**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------🙢🕮🙠---------------

**ĐỒ ÁN MÔN HỌC XÂY DỰNG PHẦN MỀM**

**THEO MÔ HÌNH PHÂN LỚP**

**ĐỀ TÀI: QUẢN LÝ KHÓA HỌC**

**NHÓM 11**

**THÀNH VIÊN:**

3121410353 – Trần Khôi Nguyên

3121410178 – Trịnh Văn Hải

3121410383 – Nguyễn Thanh Phong

3121410494 – Bùi Văn Tiến

3121410497 – Huỳnh Quốc Tiến

3121410549 – Nguyễn Thanh Tú

*Giảng viên:*

*Tp. Hồ chí minh, Ngày …. tháng …. năm ….*

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**MỤC LỤC**

**[CHỨC NĂNG 1: QUẢN LÝ THÔNG TIN CÁC KHÓA HỌC ONLINE, ONSITE 3](#_Toc11685)**

[1.1 Xử lý 1: hiển thị danh sách 3](#_Toc21586)

[1.2 Xử lý 2: hiển thị chi tiết 1 record (dòng) 6](#_Toc532)

[1.3 Xử lý 3: thêm mới thông tin 11](#_Toc7842)

[1.4 Xử lý 4: sửa/ cập nhật thông tin 18](#_Toc30186)

[1.5 Xử lý 5: tìm kiếm 27](#_Toc19950)

[1.6 Xử lý 6: xóa record 32](#_Toc15935)

**[CHỨC NĂNG 2: QUẢN LÝ THÔNG TIN PHÂN CÔNG GIẢNG DẠY (COURSEINTRUCTOR) 35](#_Toc3685)**

[2.1 Xử lý 1: hiển thị danh sách 35](#_Toc23572)

[2.2 Xử lý 2: hiển thị chi tiết 1 record (dòng) 39](#_Toc5199)

[2.3 Xử lý 3: thêm mới thông tin 43](#_Toc13497)

[2.4 Xử lý 4: sửa/ cập nhật thông tin 49](#_Toc12631)

[2.5 Xử lý 5: tìm kiếm 55](#_Toc13919)

[2.6 Xử lý 6: xóa record 59](#_Toc14675)

**[CHỨC NĂNG 3: QUẢN LÝ KẾT QUẢ KHÓA HỌC (STUDENTGRADE) 63](#_Toc12862)**

[3.1 Xử lý 1: hiển thị danh sách 63](#_Toc23124)

[3.2 Xử lý 2: hiển thị chi tiết 1 record (dòng) 66](#_Toc8446)

[3.3 Xử lý 3: thêm mới thông tin 67](#_Toc27120)

[3.4 Xử lý 4: sửa/ cập nhật thông tin 73](#_Toc27451)

[3.5 Xử lý 5: tìm kiếm 78](#_Toc31273)

[3.6 Xử lý 6: xóa record 81](#_Toc24416)

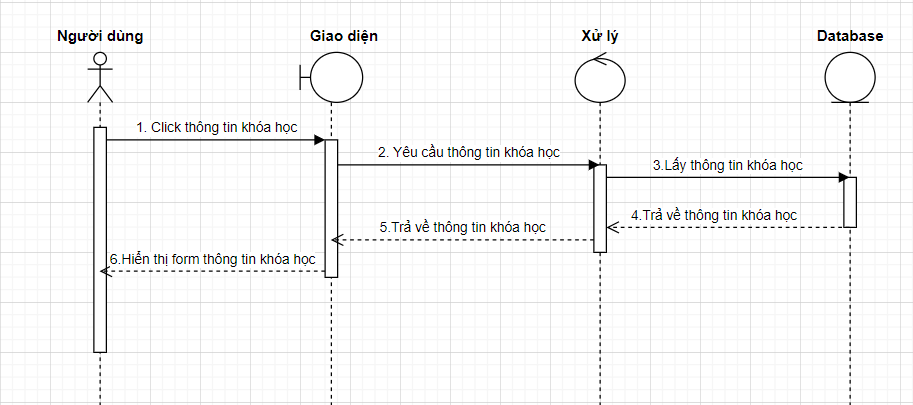
**Phân công việc làm và mức độ đóng góp:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành viên** | **Phần trăm tham gia hoàn thành đồ án (tổng 100%)** | **Công việc** |
| 1 | Trần Khôi Nguyên | 16.7% | Thêm, sửa, tìm kiếm khóa học |
| 2 | Nguyễn Thanh Phong | 16.7% | Thêm, xóa phân công |
| 3 | Trịnh Văn Hải | 16.7% | Hiển thị, hiển thị chi tiết, xóa khóa học |
| 4 | Nguyễn Thanh Tú | 16.7% | Thiết kế giao diện |
| 5 | Huỳnh Quốc Tiến | 16.7% | Hiển thị, hiển thị chi tiết, sửa, tìm kiếm phân công |
| 6 | Bùi Văn Tiến | 16.7% | Quản lý kết quả học tập |

**CHỨC NĂNG 1: QUẢN LÝ THÔNG TIN CÁC KHÓA HỌC ONLINE, ONSITE**

## **1.1 Xử lý 1:** **hiển thị danh sách**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**CourseDAL**

 public ArrayList<Course> getList() {

        ArrayList list = new ArrayList<Course>();

        String query = "select \*from Course";

        try {

            PreparedStatement pst = c.prepareStatement(query);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                Course value = new Course();

                value.setCourseId(rs.getInt(1));

                value.setTitle(rs.getString(2));

                value.setCredits(rs.getInt(3));

                value.setDepartmentId(rs.getInt(4));

                list.add(value);

            }

        } catch (SQLException ex) {

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

        }

        return list;

    }

**OnlineCourseDAL**

public ArrayList<OnlineCourse> getList() {

        ArrayList list = new ArrayList<OnlineCourse>();

        String query = "select \*from onlinecourse";

        try {

            PreparedStatement pst = c.prepareStatement(query);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                OnlineCourse value = new OnlineCourse();

                value.setCourseId(rs.getInt(1));

                value.setUrl(rs.getString(2));

                list.add(value);

            }

        } catch (SQLException ex) {

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

        }

        return list;

    }

**OnsiteCourseDAL**

public ArrayList<OnsiteCourse> getList() {

        ArrayList list = new ArrayList<OnsiteCourse>();

        String query = "select \*from onsitecourse";

        try {

            PreparedStatement pst = c.prepareStatement(query);

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                OnsiteCourse value = new OnsiteCourse();

                value.setCourseId(rs.getInt(1));

                value.setLocation(rs.getString(2));

                value.setDays(rs.getString(3));

                value.setTime(rs.getString(4));

                list.add(value);

            }

        } catch (SQLException ex) {

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

        }

        return list;

    }

**CourseBLL**

 public ArrayList<Course> getList() {

        return courseDAL.getList();

    }

**OnlineCourseBLL**

public ArrayList<OnlineCourse> getList() {

        return onlineCourseDAL.getList();

    }

**OnsiteCourseBLL**

public ArrayList<OnsiteCourse> getList() {

        return onsiteCourseDAL.getList();

    }

**CourseGUI**

public void loadToTable(ArrayList<Course> list) {

        String[] header = {"Mã khóa học", "Tên khóa học", "Tín chỉ", "Mã phòng", "Phương thức", ""};

        model = new DefaultTableModel(header, 0);

        this.jTable1.setModel(model);

        for (Course x : list) {

            if (!x.getTitle().equals("null")) {

                boolean flag = false;

                for (OnlineCourse y : onlineCourseBLL.getList()) {

                    if (x.getCourseId() == y.getCourseId()) {

                        Object[] data = {x.getCourseId(), x.getTitle(), x.getCredits(), x.getDepartmentId(), "Trực tuyến", "Click để xem chi tiết..."};

                        model.addRow(data);

                        flag = true;

                        break;

                    }

                }

                if (flag) {

                    continue;

                }

                for (OnsiteCourse y : onsiteCourseBLL.getList()) {

                    if (x.getCourseId() == y.getCourseId()) {

                        Object[] data = {x.getCourseId(), x.getTitle(), x.getCredits(), x.getDepartmentId(), "Tại lớp", "Click để xem chi tiết..."};

                        model.addRow(data);

                        break;

                    }

                }

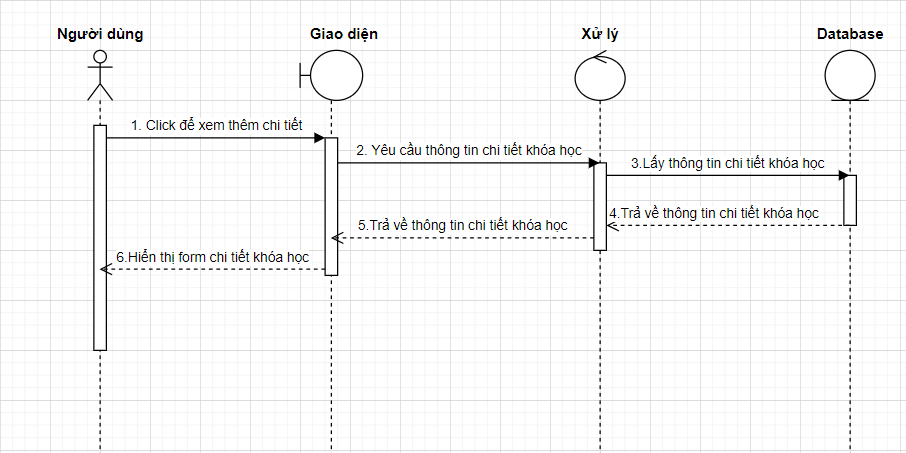
            }

        }

}

## **1.2 Xử lý 2: hiển thị chi tiết 1 record (dòng)**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**





* **Code 3 class: DAL, BLL, GUI**

**CourseGUI**

 private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jTable1MouseClicked

        int col = (int) this.jTable1.getSelectedColumn();

        int row = (int) this.jTable1.getSelectedRow();

        if (col == 5) {

            boolean flag = false;

            for (Course x : courseBLL.getList()) {

                if (x.getCourseId() == (int) this.jTable1.getValueAt(row, 0)) {

                    for (OnlineCourse y : onlineCourseBLL.getList()) {

                        if (x.getCourseId() == y.getCourseId()) {

                            CourseInsert\_Edit courseInsert\_Edit = new CourseInsert\_Edit();

                            courseInsert\_Edit.getjButton1().setVisible(false);

                            courseInsert\_Edit.getjComboBox1().setSelectedIndex(0);

                            courseInsert\_Edit.getjComboBox1().setEnabled(false);

                            courseInsert\_Edit.getjTextFieldId().setText(Integer.toString(x.getCourseId()));

                            courseInsert\_Edit.getjTextFieldId().setEditable(false);

                            courseInsert\_Edit.getjTextFieldTen().setText(x.getTitle());

                            courseInsert\_Edit.getjTextFieldTen().setEditable(false);

                            courseInsert\_Edit.getjTextFieldGia().setText(Integer.toString(x.getCredits()));

                            courseInsert\_Edit.getjTextFieldGia().setEditable(false);

                            courseInsert\_Edit.getjTextFieldUrl().setText(y.getUrl());

                            courseInsert\_Edit.getjTextFieldUrl().setEditable(false);

                            courseInsert\_Edit.getjTextFieldDiaChi().setEnabled(false);

                            courseInsert\_Edit.getjTextFieldGio().setEnabled(false);

                            courseInsert\_Edit.getjTextFieldNgay().setEnabled(false);

                            for(int i=0;i< courseInsert\_Edit.getjComboBox2().getItemCount();i++){

                                String item= (String) courseInsert\_Edit.getjComboBox2().getItemAt(i);

                                String departmentId= Integer.toString(x.getDepartmentId());

                                if(item.equals(departmentId)){

                                    courseInsert\_Edit.getjComboBox2().setSelectedIndex(i);

                                    courseInsert\_Edit.getjComboBox2().setEnabled(false);

                                    break;

                                }

                            }

                            courseInsert\_Edit.setVisible(true);

                            flag = true;

                            break;

                        }

                    }

                    if (flag) {

                        continue;

                    }

                    for (OnsiteCourse y : onsiteCourseBLL.getList()) {

                        if (x.getCourseId() == y.getCourseId()) {

                            CourseInsert\_Edit courseInsert\_Edit = new CourseInsert\_Edit();

                            courseInsert\_Edit.getjButton1().setVisible(false);

                            courseInsert\_Edit.getjComboBox1().setSelectedIndex(1);

                            courseInsert\_Edit.getjComboBox1().setEnabled(false);

                            courseInsert\_Edit.getjTextFieldId().setText(Integer.toString(x.getCourseId()));

                            courseInsert\_Edit.getjTextFieldId().setEditable(false);

                            courseInsert\_Edit.getjTextFieldTen().setText(x.getTitle());

                            courseInsert\_Edit.getjTextFieldTen().setEditable(false);

                            courseInsert\_Edit.getjTextFieldGia().setText(Integer.toString(x.getCredits()));

                            courseInsert\_Edit.getjTextFieldGia().setEditable(false);

                            courseInsert\_Edit.getjTextFieldUrl().setEnabled(false);

                            courseInsert\_Edit.getjTextFieldDiaChi().setText(y.getLocation());

                            courseInsert\_Edit.getjTextFieldDiaChi().setEditable(false);

                            courseInsert\_Edit.getjTextFieldGio().setText(y.getTime().toString());

                            courseInsert\_Edit.getjTextFieldGio().setEditable(false);

                            courseInsert\_Edit.getjTextFieldNgay().setText(y.getDays());

                            courseInsert\_Edit.getjTextFieldNgay().setEditable(false);

                            for(int i=0;i< courseInsert\_Edit.getjComboBox2().getItemCount();i++){

                                String item= (String) courseInsert\_Edit.getjComboBox2().getItemAt(i);

                                String departmentId= Integer.toString(x.getDepartmentId());

                                if(item.equals(departmentId)){

                                    courseInsert\_Edit.getjComboBox2().setSelectedIndex(i);

                                    courseInsert\_Edit.getjComboBox2().setEnabled(false);

                                    break;

                                }

                            }

                            courseInsert\_Edit.setVisible(true);

                            break;

                        }

                    }

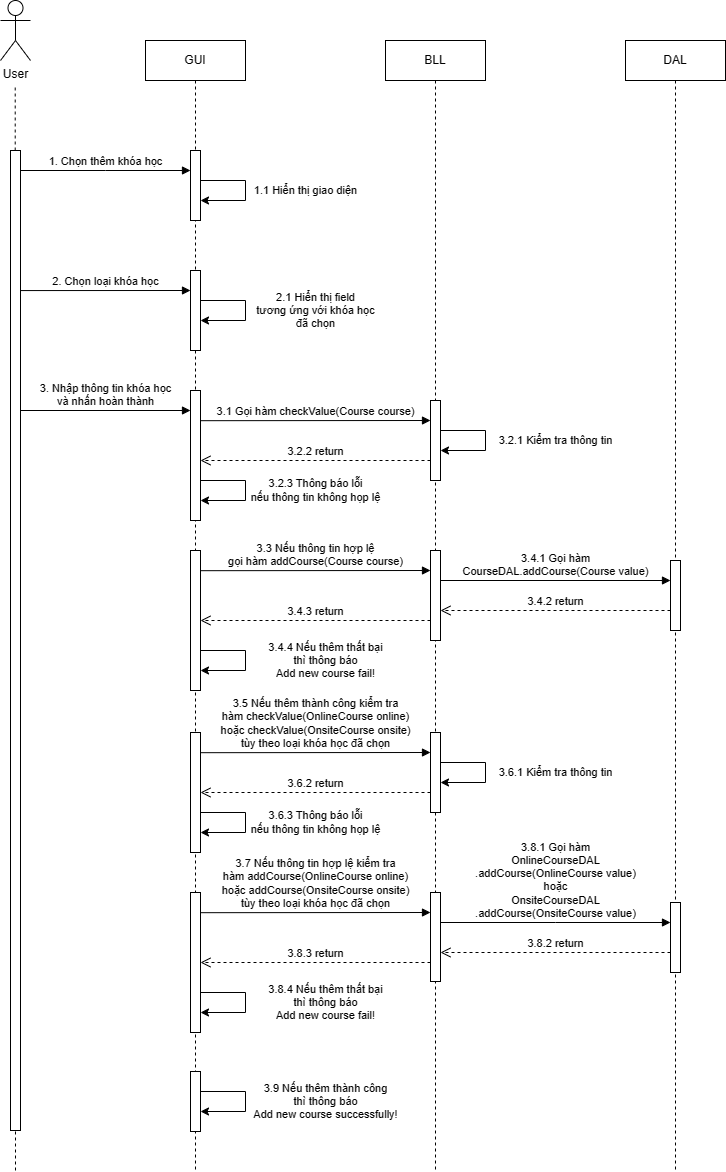
                }

            }

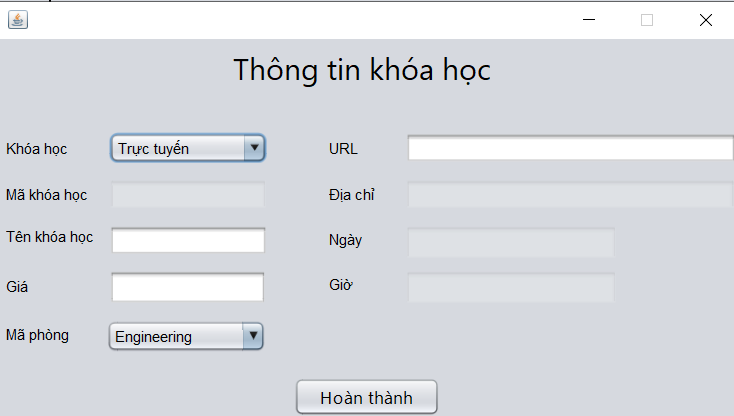
        }

## **1.3 Xử lý 3: thêm mới thông tin**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**CourseDAL**

**public** boolean addCourse(Course **value**) {

**try** {

String query = **"Insert into Course (Title, Credits, DepartmentID ) VALUES(?,?,?)"**;

PreparedStatement pst = c.prepareStatement(query);

pst.setNString(1, **value**.getTitle());

pst.setInt(2, **value**.getCredits());

pst.setInt(3, **value**.getDepartmentId());

**return** pst.executeUpdate() > 0;

} **catch** (SQLException ex) {

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

}

**return** **false**;

}

**public** **int** getLastCourseId() throws SQLException {

String query = **"Select Max(CourseId) From Course"**;

PreparedStatement pst = c.prepareStatement(query);

ResultSet rs = pst.executeQuery();

**if** (rs.next()) {

**int** id = rs.getInt(1);

**return** id;

} **else** {

*// Xử lý trường hợp không có dữ liệu, ví dụ:*

**throw** **new** SQLException(**"No data found"**);

}

}

**OnlineCourseDAL**

**public** boolean addOnlineCourse(OnlineCourse **value**) {

**try** {

String query= **"Insert into onlinecourse (CourseID, url ) VALUES(?,?)"**;

PreparedStatement pst = c.prepareStatement(query);

pst.setInt(1, **value**.getCourseId());

pst.setNString(2, **value**.getUrl());

**return** pst.executeUpdate()> 0;

} **catch** (SQLException ex) {

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

}

**return** **false**;

}

**OnsiteCourseDAL**

**public** boolean addOnsiteCourse(OnsiteCourse **value**) {

**try** {

String query= **"Insert into onsitecourse (CourseID, Location, Days, Time ) VALUES(?,?,?,?)"**;

PreparedStatement pst = c.prepareStatement(query);

pst.setInt(1, **value**.getCourseId());

pst.setNString(2, **value**.getLocation());

pst.setNString(3, **value**.getDays());

pst.setString(4, **value**.getTime());

**return** pst.executeUpdate() > 0;

} **catch** (SQLException ex) {

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

}

**return** **false**;

}

**CourseBLL**

**public** **boolean** addCourse(Course course) {

**return** courseDAL.addCourse(course);

}

**public** **int** getLastCourseId() **throws** SQLException {

**return** courseDAL.getLastCourseId();

}

**public** String checkValue(Course course) {

**if**(**""**.equals(course.getTitle())) {

**return** **"Title not null!"**;

}

**else** **if**(course.getCredits() <= 0 || course.getCredits() > 4) {

**return** **"Credits must be numbers 1 through 4!"**;

}

**return** **"valid"**;

}

**OnlineCourseBLL**

**public** **boolean** addCourse(OnlineCourse online) {

**return** onlineCourseDAL.addOnlineCourse(online);

}

**public** String checkValue(OnlineCourse online) {

**if**(**""**.equals(online.getUrl())) {

**return** **"Url not null!"**;

}

**else** {

**try** {

**new** URL(online.getUrl());

} **catch** (MalformedURLException e) {

**return** **"Invalid url!"**;

}

}

**return** **"valid"**;

}

**OnsiteCourseBLL**

**public** boolean addCourse(OnsiteCourse onsite) {

**return** onsiteCourseDAL.addOnsiteCourse(onsite);

}

**public** String checkValue(OnsiteCourse onsite) {

**if**(**""**.equals(onsite.getLocation())) {

**return** **"location not null!"**;

}

**else** **if**(**""**.equals(onsite.getDays())) {

**return** **"days not null!"**;

}

**else** **if**(!onsite.getDays().matches(**"[MTWHFS]\*"**)) {

**return** **"days must only contain M T W H F S"**;

}

**else** **if**(**""**.equals(onsite.getTime())) {

**return** **"time not null!"**;

}

**else** {

**try** {

DateTimeFormatter formatter = DateTimeFormatter.ofPattern(**"HH:mm:00"**);

LocalTime.parse(onsite.getTime(), formatter);

} **catch** (DateTimeParseException e) {

**return** **"Invalid time!"**;

}

}

**return** **"valid"**;

}

**CourseGUI**

*// TODO add your handling code here:*

CourseInsert\_Edit courseInsert\_Edit = **new** CourseInsert\_Edit();

courseInsert\_Edit.addWindowListener(**new** WindowAdapter() {

@Override

**public** **void** windowClosed(WindowEvent e) {

loadToTable(courseBLL.getList());

}

});

courseInsert\_Edit.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

courseInsert\_Edit.setInsert(**true**);

courseInsert\_Edit.setFlag(**true**);

**try** {

courseInsert\_Edit.getjComboBox1().setSelectedIndex(0);

courseInsert\_Edit.getjComboBox1().setEnabled(**true**);

courseInsert\_Edit.getjTextFieldId().setText(**""**);

courseInsert\_Edit.getjTextFieldId().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldTen().setText(**""**);

courseInsert\_Edit.getjTextFieldTen().setEditable(**true**);

courseInsert\_Edit.getjTextFieldGia().setText(**""**);

courseInsert\_Edit.getjTextFieldGia().setEditable(**true**);

courseInsert\_Edit.getjTextFieldUrl().setText(**""**);

courseInsert\_Edit.getjTextFieldUrl().setEditable(**true**);

courseInsert\_Edit.getjTextFieldDiaChi().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldGio().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldNgay().setEnabled(**false**);

ArrayList<**String**> data = **new** ArrayList<>();

**for**(Department x: departmentBLL.getListDepartment()) {

data.add(x.getName());

}

courseInsert\_Edit.getjComboBox2().setModel(**new** javax.swing.DefaultComboBoxModel<>(data.toArray(**String**[]::**new**)));

courseInsert\_Edit.getjButton1().setVisible(**true**);

courseInsert\_Edit.setVisible(**true**);

JOptionPane.showMessageDialog(**null**, **"Please enter all data!"**);

} **catch** (SQLException ex) {

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

}

**CourseInsert\_EditGUI**

Course course = new Course()*;*

course.setTitle(**jTextFieldTen.getText());** if(!**jTextFieldGia.getText().isEmpty())** {

course.setCredits(Integer.parseInt(**jTextFieldGia.getText()));** }

for(Department x: departmentBLL.getListDepartment()) {

if(x.getName().equals(**jComboBox2.getSelectedItem().toString()))** {

course.setDepartmentId(x.getDepartmentId())*;*

}

}

if(!**"valid"**.equals(courseBLL.checkValue(course))) {

**JOptionPane.showMessageDialog(this,** courseBLL.checkValue(course))*;*

}

else {

if(courseBLL.**addCourse(course))** {

if(**jComboBox1.getSelectedIndex()** == 0) {

OnlineCourse online = new OnlineCourse()*;*

online.setCourseId(courseBLL.getLastCourseId())*;*

online.setUrl(**jTextFieldUrl.getText());** if(!**"valid"**.equals(onlineBLL.checkValue(online))) {

**JOptionPane.showMessageDialog(this,** onlineBLL.checkValue(online))*;*

}

else {

if(onlineBLL.**addCourse(online))** {

**JOptionPane.showMessageDialog(this, "Add new course successfully!"**)*;*

Clear()*;*

}

else {

**JOptionPane.showMessageDialog(this, "Add new course fail!"**)*;*

}

}

}

else {

OnsiteCourse onsite = new OnsiteCourse()*;*

onsite.setCourseId(courseBLL.getLastCourseId())*;*

onsite.setLocation(**jTextFieldDiaChi.getText());** onsite.setDays(**jTextFieldNgay.getText());** onsite.setTime(**jTextFieldGio.getText());** if(!**"valid"**.equals(onsiteBLL.checkValue(onsite))) {

**JOptionPane.showMessageDialog(this,** onsiteBLL.checkValue(onsite))*;*

}

else {

if(onsiteBLL.**addCourse(onsite))** {

**JOptionPane.showMessageDialog(null, "Add new course successfully!"**)*;*

Clear()*;*

}

else {

**JOptionPane.showMessageDialog(this, "Add new course fail!"**)*;*

}

}

}

}

else {

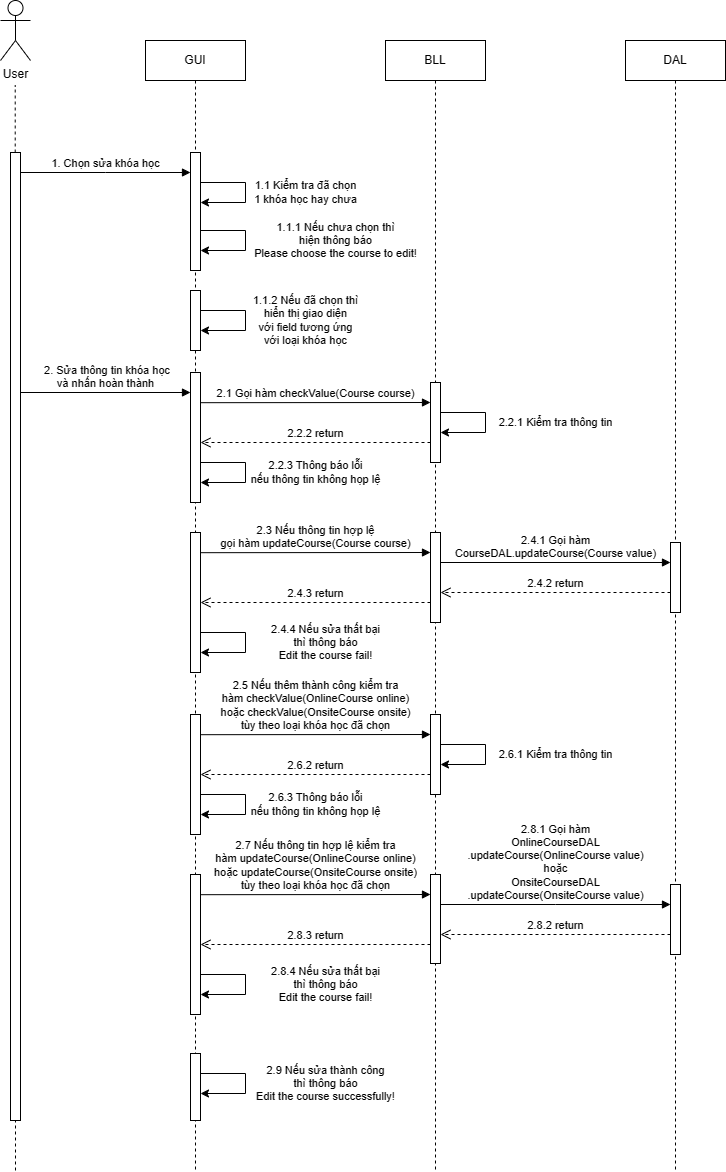
**JOptionPane.showMessageDialog(this, "Add new course fail!"**)*;*

}

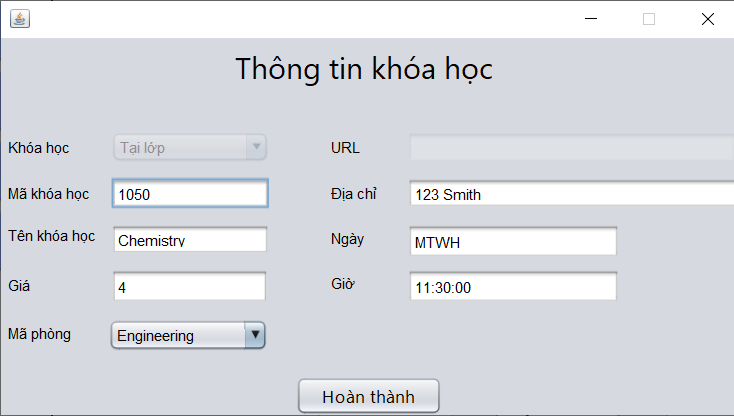
}

## **1.4 Xử lý 4: sửa/ cập nhật thông tin**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**CourseDAL**

**public** boolean updateCourse(Course **value**) {

**try** {

String query = **"update Course set Title=?, Credits=?, DepartmentID=? where CourseID=? "**;

PreparedStatement pst = c.prepareStatement(query);

pst.setNString(1, **value**.getTitle());

pst.setInt(2, **value**.getCredits());

pst.setInt(3, **value**.getDepartmentId());

pst.setInt(4, **value**.getCourseId());

**return** pst.executeUpdate() > 0;

} **catch** (SQLException ex) {

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

}

**return** **false**;

}

**OnlineCourseDAL**

**public** boolean updateOnlineCourse(OnlineCourse **value**){

**try** {

String query= **"update onlinecourse set url=? where CourseID=? "**;

PreparedStatement pst= c.prepareStatement(query);

pst.setNString(1, **value**.getUrl());

pst.setInt(2, **value**.getCourseId());

**return** pst.executeUpdate() > 0;

} **catch** (SQLException ex) {

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

}

**return** **false**;

}

**OnsiteCourseDAL**

**public** boolean updateOnsiteCourse(OnsiteCourse **value**){

**try** {

String query= **"update onsitecourse set Location=?, Days=?, Time=? where CourseID=? "**;

PreparedStatement pst= c.prepareStatement(query);

pst.setNString(1, **value**.getLocation());

pst.setNString(2, **value**.getDays());

pst.setString(3, **value**.getTime());

pst.setInt(4, **value**.getCourseId());

**return** pst.executeUpdate() > 0;

} **catch** (SQLException ex) {

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

}

**return** **false**;

}

**CourseBLL**

**public** boolean updateCourse(Course course) {

**return** courseDAL.updateCourse(course);

}

**public** String checkValue(Course course) {

**if**(**""**.equals(course.getTitle())) {

**return** **"Title not null!"**;

}

**else** **if**(course.getCredits() <= 0 || course.getCredits() > 4) {

**return** **"Credits must be numbers 1 through 4!"**;

}

**return** **"valid"**;

}

**OnlineCourseBLL**

**public** **boolean** updateCourse(OnlineCourse online) {

**return** onlineCourseDAL.updateOnlineCourse(online);

}

**public** String checkValue(OnlineCourse online) {

**if**(**""**.equals(online.getUrl())) {

**return** **"Url not null!"**;

}

**else** {

**try** {

**new** URL(online.getUrl());

} **catch** (MalformedURLException e) {

**return** **"Invalid url!"**;

}

}

**return** **"valid"**;

}

**OnsiteCourseBLL**

**public** boolean updateCourse(OnsiteCourse onsite) {

**return** onsiteCourseDAL.updateOnsiteCourse(onsite);

}

**public** String checkValue(OnsiteCourse onsite) {

**if**(**""**.equals(onsite.getLocation())) {

**return** **"location not null!"**;

}

**else** **if**(**""**.equals(onsite.getDays())) {

**return** **"days not null!"**;

}

**else** **if**(!onsite.getDays().matches(**"[MTWHFS]\*"**)) {

**return** **"days must only contain M T W H F S"**;

}

**else** **if**(**""**.equals(onsite.getTime())) {

**return** **"time not null!"**;

}

**else** {

**try** {

DateTimeFormatter formatter = DateTimeFormatter.ofPattern(**"HH:mm:00"**);

LocalTime.parse(onsite.getTime(), formatter);

} **catch** (DateTimeParseException e) {

**return** **"Invalid time!"**;

}

}

**return** **"valid"**;

}

**CourseGUI**

*// TODO add your handling code here:*

int selectedRow = jTable1.getSelectedRow();

CourseInsert\_Edit courseInsert\_Edit = **new** CourseInsert\_Edit();

courseInsert\_Edit.addWindowListener(**new** WindowAdapter() {

@Override

**public** **void** windowClosed(WindowEvent e) {

loadToTable(courseBLL.getList());

}

});

courseInsert\_Edit.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

courseInsert\_Edit.getjButton1().setVisible(**true**);

courseInsert\_Edit.setInsert(**false**);

courseInsert\_Edit.setFlag(**true**);

**if**(selectedRow != -1) {

*// Lấy dữ liệu từ hàng đã chọn*

boolean flag = **false**;

**for** (Course x : courseBLL.getList()) {

**if** (x.getCourseId() == (int) **this**.jTable1.getValueAt(selectedRow, 0)) {

**for** (OnlineCourse y : onlineCourseBLL.getList()) {

**if** (x.getCourseId() == y.getCourseId()) {

courseInsert\_Edit.getjComboBox1().setSelectedIndex(0);

courseInsert\_Edit.getjComboBox1().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldId().setText(Integer.toString(x.getCourseId()));

courseInsert\_Edit.getjTextFieldId().setEditable(**false**);

courseInsert\_Edit.getjTextFieldTen().setText(x.getTitle());

courseInsert\_Edit.getjTextFieldTen().setEditable(**true**);

courseInsert\_Edit.getjTextFieldGia().setText(Integer.toString(x.getCredits()));

courseInsert\_Edit.getjTextFieldGia().setEditable(**true**);

courseInsert\_Edit.getjTextFieldUrl().setText(y.getUrl());

courseInsert\_Edit.getjTextFieldUrl().setEditable(**true**);

courseInsert\_Edit.getjTextFieldDiaChi().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldGio().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldNgay().setEnabled(**false**);

ArrayList<**String**> data = **new** ArrayList<>();

**try** {

**for**(Department z: departmentBLL.getListDepartment()) {

data.add(z.getName());

}

courseInsert\_Edit.getjComboBox2().setModel(**new** javax.swing.DefaultComboBoxModel<>(data.toArray(**String**[]::**new**)));

**for**(int i=0;i< courseInsert\_Edit.getjComboBox2().getItemCount();i++){

**String** item= (**String**) courseInsert\_Edit.getjComboBox2().getItemAt(i);

**if**(item.equals(departmentBLL.getDeparment(x.getDepartmentId()).getName())){

courseInsert\_Edit.getjComboBox2().setSelectedIndex(i);

courseInsert\_Edit.getjComboBox2().setEnabled(**true**);

**break**;

}

}

} **catch** (SQLException ex) {

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

}

courseInsert\_Edit.setVisible(**true**);

JOptionPane.showMessageDialog(**null**, **"Please do not leave the data blank!"**);

flag = **true**;

**break**;

}

}

**if** (flag) {

**continue**;

}

**for** (OnsiteCourse y : onsiteCourseBLL.getList()) {

**if** (x.getCourseId() == y.getCourseId()) {

courseInsert\_Edit.getjComboBox1().setSelectedIndex(1);

courseInsert\_Edit.getjComboBox1().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldId().setText(Integer.toString(x.getCourseId()));

courseInsert\_Edit.getjTextFieldId().setEditable(**false**);

courseInsert\_Edit.getjTextFieldTen().setText(x.getTitle());

courseInsert\_Edit.getjTextFieldTen().setEditable(**true**);

courseInsert\_Edit.getjTextFieldGia().setText(Integer.toString(x.getCredits()));

courseInsert\_Edit.getjTextFieldGia().setEditable(**true**);

courseInsert\_Edit.getjTextFieldUrl().setEnabled(**false**);

courseInsert\_Edit.getjTextFieldDiaChi().setText(y.getLocation());

courseInsert\_Edit.getjTextFieldDiaChi().setEditable(**true**);

courseInsert\_Edit.getjTextFieldGio().setText(y.getTime().toString());

courseInsert\_Edit.getjTextFieldGio().setEditable(**true**);

courseInsert\_Edit.getjTextFieldNgay().setText(y.getDays());

courseInsert\_Edit.getjTextFieldNgay().setEditable(**true**);

ArrayList<**String**> data = **new** ArrayList<>();

**try** {

**for**(Department z: departmentBLL.getListDepartment()) {

data.add(z.getName());

}

courseInsert\_Edit.getjComboBox2().setModel(**new** javax.swing.DefaultComboBoxModel<>(data.toArray(**String**[]::**new**)));

**for**(int i=0;i< courseInsert\_Edit.getjComboBox2().getItemCount();i++){

**String** item= (**String**) courseInsert\_Edit.getjComboBox2().getItemAt(i);

**if**(item.equals(departmentBLL.getDeparment(x.getDepartmentId()).getName())){

courseInsert\_Edit.getjComboBox2().setSelectedIndex(i);

courseInsert\_Edit.getjComboBox2().setEnabled(**true**);

**break**;

}

}

} **catch** (SQLException ex) {

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

}

courseInsert\_Edit.setVisible(**true**);

JOptionPane.showMessageDialog(**null**, **"Please do not leave the data blank!"**);

**break**;

}

}

}

}

*// Mở một hộp thoại chỉnh sửa hoặc một form chỉnh sửa với dữ liệu đã chọn*

*// Ví dụ: new EditDialog(data).setVisible(true);*

} **else** {

JOptionPane.showMessageDialog(**null**, **"Please choose the course to edit!"**);

}

**CourseInsert\_EditGUI**

Course course = **new** Course();

course.setCourseId(Integer.parseInt(jTextFieldId.**getText**()));

course.setTitle(jTextFieldTen.**getText**());

**if**(!jTextFieldGia.**getText**().isEmpty()) {

course.setCredits(Integer.parseInt(jTextFieldGia.**getText**()));

}

**for**(Department x: departmentBLL.getListDepartment()) {

**if**(x.getName().equals(jComboBox2.getSelectedItem().toString())) {

course.setDepartmentId(x.getDepartmentId());

}

}

**if**(!**"valid"**.equals(courseBLL.checkValue(course))) {

JOptionPane.showMessageDialog(**this**, courseBLL.checkValue(course));

}

**else** {

**if**(courseBLL.updateCourse(course)) {

**if**(jComboBox1.getSelectedIndex() == 0) {

OnlineCourse online = **new** OnlineCourse();

online.setCourseId(Integer.parseInt(jTextFieldId.**getText**()));

online.setUrl(jTextFieldUrl.**getText**());

**if**(!**"valid"**.equals(onlineBLL.checkValue(online))) {

JOptionPane.showMessageDialog(**this**, onlineBLL.checkValue(online));

}

**else** {

**if**(onlineBLL.updateCourse(online)) {

JOptionPane.showMessageDialog(**null**, **"Edit the course successfully!"**);

Clear();

dispose();

}

**else** {

JOptionPane.showMessageDialog(**null**, **"Edit the course fail!"**);

}

}

}

**else** {

OnsiteCourse onsite = **new** OnsiteCourse();

onsite.setCourseId(Integer.parseInt(jTextFieldId.**getText**()));

onsite.setLocation(jTextFieldDiaChi.**getText**());

onsite.setDays(jTextFieldNgay.**getText**());

onsite.setTime(jTextFieldGio.**getText**());

**if**(!**"valid"**.equals(onsiteBLL.checkValue(onsite))) {

JOptionPane.showMessageDialog(**this**, onsiteBLL.checkValue(onsite));

}

**else** {

**if**(onsiteBLL.updateCourse(onsite)) {

JOptionPane.showMessageDialog(**null**, **"Edit the course successfully!"**);

Clear();

dispose();

}

**else** {

JOptionPane.showMessageDialog(**null**, **"Edit the course fail!"**);

}

}

}

}

**else** {

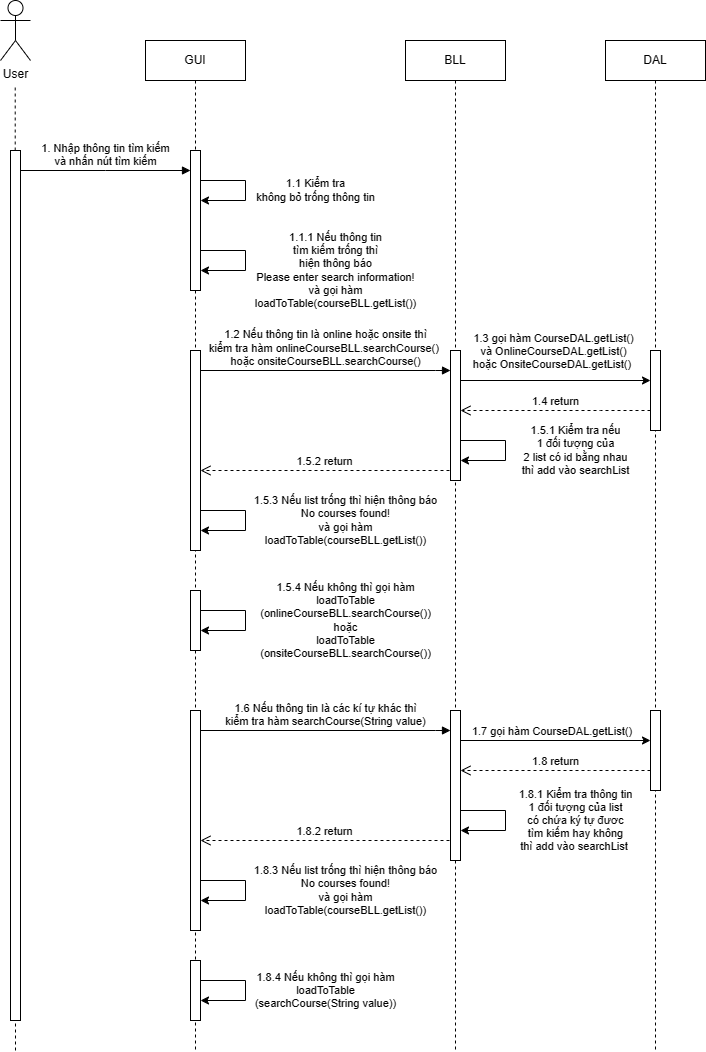
JOptionPane.showMessageDialog(**null**, **"Edit the course fail!"**);

}

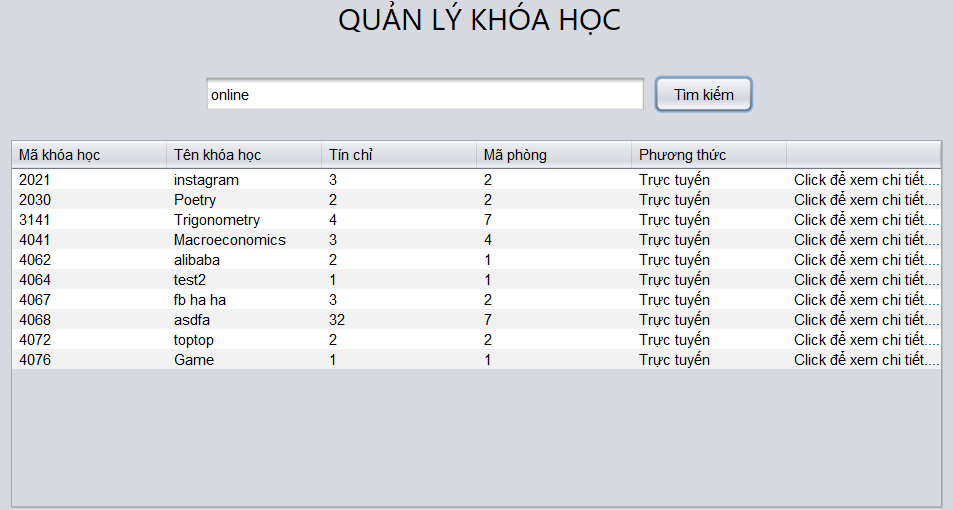
}

## **1.5 Xử lý 5: tìm kiếm**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**CourseDAL**

**public** ArrayList<Course> getList() {

ArrayList list = **new** ArrayList<Course>();

String query = **"select \*from Course"**;

**try** {

PreparedStatement pst = c.prepareStatement(query);

ResultSet rs = pst.executeQuery();

**while** (rs.next()) {

Course **value** = **new** Course();

**value**.setCourseId(rs.getInt(1));

**value**.setTitle(rs.getString(2));

**value**.setCredits(rs.getInt(3));

**value**.setDepartmentId(rs.getInt(4));

list.add(**value**);

}

} **catch** (SQLException ex) {

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

}

**return** list;

}

**OnlineCourseDAL**

**public** ArrayList<OnlineCourse> getList() {

ArrayList list = **new** ArrayList<OnlineCourse>();

String query = **"select \*from onlinecourse"**;

**try** {

PreparedStatement pst = c.prepareStatement(query);

ResultSet rs = pst.executeQuery();

**while** (rs.next()) {

OnlineCourse **value** = **new** OnlineCourse();

**value**.setCourseId(rs.getInt(1));

**value**.setUrl(rs.getString(2));

list.add(**value**);

}

} **catch** (SQLException ex) {

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

}

**return** list;

}

**OnsiteCourseDAL**

**public** ArrayList<OnsiteCourse> getList() {

ArrayList list = **new** ArrayList<OnsiteCourse>();

String query = **"select \*from onsitecourse"**;

**try** {

PreparedStatement pst = c.prepareStatement(query);

ResultSet rs = pst.executeQuery();

**while** (rs.next()) {

OnsiteCourse **value** = **new** OnsiteCourse();

**value**.setCourseId(rs.getInt(1));

**value**.setLocation(rs.getString(2));

**value**.setDays(rs.getString(3));

**value**.setTime(rs.getString(4));

list.add(**value**);

}

} **catch** (SQLException ex) {

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

}

**return** list;

}

**CourseBLL**

**public** ArrayList<Course> searchCourse(String value) {

ArrayList<Course> list = **new** ArrayList<>();

**for** (Course x : **this**.getList()) {

String id = Integer.toString(x.getCourseId()).toLowerCase();

String credits = Integer.toString(x.getCredits()).toLowerCase();

String department = Integer.toString(x.getDepartmentId()).toLowerCase();

**if** (id.contains(value.toLowerCase()) || x.getTitle().toLowerCase().contains(value.toLowerCase()) || credits.contains(value.toLowerCase()) || department.contains(value.toLowerCase())) {

list.add(x);

}

}

**return** list;

}

**OnlineCourseBLL**

**public** ArrayList<Course> searchCourse(){

ArrayList<Course> searchList= **new** ArrayList<>();

**for** (Course x : courseDAL.getList()) {

**for**(OnlineCourse y : getList()) {

**if**(x.getCourseId() == y.getCourseId()){

searchList.add(x);

}

}

}

**return** searchList;

}

**OnsiteCourseBLL**

**public** ArrayList<Course> searchCourse(){

ArrayList<Course> searchList= **new** ArrayList<>();

**for** (Course x : courseDAL.getList()) {

**for**(OnsiteCourse y : getList()) {

**if**(x.getCourseId() == y.getCourseId()){

searchList.add(x);

}

}

}

**return** searchList;

}

**CourseGUI**

*// TODO add your handling code here:*

**if**(**""**.equals(jTextField1.getText())) {

JOptionPane.showMessageDialog(**null**, **"Please enter search information!"**);

loadToTable(courseBLL.getList());

}

**else** {

**if**(**"online"**.equalsIgnoreCase(jTextField1.getText())) {

**if**(onlineCourseBLL.searchCourse().isEmpty()) {

JOptionPane.showMessageDialog(**null**, **"No courses found!"**);

}

**else** {

loadToTable(onlineCourseBLL.searchCourse());

}

}

**else** **if**(**"onsite"**.equalsIgnoreCase(jTextField1.getText())) {

**if**(onsiteCourseBLL.searchCourse().isEmpty()) {

JOptionPane.showMessageDialog(**null**, **"No courses found!"**);

}

**else** {

loadToTable(onsiteCourseBLL.searchCourse());

}

}

**else** **if**(courseBLL.searchCourse(jTextField1.getText()).isEmpty()) {

JOptionPane.showMessageDialog(**null**, **"No courses found!"**);

loadToTable(courseBLL.getList());

}

**else** {

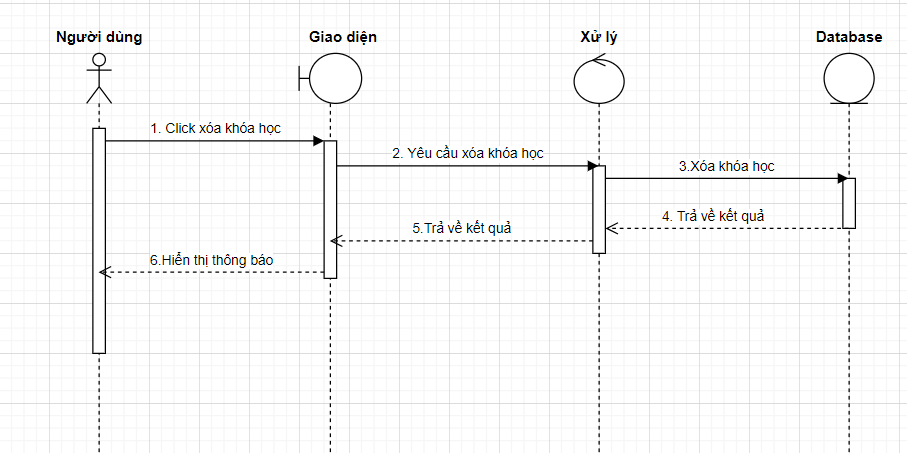
loadToTable(courseBLL.searchCourse(jTextField1.getText()));

}

}

## **1.6 Xử lý 6: xóa record**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**CourseDAL**

 public int deleteCourse(int value) {

        try {

            String query = "update Course set Title=? where CourseID=?";

            PreparedStatement pst = c.prepareStatement(query);

            pst.setNString(1, "null");

            pst.setInt(2, value);

            return pst.executeUpdate();

        } catch (SQLException ex) {

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

        }

        return 0;

    }

**OnlineCourseDAL**

public int deleteOnlineCourse(int value){

        try {

            String query="delete from onlinecourse where CourseID=?";

            PreparedStatement pst= c.prepareStatement(query);

            pst.setInt(1, value);

            return pst.executeUpdate();

        } catch (SQLException ex) {

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

        }

        return 0;

    }

**OnsiteCourseDAL**

public int deleteOnsiteCourse(int value){

        try {

            String query="delete from onsitecourse where CourseID=?";

            PreparedStatement pst= c.prepareStatement(query);

            pst.setInt(1, value);

            return pst.executeUpdate();

        } catch (SQLException ex) {

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

        }

        return 0;

    }

**CourseBLL**

 public int deleteCourse(int value) {

        return courseDAL.deleteCourse(value);

    }

**OnlineCourseBLL**

public int deleteCourse(int value) {

        boolean flag=false;

        for (OnlineCourse x : this.getList()) {

            if (x.getCourseId() == value) {

                flag=true;

            }

        }

        if(flag==false){

           JOptionPane.showMessageDialog(null, "CourseId does not exist yet");

           return 0;

        }

        return onlineCourseDAL.deleteOnlineCourse(value);

    }

**OnsiteCourseBLL**

 public int deleteCourse(int value) {

        boolean flag=false;

        for (OnsiteCourse x : this.getList()) {

            if (x.getCourseId() == value) {

                flag=true;

            }

        }

        if(flag==false){

           JOptionPane.showMessageDialog(null, "CourseId does not exist yet");

           return 0;

        }

        return onsiteCourseDAL.deleteOnsiteCourse(value);

    }

**CourseGUI**

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

        int row = (int) this.jTable1.getSelectedRow();

        if (row < 0) {

            JOptionPane.showMessageDialog(null, "Vui lòng chọn khóa học muốn xóa");

        } else {

            int result = JOptionPane.showConfirmDialog(null, "Bạn có muốn xóa không?", "Xác nhận", JOptionPane.OK\_CANCEL\_OPTION);

            if (result == JOptionPane.OK\_OPTION) {

                if (courseBLL.deleteCourse((int) this.jTable1.getValueAt(row, 0)) == 1) {

                    JOptionPane.showMessageDialog(null, "Xóa thành công!");

                    loadToTable(courseBLL.getList());

                } else {

                    JOptionPane.showMessageDialog(null, "Xóa thất bại!");

                }

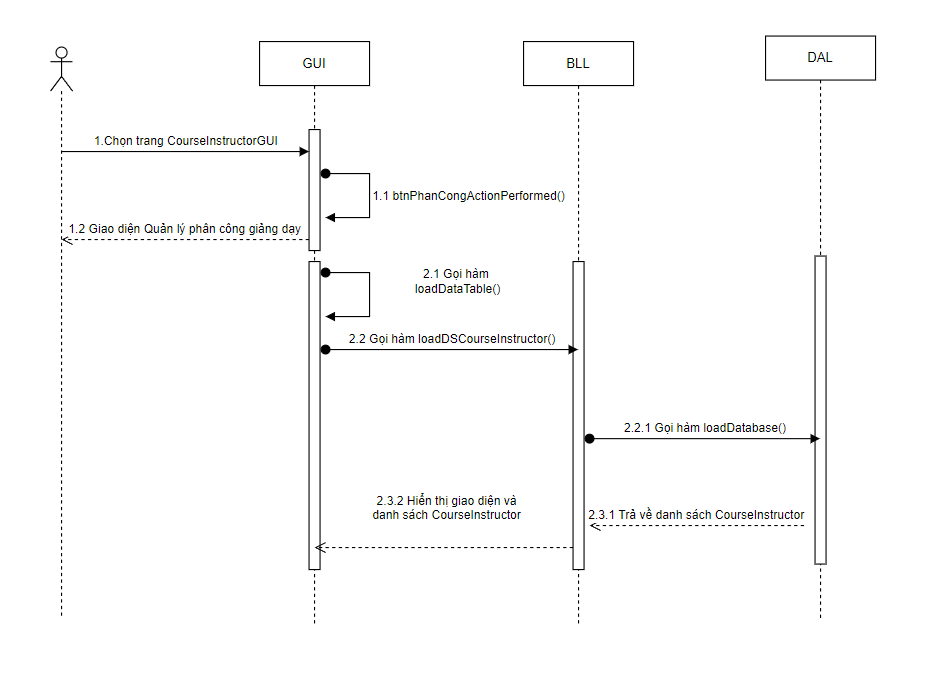
            }

        }

# **CHỨC NĂNG 2: QUẢN LÝ THÔNG TIN PHÂN CÔNG GIẢNG DẠY (COURSEINTRUCTOR)**

## **2.1 Xử lý 1: hiển thị danh sách**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**

A screenshot of a computer

Description automatically generated

* **Code 3 class: DAL, BLL, GUI**

**-CourseInstructorDAL**

**public** ArrayList<CourseInstructor> loadDatabase() throws Exception {

ArrayList<CourseInstructor> **list** = new ArrayList<>();

Statement **st** = c.createStatement();

String **query** ="SELECT CourseInstructor.CourseID, CourseInstructor.PersonID,"

+ " CONCAT(Person.LastName, ' ', Person.FirstName) AS Name,Course.Title "

+ " FROM Person "

+ "JOIN CourseInstructor ON Person.PersonID = CourseInstructor.PersonID"

+ " JOIN Course ON CourseInstructor.CourseID = Course.CourseID ";

try {

ResultSet rs = **st**.executeQuery(**query**);

**while** (rs.next()) {

CourseInstructor csin = new CourseInstructor(

rs.getInt("CourseID"),

rs.getString("Title"),

rs.getInt("PersonID"),

rs.getString("Name")

);

**list**.add(csin);

}

rs.**close**();

} catch (SQLException **ex**) {

System.**out**.println("Khong the load database CourseInstructor: " + **ex**);

}

**return** **list**;

}

**-CourseInstructorBLL**

**public** ArrayList<CourseInstructor> **loadDSCourseInstructor**() **throws** Exception{

**return** dal.loadDatabase();

}

**-CourseIntructorGUI**

**public** CourseIntructorGUI() throws Exception {

initComponents();

btnXong.setVisible(!true);

btnHuy.setVisible(!true);

loadDataTable(cIBLL.loadDSCourseInstructor());

comboBoxModel = loadDataCourseToCombobox( courseBLL.loadData());

nameCourse.setModel(comboBoxModel);

comboBoxModelPS = loadDataPersonToCombobox(personBLL.loadData());

namePerson.setModel(comboBoxModelPS);

nameCourse.setSelectedIndex(0);

namePerson.setSelectedIndex(0);

}

**public** void loadDataTable( ArrayList<CourseInstructor> data) throws Exception{

model.setRowCount(0);

model = addArrayListToTable(data);

tableCourseIns.setModel(model);

}

## **2.2 Xử lý 2: hiển thị chi tiết 1 record (dòng)**

* **Sơ đồ tuần tự**

A diagram of a diagram

Description automatically generated

* **Màn hình giao diện**

**A screenshot of a computer

Description automatically generated**

* **Code 3 class: DAL, BLL, GUI**

**- CourseInstructorDAL**

**public** ArrayList<CourseInstructor> loadDatabase() throws Exception {

ArrayList<CourseInstructor> **list** = new ArrayList<>();

Statement **st** = c.createStatement();

String **query** ="SELECT CourseInstructor.CourseID, CourseInstructor.PersonID,"

+ " CONCAT(Person.LastName, ' ', Person.FirstName) AS Name,Course.Title "

+ " FROM Person "

+ "JOIN CourseInstructor ON Person.PersonID = CourseInstructor.PersonID"

+ " JOIN Course ON CourseInstructor.CourseID = Course.CourseID ";

try {

ResultSet rs = **st**.executeQuery(**query**);

**while** (rs.next()) {

CourseInstructor csin = new CourseInstructor(

rs.getInt("CourseID"),

rs.getString("Title"),

rs.getInt("PersonID"),

rs.getString("Name")

);

**list**.add(csin);

}

rs.**close**();

} catch (SQLException **ex**) {

System.**out**.println("Khong the load database CourseInstructor: " + **ex**);

}

**return** **list**;

}

**-File CourseBLL**

**public** Course findElement(**int** id) throws **Exception**{

ArrayList<Course> **list** = courseDAL.loadDatabase();

**for**(Course co : **list**){

**if**(co.getCourseId() == id)

**return** co;

}

**return** **null**;

}

**-LecturesBLL**

**public** Lecturers findElement(**int** id) throws **Exception**{

ArrayList<Lecturers> **list** = dal.loadDatabase();

**for**(Lecturers ps : **list**){

**if**(ps.getPersonId() == id)

**return** ps;

}

**return** **null**;

}

**- CourseInstructorGUI**

**private** **void** **tableCourseInsMouseClicked**(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

**int** selectedRow = tableCourseIns.getSelectedRow();

nameCourse.setEnabled(false);

namePerson.setEnabled(false);

**int** courseID = Integer.parseInt(model.getValueAt(selectedRow, 0).toString());

**int** personID = Integer.parseInt(model.getValueAt(selectedRow, 2).toString());

**try** {

Course cour= (Course) courseBLL.findElement(courseID);

Lecturers lect = (Lecturers) personBLL.findElement(personID);

comboBoxModel.setSelectedItem(cour);

comboBoxModelPS.setSelectedItem(lect);

} **catch** (Exception ex) {

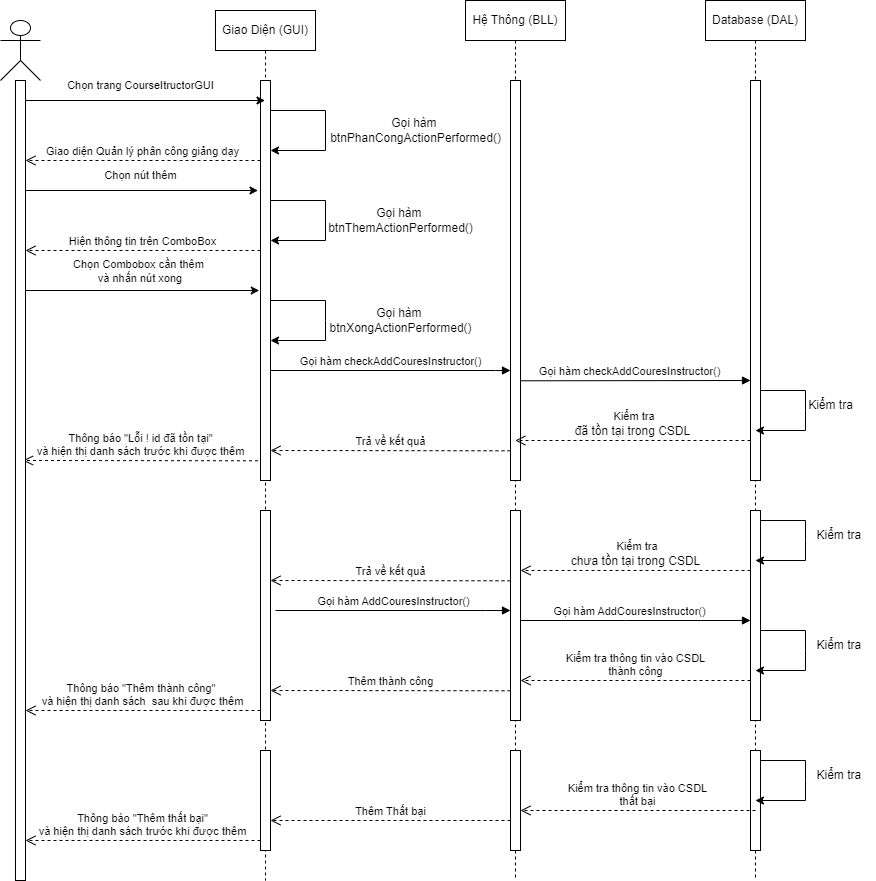
Logger.getLogger(CourseIntructorGUI.**class**.getName()).log(Level.SEVERE, null, ex);

}

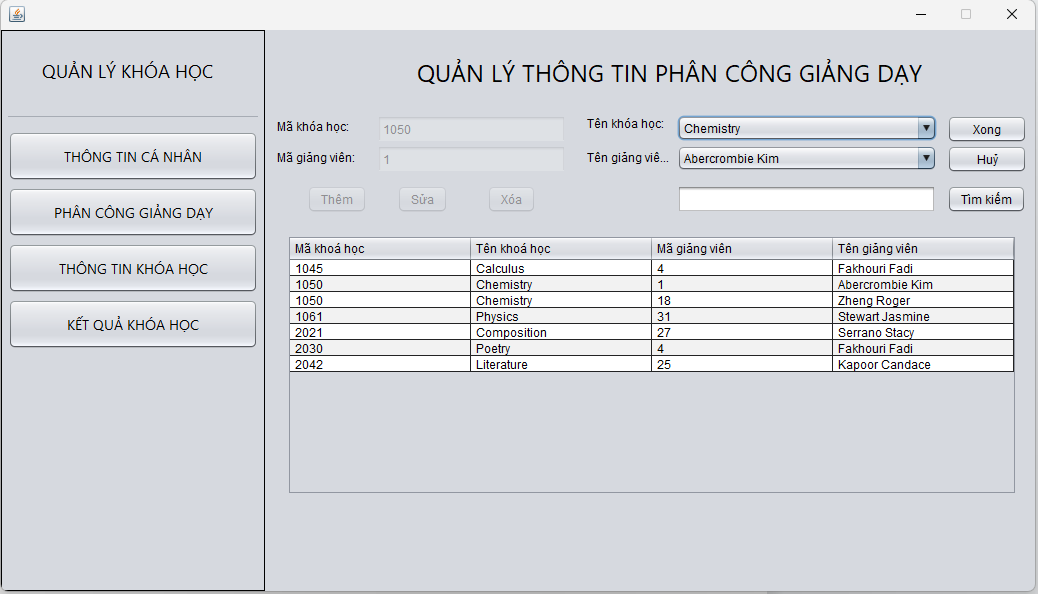
}

## **2.3 Xử lý 3: thêm mới thông tin**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**DAL: CourseIntructorDAL.java**

public boolean addCourseInstructor(int CourseID, int PersonID) throws SQLException {

        String query = "INSERT INTO CourseInstructor (CourseID, PersonID) VALUES (?, ?)";

        try (PreparedStatement p = c.prepareStatement(query)) {

            p.setInt(1, CourseID);

            p.setInt(2, PersonID);

            int rowsAffected = p.executeUpdate();

            return rowsAffected > 0;

        }

}

public int checkAddCourseInstructor(int CourseID, int PersonID) throws SQLException {

        String query = "SELECT COUNT(\*) FROM CourseInstructor WHERE CourseID = ? AND PersonID = ?";

        try (PreparedStatement p = c.prepareStatement(query)) {

            p.setInt(1, CourseID);

            p.setInt(2, PersonID);

            ResultSet rs = p.executeQuery();

            rs.next();

            int count =rs.getInt(1);

            return count ;

        }

  }

**BLL: CourseIntructorBLL.java**

public static boolean addCourseInstructor( int idcorse, int idperson) throws SQLException {

        return dal.addCourseInstructor(  idcorse, idperson);

}

public static int checkAddCourseInstructor( int idcourse, int idperson) throws SQLException {

        return dal.checkAddCourseInstructor( idcourse, idperson);

}

**GUI: CourseIntructorGUI.java**

btnThem.setText("Thêm");

btnThem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnThemActionPerformed(evt);

}

});

pprivate void btnThemActionPerformed(java.awt.event.ActionEvent evt) {

        int luaChon = JOptionPane.showConfirmDialog(null, "Có thêm không?", "Hỏi", JOptionPane.YES\_NO\_OPTION);

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

            insert = true;

            nameCourse.setEnabled(true);

            namePerson.setEnabled(true);

            btnThem.setEnabled(false);

            btnSua.setEnabled(false);

            btnXoa.setEnabled(false);

            btnXong.setVisible(true);

            btnHuy.setVisible(true);

        }

}

private void btnXongActionPerformed(java.awt.event.ActionEvent evt)

{//GEN-FIRST:event\_btnXongActionPerformed

        // TODO add your handling code here:

        if(!insert) {

            int maKhoahoc = Integer.parseInt(maKH.getText());

            int maGiangvien = Integer.parseInt(maGV.getText());

            try {

                CourseInstructor a =  cIBLL.returnOld();

                CourseInstructor b =  cIBLL.getCourseInstructorNewByID(courseBLL.findElement(maKhoahoc),personBLL.findElement(maGiangvien));

                if(!cIBLL.updateCourseInstructor(a, b)){

                    JOptionPane.showMessageDialog( null, "Update fail!!!?" );

                }else{

                  JOptionPane.showMessageDialog( null, "Update success!!!?" );

                }

                txtSearch.setText("");

                model.setRowCount(0);

                model = addArrayListToTable(cIBLL.loadDSCourseInstructor());

                tableCourseIns.setModel(model);

                btnHuyActionPerformed(evt);

            } catch (Exception ex) {

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

            }

        }

        else {

            int maKhoahoc = Integer.parseInt(maKH.getText());

            int maGiangvien = Integer.parseInt(maGV.getText());

            try {

                if (CourseInstructorBLL.checkAddCourseInstructor(maKhoahoc, maGiangvien) == 0) {

                    if(CourseInstructorBLL.addCourseInstructor(maKhoahoc, maGiangvien)) {

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

                    }

                    else {

                        JOptionPane.showMessageDialog(null, "Thêm thất bại!");

                    }

                } else {

                    JOptionPane.showMessageDialog(null, "Lỗi! id đã tồn tại");

                }

                // Cập nhật bảng hiển thị

                txtSearch.setText("");

                model.setRowCount(0);

                model = addArrayListToTable(cIBLL.loadDSCourseInstructor());

                tableCourseIns.setModel(model);

                nameCourse.setEnabled(false);

                namePerson.setEnabled(false);

                btnThem.setEnabled(!false);

                btnSua.setEnabled(!false);

                btnXoa.setEnabled(!false);

                btnXong.setVisible(!true);

                btnHuy.setVisible(!true);

            } catch (Exception e) {

                // Xử lý ngoại lệ nếu có

                e.printStackTrace();

                JOptionPane.showMessageDialog(null, "Đã xảy ra lỗi: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

            }

        }

}

## **2.4 Xử lý 4: sửa/ cập nhật thông tin**

* **Sơ đồ tuần tự**

A diagram of a company

Description automatically generated

* **Màn hình giao diện**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

* **Code 3 class: DAL, BLL, GUI**

**-CourseInstructorDAL**

**public** Boolean updateCourseInstructor(CourseInstructor csinOld, CourseInstructor csinNew) **throws** Exception {

String sql = "UPDATE CourseInstructor SET CourseID = ?, PersonID = ? WHERE CourseID = ? AND PersonID = ?";

**try** {

PreparedStatement preparedStatement = c.prepareStatement(sql);

preparedStatement.setString(1, Integer.toString(csinNew.getCourseID()));

preparedStatement.setString(2, Integer.toString(csinNew.getPersonID()));

preparedStatement.setString(3, Integer.toString(csinOld.getCourseID()));

preparedStatement.setString(4, Integer.toString(csinOld.getPersonID()));

preparedStatement.executeUpdate();

**return** true;

} **catch** (SQLException ex) {

System.out.println("Khong the Cap nhat CourseInstructor vao database !!!");

**return** false;

}

}

**-CourseInstructorBLL**

**public** boolean updateCourseInstructor(CourseInstructor csinOld, CourseInstructor csinNew) throws Exception{

**if**(csinOld.getCourseID()==csinNew.getCourseID() && csinOld.getPersonID()== csinNew.getPersonID())

**return** false;

**try** {

**return** dal.updateCourseInstructor(csinOld,csinNew);

} **catch** (Exception e) {

System.**out**.println("Khong the Cap nhat CourseInstructor vao database !!!");

}

**return** false;

}

**-CourseInstructorGUI**

**private** **void** btnXongActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

**if**(!insert) {

**int** maKhoahoc = Integer.parseInt(maKH.**getText**());

**int** maGiangvien = Integer.parseInt(maGV.**getText**());

**try** {

CourseInstructor a = cIBLL.returnOld();

CourseInstructor b = cIBLL.getCourseInstructorNewByID(courseBLL.findElement(maKhoahoc),personBLL.findElement(maGiangvien));

**if**(!cIBLL.updateCourseInstructor(a, b)){

JOptionPane.showMessageDialog( **null**, "Update fail!!!?" );

}**else**{

JOptionPane.showMessageDialog( **null**, "Update success!!!?" );

}

txtSearch.setText("");

model.setRowCount(0);

model = addArrayListToTable(cIBLL.loadDSCourseInstructor());

tableCourseIns.setModel(model);

btnHuyActionPerformed(evt);

} **catch** (Exception ex) {

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

}

}

**else** {

**int** maKhoahoc = Integer.parseInt(maKH.**getText**());

**int** maGiangvien = Integer.parseInt(maGV.**getText**());

**try** {

**if** (CourseInstructorBLL.checkAddCourseInstructor(maKhoahoc, maGiangvien) == 0) {

**if**(CourseInstructorBLL.addCourseInstructor(maKhoahoc, maGiangvien)) {

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

}

**else** {

JOptionPane.showMessageDialog(**null**, "Thêm thất bại!");

}

} **else** {

JOptionPane.showMessageDialog(**null**, "Lỗi! id đã tồn tại");

}

// Cập nhật bảng hiển thị

txtSearch.setText("");

model.setRowCount(0);

model = addArrayListToTable(cIBLL.loadDSCourseInstructor());

tableCourseIns.setModel(model);

nameCourse.setEnabled(**false**);

namePerson.setEnabled(**false**);

btnThem.setEnabled(!**false**);

btnSua.setEnabled(!**false**);

btnXoa.setEnabled(!**false**);

btnXong.setVisible(!**true**);

btnHuy.setVisible(!**true**);

} **catch** (Exception e) {

// Xử lý ngoại lệ nếu có

e.printStackTrace();

JOptionPane.showMessageDialog(**null**, "Đã xảy ra lỗi: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

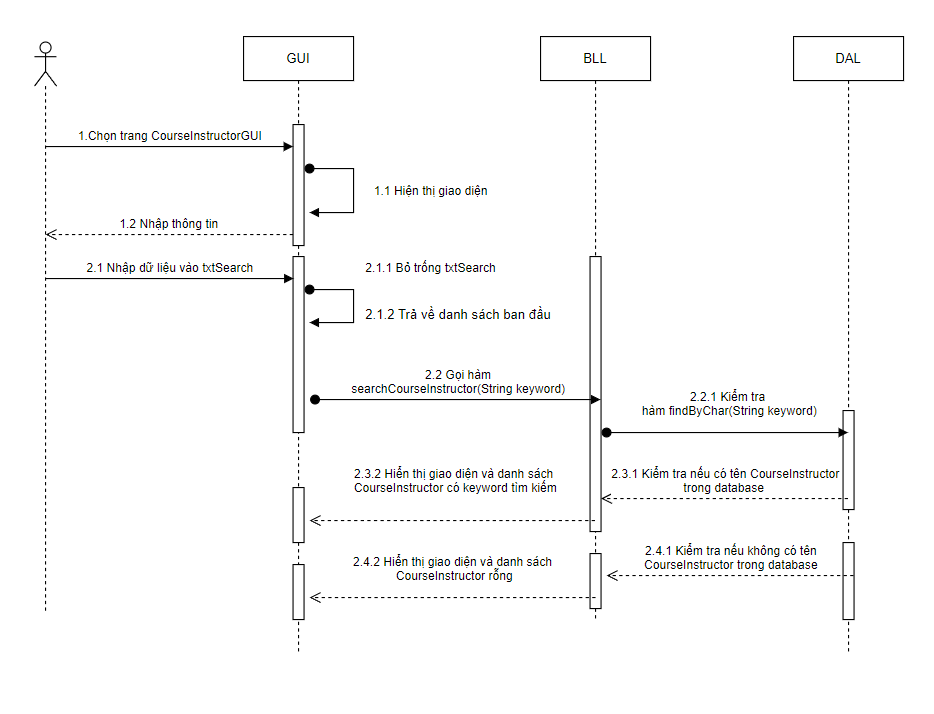
}

}

}

## **2.5 Xử lý 5: tìm kiếm**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**

**A screenshot of a computer

Description automatically generated**

* **Code 3 class: DAL, BLL, GUI**

**- CourseIntructorDAL**

**public** ArrayList<CourseInstructor> findByChar(String keyword){

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

String sql = "SELECT DISTINCT "

+ " CourseInstructor.CourseID, CourseInstructor.PersonID, "

+ " CONCAT(Person.LastName, ' ', Person.FirstName) AS Name,Course.Title "

+ " FROM Person "

+ " JOIN CourseInstructor ON Person.PersonID = CourseInstructor.PersonID "

+ " JOIN Course ON CourseInstructor.CourseID = Course.CourseID "

+ " WHERE CONCAT(Person.LastName, ' ', Person.FirstName) LIKE ? "

+ " OR Course.Title LIKE ?; ";

try{

PreparedStatement preparedStatement = c.prepareStatement(sql);

preparedStatement.setString(1, "%" + keyword + "%");

preparedStatement.setString(2, "%" + keyword + "%");

**ResultSet** rs = preparedStatement.executeQuery();

**while**(rs.next()){

CourseInstructor csin = new CourseInstructor(

rs.getInt("CourseID"),

rs.getString("Title"),

rs.getInt("PersonID"),

rs.getString("Name")

);

list.add(csin);

}

// c.close();

}

catch (SQLException ex) {

System.out.println("Khong the tim CourseInstructor !!!");

}

**return** list;

}

**-CourseInstructorBLL**

**public** ArrayList<CourseInstructor> searchCourseInstructor(String keyword)

{

**return** dal.findByChar(keyword);

}

**-** **CourseInstructorGUI**

**private** **void** txtSearchKeyReleased(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

**if**(txtSearch.**getText**().isEmpty()){

model.setRowCount(0);

**try** {

model = addArrayListToTable(cIBLL.loadDSCourseInstructor());

} **catch** (Exception ex) {

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

}

}**else** **if**(cIBLL.searchCourseInstructor(txtSearch.**getText**().trim()).**size**()>0){

model.setRowCount(0);

model = addArrayListToTable(cIBLL.searchCourseInstructor(txtSearch.**getText**().trim()));

} **else**{

model.setRowCount(0);

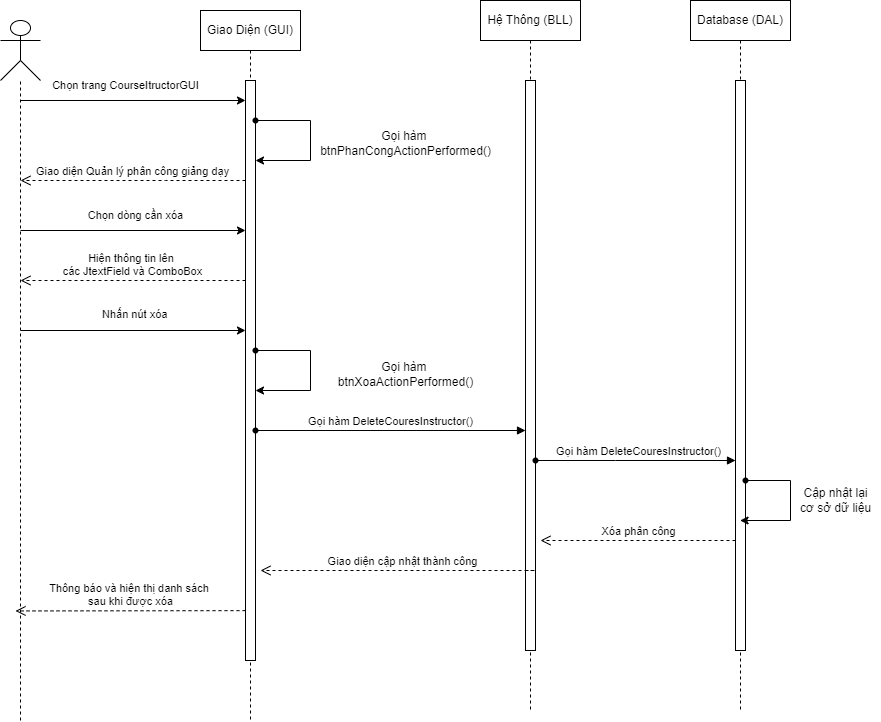
}

tableCourseIns.setModel(model);

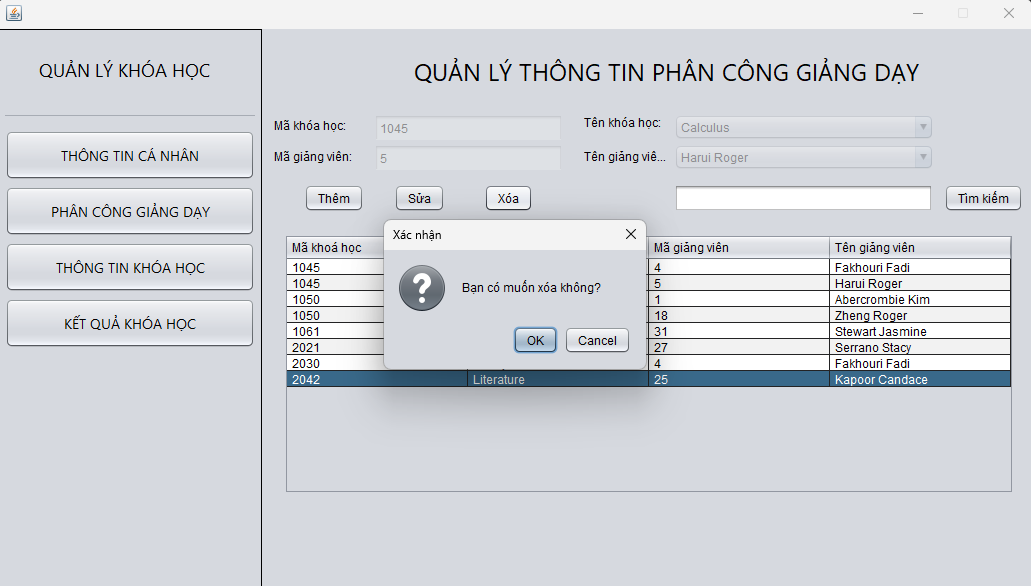
}

## **2.6 Xử lý 6: xóa record**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**DAL: CourseIntructorDAL.java**

public int deleteCourseInstructor(int CourseID, int PersonID) throws SQLException {

        String query = "DELETE FROM CourseInstructor WHERE CourseID = ? And PersonID = ?";

        PreparedStatement p = c.prepareStatement(query);

        p.setInt(1, CourseID);

        p.setInt(2, PersonID);

        int result = p.executeUpdate();

        return result;

 }

**BLL: CourseIntructorBLL.java**

public static int deleteCourseInstructor( int idcorse, int idperson) throws SQLException {

        return dal.deleteCourseInstructor(  idcorse, idperson);

}

**GUI: CourseIntructorGUI.java**

btnXoa.setText("Xóa");

        btnXoa.addActionListener(new java.awt.event.ActionListener() {

            public void actionPerformed(java.awt.event.ActionEvent evt) {

                btnXoaActionPerformed(evt);

            }

        });

        getContentPane().add(btnXoa, new org.netbeans.lib.awtextra.AbsoluteConstraints(220, 180, -1, -1));

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

        int row = (int) ((JTable) this.tableCourseIns).getSelectedRow();

        if (row < 0) {

            JOptionPane.showMessageDialog(null, "Vui lòng chọn khóa học muốn xóa");

        } else {

            int result = JOptionPane.showConfirmDialog(null, "Bạn có muốn xóa không?", "Xác nhận", JOptionPane.OK\_CANCEL\_OPTION);

            if (result == JOptionPane.OK\_OPTION) {

                try {

                    if (CourseInstructorBLL.deleteCourseInstructor(Integer.parseInt(maKH.getText()),Integer.parseInt(maGV.getText())) != 0 )  {

                        JOptionPane.showMessageDialog(null, "Xóa thành công!");

                        model = addArrayListToTable(cIBLL.loadDSCourseInstructor());

                        tableCourseIns.setModel(model);

                    } else {

                        JOptionPane.showMessageDialog(null, "Xóa thất bại!");

                    }

                } catch (Exception e) {

                    // TODO Auto-generated catch block

                    e.printStackTrace();

                }

            }

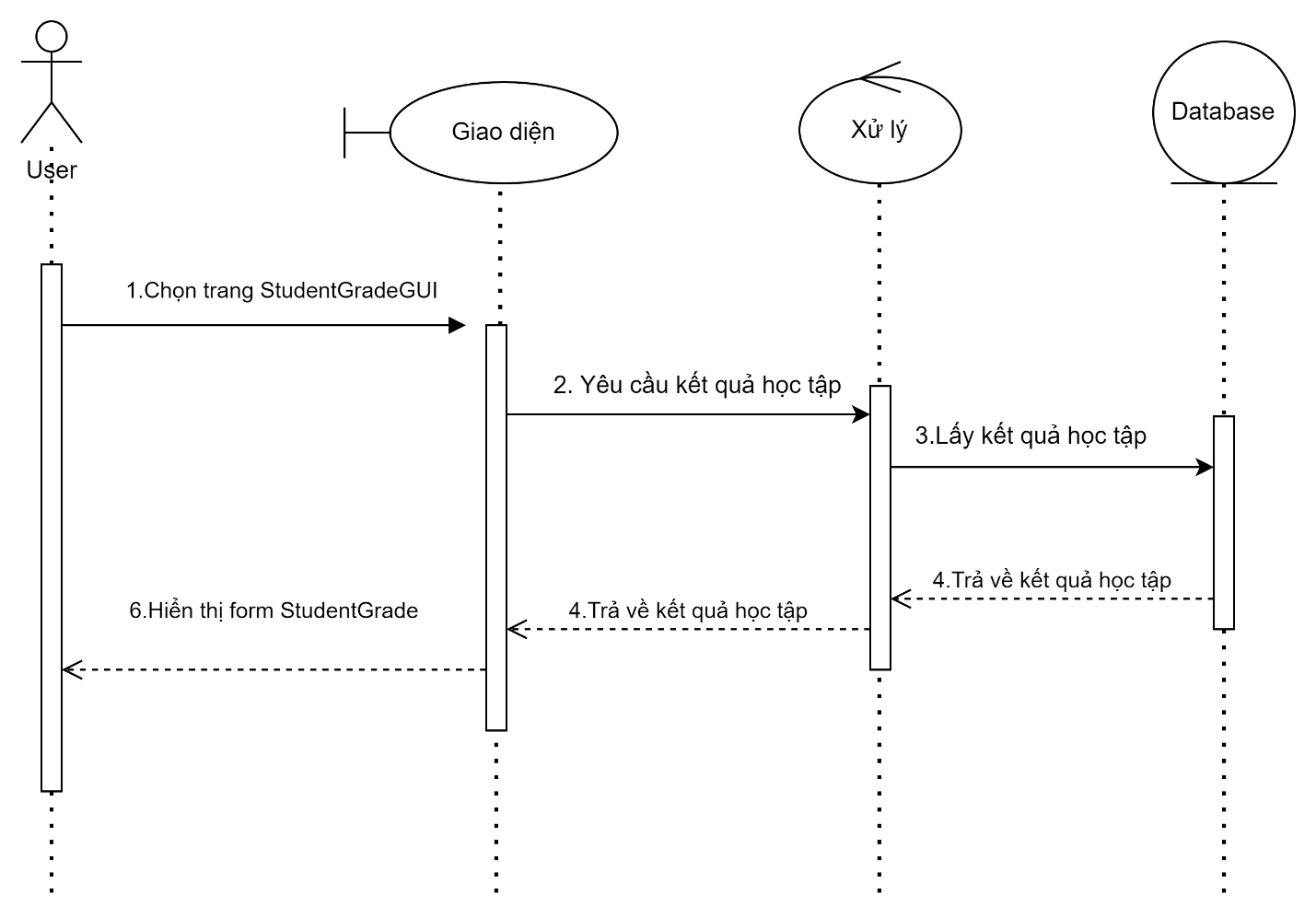
        }

 }

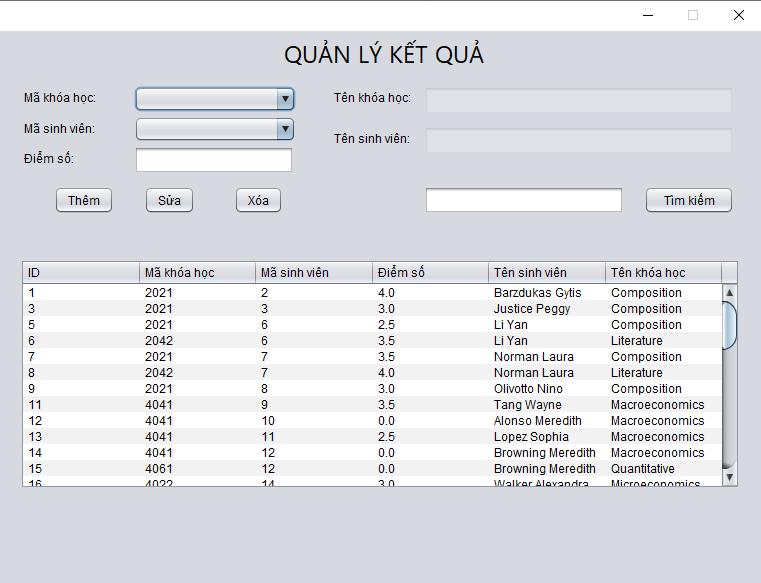
# **CHỨC NĂNG 3: QUẢN LÝ KẾT QUẢ KHÓA HỌC (STUDENTGRADE)**

## **3.1 Xử lý 1: hiển thị danh sách**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**- StudentGradeDAL**

public List<List<Object>> readStudent() {

        List<List<Object>> studentGradeList = new ArrayList<>();

        try {

            String sql = "SELECT sg.EnrollmentID, sg.CourseID, sg.StudentID, sg.Grade, p.Lastname, p.Firstname, c.Title "

                    + "FROM studentgrade sg "

                    + "INNER JOIN course c ON sg.CourseID = c.CourseID "

                    + "INNER JOIN person p ON sg.StudentID = p.PersonID "

                    + "ORDER BY sg.EnrollmentID ASC";

            ResultSet rs = this.doReadQuery(sql);

            if (rs != null) {

                while (rs.next()) {

                    List<Object> studentGradeInfo = new ArrayList<>();

                    studentGradeInfo.add(rs.getInt("EnrollmentID"));

                    studentGradeInfo.add(rs.getInt("CourseID"));

                    studentGradeInfo.add(rs.getInt("StudentID"));

                    studentGradeInfo.add(rs.getFloat("Grade"));

                    studentGradeInfo.add(rs.getString("Lastname") + " " + rs.getString("Firstname"));

                    studentGradeInfo.add(rs.getString("Title"));

                    studentGradeList.add(studentGradeInfo);

                }

            }

        } catch (SQLException e) {

            System.out.println("SQL Error: " + e.getMessage());

        }

        return studentGradeList;

    }

**- StudentGradeBLL**

public List<List<Object>> readStudent() {

        return studentGradeDAL.readStudent();

    }

**- StudentGradeGUI**

public StudentGradeGUI() {

        initComponents();

        fillTable();

        // Lấy danh sách khóa học từ CourseDAL

        CourseDAL courseDAL = new CourseDAL();

        courses = courseDAL.getCourseIDs();

        StudentBLL studentBLL = new StudentBLL();

        students = studentBLL.getStudentIDs();

        for (String course : courses) {

            cbMaKhoaHoc.addItem(course);

        }

//        txtPersonID.setEditable(false);

        for (String student : students) {

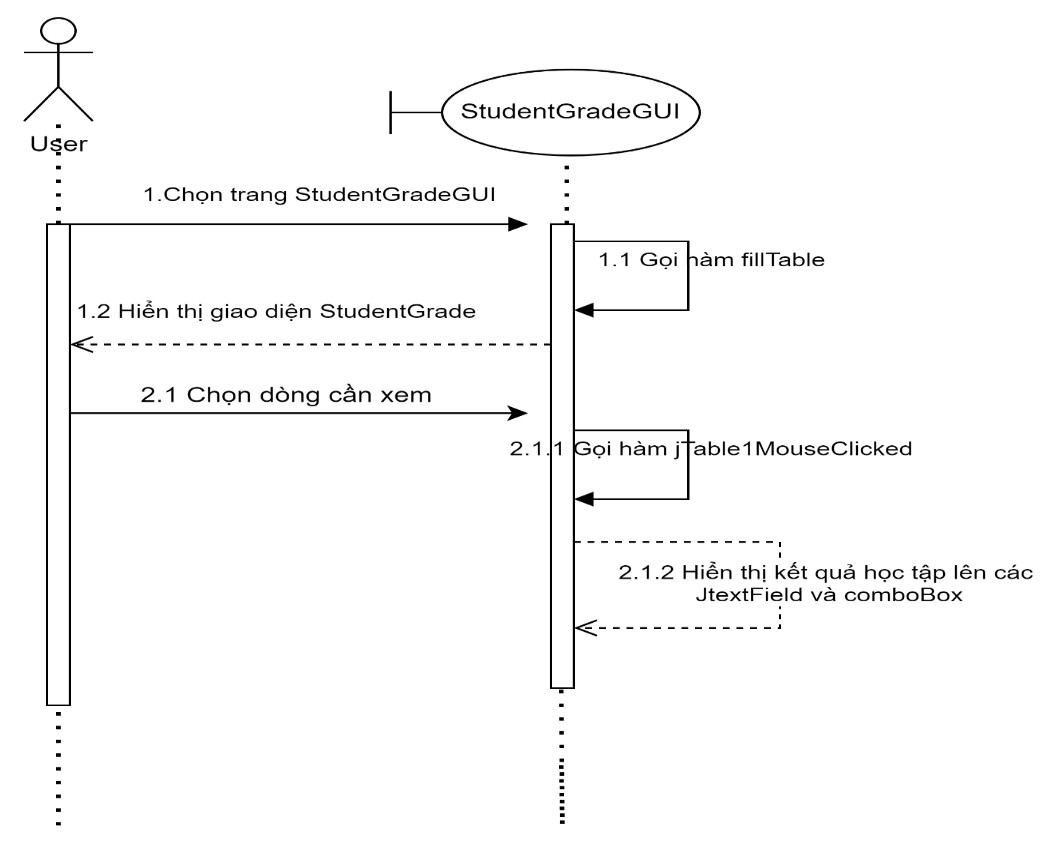
            cbMaSV.addItem(student);

        }

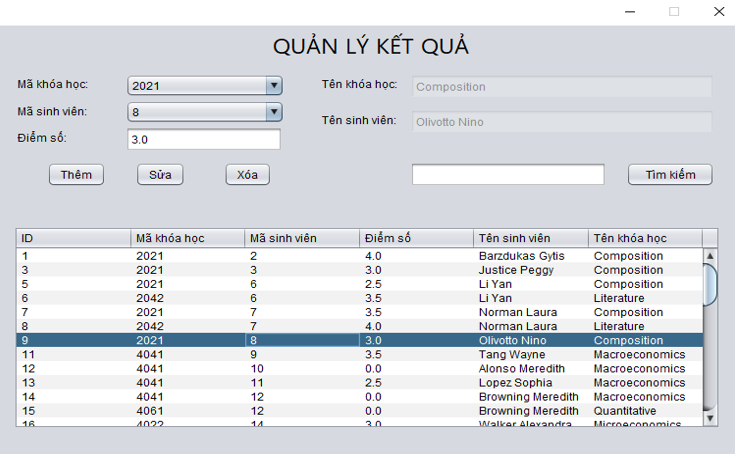
    }

## **3.2 Xử lý 2: hiển thị chi tiết 1 record (dòng)**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**- StudentGradeGUI**

private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jTable1MouseClicked

        int selectedRow = jTable1.getSelectedRow();

        if (selectedRow != -1) {

            int courseID = Integer.parseInt(jTable1.getValueAt(selectedRow, 1).toString());

            int studentID = Integer.parseInt(jTable1.getValueAt(selectedRow, 2).toString());

            String grade = jTable1.getValueAt(selectedRow, 3).toString();

            String studentName = jTable1.getValueAt(selectedRow, 4).toString();

            String courseName = jTable1.getValueAt(selectedRow, 5).toString();

            cbMaKhoaHoc.setSelectedItem(String.valueOf(courseID));

            cbMaSV.setSelectedItem(String.valueOf(studentID));

            txtTenKhoaHoc.setText(courseName);

            txtTenSV.setText(studentName);

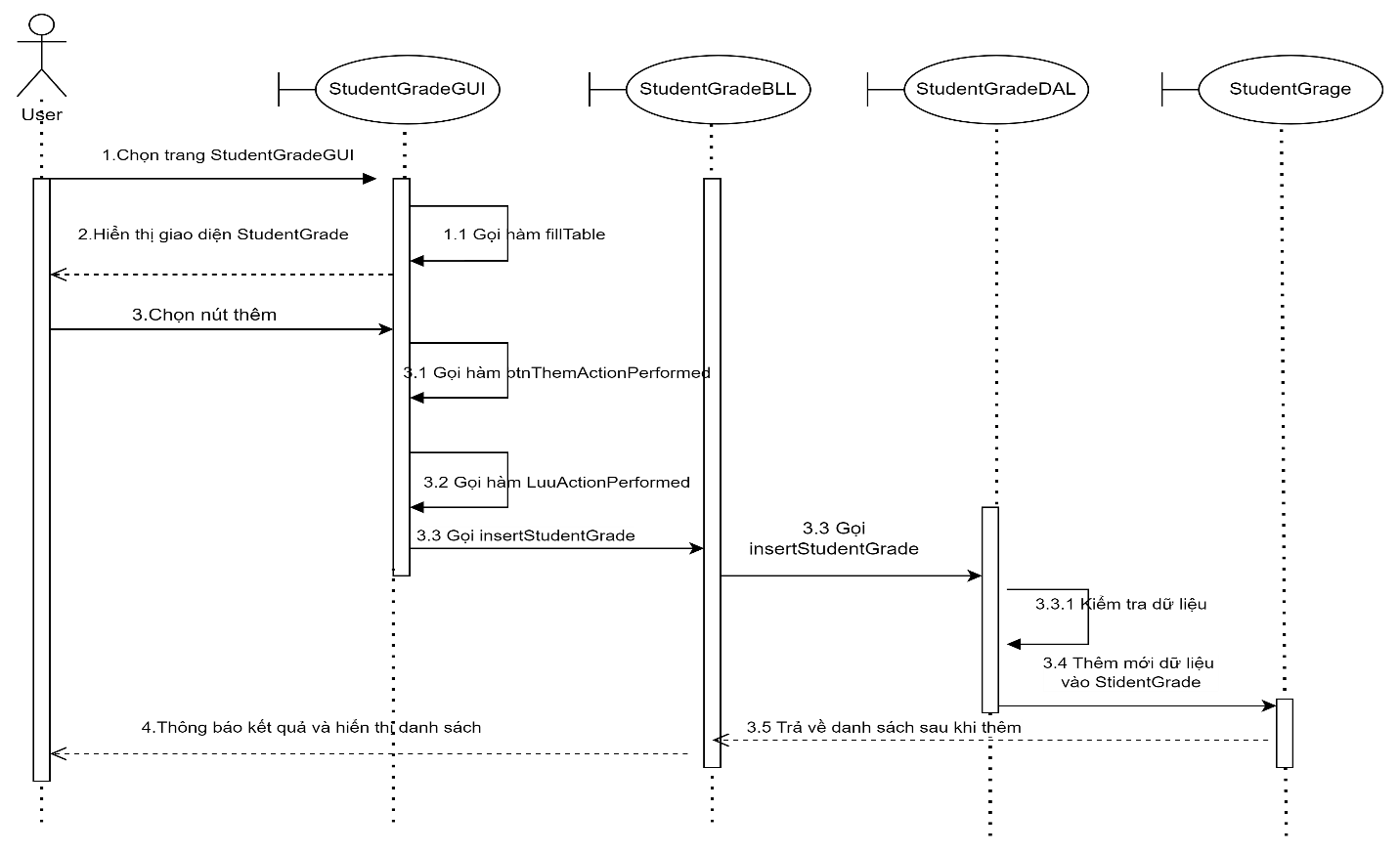
            txtDiemSo.setText(grade);

        }

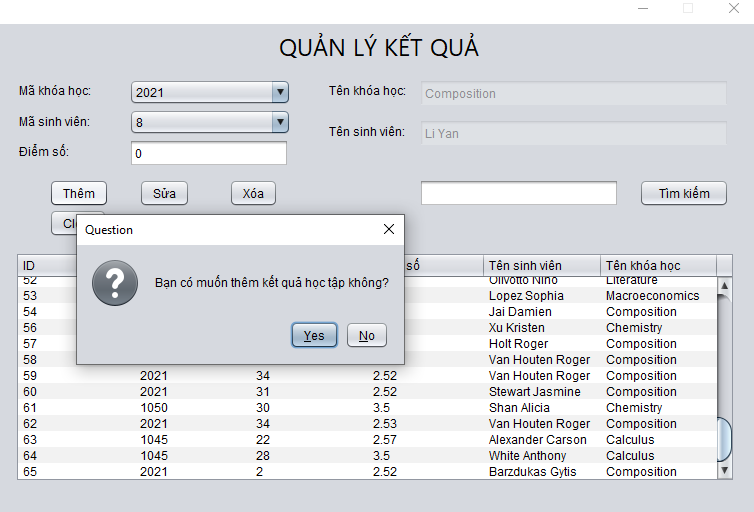
    }

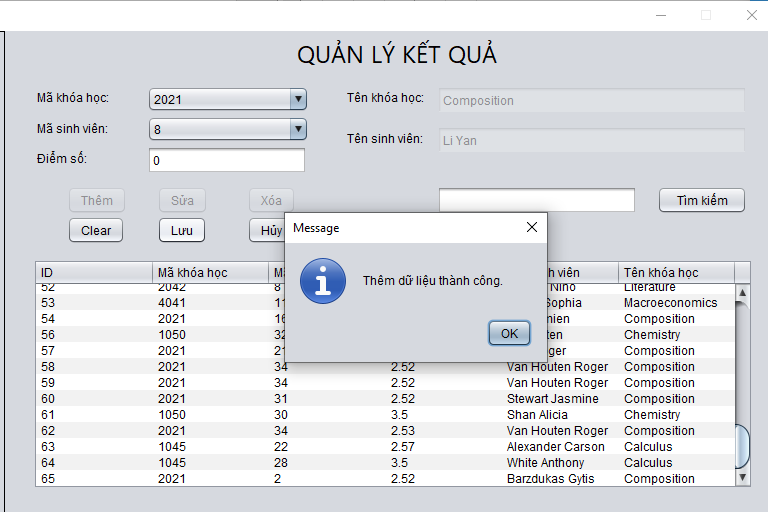
## **3.3 Xử lý 3: thêm mới thông tin**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**







* **Code 3 class: DAL, BLL, GUI**

**- StudentGradeDAL**

public int InsertStudentGradeDAL(int CourseID, int StudentID, float Grade) {

        try {

            String query = "INSERT INTO studentgrade(`CourseID`, `StudentID`, `Grade`) VALUES (?, ? ,?)";

            try (PreparedStatement pst = c.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {

                pst.setInt(1, CourseID);

                pst.setInt(2, StudentID);

                pst.setFloat(3, Grade);

                pst.executeUpdate();

                try (ResultSet generatedKeys = pst.getGeneratedKeys()) {

                    if (generatedKeys.next()) {

                        return generatedKeys.getInt(1);

                    }

                }

            }

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());

        }

        return 0;

    }

**- StudentGradeBLL**

public int insertStudentGrade(int courseID, int studentID, float grade) {

        return studentGradeDAL.InsertStudentGradeDAL(courseID, studentID, grade);

    }

**- StudentGradeGUI**

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

        // TODO add your handling code here:

        int luaChon = JOptionPane.showConfirmDialog(null, "Bạn có muốn thêm thay đổi quả học tập không?", "Question", JOptionPane.YES\_NO\_OPTION);

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

            insert = true;

            btnThem.setEnabled(false);

            btnSua.setEnabled(false);

            btnXoa.setEnabled(false);

            btnLuu.setVisible(true);

            btnHuy.setVisible(true);

        } else {

            System.out.println("Người dùng đã chọn Không");

        }

    }

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

        // Lấy dữ liệu được chọn từ bảng

        int selectedRow = jTable1.getSelectedRow();

        if (selectedRow >= 0) {

            // Lấy dữ liệu ban đầu

            int courseID = Integer.parseInt(jTable1.getValueAt(selectedRow, 1).toString());

            int studentID = Integer.parseInt(jTable1.getValueAt(selectedRow, 2).toString());

            float grade = Float.parseFloat(jTable1.getValueAt(selectedRow, 3).toString());

            // Lấy dữ liệu mới từ giao diện

            int newCourseID;

            int newStudentID;

            float newGrade;

            try {

                newCourseID = Integer.parseInt(cbMaKhoaHoc.getSelectedItem().toString());

                newStudentID = Integer.parseInt(cbMaSV.getSelectedItem().toString());

                newGrade = Float.parseFloat(txtDiemSo.getText());

                // Kiểm tra điểm số phải lớn hơn hoặc bằng 0 và bé hơn 4

                if (newGrade < 0 || newGrade >= 4) {

                    JOptionPane.showMessageDialog(this, "Điểm số phải lớn hơn hoặc bằng 0 và bé hơn 4.");

                    return;

                }

            } catch (NumberFormatException e) {

                JOptionPane.showMessageDialog(this, "Vui lòng nhập đúng định dạng số cho mã khóa học, mã sinh viên, và điểm số.");

                return;

            }

            // Kiểm tra xem có sự thay đổi không

            if (courseID != newCourseID || studentID != newStudentID || grade != newGrade) {

                // Nếu có sự thay đổi, thực hiện cập nhật dữ liệu

                if (insert) {

                    // Thực hiện thêm dữ liệu mới vào cơ sở dữ liệu

                    int enrollmentID = studentGradeBLL.insertStudentGrade(newCourseID, newStudentID, newGrade);

                    if (enrollmentID > 0) {

                        // Hiển thị thông báo khi thêm dữ liệu thành công

                        JOptionPane.showMessageDialog(this, "Thêm dữ liệu thành công.");

                        fillTable();

                        resetValue();

                    } else {

                        JOptionPane.showMessageDialog(this, "Thêm dữ liệu thất bại.");

                    }

                } else {

                    int enrollmentID = Integer.parseInt(jTable1.getValueAt(selectedRow, 0).toString());

                    boolean success = studentGradeBLL.editStudentGrade(enrollmentID, newCourseID, newStudentID, newGrade);

                    if (success) {

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

                        // Sau khi cập nhật thành công, làm mới bảng

                        fillTable();

                    } else {

                        JOptionPane.showMessageDialog(this, "Cập nhật thất bại.");

                    }

                }

                btnHuy.setVisible(false);

                btnLuu.setVisible(false);

                btnThem.setEnabled(true);

                btnSua.setEnabled(true);

                btnXoa.setEnabled(true);

                insert = false;

            } else {

                // Nếu không có sự thay đổi, hiển thị thông báo "Không có sự thay đổi!"

                JOptionPane.showMessageDialog(this, "Không có sự thay đổi!");

            }

        } else {

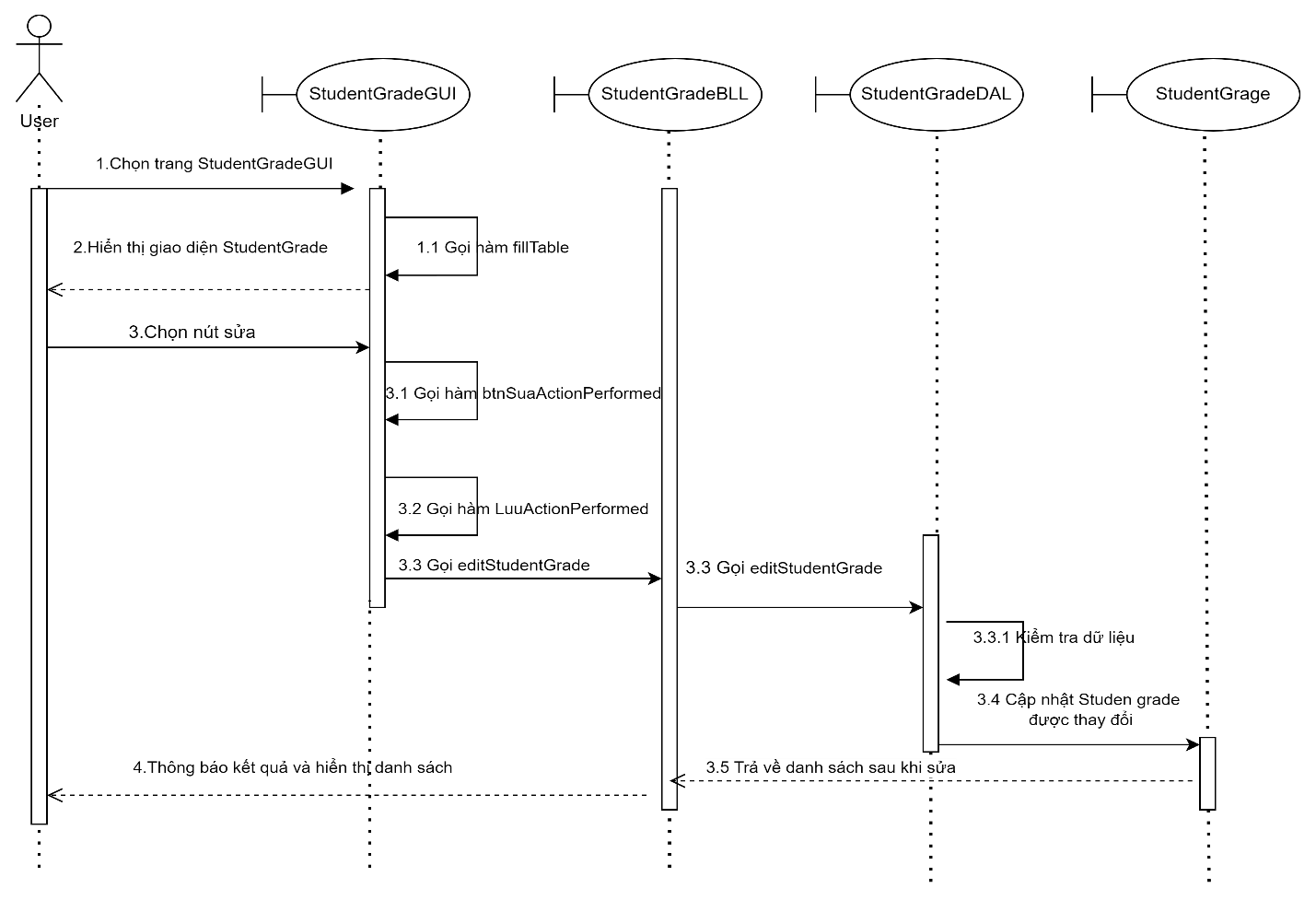
            JOptionPane.showMessageDialog(this, "Vui lòng điền đầy đủ thông tin.");

        }

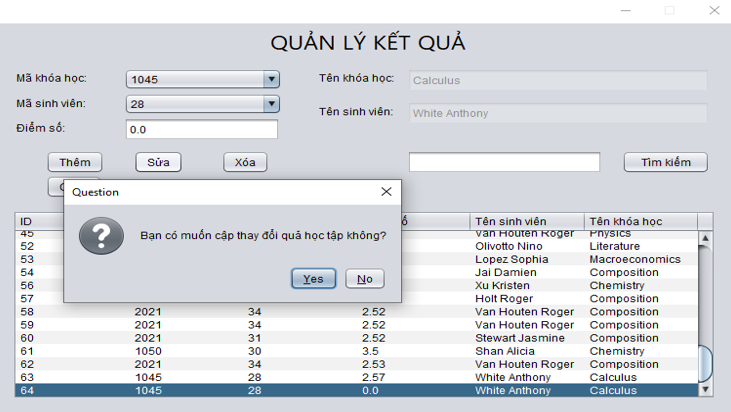
    }

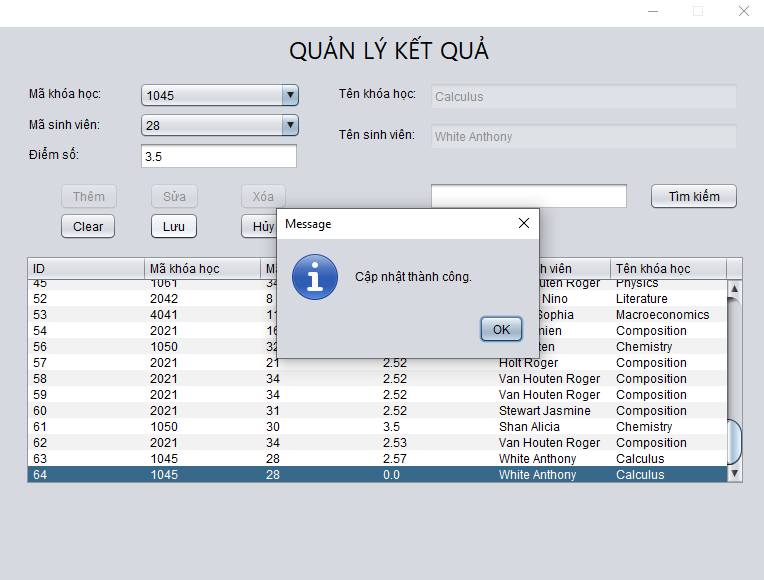
## **3.4 Xử lý 4: sửa/ cập nhật thông tin**

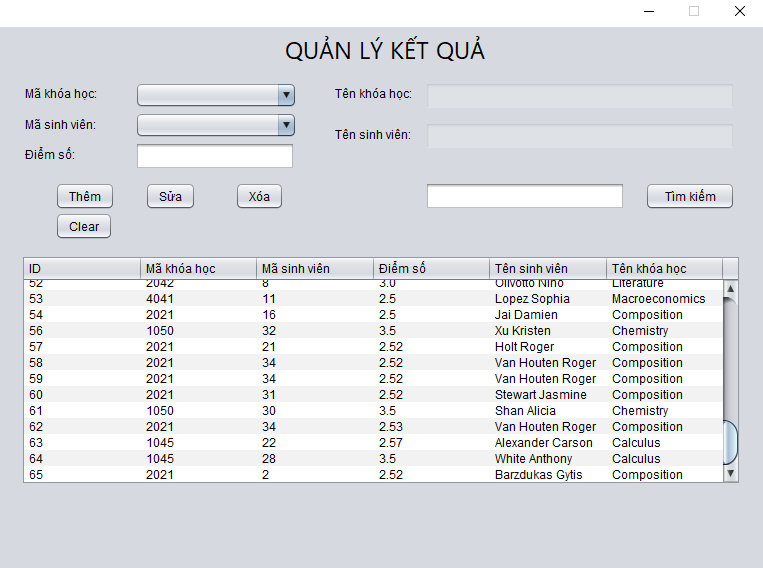
* **Sơ đồ tuần tự**



* **Màn hình giao diện**







* **Code 3 class: DAL, BLL, GUI**

**- StudentGradeDAL**

public boolean editStudentGrade(int EnrollmentID, int CourseID, int StudentID, float Grade) {

        try {

            String query = "UPDATE studentgrade SET `CourseID`=?,`StudentID`=?,`Grade`=? WHERE `EnrollmentID` = '" + EnrollmentID + "'";

            PreparedStatement pst = c.prepareStatement(query);

            pst.setInt(1, CourseID);

            pst.setInt(2, StudentID);

            pst.setFloat(3, Grade);

            pst.executeUpdate();

            return true;

        } catch (SQLException e) {

        }

        return false;

    }

**- StudentGradeBLL**

public boolean editStudentGrade(int enrollmentID, int courseID, int studentID, float grade) {

        return studentGradeDAL.editStudentGrade(enrollmentID, courseID, studentID, grade);

    }

**- StudentGradeGUI**

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

        int luaChon = JOptionPane.showConfirmDialog(null, "Bạn có muốn cập thay đổi quả học tập không?", "Question", JOptionPane.YES\_NO\_OPTION);

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

            btnThem.setEnabled(false);

            btnSua.setEnabled(false);

            btnXoa.setEnabled(false);

            btnLuu.setVisible(true);

            btnHuy.setVisible(true);

        } else {

            System.out.println("Người dùng đã chọn Không");

        }

    }

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

        // Lấy dữ liệu được chọn từ bảng

        int selectedRow = jTable1.getSelectedRow();

        if (selectedRow >= 0) {

            // Lấy dữ liệu ban đầu

            int courseID = Integer.parseInt(jTable1.getValueAt(selectedRow, 1).toString());

            int studentID = Integer.parseInt(jTable1.getValueAt(selectedRow, 2).toString());

            float grade = Float.parseFloat(jTable1.getValueAt(selectedRow, 3).toString());

            // Lấy dữ liệu mới từ giao diện

            int newCourseID;

            int newStudentID;

            float newGrade;

            try {

                newCourseID = Integer.parseInt(cbMaKhoaHoc.getSelectedItem().toString());

                newStudentID = Integer.parseInt(cbMaSV.getSelectedItem().toString());

                newGrade = Float.parseFloat(txtDiemSo.getText());

                // Kiểm tra điểm số phải lớn hơn hoặc bằng 0 và bé hơn 4

                if (newGrade < 0 || newGrade >= 4) {

                    JOptionPane.showMessageDialog(this, "Điểm số phải lớn hơn hoặc bằng 0 và bé hơn 4.");

                    return;

                }

            } catch (NumberFormatException e) {

                JOptionPane.showMessageDialog(this, "Vui lòng nhập đúng định dạng số cho mã khóa học, mã sinh viên, và điểm số.");

                return;

            }

            // Kiểm tra xem có sự thay đổi không

            if (courseID != newCourseID || studentID != newStudentID || grade != newGrade) {

                // Nếu có sự thay đổi, thực hiện cập nhật dữ liệu

                if (insert) {

                    // Thực hiện thêm dữ liệu mới vào cơ sở dữ liệu

                    int enrollmentID = studentGradeBLL.insertStudentGrade(newCourseID, newStudentID, newGrade);

                    if (enrollmentID > 0) {

                        // Hiển thị thông báo khi thêm dữ liệu thành công

                        JOptionPane.showMessageDialog(this, "Thêm dữ liệu thành công.");

                        fillTable();

                        resetValue();

                    } else {

                        JOptionPane.showMessageDialog(this, "Thêm dữ liệu thất bại.");

                    }

                } else {

                    int enrollmentID = Integer.parseInt(jTable1.getValueAt(selectedRow, 0).toString());

                    boolean success = studentGradeBLL.editStudentGrade(enrollmentID, newCourseID, newStudentID, newGrade);

                    if (success) {

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

                        // Sau khi cập nhật thành công, làm mới bảng

                        fillTable();

                    } else {

                        JOptionPane.showMessageDialog(this, "Cập nhật thất bại.");

                    }

                }

                btnHuy.setVisible(false);

                btnLuu.setVisible(false);

                btnThem.setEnabled(true);

                btnSua.setEnabled(true);

                btnXoa.setEnabled(true);

                insert = false;

            } else {

                // Nếu không có sự thay đổi, hiển thị thông báo "Không có sự thay đổi!"

                JOptionPane.showMessageDialog(this, "Không có sự thay đổi!");

            }

        } else {

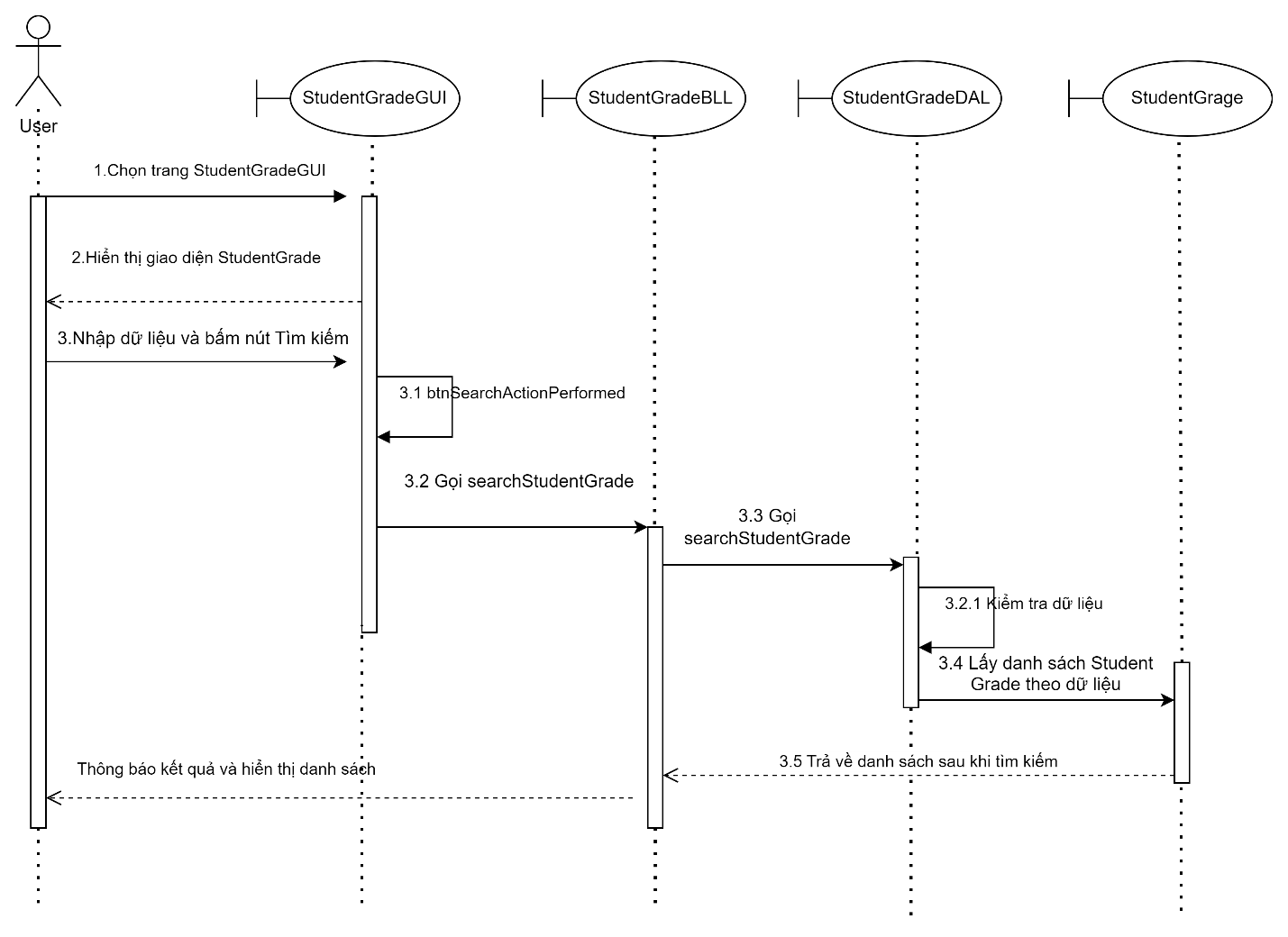
            JOptionPane.showMessageDialog(this, "Vui lòng điền đầy đủ thông tin.");

        }

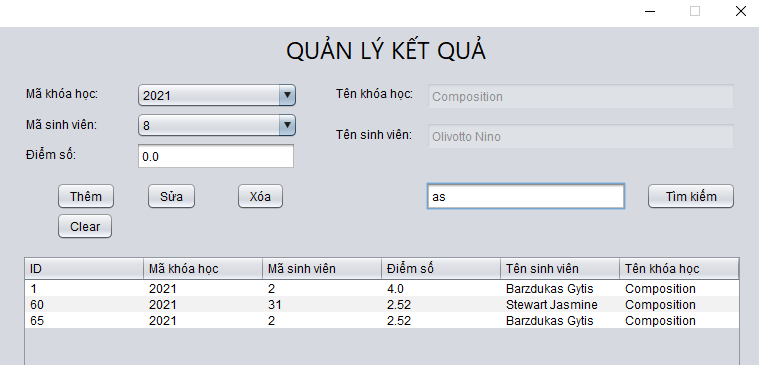
    }

## **3.5 Xử lý 5: tìm kiếm**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**



* **Code 3 class: DAL, BLL, GUI**

**- StudentGradeDAL**

public List<List<Object>> searchStudentGrade(String keyword) {

        List<List<Object>> searchResult = new ArrayList<>();

        try {

            String sql = "SELECT sg.EnrollmentID, sg.CourseID, sg.StudentID, sg.Grade, p.Lastname, p.Firstname, c.Title "

                    + "FROM studentgrade sg "

                    + "INNER JOIN course c ON sg.CourseID = c.CourseID "

                    + "INNER JOIN person p ON sg.StudentID = p.PersonID "

                    + "WHERE sg.CourseID LIKE ? "

                    + "OR sg.StudentID LIKE ? "

                    + "OR sg.Grade LIKE ? "

                    + "OR CONCAT\_WS(' ', p.Lastname, p.Firstname) LIKE ? "

                    + "OR c.Title LIKE ? "

                    + "ORDER BY sg.EnrollmentID ASC";

            PreparedStatement pst = c.prepareStatement(sql);

            pst.setString(1, "%" + keyword + "%");

            pst.setString(2, "%" + keyword + "%");

            pst.setString(3, "%" + keyword + "%");

            pst.setString(4, "%" + keyword + "%");

            pst.setString(5, "%" + keyword + "%");

            ResultSet rs = pst.executeQuery();

            while (rs.next()) {

                List<Object> studentGradeInfo = new ArrayList<>();

                studentGradeInfo.add(rs.getInt("EnrollmentID"));

                studentGradeInfo.add(rs.getInt("CourseID"));

                studentGradeInfo.add(rs.getInt("StudentID"));

                studentGradeInfo.add(rs.getFloat("Grade"));

                studentGradeInfo.add(rs.getString("Lastname") + " " + rs.getString("Firstname"));

                studentGradeInfo.add(rs.getString("Title"));

                searchResult.add(studentGradeInfo);

            }

        } catch (SQLException e) {

            System.out.println("SQL Error: " + e.getMessage());

        }

        return searchResult;

    }

**- StudentGradeBLL**

public List<List<Object>> searchStudentGrade(String keyword) {

        return studentGradeDAL.searchStudentGrade(keyword);

    }

**- StudentGradeGUI**

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

        String searchQuery = jtfSearch.getText().trim();

        DefaultTableModel tableModel = (DefaultTableModel) jTable1.getModel();

        tableModel.setRowCount(0);

        if (!searchQuery.isEmpty()) {

            List<List<Object>> searchResult = studentGradeBLL.searchStudentGrade(searchQuery);

            if (searchResult != null) {

                tableModel.setRowCount(0);

                for (List<Object> studentGradeInfo : searchResult) {

                    tableModel.addRow(studentGradeInfo.toArray());

                }

            }

        } else {

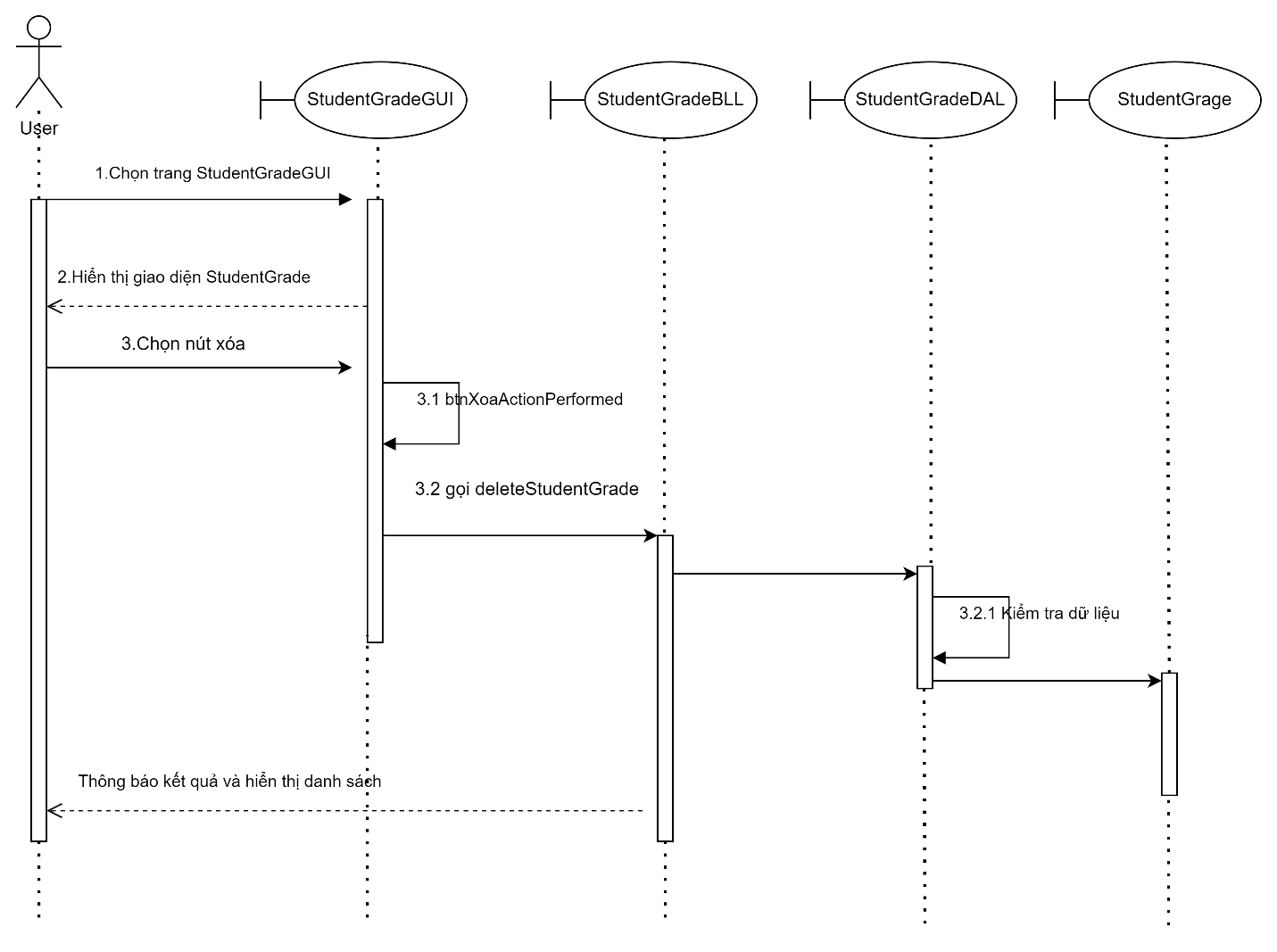
            fillTable();

        }

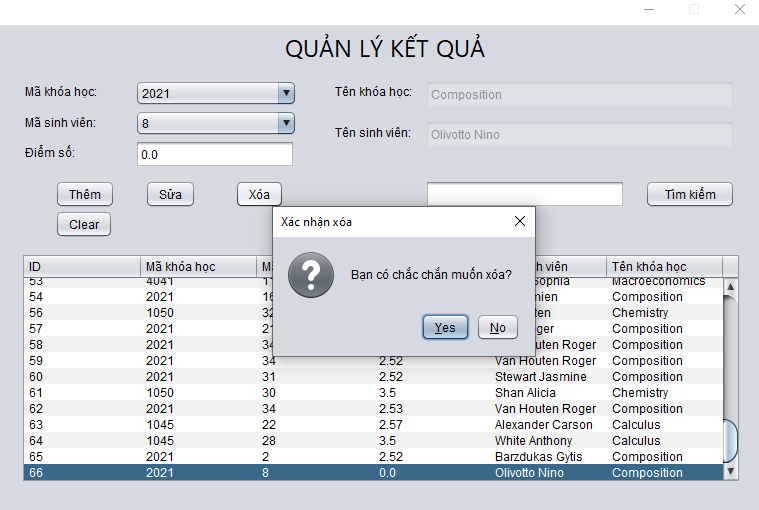
    }

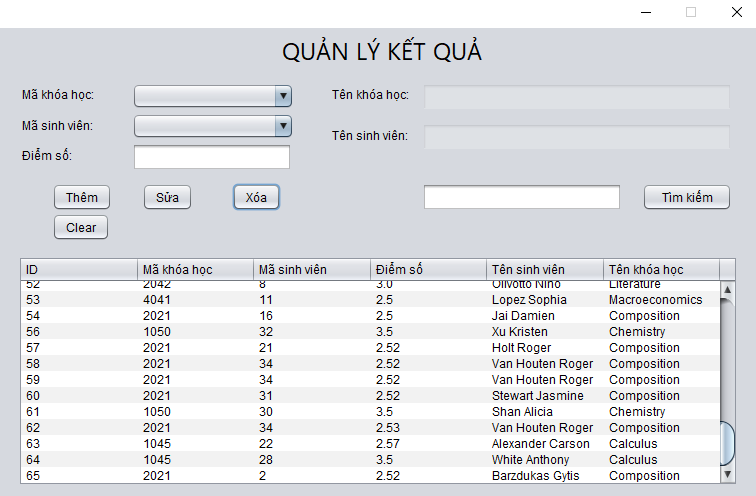
## **3.6 Xử lý 6: xóa record**

* **Sơ đồ tuần tự**



* **Màn hình giao diện**





* **Code 3 class: DAL, BLL, GUI**

**- StudentGradeDAL**

public boolean deleteStudentgradeDAL(int EnrollmentID) {

        try {

            String query = "DELETE FROM studentgrade WHERE `EnrollMentID` = '" + EnrollmentID + "'";

            PreparedStatement pst = c.prepareStatement(query);

            pst.executeUpdate();

            return true;

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(null, "Error");

        }

        return false;

    }

**- StudentGradeBLL**

public boolean deleteStudentGrade(int enrollmentID) {

        return studentGradeDAL.deleteStudentgradeDAL(enrollmentID);

    }

**- StudentGradeGUI**

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

        int selectedRow = jTable1.getSelectedRow();

            int enrollmentID = Integer.parseInt(jTable1.getValueAt(selectedRow, 0).toString());

            boolean success = studentGradeBLL.deleteStudentGrade(enrollmentID);

            if (success) {

                JOptionPane.showMessageDialog(this, "Data deleted successfully.");

            } else {

                JOptionPane.showMessageDialog(this, "Failed to delete data.");

            }

        } else {

            JOptionPane.showMessageDialog(this, "Please select a row to delete.");

        }

    }