession();

        ArrayList<ThongTinSuDungDTO> list = new ArrayList<>();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            list = (ArrayList<ThongTinSuDungDTO>) session.createQuery("FROM ThongTinSuDungDTO").list();

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

        return list;

    }

**+ Thống kê theo tháng**

public List<Object[]> getThongKeThang() {

        try (Session session = factory.openSession()) {

            String sql = "SELECT DATE\_FORMAT(TGVao, '%d/%m/%y') AS Ngay, COUNT(TGVao) AS SoLuongThoiGianVao, COUNT(TGMuon) AS SoLuongThoiGianMuon FROM ThongTinSuDungDTO GROUP BY DATE\_FORMAT(TGVao, '%d-%m') ORDER BY TGVao ASC";

            Query<Object[]> query = session.createQuery(sql);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

            return null;

        }

    }

**+ Thống kê theo ngày**

public List<Object[]> getThongKeNgay(String datefind) {

        try (Session session = factory.openSession()) {

            String sql = "SELECT DATE\_FORMAT(TGVao, '%H:00') AS Gio, " +

                                "       COUNT(TGVao) AS SoLuongThoiGianVao, " +

                                "       COUNT(TGMuon) AS SoLuongThoiGianMuon " +

                                "FROM ThongTinSuDungDTO " +

                                "WHERE DATE(TGVao) = '" + datefind + "' " +

                                "GROUP BY DATE\_FORMAT(TGVao, '%H:00') " +

                                "ORDER BY Gio ASC";

            Query<Object[]> query = session.createQuery(sql);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

            return null;

        }

    }

**- ThanhVienDAL**

**+ Lấy thành viên theo ID**

public ThanhVienDTO getThanhVienById(int id) {

        Session session = factory.openSession();

        ThanhVienDTO thanhVien = null;

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            // Sử dụng phương thức get của session để lấy đối tượng ThanhVienDTO theo id

            thanhVien = session.get(ThanhVienDTO.class, id);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

        return thanhVien;

    }

**- BLL**

**- ThongTinSuDungBLL**

public ArrayList<ThongTinSuDungDTO> listThongTinSuDung() {

        return ttDAL.listThongTinSuDung();

    }

public List<Object[]> getThongKeThang() {

        return ttDAL.getThongKeThang();

    }

public List<Object[]> getThongKeNgay(String datefind) {

        DateTimeFormatter currentFormatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        // Định nghĩa định dạng mới của chuỗi ngày

        DateTimeFormatter newFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

        // Chuyển chuỗi ngày sang LocalDate

        LocalDate date = LocalDate.parse(datefind, currentFormatter);

        // Đổi định dạng của LocalDate sang chuỗi với định dạng mới

        String newDateString = date.format(newFormatter);

        return ttDAL.getThongKeNgay(newDateString);

    }

**- ThanhVienBLL**

public ThanhVienDTO getThanhVienByID(int id)

    {

        return tvDAL.getThanhVienById(id);

    }

**- GUI**

**- Thống kê theo thời gian**

**+ Chọn thời gian để thống kê**

tinSuDungBLL = new ThongTinSuDungBLL();

        tvBLL = new ThanhVienBLL();

        String listMonth[] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

        String listDays[] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};

        UIManager.put("DateChooser.listMonth", listMonth);

        UIManager.put("DateChooser.listDay", listDays);

        FlatAnimatedLafChange.showSnapshot();

        FlatMacLightLaf.setup();

        FlatLaf.updateUI();

        FlatAnimatedLafChange.hideSnapshotWithAnimation();

        ch = new DateChooser();

        ch.setDateSelectable(new DateSelectable() {

            @Override

            public boolean isDateSelectable(Date date) {

                return date.before(new Date());

            }

        });

        ch.addActionDateChooserListener(

                new DateChooserAdapter() {

                    @Override

                    public void dateChanged(Date date, DateChooserAction action) {

                        System.out.println("date single selected...");

                    }

                    @Override

                    public void dateBetweenChanged(DateBetween date, DateChooserAction action) {

                        System.out.println(date);

                    }

                });

        ch.setTextField(jTextField1);

        ch.setDateSelectionMode(DateChooser.DateSelectionMode.BETWEEN\_DATE\_SELECTED);

+ Kiểm tra thời gian chọn

public boolean isSameDate(Date date1, Date date2) {

        Calendar cal1 = Calendar.getInstance();

        cal1.setTime(date1);

        Calendar cal2 = Calendar.getInstance();

        cal2.setTime(date2);

        return cal1.get(Calendar.YEAR) == cal2.get(Calendar.YEAR) &&

                cal1.get(Calendar.MONTH) == cal2.get(Calendar.MONTH) &&

                cal1.get(Calendar.DAY\_OF\_MONTH) == cal2.get(Calendar.DAY\_OF\_MONTH);

    }

    private boolean jtextviewToDateToData(Date inputDate){

        String dateRange = jTextField1.getText().toString();

        // Chuyển đổi chuỗi ngày bắt đầu và kết thúc thành đối tượng Date

        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");

        Date startDate = null;

        Date endDate = null;

        if (dateRange.length() < 11)  {

            try {

                startDate = dateFormat.parse(dateRange);

            } catch (ParseException ex) {

                Logger.getLogger(ThongKeGUI.class.getName()).log(Level.SEVERE, null, ex);

            }

            return isSameDate(startDate, inputDate);

        }

        String startDateStr = dateRange.split(" to ")[0];

        String endDateStr = dateRange.split(" to ")[1];

        try {

            startDate = dateFormat.parse(startDateStr);

            endDate = dateFormat.parse(endDateStr);

        } catch (ParseException e) {

            e.printStackTrace();

        }

        // Kiểm tra xem ngày đầu vào có nằm trong phạm vi không

       return !(inputDate.before(startDate) || inputDate.after(endDate));

    }

**- Thống kê theo khoa**

private void drawPieChart1(ArrayList<ThongTinSuDungDTO> listtemp){

        pieChart1.clearData();

        Map<String, Integer> khoaCounts = new HashMap<>();

        // Lặp qua danh sách thông tin sử dụng và cập nhật số lượng của từng loại khoa trong Map

        for (ThongTinSuDungDTO t : listtemp) {

            if (t.getTGVao() == null) continue;

            String khoa = tvBLL.getThanhVienByID(t.getMaTV().getMaTV()).getKhoa();

            khoa = khoa.toUpperCase(Locale.ROOT);

            khoaCounts.put(khoa, khoaCounts.getOrDefault(khoa, 0) + 1);

        }

        // Lặp qua Map và thêm dữ liệu vào biểu đồ

        for (Map.Entry<String, Integer> entry : khoaCounts.entrySet()) {

            String khoa = entry.getKey();

            int count = entry.getValue();

            Color color = getColorForKhoa(khoa);

            pieChart1.addData(new ModelPieChart(khoa, count, color));

        }

        jLabel3.setText("Theo Khoa (" + listtemp.size() + ")");

    }

**- Thống kê theo ngành**

private void CallPieChart(String khoa, ArrayList<ThongTinSuDungDTO> listtemp) {

        if (khoa == null) return;

        pieChart2.clearData();

        // Tạo Map ánh xạ từng loại khoa tới danh sách các ngành

        Map<String, List<String>> nganhKhoaMap = new HashMap<>();

        nganhKhoaMap.put("SP KHXH", List.of("ĐỊA", "SỬ", "VĂN"));

        nganhKhoaMap.put("SP KHTN", List.of("LÍ", "HÓA", "SINH"));

        nganhKhoaMap.put("NGOẠI NGỮ", List.of("ANH", "NNA"));

        nganhKhoaMap.put("QTKD", List.of("QTKD"));

        nganhKhoaMap.put("QLGD", List.of("TLH", "QLGD"));

        nganhKhoaMap.put("TOÁN UD", List.of("TOÁN", "TOÁN UD"));

        nganhKhoaMap.put("CNTT", List.of("CNTT", "KTPM", "HTTT"));

        // Lấy danh sách ngành tương ứng với loại khoa

        List<String> nganhList = nganhKhoaMap.getOrDefault(khoa.toUpperCase(Locale.ROOT), List.of());

        // Mảng màu cho các ngành

        Color khoaColor = getColorForKhoa(khoa); // Màu của khoa đầu vào

        Color[] color\_pie = new Color[4];

        color\_pie[0] = khoaColor;

        // Tạo mảng màu cho các ngành khác không trùng với màu của khoa đầu vào

        int colorIndex = 1;

        for (Color color : COLOR\_OPTIONS) {

            if (!color.equals(khoaColor)) {

                color\_pie[colorIndex++] = color;

                if (colorIndex >= 4) break;

            }

        }

        int tong = 0;

        // Lặp qua danh sách ngành và thêm dữ liệu vào biểu đồ

        for (int i = 0; i < nganhList.size(); i++) {

            String nganh = nganhList.get(i);

            int dem = 0;

            for (ThongTinSuDungDTO t : listtemp) {

                if (t.getTGVao() == null) continue;

                String nganhTV = tvBLL.getThanhVienByID(t.getMaTV().getMaTV()).getNganh();

                if (nganhTV.toUpperCase(Locale.ROOT).equals(nganh)) {

                    dem++;

                }

            }

            if (dem > 0) {

                pieChart2.addData(new ModelPieChart(nganh, dem, color\_pie[i]));

                tong += dem;

            }

        }

        jLabel2.setText("Theo Ngành (" + khoa + ": " + tong + ")");

**Xử lý 2: Thống kê sử dụng thiết bị**

* Sequence diagram:

A diagram of a diagram

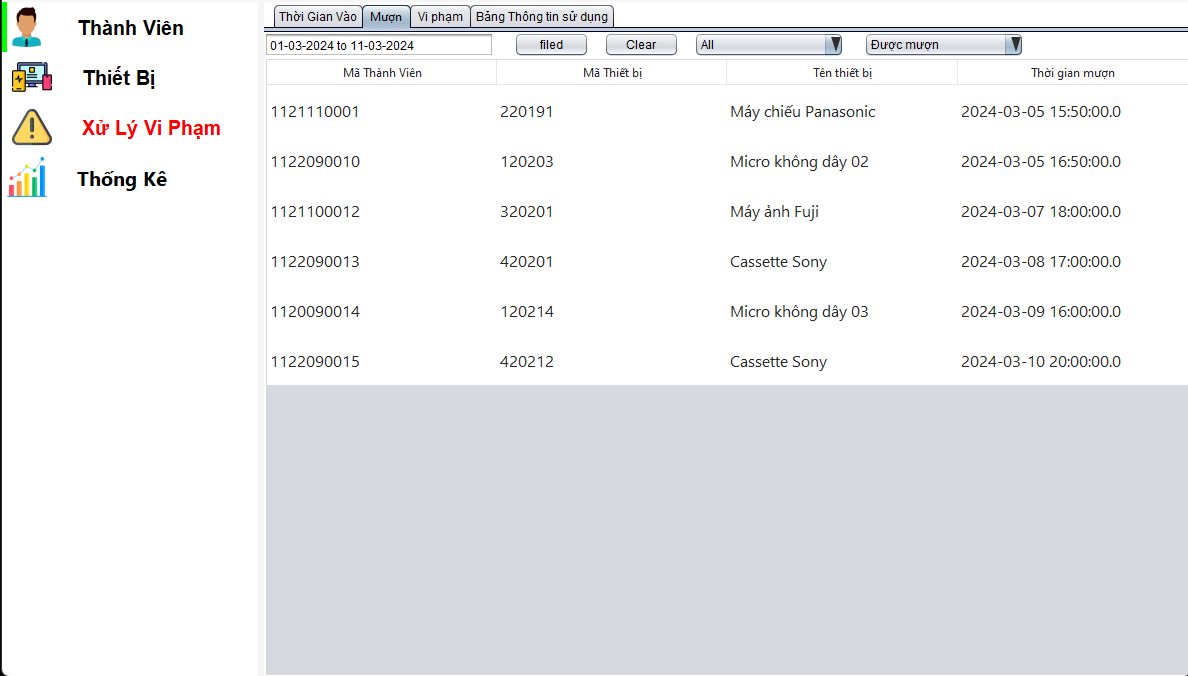
Description automatically generated

* Giao diện
* Giao diện thống kê

A screenshot of a computer

Description automatically generated

* Thống kê theo thời gian



* Thống kê theo tên thiết bị

A screenshot of a computer

Description automatically generated

* Thống kê đang mượn

A screenshot of a computer

Description automatically generated

* Code

**- DAL**

**- ThongTinSuDungDAL**

**+ Thống kê thiết bị được mượn**

public List<Object[]> getThongKeMuon() {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT MaTV.MaTV, MaTB.MaTB, MaTB.TenTB, TGMuon FROM ThongTinSuDungDTO WHERE MaTB is not NULL";

            Query<Object[]> query = session.createQuery(queryVao);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return null;

    }

public List<Object[]> getThongKeMuonDaTra() {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT MaTV.MaTV, MaTB.MaTB, MaTB.TenTB, TGMuon FROM ThongTinSuDungDTO WHERE MaTB is not NULL and TGTra is not null ";

            Query<Object[]> query = session.createQuery(queryVao);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return null;

    }

**+ Thống kê thiết bị được mượn theo thời gian**

public List<Object[]> getThongKeMuon(String datestart, String dateend) {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT MaTV.MaTV, MaTB.MaTB, MaTB.TenTB, TGMuon FROM ThongTinSuDungDTO WHERE MaTB is not NULL "

                    + "AND TGMuon BETWEEN '"+ datestart +"' AND '"+ dateend +"' ";

            Query<Object[]> query = session.createQuery(queryVao);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return null;

    }

public List<Object[]> getThongKeMuonDaTra(String datestart, String dateend) {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT MaTV.MaTV, MaTB.MaTB, MaTB.TenTB, TGMuon FROM ThongTinSuDungDTO WHERE MaTB is not NULL and TGTra is not null "

                    + "AND TGMuon BETWEEN '"+ datestart +"' AND '"+ dateend +"' ";

            Query<Object[]> query = session.createQuery(queryVao);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return null;

    }

**- BLL**

**- ThongTinSuDungBLL**

public List<Object[]> getThongKeMuon() {

        return ttDAL.getThongKeMuon();

    }

    public List<Object[]> getThongKeMuon(String dateFind) {

        String startDateStr = dateFind.split(" to ")[0];

        String endDateStr = dateFind.split(" to ")[1];

        System.out.println("BLL.ThongTinSuDungBLL.getThongKeMuon() + " +chagneDateToIDK(startDateStr) + " " + chagneDateToIDK(endDateStr));

        return ttDAL.getThongKeMuon(chagneDateToIDK(startDateStr) , chagneDateToIDK(endDateStr));

    }

    public List<Object[]> getThongKeMuonDaTra() {

        return ttDAL.getThongKeMuonDaTra();

    }

    public List<Object[]> getThongKeMuonDaTra(String dateFind) {

        String startDateStr = dateFind.split(" to ")[0];

        String endDateStr = dateFind.split(" to ")[1];

        System.out.println("BLL.ThongTinSuDungBLL.getThongKeMuon() + " +chagneDateToIDK(startDateStr) + " " + chagneDateToIDK(endDateStr));

        return ttDAL.getThongKeMuonDaTra(chagneDateToIDK(startDateStr) , chagneDateToIDK(endDateStr));

    }

**- GUI**

**+ Thống kê thiết bị được mượn**

jTable2 = new javax.swing.JTable()

        {

            public Component prepareRenderer(TableCellRenderer renderer, int row, int column)

            {

                Component c = super.prepareRenderer(renderer, row, column);

                Object cellValue = jTable2.getValueAt(row, column);

                //  Alternate row color

                if (cellValue == "NULL") {

                    c.setForeground(Color.RED);

                } else {

                    // Thiết lập màu nền mặc định cho các ô khác

                    c.setForeground(jTable2.getForeground());

                }

                return c;

            }

        };

        jTable2.setModel(new javax.swing.table.DefaultTableModel(

            new Object [][] {

            },

            new String [] {

                "Mã Thành Viên", "Mã Thiết bị", "Tên thiết bị", "Thời gian mượn"

            }

        ));

        jScrollPane2.setViewportView(jTable2);

        addTableMuon();

    }

public void addTableMuon() {

        DefaultTableModel tableModel = (DefaultTableModel) jTable2.getModel();

        while (tableModel.getRowCount() > 0) {

            tableModel.removeRow(0);

        }

        if (jComboBox2.getSelectedIndex() == 0) {

            for (Object[] t: tinSuDungBLL.getThongKeMuon()) {

                tableModel.insertRow(tableModel.getRowCount(), t);

            }

        } else {

            for (Object[] t: tinSuDungBLL.getThongKeMuonDaTra()) {

                tableModel.insertRow(tableModel.getRowCount(), t);

            }

        }

    }

**+ Thống kê theo thời gian mượn thiết bị**

* Hàm chọn thời gian thống kê

ch1 = new DateChooser();

        ch1.setDateSelectable(new DateSelectable() {

            @Override

            public boolean isDateSelectable(Date date) {

                return date.before(new Date());

            }

        });

        ch1.addActionDateChooserListener(

                new DateChooserAdapter() {

                    @Override

                    public void dateChanged(Date date, DateChooserAction action) {

                        System.out.println("date single selected...");

                    }

                    @Override

                    public void dateBetweenChanged(DateBetween date, DateChooserAction action) {

                        System.out.println(date);

                    }

                });

        ch1.setTextField(jTextField2);

        ch1.setDateSelectionMode(DateChooser.DateSelectionMode.BETWEEN\_DATE\_SELECTED);

* Xử lý thống kê theo thời gian

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton3ActionPerformed

        String datefind = jTextField2.getText().toString();

        if (datefind.length() < 11) return;

        DefaultTableModel tableModel = (DefaultTableModel) jTable2.getModel();

        while (tableModel.getRowCount() > 0) {

            tableModel.removeRow(0);

        }

        if (jComboBox2.getSelectedIndex() == 0) {

            for (Object[] t: tinSuDungBLL.getThongKeMuon(datefind)){

                tableModel.insertRow(tableModel.getRowCount(), t);

            }

        } else {

            for (Object[] t: tinSuDungBLL.getThongKeMuonDaTra(datefind)){

                tableModel.insertRow(tableModel.getRowCount(), t);

            }

        }

    }

**+ Thống kê theo tên các thiết bị**

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jComboBox1ActionPerformed

        String selectedIndex = "" +  jComboBox1.getSelectedIndex();

        if (selectedIndex.equals("0")) {

            selectedIndex = "";

        }

        System.out.println("Selected Item: " + selectedIndex);

        TableRowSorter<DefaultTableModel> rowSorter = new TableRowSorter<>((DefaultTableModel) jTable2.getModel());

        jTable2.setRowSorter(rowSorter);

        rowSorter.setRowFilter(RowFilter.regexFilter("^" + selectedIndex, 1)); // Lọc theo cột thứ 2

    }

**+ Thống kê được mượn và đang mượn**

public void addTableMuon() {

        DefaultTableModel tableModel = (DefaultTableModel) jTable2.getModel();

        while (tableModel.getRowCount() > 0) {

            tableModel.removeRow(0);

        }

        if (jComboBox2.getSelectedIndex() == 0) {

            for (Object[] t: tinSuDungBLL.getThongKeMuon()) {

                tableModel.insertRow(tableModel.getRowCount(), t);

            }

        } else {

            for (Object[] t: tinSuDungBLL.getThongKeMuonDaTra()) {

                tableModel.insertRow(tableModel.getRowCount(), t);

            }

        }

    }

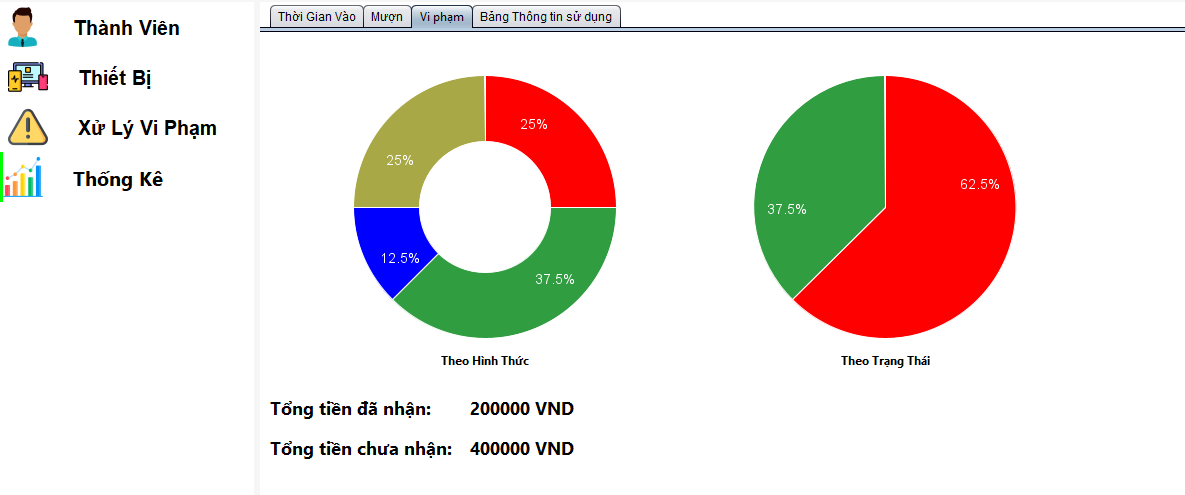
**Xử lý 3: Thống kê vi phạm**

* Sequence diagram:

A diagram of a diagram

Description automatically generated with medium confidence

* Giao diện



* Code

**- DAL**

**- XuLyViPhamDAL**

**+** Hình thức

public List<Object[]> getThongKeHinhThucXL() {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT HinhThucXL, COUNT(\*) AS Tong FROM XuLyViPhamDTO GROUP BY HinhThucXL";

            Query<Object[]> query = session.createQuery(queryVao);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return null;

    }

**+** Trạng thái

public List<Object[]> getThongKeTrangThaiXL() {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT TrangThaiXL, COUNT(\*) AS Tong FROM XuLyViPhamDTO GROUP BY TrangThaiXL";

            Query<Object[]> query = session.createQuery(queryVao);

            List<Object[]> results = query.getResultList();

            return results;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return null;

    }

**+** Tổng tiền đã xử lý

public long getThongKeTongTienDaXL() {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT SUM(SoTien) AS TongSoTien FROM XuLyViPhamDTO WHERE TrangThaiXL=1";

            Query<Object[]> query = session.createQuery(queryVao);

            Object result = query.uniqueResult();

            return (long) result;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return 0;

    }

**+** Tổng tiền chưa xử lý

public long getThongKeTongTienChuaXL() {

        try (Session session = factory.openSession()) {

            String queryVao = "SELECT SUM(SoTien) AS TongSoTien FROM XuLyViPhamDTO WHERE TrangThaiXL=0";

            Query<Object[]> query = session.createQuery(queryVao);

            Object result = query.uniqueResult();

            return (long) result;

        } catch (Exception e) {

            e.printStackTrace();

        }

        return 0;

    }

**- BLL**

**- XuLyViPhamBLL**

**+** Hình thức

public List<Object[]> getThongKeHinhThucXL() {

        return xlDAL.getThongKeHinhThucXL();

    }

**+** Trạng thái

    public List<Object[]> getThongKeTrangThaiXL() {

        return xlDAL.getThongKeTrangThaiXL();

    }

**+** Tổng tiền đã xử lý

public long getThongKeTongTienDaXL() {

        return xlDAL.getThongKeTongTienDaXL();

    }

**+** Tổng tiền chưa xử lý

    public long getThongKeTongTienChuaXL() {

        return xlDAL.getThongKeTongTienChuaXL();

    }

**- GUI**

**+ Theo hình thức**

private void drawPieChart3() {

        pieChart3.clearData();

        int indexx = 0;

        for (Object[] t: xlBLL.getThongKeHinhThucXL()) {

            pieChart3.addData(new ModelPieChart(t[0].toString(), (long) t[1], COLOR\_OPTIONS[indexx]));

            indexx++;

        }

    }

**+ Theo trạng thái**

private void drawPieChart4() {

        pieChart4.clearData();

        int indexx = 0;

        for (Object[] t: xlBLL.getThongKeTrangThaiXL()) {

            String trangthai = t[0].toString().equals("0") ? "Chưa xử lý" : "Đã xử lý";

            pieChart4.addData(new ModelPieChart(trangthai, (long) t[1], COLOR\_OPTIONS[indexx]));

            indexx++;

        }

    }

**+ Theo trạng thái**

 jLabel6.setText(xlBLL.getThongKeTongTienDaXL() + " VND");

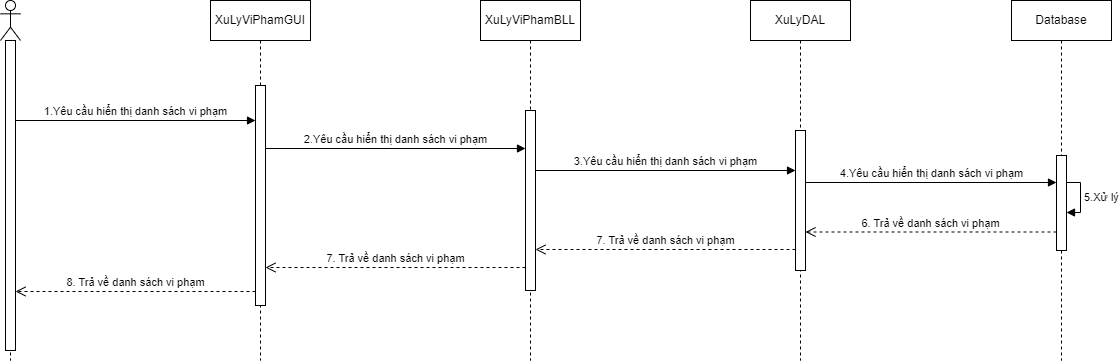
**+ Theo trạng thái**

 jLabel8.setText(xlBLL.getThongKeTongTienChuaXL()+ " VND");

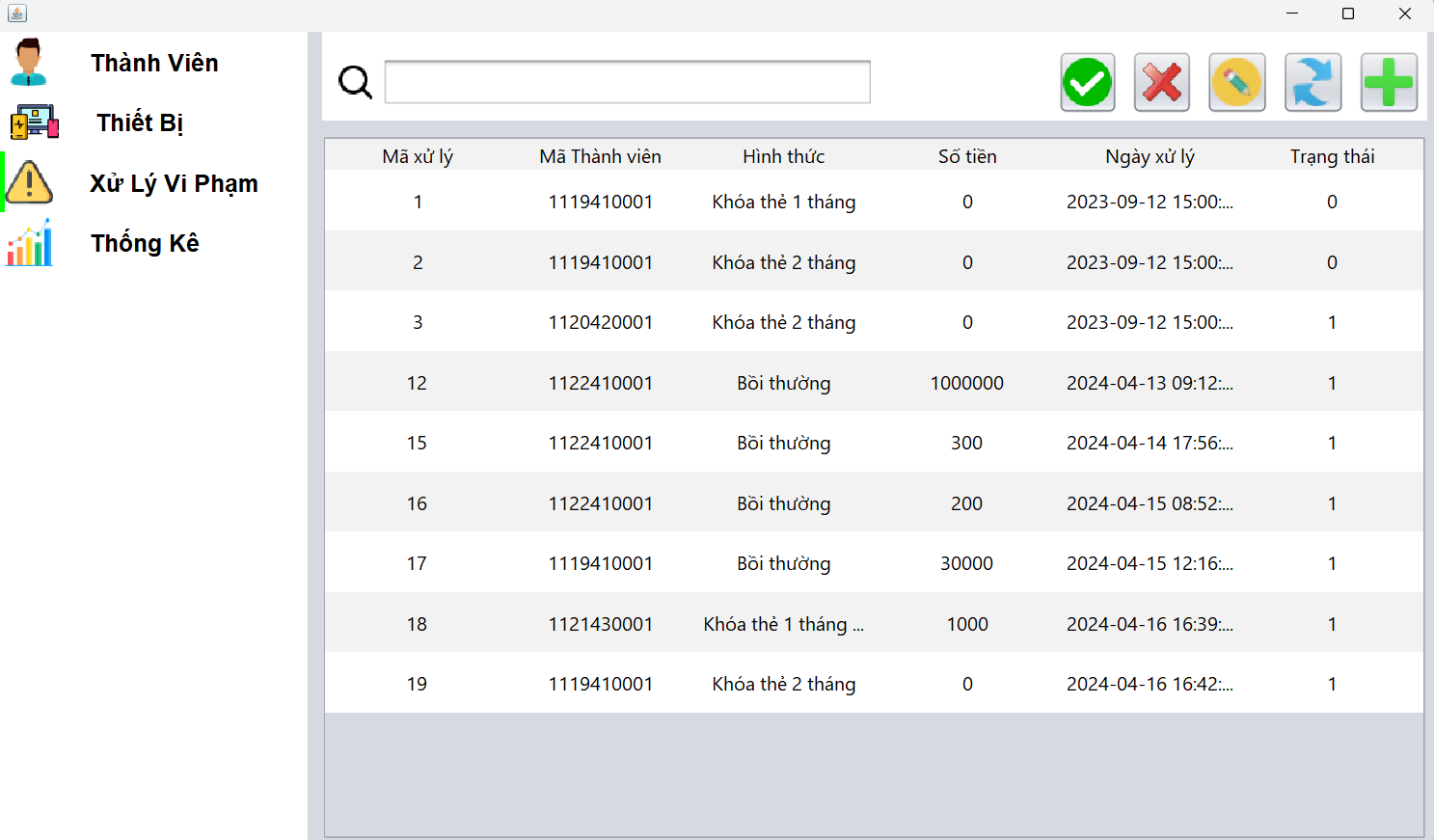
**Chức năng 4: Xử lý vi phạm**

**Xử lý 1: Hiển thị danh sách vi phạm**

* Sequence diagram



* Giao diện



* Code
* **DAL**

     public ArrayList<XuLyViPhamDTO> listXuLyViPham() {

        Session session = factory.openSession();

        ArrayList<XuLyViPhamDTO> list = new ArrayList<>();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            list = (ArrayList<XuLyViPhamDTO>) session.createQuery("FROM XuLyViPhamDTO").list();

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

        return list;

    }

* **BLL**

    public ArrayList<XuLyViPhamDTO> search(String s){

        return xlDAL.search(s);

    }

* **GUI**

    public void loadXuLyViPham(){

        arrXuLy = xlvpBLL.listxlvp();

        modelXL.setRowCount(0);

        for(int i = 0; i < arrXuLy.size(); i++){

            XuLyViPhamDTO em = arrXuLy.get(i);

            int idXL = em.getMaXL();

            ThanhVienDTO thanhvien = em.getMaTV();

            String hinhthuc = em.getHinhThucXL();

            int sotien = em.getSoTien();

            Date ngayxl = em.getNgayXL();

            int trangthai = em.getTrangThaiXL();

            Object[] row = {idXL, thanhvien.getMaTV(), hinhthuc, sotien, ngayxl, trangthai};

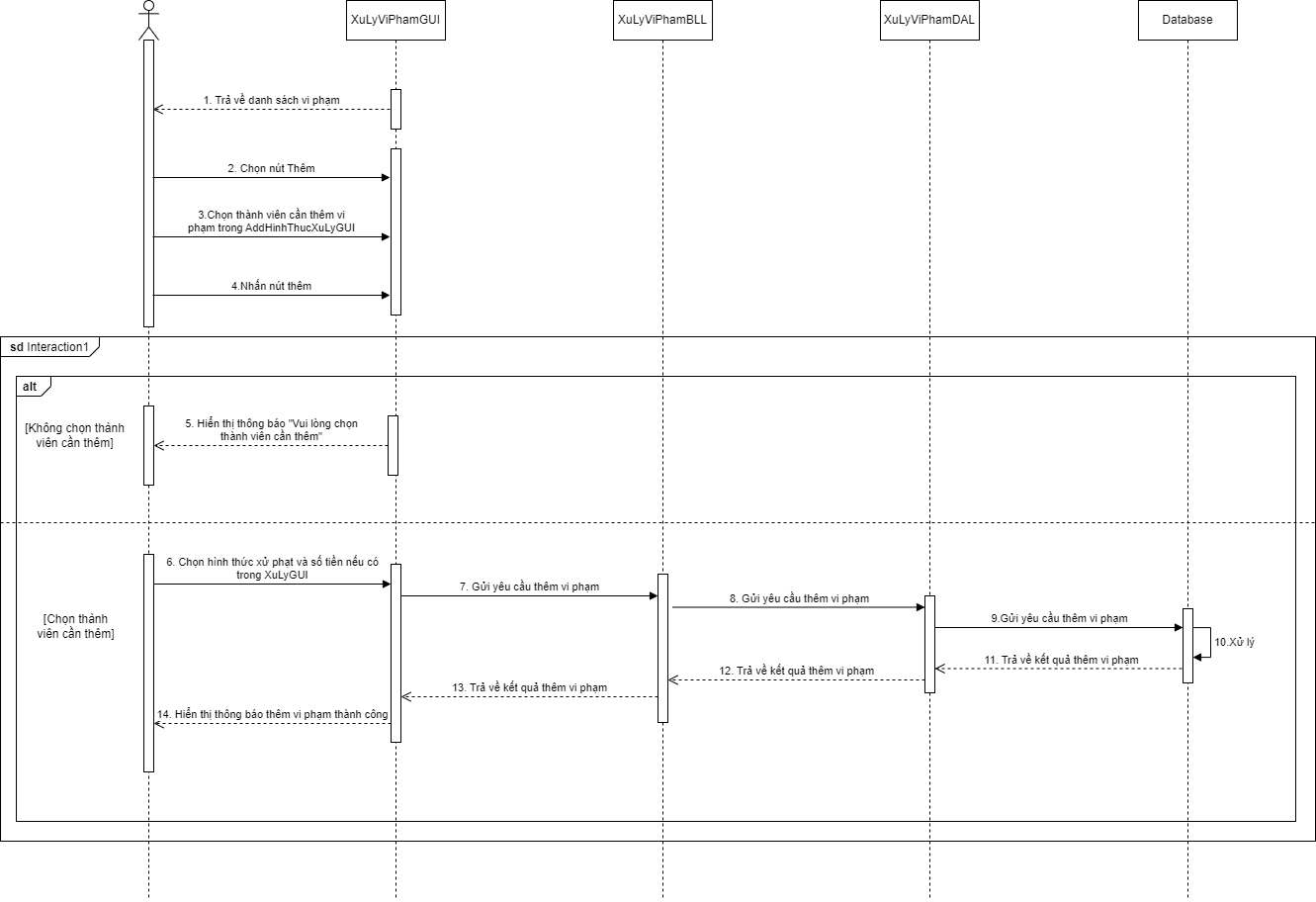
            modelXL.addRow(row);

        }

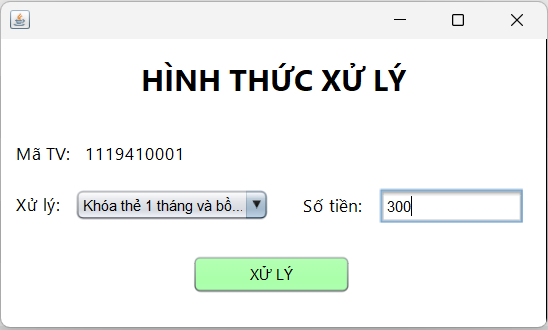
    }

**Xử lý 2: Thêm vi phạm**

* Sequence diagram



* Giao diện



* Code
* **DAL**

    public void themXuLyViPham(XuLyViPhamDTO xuLyViPham) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            session.save(xuLyViPham);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL**

    public void themXuLyViPham(XuLyViPhamDTO xuLyViPham) {

        xlDAL.themXuLyViPham(xuLyViPham);

        JOptionPane.showMessageDialog(null, "Thêm vi phạm thành công.");

    }

* **GUI**

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        ThanhVienDTO thanhVienDTO = xuLyViPham.getMaTV();

        int maThanhVien = Integer.parseInt(jLabel3.getText());

        if (thanhVienDTO == null) {

            thanhVienDTO = new ThanhVienDTO();

        }

        thanhVienDTO.setMaTV(maThanhVien);

        xuLyViPham.setMaTV(thanhVienDTO);

        String hinhThucXuLy = (String) jComboBox1.getSelectedItem();

        int soTien = jTextField1.getText().isEmpty() ? 0 : Integer.parseInt(jTextField1.getText());

        int trangThaiXuLy = 0;

        Date ngayXuLy = new Date();

        xuLyViPham.setMaXL(0);

        xuLyViPham.setHinhThucXL(hinhThucXuLy);

        xuLyViPham.setSoTien(soTien);

        xuLyViPham.setNgayXL(ngayXuLy);

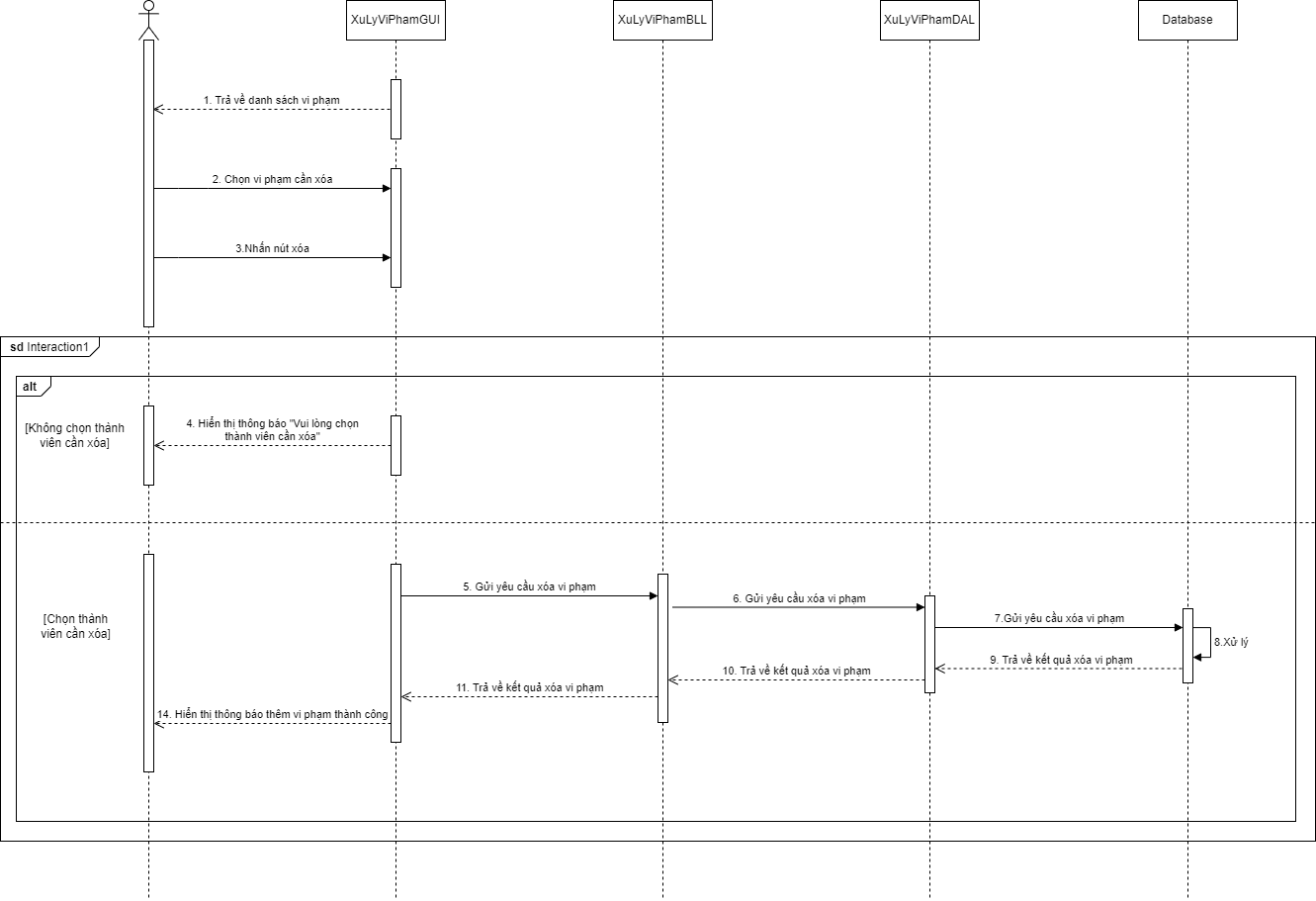
        xuLyViPham.setTrangThaiXL(trangThaiXuLy);

        xuLyViPhamBLL.themXuLyViPham(xuLyViPham);

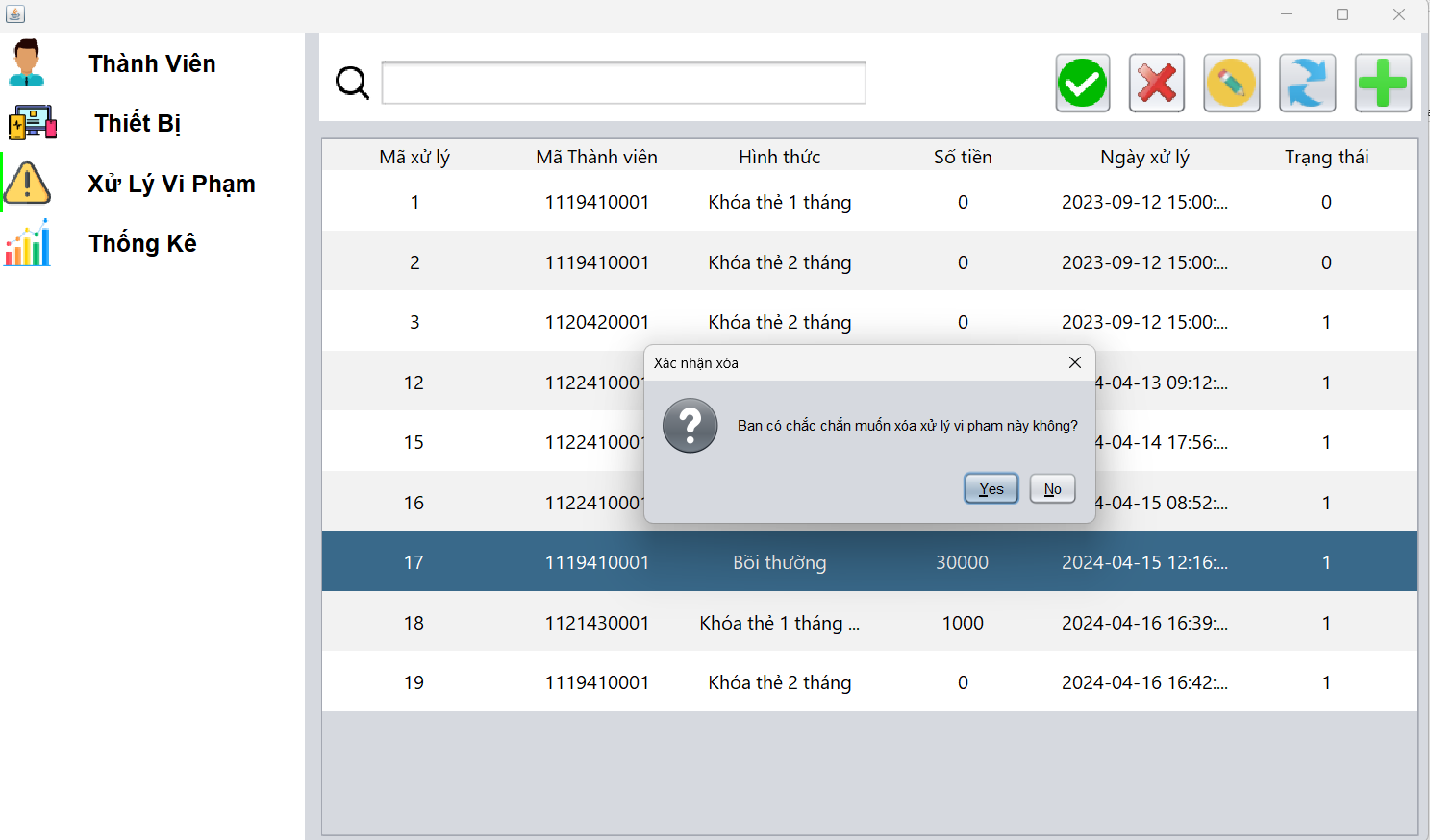
    }

**Xử lý 3: Xóa vi phạm**

* Sequence diagram



* Giao diện



* Code
* **DAL**

    public void xoaXuLyViPham(int id) {

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            XuLyViPhamDTO xlvp = (XuLyViPhamDTO) session.get(XuLyViPhamDTO.class, id);

            session.delete(xlvp);

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) tx.rollback();

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL**

    public void xoaXuLyViPham(int id) {

        xlDAL.xoaXuLyViPham(id);

    }

* **GUI**

    private void jButton\_DeleteActionPerformed(java.awt.event.ActionEvent evt) {

        int selectedRow = jTable\_XuLy.getSelectedRow();

        if (selectedRow == -1) {

            JOptionPane.showMessageDialog(this, "Vui lòng chọn một hàng để xóa.", "Thông báo", JOptionPane.WARNING\_MESSAGE);

            return;

        }

        int maXL = (int) jTable\_XuLy.getValueAt(selectedRow, 0);

        int choice = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa xử lý vi phạm này không?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

        if (choice == JOptionPane.YES\_OPTION) {

            id = (int) jTable\_XuLy.getValueAt(selectedRow, 0);

            xlvpBLL.xoaXuLyViPham(id);

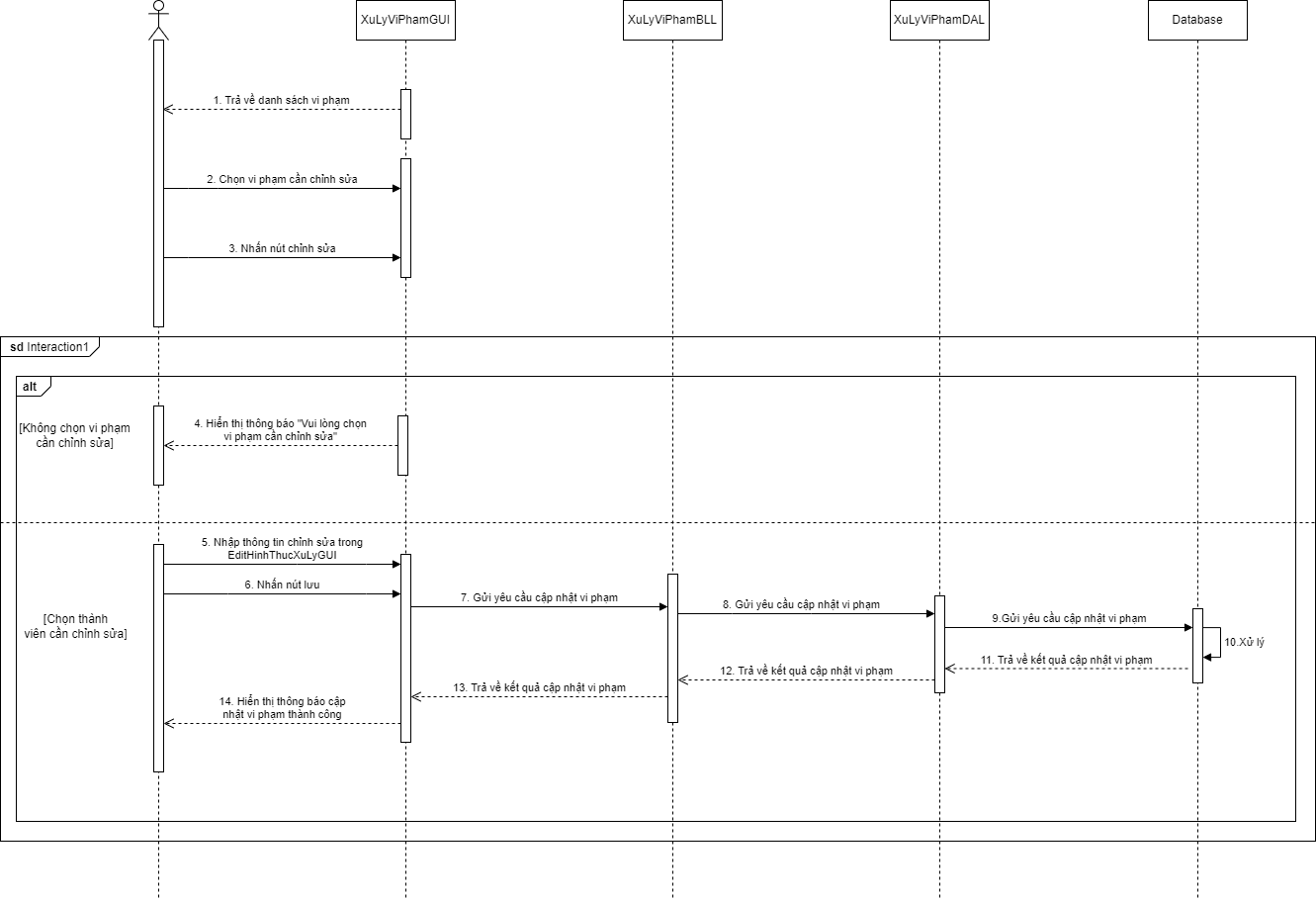
            loadXuLyViPham();

        }

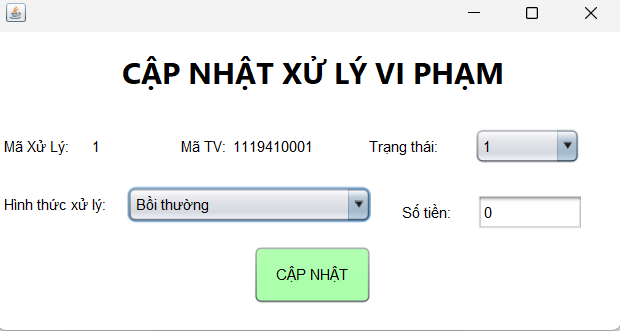
    }

**Xử lý 4: Chỉnh sửa vi phạm**

* Sequence diagram



* Giao diện



* Code
* **DAL**

    public void updateXuLyViPham(XuLyViPhamDTO xuLyViPhamDTO) {

        Session session = factory.openSession();

        Transaction transaction = null;

        try {

            transaction = session.beginTransaction();

            XuLyViPhamDTO existingXuLyViPham = session.get(XuLyViPhamDTO.class, xuLyViPhamDTO.getMaXL());

            if (existingXuLyViPham != null) {

                existingXuLyViPham.setHinhThucXL(xuLyViPhamDTO.getHinhThucXL());

                existingXuLyViPham.setSoTien(xuLyViPhamDTO.getSoTien());

                existingXuLyViPham.setTrangThaiXL(xuLyViPhamDTO.getTrangThaiXL());

                session.update(existingXuLyViPham);

                transaction.commit();

            } else {

                System.out.println("Không tìm thấy đối tượng cần cập nhật.");

            }

        } catch (Exception e) {

            if (transaction != null) {

                transaction.rollback();

            }

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL**

    public void updateXuLyViPham(XuLyViPhamDTO xuLyViPhamDTO) {

        xlDAL.updateXuLyViPham(xuLyViPhamDTO);

    }

* **GUI**

    private void jButton\_UpdateActionPerformed(java.awt.event.ActionEvent evt) {

        String hinhThuc = (String) jComboBox1.getSelectedItem();

        String trangthai = (String) jComboBox2.getSelectedItem();

        int trangthaixl = Integer.parseInt(trangthai);

        int soTien = Integer.parseInt(jTextField\_SoTien.getText());

        XuLyViPhamDTO xuLyViPhamDTO = new XuLyViPhamDTO();

        xuLyViPhamDTO.setMaXL(maXuLy);

        xuLyViPhamDTO.setHinhThucXL(hinhThuc);

        xuLyViPhamDTO.setSoTien(soTien);

        xuLyViPhamDTO.setTrangThaiXL(trangthaixl);

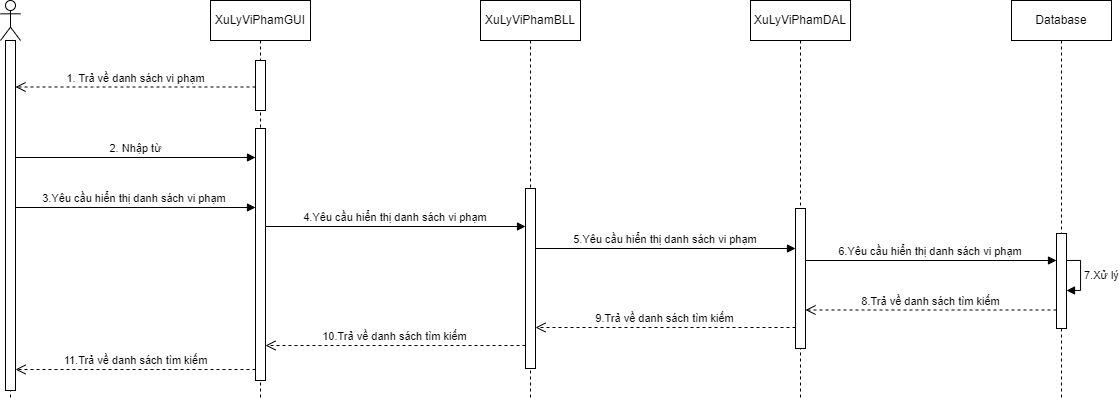
        xuLyViPhamBLL.updateXuLyViPham(xuLyViPhamDTO);

        JOptionPane.showMessageDialog(this, "Cập nhật thành công");

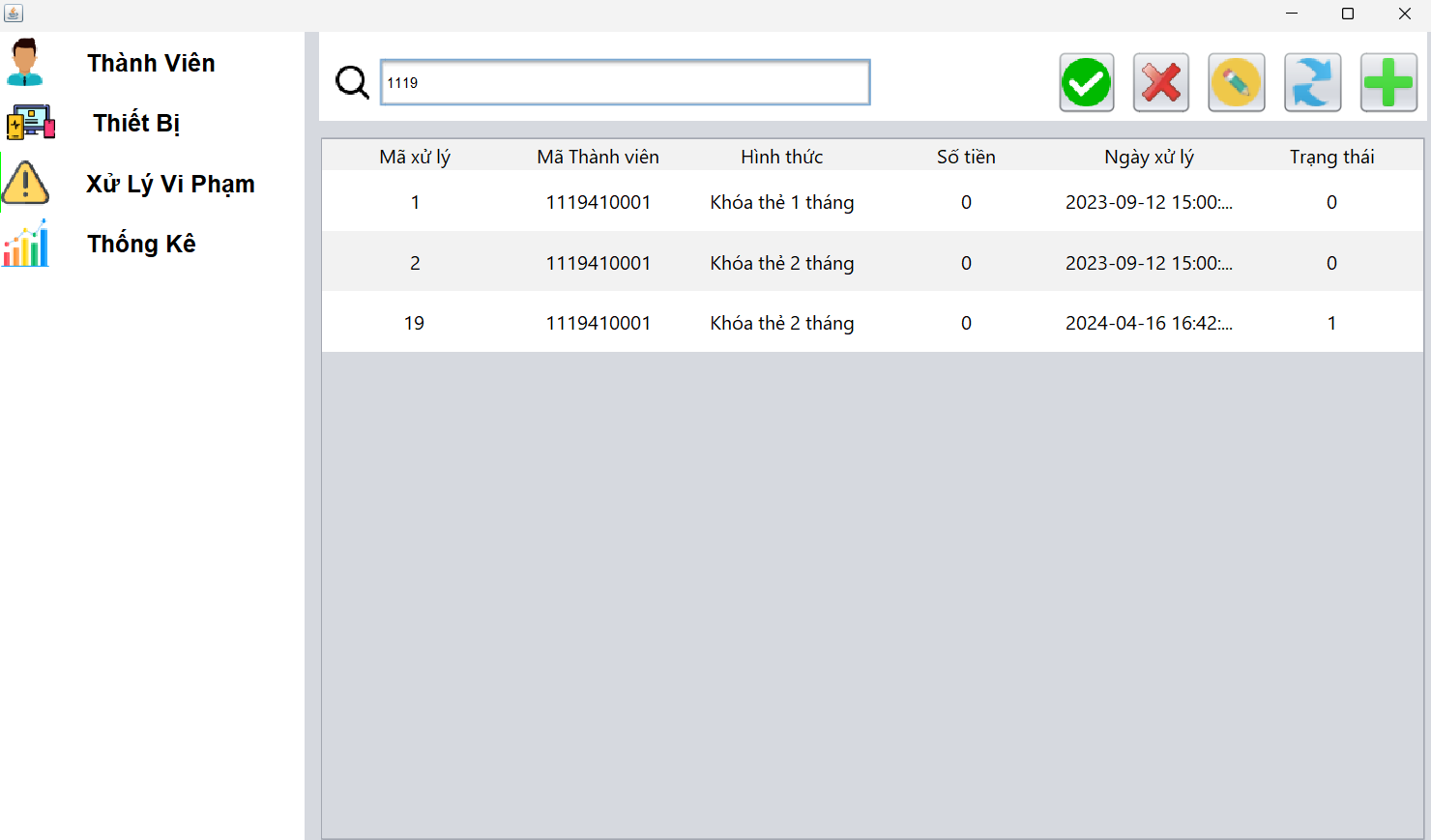
    }

**Xử lý 5: Tìm kiếm vi phạm**

* Sequence diagram



* Giao diện



* Code
* **DAL**

    public ArrayList<XuLyViPhamDTO> search(String s) {

        ArrayList<XuLyViPhamDTO> list = new ArrayList<>();

        Session session = factory.openSession();

        Transaction tx = null;

        try {

            tx = session.beginTransaction();

            String hql = "FROM XuLyViPhamDTO WHERE CAST(MaXL AS string) LIKE :keyword OR MaTV LIKE :keyword OR HinhThucXL LIKE :keyword";

            Query<XuLyViPhamDTO> query = session.createQuery(hql, XuLyViPhamDTO.class);

            query.setParameter("keyword", "%" + s + "%");

            list = (ArrayList<XuLyViPhamDTO>) query.getResultList();

            tx.commit();

        } catch (HibernateException e) {

            if (tx != null) {

                tx.rollback();

            }

            e.printStackTrace();

        } finally {

            session.close();

        }

        return list;

    }

* **BLL**

    public void updateXuLyViPham(XuLyViPhamDTO xuLyViPhamDTO) {

        xlDAL.updateXuLyViPham(xuLyViPhamDTO);

    }

* **GUI**

    private void jTextField\_SearchKeyReleased(java.awt.event.KeyEvent evt) {

        String searchText = jTextField\_Search.getText().trim();

        ArrayList<XuLyViPhamDTO> searchResult = xlvpBLL.search(searchText);

        for(int i = modelXL.getRowCount()-1;i>=0;i--)

        modelXL.removeRow(i);

        for(int i = 0; i<searchResult.size();i++){

            XuLyViPhamDTO em= searchResult.get(i);

            int idXL = em.getMaXL();

            ThanhVienDTO thanhvien = em.getMaTV();

            String hinhthuc = em.getHinhThucXL();

            int sotien = em.getSoTien();

            Date ngayxl = em.getNgayXL();

            int trangthai = em.getTrangThaiXL();

            Object[] row = {idXL, thanhvien.getMaTV(), hinhthuc, sotien, ngayxl, trangthai};

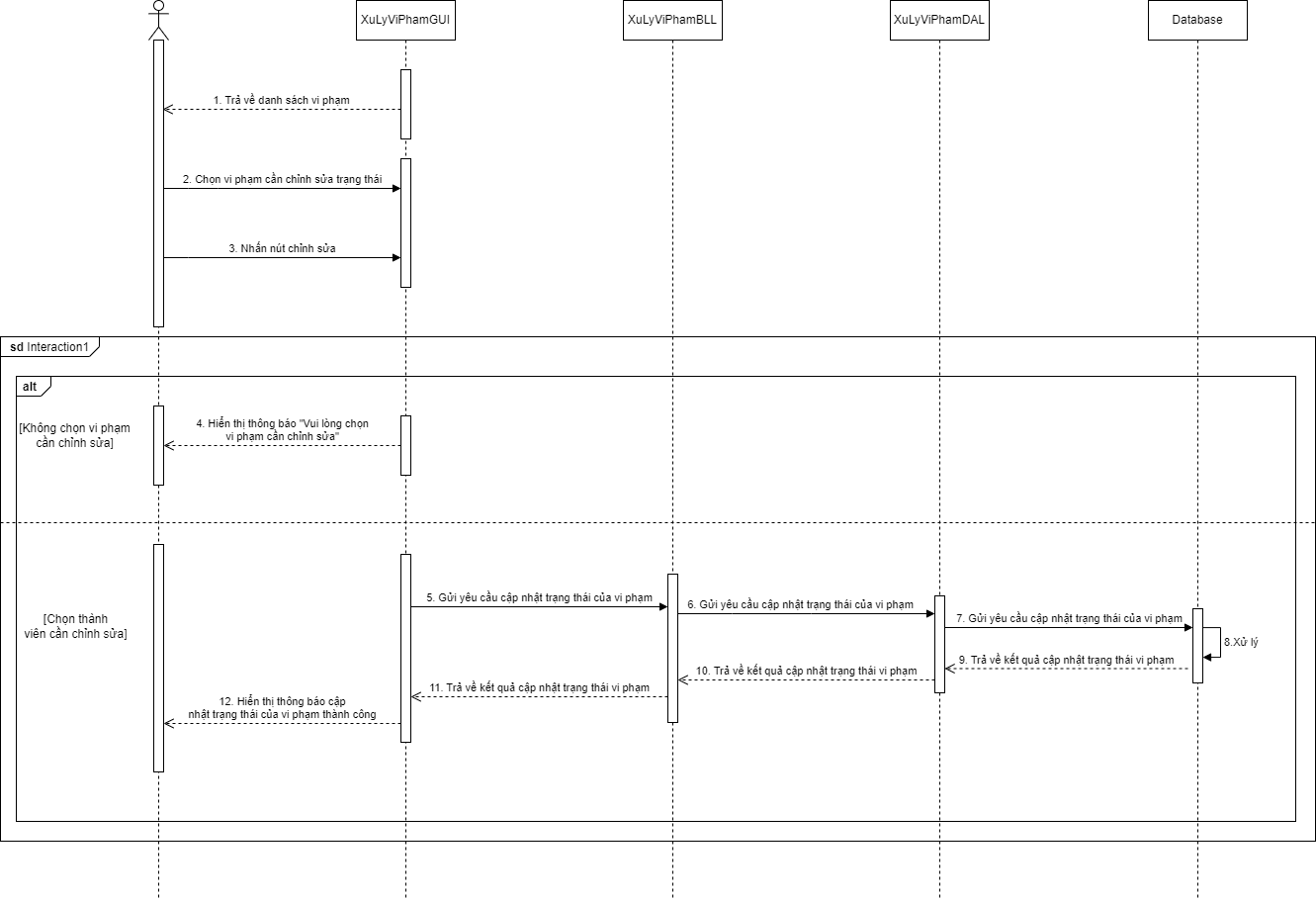
            modelXL.addRow(row);

        }

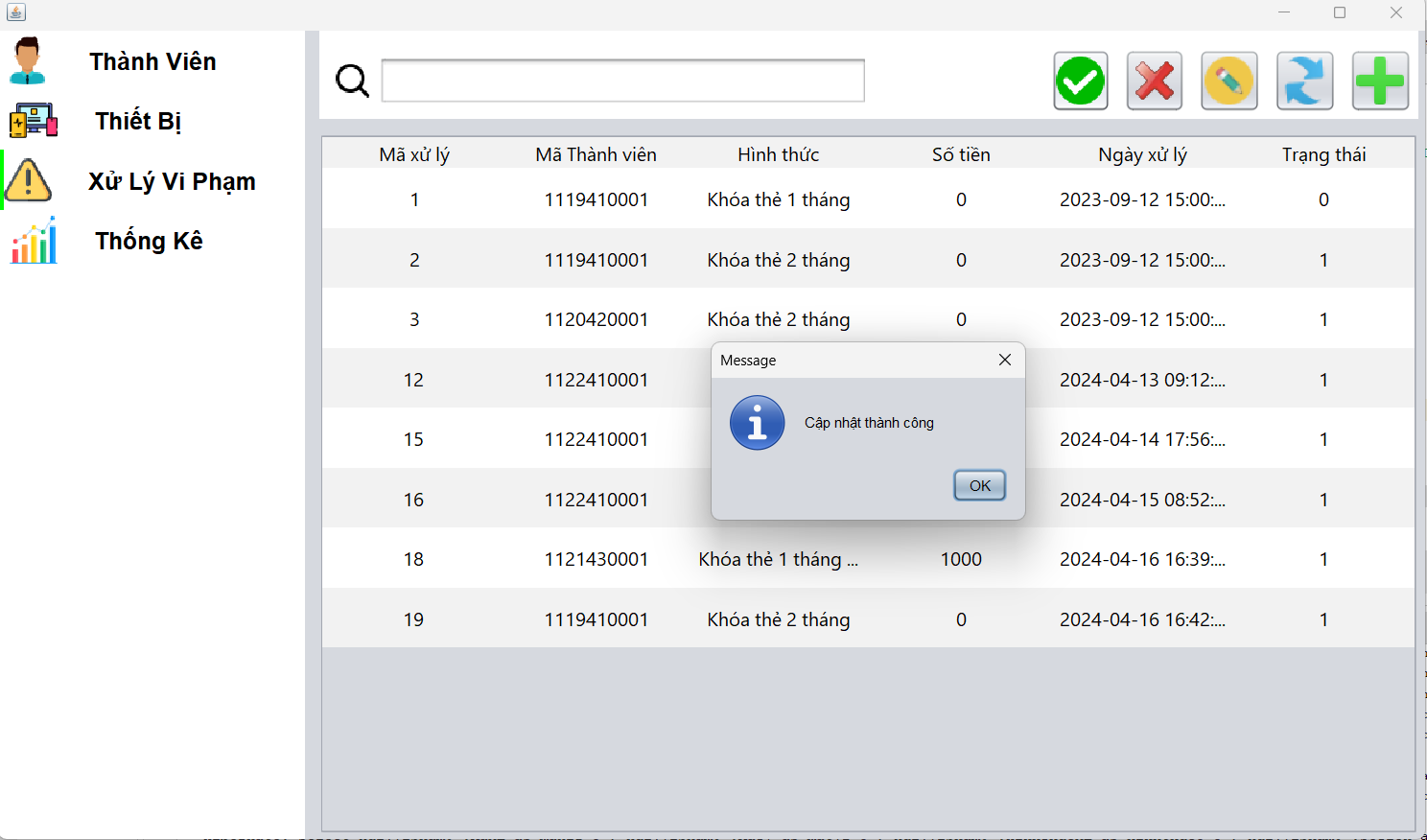
    }

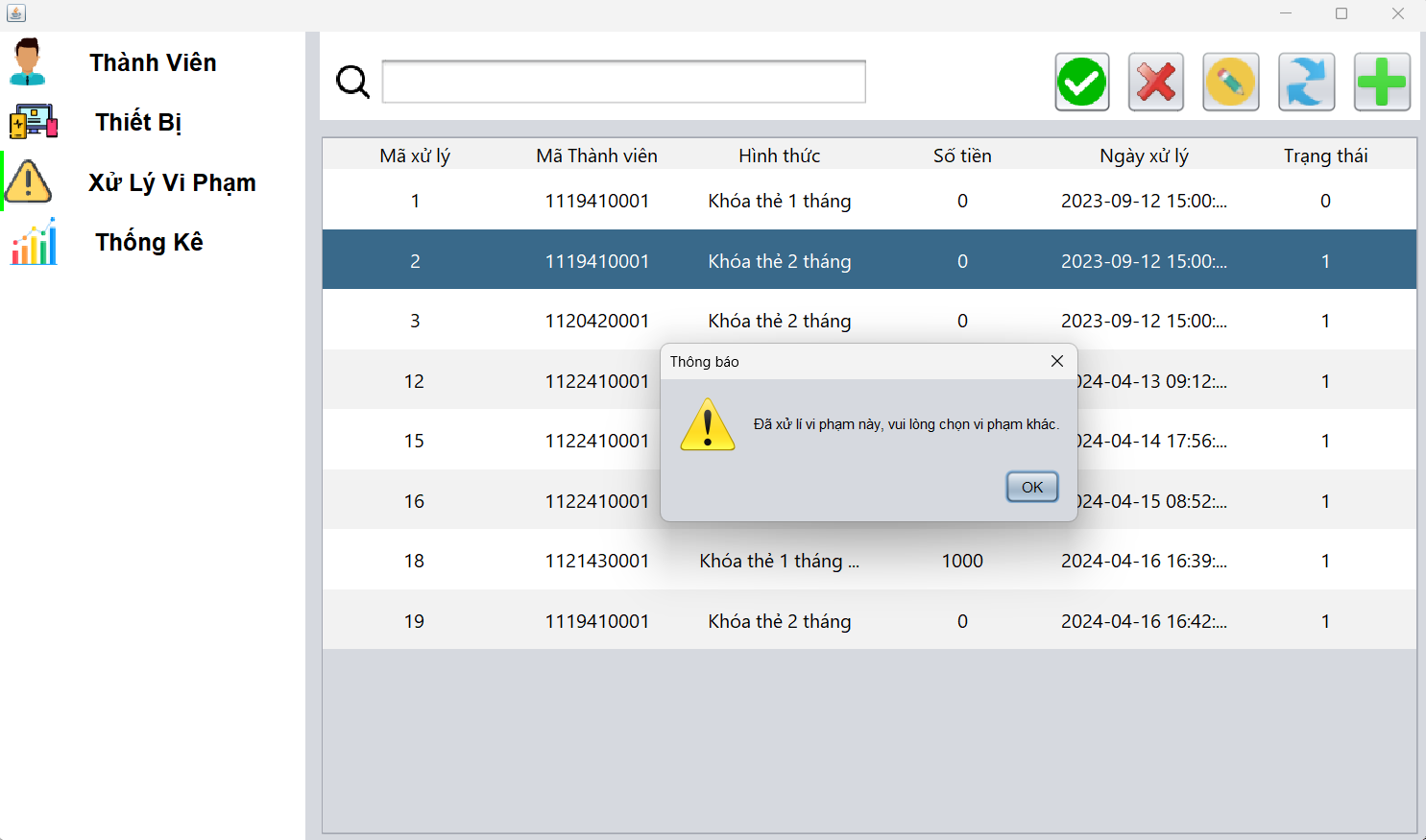
**Xử lý 6: Cập nhật trạng thái của vi phạm**

* Sequence diagram



* Giao diện





* Code
* **DAL**

    public void updateXuLyTrangThai(XuLyViPhamDTO xuLyViPhamDTO) {

        Session session = factory.openSession();

        Transaction transaction = null;

        try {

            transaction = session.beginTransaction();

            XuLyViPhamDTO existingXuLyViPham = session.get(XuLyViPhamDTO.class, xuLyViPhamDTO.getMaXL());

            if (existingXuLyViPham != null) {

                existingXuLyViPham.setTrangThaiXL(xuLyViPhamDTO.getTrangThaiXL());

                session.update(existingXuLyViPham);

                transaction.commit();

            } else {

                System.out.println("Không tìm thấy đối tượng cần cập nhật.");

            }

        } catch (Exception e) {

            if (transaction != null) {

                transaction.rollback();

            }

            e.printStackTrace();

        } finally {

            session.close();

        }

    }

* **BLL**

    public void updateXuLyTrangThai(XuLyViPhamDTO xuLyViPhamDTO) {

        xlDAL.updateXuLyTrangThai(xuLyViPhamDTO);

    }

* **GUI**

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        int selectedRow = jTable\_XuLy.getSelectedRow();

        if (selectedRow == -1) {

            JOptionPane.showMessageDialog(this, "Vui lòng chọn một hàng để xóa.", "Thông báo", JOptionPane.WARNING\_MESSAGE);

            return;

        }

        int maXL = (int) jTable\_XuLy.getValueAt(selectedRow, 0);

        int trangthai = (int) jTable\_XuLy.getValueAt(selectedRow, 5);

        if (trangthai == 1){

            JOptionPane.showMessageDialog(this, "Đã xử lí vi phạm này, vui lòng chọn vi phạm khác.", "Thông báo", JOptionPane.WARNING\_MESSAGE);

        }else{

            int trangthaixl = 1;

        XuLyViPhamDTO xuLyViPhamDTO = new XuLyViPhamDTO();

        xuLyViPhamDTO.setMaXL(maXL);

        xuLyViPhamDTO.setTrangThaiXL(trangthaixl);

        xlvpBLL.updateXuLyTrangThai(xuLyViPhamDTO);

        loadXuLyViPham();

        JOptionPane.showMessageDialog(this, "Cập nhật thành công");

    }

HẾT