QUẢN LÝ KHÓA HỌC

DANH SÁCH THÀNH VIÊN

NGUYỄN THỊ YẾN NHI	3119410288
NGUYỄN THỊ PHƯƠNG NHUNG	3119410291
ĐẶNG THỊ NGỌC NGÂN	3119410263
MAI VĂN THINH	3119410413
TRẦN THÚY NGÂN	3119410265
NGUYỄN LÊ TÂM NHƯ	3119410292
LÊ HOÀI LÂN	3119410223
PHẠM DUY LUÂN	3119410240
DIỆC TRÁC LÂM	3119410220

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

MỤC LỤC

PHÂN CÔNG THÀNH VIÊN

Tên thành viên	Nhiệm vụ		Phần trăm
Nguyễn Thị Yến Nhi	Quản lý Project: + Phân chia yêu cầu theo dõi tiến độ + Hỗ trợ kỹ thuật code, các vấn đề khó khăn + Quản lý code chung Chỉnh sửa thiết kế mẫu giao diện Tổng hợp code và test chức năng		11.7%
Đặng Thị Ngọc Ngân	+ Yêu cầu chức năng 1: Quản lý Person	+ Thiết kế giao diện + Code chức năng (hiển thị, tìm kiếm, thêm, sửa) của Teacher + Vẽ sơ đồ class, sơ đồ trình tự + Làm tài liệu báo cáo	11.7%
Mai Văn Thinh	+ Làm tài liệu báo cáo	+ Code chức năng (hiển thị, tìm kiếm, thêm, sửa, xóa) của Student, chức năng Xóa của Teacher + Chỉnh sửa giao diện	11.4%
Nguyễn Lê Tâm Như		+ Code chức năng (hiển thị, tìm kiếm, thêm, sửa , xóa) + Giao diện	11.7%

	Yêu cầu chức năng 2: Quản lý Course	+ Vẽ sơ đồ class + Tài liệu chức năng Course	
Phạm Duy Luân		+ Code chức năng (hiển thị, tìm kiếm, thêm, sửa) Online Course + Vẽ sơ đồ trình tự	8.2%
Nguyễn Thị Phương Nhung	+ Yêu cầu chức năng 3: Quản lý Course Instructor	+ Vẽ sơ đồ class, sơ đồ trình tự + Thiết kế và làm giao diện Menu + Tổng hợp tài liệu chức năng Course Instructor + Chỉnh sửa giao diện chức năng Course Instructor	11.4%
Lê Hoài Lân	+ Tổng hợp chỉnh sửa code	+ Code chức năng (hiển thị, tìm kiếm, thêm, sửa, xóa) + Code chức năng main code (menu chính) + Vẽ giao diện chức năng Course Instructor	11.7%
Trần Thúy Ngân		+ Thiết kế giao diện + Vẽ sơ đồ class, sơ đồ trình tự	11.1%

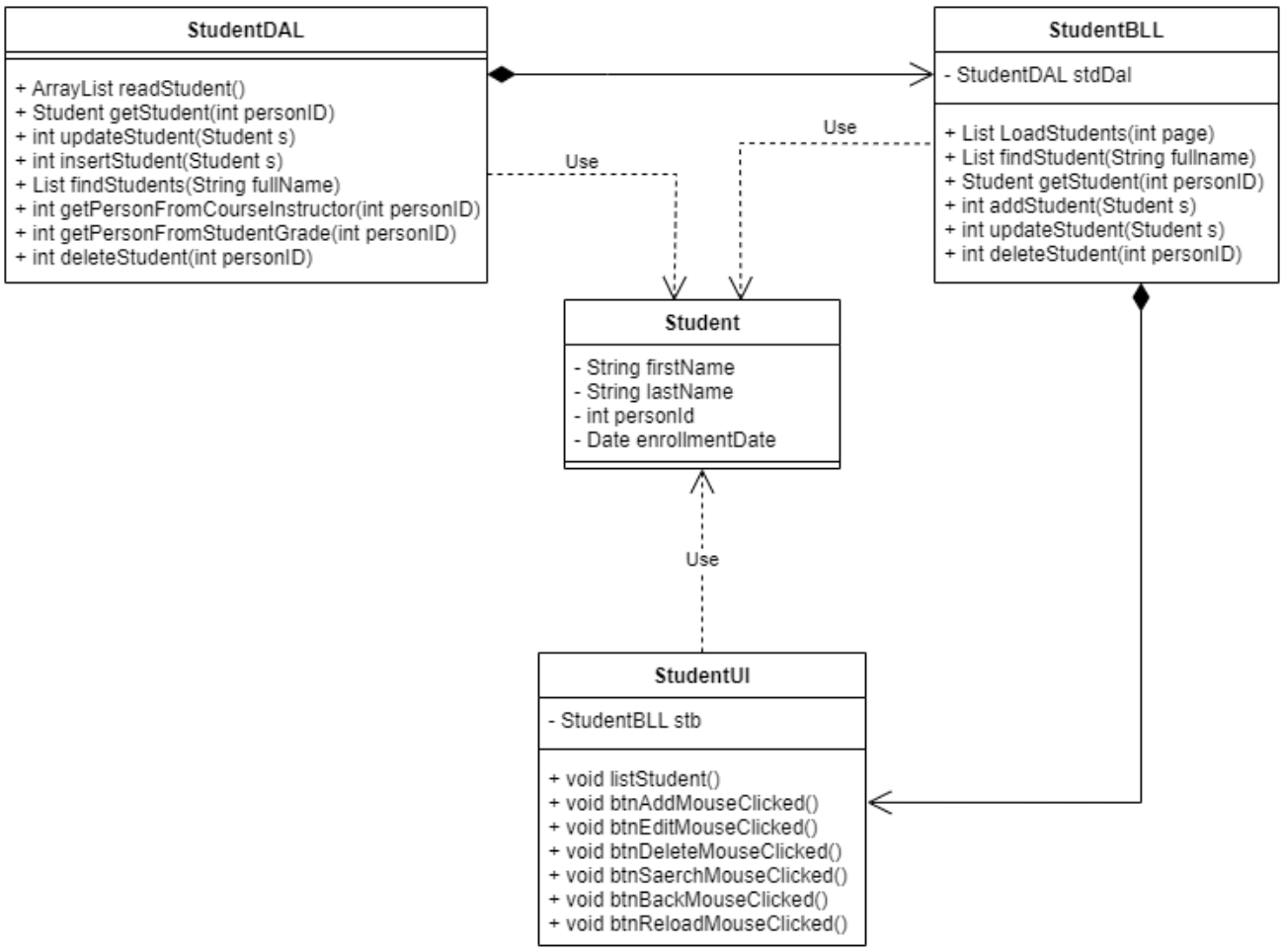
	Yêu cầu chức năng 4: Quản lý Student Grade	+ Làm tài liệu cho chức năng 4	
Diệc Trác Lâm		Code chức năng (hiển thị, tìm kiếm, thêm, sửa , xóa)	11.1%

QUẢN LÝ KHÓA HỌC

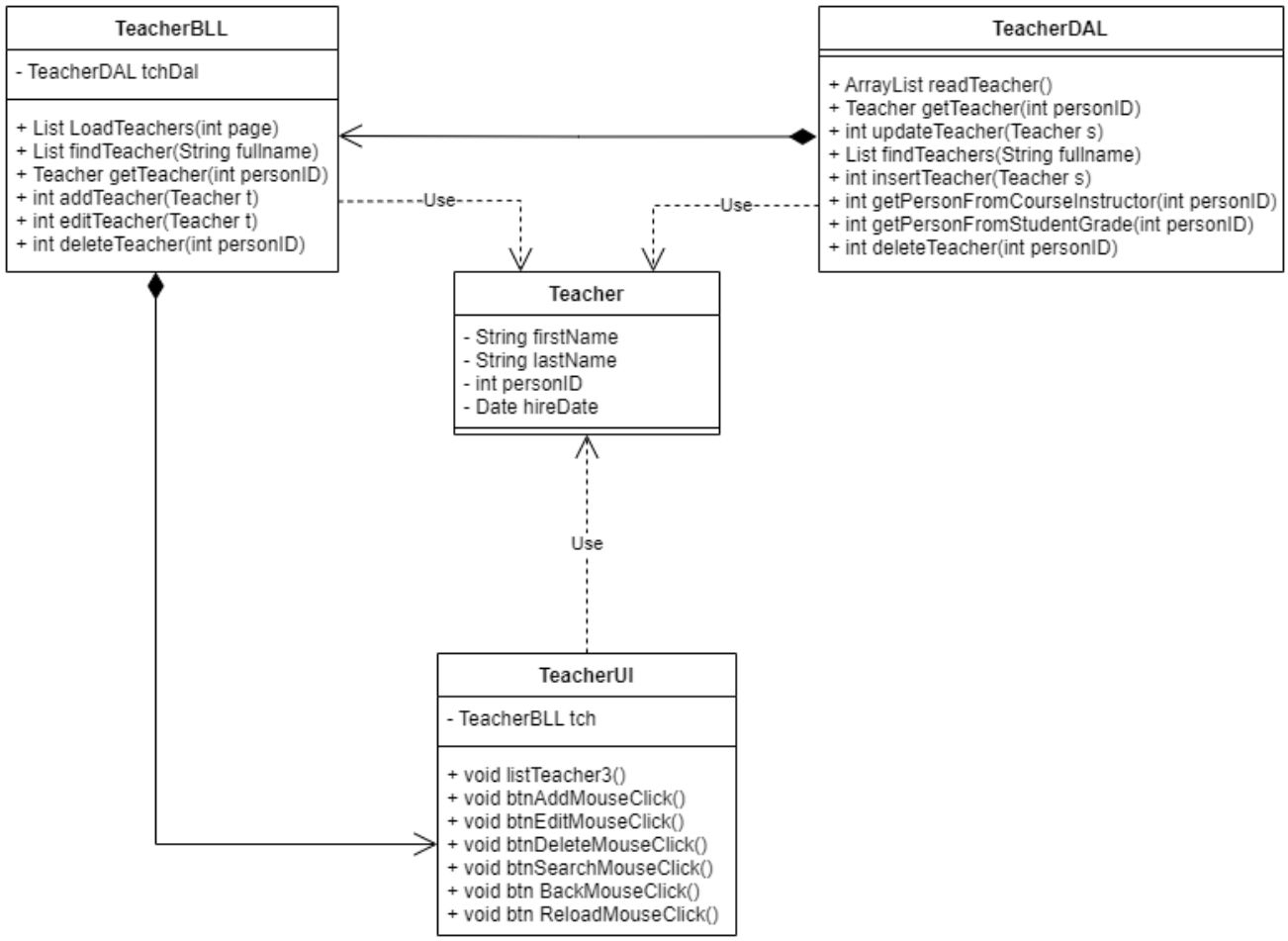
I. Chức năng 1: Quản lý sinh viên, người dạy (Person)

1. Sơ đồ class

1.1. Sinh viên

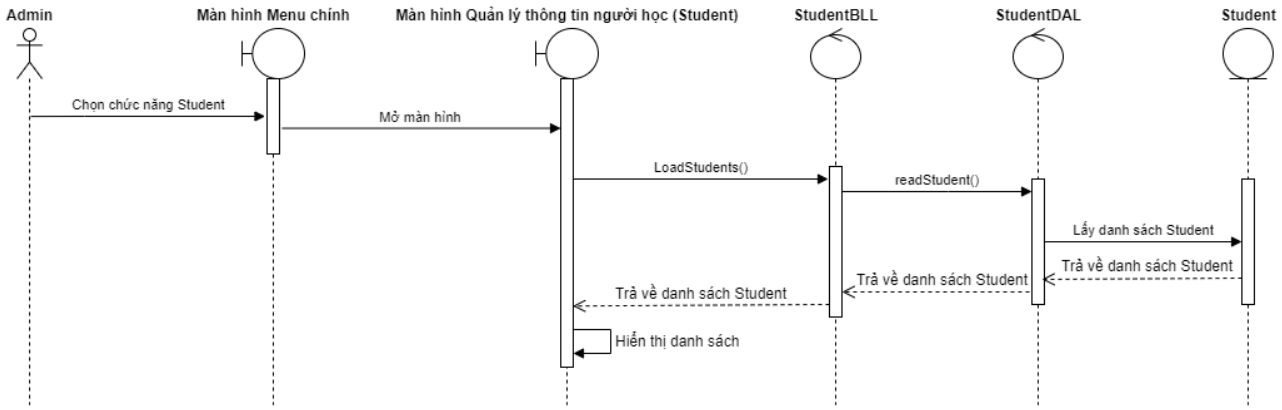


1.2. Người dạy



2. Hiển thị danh sách sinh viên

2.1. Sơ đồ tuần tự



2.2. Code 3 class:

2.2.1. DAL

```
public ArrayList readStudent() throws SQLException {
    String query = "SELECT * FROM Person WHERE EnrollmentDate > 0";
    ResultSet rs = StudentDAL.doReadQuery( sql: query);
    ArrayList list = new ArrayList();

    if (rs != null) {
        while (rs.next()) {
            Student s = new Student();
            s.setPersonId( personId: rs.getInt( columnLabel: "PersonID"));
            s.setFirstName( firstName: rs.getString( columnLabel: "FirstName"));
            s.setLastName( lastName: rs.getString( columnLabel: "LastName"));
            s.setEnrollmentDate( enrollmentDate: Date.valueOf(rs.getString( columnLabel: "EnrollmentDate").split( regex: " ")[0]));
            list.add( e: s);
        }
    }
    return list;
}
```

2.2.2. BLL

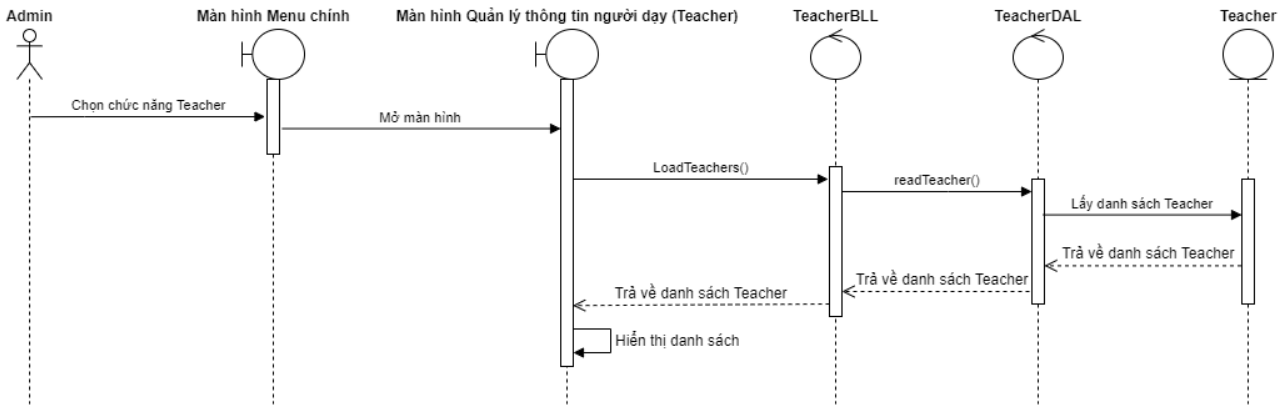
```
public List LoadStudents() throws SQLException {
    ArrayList list = stdDal.readStudent();
    return list;
}
```

2.2.3. UI

```
public StudentForm() {
    this.setTitle( title: "Student");
    initComponents();
    tbDS.fixTable( scroll: jScrollPane1);
    getContentPane().setBackground( c: Color.WHITE);
    try {
        listStudent();
    } catch (SQLException ex) {
        Logger.getLogger( name: StudentForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
}

private void listStudent() throws SQLException {
    List list = stb.LoadStudents();
    DefaultTableModel model = convertStudent(list);
    tbDS.setModel( dataModel: model);
}

private DefaultTableModel convertStudent(List list) {
    String[] columnNames = {"Person ID", "First Name", "Last Name", "Enrollment Date"};
    Object[][] data = new Object[list.size()][4];
    for (int i = 0; i < list.size(); i++) {
        Student s = (Student) list.get( index: i);
        data[i][0] = s.getPersonId();
        data[i][1] = s.getFirstName();
        data[i][2] = s.getLastName();
        data[i][3] = s.getEnrollmentDate();
    }
    DefaultTableModel model = new DefaultTableModel(data, columnNames);
    return model;
}
```

3. Hiển thị danh sách người dạy

3.1. Sơ đồ tuần tự



3.2. Code 3 class

3.3.1. DAL

```
public ArrayList readTeacher() throws SQLException {
    String query = "SELECT * FROM Person WHERE HireDate > 0";
    ResultSet rs = TeacherDAL.doReadQuery( sql: query);
    ArrayList list = new ArrayList();

    if (rs != null) {
        while (rs.next()) {
            Teacher s = new Teacher();
            s.setPersonID( personID: rs.getInt( columnLabel: "PersonID"));
            s.setFirstName( firstName: rs.getString( columnLabel: "FirstName"));
            s.setLastName( lastName: rs.getString( columnLabel: "LastName"));
            s.setHireDate( hireDate: Date.valueOf(rs.getString( columnLabel: "HireDate").split( regex: " ")[0]));
            list.add( e: s);
        }
    }
    return list;
}
```

3.3.2. BLL

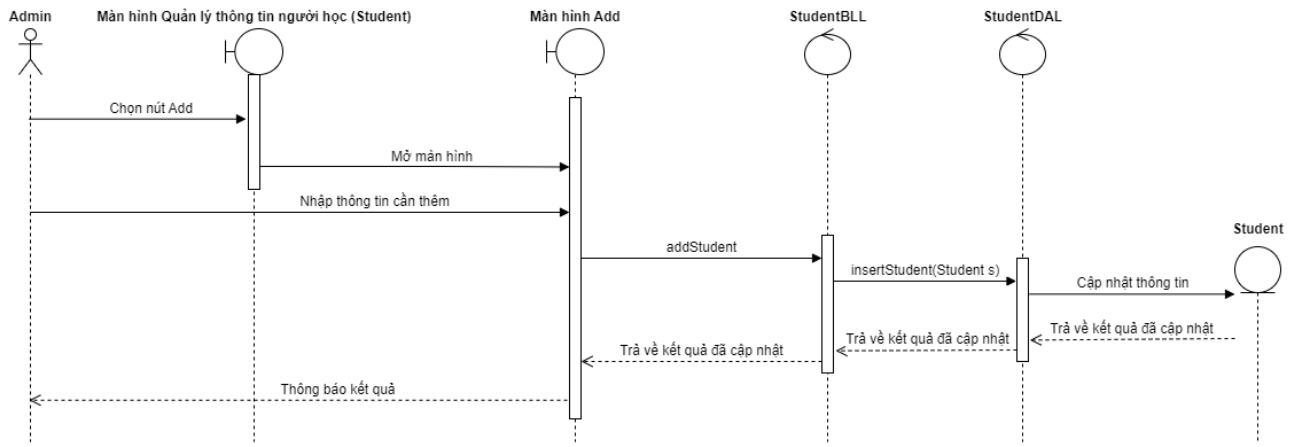
```
public List LoadTeachers() throws SQLException {
    ArrayList list = tchDal.readTeacher();
    return list;
}
```

3.3.3. UI

```
public TeacherForm() {
    this.setTitle( title: "Teacher");
    initComponents();
    tbDS.fixTable(jScrollPane1);
    getContentPane().setBackground( c: Color.WHITE);

    try {
        listTeacher3();
    } catch (SQLException ex) {
        Logger.getLogger( name: TeacherForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}

private void listTeacher3() throws SQLException {
    List list = tch.LoadTeachers();
    DefaultTableModel model = convertTeacher(list);
    tbDS.setModel(model);
}

private DefaultTableModel convertTeacher(List list) {
    String[] columnNames = {"Person ID", "First Name", "Last Name", "Hire Date"};
    Object[][] data = new Object[list.size()][5];
    for (int i = 0; i < list.size(); i++) {
        Teacher t = (Teacher) list.get( index: i);
        data[i][0] = t.getPersonID();
        data[i][1] = t.getFirstName();
        data[i][2] = t.getLastName();
        data[i][3] = t.getHireDate();
    }
    DefaultTableModel model = new DefaultTableModel(data, columnNames);
    return model;
}
```

4. Thêm mới thông tin sinh viên

4.1. Sơ đồ tuần tự



4.2. Code 3 class

4.2.1. DAL

```
public int insertStudent(Student s) throws SQLException {
    String query = "Insert Person (FirstName, LastName, EnrollmentDate) VALUES (?, ?, ?)";
    PreparedStatement p = StudentDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1,  x: s.getFirstName());
    p.setString( parameterIndex: 2,  x: s.getLastName());
    p.setString( parameterIndex: 3,  x: s.getEnrollmentDate().toString());
    int result = p.executeUpdate();
    return result;
}
```

4.2.2. BLL

```
public int addStudent(Student s) throws SQLException {
    int result = stdDal.insertStudent(s);
    return result;
}
```

4.2.3. UI

- StudentAddForm:

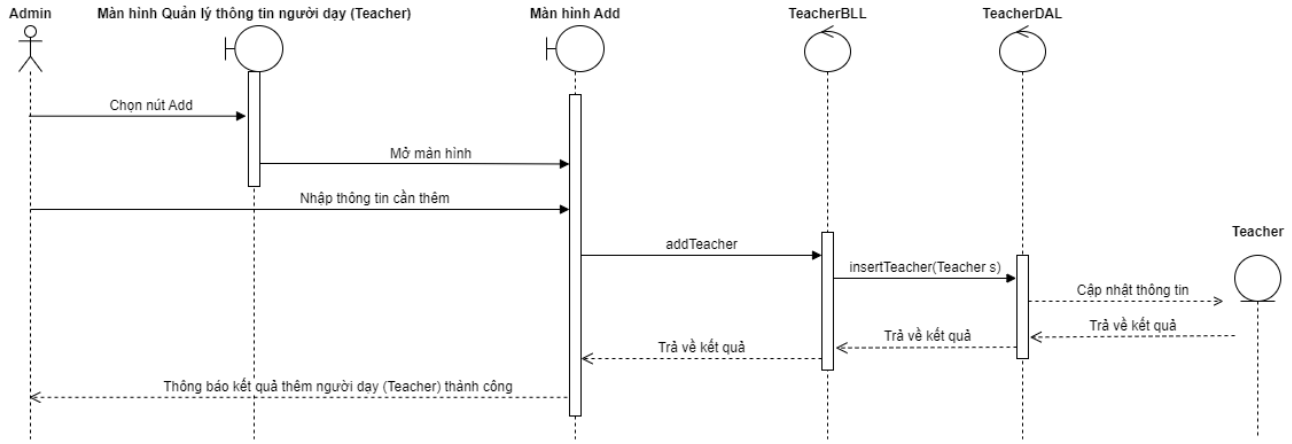
```
private void btnSaveMouseClicked(java.awt.event.MouseEvent evt) {
    Student s = new Student();
    s.setFirstName( firstName: txtFirstName.getText());
    s.setLastName( lastName: txtLastName.getText());
    Date date = Date.valueOf( s: txtEnrollmentDate.getText());
    s.setEnrollmentDate( enrollmentDate: date);

    try {
        if (stb.addStudent(s) > 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "You have completed to add student successfully!",
                    title: "Message",  messageType: JOptionPane.PLAIN_MESSAGE);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: StudentAddForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

- StudentForm:

```
private void btnAddMouseClicked(java.awt.event.MouseEvent evt) {
    StudentAddForm addform = new StudentAddForm();
    addform.setVisible( b: true);
}
```

5. Thêm mới thông tin người dạy

5.1. Sơ đồ tuần tự



5.2. Code 3 class

5.2.1. DAL

```
public int insertTeacher(Teacher s) throws SQLException {
    String query = "Insert Person (FirstName, LastName, HireDate) VALUES (?, ?, ?)";
    PreparedStatement p = TeacherDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1,  x: s.getFirstName());
    p.setString( parameterIndex: 2,  x: s.getLastName());
    p.setString( parameterIndex: 3,  x: s.getHireDate().toString());
    int result = p.executeUpdate();
    return result;
}
```

5.2.2. BLL

```
public int addTeacher(Teacher t) throws SQLException {
    int result = tchDal.insertTeacher( s: t);
    return result;
}
```

5.2.3. UI

- TeacherAddForm:

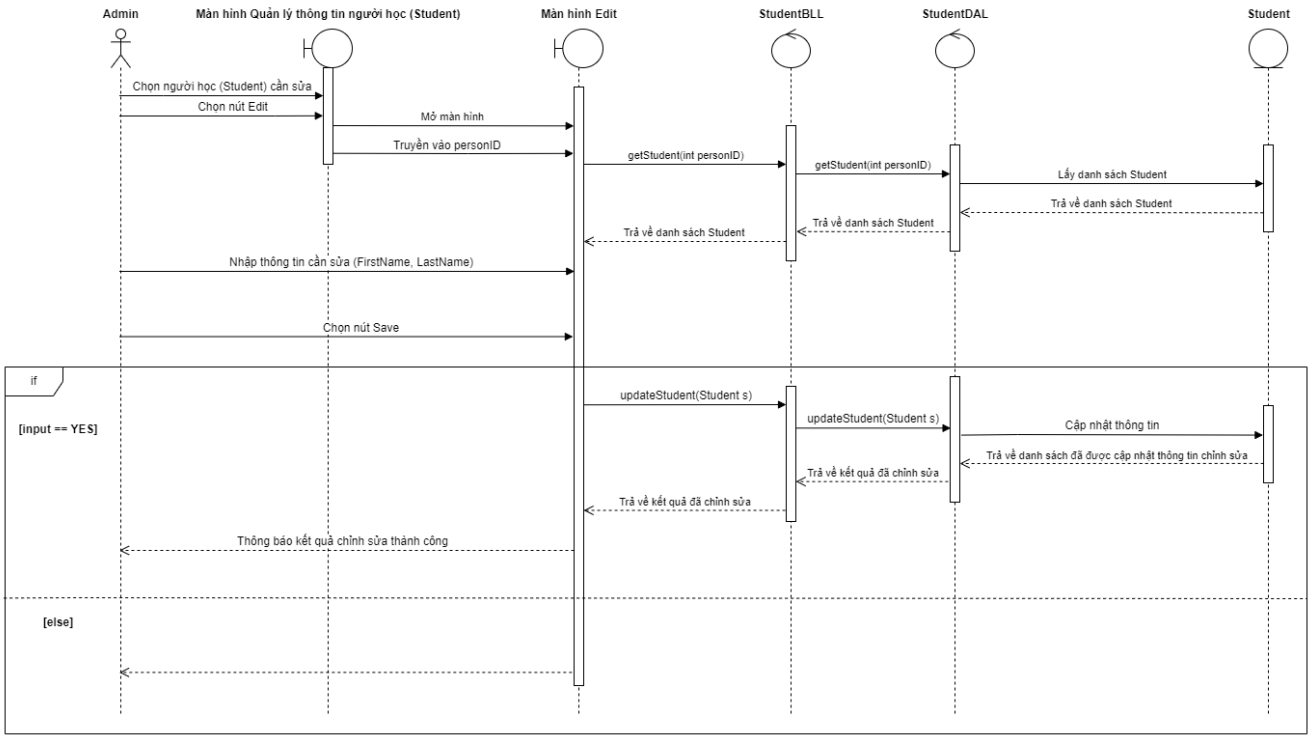
```
private void btnSaveMouseClicked(java.awt.event.MouseEvent evt) {
    Teacher t = new Teacher();
    t.setFirstName( firstName: txtFirstName.getText());
    t.setLastName( lastName: txtLastName.getText());
    Date date = Date.valueOf( s: txtHireDate.getText());
    t.setHireDate( hireDate: date);

    try {
        if (tch.addTeacher(t) > 0) {
            JOptionPane.showMessageDialog( parentComponent: this, message: "You have completed to add teacher successfully!",
                    title: "Message", messageType: JOptionPane.PLAIN_MESSAGE);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: TeacherAddForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
}
```

- TeacherForm:

```
private void btnAddMouseClicked(java.awt.event.MouseEvent evt) {
    TeacherAddForm addform = new TeacherAddForm();
    addform.setVisible( b: true);
}
```

6. Sửa / Cập nhật thông tin sinh viên

6.1. Sơ đồ tuần tự



6.2. Code 3 class

6.2.1. DAL

```
public int updateStudent(Student s) throws SQLException {
    String query = "Update Person SET FirstName = ? , LastName = ? "
            + " WHERE PersonID = ?";
    PreparedStatement p = StudentDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1,  x: s.getFirstName());
    p.setString( parameterIndex: 2,  x: s.getLastName());
    p.setInt( parameterIndex: 3,  x: s.getPersonId());
    int result = p.executeUpdate();
    return result;
}
```

6.2.2. BLL

```
public int updateStudent(Student s) throws SQLException {
    int result = stdDal.updateStudent(s);
    return result;
}
```

6.2.3. UI

- StudentEditForm:

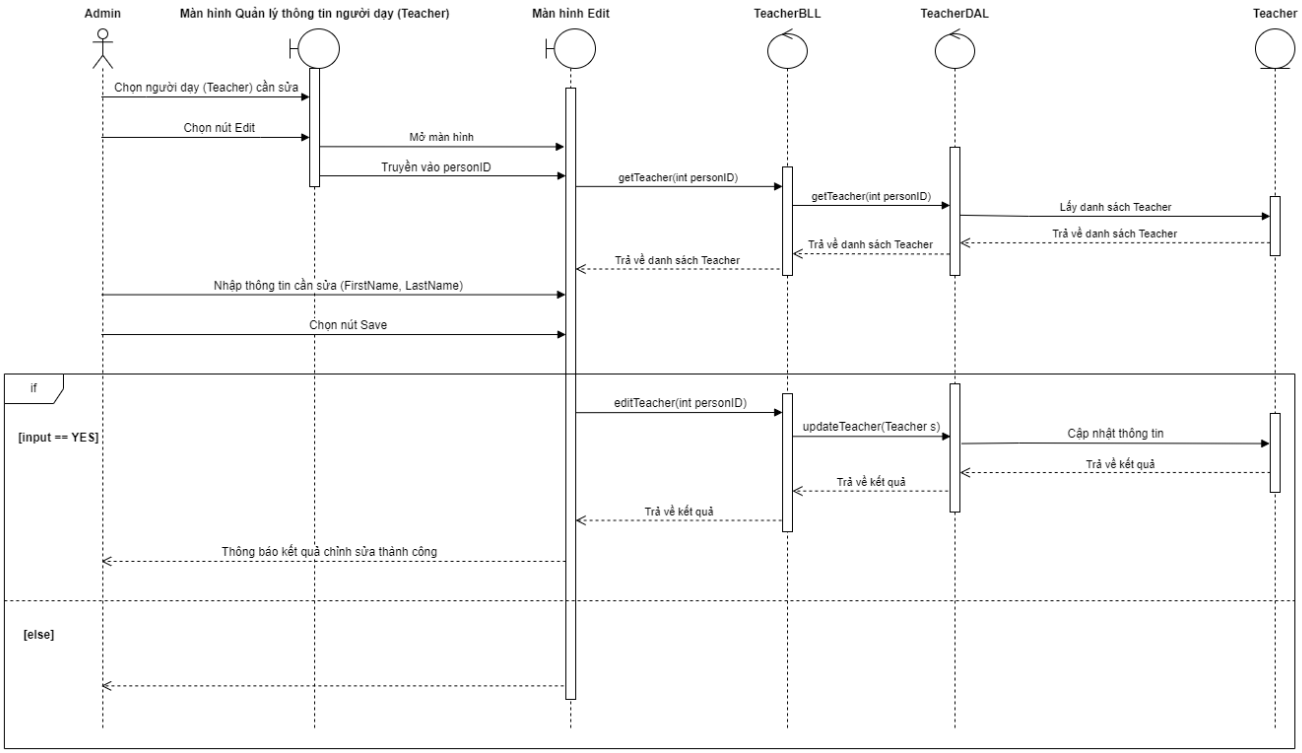
```
private void btnSaveMouseClicked(java.awt.event.MouseEvent evt) {
    s.setFirstName( firstName: txtFirstName.getText());
    s.setLastName( lastName: txtLastName.getText());
    Date date = Date.valueOf( s: txtEnrollmentDate.getText());
    s.setEnrollmentDate( enrollmentDate: date);

    try {
        int choice = JOptionPane.showConfirmDialog( parentComponent: null,  message: "Do you want to edit this Student?",
                title: "Warning!",  optionType: JOptionPane.YES_NO_OPTION);
        if (choice == JOptionPane.NO_OPTION) {
            return;
        } else {
            if (std.updateStudent(s) > 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "You have completed to edit student successfully!",
                        title: "Message",  messageType: JOptionPane.PLAIN_MESSAGE);
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: StudentEditForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

- StudentForm:

```
private void btnEditMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();

        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table !",
                    title: "WARNING!",  messageType: JOptionPane.WARNING_MESSAGE);
        } else {
            int personID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 0).toString());
            StudentEditForm f = new StudentEditForm(personID);
            f.setVisible( b: true);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: StudentForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

7. Sửa / Cập nhật thông tin người dạy

7.1. Sơ đồ tuần tự



7.2. Code 3 class

7.2.1. DAL

```
public int updateTeacher(Teacher s) throws SQLException {
    String query = "Update Person SET FirstName = ? , LastName = ? "
            + " WHERE PersonID = ?";
    PreparedStatement p = TeacherDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1,  x: s.getFirstName());
    p.setString( parameterIndex: 2,  x: s.getLastName());
    p.setInt( parameterIndex: 3,  x: s.getPersonID());
    int result = p.executeUpdate();
    return result;
}
```

7.2.2. BLL

```
public int editTeacher(Teacher t) throws SQLException {
    int edit = tchDal.updateTeacher( s: t);
    return edit;
}
```

7.2.3. UI

- TeacherEditForm:

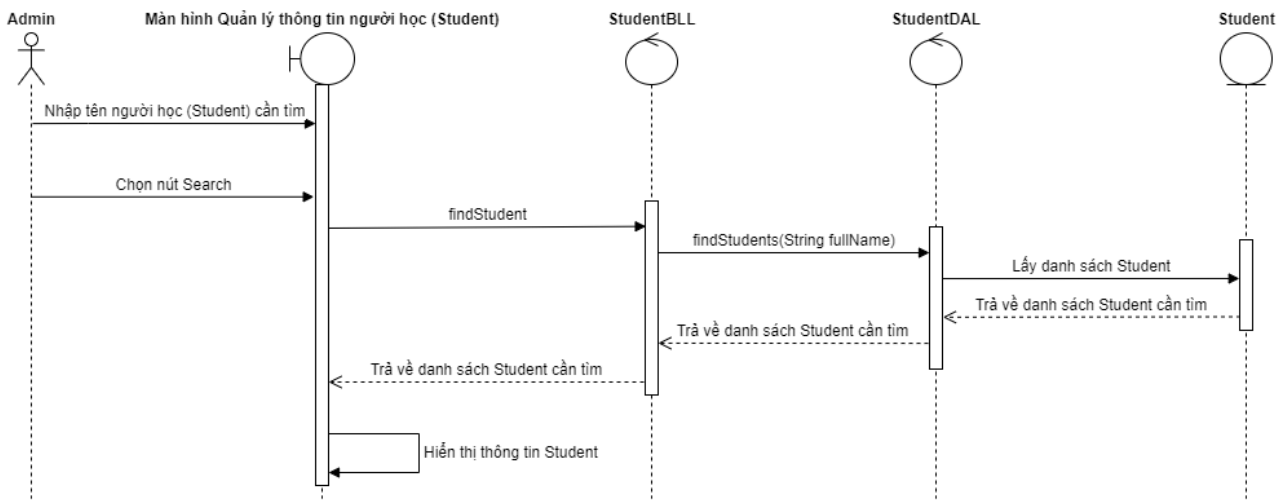
```
private void btnSaveMouseClicked(java.awt.event.MouseEvent evt) {
    t.setFirstName( firstName: txtFirstName.getText());
    t.setLastName( lastName: txtLastName.getText());
    Date date = Date.valueOf( s: txtHireDate.getText());
    t.setHireDate( hireDate: date);

    try {
        int choice = JOptionPane.showConfirmDialog( parentComponent: null,  message: "Do you want to edit this Teacher?",
                title: "Warning!",  optionType: JOptionPane.YES_NO_OPTION);
        if (choice == JOptionPane.NO_OPTION) {
            return;
        } else {
            if (tch.editTeacher(t) > 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "You have completed to edit teacher successfully!",
                        title: "Message",  messageType: JOptionPane.PLAIN_MESSAGE);
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: TeacherEditForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

- TeacherForm:

```
private void btnEditMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();
        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table!",
                    title: "WARNING!",  messageType: JOptionPane.WARNING_MESSAGE);
        } else {
            int personID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 0).toString());
            TeacherEditForm editform = new TeacherEditForm(personID);
            editform.setVisible( b: true);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: TeacherForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

8. Tìm kiếm sinh viên

8.1. Sơ đồ tuần tự



8.2. Code 3 class

8.2.1. DAL

```
public List findStudents(String fullName) throws SQLException {
    String query = "SELECT * FROM Person WHERE concat(FirstName, ' ', LastName)  LIKE ? AND EnrollmentDate IS NOT NULL";
    PreparedStatement p = StudentDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1, "%" + fullName + "%");
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {
        while (rs.next()) {
            Student s = new Student();
            s.setPersonId( personId: rs.getInt( columnLabel: "PersonID"));
            s.setFirstName( firstName: rs.getString( columnLabel: "FirstName"));
            s.setLastName( lastName: rs.getString( columnLabel: "LastName"));
            s.setEnrollmentDate( enrollmentDate: Date.valueOf(rs.getString( columnLabel: "EnrollmentDate").split( regex: " ")[0]));
            list.add( e: s);
        }
    }
    return list;
}
```

8.2.2. BLL

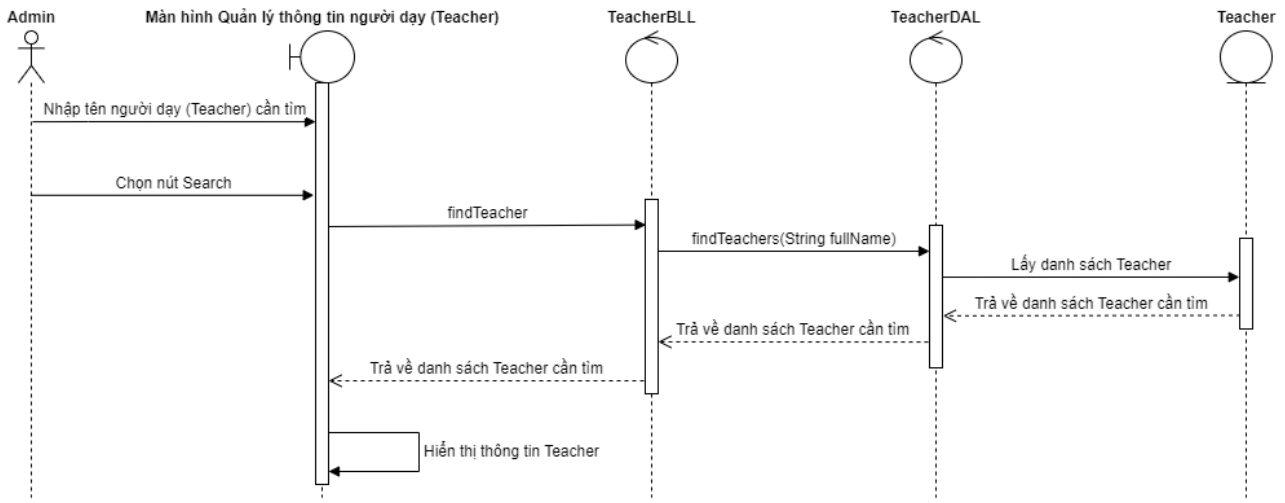
```
public List findStudent(String fullname) throws SQLException {
    List list = new ArrayList();

    list = stdDal.findStudents( fullName: fullname);

    return list;
}
```

8.2.3. UI

```
private void btnSearchMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        String fullname = txtSearch.getText();
        if (fullname.isBlank() == false) {
            List list = stb.findStudent(fullname);
            DefaultTableModel model = convertStudent(list);
            tbDS.setModel( dataModel: model);
        } else {
            List list = stb.LoadStudents();
            DefaultTableModel model = convertStudent(list);
            tbDS.setModel( dataModel: model);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: StudentForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

9. Tìm kiếm người dạy

9.1. Sơ đồ tuần tự



9.2. Code 3 class

9.2.1. DAL

```
public List findTeachers(String fullName) throws SQLException {
    String query = "SELECT * FROM Person WHERE concat(FirstName, ' ', LastName)  LIKE ? AND HireDate IS NOT NULL";
    PreparedStatement p = TeacherDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1, "%" + fullName + "%");
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {
        while (rs.next()) {
            Teacher s = new Teacher();
            s.setPersonID( personID: rs.getInt( columnLabel: "PersonID"));
            s.setFirstName( firstName: rs.getString( columnLabel: "FirstName"));
            s.setLastName( lastName: rs.getString( columnLabel: "LastName"));
            s.setHireDate( hireDate: Date.valueOf(rs.getString( columnLabel: "HireDate").split( regex: " ")[0]));
            list.add( e: s);
        }
    }
    return list;
}
```

9.2.2. BLL

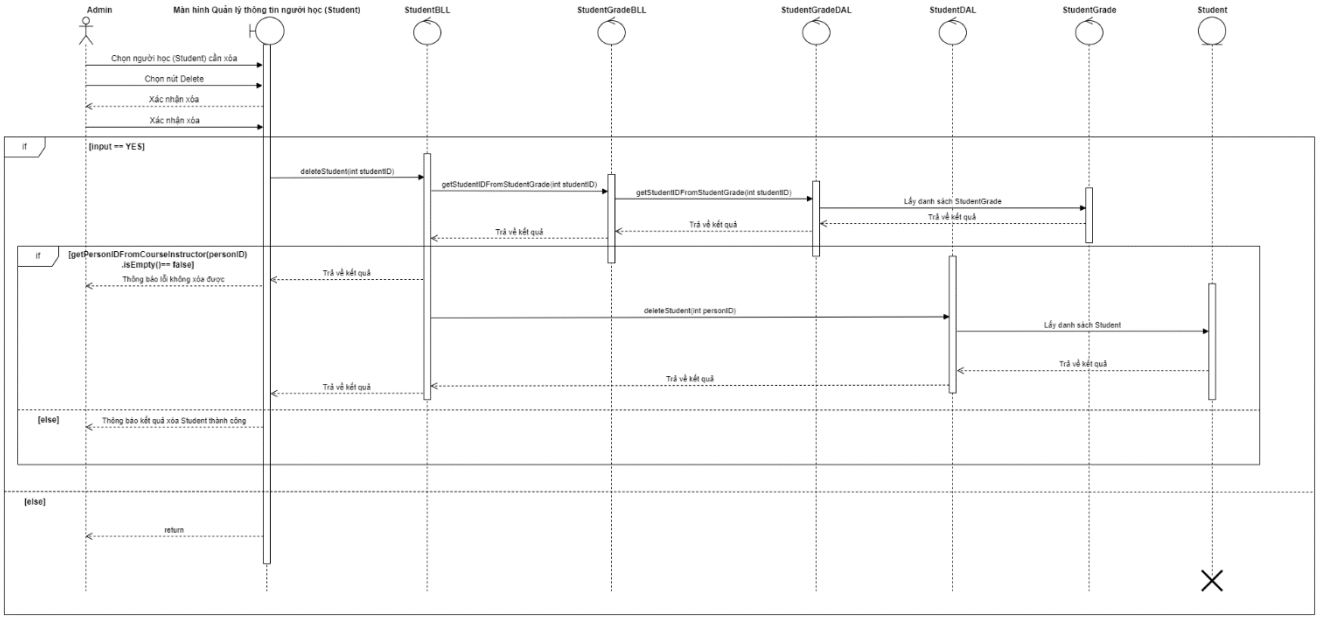
```
public List findTeacher(String fullname) throws SQLException {
    List list = new ArrayList();

    list = tchDal.findTeachers( fullName: fullname);

    return list;
}
```

9.2.3. UI

```
private void btnSearchMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        String fullname = txtSearch.getText();
        if (fullname.isBlank() == false) {
            List list = tch.findTeacher(fullname);
            DefaultTableModel model = convertTeacher(list);
            tbDS.setModel( dataModel: model);
        } else {
            List list = tch.LoadTeachers();
            DefaultTableModel model = convertTeacher(list);
            tbDS.setModel( dataModel: model);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: TeacherForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

10. Xóa record sinh viên

10.1. Sơ đồ tuần tự



10.2. Code 3 class

10.2.1. DAL

- StudentDAL

```
public int deleteStudent(int personID) throws SQLException {
    String query = "DELETE FROM Person WHERE PersonID = ?";
    PreparedStatement p = StudentDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: personID);
    int result = p.executeUpdate();
    return result;
}
```

- StudentGradeDAL

```
public List getStudentIDFromStudentGrade(int studentID) throws SQLException {

    String query = "SELECT StudentID FROM studentgrade WHERE StudentID = ? ";

    PreparedStatement p = this.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: studentID);
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {

        while (rs.next()) {
            StudentGrade s = new StudentGrade();
            s.setStudentID( StudentID: rs.getInt( columnLabel: "StudentID"));
            list.add( e: s);
        }
    }
    return list;
}
```

10.2.2. BLL

- StudentBLL

```
public int deleteStudent(int studentID) throws SQLException {
    sgb = new StudentGradeBLL();
    if (sgb.getStudentIDFromStudentGrade(studentID).isEmpty() == false) {
        return 0;
    }
    int result = stdDal.deleteStudent( personID: studentID);
    return result;
}
```

- StudentGradeBLL

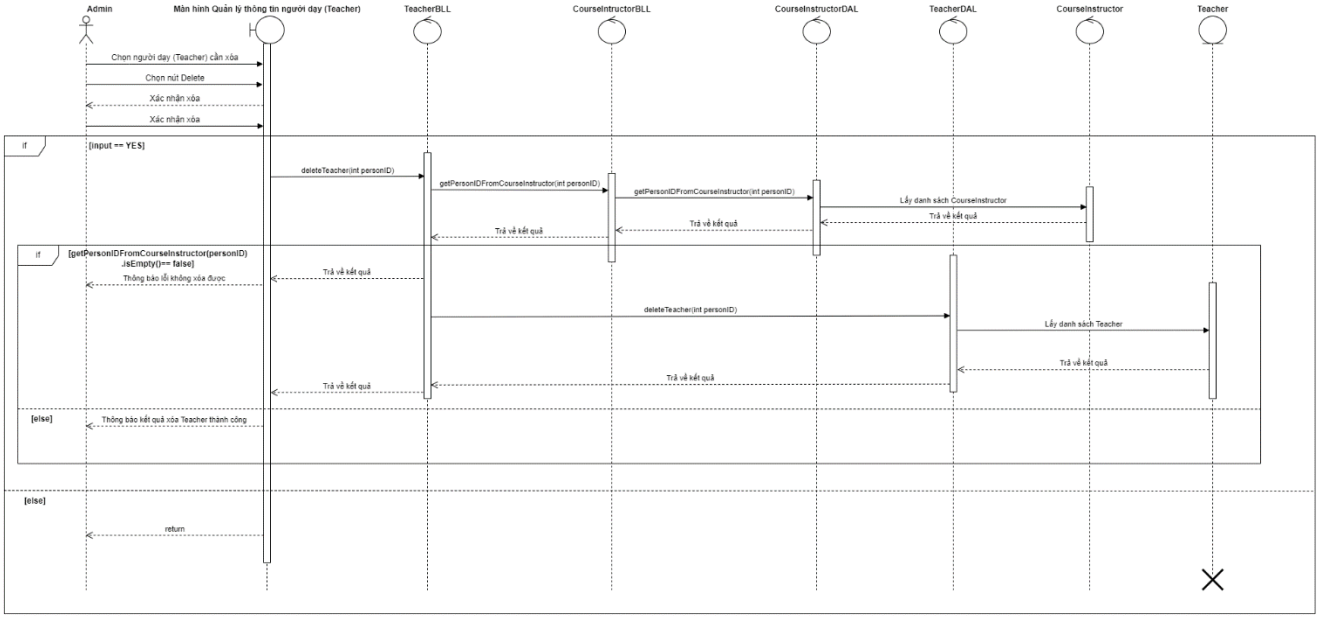
```
public List getStudentIDFromStudentGrade(int studentID) throws SQLException {
    List<StudentGrade> tempt;
    tempt = get.getStudentIDFromStudentGrade(studentID);
    return tempt;
}
```

10.2.3. UI

```
private void btnDeleteMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();
        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table!",
                    title: "WARNING!",  messageType: JOptionPane.WARNING_MESSAGE);
        } else {
            int personID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 0).toString());
            int input = JOptionPane.showConfirmDialog( parentComponent: null,
                    message: "Do you want to delete this Student?",  title: "WARNING!",  optionType: JOptionPane.YES_NO_OPTION);
            if (input == JOptionPane.YES_OPTION) {
                if (stb.deleteStudent( studentID: personID) > 0) {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "You have completed to delete student successfully!",
                            title: "Message",  messageType: JOptionPane.PLAIN_MESSAGE);
                    List list = stb.LoadStudents();
                    model = convertStudent(list);
                    tbDS.setModel( dataModel: model);
                } else {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Error because the information is binding!",
                            title: "ERROR!",  messageType: JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: StudentForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

11. Xóa record người dạy

11.1. Sơ đồ tuần tự



11.2. Code 3 class

11.2.1. DAL

- TeacherDAL

```
public int deleteTeacher(int personID) throws SQLException {
    String query = "DELETE FROM Person WHERE PersonID = ?";
    PreparedStatement p = TeacherDAL.getConnection().prepareStatement( sql:query);
    p.setInt( parameterIndex: 1,  x: personID);
    int result = p.executeUpdate();
    return result;
}
```

- CourseInstructorDAL

```
public List getPersonIDFromCourseInstructor(int personID) throws SQLException {

    String query = "SELECT PersonID FROM courseinstructor WHERE PersonID = ? ";

    PreparedStatement p = CourseInstructorDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: personID);
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {

        while (rs.next()) {
            CourseInstructor s = new CourseInstructor();
            s.setPersonID( PersonID: rs.getInt( columnLabel: "PersonID"));
            list.add( e: s);
        }
    }
    return list;
}
```

11.2.2. BLL

- TeacherBLL

```
public int deleteTeacher(int personID) throws SQLException {
    cib = new CourseIntructorBLL();
    if (cib.getPersonIDFromCourseInstructor(personID).isEmpty() == false) {
        return 0;
    }

    int del = tchDal.deleteTeacher(personID);
    return del;
}
```

- CourseInstructorBLL

```
public List getPersonIDFromCourseInstructor(int personID) throws SQLException {
    List<CourseInstructor> tempt;
    tempt = cid.getPersonIDFromCourseInstructor(personID);
    return tempt;
}
```

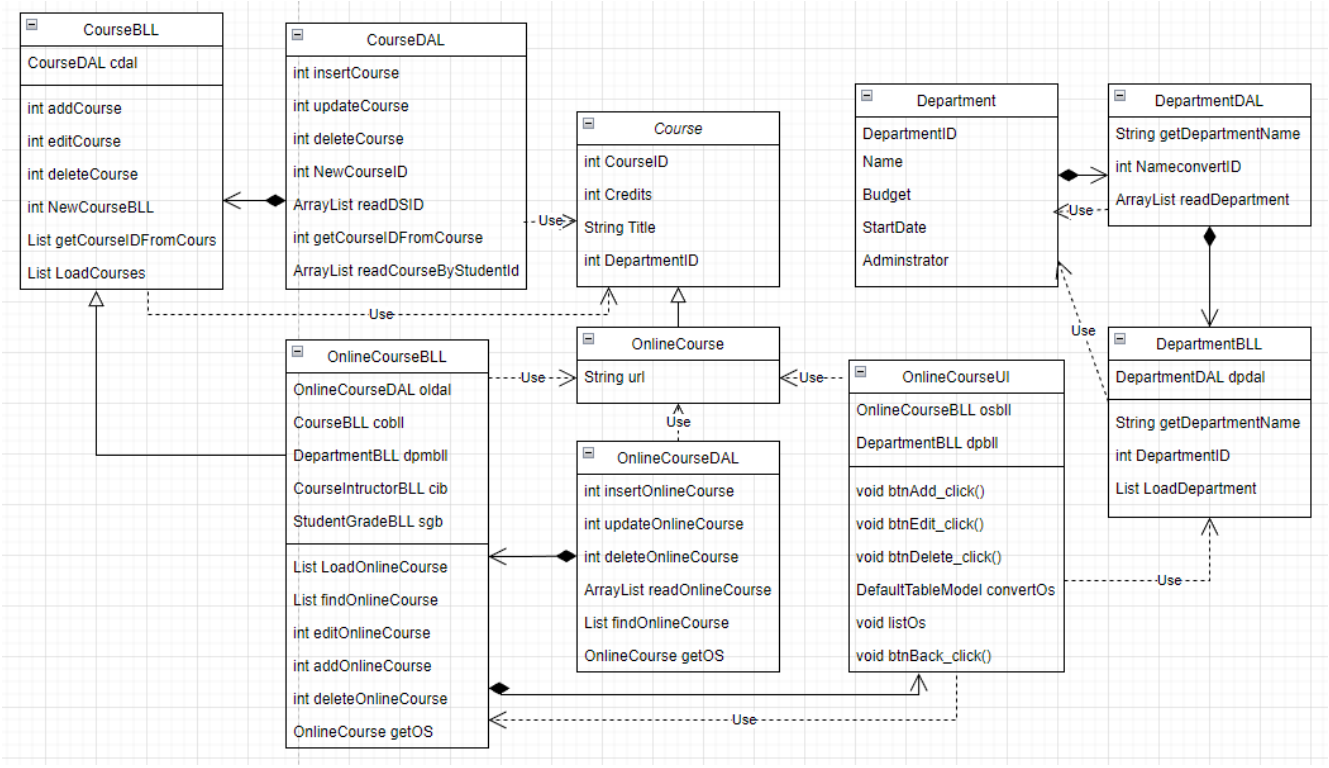
11.2.3. UI

```
private void btnDeleteMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();
        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table!",
                    title: "WARNING!",  messageType: JOptionPane.WARNING_MESSAGE);
        } else {
            int personID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 0).toString());

            int input = JOptionPane.showConfirmDialog( parentComponent: null,
                    message: "Do you want to delete this Teacher?",  title: "MESSAGE!",  optionType: JOptionPane.YES_NO_OPTION);
            if (input == JOptionPane.YES_OPTION) {
                if (tch.deleteTeacher(personID) > 0) {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "You have completed to delete teacher successfully!",
                            title: "Message",  messageType: JOptionPane.PLAIN_MESSAGE);
                    List list = tch.LoadTeachers();
                    model = convertTeacher(list);
                    tbDS.setModel( dataModel: model);
                }else{
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Error because the information is binding!",
                            title: "ERROR!",  messageType: JOptionPane.ERROR_MESSAGE);
                }
            }else{
                return;
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: TeacherForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

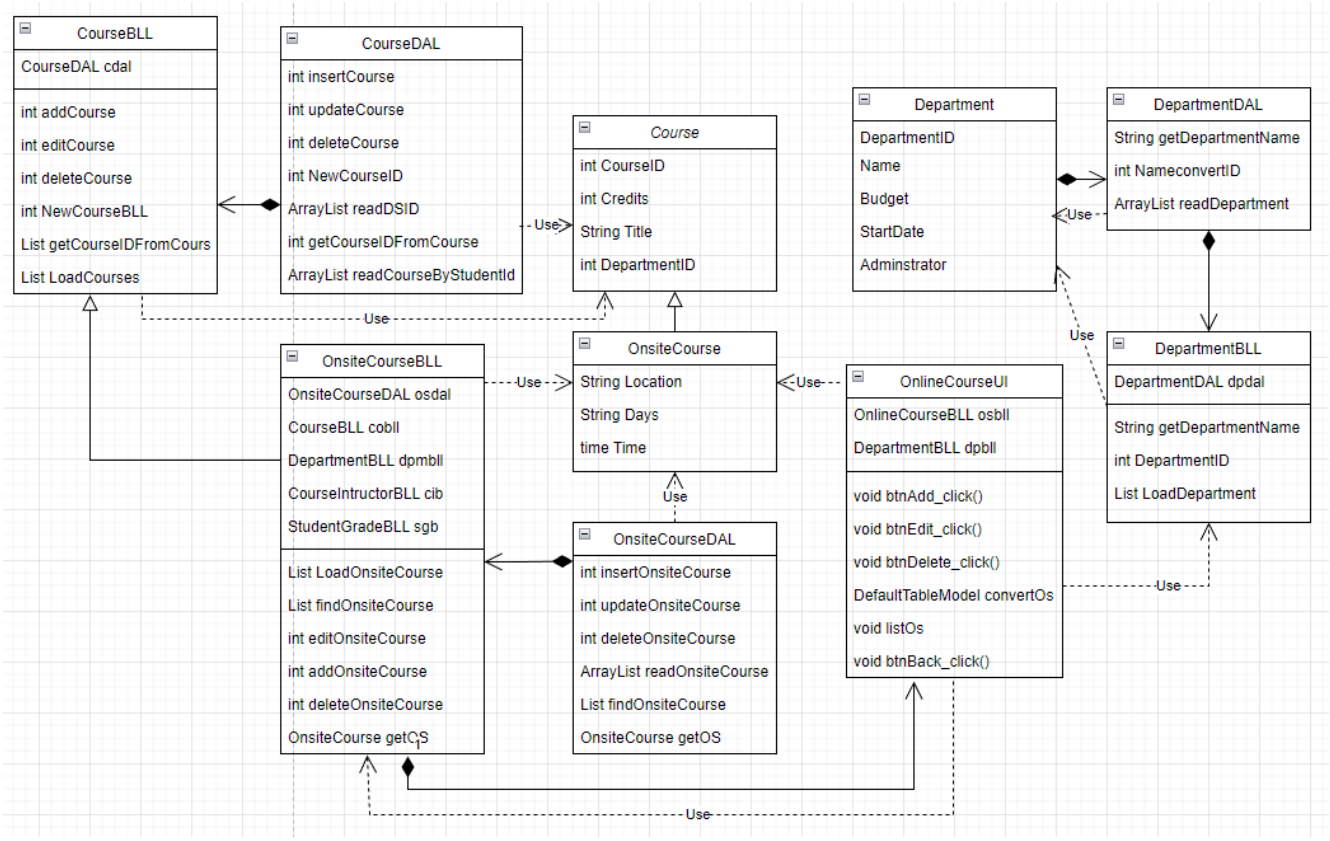
II. Chức năng 2: Quản lý thông tin các khóa học online, onsite

1. Sơ đồ class

1.1. Online

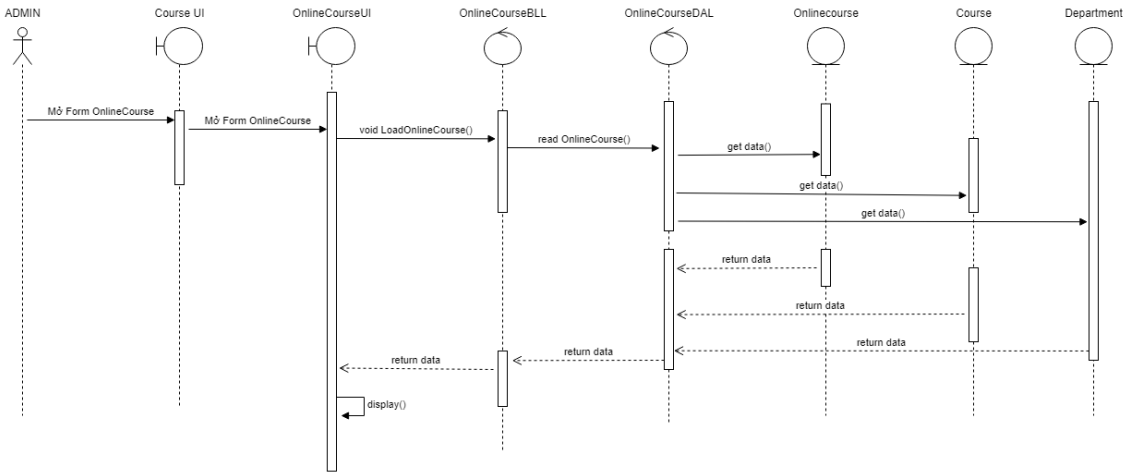


1.2. Onsite



2. Hiển thị danh sách các khóa học online

2.1. Sơ đồ tuần tự



2.2. Code 3 class:

2.2.1. DAL

```
public ArrayList readOnlineCourse() throws SQLException {
    ArrayList list = new ArrayList();
    try {
        String query = "SELECT c.CourseID, c.Title , c.Credits, c.DepartmentID, ol.url "
                + "FROM `onlinecourse` ol, `course` c WHERE c.CourseID=ol.CourseID";
        ResultSet rs = OnlineCourseDAL.doReadQuery( sql: query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                OnlineCourse s = new OnlineCourse();
                s.setCourseID( CourseID: rs.getInt( columnLabel: "CourseID"));
                s.setTitle( Title: rs.getString( columnLabel: "Title"));
                s.setCredits( Credits: rs.getInt( columnLabel: "Credits"));
                s.setDepartmentID( DepartmentID: rs.getInt( columnLabel: "DepartmentID"));
                s.setURL( URL: rs.getString( columnLabel: "url"));
                list.add( e: s);
            }

        }

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "-- ERROR! Lỗi đọc dữ liệu bảng bàn");
        ex.printStackTrace();
    } finally {
    }

    return list;
}
```

2.2.2. BLL

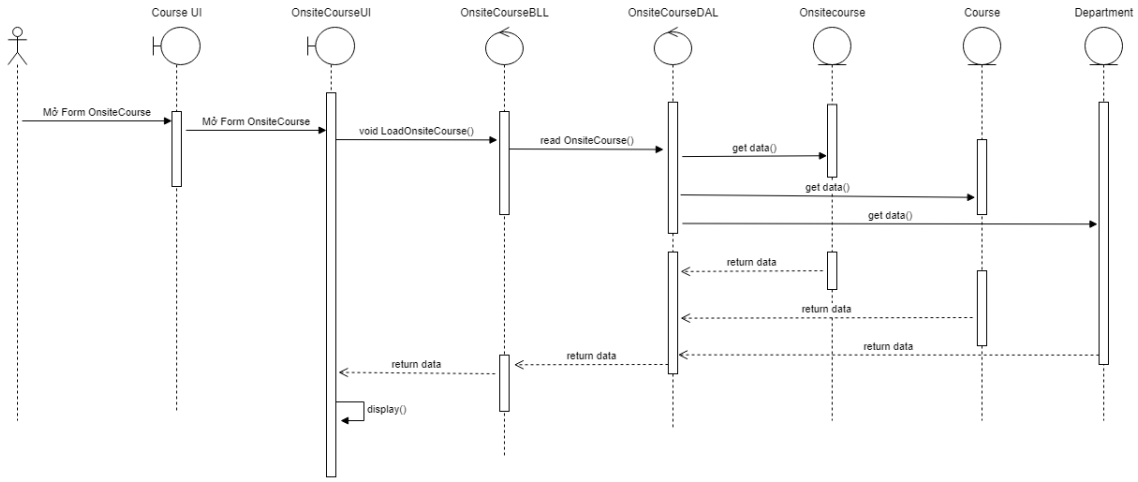
```
public List LoadOnlineCourse() throws SQLException {
    ArrayList list = oldal.readOnlineCourse();
    return list;
}
```

2.2.3. UI

```
private DefaultTableModel convertOl(List list) throws SQLException {
    String[] columnNames = {"STT", "CourseID", "Title", "Credits", "DepartmentName", "url"};
    Object[][] data = new Object[list.size()][6];
    for (int i = 0; i < list.size(); i++) {
        OnlineCourse os = (OnlineCourse) list.get( index: i);
        data[i][0] = i + 1;
        data[i][1] = os.getCourseID();
        data[i][2] = os.getTitle();
        data[i][3] = os.getCredits();
        data[i][4] = dpbll.getDepartmentName( DepartmentID: os.getDepartmentID());
        data[i][5] = os.getURL();
    }
    DefaultTableModel model = new DefaultTableModel(data, columnNames);
    return model;
}

private void listOl() throws SQLException {
    List list = osbll.LoadOnlineCourse();

    DefaultTableModel model = convertOl(list);
    tbDS.setModel( dataModel: model);

}
```

3. Hiển thị danh sách các khóa học onsite

3.1. Sơ đồ tuần tự



3.2. Code 3 class

3.3.1. DAL

```
public ArrayList readOnsiteCourse() throws SQLException {
    ArrayList list = new ArrayList();
    try {
        String query = "SELECT c.CourseID, c.Title , c.Credits, c.DepartmentID, os.Location, os.Days, os.Time "
                + "FROM `onsitecourse` os, `course` c WHERE c.CourseID=os.CourseID";
        ResultSet rs = OnsiteCourseDAL.doReadQuery( sql: query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                OnsiteCourse s = new OnsiteCourse();
                s.setCourseID( CourseID: rs.getInt( columnLabel: "c.CourseID"));
                s.setTitle( Title: rs.getString( columnLabel: "c.Title"));
                s.setCredits( Credits: rs.getInt( columnLabel: "c.Credits"));
                s.setDepartmentID( DepartmentID: rs.getInt( columnLabel: "c.DepartmentID"));
                s.setLocation( Location: rs.getString( columnLabel: "os.Location"));
                s.setDays( Days: rs.getString( columnLabel: "os.Days"));
                s.setTime( Time: rs.getTime( columnLabel: "os.Time"));
                list.add( e: s);
            }
        }

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "-- ERROR! Lỗi đọc dữ liệu bảng bàn");
    } finally {
    }

    return list;
```

3.3.2. BLL

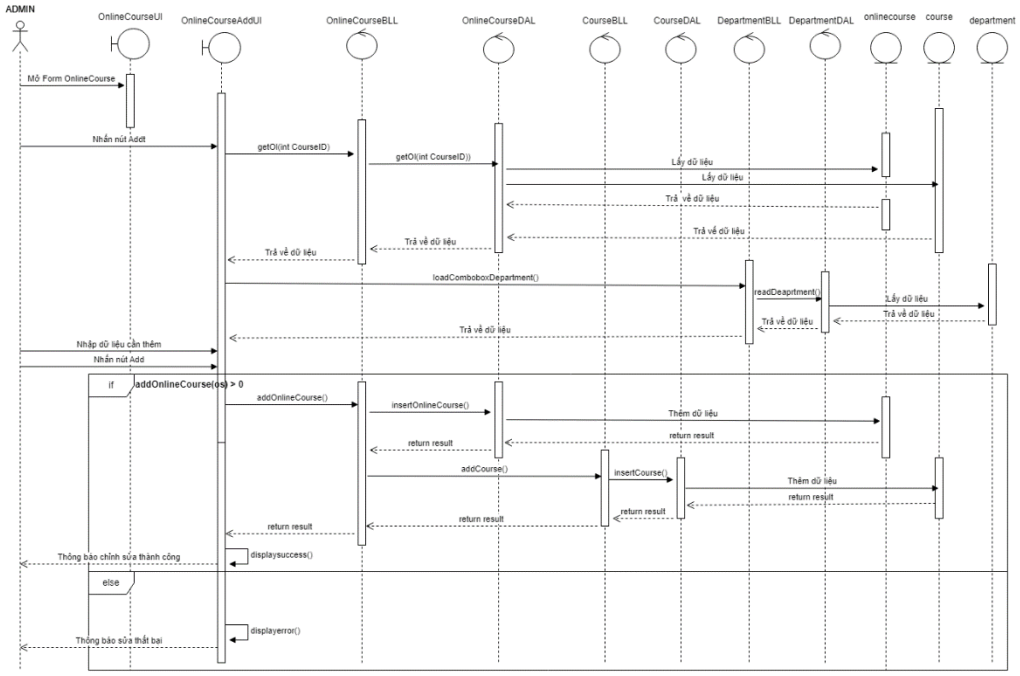
```
public List LoadOnsiteCourse() throws SQLException {
    ArrayList list = osdal.readOnsiteCourse();
    return list;
}
```

3.3.3. UI

```
private DefaultTableModel convertOs(List list) throws SQLException {
    String[] columnNames = {"STT", "CourseID", "Title", "Credits", "DepartmentID", "Location", "Days", "Time"};
    Object[][] data = new Object[list.size()][8];
    for (int i = 0; i < list.size(); i++) {
        OnsiteCourse os = (OnsiteCourse) list.get( index: i);
        data[i][0] = i + 1;
        data[i][1] = os.getCourseID();
        data[i][2] = os.getTitle();
        data[i][3] = os.getCredits();
        data[i][4] = dpbll.getDepartmentName( DepartmentID: os.getDepartmentID());
        data[i][5] = os.getLocation();
        data[i][6] = os.getDays();
        data[i][7] = os.getTime();
    }
    DefaultTableModel model = new DefaultTableModel(data, columnNames);
    return model;
}

private void listOs() throws SQLException {
    List list = osbll.LoadOnsiteCourse();

    DefaultTableModel model = convertOs(list);
    tbDS.setModel( dataModel: model);

}
```

4. Thêm mới thông tin các khóa học online

4.1. Sơ đồ tuần tự



addOnlineCourse(os) > 0

4.2. Code 3 class

4.2.1. DAL

- Course

```
public int insertCourse(Course s) throws SQLException {
    String query = "Insert course (Credits, DepartmentID, Title) VALUES (?, ?, ?)";
    PreparedStatement p = CourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: s.getCredits());
    p.setInt( parameterIndex: 2,  x: s.getDepartmentID());
    p.setString( parameterIndex: 3,  x: s.getTitle());
    int result = p.executeUpdate();
    return result;
}
```

- **OnlineCourse**

```
public int insertOnlineCourse(OnlineCourse s) throws SQLException {
    String query = "Insert onlinecourse (CourseID, url) VALUES (?, ?)";
    PreparedStatement p = OnlineCourseDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 2,  x: s.getURL().toString());
    p.setInt( parameterIndex: 1,  x: s.getCourseID());
    int result = p.executeUpdate();
    return result;
}
```

4.2.2. BLL

- **Course**

```
public int addCourse(Course s) throws SQLException {
    int result = cdal.insertCourse(s);
    return result;
}
```

- **OnlineCourse**

```
public int addOnlineCourse(OnlineCourse s) throws SQLException {
    this.addCourse(s);
    int id = this.NewCourseBLL();
    s.setCourseID( CourseID: id);
    int result = oldal.insertOnlineCourse(s);
    return result;
}
```

4.2.3. UI

- **OnlineCourseForm**

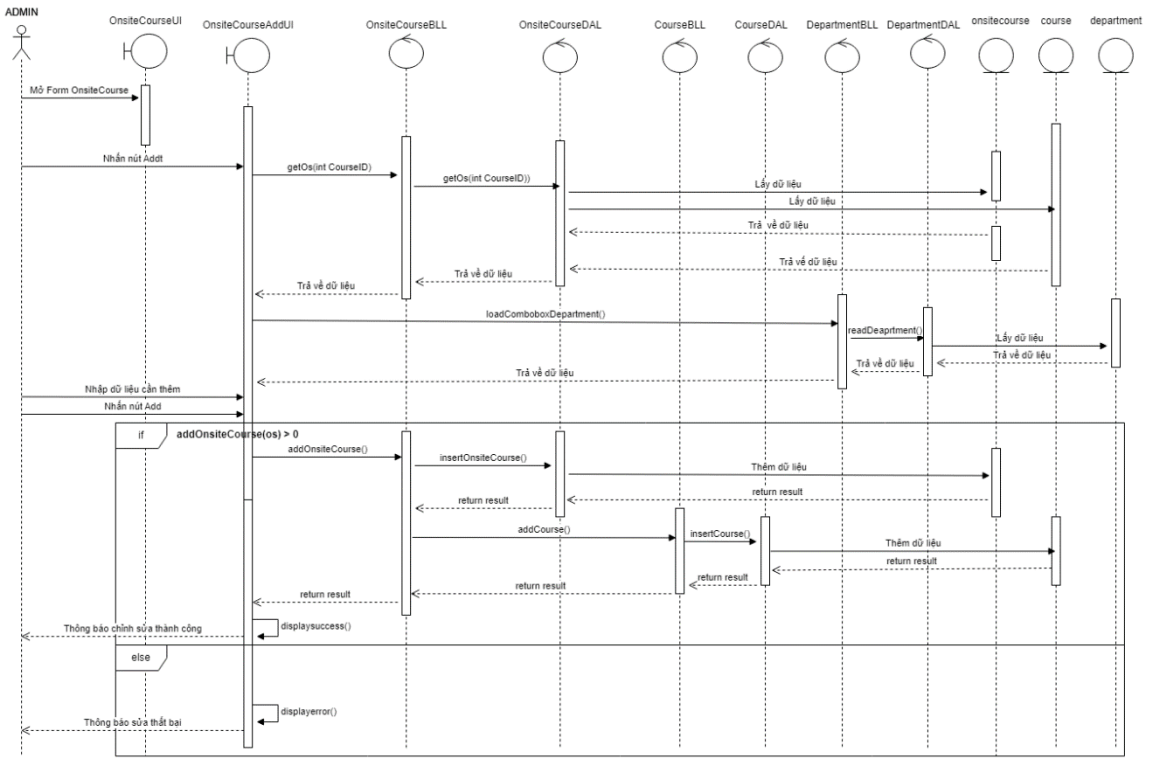
```
private void btnAddMouseClicked(java.awt.event.MouseEvent evt) {
    OnlineCourseAddForm add = new OnlineCourseAddForm();
    add.setVisible( b: true);
}
```

- **OnlineCourseAddForm**

```
private void btnAddMouseClicked(java.awt.event.MouseEvent evt) {
    OnlineCourse os = new OnlineCourse();
    os.setCredits( Credits: Integer.parseInt( s: textField1.getText()));
    int dpid = DepartmentIdArr[DpComboBox.getSelectedIndex()];
    os.setDepartmentID( DepartmentID: dpid);
    os.setTitle( Title: textField2.getText());
    os.setURL( URL: textField4.getText());
    try {
        if (osbll.addOnlineCourse( s: os) > 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Complete add OnlineCourse",  title: "Message",  messageType: JOptionPane.INFORMA
        } else {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Error add OnlineCourse",  title: "Message",  messageType: JOptionPane.ERROR_MESS
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnlineCourseAddForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
    this.setVisible( b: true);
    this.dispose();

}
```

5. Thêm mới thông tin các khóa học onsite

5.1. Sơ đồ tuần tự



5.2. Code 3 class

5.2.1. DAL

- Course

```
public int insertCourse(Course s) throws SQLException {
    String query = "Insert course (Credits, DepartmentID, Title) VALUES (?, ?, ?)";
    PreparedStatement p = CourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: s.getCredits());
    p.setInt( parameterIndex: 2,  x: s.getDepartmentID());
    p.setString( parameterIndex: 3,  x: s.getTitle());
    int result = p.executeUpdate();
    return result;
}
```

- OnsiteCourse

```
public int insertOnsiteCourse(OnsiteCourse s) throws SQLException {
    String query = "Insert onsitecourse (CourseID, Location, Days, Time) VALUES (?, ?, ?, ?)";
    PreparedStatement p = OnsiteCourseDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 2,  x: s.getLocation());
    p.setString( parameterIndex: 3,  x: s.getDays());
    p.setString( parameterIndex: 4,  x: s.getTime().toString());
    p.setInt( parameterIndex: 1,  x: s.getCourseID());
    int result = p.executeUpdate();
    return result;
}
```

5.2.2. BLL

- Course

```
public int addCourse(Course s) throws SQLException {
    int result = cdal.insertCourse(s);
    return result;
}
```

- OnsiteCourse

```
public int addOnsiteCourse(OnsiteCourse s) throws SQLException {
    this.addCourse(s);
    int id = this.NewCourseBLL();
    s.setCourseID( CourseID: id);
    int result = osdal.insertOnsiteCourse(s);
    return result;
}
```

5.2.3. UI

- OnsiteCourseForm

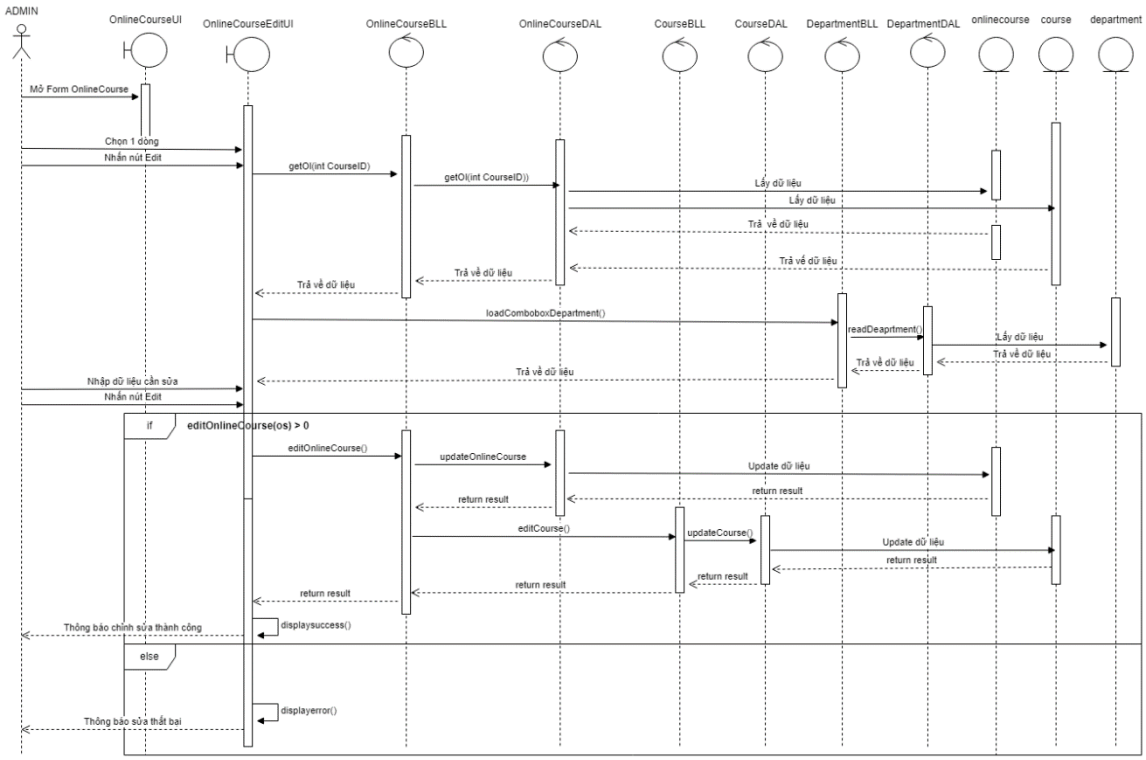
```
private void btnAddMouseClicked(java.awt.event.MouseEvent evt) {
    OnsiteCourseAddForm add = new OnsiteCourseAddForm();
    add.setVisible( b: true);
}
```

- OnsiteCourseAddForm

```
private void btnAddMouseClicked(java.awt.event.MouseEvent evt) {
    OnsiteCourse os = new OnsiteCourse();
    os.setCredits( Credits: Integer.parseInt( s: textField1.getText()));
    String name = DpComboBox.getSelectedItem().toString();
    int dpid = 0;
    try {
        dpid = dbll.DepartmentID(name);
    } catch (SQLException ex) {
        Logger.getLogger( name: OnsiteCourseAddForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
    os.setDepartmentID( DepartmentID: dpid);
    os.setTitle( Title: textField2.getText());
    os.setLocation( Location: textField4.getText());
    os.setDays( Days: textField5.getText());
    Time time = Time.valueOf( s: textField6.getText());
    os.setTime( Time: time);
    try {
        if (osbll.addOnsiteCourse( s: os) > 0) {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Complete add OnsiteCourse", title: "Message", messageType: JOptionPane.INFORMA
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Error add OnsiteCourse", title: "Message", messageType: JOptionPane.ERROR_MESS.
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnsiteCourseAddForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
    this.setVisible( b: true);
```

6. Sửa / Cập nhật thông tin các khóa học online

6.1. Sơ đồ tuần tự



6.2. Code 3 class

6.2.1. DAL

- Course

```
public int updateCourse(Course s) throws SQLException {
    String query = "Update course SET Credits = ? , DepartmentID = ? , Title = ? "
            + " WHERE CourseID = ?";
    PreparedStatement p = CourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: s.getCredits());
    p.setInt( parameterIndex: 2,  x: s.getDepartmentID());
    p.setString( parameterIndex: 3,  x: s.getTitle());
    p.setInt( parameterIndex: 4,  x: s.getCourseID());
    int result = p.executeUpdate();
    return result;
}
```

- OnlineCourse

```
public int updateOnlineCourse(OnlineCourse s) throws SQLException {
    String query = "Update onlinecourse SET url = ? "
            + " WHERE CourseID = ?";
    PreparedStatement p = OnlineCourseDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1,  x: s.getURL());
    p.setInt( parameterIndex: 2,  x: s.getCourseID());
    int result = p.executeUpdate();
    return result;
}
```

6.2.2. BLL

- Course

```
public int editCourse(Course s) throws SQLException {
    int result = cdal.updateCourse(s);
    return result;
}
```

- OnlineCourse

```
public int editOnlineCourse(OnlineCourse s) throws SQLException {
    this.editCourse(s);
    int result = oldal.updateOnlineCourse(s);
    return result;
}
```

6.2.3. UI

- OnlineCourseForm

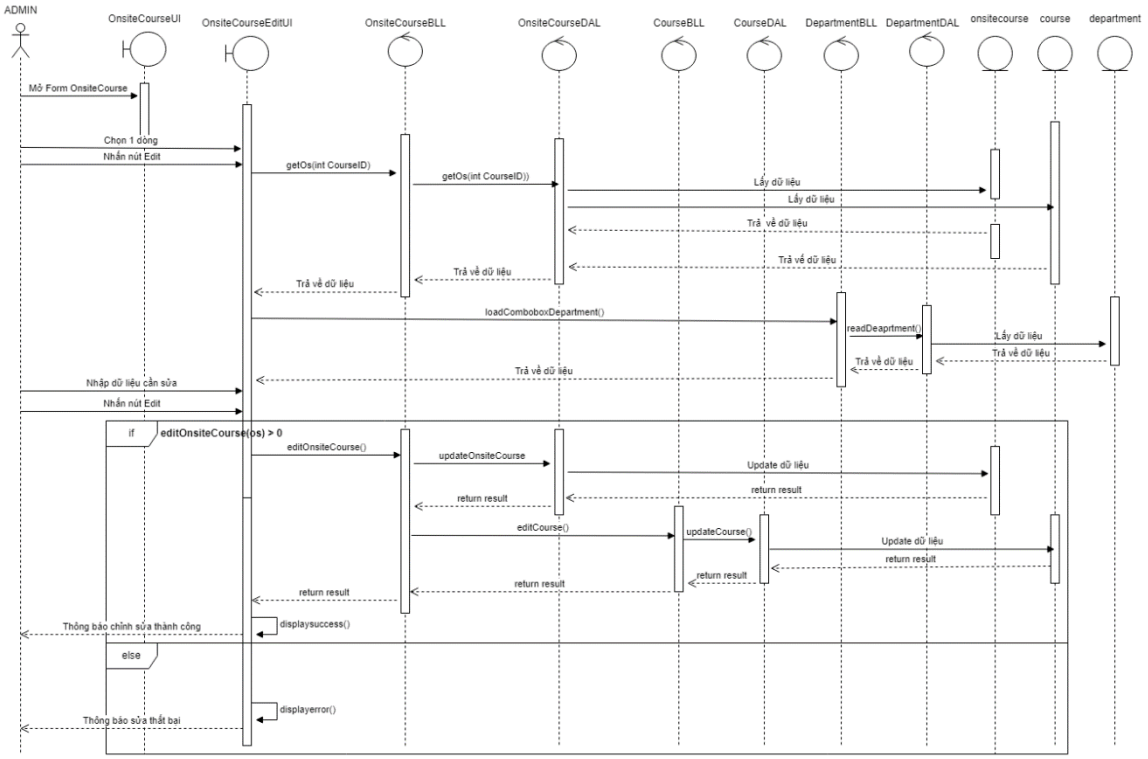
```
    private void btnEditMouseClicked(java.awt.event.MouseEvent evt) {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();
        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table !",  title: "Message",  messageType: JOptionPane.ERRO
        } else {
            int CourseID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 1).toString());
            OnlineCourseEditForm f;
            try {
                f = new OnlineCourseEditForm(CourseID);
                f.setVisible( b: true);
            } catch (SQLException ex) {
                Logger.getLogger( name: OnlineCourseForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
            }

        }
    }
```

- OnlineCourseEditForm

```
private void btnEdit2MouseClicked(java.awt.event.MouseEvent evt) {

    OnlineCourse os = new OnlineCourse();
    os.setCredits( Credits: Integer.parseInt( s: textField1.getText()));
    int dpid = DepartmentIdArr[DpComboBox.getSelectedIndex()];
    os.setDepartmentID( DepartmentID: dpid);
    os.setTitle( Title: textField2.getText());
    os.setURL( URL: textField4.getText());
    os.setCourseID( CourseID: this.CourseID);
    try {
        if (osbll.editOnlineCourse( s: os) > 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Complete edit OnlineCourse",  title: "Message",  messageType: JOptionPane.INFORM
        } else {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Error edit OnlineCourse",  title: "Message",  messageType: JOptionPane.ERROR_MES
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnlineCourseEditForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
    this.setVisible( b: true);
    this.dispose();
}
```

7. Sửa / Cập nhật thông tin các khóa học onsite

7.1. Sơ đồ tuần tự



7.2. Code 3 class

7.2.1. DAL

- Course

```
public int updateCourse(Course s) throws SQLException {
    String query = "Update course SET Credits = ? , DepartmentID = ? , Title = ? "
            + " WHERE CourseID = ?";
    PreparedStatement p = CourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: s.getCredits());
    p.setInt( parameterIndex: 2,  x: s.getDepartmentID());
    p.setString( parameterIndex: 3,  x: s.getTitle());
    p.setInt( parameterIndex: 4,  x: s.getCourseID());
    int result = p.executeUpdate();
    return result;
}
```

- OnsiteCourse

```
public int updateOnsiteCourse(OnsiteCourse s) throws SQLException {
    String query = "Update onsitecourse SET Location = ? , Days = ? , Time = ? "
            + " WHERE CourseID = ?";
    PreparedStatement p = OnsiteCourseDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1,  x: s.getLocation());
    p.setString( parameterIndex: 2,  x: s.getDays());
    p.setTime( parameterIndex: 3,  x: s.getTime());
    p.setInt( parameterIndex: 4,  x: s.getCourseID());
    int result = p.executeUpdate();
    return result;
}
```

7.2.2. BLL

- Course

```
public int editCourse(Course s) throws SQLException {
    int result = cdal.updateCourse(s);
    return result;
}
```

- OnsiteCourse

```
public int editOnsiteCourse(OnsiteCourse s) throws SQLException {
    this.editCourse(s);
    int result = osdal.updateOnsiteCourse(s);
    return result;
}
```

7.2.3. UI

- OnsiteCourseForm

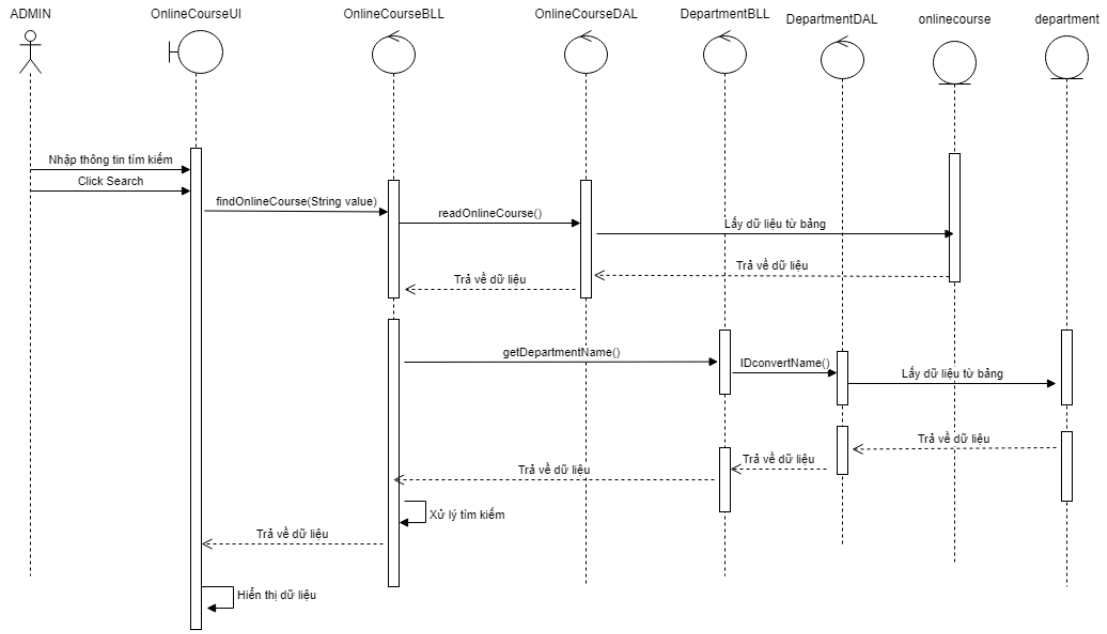
```
private void btnEditMouseClicked(java.awt.event.MouseEvent evt) {
    int row = tbDS.getSelectedRow();
    TableModel model = tbDS.getModel();
    if (row < 0) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table !",  title: "Message",  messageType: JOptionPane.ERR
    } else {
        int CourseID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 1).toString());
        OnsiteCourseEditForm f;
        try {
            f = new OnsiteCourseEditForm(CourseID);
            f.setVisible( b: true);
        } catch (SQLException ex) {
            Logger.getLogger( name: OnsiteCourseForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
        }

    }
}
```

- OnsiteCourseEditForm

```
private void btnEdit2MouseClicked(java.awt.event.MouseEvent evt) {
    OnsiteCourse os = new OnsiteCourse();
    os.setCredits( Credits: Integer.parseInt( s: textField1.getText()));
    String name = DpComboBox.getSelectedItem().toString();
    int dpid = 0;
    try {
        dpid = dbll.DepartmentID(name);
    } catch (SQLException ex) {
        Logger.getLogger( name: OnsiteCourseEditForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
    os.setDepartmentID( DepartmentID: dpid);
    os.setTitle( Title: textField2.getText());
    os.setLocation( Location: textField4.getText());
    os.setDays( Days: textField5.getText());
    Time time = Time.valueOf( s: textField6.getText());
    os.setTime( Time: time);
    os.setCourseID( CourseID: this.CourseID);
    try {
        if (osbll.editOnsiteCourse( s: os) > 0) {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Complete edit OnsiteCourse", title: "Message", messageType: JOptionPane.INFORM
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Error edit OnsiteCourse", title: "Message", messageType: JOptionPane.ERROR_MES
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnsiteCourseEditForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
    this.setVisible( b: true);
    this.dispose();
}
```

8. Tìm kiếm các khóa học online

8.1. Sơ đồ tuần tự



8.2. Code 3 class

8.2.1. DAL

```
public List findOnlineCourse(String value) throws SQLException {
    String query = "SELECT * FROM `onlinecourse` os, `course` c, `department` dp "
            + "WHERE os.CourseID=c.CourseID AND c.DepartmentID=dp.DepartmentID "
            + "AND c.Title LIKE ?";
    PreparedStatement p = OnlineCourseDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1, "%" + value + "%");
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {
        int i = 1;

        while (rs.next()) {
            OnlineCourse os = new OnlineCourse();
            os.setURL( URL: rs.getString( columnLabel: "os.url"));
            os.setCourseID( CourseID: rs.getInt( columnLabel: "c.CourseID"));
            os.setCredits( Credits: rs.getInt( columnLabel: "c.Credits"));
            os.setDepartmentID( DepartmentID: rs.getInt( columnLabel: "c.DepartmentID"));
            os.setTitle( Title: rs.getString( columnLabel: "c.Title"));
            list.add( e: os);
        }
    }
    return list;
}
```

8.2.2. BLL

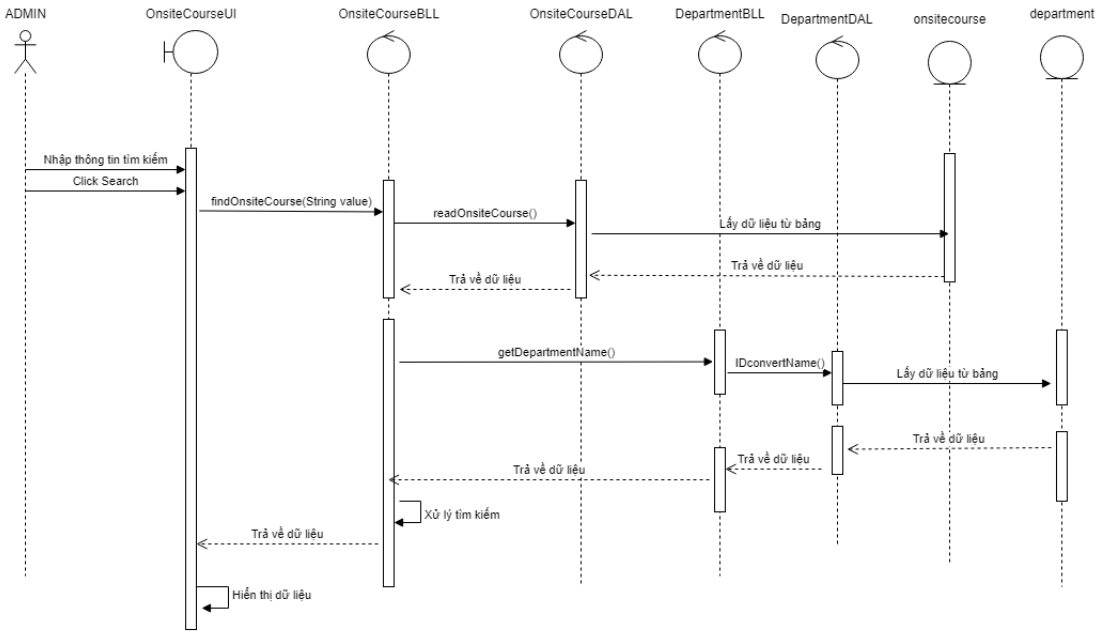
```
public List findOnlineCourse(String condititon) throws SQLException {

    ArrayList<OnlineCourse> onlclist = oldal.readOnlineCourse();
    ArrayList<OnlineCourse> onlcsearch = new ArrayList<OnlineCourse>();
    condititon = condititon.trim().replaceAll( regex: "  +", replacement: " ").toLowerCase();
    String oldCondition = condititon;
    String[] conditions = condititon.split( regex: " ");

    for (int i = 0; i < onlclist.size(); i++) {
        String regex = onlclist.get( index: i).getTitle() + " " + dpmbll.DepartmentName( id: onlclist.get( index: i).getDepartmentID());
        for (int j = 0; j < conditions.length; j++) {
            String oldChirlCondition = conditions[j];
            conditions[j] = "(.*)" + conditions[j] + "(.*)";
            if (regex.toLowerCase().matches(conditions[j])) {
                onlcsearch.add( e: onlclist.get( index: i));
                break;
            }
            conditions[j] = oldChirlCondition;
        }

    }
    if (onlcsearch.size() == 0) {
        return onlclist;
    }
    return onlcsearch;

}
```

8.2.3. UI

```
private void btnSaerchMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        String condition = txtSearch.getText();
        if (condition.isBlank() == false) {
            List list = osbll.findOnlineCourse( condititon: condition);
            DefaultTableModel model = convertOl(list);
            tbDS.setModel( dataModel: model);

        } else {
            //JOptionPane.showMessageDialog(this, "fullname is empty", "Message", JOptionPane.ERROR_MESSAGE);

            List list = osbll.LoadOnlineCourse();
            DefaultTableModel model = convertOl(list);
            tbDS.setModel( dataModel: model);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnlineCourseForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

9. Tìm kiếm các khóa học onsite

9.1. Sơ đồ tuần tự



9.2. Code 3 class

9.2.1. DAL

```
public List findOnsiteCourse(String value) throws SQLException {
    String query = "SELECT * FROM `onsitecourse` os, `course` c, `department` dp "
            + "WHERE os.CourseID=c.CourseID AND c.DepartmentID=dp.DepartmentID "
            + "AND c.Title LIKE ?";
    PreparedStatement p = OnsiteCourseDAL.getConnection().prepareStatement( sql: query);
    p.setString( parameterIndex: 1, "%" + value + "%");
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {
        int i = 1;

        while (rs.next()) {
            OnsiteCourse os = new OnsiteCourse();
            os.setLocation( Location: rs.getString( columnLabel: "os.Location"));
            os.setDays( Days: rs.getString( columnLabel: "os.Days"));
            os.setTime( Time: rs.getTime( columnLabel: "os.Time"));
            os.setCourseID( CourseID: rs.getInt( columnLabel: "c.CourseID"));
            os.setCredits( Credits: rs.getInt( columnLabel: "c.Credits"));
            os.setDepartmentID( DepartmentID: rs.getInt( columnLabel: "c.DepartmentID"));
            os.setTitle( Title: rs.getString( columnLabel: "c.Title"));
            list.add( e: os);
        }
    }
    return list;
}
```

9.2.2. BLL

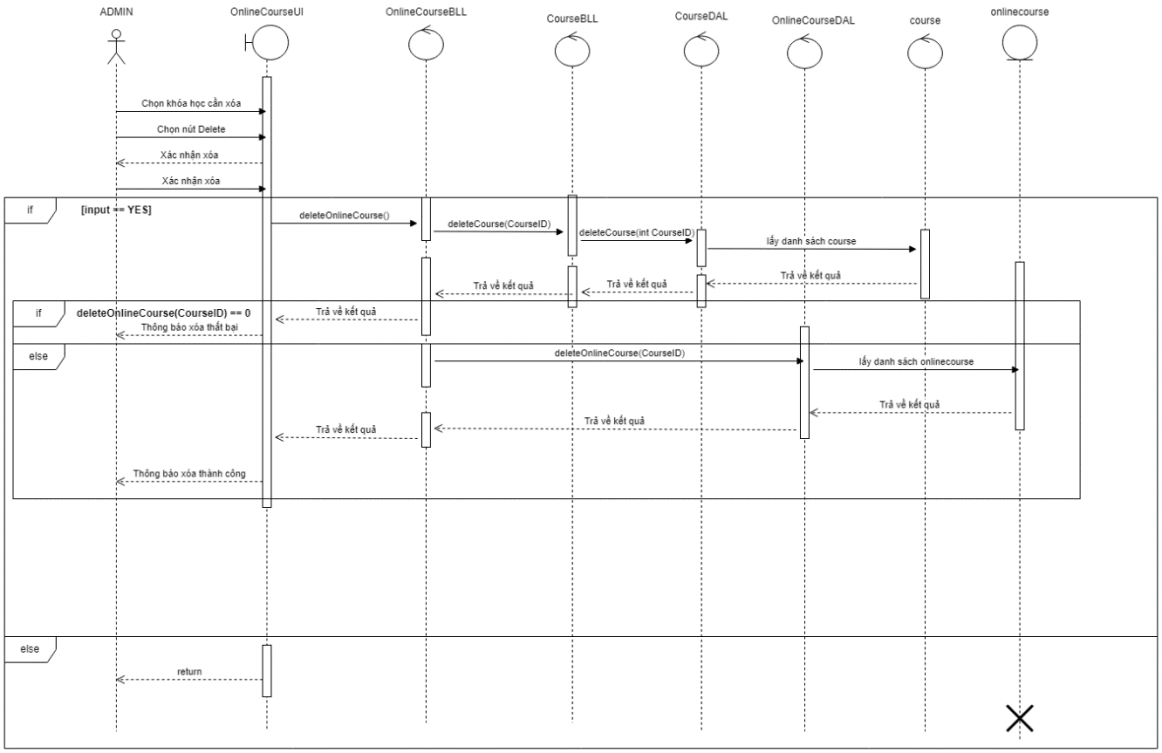
```
public List findOnsiteCourse(String condititon) throws SQLException {

    ArrayList<OnsiteCourse> onlclist = osdal.readOnsiteCourse();
    ArrayList<OnsiteCourse> onlcsearch = new ArrayList<OnsiteCourse>();
    condititon = condititon.trim().replaceAll( regex: "  +", replacement: " ").toLowerCase();
    String oldCondition = condititon;
    String[] conditions = condititon.split( regex: " ");

    for (int i = 0; i < onlclist.size(); i++) {
        String regex = onlclist.get( index: i).getTitle() + " " + dbll.DepartmentName( id: onlclist.get( index: i).getDepartmentID());
        for (int j = 0; j < conditions.length; j++) {
            String oldChirlCondition = conditions[j];
            conditions[j] = "(.*)" + conditions[j] + "(.*)";
            if (regex.toLowerCase().matches(conditions[j])) {
                onlcsearch.add( e: onlclist.get( index: i));
                break;
            }
            conditions[j] = oldChirlCondition;
        }

    }
    if (onlcsearch.size() == 0) {
        return onlclist;
    }
    return onlcsearch;

}
```

9.2.3. UI

```
private void btnSaerchMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        String Title = txtSearch.getText();
        if (Title.isBlank() == false) {
            List list = osbll.findOnsiteCourse( condititon: Title);
            DefaultTableModel model = convertOs(list);
            tbDS.setModel( dataModel: model);

        } else {
            //JOptionPane.showMessageDialog(this, "fullname is empty", "Message", JOptionPane.ERROR_MESSAGE);
            List list = osbll.LoadOnsiteCourse();
            DefaultTableModel model = convertOs(list);
            tbDS.setModel( dataModel: model);
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnsiteCourseForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

10. Xóa record các khóa học online

10.1. Sơ đồ tuần tự



10.2. Code 3 class

10.2.1. DAL

- Course

```
public int deleteCourse(int CourseID) throws SQLException {
    String query = "DELETE FROM course WHERE CourseID = ?";
    PreparedStatement p = CourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: CourseID);
    int result = p.executeUpdate();
    return result;
}
```

- OnlineCourse

```
public int deleteOnlineCourse(int CourseID) throws SQLException {
    String query = "DELETE FROM onlinecourse WHERE CourseID = ?";
    PreparedStatement p = OnlineCourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: CourseID);
    int result = p.executeUpdate();
    return result;
}
```

10.2.2. BLL

- Course

```
public int deleteCourse(int CourseID) throws SQLException {
    int result = cdal.deleteCourse(CourseID);
    return result;
}
```

- OnlineCourse

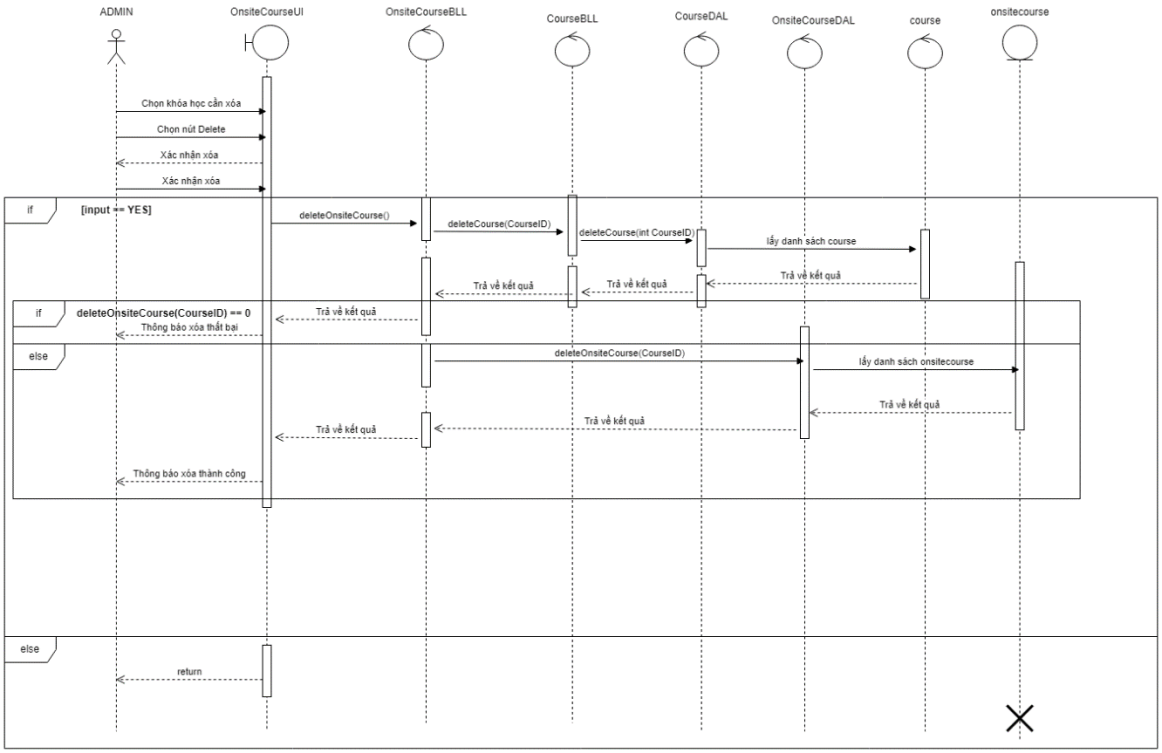
```
public int deleteOnlineCourse(int CourseID) throws SQLException {
    int result;
    cib = new CourseIntructorBLL();
    sgb = new StudentGradeBLL();
    cobll = new CourseBLL();
    if ((cib.getCourseIDFromCourseInstructor( courseID: CourseID).isEmpty() == false && cobll.getCourseIDFromCourse( courseID: CourseID).isEmpty() =
            || (sgb.getCourseIDFromStudentGrade( courseID: CourseID).isEmpty() == false && cobll.getCourseIDFromCourse( courseID: CourseID).isEmpty(
        result = 0;
    } else {
        oldal.deleteOnlineCourse(CourseID);
        cobll.deleteCourse(CourseID);
        result = 1;
    }
    return result;
}
```

10.2.3. UI

```
private void btnDeleteMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();

        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please choose one row in table !",  title: "Message",  messageType: JOptionPane.
        } else {
            int CourseID = Integer.parseInt( s: model.getValueAt( rowIndex: row,  columnIndex: 1).toString());
            int input = JOptionPane.showConfirmDialog( parentComponent: null,
                    message: "Do you want to delete this OnlineCourse?",  title: "Warning!",  optionType: JOptionPane.YES_NO_OPTION);

            if (input == 0) {
                if (osbll.deleteOnlineCourse(CourseID) == 1) {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "You have completed to delete online course successfully!",  title: "
                    List list = osbll.LoadOnlineCourse();
                    model = convertOl(list);
                    tbDS.setModel( dataModel: model);
                } else {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Error for Informating binding!",  title: "Message",  messageType: JOptio

                }
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger( name: OnlineCourseForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
    }
}
```

11. Xóa record các khóa học onsite

11.1. Sơ đồ tuần tự



11.2. Code 3 class

11.2.1. DAL

- Course

```
public int deleteCourse(int CourseID) throws SQLException {
    String query = "DELETE FROM course WHERE CourseID = ?";
    PreparedStatement p = CourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: CourseID);
    int result = p.executeUpdate();
    return result;
}
```

- OnsiteCourse

```
public int deleteOnsiteCourse(int CourseID) throws SQLException {
    String query = "DELETE FROM onsitecourse WHERE CourseID = ?";
    PreparedStatement p = OnsiteCourseDAL.getConnection().prepareStatement( sql: query);
    p.setInt( parameterIndex: 1,  x: CourseID);
    int result=p.executeUpdate();
    return result;
}
```

11.2.2. BLL

- Course

```
public int deleteCourse(int CourseID) throws SQLException {
    int result = cdal.deleteCourse(CourseID);
    return result;
}
```

```
public List getCourseIDFromCourse(int courseID) throws SQLException {
    List<Course> listTemp;
    listTemp = cdal.getCourseIDFromCourse(courseID);
    return listTemp;
}
```

- OnsiteCourse

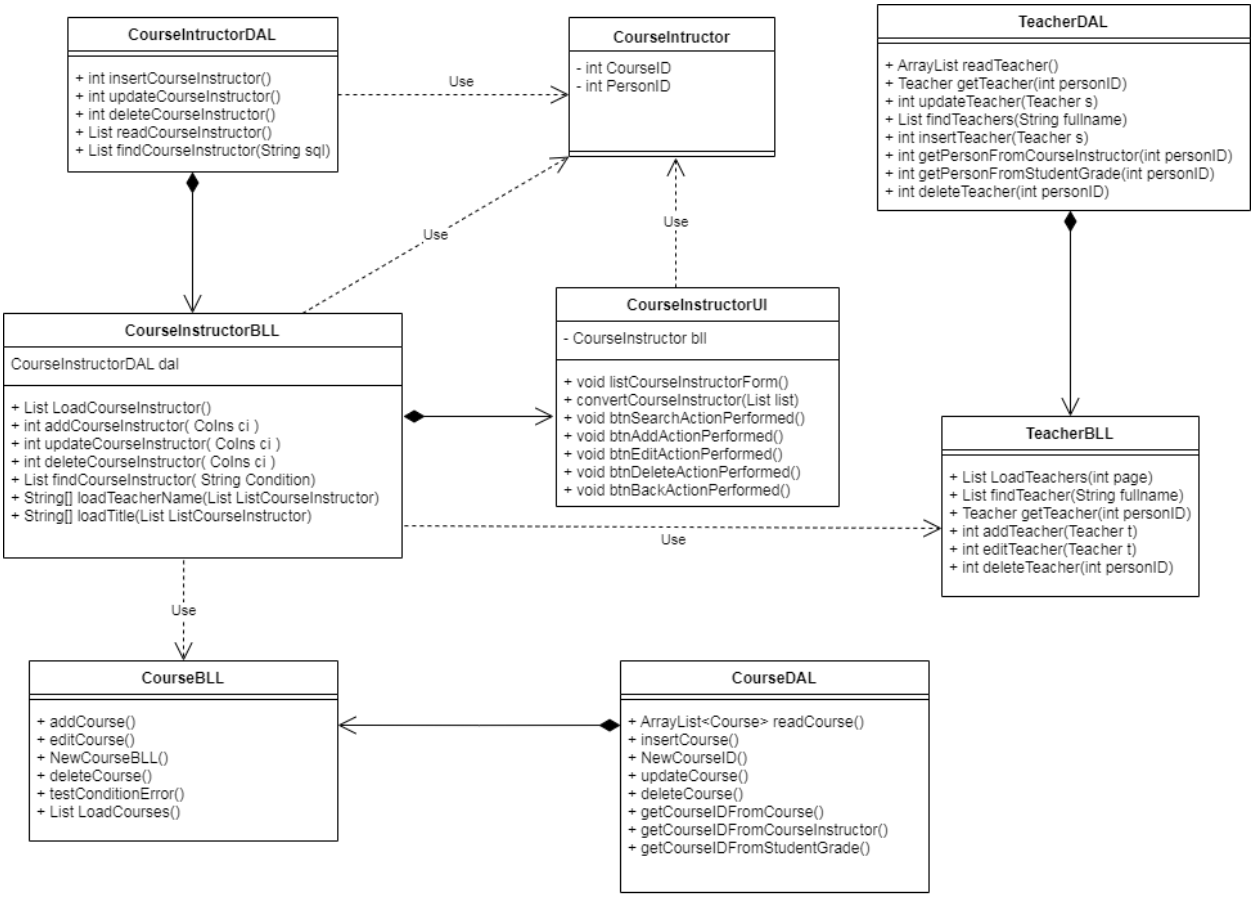
```
public int deleteOnsiteCourse(int CourseID) throws SQLException {
    int result;
    cib = new CourseIntructorBLL();
    sgb = new StudentGradeBLL();
    cobll = new CourseBLL();
    if ((cib.getCourseIDFromCourseInstructor( courseID:CourseID).isEmpty() == false
            && cobll.getCourseIDFromCourse( courseID:CourseID).isEmpty() == false)
            || (sgb.getCourseIDFromStudentGrade( courseID:CourseID).isEmpty() == false
            && cobll.getCourseIDFromCourse( courseID:CourseID).isEmpty() == false)) {
        result = 0;
    } else {
        osdal.deleteOnsiteCourse(CourseID);
        cobll.deleteCourse(CourseID);
        result = 1;
    }
    return result;
}
```

11.2.3. UI

```
private void btnDeleteMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int row = tbDS.getSelectedRow();
        TableModel model = tbDS.getModel();

        if (row < 0) {
            JOptionPane.showMessageDialog( parentComponent:this,  message:"Please choose one row in table !",  title:"Message",  messageType:JOptionPane.
        } else {
            int CourseID = Integer.parseInt( s:model.getValueAt( rowIndex:row,  columnIndex:1).toString());
            int input = JOptionPane.showConfirmDialog( parentComponent:null,
                    message:"Do you want to delete this OnsiteCourse?",  title:"Warning!",  optionType:JOptionPane.YES_NO_OPTION);

            if (input == 0) {
                if (osbll.deleteOnsiteCourse(CourseID) == 1) {
                    JOptionPane.showMessageDialog( parentComponent:this,  message:"You have completed to delete onsite course successfully!",  title:"
                    List list = osbll.LoadOnsiteCourse();
                    model = convertOs(list);
                    tbDS.setModel( dataModel:model);
                } else {
                    JOptionPane.showMessageDialog( parentComponent:this,  message:"Error for Informating binding!",  title:"Message",  messageType:JOptio

                }
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger( name:OnsiteCourseForm.class.getName()).log( level:Level.SEVERE,  msg:null,  thrown:ex);
    }
```

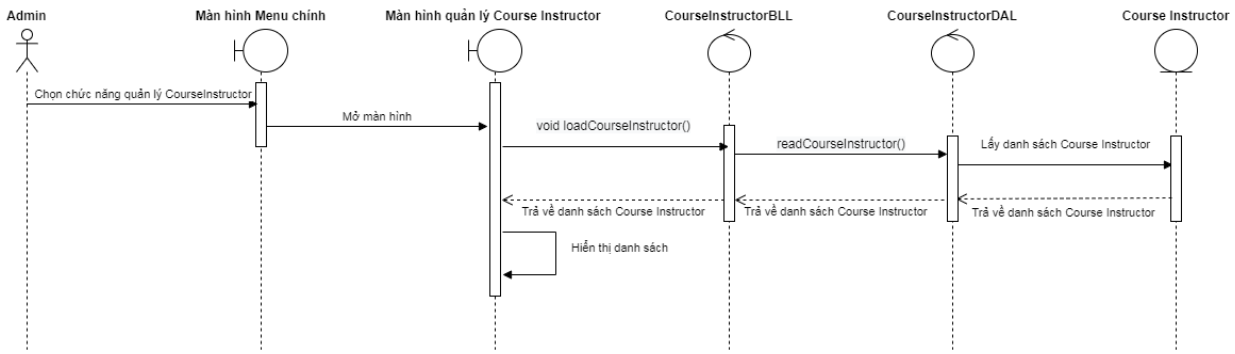
III. Chức năng 3: Quản lý thông tin phân công giảng dạy (CourseIntructor)

1. Sơ đồ class



2. Hiển thị danh sách thông tin phân công giảng dạy

2.1. Sơ đồ tuần tự



2.2. Code 3 class:

2.2.1. DAL

```
public List readCourseInstructor() throws SQLException {
    String query = "SELECT * FROM course as ce, courseinstructor as c, person as p where c.personID = p.personID and c.courseid = ce.courseid";
    ResultSet rs = CourseInstructorDAL.doReadQuery(query);
    ArrayList list = new ArrayList();

    if (rs != null) {
        int i = 1;
        while (rs.next()) {
            CourseInstructor c = new CourseInstructor();
            c.setPersonID(rs.getInt("PersonID"));
            c.setCourseID(rs.getInt("CourseID"));
            list.add(c);
        }
    }
    return list;
}
```

2.2.2. BLL

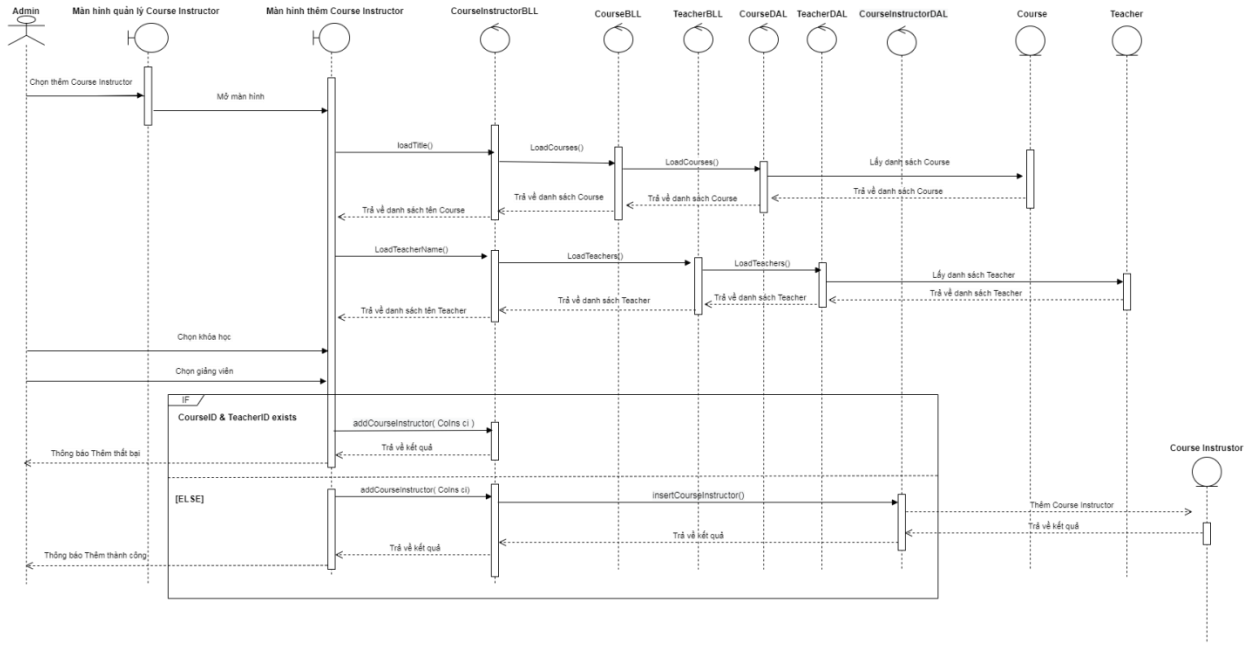
```
public CourseIntructorBLL() {
    try {
        this.list = cid.readCourseInstructor();
    } catch (SQLException ex) {
        Logger.getLogger(CourseIntructorBLL.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public List loadCourseInstructor() throws SQLException {
    list = cid.readCourseInstructor();
    return list;
}
```

2.2.3. UI

```
protected void listCourseInstructorForm() throws SQLException {
    List list = c.loadCourseInstructor();
    DefaultTableModel model = convertCourseInstructor(list);
    tableDark1.setModel(model);
}

private DefaultTableModel convertCourseInstructor(List list) throws SQLException {
    String[] columnNames = {"TT", "PERSONID", "NAME TEACHER", "COURSEID", "TITLE COURSE"};
    String[] teacherNames = c.loadTeacheName(list);
    String[] titles = c.loadTitle(list);
    Object[][] data;
    data = new Object[list.size()][5];
    for (int i = 0; i < list.size(); i++) {
        CourseInstructor c = (CourseInstructor) list.get(i);
        data[i][0] = i + 1;
        data[i][1] = c.getPersonID();
        //courseBLL
        data[i][4] = titles[i];
        // personBLL
        data[i][2] = teacherNames[i];
        //nameteacher
        data[i][3] = c.getCourseID();

    }
    DefaultTableModel model = new DefaultTableModel(data, columnNames);
    return model;
}
```

3. Thêm mới thông tin phân công giảng dạy

3.1. Sơ đồ tuần tự



3.2. Code 3 class

3.2.1. DAL

```
public int insertCourseInstructor(CourseInstructor c) throws SQLException {
    String query = "Insert courseinstructor (courseID, personID) VALUES (?, ?)";
    PreparedStatement p = CourseInstructorDAL.getConnection().prepareStatement(query);
    p.setString(1, c.getCourseID() + "");
    p.setString(2, c.getPersonID() + "");

    int result = p.executeUpdate();
    return result;
}
```

3.2.2. BLL

```
public int addCourseInstructor(CourseInstructor c) throws SQLException {
    // check duplicate
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i).getPersonID() == c.getPersonID() && list.get(i).getCourseID() == c.getCourseID()) {
            return -2;
        }
    }
    int result = cid.insertCourseInstructor(c);
    return result;
}
```

3.2.3. UI

- CourseInstructorAddForm

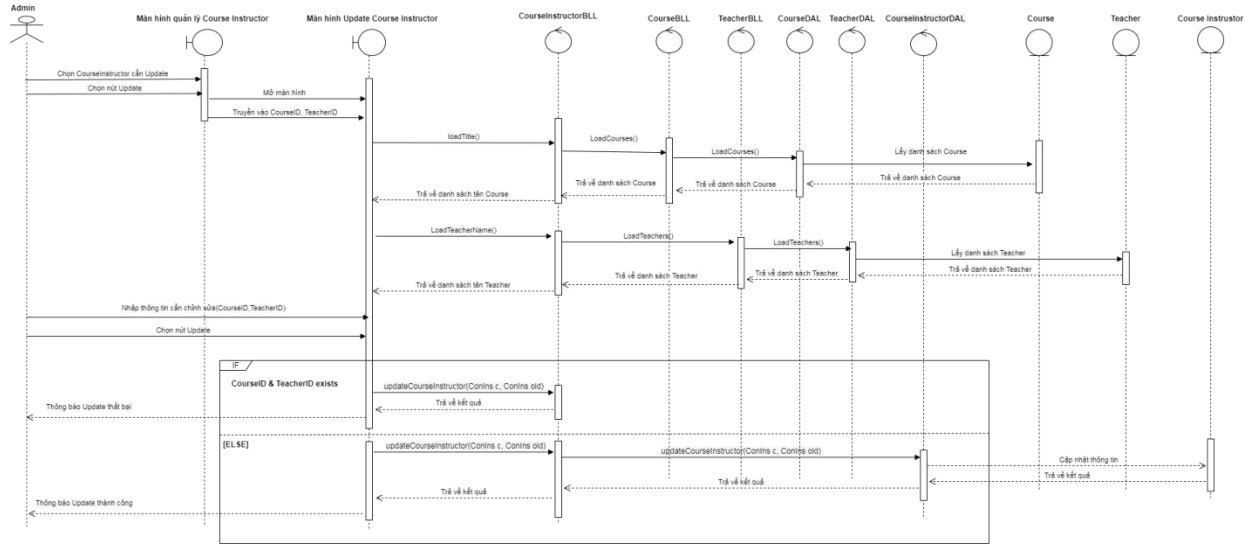
```
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    int result = -1;
    try {

        result = c.addCourseInstructor(
                new CourseInstructor(
                        CourseID: Integer.parseInt( s: comboBoxSuggestion1.getSelectedItem().toString()),
                        PersonID: Integer.parseInt( s: personIDCBB.getSelectedItem().toString())
                )
        );
    } catch (SQLException ex) {
        Logger.getLogger( name: CourseInstructorAddForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
    }
    try {
        if (result > 0) {
            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "You have completed to add Course Instruction successfully!", title: "Message", messageType: JO
            home.setVisible( b: true);
            home.initTable();
            this.dispose();

        } else if (result == -2) {

            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "Cannot add due to duplicate course ID and person ID", title: "Message", messageType: JOptionPa
        } else {
            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "can't add data", title: "Message", messageType: JOptionPane.WARNING_MESSAGE);
        }
    } catch (Exception e) {
    }
}
```

- CourseInstructorForm

```
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    CourseInstructorAddForm C_ADD_FORM = new CourseInstructorAddForm( parent: this,  modal: false);
    C_ADD_FORM.setVisible( b: true);
    C_ADD_FORM.setLocationRelativeTo( c: null);
}
```

4. Sửa / Cập nhật thông tin phân công giảng dạy

4.1. Sơ đồ tuần tự



4.2. Code 3 class

4.2.1. DAL

```
public int updateCourseInstructor(CourseInstructor c, CourseInstructor old ) throws SQLException {
    String query = "Update  CourseInstructor SET CourseID = ? , PersonID = ? where CourseID = ? and PersonID = ? ";
    PreparedStatement p = CourseInstructorDAL.getConnection().prepareStatement(query);
    p.setInt(1, c.getCourseID());
    p.setInt(2, c.getPersonID());
    p.setInt(3, old.getCourseID());
    p.setInt(4, old.getPersonID());
    int result = p.executeUpdate();
    return result;
}
```

4.2.2. BLL

```
public int updateCourseInstructor(CourseInstructor c, CourseInstructor old) throws SQLException {
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i).getPersonID() == c.getPersonID() && list.get(i).getCourseID() == c.getCourseID()) {
            return -2;
        }
    }
    int result = cid.updateCourseInstructor(c, old);
    return result;
}
```

4.2.3. UI

- CourseInstructorEditForm

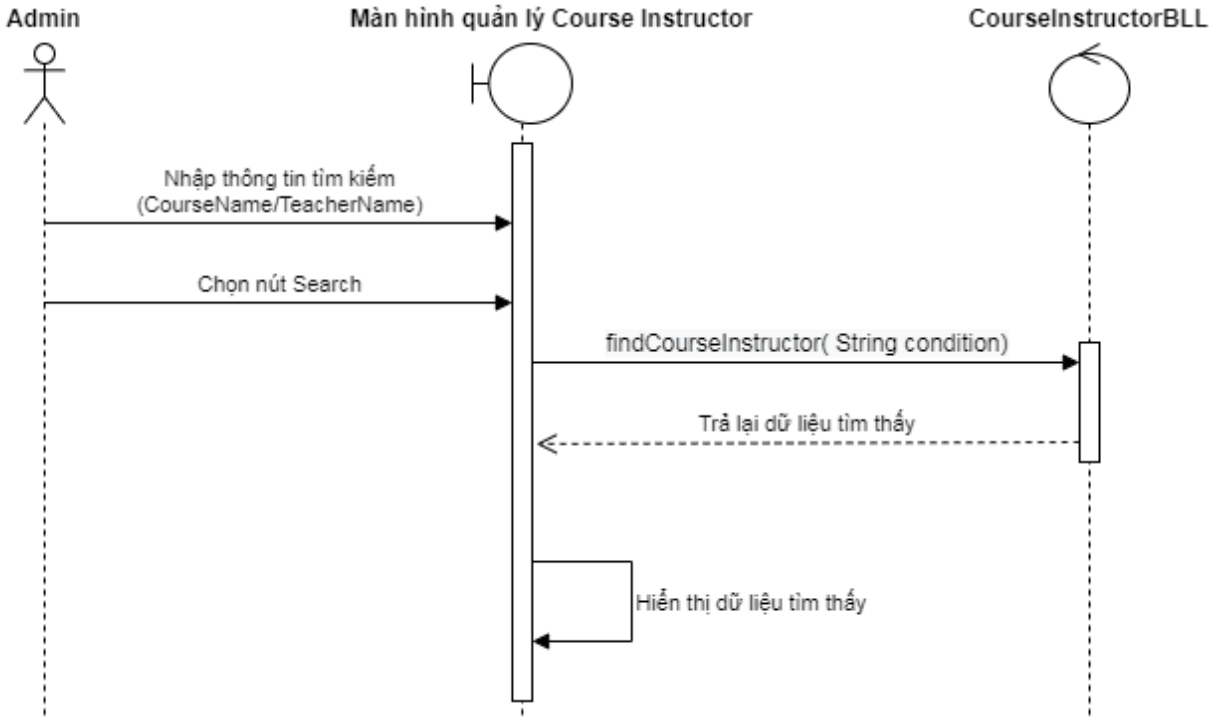
```
private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {

    int result = -1;
    int CourseID = Integer.parseInt(IDS.split( regex: "\\|")[0]);
    int PersonID = Integer.parseInt(IDS.split( regex: "\\|")[1]);
    int newCourseID = Integer.parseInt( s: comboBoxSuggestion1.getSelectedItem().toString());
    int newPersonID = Integer.parseInt( s: comboBoxSuggestion3.getSelectedItem().toString());

    try {
        if (CourseID == newCourseID && PersonID == newPersonID) {
            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "Old CourseInstruction", title: "Message", messageType: JOptionPane.INFORMATION_MESSAGE);
        } else if (CourseID != newCourseID || PersonID != newCourseID) {
            try {

                result = c.updateCourseInstructor(
                        new CourseInstructor( CourseID: newCourseID, PersonID: newPersonID),
                        new CourseInstructor(CourseID, PersonID)
                );

            } catch (SQLException ex) {
                Logger.getLogger( name: CourseInstructorAddForm.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);
            }
            if (result > 0) {
                IDS = newCourseID + "|" + newPersonID+ "";
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "You have completed to edit Course Instrustor successfully!", title: "Message", messageType: JOptionPane.INFORMATION_MES
                home.listCourseInstructorForm();
            } else if (result == -2) {

                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "You haven't completed to edit Course Instrustion! foregin key", title: "Message", messageType: JOptionPane.ERROR_MESSAG
            } else {
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "Error different", title: "Message", messageType: JOptionPane.ERROR_MESSAGE);

            }
        }
    } catch (Exception e) {
    }
}
```

- CourseInstructorForm

```
private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {

    if (tableDark1.getSelectedRow() == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,
                message: "Pleases choose row!", title: "Message", messageType: JOptionPane.ERROR_MESSAGE);
    } else {
        try {
            List<CourseInstructor> list = c.loadCourseInstructor();
            IDS = list.get( index: tableDark1.getSelectedRow()).getCourseID() + "|"
                    + list.get( index: tableDark1.getSelectedRow()).getPersonID() + "";
        } catch (SQLException ex) {
            Logger.getLogger( name: CourseInstructorForm.class.getName()).log( level: Level.SEVERE, msg: null,
        }
        CourseInstructorEditForm C_EDIT_FORM = new CourseInstructorEditForm( parent: this, modal: false);
        C_EDIT_FORM.setVisible( b: true);
        C_EDIT_FORM.setLocationRelativeTo( c: null);
    }
```

5. Tìm kiếm thông tin phân công giảng dạy

5.1. Sơ đồ tuần tự



5.2. Code 3 class

5.2.1. DAL

```
public List findCourseInstructor(String sql) throws SQLException {
    String query = "SELECT * FROM course as ce, courseinstructor as c, person as p where c.personID = p.personID and c.courseid = ce.courseid";
    ResultSet rs = CourseInstructorDAL.doReadQuery(query);
    ArrayList list = new ArrayList();

    if (rs != null) {
        int i = 1;
        while (rs.next()) {
            CourseInstructor c = new CourseInstructor();
            c.setPersonID(rs.getInt("PersonID"));
            c.setCourseID(rs.getInt("CourseID"));
            list.add(c);
        }
    }
    return list;
}
```

5.2.2. BLL

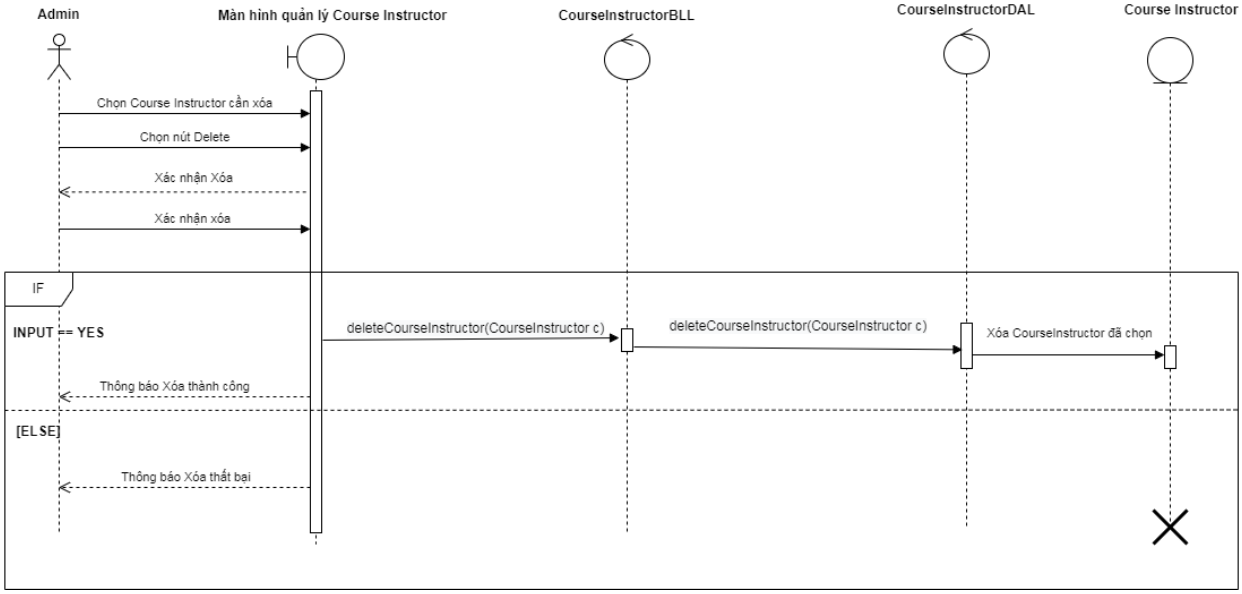
```
public List findCourseInstructor(String condition) throws SQLException {
    ArrayList<CourseInstructor> list = (ArrayList<CourseInstructor>) cid.readCourseInstructor();
    ArrayList<CourseInstructor> listSearchs = new ArrayList<CourseInstructor>();
    condition = condition.trim().replaceAll("  +", " ").toLowerCase();
    String oldCondition = condition;
    String[] conditions = condition.split(" ");

    for (int i = 0; i < list.size(); i++) {
        String regex = list.get(i).getALL() + teacherNames[i] + " " + Titles[i];
        for (int j = 0; j < conditions.length; j++) {
            String oldChirlCondition = conditions[j];
            conditions[j] = "(.*)" + conditions[j] + "(.*)";
            if (regex.toLowerCase().matches(conditions[j])) {
/                   System.out.println(list.get(i).getPersonID()+" -> ID" + list.get(i).getCourseID());
                listSearchs.add(list.get(i));
                break;
            }
            conditions[j] = oldChirlCondition;
        }

    }
    if(listSearchs.size() == 0) return listSearchs;
/     System.out.println(listSearchs.size());
    return listSearchs;
}
```

5.2.3. UI

```
private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        listCourseInstructorForm();
    } catch (SQLException ex) {
        Logger.getLogger( name: CourseInstructorForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thr
    }
    String stringToSearchs = searchField1.getText();
    try {

        List list = c.findCourseInstructor( condition: stringToSearchs);
        DefaultTableModel model = convertCourseInstructor(list);
        tableDark1.setModel( dataModel: model);
    } catch (SQLException ex) {
        Logger.getLogger( name: CourseInstructorForm.class.getName()).log( level: Level.SEVERE,  msg: null,  thr
    }
}
```

6. Xóa record thông tin phân công giảng dạy

6.1. Sơ đồ tuần tự



6.2. Code 3 class

6.2.1. DAL

```
public int deleteCourseInstructor(CourseInstructor c) throws SQLException {
    int result = -1;
    String query = "DELETE FROM CourseInstructor WHERE PersonID = ? and CourseID = ?";
    PreparedStatement p = CourseInstructorDAL.getConnection().prepareStatement(query);
    p.setInt(1, c.getPersonID());
    p.setInt(2, c.getCourseID());
    result = p.executeUpdate();

    return result;
}
```

6.2.2. BLL

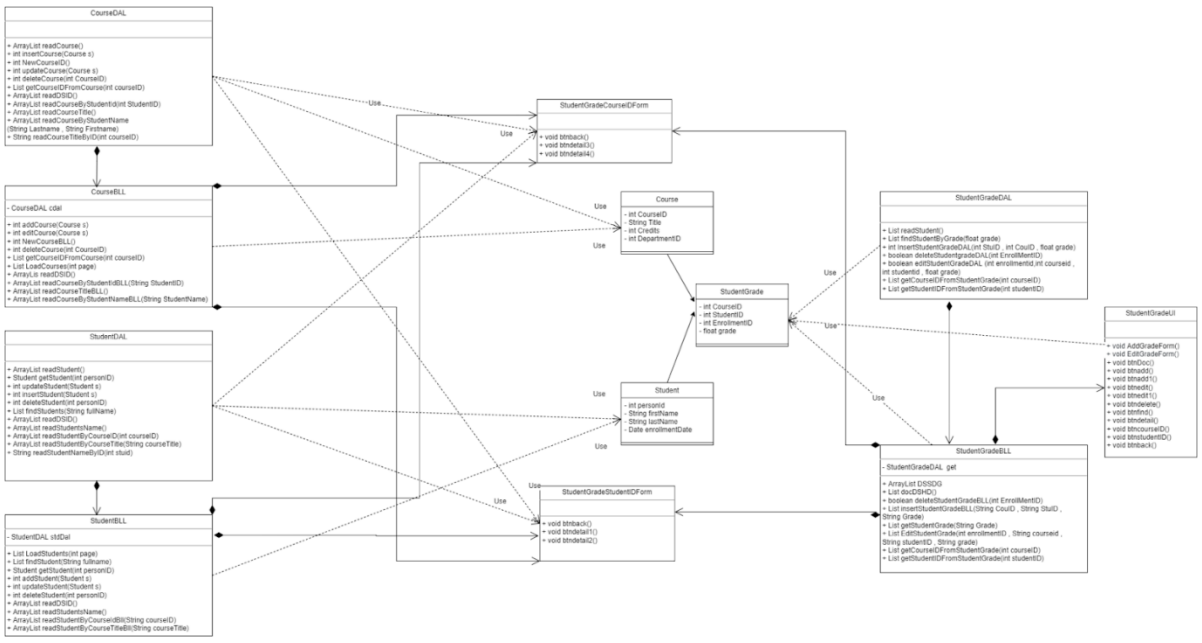
```
public int deleteCourseInstructor(CourseInstructor c) throws SQLException {
     int result = cid.deleteCourseInstructor(c);
     return result;
}
```

6.2.3. UI

```
    private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
        if (tableDark1.getSelectedRow() == -1) {
            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "chose row!",  title: "Message",  messageType: JOptionPane.ERROR_MESSAGE);
        } else {
            int confirm = JOptionPane.showConfirmDialog( parentComponent: this,
                    message: "You want delete this row",
                    title: "Yes",
                    optionType: JOptionPane.YES_NO_OPTION,
                    messageType: JOptionPane.QUESTION_MESSAGE);
            if (confirm == JOptionPane.YES_OPTION) {
                try {
                    List<CourseInstructor> list = c.loadCourseInstructor();
                    int result = c.deleteCourseInstructor(new CourseInstructor( CourseID: list.get( index: tableDa
                            PersonID: list.get( index: tableDark1.getSelectedRow()).getPersonID()));
                    if (result == -1) {
                        JOptionPane.showMessageDialog( parentComponent: this,
                                message: "Deleted fail!",  title: "Message",  messageType: JOptionPane.ERROR_MESSAGE
                    } else {
                        JOptionPane.showMessageDialog( parentComponent: this,
                                message: "Deleted success!",  title: "Message",  messageType: JOptionPane.INFORMATIC
                        initTable();
                    }

                } catch (SQLException ex) {
                    Logger.getLogger( name: CourseInstructorForm.class.getName()).log( level: Level.SEVERE,  msg: n
                }
            }

        }
```

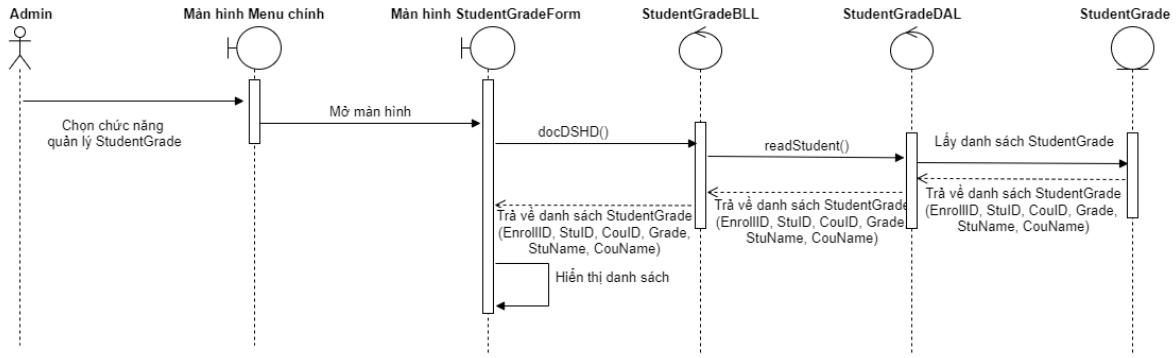
IV. Chức năng 4: Quản lý kết quả khóa học (Student Grade)

1. Sơ đồ class



2. Hiển thị danh sách kết quả khóa học

2.1. Sơ đồ tuần tự



2.2. Code 3 class:

2.2.1. DAL

```
public List readStudent() {
    List list = new ArrayList();
    try {
        String query = "SELECT studentgrade.* , Lastname , Firstname , Title FROM `course`,`person`,`studentgrade` "
                + "WHERE studentgrade.CourseID = course.CourseID and StudentID = PersonID ORDER By EnrollmentID ASC";
        ResultSet rs = this.doReadQuery( sql: query);
        if(rs !=null){
            while(rs.next()){

                List list2 = new ArrayList();
                StudentGrade s = new StudentGrade();

                s.setEnrollmentID( EnrollmentID: rs.getInt( columnLabel: "EnrollmentID"));
                s.setCourseID( CourseID: rs.getInt( columnLabel: "CourseID"));
                s.setStudentID( StudentID: rs.getInt( columnLabel: "StudentID"));
                s.setGrade( grade: rs.getFloat( columnLabel: "Grade"));

                String fullname = rs.getString( columnLabel: "Lastname") + " " + rs.getString( columnLabel: "Firstname");
                String title = rs.getString( columnLabel: "Title");

                list2.add( e: s);
                list2.add( e: fullname);
                list2.add( e: title);
                list.add( e: list2);
            }
        }
    } catch (Exception e) {
        System.out.println( x: "Eroor");
    }
    return list;
}
```

2.2.2. BLL

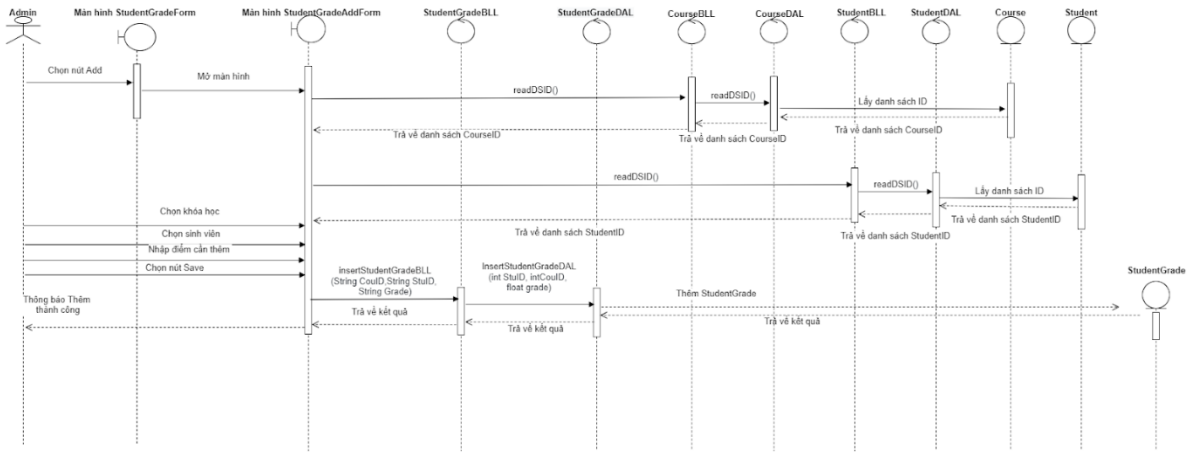
```
public List docDSHD(){

    List list = new ArrayList();
    if(DSSDG==null){
        StudentGradeDAL get=new StudentGradeDAL();
        DSSDG = new ArrayList<>();
        list = get.readStudent();
        for(int i=0;i<list.size();i++){
            List l2 = (List)list.get( index: i);
            DSSDG.add((StudentGrade)l2.get( index: 0));
        }
    }
    return list;
}
```

2.2.3. UI

```
public void btnDoc(){
    List list = new ArrayList();
    StudentGradeBLL bus=new StudentGradeBLL();
    if(StudentGradeBLL.DSSDG==null)
        list = bus.docDSHD();
    int count=1;
    DefaultTableModel model =(DefaultTableModel)tbl.getModel();
    model.setRowCount( rowCount: 0);
    for(StudentGrade s :StudentGradeBLL.DSSDG){
            Vector row=new Vector();
            List tamp = (List)list.get(count -1);
            String name =(String) tamp.get( index: 1);
            String title =  (String) tamp.get( index: 2);
            row.add( e: count);
            row.add( e: s.getEnrollmentID());
            row.add( e: s.getCourseID());
            row.add( e: s.getStudentID());
            row.add( e: s.getGrade());
            row.add( e: name);
            row.add( e: title);
            model.addRow( rowData: row);
            count++;
    }
}
```

3. Thêm mới thông tin kết quả khóa học

3.1. Sơ đồ tuần tự



3.2. Code 3 class

3.2.1. DAL

```
public int InsertStudentGradeDAL(int StuID , int CouID , float grade){
    try{
        String sql = "INSERT INTO `studentgrade` (`CourseID`, `StudentID`, `Grade`) VALUES (?,?,?)";
        PreparedStatement pstmt =this.getConnection().prepareStatement(sql, autoGeneratedKeys: Statement.RETURN_GENERATED_KEYS);
        pstmt.setInt( parameterIndex: 1,  x: CouID);
        pstmt.setInt( parameterIndex: 2,  x: StuID);
        pstmt.setFloat( parameterIndex: 3, x: grade);
        pstmt.executeUpdate();
        int generakey = 0;
        try (ResultSet generatedKeys = pstmt.getGeneratedKeys()) {
            if (generatedKeys.next()) {
                generakey = generatedKeys.getInt( columnIndex: 1);
            }
        }
        return generakey;
    }catch(Exception ex){
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Lỗi kết nối");
    }
    return -2;
}
```

3.2.2. BLL

```
public List insertStudentGradeBLL(String CouID , String StuID , String Grade){
    List list = new ArrayList();
    int courseid = Integer.parseInt( s: CouID);
    int stdentid = Integer.parseInt( s: StuID);
    float grade = Float.parseFloat( s: Grade);

    int Enrollmentid = get.InsertStudentGradeDAL( StuID: stdentid,  CouID: courseid, grade);
    if(Enrollmentid >= 0){
        StudentDAL sdal = new StudentDAL();
        CourseDAL c = new CourseDAL();
        StudentGrade g = new StudentGrade();
        g.setEnrollmentID( EnrollmentID: Enrollmentid);
        g.setCourseID( CourseID: courseid);
        g.setStudentID( StudentID: stdentid);
        g.setGrade(grade);
        String name = sdal.readStudentNameByID( stuid: stdentid);
        String title = c.readCourseTitleByID( courseID: courseid);
        DSSDG.add( e: g);

        list.add( e: g);
        list.add( e: name);
        list.add( e: title);
    }
    return list;
}
```

3.2.3. UI

- StudentGradeForm

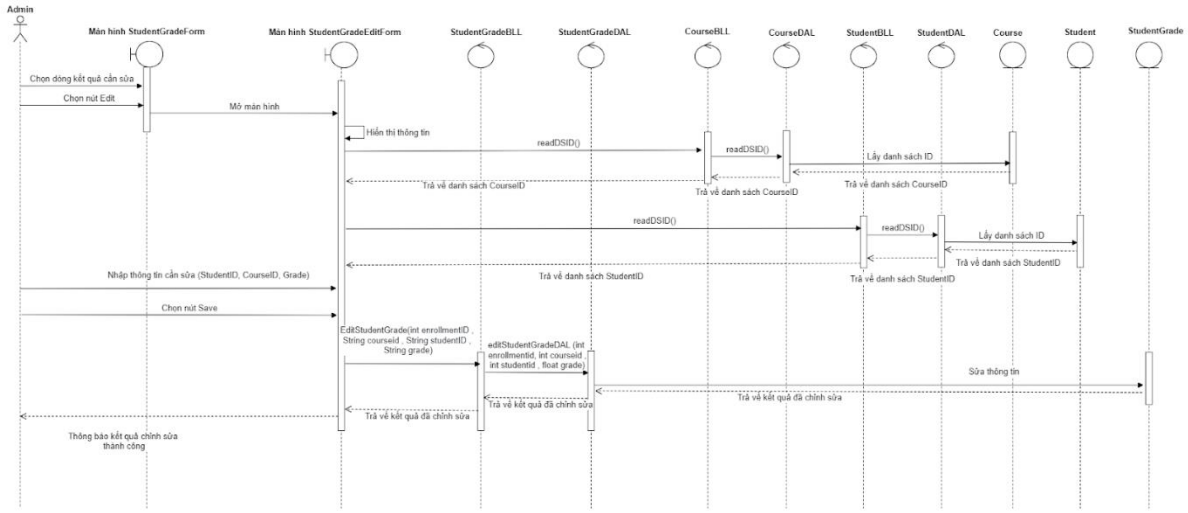
```
public static void AddGradeForm(StudentGrade s , String name , String title)
{
    DefaultTableModel model = (DefaultTableModel)tbl.getModel();
    int Rowcount = tbl.getRowCount() + 1;
    Vector row=new Vector();
    row.add( e: Rowcount);
    row.add( e: s.getEnrollmentID());
    row.add( e: s.getCourseID());
    row.add( e: s.getStudentID());
    row.add( e: s.getGrade());
    row.add( e: name);
    row.add( e: title);
    model.addRow( rowData: row);
}
```

- StudentGradeAddForm

```
private void btnadd1ActionPerformed(java.awt.event.ActionEvent evt) {
    String courseid = cbcourse.getSelectedItem().toString();
    String stdentid = cbstudent.getSelectedItem().toString();
    String grade = txtgrade.getText();
     if(grade.equals( anObject: "")){
        JOptionPane.showMessageDialog( parentComponent: rootPane,  message: "The data must be filled in completely");
     }
     else{
        StudentGradeBLL bll = new StudentGradeBLL();
        List list = bll.insertStudentGradeBLL( CouID: courseid,  StuID: stdentid,  Grade: grade);

        if(list.size() > 0){
            JOptionPane.showMessageDialog( parentComponent: rootPane,  message: "Add successfully");
            StudentGrade s = StudentGradeBLL.DSSDG.get(StudentGradeBLL.DSSDG.size()-1);
            String name = (String) list.get( index: 1);
            String title = (String) list.get( index: 2);
            StudentGradeForm.AddGradeForm(s,name,title);
            txtgrade.setText( t: "");
        }
     }

}
```

4. Sửa / Cập nhật thông tin kết quả khóa học

4.1. Sơ đồ tuần tự



4.2. Code 3 class

4.2.1. DAL

```
public boolean editStudentGradeDAL (int enrollmentid,int courseid , int studentid , float grade){
    try {
        String query = "UPDATE `studentgrade` SET `CourseID`=?,`StudentID`=?,`Grade`=? WHERE `EnrollmentID` = '"+enrollmentid+"'";
        PreparedStatement stament=this.getConnection().prepareStatement( sql: query);
        stament.setInt( parameterIndex: 1,  x: courseid);
        stament.setInt( parameterIndex: 2,  x: studentid);
        stament.setFloat( parameterIndex: 3,  x: grade);
        stament.executeUpdate();
        return true;
    } catch (Exception e) {
    }
    return false;
}
```

4.2.2. BLL

```
public List EditStudentGrade(int enrollmentID , String courseid , String studentID , String grade){
    List list = new ArrayList();
    int courseID2 = Integer.parseInt( s: courseid);
    int studentID2 = Integer.parseInt( s: studentID);
    float grade2 = Float.parseFloat( s: grade);

    boolean check = get.editStudentGradeDAL( enrollmentid: enrollmentID,  courseid: courseID2,  studentid: studentID2,  grade: grade2);
    int i=0;
    for(StudentGrade s : DSSDG){
        if(s.getEnrollmentID() == enrollmentID){
            StudentGrade s2 = new StudentGrade();
            s2.setEnrollmentID( EnrollmentID: enrollmentID);
            s2.setCourseID( CourseID: courseID2);
            s2.setStudentID( StudentID: studentID2);
            s2.setGrade( grade: grade2);
            DSSDG.set( index: i,  element: s2);
            list.add( e: DSSDG.get( index: i));
            break;
        }
        i++;
    }
    StudentDAL sdal = new StudentDAL();
    CourseDAL c = new CourseDAL();
    String name = sdal.readStudentNameByID( stuid: studentID2);
    String title = c.readCourseTitleByID( courseID: courseID2);
    list.add( e: name);
    list.add( e: title);
    return list;
}
```

4.2.3. UI

- StudentGradeForm

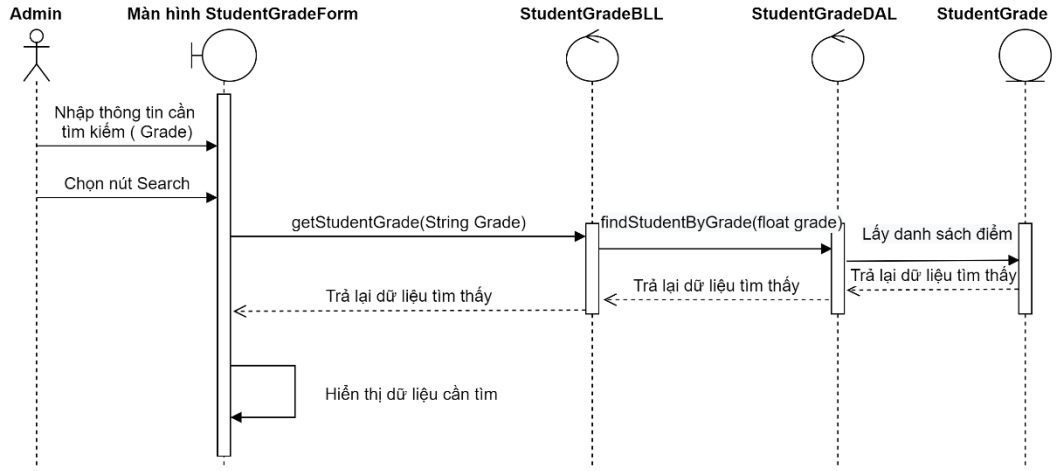
```
public static void EditGradeForm(StudentGrade s,String name, String title){
    DefaultTableModel model = (DefaultTableModel)tbl.getModel();
    int i = tbl.getSelectedRow();
    model.setValueAt( aValue: s.getEnrollmentID(),  row: i,  column: 1);
    model.setValueAt( aValue: s.getCourseID(),  row: i,  column: 2);
    model.setValueAt( aValue: s.getStudentID(),  row: i,  column: 3);
    model.setValueAt( aValue: s.getGrade(),  row: i,  column: 4);
    model.setValueAt( aValue: name,  row: i,  column: 5);
    model.setValueAt( aValue: title,  row: i,  column: 6);
}
```

- StudentGradeEditForm

```
private void btneditlActionPerformed(java.awt.event.ActionEvent evt) {
    String courseid = cbcourse.getSelectedItem().toString();
    String stdentid = cbstudent.getSelectedItem().toString();
    String grade = txtgrade.getText();
    if(grade.equals( anObject: "")){
        JOptionPane.showMessageDialog( parentComponent: rootPane,  message: "The data must be filled in completely");
    }
    else{
        StudentGradeBLL bll = new StudentGradeBLL();
        List list = bll.EditStudentGrade( enrollmentID: EnrollmentID, courseid,  studentID: stdentid, grade);

        if(list.size() > 0){
            JOptionPane.showMessageDialog( parentComponent: rootPane,  message: "Edit success");
            StudentGrade s = (StudentGrade) list.get( index: 0);
            String name = (String) list.get( index: 1);
            String title = (String) list.get( index: 2);
            StudentGradeForm.EditGradeForm(s,name,title);
            txtgrade.setText( t: "");
        }
        else{
            JOptionPane.showMessageDialog( parentComponent: rootPane,  message: "Edit fail");
        }
    }
}
```

5. Tìm kiếm kết quả khóa học

5.1. Sơ đồ tuần tự



5.2. Code 3 class

5.2.1. DAL

```
public List findStudentByGrade(float grade){
    List list = new ArrayList();
    try {
        String query = "SELECT studentgrade.* , Lastname , Firstname , Title FROM `course`,`person`,`studentgrade` "
                + "WHERE studentgrade.CourseID = course.CourseID and StudentID = PersonID and Grade = '"+grade+"' ORDER By EnrollmentID ASC";
        ResultSet rs = this.doReadQuery( sql: query);
        if(rs !=null){
            while(rs.next()){

                List list2 = new ArrayList();
                StudentGrade s = new StudentGrade();

                s.setEnrollmentID( EnrollmentID: rs.getInt( columnLabel: "EnrollmentID"));
                s.setCourseID( CourseID: rs.getInt( columnLabel: "CourseID"));
                s.setStudentID( StudentID: rs.getInt( columnLabel: "StudentID"));
                s.setGrade( grade: rs.getFloat( columnLabel: "Grade"));

                String fullname = rs.getString( columnLabel: "Lastname") + " " + rs.getString( columnLabel: "Firstname");
                String title = rs.getString( columnLabel: "Title");

                list2.add( e: s);
                list2.add( e: fullname);
                list2.add( e: title);
                list.add( e: list2);
            }
        }
    } catch (Exception e) {
    }
    return list;
}
```

5.2.2. BLL

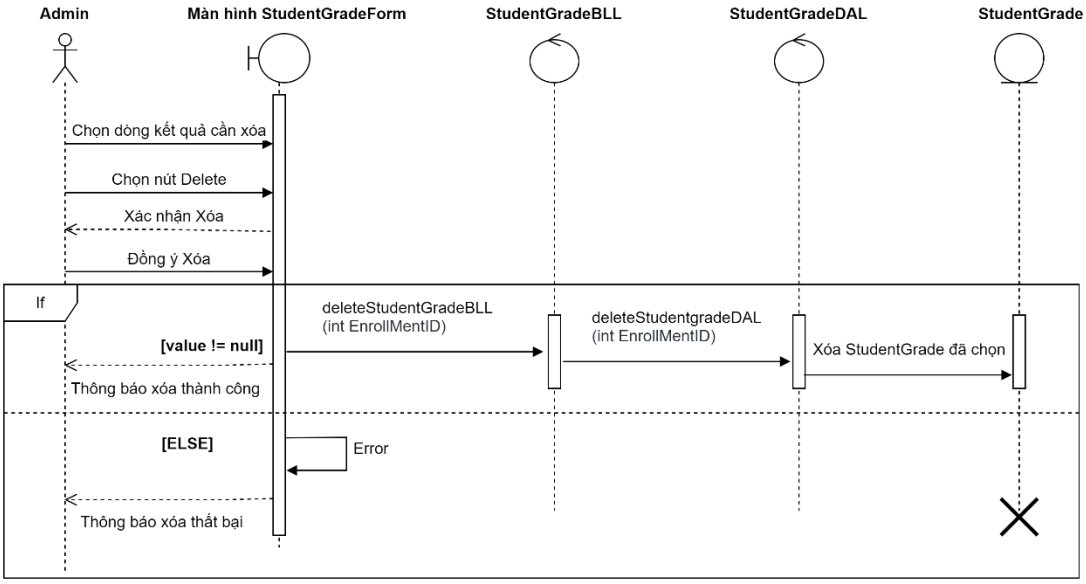
```
public List getStudentGrade(String Grade){
    float grade = Float.parseFloat( s: Grade);
    List list = new ArrayList();
    StudentGradeDAL get=new StudentGradeDAL();
    list = get.findStudentByGrade(grade);
    return list;
}
```

5.2.3. UI

```
private void btnfindActionPerformed(java.awt.event.ActionEvent evt) {
    if(txtfind.getText().equals( anObject: "")){
        JOptionPane.showMessageDialog( parentComponent: rootPane, message: "Please enter the grade to find ");
     }
     else{
        StudentGradeBLL bll = new StudentGradeBLL();
        List list = bll.getStudentGrade( Grade: txtfind.getText());
        if(list.size() > 0){
            DefaultTableModel model = (DefaultTableModel) tbl.getModel();
            model.setRowCount( rowCount: 0);
            int count = 0 ;
            for(int i=0 ;i<list.size() ;i++){
                List tamp = (List)list.get( index: count);
                StudentGrade s = (StudentGrade) tamp.get( index: 0);
                String name =(String) tamp.get( index: 1);
                String title =  (String) tamp.get( index: 2);
                Vector row=new Vector();
                row.add( e: count);
                row.add( e: s.getEnrollmentID());
                row.add( e: s.getCourseID());
                row.add( e: s.getStudentID());
                row.add( e: s.getGrade());
                row.add( e: name);
                row.add( e: title);
                model.addRow( rowData: row);
                count++;
            }
        }
        else{
          JOptionPane.showMessageDialog( parentComponent: rootPane, message: "Not found ");
        }
     }
}
```

6. Xóa record kết quả khóa học

6.1. Sơ đồ tuần tự



6.2. Code 3 class

6.2.1. DAL

```
public boolean deleteStudentgradeDAL(int EnrollMentID){
    try {
        String query = "DELETE FROM `studentgrade` WHERE `EnrollmentID` = '"+ EnrollMentID + "'";
        PreparedStatement stament=this.getConnection().prepareStatement( sql: query);
        stament.executeUpdate();
        return true;
    } catch (Exception e) {
    }
    return false;
}
```

6.2.2. BLL

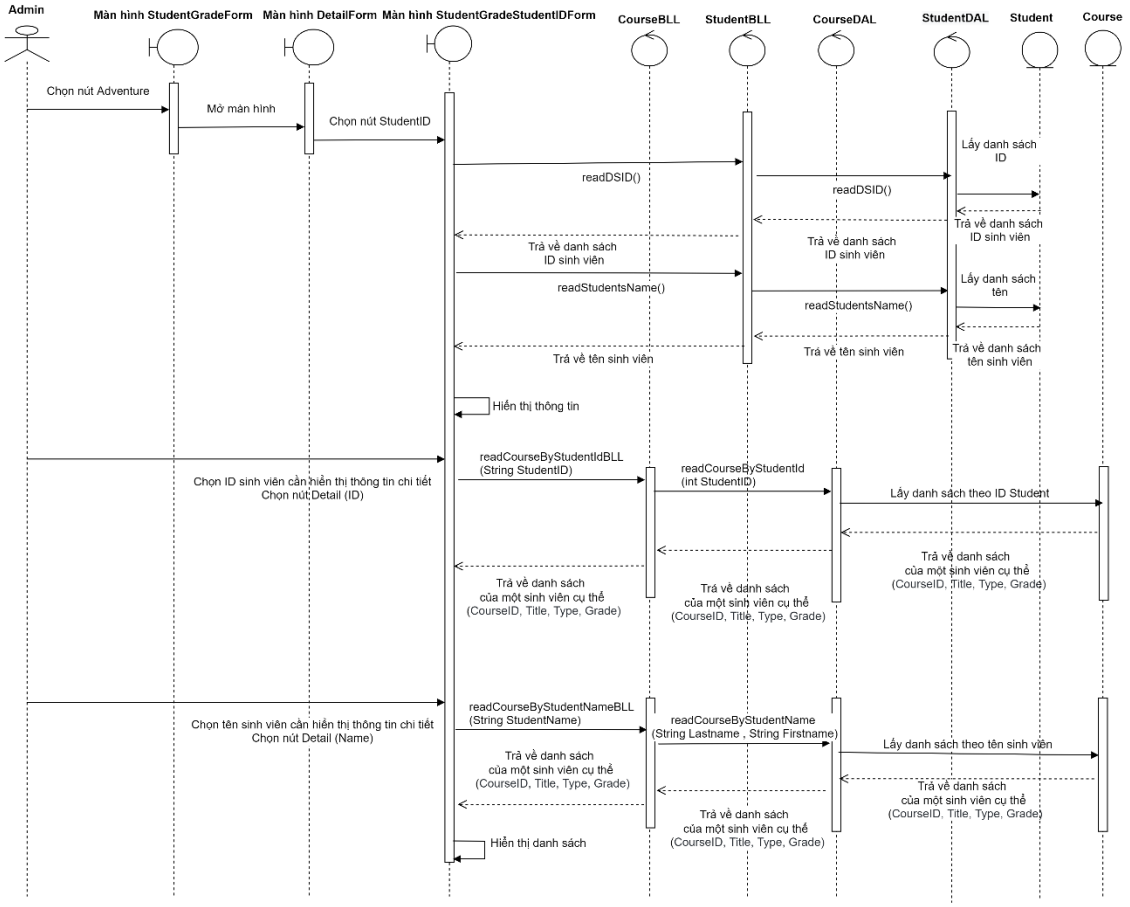
```
public boolean deleteStudentGradeBLL(int EnrollMentID){
    boolean check = get.deleteStudentgradeDAL(EnrollMentID);
    return check;
}
```

6.2.3. UI

```
private void btndeleteActionPerformed(java.awt.event.ActionEvent evt) {
    if(tblRow !=null){
        int input=JOptionPane.showConfirmDialog( parentComponent: null,  message: "Are you sure you want to delete",  title: "Xóa dịch vụ",  optionType: JOptionPane.YES_NO_OPTION);
        if (input==0){
            StudentGradeBLL bll = new StudentGradeBLL();
            if(bll.deleteStudentGradeBLL((int)tblRow.get( index: 0))){
                int i = tbl.getSelectedRow();
                DefaultTableModel model = (DefaultTableModel) tbl.getModel();
                model.removeRow( row: i);
                tbl.setModel( dataModel: model);
               JOptionPane.showMessageDialog( parentComponent: rootPane, message: "Delete successfully ");
            }
            else{
                JOptionPane.showMessageDialog( parentComponent: rootPane, message: "Delete failed ");
            }
        }

    }
    else{
        JOptionPane.showMessageDialog( parentComponent: rootPane, message: "Choose a student to delete ");
    }
}
```

7. Xem thông tin chi tiết của Student

7.1. Sơ đồ trình tự



7.2. Code 3 class

7.2.1. DAL

```
public ArrayList<String[]> readCourseByStudentId(int StudentID){
    ArrayList<String[]> list = new ArrayList<>();
    try {
       String queryOnline = "SELECT onlinecourse.CourseID , Title , Grade FROM `course`,`studentgrade`,`onlinecourse` "
               + "WHERE course.CourseID = onlinecourse.CourseID and studentgrade.CourseID = onlinecourse.CourseID and StudentID = '"+StudentID+"'";
        String queryOnsite = "SELECT onsitecourse.CourseID , Title , Grade FROM `course`,`studentgrade`,`onsitecourse` "
               + "WHERE course.CourseID = onsitecourse.CourseID and studentgrade.CourseID = onsitecourse.CourseID and StudentID = '"+StudentID+"'";
       ResultSet rs = this.doReadQuery( sql:queryOnline);

       if(rs !=null){
           while(rs.next()){
               String CourseID = rs.getString( columnLabel:"CourseID");
               String Title = rs.getString( columnLabel:"Title");
               String Type = "On";
               String Grade = rs.getString( columnLabel:"Grade");
               String[] s = {CourseID,Title,Type,Grade};
               list.add( e:s);
           }
        }
       ResultSet rs2 = this.doReadQuery( sql:queryOnsite);
       if(rs2 !=null){
           while(rs2.next()){
               String CourseID = rs2.getString( columnLabel:"CourseID");
               String Title = rs2.getString( columnLabel:"Title");
               String Type = "Off";
               String Grade = rs2.getString( columnLabel:"Grade");
               String[] s = {CourseID,Title,Type,Grade};
               list.add( e:s);
           }
        }

    } catch (Exception e) {
    }
    return list;
}
```

```
public ArrayList<String[]> readCourseByStudentName(String Lastname , String Firstname){
    ArrayList<String[]> list = new ArrayList<>();
    try {
       String queryOnline = "SELECT onlinecourse.CourseID , Title , Grade FROM `course`,`studentgrade`,`onlinecourse` ,`person`"
               + "WHERE PersonID = studentgrade.StudentID and course.CourseID = onlinecourse.CourseID and studentgrade.CourseID = onlinecourse.CourseID "
              + "and Lastname = '"+Lastname+"' and Firstname = '"+ Firstname + "'";

        String queryOnsite = "SELECT onsitecourse.CourseID , Title , Grade FROM `course`,`studentgrade`,`onsitecourse` , `person` "
               + "WHERE PersonID = studentgrade.StudentID and course.CourseID = onsitecourse.CourseID and studentgrade.CourseID = onsitecourse.CourseID "
               + "and Lastname = '"+Lastname+"' and Firstname = '"+ Firstname + "'";
       ResultSet rs = this.doReadQuery( sql:queryOnline);

       if(rs !=null){
           while(rs.next()){
               String CourseID = rs.getString( columnLabel:"CourseID");
               String Title = rs.getString( columnLabel:"Title");
               String Type = "On";
               String Grade = rs.getString( columnLabel:"Grade");
               String[] s = {CourseID,Title,Type,Grade};
               list.add( e:s);
           }
        }
       ResultSet rs2 = this.doReadQuery( sql:queryOnsite);
       if(rs2 !=null){
           while(rs2.next()){
               String CourseID = rs2.getString( columnLabel:"CourseID");
               String Title = rs2.getString( columnLabel:"Title");
               String Type = "Off";
               String Grade = rs2.getString( columnLabel:"Grade");
               String[] s = {CourseID,Title,Type,Grade};
               list.add( e:s);
           }
        }

    } catch (Exception e) {
    }
    return list;
}
```

7.2.2. BLL

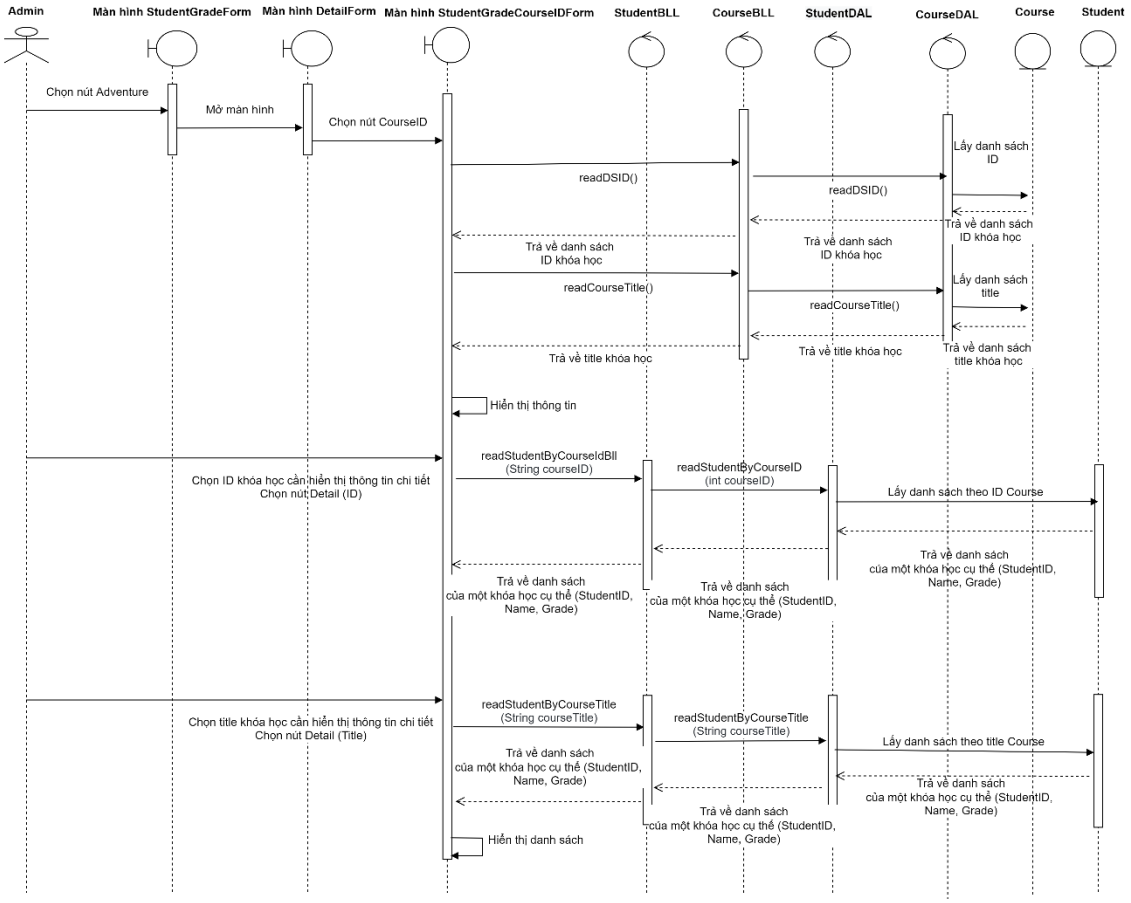
```
public ArrayList<String[]> readCourseByStudentIdBLL(int StudentID){
    CourseDAL std = new CourseDAL();
    ArrayList<String[]> list = std.readCourseByStudentId(StudentID);
    return list;
}
public ArrayList<String[]> readCourseByStudentNameBLL(String StudentName){
    String[] words = StudentName.split( regex: "\\s");
    String LastName = words[0];
    String FirstName = words[1];
    System.out.println(LastName + " " + FirstName);
    CourseDAL std = new CourseDAL();
    ArrayList<String[]> list = std.readCourseByStudentName( Lastname: LastName ,  Firstname: FirstName);
    return list;
}
```

7.2.3. UI

```
private void btndetail1ActionPerformed(java.awt.event.ActionEvent evt) {
    CourseBLL bll = new CourseBLL();
    String stdentid = cbdetail1.getSelectedItem().toString();
    ArrayList<String[]> ds = bll.readCourseByStudentIdBLL( StudentID: stdentid);
    model.setRowCount( rowCount: 0);
    int count = 0 ;
    for(String[] s : ds){
        Vector row=new Vector();
        row.add( e: count);
        row.add(s[0]);
        row.add(s[1]);
        row.add(s[2]);
        row.add(s[3]);
        model.addRow( rowData: row);
        count++;
    }
    tb2.setModel( dataModel: model);
}
private void btndetail2ActionPerformed(java.awt.event.ActionEvent evt) {
    CourseBLL bll = new CourseBLL();
    String stdentName = cbdetail2.getSelectedItem().toString();
    ArrayList<String[]> ds = bll.readCourseByStudentNameBLL( StudentName: stdentName);
    model.setRowCount( rowCount: 0);
    int count = 0 ;
    for(String[] s : ds){
        Vector row=new Vector();
        row.add( e: count);
        row.add(s[0]);
        row.add(s[1]);
        row.add(s[2]);
        row.add(s[3]);
        model.addRow( rowData: row);
        count++;
    }
    tb2.setModel( dataModel: model);
}
```

8. Xem thông tin chi tiết của Course

8.1. Sơ đồ trình tự



8.2. Code 3 class

8.2.1. DAL

```
public ArrayList<String[]> readStudentByCourseID(int courseID){
    ArrayList<String[]> list = new ArrayList<>();
    try {
        String query = "SELECT PersonID , LastName , Firstname , Grade FROM `person`, `studentgrade` "
                + "WHERE PersonID = StudentID and CourseID = '"+courseID+"'";
        ResultSet rs = this.doReadQuery( sql: query);
        if(rs !=null){
            while(rs.next()){
                String PersonID = rs.getString( columnLabel: "PersonID");
                String FullName = rs.getString( columnLabel: "Lastname")+" "+rs.getString( columnLabel: "Firstname");
                String grade = rs.getString( columnLabel: "Grade");
                System.out.println( x: FullName);
                String[] s = {PersonID,FullName,grade};
                list.add( e: s);
            }
        }
        return list;
    } catch (Exception e) {
    }
    return list;
}
public ArrayList<String[]> readStudentByCourseTitle(String courseTitle){
     ArrayList<String[]> list = new ArrayList<>();
    try {
        //fix
        String query = "SELECT PersonID , LastName , Firstname , Grade FROM `person`, `studentgrade` "
                + "WHERE PersonID = StudentID and CourseID = '"+courseTitle+"'";
        ResultSet rs = this.doReadQuery( sql: query);
        if(rs !=null){
            while(rs.next()){
                String PersonID = rs.getString( columnLabel: "PersonID");
                String FullName = rs.getString( columnLabel: "Lastname")+" "+rs.getString( columnLabel: "Firstname");
                String grade = rs.getString( columnLabel: "Grade");
                System.out.println( x: FullName);
                String[] s = {PersonID,FullName,grade};
                list.add( e: s);
            }
        }
        return list;
    } catch (Exception e) {
    }
    return list;
}
```

8.2.2. BLL

```
 public ArrayList<String[]> readStudentByCourseIdBll(String courseID){
     int courseid = Integer.parseInt( s: courseID);
     StudentDAL std = new StudentDAL();
     ArrayList<String[]> list = std.readStudentByCourseID( courseID: courseid);
     return list;
 }

 public ArrayList<String[]> readStudentByCourseTitleBll(String courseTitle){
     StudentDAL std = new StudentDAL();
     ArrayList<String[]> list = std.readStudentByCourseTitle(courseTitle);
     return list;
 }
}
```

8.2.3. UI

```
private void btndetail3ActionPerformed(java.awt.event.ActionEvent evt) {
    StudentBLL bll = new StudentBLL();
    String courseid = cbdetailID.getSelectedItem().toString();
    ArrayList<String[]> ds = bll.readStudentByCourseIdBll( courseID: courseid);
    model.setRowCount( rowCount: 0);
    int count = 0 ;
    System.out.println( x: ds.size());
    if(ds.size() > 0){
        for(String[] s : ds){
            Vector row=new Vector();
            row.add( e: count);
            row.add(s[0]);
            row.add(s[1]);
            row.add(s[2]);
            model.addRow( rowData: row);
            count++;
        }
    }
    tb3.setModel( dataModel: model);
}
```

```
private void btndetail4ActionPerformed(java.awt.event.ActionEvent evt) {
    StudentBLL bll = new StudentBLL();
    String courseid = cbdetailName.getSelectedItem().toString();
    ArrayList<String[]> ds = bll.readStudentByCourseTitleBll( courseTitle: courseid);
    model.setRowCount( rowCount: 0);
    int count = 0 ;
    System.out.println( x: ds.size());
    if(ds.size() > 0){
        for(String[] s : ds){
            Vector row=new Vector();
            row.add( e: count);
            row.add(s[0]);
            row.add(s[1]);
            row.add(s[2]);
            model.addRow( rowData: row);
            count++;
        }
    }
    tb3.setModel( dataModel: model);
}
```

HẾT