

ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN

------



## BÁO CÁO

**MÔN: XÂY DỰNG PHẦN MỀM PHÂN LỚP**

**TÊN ĐỀ TÀI: PHẦN MỀM QUẢN LÝ KHÓA HỌC**

Người thực hiện tiểu luận

MSSV	Họ Tên

**GIẢNG VIÊN ĐÁNH GIÁ:**

**TP. HCM, tháng 10/2022**

# MỤC LỤC

<b>Chương 1. Bảng phân công nhiệm vụ .....</b>	<b>3</b>
<b>    1.1. GIỚI THIỆU ĐỀ TÀI .....</b>	<b>4</b>
<b>Chương 2. CÁC CHỨC NĂNG QUẢN LÝ QUAN TRỌNG .....</b>	<b>5</b>
<b>    2.1. Chức năng Quản Lý Giảng viên – Học viên .....</b>	<b>5</b>
2.1.1. <i>Sơ đồ Class Chung</i> .....	5
2.1.2. <i>Quản lý giáo viên</i> .....	6
2.1.3. <i>Quản lý Học viên</i> .....	17
<b>    2.2. Chức năng Quản lý khóa học .....</b>	<b>29</b>
2.2.1. <i>Sơ đồ Class Chung</i> .....	29
2.2.2. <i>Xử lý 1: Hiển thị danh sách khóa học</i> .....	30
2.2.3. <i>Xử lý 2: Thêm khóa học</i> .....	33
2.2.4. <i>Xử lý 3: Cập nhật thông tin khóa học</i> .....	35
2.2.5. <i>Xử lý 4: Xóa khóa học</i> .....	37
2.2.6. <i>Xử lý 5: Tìm kiếm khóa học</i> .....	39
2.2.7. <i>Xử lý 6: Hiện thị mã phòng (combobox)</i> .....	41
2.2.8. <i>Xử lý 7: Hiển thị 1 dòng (record)</i> .....	44
<b>    2.3. Chức năng phân công giảng dạy .....</b>	<b>44</b>
2.3.1. <i>Sơ đồ class chung</i> .....	44
2.3.2. <i>Xử lý 1: Hiển thị danh sách phân công giảng dạy</i> .....	47
2.3.3. <i>Xử lý 2: Thêm phân công giảng dạy</i> .....	48
2.3.4. <i>Xử lý 3: Sửa phân công giảng dạy</i> .....	49
2.3.5. <i>Xử lý 4: Xóa phân công giảng dạy</i> .....	51
2.3.6. <i>Xử lý 5: Lọc &amp; Tìm kiếm phân công giảng dạy</i> .....	52
2.3.7. <i>Xử lý 6: Hiển thị 1 dòng (record)</i> .....	55
2.3.8. <i>Xử lý 7: Hiển thị bảng chọn nhanh (1 cột)</i> .....	56
<b>    2.4. Chức Năng Quản Lý Kết Quả Khoa Học .....</b>	<b>61</b>
2.4.1. <i>Sơ đồ class chung</i> .....	61
2.4.2. <i>Xử lý 1: Hiển thị danh sách kết quả khoa học</i> .....	62
2.4.3. <i>Xử lý 2: Thêm kết quả khoa học</i> .....	64
2.4.4. <i>Xử lý 3: Sửa kết quả khoa học</i> .....	66
2.4.5. <i>Xử lý 4: Xóa kết quả khoa học</i> .....	68
2.4.6. <i>Xử lý 5: Lọc và tìm kiếm kết quả khoa học</i> .....	70

2.4.7. Xử lý 6: Hiển thị 1 dòng (record).....	72
2.4.8. Xử lý 7: Hiển thị bảng chọn nhanh.....	73

## **Chương 3. SOURCE CODE KẾT NỐI CSDL, HƯỚNG DẪN CÀI ĐẶT CHƯƠNG TRÌNH VÀ LINK CHÚA SOURCE CODE ĐỒ ÁN ..... 77**

3.1. Hướng dẫn cài đặt chương trình.....	77
--	----

3.2. Link Chứa Source Code Đồ Án .....	77
--	----

## Chương 1. Bảng phân công nhiệm vụ

Tên người thực hiện	Công việc thực hiện
Nguyễn Đức Minh Trung	Thực hiện tạo vẽ sơ đồ tuần tự Xây dựng chức năng phân công giảng dạy
Hồ Tân Thuận	Thực hiện tạo vẽ sơ đồ tuần tự Xây dựng chức năng quản lý giảng viên, học viên
Trần Kim Phú	Thực hiện tạo vẽ sơ đồ tuần tự Xây dựng chức năng quản lý kết quả khóa học
Minh Hiếu Calan Tog	Thực hiện tạo vẽ sơ đồ tuần tự Xây dựng chức quản lý khóa học
Võ Hoàng Quỳnh Như	Viết báo cáo Xây dựng chức năng quản lý khóa học

## 1.1.GIỚI THIỆU ĐỀ TÀI

### 1. Giới thiệu đề tài:

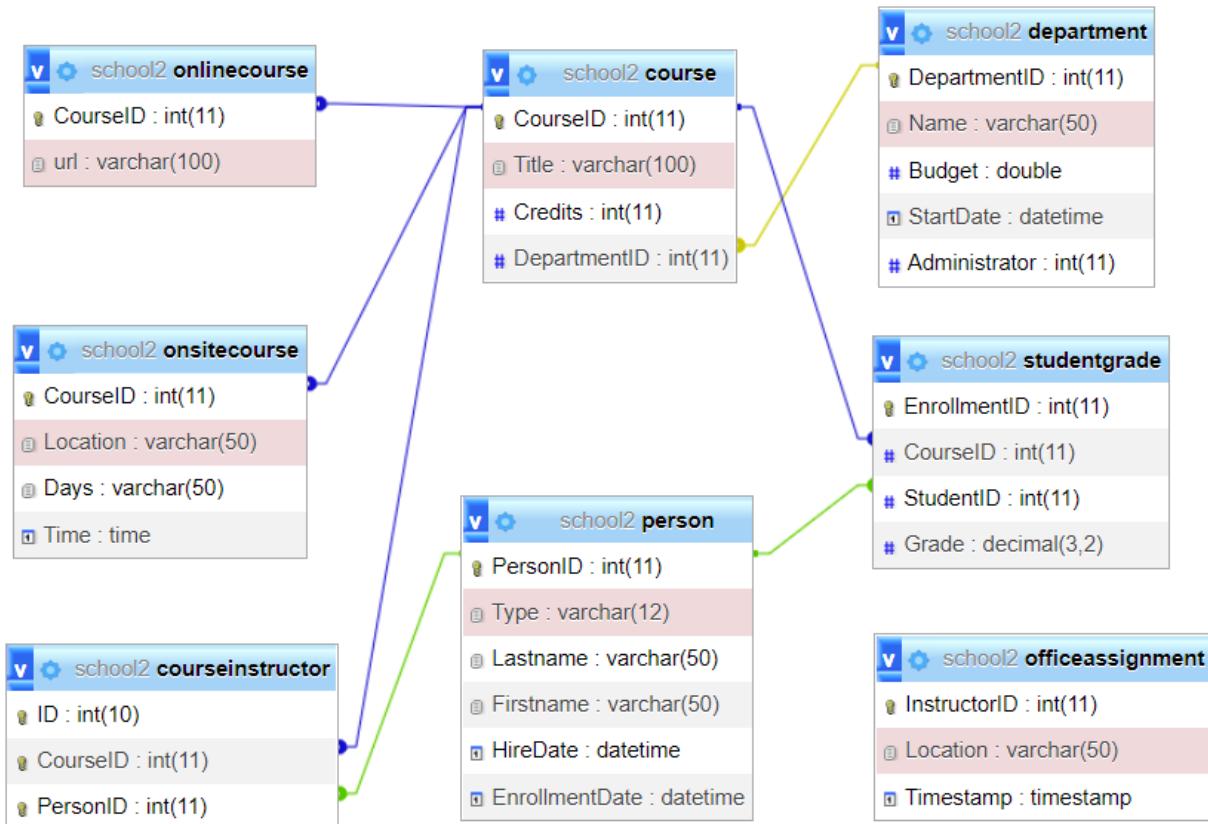
Tên đề tài: Phần mềm quản lý khóa học.

Mô tả:

Ứng dụng phần mềm vào việc quản lý khóa học là một nhu cầu tất yếu nhằm nâng cao hiệu quả quản lý và phân bổ thời gian cho các khóa học, giúp học viên và giảng viên giảm bớt thời gian lên lịch dạy và học, hạn chế được về vấn đề trùng giờ. Đồng thời mang lại quản lý được lượng thông tin nhanh chóng và hiệu quả nhất, tiết kiệm được chi phí quản lý giấy tờ sổ sách và các yếu tố khác như lỗi, mệt hoặc mất mát tài liệu. Thế nên việc dùng phần mềm trong quản lý sẽ giúp đỡ rất nhiều cho người quản lý chỉ bằng vài thao tác xử lý trên hệ thống sẽ kiểm soát được mọi thông tin cần thiết và tra cứu thông tin một cách nhanh chóng nhất, tiện lợi nhất.

Chính vì những lí do đó, nhóm chúng em đã triển khai một phần mềm quản lý hỗ trợ quản lý khóa học.

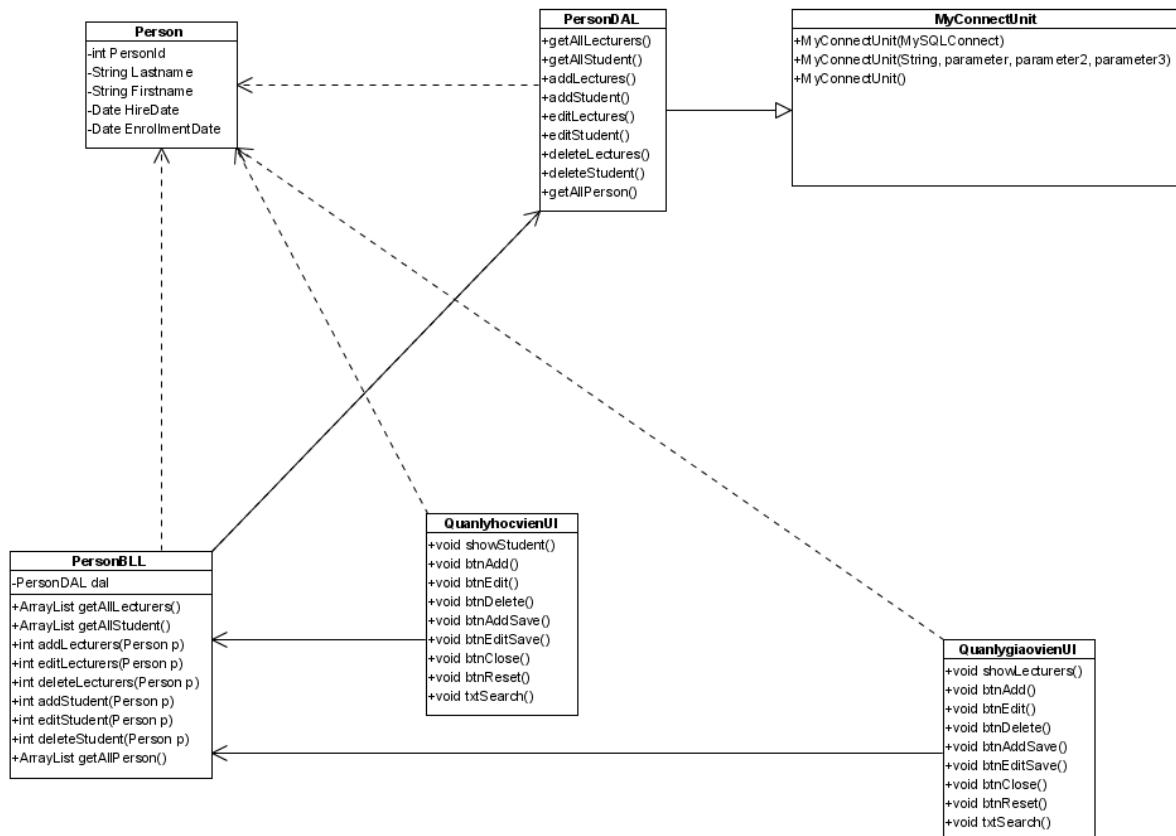
### 2. Mô hình cơ sở dữ liệu ở mức cài đặt:



## Chương 2. CÁC CHỨC NĂNG QUẢN LÝ QUAN TRỌNG

### 2.1. Chức năng Quản Lý Giảng viên – Học viên

#### 2.1.1. Sơ đồ Class Chung



##### 2.1.1.1. Constructor PersonDAL

```
public class PersonDAL extends MyConnectUnit{
    ArrayList<Person> listPersonDTOs;
    public PersonDAL() {
        super();
    }
}
```

##### 2.1.1.2. Constructor PersonBLL

```
public class PersonBLL {
    static ArrayList<Person> listPerson;
    static ArrayList<Person> listPersonStudent;
    static ArrayList<Person> listPersonLecturers;
    private PersonDAL data = new PersonDAL();
    private ArrayList<CourseInstructorDTO> courseIst;
```

### 2.1.1.3. Contructor UI QuanLyHocVien

```
27  public class QuanLyHocVien extends javax.swing.JPanel {  
28  
29      private int DEFALUT_WIDTH;  
30      private PersonBLL personBLL = new PersonBLL();  
31      DefaultTableModel model;  
32      static ArrayList<Person> listLecturers = new PersonBLL().getListPerson();  
33      Person studentDTO = new Person();  
34      String formatTime = "yyyy-MM-dd      ";  
35      DateFormat fm = new SimpleDateFormat(formatTime);  
36  
```

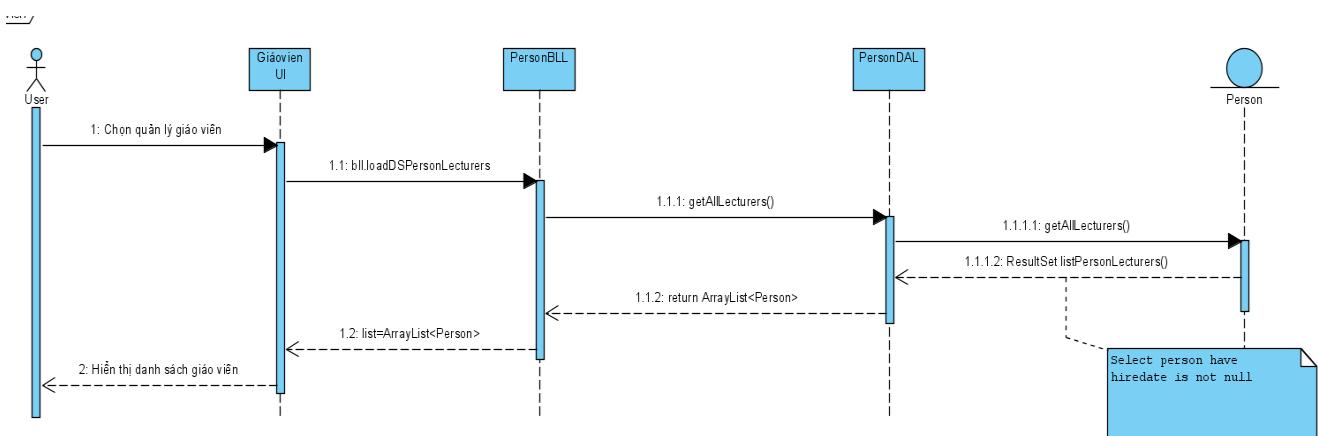
### 2.1.1.4. Contructor UI QuanLyGiaoVien

```
27  public class QuanLyGiaoVien extends javax.swing.JPanel {  
28  
29      private int DEFALUT_WIDTH;  
30      private PersonBLL personBUS = new PersonBLL();  
31      Person lecturerDTO = new Person();  
32      DefaultTableModel model;  
33      static ArrayList<Person> listLecturers = new PersonBLL().getListPerson();  
34      String formatTime = "yyyy-MM-dd      ";  
35      DateFormat fm = new SimpleDateFormat(formatTime);  
36  
```

## 2.1.2. Quản lý giáo viên

### 2.1.2.1. Xử lý 1: Hiển thị danh sách giảng viên

#### ❖ Sơ đồ tuần tự



- Code class DAL

```
91     public ArrayList<Person> getAllLecturers() throws Exception {
92         listPersonDTOs = new ArrayList<>();
93         try {
94
95             ResultSet rs = this.SelectCustom("person as ps", "ps.PersonID,"
96                     + "ps.Lastname, ps.Firstname, ps.HireDate, ps.EnrollmentDate"
97                     + "ps.HireDate IS NOT NULL ORDER BY PersonID DESC");
98             while(rs.next())
99             {
100                 Person psDTO=new Person(
101                     rs.getInt("PersonID"),
102                     rs.getString("Lastname"),
103                     rs.getString("Firstname"),
104                     rs.getDate("HireDate"),
105                     rs.getDate("EnrollmentDate")
106                 );
107                 listPersonDTOs.add(psDTO);
108             }
109             rs.close();
110             this.Close(); //dong ket noi;
111
112         } catch (SQLException ex) {
113             System.out.println("Khong the load database Person: "+ex);
114         }
115
116         return listPersonDTOs;
117     }
118 }
```

- Code class BLL

```
118     // Giao vien
119     public void loadDSPersonLecturers() throws Exception {
120
121         if (listPersonLecturers == null) {
122             listPersonLecturers = new ArrayList<Person>();
123         }
124         listPersonLecturers = data.getAllLecturers();
125     }
126 }
```

- Code class UI

```

89     private void outModel(DefaultTableModel model, ArrayList<Person> personDTO)
90     {
91         Vector data;
92         model.setRowCount(0);
93
94         int i = 1;
95         for (Person cs : personDTO) {
96             data = new Vector();
97             data.add(i);
98             data.add(cs.getPersonID());
99             data.add(cs.getFirstname());
100            data.add(cs.getLastname());
101            data.add(fm.format(cs.getHireDate()));
102
103            model.addRow(data);
104            i++;
105        }
106        tbl_giangvien.setModel(model);
107    }
108

```

```

75     private void insertHeader() {
76         Vector header = new Vector();
77         header.add(e: "STT");
78         header.add(e: "Mã giáo viên");
79         header.add(e: "Họ");
80         header.add(e: "Tên");
81         header.add(e: "Ngày thuê");
82         //if (model.getRowCount() == 0)
83         model = new DefaultTableModel(columnNames: header, rowCount: 0);
84     }
85

```

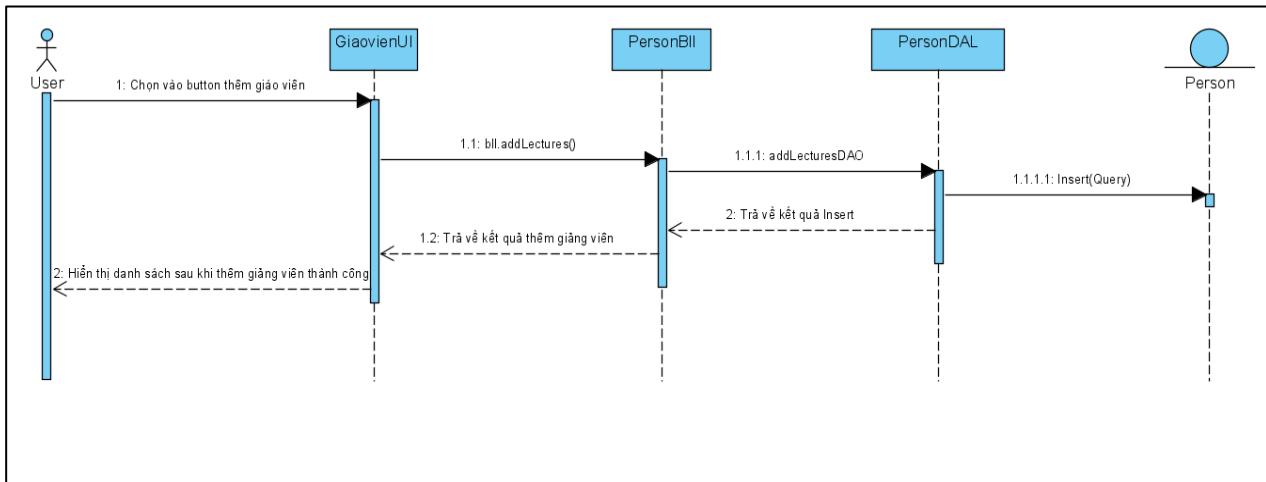
```

61     private void ShowLecturers() throws Exception {
62
63         try {
64
65             if (PersonBLL.getListPersonLecturers() == null) {
66                 personBLL.loadDSPersonLecturers();
67             }
68             insertHeader();
69             outModel(model, PersonBLL.getListPersonLecturers());
70         } catch (Exception e) {
71             JOptionPane.showMessageDialog(this, "Không Thể Load Database",
72                                     "Thông Báo Lỗi", JOptionPane.ERROR_MESSAGE);
73         }
74     }
75 }
76

```

### 2.1.2.2. Xử lý 2: Thêm giáo viên

- Sơ đồ tuần tự



#### ▪ Code class DAL

```

121  public void addLectures(Person personDTO) throws Exception {
122      DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
123      HashMap<String, Object> objMap = new HashMap<String, Object>();
124      objMap.put("PersonID", personDTO.getPersonID());
125      objMap.put("Lastname", personDTO.getLastname());
126      objMap.put("Firstname", personDTO.getFirstname());
127      objMap.put("HireDate", dateFormat.format(personDTO.getHireDate()));
128
129      try {
130          this.Insert("person", objMap);
131      } catch (Exception e) {
132          System.out.println("Không thể thêm giảng viên mới. Vui lòng kiểm tra lại ");
133      }
134  }
135
  
```

#### ▪ Code class BLL

```

131  public void addLectures(Person ps) throws Exception {
132      // validate data
133
134      data.addLectures(ps);
135      listPersonLecturers.add(ps);
136
137  }
138
  
```

#### ▪ Code class UI

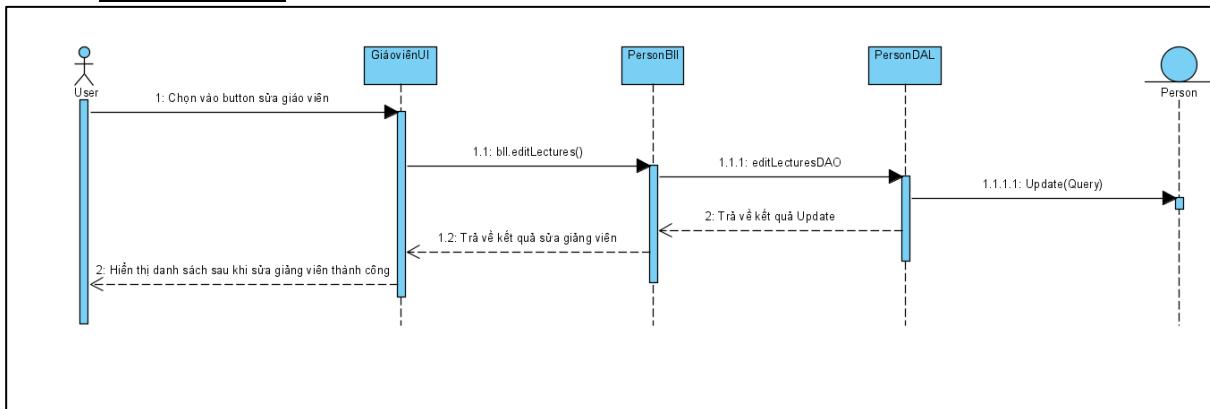
```

539     private void btnSave1ActionPerformed(java.awt.event.ActionEvent evt) {
540         // TODO add your handling code here:
541
542         try {
543
544             if (txtId_Lecturers.getText().length() != 0
545                 && txtHireDateGV.getDate() != null
546                 && txtLastNameGV.getText().length() != 0
547                 && txtFirstNameGV.getText().length() != 0)//xem thử đã nhập hay chưa(textflied)
548             {
549                 lecturerDTO.setPersonID(Integer.valueOf(txtId_Lecturers.getText()));
550                 lecturerDTO.setFirstname(txtFirstNameGV.getText());
551                 lecturerDTO.setLastname(txtLastNameGV.getText());
552                 lecturerDTO.setHireDate(txtHireDateGV.getDate());
553                 personBLL.addLectures(lecturerDTO);
554                 personBLL.loadDSPersonLecturers();
555                 add_Lecturers.dispose();
556                 ShowLecturers();
557                 txtFirstNameGV.setText("");
558                 txtLastNameGV.setText("");
559                 txtHireDateGV.setDate(null);
560
561             } else {
562                 JOptionPane.showMessageDialog(null, "vui lòng nhập đầy đủ thông tin");
563             }
564         } catch (Exception e) {
565             JOptionPane.showMessageDialog(null, "Failed!");
566             e.printStackTrace();
567         }
568     }

```

### 2.1.2.3. Xử lý 3: Sửa giáo viên

- Sơ đồ tuần tự



- Code class DAL

```

136     // Sửa
137     public void editLectures(Person person) throws Exception {
138         DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
139         HashMap<String, Object> objMap = new HashMap<String, Object>();
140         objMap.put("PersonID", person.getPersonID());
141         objMap.put("Lastname", person.getLastname());
142         objMap.put("Firstname", person.getFirstname());
143         objMap.put("HireDate", dateFormat.format(person.getHireDate()));
144
145         try {
146             this.Update("person", objMap, "PersonID='"+person.getPersonID()+"'");
147         } catch (Exception e) {
148             System.out.println("Không thể thêm giảng viên mới . Vui lòng kiểm tra lại ");
149         }
150     }
151 }

```

- Code class BLL

```

138
139     public void editLectures(Person ps) throws Exception {
140         // validate data
141
142         data.editLectures(ps);
143         listPersonLecturers.add(ps);
144
145     }
146

```

- Code class UI

```

500     private void btn_editLecturersActionPerformed(java.awt.event.ActionEvent evt) {
501         // TODO add your handling code here:
502         int i = tbl_giangvien.getSelectedRow();
503         if (i == -1) {
504             JOptionPane.showMessageDialog(null, "Vui lòng chọn dòng cần sửa");
505         } else {
506             txtEditId_Lecturers.setText(String.valueOf(tbl_giangvien.getValueAt(i, 1)));
507             txtEditFirstNameGV.setText(String.valueOf(tbl_giangvien.getValueAt(i, 2)));
508             txtEditLastNameGV.setText(String.valueOf(tbl_giangvien.getValueAt(i, 3)));
509
510             String date = (String) tbl_giangvien.getValueAt(i, 4);
511
512             try {
513                 txtEditHireDateGV.setDate((Date) fm.parse(date));
514             } catch (ParseException ex) {
515                 Logger.getLogger(QuanLyGiaoVien.class.getName()).log(Level.SEVERE, null, ex);
516             }
517
518             edit_Lecturers.setVisible(true);
519         }
520     }

```

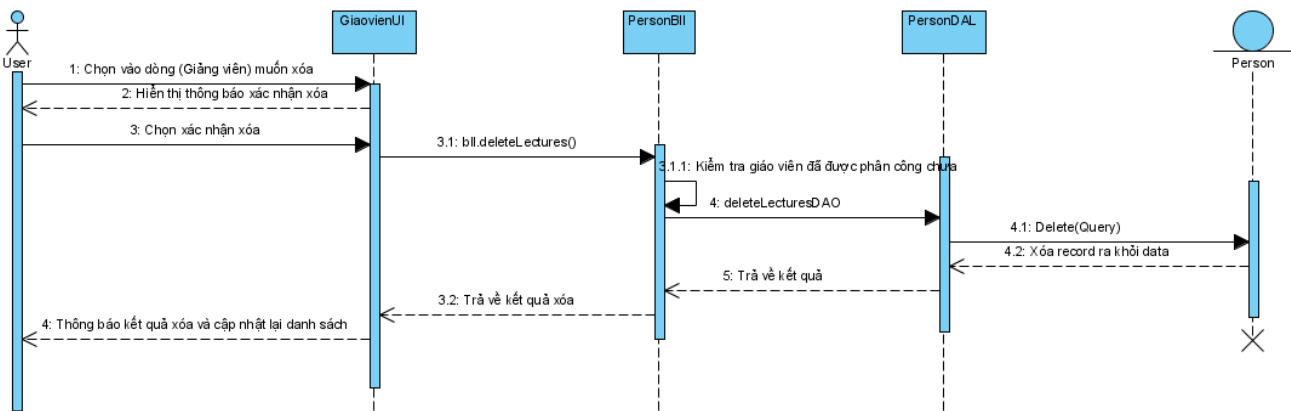
```

535     private void btnSaveEditLecturersActionPerformed(java.awt.event.ActionEvent evt) {
536         // TODO add your handling code here:
537         try {
538
539             if (txtEditId_Lecturers.getText().length() != 0
540                 && txtEditHireDateGV.getDate() != null
541                 && txtEditLastNameGV.getText().length() != 0
542                 && txtEditFirstNameGV.getText().length() != 0)//xem thử đã nhập hay chưa(textfield)
543             {
544                 lecturerDTO.setPersonID(Integer.valueOf(txtEditId_Lecturers.getText()));
545                 lecturerDTO.setFirstname(txtEditFirstNameGV.getText());
546                 lecturerDTO.setLastname(txtEditLastNameGV.getText());
547                 lecturerDTO.setHireDate(txtEditHireDateGV.getDate());
548                 personBLL.editLectures(lecturerDTO);
549                 personBLL.loadDSPersonLecturers();
550                 JOptionPane.showMessageDialog(null, "Sửa giảng viên thành công !");
551                 edit_Lecturers.dispose();
552                 ShowLecturers();
553
554             } else {
555                 JOptionPane.showMessageDialog(null, "Vui lòng nhập đầy đủ thông tin");
556             }
557         } catch (Exception e) {
558             JOptionPane.showMessageDialog(null, "Failed!");
559             e.printStackTrace();
560         }
561     }

```

#### 2.1.2.4. Xử lý 4: Xóa giáo viên

- Sơ đồ tuần tự



#### Code class DAL

```

153     // Xóa
154     public void deleteLectures(int PersonID)
155     {
156         try {
157             this.Delete("person", "PersonID ='" + PersonID + "'");
158         } catch (Exception e) {
159             System.out.println("Lỗi không thể xóa giảng viên !");
160         }
161     }
162 }
163 // Hỗn loạn

```

#### Code class BLL

```

157     public void deleteLectures(int PersonID) throws Exception {
158
159         for (Person ps : listPersonLecturers) {
160             if (ps.getPersonID() == PersonID) {
161                 for (CourseInstructorDTO cs : courseIst) {
162                     try {
163                         if (cs.getCourseID() != PersonID) {
164                             listPersonLecturers.remove(ps);
165                             data.deleteLectures(PersonID);
166                         }
167                     } catch (Exception e) {
168                         System.out.println("Giáo viên đang được phân công không được xóa !!!");
169                     }
170                 }
171             }
172         }
173         return;
174     }
175 }
176
177 }
178

```

#### Code class UI

```

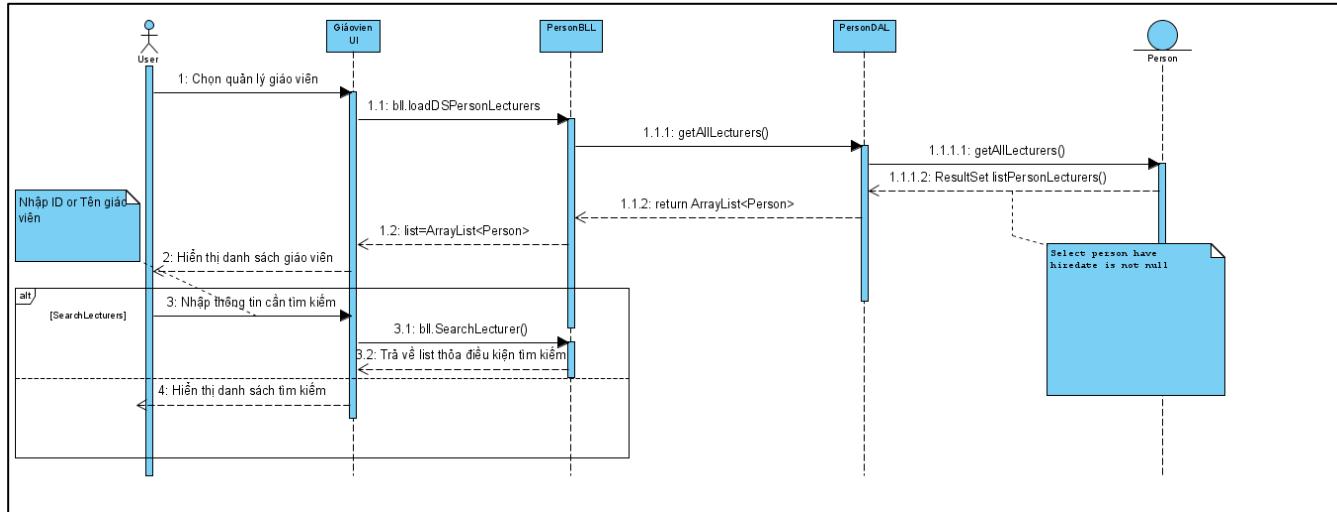
private void btn_deleteLecturersActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int i = tbl_giangvien.getSelectedRow();
    if (i == -1) {
        JOptionPane.showMessageDialog(null, "Vui lòng chọn dòng cần xóa");
    } else if (JOptionPane.showConfirmDialog(this, "Bạn chắc chắn muốn xóa giảng viên này?", "Confirm", JOptionPane.YES_OPTION, JOptionPane.NO_OPTION) == JOptionPane.YES_OPTION) {
        int select = (int) tbl_giangvien.getModel().getValueAt(i, 1);

        lecturerDTO.setPersonID(select);
        try {
            personBLL.deleteLectures(lecturerDTO.getPersonID());
            showMessageDialog(null, "Xóa thành công");
            personBLL.loadDSPersonLecturers();
            ShowLecturers();
        } catch (Exception e) {
            showMessageDialog(null, "Giáo viên đang được phân công không được xóa !!!");
        }
    }
}

```

### 2.1.2.5. Xử lý 5: Tìm kiếm giáo viên

- Sơ đồ tuần tự



- Code class BLL

```

66     public ArrayList<Person> searchPersonWithID(int personID) {
67         ArrayList<Person> search = new ArrayList<>();
68         //personID=personID.isEmpty() ? personID="" : personID;
69
70         for (Person ps : listPerson) {
71
72             if (ps.getPersonID() == personID) {
73
74                 search.add(ps);
75             }
76         }
77     }
78
79     return search;
}

```

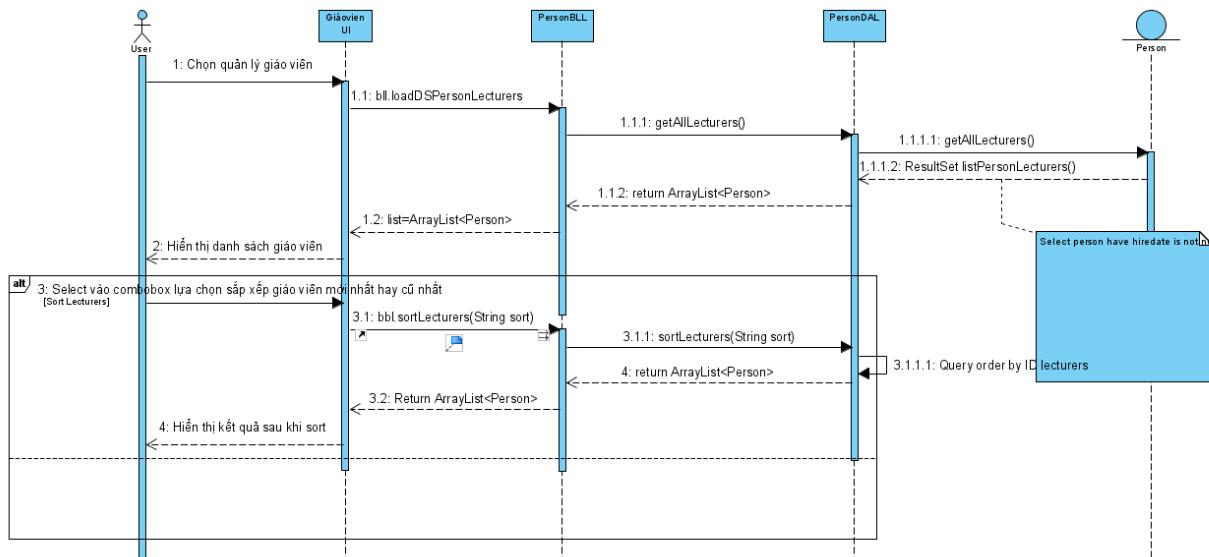
```
81     public ArrayList<Person> searchPersonWithFirstName(String firstName) {  
82  
83         firstName = firstName.isEmpty() ? firstName = "" : firstName;  
84  
85         for (Person ps : listPerson) {  
86  
87             if (ps.getFirstrname().contains(firstName)) {  
88  
89                 listPerson.add(ps);  
90             }  
91         }  
92         return listPerson;  
93     }  
94  
95     public ArrayList<Person> searchPersonWithLastName(String lastName) {  
96  
97         lastName = lastName.isEmpty() ? lastName = "" : lastName;  
98  
99         for (Person ps : listPerson) {  
100             if (ps.getLastname().contains(lastName)) {  
101                 listPerson.add(ps);  
102             }  
103         }  
104         return listPerson;  
105     }  
106 }  
107 }  
108 }  
109 }
```

- Code class UI

```
691     private void txFind_GVKeyPressed(java.awt.event.KeyEvent evt) {  
692         // TODO add your handling code here:  
693         TableRowSorter<DefaultTableModel> tr = new TableRowSorter<DefaultTableModel>(model);  
694         tbl_giangvien.setRowSorter(tr);  
695         tr.setRowFilter(RowFilter.regexFilter(txFind_GV.getText().trim()));  
696     }  
697 }  
698 }
```

### 2.1.2.6. Xử lý 6: Sort giáo viên theo ngày thêm mới nhất và cũ nhất

- **Sơ đồ tuần tự**



- **Code class DAL**

```

public ArrayList<Person> sortLecturers(String value) throws Exception {
    listPersonDTOs = new ArrayList<>();
    try {
        ResultSet rs = this.SelectCustom("person as ps", "ps.PersonID, "
            + "ps.Lastname, ps.Firstname, ps.HireDate, ps.EnrollmentDate"
            + "ps.HireDate IS NOT NULL ORDER BY PersonID" + value);
        while(rs.next())
        {
            Person psDTO=new Person(
                rs.getInt("PersonID"),
                rs.getString("Lastname"),
                rs.getString("Firstname"),
                rs.getDate("HireDate"),
                rs.getDate("EnrollmentDate")
            );
            listPersonDTOs.add(psDTO);
        }
        rs.close();
        this.Close(); //dong ket noi;
    } catch (SQLException ex) {
        System.out.println("Khong the load database Person: " + ex);
    }
    return listPersonDTOs;
}
  
```

- **Code class BLL**

```

public ArrayList<Person> sortLecturers(String value) throws Exception {
    ArrayList<Person> listAllPerson = new ArrayList<Person>();
    listAllPerson = data.sortLecturers(value);
    return listAllPerson;
}
  
```

- **Code class UI**

```

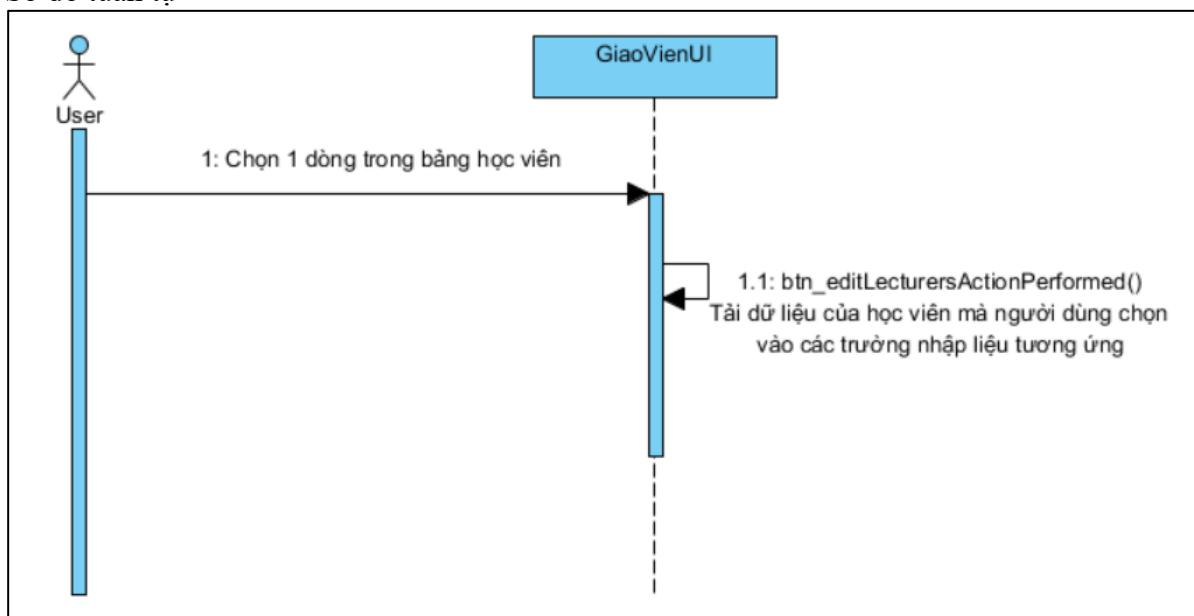
private void SortLecturersActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String s = (String) SortLecturers.getSelectedItem();

    if (s.equals("DESC")) {
        try {
            listLecturers=personBLL.sortLecturers("DESC");
        } catch (Exception ex) {
            Logger.getLogger(QuanLyGiaoVien.class.getName()).log(Level.SEVERE, null, ex);
        }
        outModel(model, listLecturers);
    } else if (s.equals("ASC")) {
        try {
            listLecturers=personBLL.sortLecturers("ASC");
        } catch (Exception ex) {
            Logger.getLogger(QuanLyGiaoVien.class.getName()).log(Level.SEVERE, null, ex);
        }
        outModel(model, listLecturers);
    }
}

```

### 2.1.2.7. Xử lý 7: Hiển thị chi tiết 1 dòng (record)

Sơ đồ tuần tự



Code class UI

```

private void btn_editLecturersActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int i = tbl_giangvien.getSelectedRow();
    System.out.println((Date)tbl_giangvien.getValueAt(i, 4));
    if (i == -1) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Vui lòng chọn dòng cần sửa");
    } else {
        txtEditId_Lecturers.setText(t: String.valueOf(obj:tbl_giangvien.getValueAt(row: i, column: 1)));
        txtEditFirstNameGV.setText(t: String.valueOf(obj:tbl_giangvien.getValueAt(row: i, column: 2)));
        txtEditLastNameGV.setText(t: String.valueOf(obj:tbl_giangvien.getValueAt(row: i, column: 3)));

        String date = (String)tbl_giangvien.getValueAt(row: i, column: 4);
        try {
            txtEditHireDateGV.setDate((Date)fm.parse(source: date));
        } catch (ParseException ex) {
            Logger.getLogger(name:QuanLyGiaoVien.class.getName()).log(level:Level.SEVERE, msg: null, thrown:ex);
        }
        edit_Lecturers.setVisible(b: true);
    }
}

```

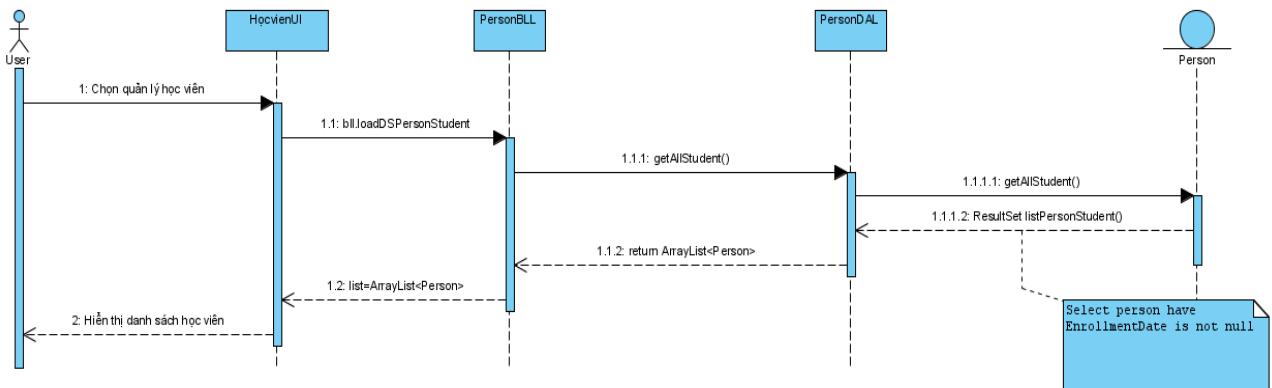
UI:

	Tên	Ngày thuê
1	Van Houten	2000-12-07
2	Xu	2001-07-23
3	Stewart	1997-10-09
4	Serrano	1999-06-01
5	Kapoor	2001-01-15
6	Zheng	2004-02-12
7	Harui	1998-07-01
8	Fakhouri	2002-08-06

### 2.1.3. Quản lý Học viên

#### 2.1.3.1. Xử lý 1: Hiển thị danh sách học viên

- Sơ đồ tuần tự



#### ▪ Code class DAL

```

165     public ArrayList<Person> getAllStudent() throws Exception {
166         listPersonDTOs = new ArrayList<>();
167         try {
168             ResultSet rs = this.SelectCustom("person as ps", "ps.PersonID,ps.Lastname,"
169                     + "ps.Firstname,ps.HireDate,"
170                     + "ps.EnrollmentDate",
171                     + "ps.EnrollmentDate IS NOT NULL ORDER BY PersonID DESC");
172             while(rs.next())
173             {
174                 Person psDTO=new Person(
175                     rs.getInt("PersonID"),
176                     rs.getString("Lastname"),
177                     rs.getString("Firstname"),
178                     rs.getDate("HireDate"),
179                     rs.getDate("EnrollmentDate")
180                 );
181                 listPersonDTOs.add(psDTO);
182             }
183             rs.close();
184             this.Close(); //dong ket noi;
185         } catch (SQLException ex) {
186             System.out.println("Khong the load database Person: "+ex);
187         }
188         return listPersonDTOs;
189     }
190 }
191
192 }
193

```

#### ▪ Code class BLL

```

165     public void loadDSPersonStudent() throws Exception {
166
167         if (listPersonStudent == null) {
168             listPersonStudent = new ArrayList<Person>();
169         }
170         listPersonStudent = data.getAllStudent();
171     }
172

```

- Code class UI

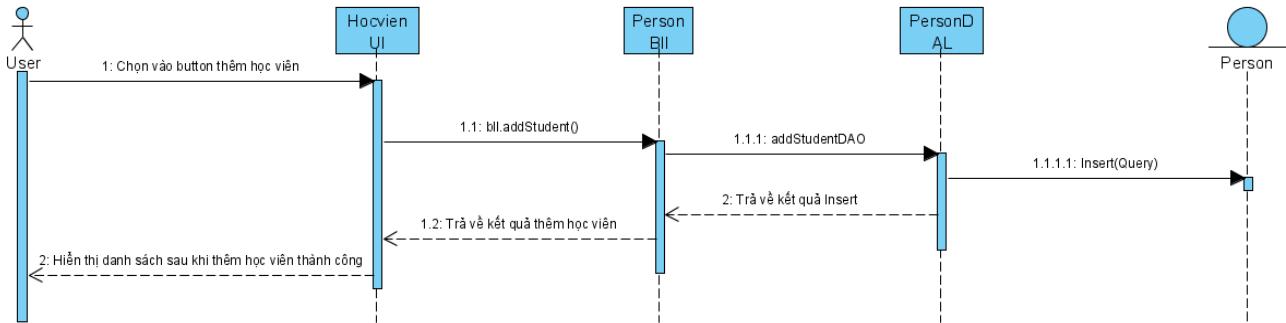
```
58     private void ShowStudent() throws Exception {
59
60         try {
61
62             if (PersonBLL.getListPersonStudent() == null) {
63                 personBLL.loadDSPersonStudent();
64             }
65             insertHeader();
66             outModel(model, PersonBLL.getListPersonStudent());
67         } catch (Exception e) {
68             JOptionPane.showMessageDialog(this, "Không Thể Load Database ", "Thông Báo Lỗi",
69             JOptionPane.ERROR_MESSAGE);
70         }
71     }
72 }
73 }
```

```
36     private void outModel(DefaultTableModel model, ArrayList<Person> personDTO)
37     {
38         Vector data;
39         model.setRowCount(0);
40
41         int i = 1;
42         for (Person cs : personDTO) {
43             data = new Vector();
44             data.add(i);
45             data.add(cs.getPersonID());
46             data.add(cs.getFirstname());
47             data.add(cs.getLastname());
48             data.add(fm.format(cs.getEnrollmentDate()));
49
50             model.addRow(data);
51             i++;
52         }
53         tbl_hocvien.setModel(model);
54     }
55 }
```

```
private void insertHeader() {  
    Vector header = new Vector();  
    header.add( e:"STT");  
    header.add( e:"Mã học viên");  
    header.add( e:"Họ");  
    header.add( e:"Tên");  
    header.add( e:"Ngày đăng ký");  
    model = new DefaultTableModel( columnNames:header, rowCount:0 );  
}
```

### 2.1.3.2. Xử lý 2: Thêm học viên

- #### ▪ *Sơ đồ tuần tự*



- Code class DAL

```

197     public void addstudent(Person person) throws Exception {
198         DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
199         HashMap<String, Object> objMap = new HashMap<String, Object>();
200         objMap.put("PersonID", person.getPersonID());
201         objMap.put("Lastname", person.getLastname());
202         objMap.put("Firstname", person.getFirstname());
203         objMap.put("EnrollmentDate", dateFormat.format(person.getEnrollmentDate()));
204
205         try {
206             this.Insert("person", objMap);
207         } catch (Exception e) {
208             System.out.println("Không thể thêm học viên mới . Vui lòng kiểm tra lại ");
209         }
210     }
  
```

- Code class BLL

```

177     public void addStudent(Person ps) throws Exception {
178         // validate data
179         data.addStudent(ps);
180         listPersonStudent.add(ps);
181
182     }
  
```

- Code class UI

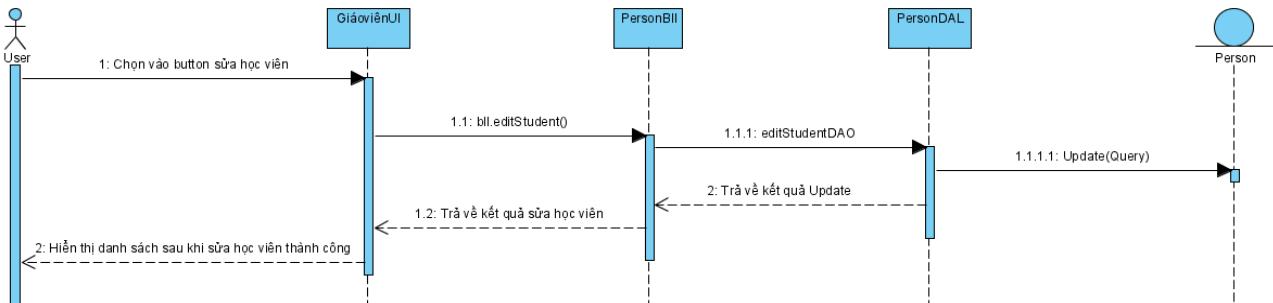
```

private void btnSaveHVActionPerformed(java.awt.event.ActionEvent evt) {
627     // TODO add your handling code here:
628     try {
629         if (txtId_Student.getText().length() != 0
630             && txtEnrollmentDateHV.getDate() != null
631             && txtLastNameHV.getText().length() != 0
632             && txtFirstNameHV.getText().length() != 0)//xem thử đã nhập hay chưa(textflied)
633         {
634             studentDTO.setPersonID(Integer.valueOf(txtId_Student.getText()));
635             studentDTO.setFirstname(txtFirstNameHV.getText());
636             studentDTO.setLastname(txtLastNameHV.getText());
637             studentDTO.setEnrollmentDate(txtEnrollmentDateHV.getDate());
638             personBLL.addStudent(studentDTO);
639             personBLL.loadDSPersonStudent();
640             JOptionPane.showMessageDialog(null, "Thêm học viên thành công !");
641             add_Student.dispose();
642             ShowStudent();
643             txtFirstNameHV.setText("");
644             txtLastNameHV.setText("");
645             txtEnrollmentDateHV.setDate(null);
646         } else {
647             JOptionPane.showMessageDialog(null, "vui lòng nhập đầy đủ thông tin");
648         }
649     } catch (Exception e) {
650         JOptionPane.showMessageDialog(null, "Failed!");
651         e.printStackTrace();
652     }
653 }
654

```

### 2.1.3.3. Xử lý 3: Sửa học viên

- Sơ đồ tuần tự



- Code class DAL

```

212     public void editstudent(Person person) throws Exception {
213         DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
214         HashMap<String, Object> objMap = new HashMap<String, Object>();
215         objMap.put("PersonID", person.getPersonID());
216         objMap.put("Lastname", person.getLastname());
217         objMap.put("Firstname", person.getFirstname());
218         objMap.put("EnrollmentDate", dateFormat.format(person.getEnrollmentDate()));
219
220         try {
221             this.Update("person", objMap, "PersonID='"+person.getPersonID()+"'");
222         } catch (Exception e) {
223             System.out.println("Không thể thêm học viên mới . Vui lòng kiểm tra lại ");
224         }
225     }
226 }
227

```

#### ▪ Code class BLL

```

184     public void editstudent(Person ps) throws Exception {
185         // validate data
186
187         data.editStudent(ps);
188         listPersonStudent.add(ps);
189     }
190
191

```

#### ▪ Code class UI

```

562     private void btn_editstudentActionPerformed(java.awt.event.ActionEvent evt) {
563         // TODO add your handling code here:
564         int i = tbl_hocvien.getSelectedRow();
565         // System.out.println((Date)tbl_giangvien.getValueAt(i, 4));
566         if (i == -1) {
567             JOptionPane.showMessageDialog(null, "Vui lòng chọn dòng cần sửa");
568         } else {
569             txtEditId_Student.setText(String.valueOf(tbl_hocvien.getValueAt(i, 1)));
570             txtEditFirstNameHV.setText(String.valueOf(tbl_hocvien.getValueAt(i, 2)));
571             txtEditLastNameHV.setText(String.valueOf(tbl_hocvien.getValueAt(i, 3)));
572
573             String date = (String)tbl_hocvien.getValueAt(i, 4);
574
575             try {
576                 txtAddEnrollmentDateHV.setDate((Date)fm.parse(date));
577             } catch (ParseException ex) {
578                 Logger.getLogger(QuanLyGiaoVien.class.getName()).log(Level.SEVERE, null, ex);
579             }
580
581             edit_Student.setVisible(true);
582         }
583     }

```

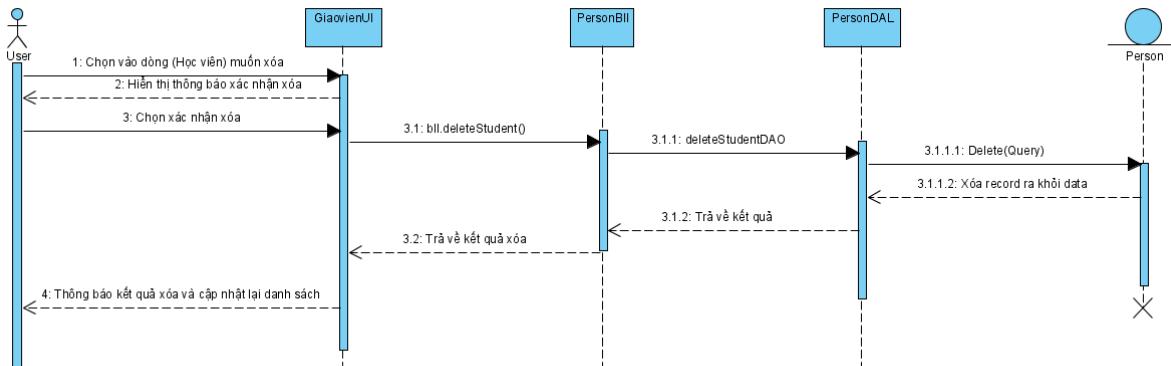
```

private void btnSaveEditStudentActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        if (txtEditId_Student.getText().length() != 0
            && txtAddEnrollmentDateHV.getDate() != null
            && txtEditLastNameHV.getText().length() != 0
            && txtEditFirstNameHV.getText().length() != 0)//xem thử đã nhập hay chưa(textflied)
        {
            studentDTO.setPersonID(Integer.valueOf(txtEditId_Student.getText()));
            studentDTO.setFirstname(txtEditFirstNameHV.getText());
            studentDTO.setLastname(txtEditLastNameHV.getText());
            studentDTO.setEnrollmentDate(txtAddEnrollmentDateHV.getDate());
            personBLL.editStudent(studentDTO);
            personBLL.loadDSPersonStudent();
            JOptionPane.showMessageDialog(null, "Sửa học viên thành công !");
            edit_Student.dispose();
            ShowStudent();
        } else {
            JOptionPane.showMessageDialog(null, "Vui lòng nhập đầy đủ thông tin");
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Failed!");
        e.printStackTrace();
    }
}

```

#### 2.1.3.4. Xử lý 4: Xóa học viên

- Sơ đồ tuần tự



- Code class DAL

```

public void deleteStudent(int PersonID)
{
    try {
        this.Delete("person", "PersonID ='" + PersonID + "'");
    } catch (Exception e) {
        System.out.println("Lỗi không thể xóa học viên !");
    }
}

```

- Code class BLL

```

192     public void deleteStudent(int PersonID) throws Exception {
193
194         for (Person ps : listPersonStudent) {
195             if (ps.getPersonID() == PersonID) {
196                 try {
197                     listPersonStudent.remove(ps);
198                     data.deleteStudent(PersonID);
199                 } catch (Exception e) {
200                     System.out.println("Không thể xóa học viên !!!");
201                 }
202                 return;
203             }
204         }
205     }
206 }
207
208

```

- Code class UI

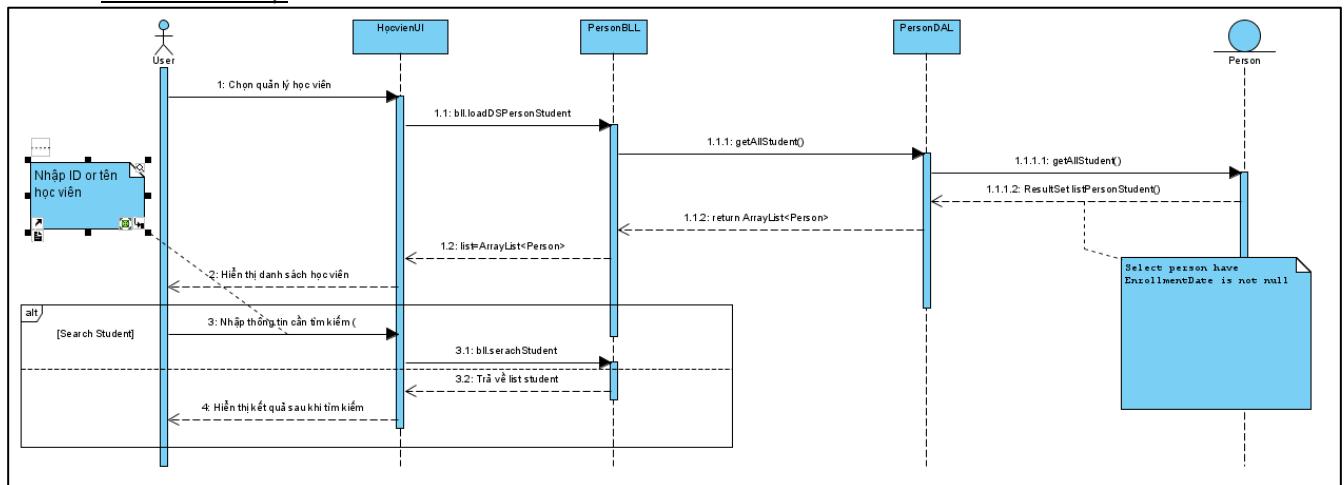
```

588     private void btn_deletestudentActionPerformed(java.awt.event.ActionEvent evt) {
589         // TODO add your handling code here:
590         int i = tbl_hocvien.getSelectedRow();
591         if (i == -1) {
592             JOptionPane.showMessageDialog(null, "Vui lòng chọn dòng cần xóa");
593
594         } else if (JOptionPane.showConfirmDialog(this, "Bạn chắc chắn muốn xóa học viên này?", "Confirm",
595             JOptionPane.YES_OPTION, JOptionPane.NO_OPTION) == JOptionPane.YES_OPTION) {
596             int select = (int)tbl_hocvien.getModel().getValueAt(i, 1);
597
598             studentDTO.setPersonID(select);
599             try {
600                 personBLL.deleteStudent(studentDTO.getPersonID());
601                 showMessageDialog(null, "Xóa thành công");
602                 personBLL.loadDSPersonStudent();
603                 ShowStudent();
604             } catch (Exception e) {
605                 showMessageDialog(null, "Xóa thất bại");
606             }
607         }
608     }
609 }
610

```

### 2.1.3.5. Xử lý 5. Tìm kiếm học viên

- So đồ tuần tự



- Code class BLL

```

66     public ArrayList<Person> searchPersonWithID(int personID) {
67         ArrayList<Person> search = new ArrayList<>();
68         //personID=personID.isEmpty() ? personID="" : personID;
69
70         for (Person ps : listPerson) {
71
72             if (ps.getPersonID() == personID) {
73
74                 search.add(ps);
75             }
76         }
77         return search;
78     }
79
80

```

```

81     public ArrayList<Person> searchPersonWithFirstName(String firstName) {
82
83         firstName = firstName.isEmpty() ? firstName = "" : firstName;
84
85         for (Person ps : listPerson) {
86
87             if (ps.getFirstrname().contains(firstName)) {
88
89                 listPerson.add(ps);
90             }
91         }
92         return listPerson;
93     }
94
95     public ArrayList<Person> searchPersonWithLastName(String lastName) {
96
97
98         lastName = lastName.isEmpty() ? lastName = "" : lastName;
99
100        for (Person ps : listPerson) {
101
102            if (ps.getLastname().contains(lastName)) {
103
104                listPerson.add(ps);
105            }
106        }
107        return listPerson;
108    }
109

```

- Code class UI

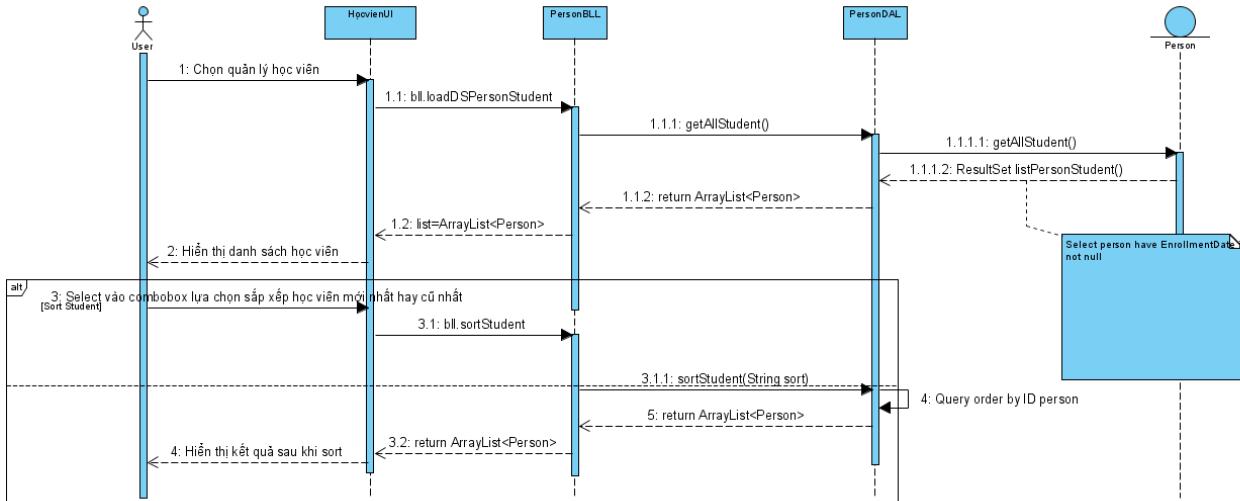
```

private void txFind_HVKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    TableRowSorter<DefaultTableModel> tr = new TableRowSorter<DefaultTableModel>(model);
    tbl_hocvien.setRowSorter(tr);
    tr.setRowFilter(RowFilter.regexFilter(txFind_HV.getText().trim()));
}

```

#### 2.1.3.6. Xử lý 6: Sort học viên theo thứ tự mới nhất và cũ nhất

- Sắp xếp tuần tự



- **Code class DAL**

```

public ArrayList<Person> sortStudent(String value) throws Exception {
    listPersonDTOs = new ArrayList<>();
    try {

        ResultSet rs = this.SelectCustom("person as ps", "ps.PersonID",
            + "ps.Lastname, ps.Firstname, ps.HireDate, ps.EnrollmentDate"
            + "ps.EnrollmentDate IS NOT NULL ORDER BY PersonID" + value);
        while(rs.next())
        {
            Person psDTO=new Person(
                rs.getInt("PersonID"),
                rs.getString("Lastname"),
                rs.getString("Firstname"),
                rs.getDate("HireDate"),
                rs.getDate("EnrollmentDate")
            );
            listPersonDTOs.add(psDTO);
        }
        rs.close();
        this.Close(); //dong ket noi;

    } catch (SQLException ex) {
        System.out.println("Khong the load database Person: "+ex);
    }

    return listPersonDTOs;
}

```

- **Code class BLL**

```

public ArrayList<Person> sortStudent(String value) throws Exception {

    ArrayList<Person> listAllPerson = new ArrayList<Person>();
    listAllPerson = data.sortStudent(value);
    return listAllPerson;
}

```

- Code class UI

```

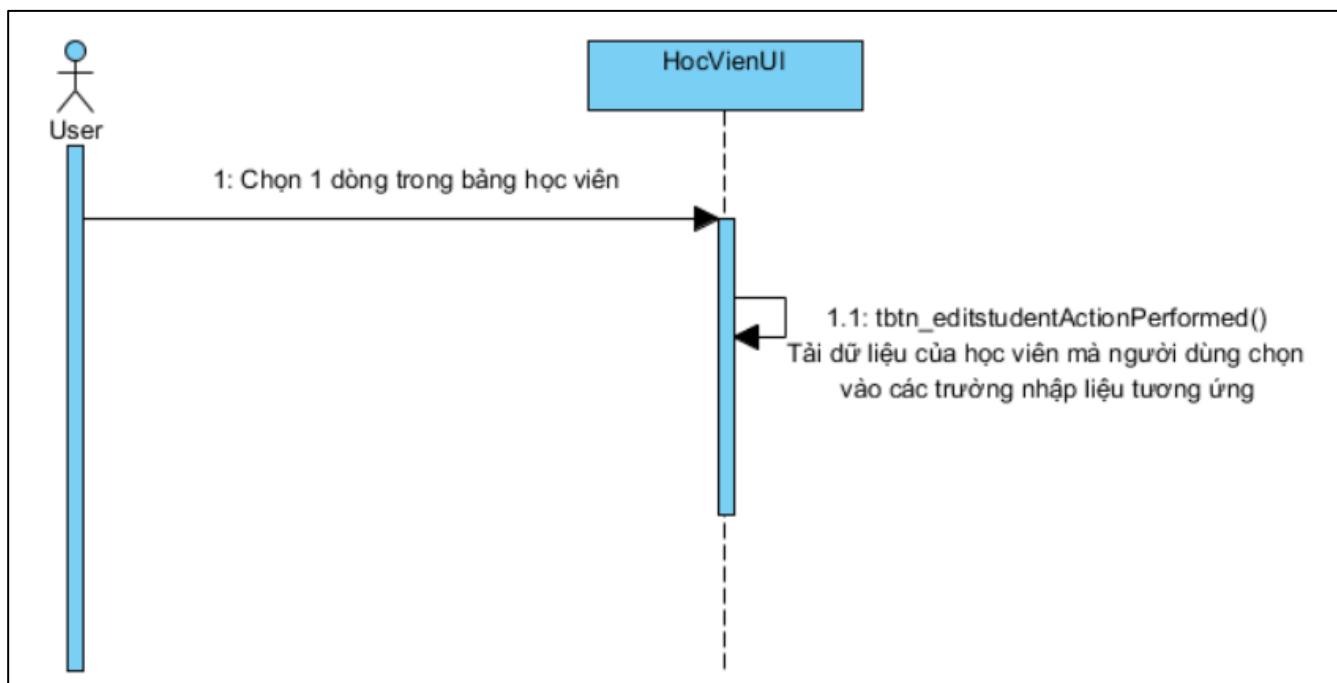
private void SortStudentActionPerfomed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String s = (String) SortStudent.getSelectedItem();

    if (s.equals("DESC")) {
        try {
            listStudent = personBLL.sortStudent("DESC");
        } catch (Exception ex) {
            Logger.getLogger(QuanLyHocVien.class.getName()).log(Level.SEVERE, null, ex);
        }
        outModel(model, listStudent);
    } else if (s.equals("ASC")) {
        try {
            listStudent=personBLL.sortStudent("ASC");
        } catch (Exception ex) {
            Logger.getLogger(QuanLyHocVien.class.getName()).log(Level.SEVERE, null, ex);
        }
        outModel(model, listStudent);
    }
}

```

#### 2.1.3.1. Xử lý 7: Hiển thị chi tiết 1 dòng (record)

Sơ đồ tuần tự



Code class UI

```

private void btn_editstudentActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int i = tbl_hocvien.getSelectedRow();
    // System.out.println((Date)tbl_giangvien.getValueAt(i, 4));
    if (i == -1) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Vui lòng chọn dòng cần sửa");
    } else {
        txtEditId_Student.setText(t: String.valueOf(obj:tbl_hocvien.getValueAt(row: i, column: 1)));
        txtEditFirstNameHV.setText(t: String.valueOf(obj:tbl_hocvien.getValueAt(row: i, column: 2)));
        txtEditLastNameHV.setText(t: String.valueOf(obj:tbl_hocvien.getValueAt(row: i, column: 3)));

        String date = (String)tbl_hocvien.getValueAt(row: i, column: 4);

        try {
            txtAddEnrollmentDateHV.setDate((Date)fm.parse(source: date));
        } catch (ParseException ex) {
            Logger.getLogger(name:QuanLyGiaoVien.class.getName()).log(level:Level.SEVERE, msg: null, thrown:ex);
        }

        edit_Student.setVisible(b: true);
    }
}

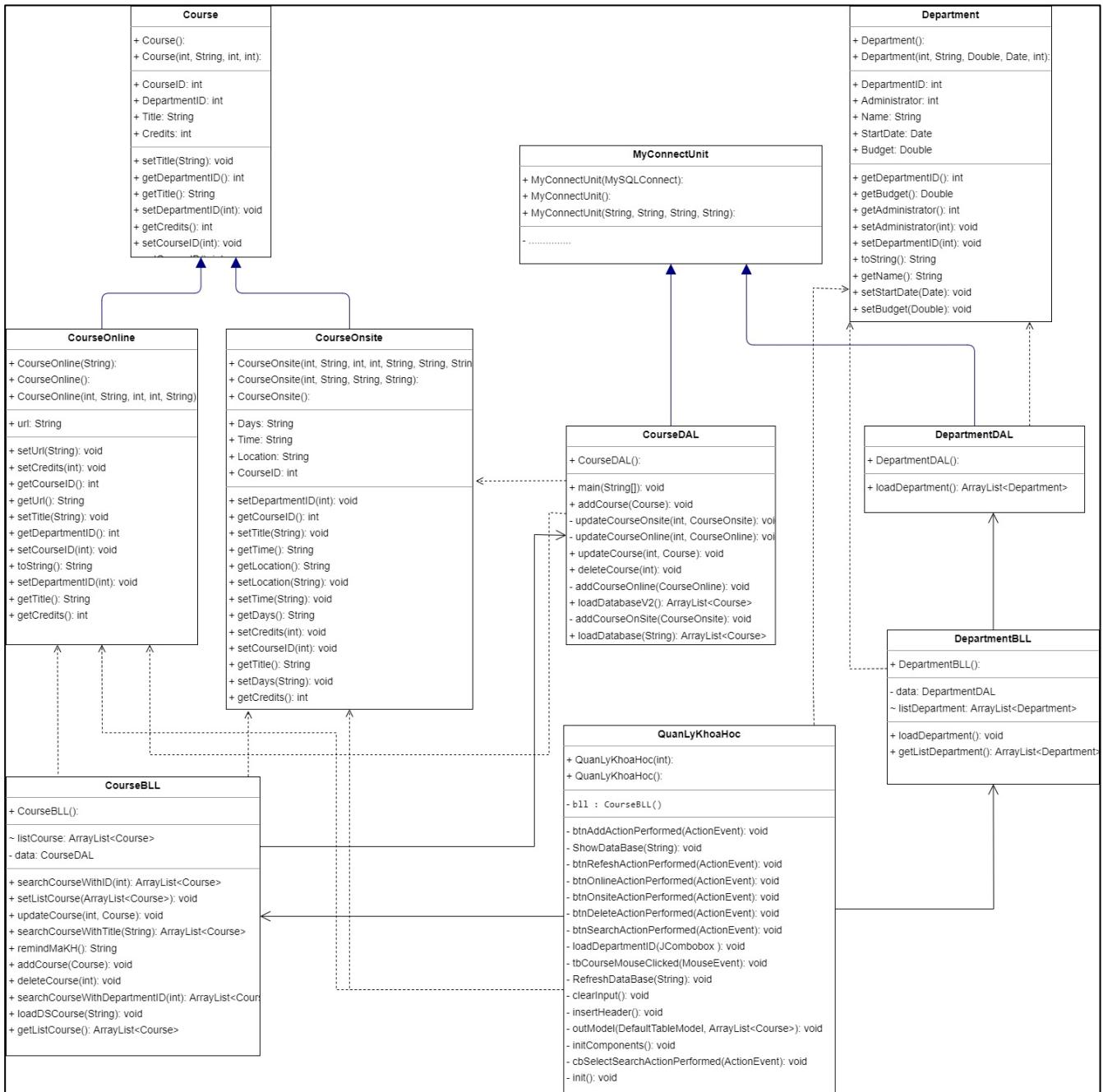
```

UI:

STT	Mã học viên	Họ	Tên	Ngày đăng ký
1	33	Erica	Gao	2003-01-30
2	<b>Sửa Học viên</b>			
3	ID Học viên 28		Shan	2003-09-01
4	Họ Học viên Anthony	Tên Học viên White	Griffin	2004-09-01
5	Enrollment Date: Sep 1, 2001			White 2001-09-01
6				Rogers 2002-09-01
7	<input type="button" value="Lưu"/> <input type="button" value="Hủy"/>			Martin 2005-09-01
8				Morgan 2001-09-01
				Alexander 2005-09-01

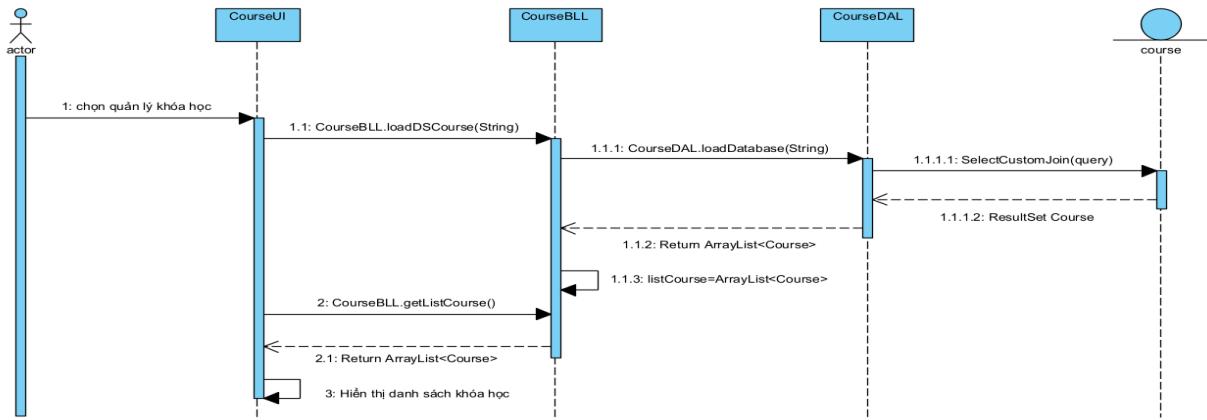
## 2.2.Chức năng Quản lý khóa học

### 2.2.1. Sơ đồ Class Chung



## 2.2.2. Xử lý 1: Hiển thị danh sách khóa học

### ❖ Sơ đồ tuần tự



#### ▪ Code class DAL

```

public ArrayList<Course> loadDatabase(String orderby) throws Exception {
    ArrayList<Course> listCourse = new ArrayList<>();
    try {
        ResultSet rs = this.SelectCustomJoin(tableName:"course as cs",
            Custom: "cs.CourseID,cs.Title,cs.Credits,cs.DepartmentID, cs.on,cssite.Location, cssite.Days,cssite.Time",
            Join: "LEFT OUTER JOIN onsitecourse as cssite ON cs.CourseID=cssite.CourseID LEFT OUTER JOIN onlinecourse as cson ON cs.C
            "cs.CourseID " + orderby
        );
        while (rs.next()) {
            if (rs.getString(columnLabel:"url") == null) {
                Course cssite = new CourseOnsite(
                    CourseID:rs.getInt(columnLabel:"CourseID"), title:rs.getString(columnLabel:"Title"),
                    Credits:rs.getInt(columnLabel:"Credits"), DepartmentID:rs.getInt(columnLabel:"DepartmentID"),
                    Location:rs.getString(columnLabel:"Location"), Days:rs.getString(columnLabel:"Days"), Time:rs.getString(columnLabel:"Time")
                );
                listCourse.add((Course) cssite);
            } else {
                Course cson = new CourseOnline(
                    CourseID:rs.getInt(columnLabel:"CourseID"), title:rs.getString(columnLabel:"Title"),
                    Credits:rs.getInt(columnLabel:"Credits"), DepartmentID:rs.getInt(columnLabel:"DepartmentID"),
                    url:rs.getString(columnLabel:"url")
                );
                listCourse.add((Course) cson);
            }
        }
        rs.close();
        this.Close(); // đóng kết nối;
    } catch (SQLException ex) {
        System.out.println(ex:"Không thể load database Course");
    }
    return listCourse;
}
  
```

#### ▪ Code class BLL

```

public class CourseBLL {

    static ArrayList<Course> listCourse;
    private CourseDAL data = new CourseDAL();

    public CourseBLL() {
    }

    public static ArrayList<Course> getListCourse() {
        return listCourse;
    }

    public static void setListCourse(ArrayList<Course> listCourse) {
        CourseBLL.listCourse = listCourse;
    }

    public void loadDSCourse(String orderby) throws Exception {

        if (listCourse == null) {
            listCourse = new ArrayList<Course>();
        }
        listCourse = data.loadDatabase(orderby); // gọi Layer DAL hàm đọc data từ CSDL
    }
}

```

- *Code class UI*

```

27  public class QuanLyKhoaHoc extends javax.swing.JPanel {

28
29      private int DEFALUT_WIDTH;
30      private DefaultTableModel model;
31      CourseBLL bll = new CourseBLL();
32      DepartmentBLL dll = new DepartmentBLL();
33
34      public QuanLyKhoaHoc() {
35          this.setSize(width:1090, height:750);
36          initComponents();
37      }
38
39      public QuanLyKhoaHoc(int width) throws Exception {
40          DEFALUT_WIDTH = width;
41          initComponents();
42          this.setSize(this.DEFALUT_WIDTH - 200, height:750);
43          init();
44      }
45
46      private void init() throws Exception {
47          ShowDataBase(orderby: "ASC");
48          loadDepartmentID(cb:cbDepartment);
49      }
}

```

```

private void insertHeader() {
    Vector header = new Vector();
    header.add( e: "Mã Khoa Học");
    header.add( e: "Tên ");
    header.add( e: "Giá");
    header.add( e: "Phòng");
    header.add( e: "URL");
    header.add( e: "Địa điểm");
    header.add( e: "Ngày");
    header.add( e: "Giờ");

    //if (model.getRowCount()==0)
    model = new DefaultTableModel( columnNames:header, rowCount: 0);

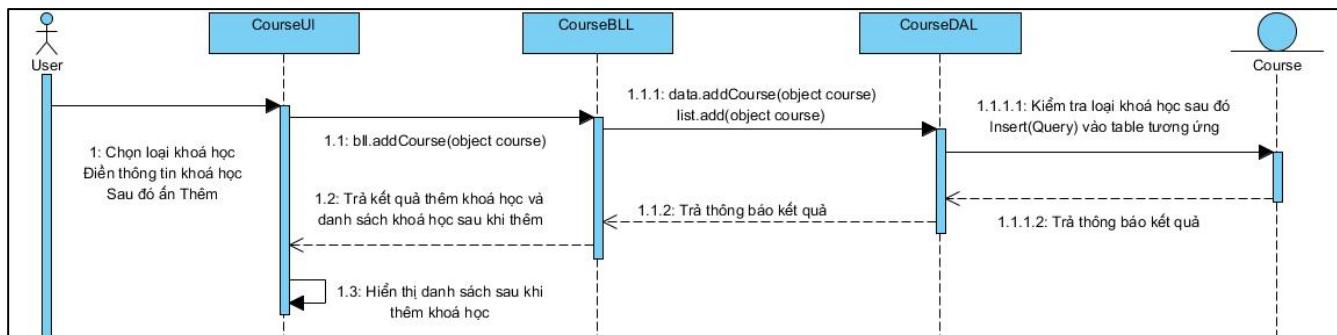
}

private void ShowDataBase(String orderby) throws Exception {
    try {
        if (CourseBLL.getListCourse() == null) {
            bll.loadDSCourse(orderby);
        }
        insertHeader();
        outModel(model, course:CourseBLL.getListCourse());
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: this, message:"Không Thể Load Database ", title: "Thông Báo Lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

### 2.2.3. Xử lý 2: Thêm khóa học

#### ❖ Sơ đồ tuần tự



- Code class DAL

```

public void addCourse(Course cs) throws Exception {
    try {
        if (cs instanceof CourseOnline) {
            this.addCourseOnline((CourseOnline) cs);
        } else {
            this.addCourseOnSite((CourseOnsite) cs);
        }
    } catch (Exception ex) {
        System.out.println("Khong the them Courses vao database !!!");
    }
}

```

```

private void addCourseOnline(CourseOnline cs) throws Exception {
    HashMap<String, Object> InsertvaluesCourse = new HashMap<String, Object>();
    HashMap<String, Object> InsertvaluesCourseOnline = new HashMap<String, Object>();

    InsertvaluesCourse.put(key: "CourseID", value: cs.getCourseID());
    InsertvaluesCourse.put(key: "Title", value: cs.getTitle());
    InsertvaluesCourse.put(key: "Credits", value: cs.getCredits());
    InsertvaluesCourse.put(key: "DepartmentID", value: cs.getDepartmentID());
    InsertvaluesCourseOnline.put(key: "url", value: cs.getUrl());

    try {
        this.Insert(tableName: "course", columnValue: InsertvaluesCourse);
        ResultSet rs = this.SelectCustomorderby(tableName: "course", Custom: "CourseID", condition: null, orderby: "CourseID DESC limit 1");
        while (rs.next()) {
            InsertvaluesCourseOnline.put(key: "CourseID", value: rs.getInt(columnLabel: "CourseID"));
            System.out.print(rs.getInt(columnLabel: "CourseID"));
        }

        this.Insert(tableName: "onlinecourse", columnValue: InsertvaluesCourseOnline);
    } catch (SQLException ex) {
        System.out.println("Khong the them CourseOnline vao database !!!");
    }
}

```

```

private void addCourseOnSite(CourseOnsite cs) throws Exception {
    HashMap<String, Object> InsertvaluesCourse = new HashMap<String, Object>();
    HashMap<String, Object> InsertvaluesCourseOnsite = new HashMap<String, Object>();

    InsertvaluesCourse.put(key: "CourseID", value: cs.getCourseID());
    InsertvaluesCourse.put(key: "Title", value: cs.getTitle());
    InsertvaluesCourse.put(key: "Credits", value: cs.getCredits());
    InsertvaluesCourse.put(key: "DepartmentID", value: cs.getDepartmentID());
    InsertvaluesCourseOnsite.put(key: "Location", value: cs.getLocation());
    InsertvaluesCourseOnsite.put(key: "Days", value: cs.getDays());
    InsertvaluesCourseOnsite.put(key: "Time", value: cs.getTime());
    try {
        this.Insert(tableName: "course", columnValue: InsertvaluesCourse);
        ResultSet rs = this.SelectCustomorderby(tableName: "course", Custom: "CourseID", condition: null, orderby: "CourseID DESC limit 1");
        while (rs.next()) {
            InsertvaluesCourseOnsite.put(key: "CourseID", value: rs.getInt(columnLabel: "CourseID"));
        }

        this.Insert(tableName: "onsitecourse", columnValue: InsertvaluesCourseOnsite);
    } catch (SQLException ex) {
        System.out.println("Khong the them CourseOnsite vao database !!!");
    }
}

```

- Code class BLL

```

public void addCourse(Course cs) throws Exception {
    try{
        data.addCourse(cs);
        listCourse.add( e:cs);
    }
    catch(Exception ex){
        System.out.println( x:"Khong the them Course Item vao database ");
    }
}

```

#### ▪ Code class UI

```

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

    int CourseID = Integer.parseInt(s:txtCourseID.getText());
    int DepartmentID = Integer.parseInt(s:cbDepartment.getSelectedItem().toString());
    int Credits = Integer.parseInt(s:txtCredits.getText());
    String Title = txtTitle.getText();
    String Url = txtUrl.getText();
    String Location = txtLocation.getText();
    String Days = txtDate.getText().toString();
    String Time = txtTime.getText();

    try {
        if (!txtUrl.getText().isEmpty()) {
            Course courseOnline = new CourseOnline(CourseID, title:Title, Credits, DepartmentID, url:Url);

            bll.addCourse( cs:courseOnline);
        } else {
            Course courseOnsite = new CourseOnsite(CourseID, title:Title, Credits, DepartmentID, Location, Days, Time);

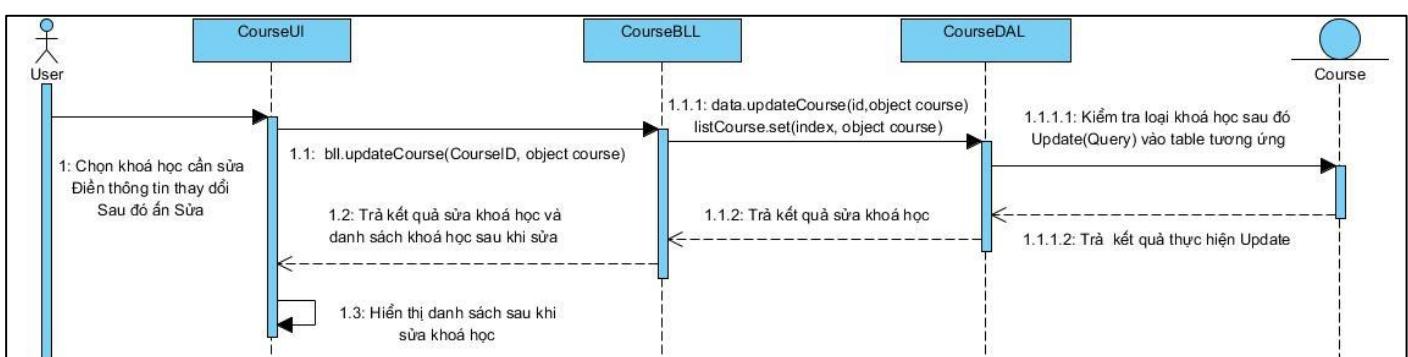
            bll.addCourse( cs:courseOnsite); // gọi Layer Bll Thêm khóa học
        }

        clearInput();
        RefreshDataBase( orderby: "DESC");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Không thể tạo phân công giảng dạy",
            title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

#### 2.2.4. Xử lý 3: Cập nhật thông tin khóa học

##### ❖ Sơ đồ tuần tự



## ▪ Code class DAL

```

private void updateCourseOnline(int id, CourseOnline cs) throws Exception {

    HashMap<String, Object> UpdatevaluesCourseOnline = new HashMap<String, Object>();
    UpdatevaluesCourseOnline.put(key: "url", value: cs.getUrl());
    try {
        this.Update(tableName: "onlinecourse", columnValue: UpdatevaluesCourseOnline, "CourseID ='" + id + "'");
    } catch (SQLException ex) {
        System.out.println(x: "Khong the cap nhat CourseOnline !!!");
    }
}

private void updateCourseOnsite(int id, CourseOnsite cs) throws Exception {

    HashMap<String, Object> UpdatevaluesCourseOnsite = new HashMap<String, Object>();
    UpdatevaluesCourseOnsite.put(key: "Location", value: cs.getLocation());
    UpdatevaluesCourseOnsite.put(key: "Days", value: cs.getDays());
    UpdatevaluesCourseOnsite.put(key: "Time", value: cs.getTime());

    try {
        this.Update(tableName: "onsitecourse", columnValue: UpdatevaluesCourseOnsite, "CourseID ='" + id + "'");
    } catch (SQLException ex) {
        System.out.println(x: "Khong the cap nhat CourseOnline !!!");
    }
}

public void updateCourse(int id, Course cs) throws Exception {
    HashMap<String, Object> UpdatevaluesCourse = new HashMap<String, Object>();
    UpdatevaluesCourse.put(key: "CourseID", value: cs.getCourseID());
    UpdatevaluesCourse.put(key: "Title", value: cs.getTitle());
    UpdatevaluesCourse.put(key: "Credits", value: cs.getCredits());
    UpdatevaluesCourse.put(key: "DepartmentID", value: cs.getDepartmentID());
    try {

        this.Update(tableName: "course", columnValue: UpdatevaluesCourse, "CourseID ='" + id + "'");
        if (cs instanceof CourseOnsite) {
            this.updateCourseOnsite(id, (CourseOnsite) cs);
        } else {
            this.updateCourseOnline(id, (CourseOnline) cs);
        }
    } catch (SQLException ex) {
        System.out.println(x: "Khong the Cap nhat Course vao database !!!");
    }
}

```

## ▪ Code class BLL

```

public void updateCourse(int id, Course cs) throws Exception {
    for (int i = 0; i < listCourse.size(); i++) {
        if (listCourse.get(index: i).getCourseID() == cs.getCourseID()) {
            try {
                data.updateCourse(id, cs);
                listCourse.set(index: i, element: cs);
            } catch (Exception e) {
                System.out.println(x: "Khong the Cap nhat Course vao database !!!");
            }
        }
    }
}

```

- Code class UI

```

private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {
    int CourseID = Integer.parseInt(s:txtCourseID.getText());
    int DepartmentID = Integer.parseInt(s:cbDepartment.getSelectedItem().toString());
    int Credits = Integer.parseInt(s:txtCredits.getText());
    String Title = txtTitle.getText();
    String Url = txtUrl.getText();
    String Location = txtLocation.getText();
    String Days = txtDate.getText();
    String Time = txtTime.getText();

    try {
        if (!txtUrl.getText().isEmpty()) {
            Course courseOnline = new CourseOnline(CourseID, title:Title, Credits, DepartmentID, uri:Url);

            bll.updateCourse( id:CourseID, cs:courseOnline);
        } else {
            Course courseOnsite = new CourseOnsite(CourseID, title:Title, Credits, DepartmentID, Location, Days, Time);

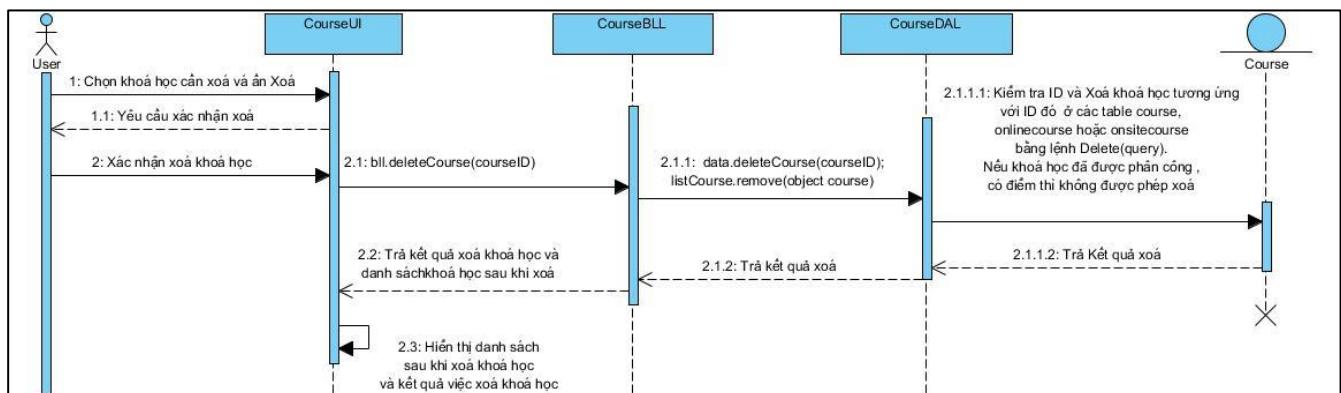
            bll.updateCourse( id:CourseID, cs:courseOnsite); // gọi Layer Bll Thêm khóa học
        }

        clearInput();
        RefreshDataBase( orderby: "DESC" );
    } catch (Exception ex) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Không thể cập nhật CourseInstructor " ,
                                     title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

### 2.2.5. Xử lý 4: Xóa khóa học

- ❖ Sơ đồ tuần tự



- Code class DAL

```

public void deleteCourse(int courseID) {

    try {
        ResultSet rs = this.Select(tableName: "onlinecourse", "CourseID ='" + courseID + "'");
        int check = -1;
        while (rs.next()) {
            check = rs.getInt(columnLabel: "CourseID");
        }
        System.out.println("check: " + check);
        if (check == -1) {
            this.Delete(tableName: "onsitecourse", "CourseID ='" + courseID + "'");
            this.Delete(tableName: "course", "CourseID ='" + courseID + "'");
        } else {
            this.Delete(tableName: "onlinecourse", "CourseID ='" + courseID + "'");
            this.Delete(tableName: "course", "CourseID ='" + courseID + "'");
        }
    } catch (Exception e) {
        System.out.println("Lỗi không thể xóa course item !!!");
    }
}

```

- Code class BLL

```

public void deleteCourse(int courseID) throws Exception {
    for (Course csin : listCourse) {
        if (csin.getCourseID() == courseID) {
            try {
                data.deleteCourse(csin);
                listCourse.remove(csin); // xoá trong arraylist
            } catch (Exception e) {
                System.out.println("Không thể Xóa Khoa Học !!!");
            }
            return;
        }
    }
}

```

- Code class UI

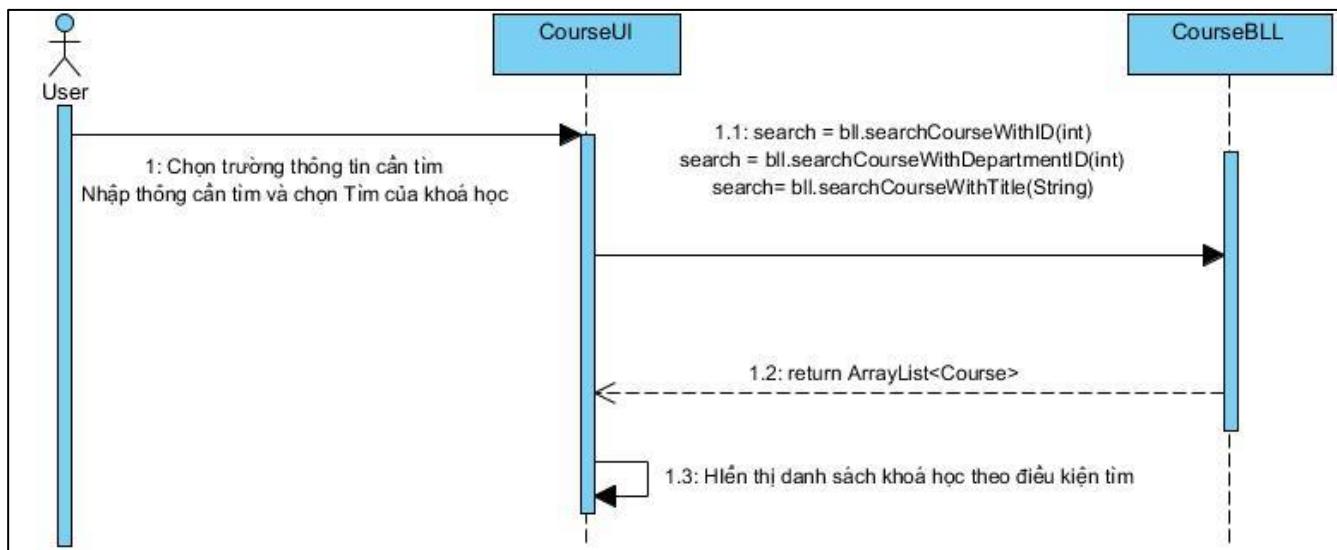
```

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    int confirm = JOptionPane.showConfirmDialog(parentComponent: null, message: "Bạn Thực Sự Muốn Xóa Khóa Học Này ?",
        title: "Thông Báo", optionType: JOptionPane.YES_NO_OPTION);
    if (confirm == 0)
        try {
            int courseID = Integer.parseInt(s:txtCourseID.getText());
            bll.deleteCourse(courseID); //gọi Layer BLL xoá
            insertHeader(); // chèn header cho table
            outModel(model, course:CourseBLL.getListCourse()); // đổ data vào table
            clearInput();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(parentComponent: this, message: "Không thể xoá Khóa Học này",
                title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
        } else
            return;
    }
}

```

## 2.2.6. Xử lý 5: Tìm kiếm khóa học

### ❖ Sơ đồ tuần tự



### ▪ Code class BLL

```

public ArrayList<Course> searchCourseWithID(int courseID) {
    ArrayList<Course> search = new ArrayList<>();

    for (Course ps : listCourse) {

        if (ps.getCourseID() == courseID) {
            search.add( e:ps);
        }
    }
    return search;
}

public ArrayList<Course> searchCourseWithTitle(String title) {
    ArrayList<Course> search = new ArrayList<>();
    //courseID=courseID.isEmpty() ?courseID="":courseID;

    for (Course ps : listCourse) {

        if (ps.getTitle().trim().toLowerCase().contains( s:title.trim().toLowerCase())) {

            search.add( e:ps);
        }
    }
    return search;
}

public ArrayList<Course> searchCourseWithDepartmentID(int id) {
    ArrayList<Course> search = new ArrayList<>();
    for (Course ps : listCourse) {

        if (ps.getDepartmentID() == id) {

            search.add( e:ps);
        }
    }
    return search;
}

```

- Code class UI

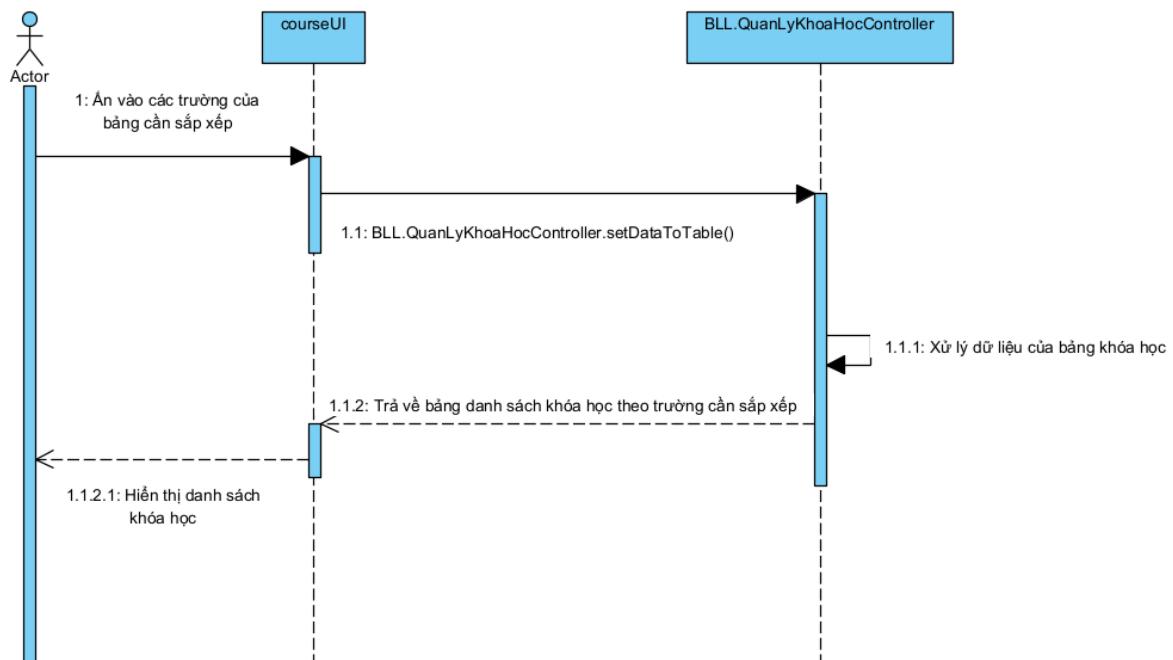
```

private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultTableModel temp = new DefaultTableModel();
    ArrayList<Course> search = new ArrayList<>();
    Vector header = new Vector();
    header.add( e: "Mã Khoa Học");
    header.add( e: "Tên ");
    header.add( e: "Giá");
    header.add( e: "Phòng");
    header.add( e: "URL");
    header.add( e: "Địa điểm");
    header.add( e: "Ngày");
    header.add( e: "Giờ");
    String optionSearch = cbSelectSearch.getSelectedItem().toString();
    try {
        if (!txtSearch.getText().isEmpty()) {
            String inputSearch = txtSearch.getText();
            switch (optionSearch) {
                case "Mã Khoa Học":
                    search = bll.searchCourseWithID( courseId: Integer.parseInt( e:inputSearch));
                    break;
                case "Tên Khoa Học":
                    search = bll.searchCourseWithTitle( title: inputSearch);
                    break;
                case "Số Phòng":
                    search = bll.searchCourseWithDepartmentID( id: Integer.parseInt( e:inputSearch));
                    break;
                default:
                    break;
            }
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Vui lòng nhập điều kiện tìm",
                                         title: "Thông Báo", messageType: JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Không Tìm Kiếm",
                                         title: "Thông Báo Lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
    if (search != null && search.size() > 0) {
        if (temp.getRowCount() == 0) {
            temp = new DefaultTableModel( columnNames: header, rowCount: 0);
        }
        for (int i = 0; i < search.size(); i++) {
            Vector row = new Vector();
            row.add( e: search.get( index: i).getCourseID());
            row.add( e: search.get( index: i).getTitle());
            row.add( e: search.get( index: i).getCredits());
            row.add( e: search.get( index: i).getDepartmentID());
            temp.addRow( row);
        }
    }
}

```

### 2.2.7. Xử lý 6: Hiện thị mã phòng (combobox)

- ❖ Sơ đồ tuần tự



- Code class BLL

```

public void loadDepartment () throws Exception {
    if (listDepartment == null) {
        listDepartment = new ArrayList<Department>();
    }
    listDepartment = data.loadDepartment();
}
  
```

- Code class DAL

```

public ArrayList<Department> loadDepartment() throws Exception{
    ArrayList<Department> listDepartment = new ArrayList<>();
    try {
        ResultSet rs = this.Select( tableName: "department");
        while (rs.next()) {
            Department department = new Department(
                DepartmentID: rs.getInt( columnLabel: "DepartmentID"),
                Name: rs.getString( columnLabel: "Name"),
                Budget: rs.getDouble( columnLabel: "Budget"),
                StartDate: rs.getDate( columnLabel: "StartDate"),
                Administrator: rs.getInt( columnLabel: "Administrator")
            );
            listDepartment.add( e:department);
        }
        rs.close();
        this.Close(); //dong ket noi;
    } catch (SQLException ex) {
        System.out.println( x: "Khong the load database Course");
    }
    return listDepartment;
}

```

- Code class UI

```

public QuanLyKhoaHoc(int width) throws Exception {
    DEFALUT_WIDTH = width;
    initComponents();
    this.setSize(this.DEFALUT_WIDTH - 200, height: 750);
    init();
}

```

```

private void init() throws Exception {
    ShowDataBase( orderby: "ASC");
    loadDepartmentID( cb: cbDepartment);
}

```

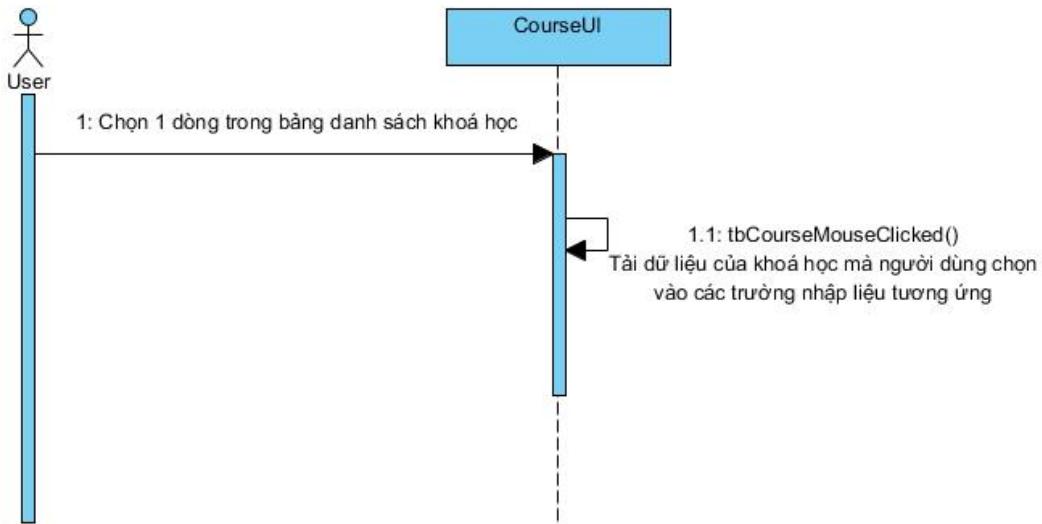
```

public void loadDepartmentID(JComboBox cb) throws Exception {
    if (dll.getListDepartment() == null) {
        dll.loadDepartment();
    }
    ArrayList<Department> departmentList = dll.getListDepartment();
    for (Department d : departmentList) {
        cb.addItem( item: d.getDepartmentID());
    }
}

```

## 2.2.8. Xử lý 7: Hiển thị 1 dòng (record)

### ❖ Sơ đồ tuần tự



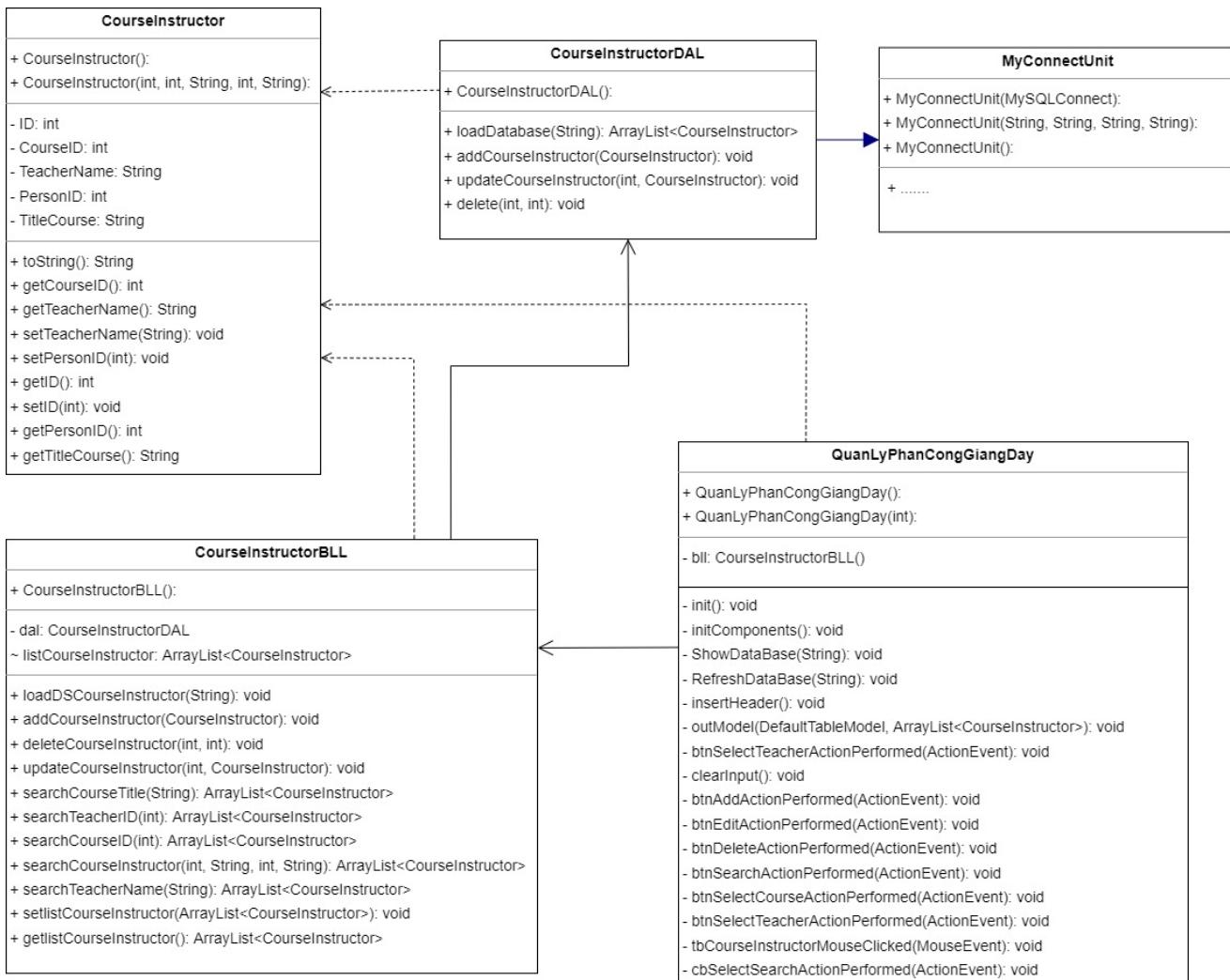
### ▪ Code class UI

```
private void tbCourseMouseClicked(java.awt.event.MouseEvent evt) {
    int i = tbCourse.getSelectedRow();
    clearInput();
    if (i >= 0) {
        if (tbCourse.getRowSorter() != null) {
            i = tbCourse.getRowSorter().convertRowIndexToModel(index:i);
        }

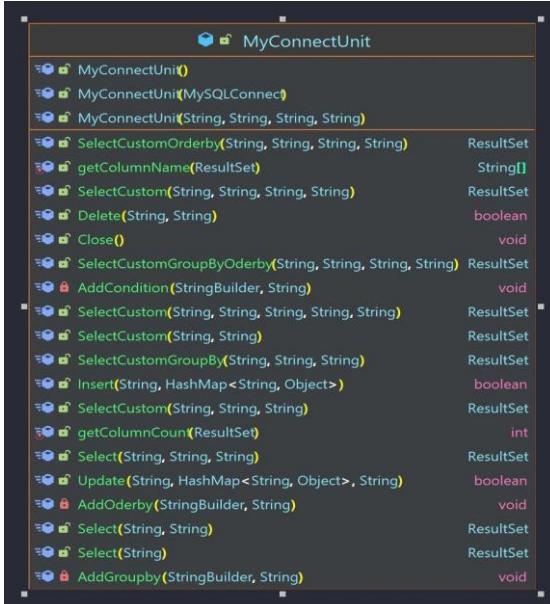
        txtCourseId.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:0).toString());
        txtTitle.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:1).toString());
        txtCredits.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:2).toString());
        cbDepartment.setSelectedItem(anObject:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:3));
        if (tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:4) == null) {
            txtLocation.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:5).toString());
            txtDate.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:6).toString());
            txtTime.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:7).toString());
        } else {
            txtUrl.setText(t:tbCourse.getModel().getValueAt(rowIndex:i, columnIndex:4).toString());
        }
    }
}
```

## 2.3. Chức năng phân công giảng dạy

### 2.3.1. Sơ đồ class chung



## 1. Class MyConnectUnit: định nghĩa các hàm dùng chung thuận tiện cho việc truy vấn CSDL.



## 2. Class CourseInstructorBLL

```

public class CourseInstructorBLL {
    static ArrayList<CourseInstructor> listCourseInstructor;
    private CourseInstructorDAL dal=new CourseInstructorDAL();
    public CourseInstructorBLL() {
    }
}

```

### 3. Class CourseInstructorDAL

```

public class CourseInstructorDAL extends MyConnectUnit {
    public CourseInstructorDAL() {
        super();
    }
}

```

### 4. UI QuanLyPhanCongGiangDay

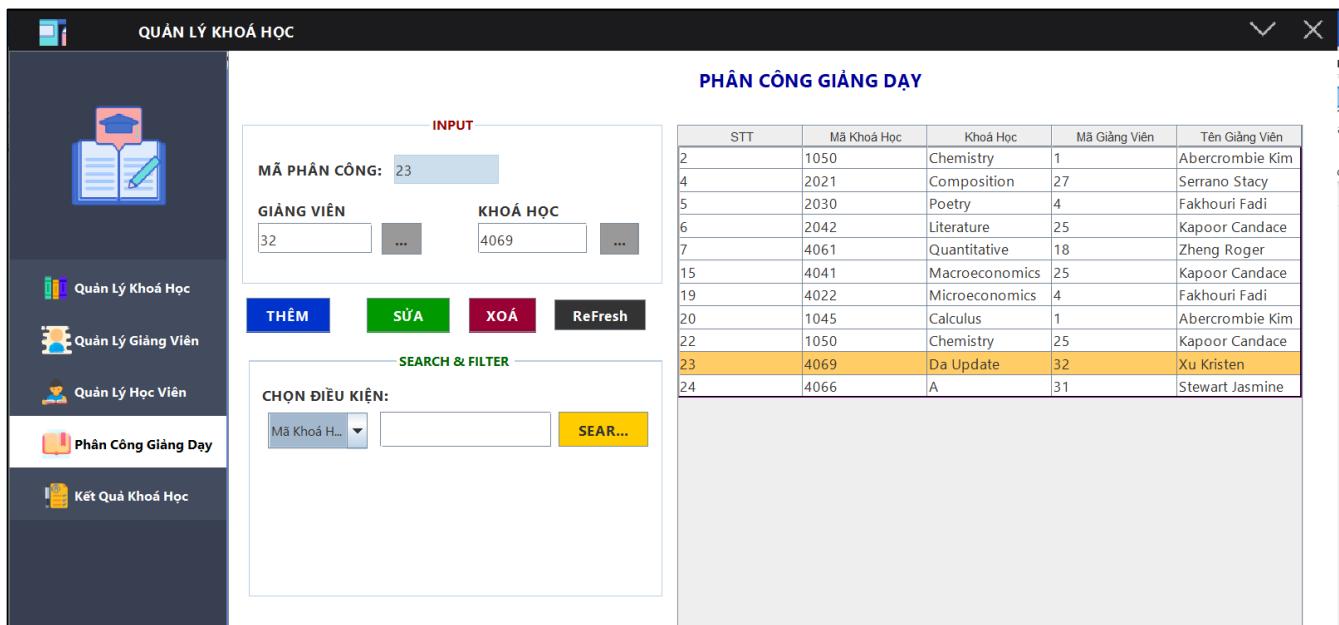
```

public class QuanLyPhanCongGiangDay extends javax.swing.JPanel {

    private int DEFALUT_WIDTH;
    private DefaultTableModel model;
    private CourseInstructorBLL bll = new CourseInstructorBLL();
}

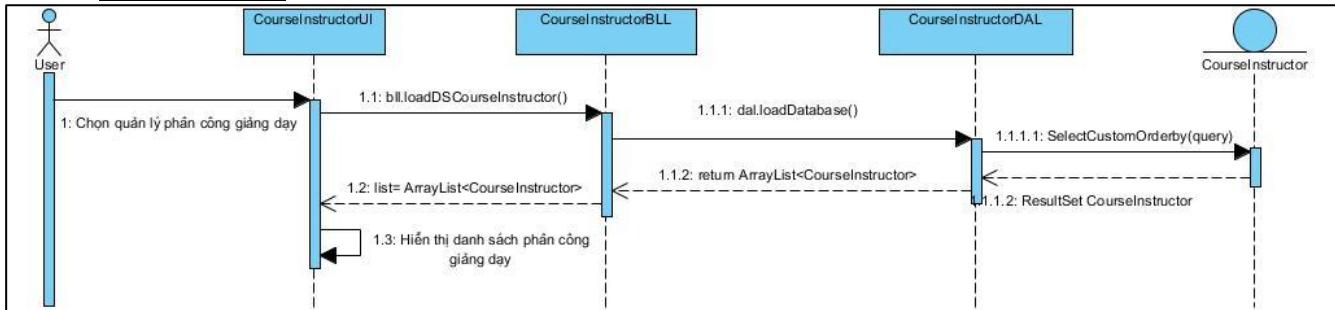
```

UI:



### 2.3.2. Xử lý 1: Hiển thị danh sách phân công giảng dạy

#### ❖ Sơ đồ tuần tự



#### ▪ Code class DAL

```

public ArrayList<CourseInstructor> loadDatabase(String orderby) throws Exception
{
    ArrayList<CourseInstructor> list = new ArrayList<>();
    try {
        ResultSet rs = this.SelectCustomOrderby(tableName: "course as cs , person as ps, courseinstructor as csin",
            Custom: "csin.ID,cs.CourseID,cs.Title, ps.PersonID, ps.Lastname, ps.Firstname",
            condition: "cs.CourseID=csin.CourseID AND ps.PersonID=csin.PersonID",
            "csin.ID "+orderby);
        while(rs.next())
        {
            CourseInstructor csin = new CourseInstructor(
                ID:rs.getInt(columnLabel: "ID"), CourseID:rs.getInt(columnLabel: "CourseID"),
                TitleCourse: rs.getString(columnLabel: "Title"), PersonID:rs.getInt(columnLabel: "PersonID"),
                rs.getString(columnLabel: "Lastname")+" "+rs.getString(columnLabel: "Firstname")
            );
            list.add(e:csin);
        }
        rs.close();
        this.Close(); //đóng kết nối;
    } catch (SQLException ex) {
        System.out.println("Không thể load database CourseInstructor: "+ex);
    }
    return list;
}
  
```

#### ▪ Code class BLL

```

public void loadDSCourseInstructor(String orderby) throws Exception{
    if(listCourseInstructor==null) listCourseInstructor = new ArrayList<CourseInstructor>();
    listCourseInstructor=dal.loadDatabase(orderby);
}
  
```

#### ▪ Code class UI

```

public QuanLyPhanCongGiangDay(int width) throws Exception {
    DEFALUT_WIDTH = width;
    initComponents();
    this.setSize(this.DEFALUT_WIDTH - 200, height: 750);
    ShowDataBase(orderby: "ASC");
}
  
```

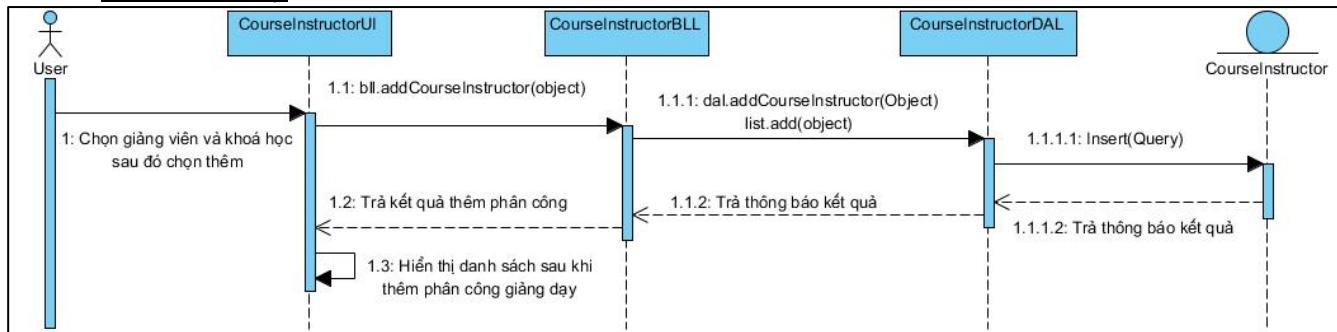
```

private void ShowDataBase(String orderby) throws Exception {
    try {
        if (CourseInstructorBLL.getListCourseInstructor() == null) {
            bll.loadDSCourseInstructor(orderby);
        }
        insertHeader();
        outModel(model, courseinstructor:CourseInstructorBLL.getListCourseInstructor());
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Không Thể Load Database ", title: "Thông Báo Lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

### 2.3.3. Xử lý 2: Thêm phân công giảng dạy

#### ❖ Sơ đồ tuần tự



#### ▪ Code class DAL

```

public void addCourseInstructor(CourseInstructor csin) throws Exception {
    {
        HashMap<String, Object> Insertvalues =new  HashMap<String, Object>();
        Insertvalues.put( key: "CourseID", value: csin.getCourseID());
        Insertvalues.put( key: "PersonID",  value: csin.getPersonID());
        try {
            this.Insert( tableName: "courseinstructor", columnValue: Insertvalues);
        } catch (SQLException ex) {
            System.out.println( x: "Khong the them CourseInstructor vao database !!!");
        }
    }
}

```

#### ▪ Code class BLL

```

public void addCourseInstructor(CourseInstructor csin) throws Exception{
    try{
        dal.addCourseInstructor(csin);
        listCourseInstructor.add( e:csin);
    }
    catch(Exception ex){
        System.out.println( x: "Khong the them CourseInstructor Item vao database ");
    }
}

```

- *Code class UI*

```

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

    try {
        CourseInstructor csin = new CourseInstructor();
        csin.setCourseID((int) Integer.parseInt( s:txCourse.getText()));
        csin.setPersonID((int) Integer.parseInt( s:txTeacher.getText()));
        bll.addCourseInstructor(csin); // gọi Layer Bll Thêm phân công
        clearInput();
        RefreshDataBase( orderby: "DESC");

    } catch (Exception ex) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Không thể tạo phân công giảng dạy",
                                     title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void clearInput() {
    txID.setText( t: "" );
    txCourse.setText( t: "" );
    txTeacher.setText( t: "" );
}

```

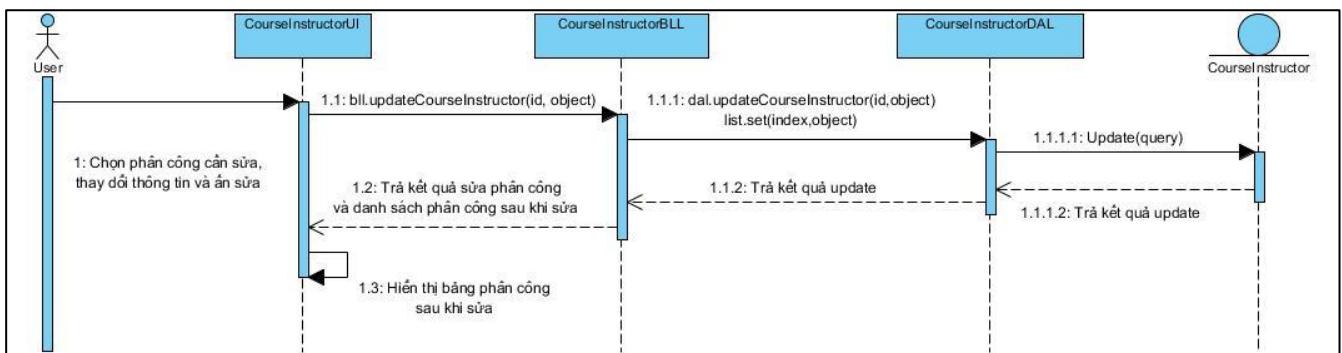
```

private void RefreshDataBase(String orderby) throws Exception {
    try {
        bll.loadDSCourseInstructor(orderby);
        insertHeader();
        outModel(model, courseinstructor:CourseInstructorBLL.getListCourseInstructor());
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: this, message: "Không Thể Load Database",
                                     title: "Thông Báo Lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

#### 2.3.4. Xử lý 3: Sửa phân công giảng dạy

- ❖ Sơ đồ tuần tự



#### ▪ Code class DAL

```

public void updateCourseInstructor(int id, CourseInstructor csin) throws Exception {
    HashMap<String, Object> Updatevalues = new HashMap<String, Object>();
    Updatevalues.put("CourseID", csin.getCourseID());
    Updatevalues.put("PersonID", csin.getPersonID());

    try {
        this.Update(tableName: "courseinstructor", columnValue: Updatevalues, "ID ='" + id + "'");
    } catch (SQLException ex) {
        System.out.println("Khong the Cap nhat CourseInstructor vao database !!!");
    }
}
  
```

#### ▪ Code class BLL

```

public void updateCourseInstructor(int id, CourseInstructor csin) throws Exception {
    for(int i = 0 ; i < listCourseInstructor.size() ; i++) {
        if(listCourseInstructor.get(index: i).getCourseID() == csin.getCourseID()) {
            try {
                dal.updateCourseInstructor(id, csin);
                listCourseInstructor.set(index: i, element: csin);
            } catch (Exception e) {
                System.out.println("Khong the Cap nhat CourseInstructor vao database !!!");
            }
        }
    }
}
  
```

#### ▪ Code class UI

```

private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int id = Integer.parseInt(s:txID.getText());
        int teacherID = Integer.parseInt(s:txTeacher.getText());
        int courseID = Integer.parseInt(s:txCourse.getText());
        CourseInstructor csin = new CourseInstructor();
        csin.setCourseID((int) courseID);
        csin.setPersonID((int) teacherID);
        // gọi Layer BLL cập nhật phân công
        bll.updateCourseInstructor(id, csin);
        clearInput();
        RefreshDataBase(orderby: "DESC");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(parentComponent: this, message: "Không thể cập nhật CourseInstructor",
            title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void clearInput() {
    txID.setText(t: "");
    txCourse.setText(t: "");
    txTeacher.setText(t: "");
}

```

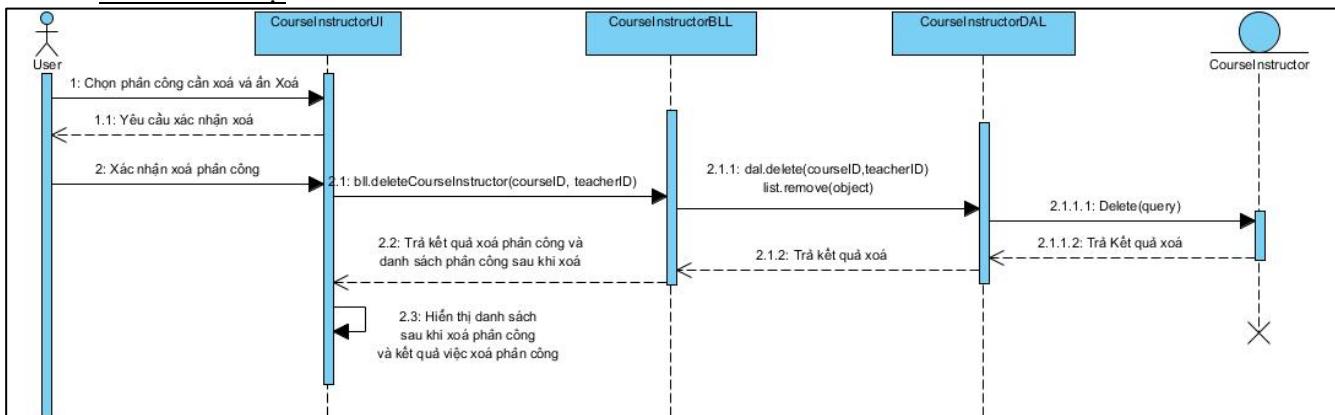
```

private void RefreshDataBase(String orderby) throws Exception {
    try {
        bll.loadDSCourseInstructor(orderby);
        insertHeader();
        outModel(model, courseinstructor: CourseInstructorBLL.getListCourseInstructor());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(parentComponent: this, message: "Không Thể Load Database",
            title: "Thông Báo Lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```

### 2.3.5. Xử lý 4: Xoá phân công giảng dạy

#### ❖ Sơ đồ tuần tự



#### ▪ Code class DAL

```

public void delete(int courseID,int teacherID)
{
    try {
        this.Delete( tableName:"courseinstructor",
        "CourseID ='"+courseID+"'AND PersonID ='"+teacherID+"'");
    }
    catch (Exception e) {
        System.out.println( x:"Lỗi không thể xóa courseinstructor item !!!");
    }
}

```

- Code class BLL

```

public void deleteCourseInstructor(int courseID,int teacherID) throws Exception{
    for(CourseInstructor csin : listCourseInstructor )
    {
        if(csin.getCourseID()==courseID)
        {
            try {
                dal.delete(courseID,teacherID);
                listCourseInstructor.remove( o:csin); // xoá trong arraylist
            } catch (Exception e) {
                System.out.println( x:"Khong the Xoa CourseInstructor vao database !!!");
            }
            return;
        }
    }
}

```

- Code class UI

```

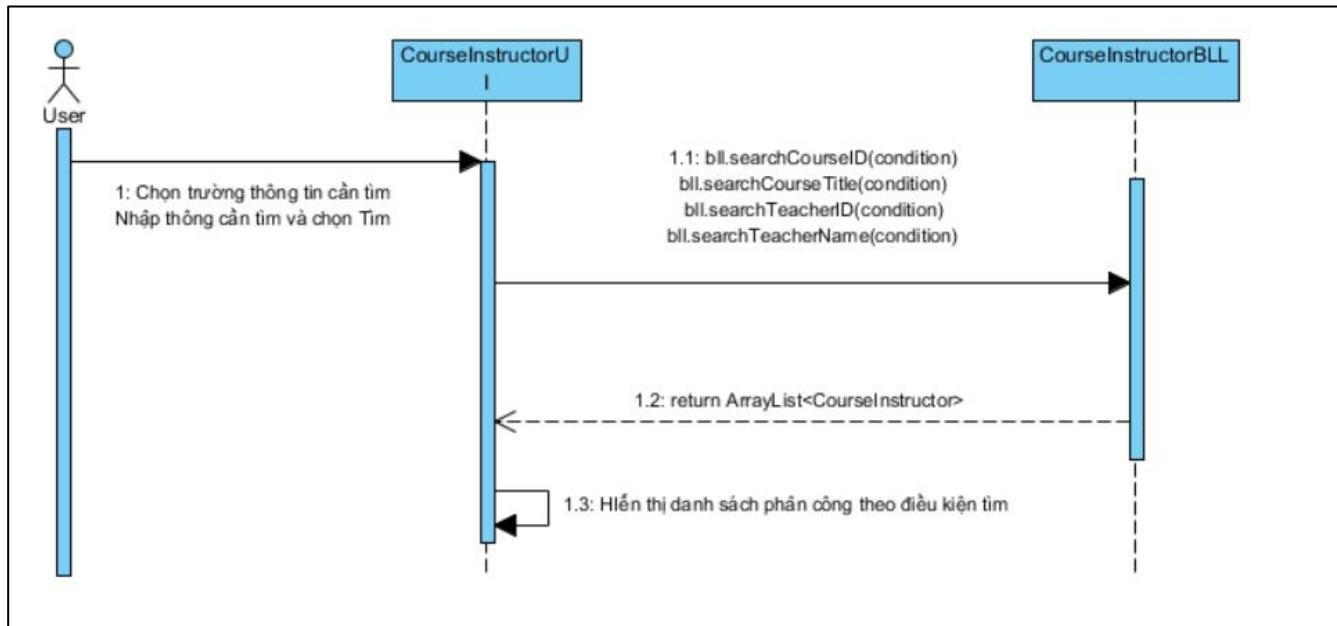
private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    int confirm = JOptionPane.showConfirmDialog( parentComponent: null, message: "Bạn Thực Sự Muốn Xóa Phân Công Này ?",
        title: "Thông Báo", optionType: JOptionPane.YES_NO_OPTION);
    if (confirm == 0)
        try {
            int teacherID = Integer.parseInt( s:txTeacher.getText());
            int courseID = Integer.parseInt( s:txCourse.getText());
            bll.deleteCourseInstructor(courseID, teacherID); //gọi Layer BLL xoá phân công
            insertHeader(); // chèn header cho table
            outModel(model, courseinstructor:CourseInstructorBLL.getListCourseInstructor()); // đổ data vào table
            clearInput();

        } catch (Exception ex) {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Không thể xóa CourseInstructor ",
                title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
        } else
            return;
    }
}

```

### 2.3.6. Xử lý 5: Lọc & Tìm kiếm phân công giảng dạy

- ❖ Sơ đồ tuần tự



#### ▪ Code class BLL

```

public ArrayList<CourseInstructor> searchCourseID (int courseID)
{
    ArrayList<CourseInstructor> search = new ArrayList<>();
    for(CourseInstructor csin : listCourseInstructor)
    {
        if( csin.getCourseID()==courseID )
            search.add( e:csin);
    }
    return search;
}

public ArrayList<CourseInstructor> searchCourseTitle (String courseTitle)
{
    ArrayList<CourseInstructor> search = new ArrayList<>();
    for(CourseInstructor csin : listCourseInstructor)
    {
        if( csin.getTitleCourse().contains( s:courseTitle) )
            search.add( e:csin);
    }
    return search;
}
  
```

```

public ArrayList<CourseInstructor> searchTeacherName (String teacherName)
{
    ArrayList<CourseInstructor> search = new ArrayList<>();
    for(CourseInstructor csin : listCourseInstructor)
    {
        if( csin.getTeacherName().contains( s:teacherName) )
            search.add( e:csin);
    }
    return search;
}

public ArrayList<CourseInstructor> searchTeacherID (int teacherID)
{
    ArrayList<CourseInstructor> search = new ArrayList<>();
    for(CourseInstructor csin : listCourseInstructor)
    {
        if( csin.getPersonID()==teacherID )
            search.add( e:csin);
    }
    return search;
}

```

- *Code class UI*

```

private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultTableModel temp = new DefaultTableModel();
    ArrayList<CourseInstructor> search = new ArrayList<>();
    Vector header = new Vector();
    header.add( e:"STT");
    header.add( e:"Mã Khoa Học");
    header.add( e:"Khoa Học");
    header.add( e:"Mã Giảng Viên");
    header.add( e:"Tên Giảng Viên");
    String optionSearch=cbSelectSearch.getSelectedItem().toString();
    try {
        if(!txtSearch.getText().isEmpty()){
            String inputSearch=txtSearch.getText();
            switch(optionSearch){
                case "Mã Khoa Học":
                    search=bll.searchCourseID( courseID: Integer.parseInt( s:inputSearch));
                    break;
                case "Mã Giảng Viên":
                    search=bll.searchTeacherID( teacherID: Integer.parseInt( s:inputSearch));
                    break;
                case "Tên Khoa Học":
                    search=bll.searchCourseTitle( courseTitle:inputSearch);
                    break;
                case "Tên Giảng Viên":
                    search=bll.searchTeacherName ( teacherName:inputSearch);
                    break;
                default:
                    break;
            }
        }
    }

```

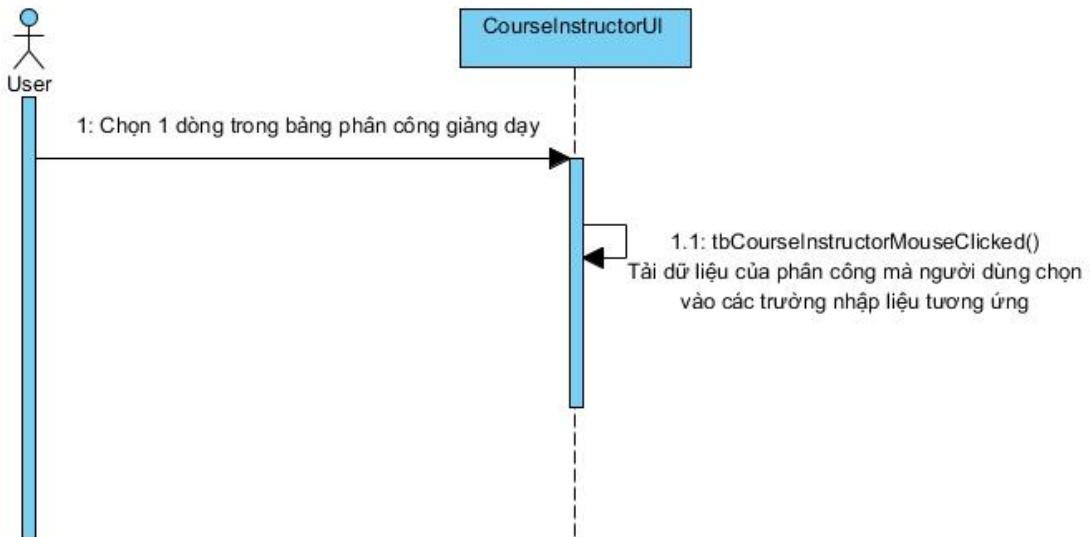
```

        else
            JOptionPane.showMessageDialog( parentComponent: this, message: "Vui lòng nhập điều kiện tìm",
                title: "Thông Báo", messageType: JOptionPane.ERROR_MESSAGE);
        } catch (Exception e) {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Không Thể Tìm Kiếm",
                title: "Thông Báo Lỗi", messageType: JOptionPane.ERROR_MESSAGE);
        }
        if (search != null && search.size() > 0) {
            if (temp.getRowCount() == 0) {
                temp = new DefaultTableModel( columnNames:header, rowCount:0);
            }
            for (int i = 0; i < search.size(); i++) {
                Vector row = new Vector();
                row.add( e:search.get( index:i ).getID());
                row.add( e:search.get( index:i ).getCourseID());
                row.add( e:search.get( index:i ).getTitleCourse());
                row.add( e:search.get( index:i ).getPersonID());
                row.add( e:search.get( index:i ).getTeacherName());
                temp.addRow( rowData:row );
            }
            tbCourseInstructor.setModel( dataModel:temp );
        }
    }
}

```

### 2.3.7. Xử lý 6: Hiển thị 1 dòng (record)

#### ❖ Sơ đồ tuần tự



#### ▪ Code class UI

```

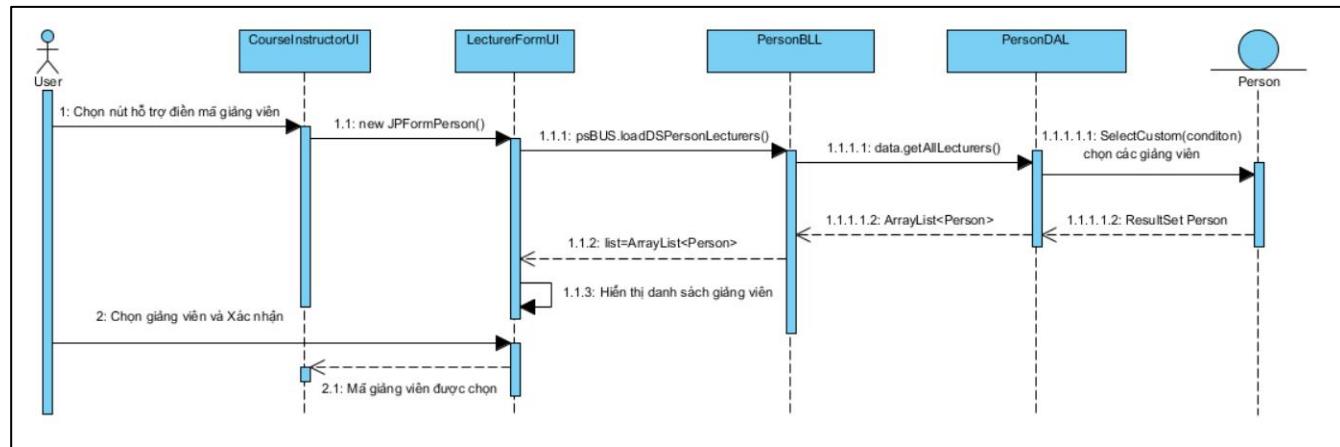
private void tbCourseInstructorMouseClicked(java.awt.event.MouseEvent evt) {
    int i = tbCourseInstructor.getSelectedRow();
    if (i >= 0) {
        if (tbCourseInstructor.getRowSorter() != null) {
            i = tbCourseInstructor.getRowSorter().convertRowIndexToModel(i);
        }
        txID.setText(t:tbCourseInstructor.getModel().getValueAt(rowIndex:i, columnIndex:0).toString());
        txCourse.setText(t:tbCourseInstructor.getModel().getValueAt(rowIndex:i, columnIndex:1).toString());
        txTeacher.setText(t:tbCourseInstructor.getModel().getValueAt(rowIndex:i, columnIndex:3).toString());
    }
}

```

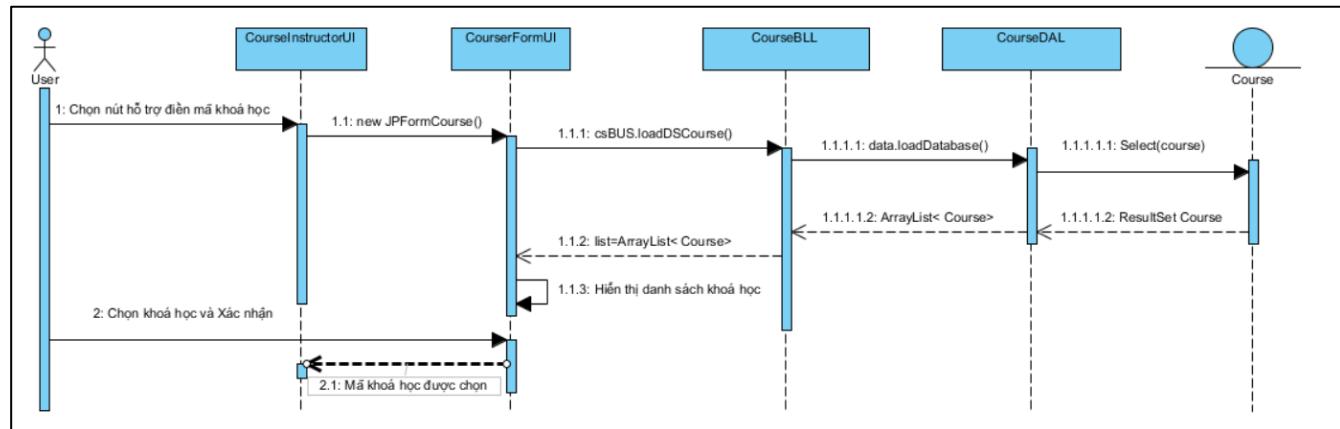
### 2.3.8. Xử lý 7: Hiển thị bảng chọn nhanh (1 cột)

#### ❖ Sơ đồ tuần tự

##### PersonID



##### CourseID:



#### ❖ Với Chọn PersonID:

- Code class DAL

```

public ArrayList<Person> getAllLecturers() throws Exception {
    ArrayList<Person> listLecturers = new ArrayList<>();
    try {
        ResultSet rs = this.SelectCustom( tableName: "person as ps",
            Custom: "ps.PersonID,ps.Lastname, ps.Firstname,ps.HireDate,ps.EnrollmentDate",
            condition: "ps.HireDate IS NOT NULL ORDER BY PersonID DESC");
        while(rs.next())
        {
            Person psDTO=new Person(
                PersonID: rs.getInt( columnLabel: "PersonID" ), Lastname: rs.getString( columnLabel: "Lastname" ),
                Firstname: rs.getString( columnLabel: "Firstname" ), HireDate: rs.getDate( columnLabel: "HireDate" ),
                EnrollmentDate: rs.getDate( columnLabel: "EnrollmentDate" )
            );
            listLecturers.add( e:psDTO );
        }
        rs.close();
        this.Close(); //đóng kết nối
    } catch (SQLException ex) {
        System.out.println( x: "Không thể load danh sách Giảng Viên: " );
    }
    return listLecturers;
}

```

- Code class BLL

```

public void loadDSPersonLecturers() throws Exception {
    if (listPersonLecturers == null) {
        listPersonLecturers = new ArrayList<Person>();
    }
    listPersonLecturers = data.getAllLecturers(); // gọi layer DAL đọc Database
}

```

- Code class UI

1. UI Quản lý phân công – Nút hỗ trợ điền mã giảng viên

Chức năng chọn mã giảng viên thông qua hộp thoại danh sách giảng viên.

```

555     private void btnSelectTeacherActionPerformed(java.awt.event.ActionEvent evt) {
556         JPFormPerson tbps;
557         try {
558             tbps = new JPFormPerson();
559             int personID = tbps.getPersonID(); // lấy ID Giảng viên từ Form chọn giảng viên
560             txTeacher.setText( t:String.valueOf(i:personID));
561         } catch (Exception ex) {
562             JOptionPane.showMessageDialog( parentComponent: this, message: "Không thể tải lên danh sách giảng viên ",
563                                         title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
564         }
565     }
566

```

Người dùng ấn vào nút hỗ trợ điền mã giảng viên. Hệ thống xuất hiện danh sách giảng viên sau đó người dọn giảng viên và ấn Xác Nhận. Mã giảng viên sẽ được truyền vào trường mã giảng viên trong phần nhập liệu phân công giảng dạy.

2. UI Form hỗ trợ điền mã giảng viên

```

29     public JPFormPerson() throws Exception {
30         setModal( modal:true);
31         initComponents();
32         txTim.requestFocus();
33          this.LoadListLecturers();
34         setVisible( b:true);
35     }

```

```

public void LoadListLecturers() throws Exception {
    PersonBLL psBUS = new PersonBLL();
    Vector header = new Vector();
    header.add( e:"TeacherID ");
    header.add( e:"FirstName");
    header.add( e:"LastName");
    header.add( e:"HireDate");
    if (model.getRowCount() == 0) {
        model = new DefaultTableModel( columnNames:header, rowCount:0);
    }
    if (psBUS.getListPersonLecturers() == null) {
        psBUS.loadDSPersonLecturers(); // lấy danh sách giảng viên từ bảng Person
    }
    ArrayList<Person> listPerson = new ArrayList<>();
    listPerson = psBUS.getListPersonLecturers();
    outModel(model, listPerson); // truyền data vào table
}

```

```

public void outModel(DefaultTableModel model, ArrayList<Person> listPerson) {
    Vector row;
    model.setRowCount( rowCount:0 );
    for (Person ps : listPerson) {
        row = new Vector();
        row.add( e:ps.getPersonID());
        row.add( e:ps.getFirstrname());
        row.add( e:ps.getLastname());
        row.add( e:ps.getHireDate());
        model.addRow( rowData:row );
    }
    tbPerson.setModel( dataModel:model );
}

```

(Dòng 558- Hàm `btnSelectTeacherActionPerformed` gọi đến)  
Hàm lấy mã Giảng viên sau khi chọn xác nhận.

```

public int getPersonID() {
    int personID = Integer.parseInt( s:txPersonID.getText() );
    return personID;
}

```

## ❖ Với Chọn CourseID:

### ▪ Code class DAL

```
public ArrayList<Course> loadDatabase() throws Exception
{
    ArrayList<Course> listCourse = new ArrayList<>();
    try {
        ResultSet rs = this.Select(tableName: "course");
        while(rs.next())
        {
            Course csin = new Course(
                CourseID:rs.getInt(columnLabel: "CourseID") , title:rs.getString(columnLabel: "Title"),
                Credits: rs.getInt(columnLabel: "Credits") , DepartmentID: rs.getInt(columnLabel: "DepartmentID"));
            listCourse.add(e:csin);
        }
        rs.close();
        this.Close(); //đóng kết nối;
    } catch (SQLException ex) {
        System.out.println(x: "Không thể load database Course");
    }
    return listCourse;
}
```

### ▪ Code class BLL

```
public void loadDSCourse() throws Exception{
    if(listCourse==null) listCourse = new ArrayList<Course>();
    listCourse=data.loadDatabase(); // gọi Layer DAL hàm đọc data từ CSDL
}
```

### ▪ Code class UI

Chức năng chọn mã khoá học thông qua hộp thoại danh sách khoá học :

#### 1. UI Quản lý phân công – Nút hỗ trợ điền mã khoá học



```
private void btnSelectCourseActionPerformed(java.awt.event.ActionEvent evt) {
    JPFFormCourse tbcs;
    try {
        tbcs = new JPFFormCourse();
        int courseID = tbcs.getCourseID();
        txCourse.setText(t: String.valueOf(i:courseID));
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(parentComponent: this, message: "Không thể tải lên danh sách khoá học",
            title: "Thông báo lỗi", messageType: JOptionPane.ERROR_MESSAGE);
    }
}
```

Người dùng ấn vào nút hỗ trợ điền mã khoá học. Hệ thống xuất hiện danh sách khoá học sau đó người dọn khoá học và ấn Xác Nhận. Mã khoá học sẽ được truyền vào trường mã khoá học trong phần nhập liệu phân công giảng dạy.

#### 2. UI Form Hỗ trợ điền mã khoá học

```

29     public JPFFormCourse() throws Exception {
30         setModal( modal:true );
31         initComponents();
32         txTim.requestFocus();
33         this.LoadDSCourse();
34         setVisible( b:true );
35     }
36

```

```

public void LoadDSCourse() throws Exception {
    CourseBLL csBUS = new CourseBLL();
    Vector header = new Vector();
    header.add( e:"CourseID" );
    header.add( e:"Title" );
    header.add( e:"Credits" );
    header.add( e:"Department" );
    if (model.getRowCount() == 0) {
        model = new DefaultTableModel( columnNames:header, rowCount: 0 );
    }
    if (csBUS.getListCourse() == null) {
        csBUS.loadDSCourse(); // lấy dữ liệu các khóa học
    }
    ArrayList<Course> listCourse = new ArrayList<>();
    listCourse = csBUS.getListCourse();
    outModel(model, listCourse); // truyền data vào table
}

```

```

public void outModel(DefaultTableModel model, ArrayList<Course> listCourse) {
    Vector row;
    model.setRowCount( rowCount: 0 );
    for (Course cs : listCourse) {
        row = new Vector();
        row.add( e:cs.getCourseID() );
        row.add( e:cs.getTitle() );
        row.add( e:cs.getCredits() );
        row.add( e:cs.getDepartmentID() );
        model.addRow( rowData:row );
    }
    tbCourse.setModel( dataModel:model );
}

```

Hàm lấy mã Khoa Học sau khi chọn xác nhận.

(Dòng 571- Hàm btnSelectCourseActionPerformed gọi đến)

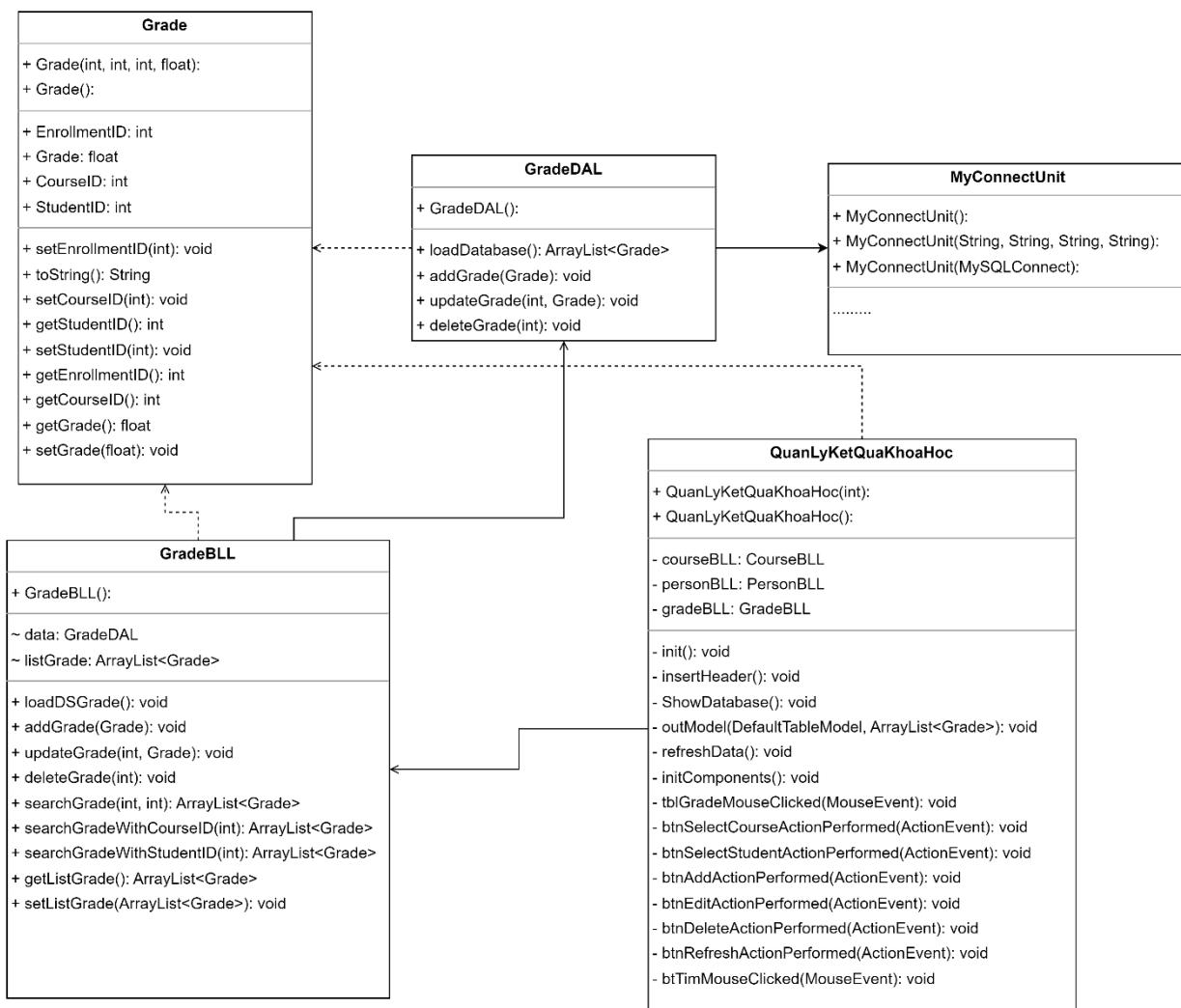
```

public int getCourseID() {
    int personID = Integer.parseInt( s:txCourseID.getText() );
    return personID;
}

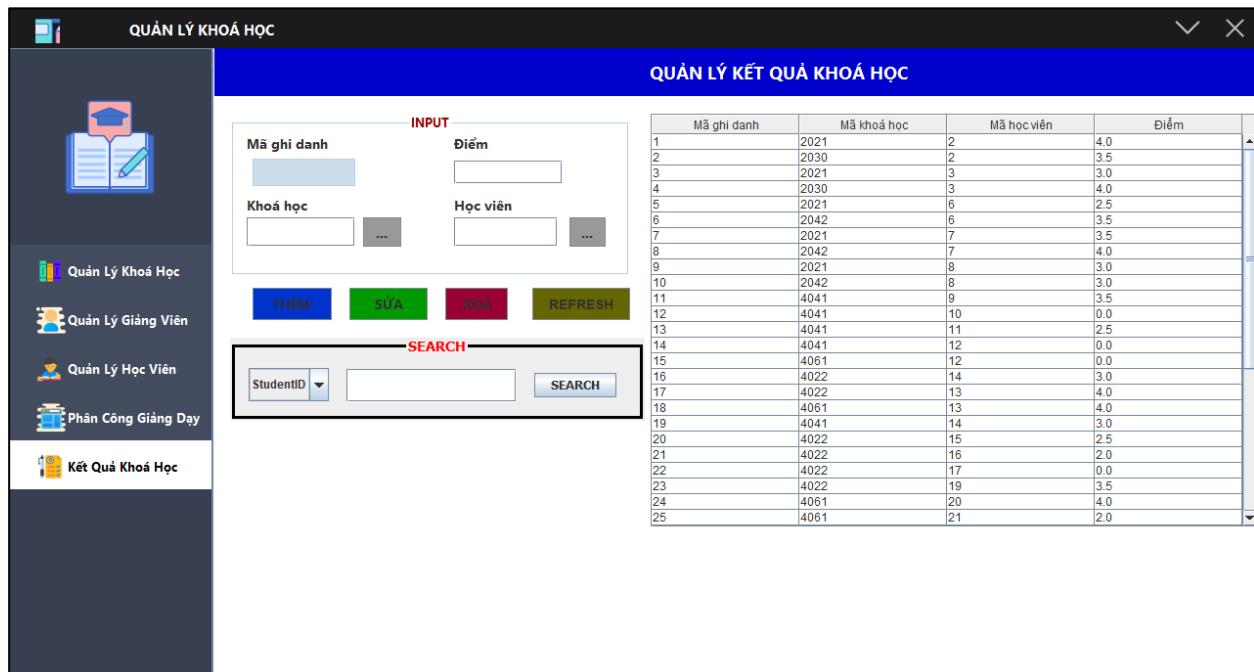
```

## 2.4. Chức Năng Quản Lý Kết Quả Khoa Học

### 2.4.1. Sơ đồ class chung



### GUI



## 1. Class GradeBLL

```
public class GradeBLL {  
  
    static ArrayList<GradeDTO> listGrade;  
    GradeDAL data = new GradeDAL();  
  
    public GradeBLL() {  
  
    }  
}
```

## 2. Class GradeDAL

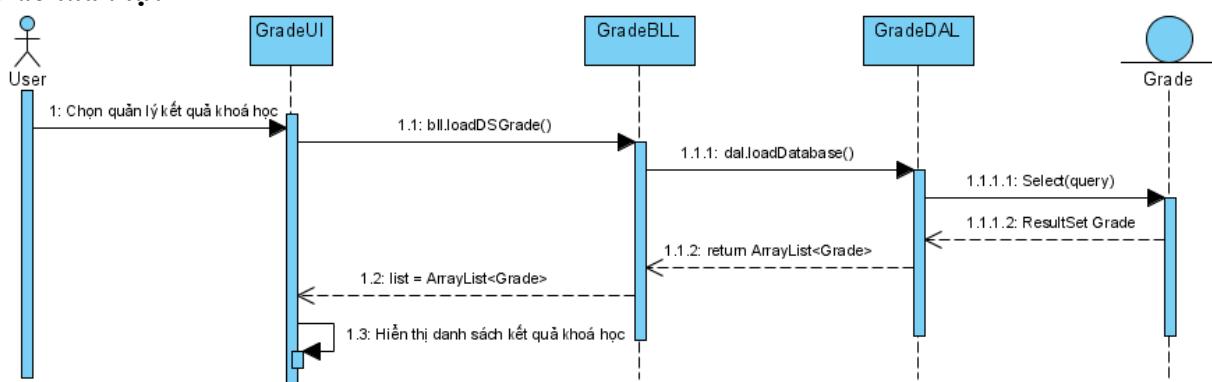
```
public class GradeDAL extends MyConnectUnit {  
  
    public GradeDAL() {  
        super();  
    }  
}
```

## 3. Class QuanLyKetQuaKhoaHoc

```
public class QuanLyKetQuaKhoaHoc extends javax.swing.JPanel {  
  
    /**  
     * Creates new form QuanLyKetQuaKhoaHoc  
     */  
    private int DEFALUT_WIDTH;  
    private DefaultTableModel model;  
    private GradeBLL gradeBLL = new GradeBLL();  
    private CourseBLL courseBLL = new CourseBLL();  
    private PersonBLL personBLL = new PersonBLL();  
}
```

### 2.4.2. Xử lý 1: Hiển thị danh sách kết quả khoá học

Sơ đồ tuần tự:



Code:

DAL:

```

public ArrayList<Grade> loadDatabase() throws Exception {
    ArrayList<Grade> dsach = new ArrayList<>();
    try {
        ResultSet rs = this.Select("studentgrade");
        while (rs.next()) {
            Grade g = new Grade(rs.getInt("EnrollmentID"), rs.getInt("CourseID"),
                rs.getInt("StudentID"), rs.getFloat("Grade"));

            dsach.add(g);
        }
        rs.close();
        this.Close();
    } catch (SQLException ex) {
        System.out.println("Không thể load database StudentGrade: " + ex);
    }
    return dsach;
}

```

BLL:

```

public void loadDSGrade() throws Exception {
    if (listGrade == null) {
        listGrade = new ArrayList<Grade>();
    }
    listGrade = data.loadDatabase();
}

```

UI:

```

public QuanLyKetQuaKhoaHoc(int width) throws Exception {
    DEFALUT_WIDTH = width;
    initComponents();
    this.setSize(this.DEFALUT_WIDTH - 200, 750);
    // this.setVisible(true);
    init();

}

private void init() throws Exception {
    ShowDatabase();
}

private void ShowDatabase() throws Exception {
    try {
        if (GradeBLL.getListGrade() == null) {
            gradeBLL.loadDSGrade();
        }
        insertHeader();
        outModel(model, GradeBLL.getListGrade());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Không Thể Load Database ",
            "Thông Báo Lỗi", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void outModel(DefaultTableModel model, ArrayList<Grade> grade)
{
    Vector data;
    model.setRowCount(0);
    for (Grade g : grade) {
        data = new Vector();
        data.add(g.getEnrollmentID());
        data.add(g.getCourseID());
        data.add(g.getStudentID());
        data.add(g.getGrade());
        model.addRow(data);
    }
    tblGrade.setModel(model);
}

private void insertHeader() {
    Vector header = new Vector();
    header.add("Mã ghi danh");
    header.add("Mã khoá học");
    header.add("Mã học viên");
    header.add("Điểm");

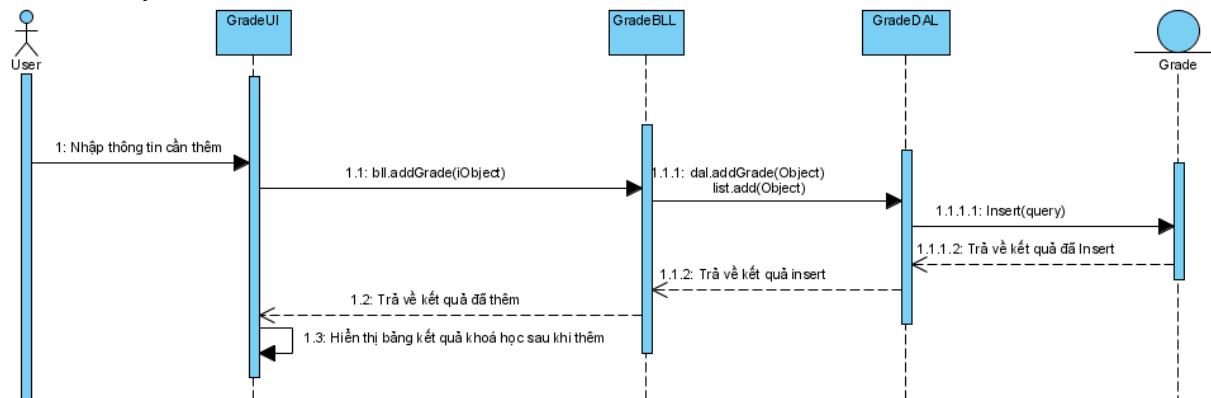
    //if (model.getRowCount() == 0)
    model = new DefaultTableModel(header, 0);
}

}

```

### 2.4.3. Xử lý 2: Thêm kết quả khoá học

Sơ đồ tuần tự:



Code:

DAL:

```

public void addGrade(Grade g) throws Exception {
    HashMap<String, Object> Insertvalues = new HashMap<String, Object>();
    Insertvalues.put("EnrollmentID", g.getEnrollmentID());
    Insertvalues.put("CourseID", g.getCourseID());
    Insertvalues.put("StudentID", g.getStudentID());
    Insertvalues.put("Grade", g.getGrade());

    try {
        this.Insert("studentgrade", Insertvalues);
    } catch (SQLException ex) {
        System.out.println("Khong the them Grade vao database !!!");
    }
}

```

BLL:

```

public void addGrade(Grade g) throws Exception {
    data.addGrade(g);
    listGrade.add(g);
}

```

UI:

```

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Grade g = new Grade();
        g.setCourseID((int) Integer.parseInt(txtCourse.getText()));
        g.setStudentID((int) Integer.parseInt(txtStuID.getText()));
        g.setGrade((float) Float.parseFloat(txtGrade.getText()));

        gradeBLL.addGrade(g);
        refreshData();

    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Can't create Grade ",
                                   "Error Message", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void refreshData() {
    try {
        gradeBLL.loadDSGrade();
        insertHeader();
        outModel(model, GradeBLL.getListGrade());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Không Thể Load Database ",
                                   "Thông Báo Lỗi", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void outModel(DefaultTableModel model, ArrayList<Grade> grade)
{
    Vector data;
    model.setRowCount(0);
    for (Grade g : grade) {
        data = new Vector();
        data.add(g.getEnrollmentID());
        data.add(g.getCourseID());
        data.add(g.getStudentID());
        data.add(g.getGrade());
        model.addRow(data);
    }
    tblGrade.setModel(model);
}

private void insertHeader() {
    Vector header = new Vector();
    header.add("Mã ghi danh");
    header.add("Mã khoá học");
    header.add("Mã học viên");
    header.add("Điểm");

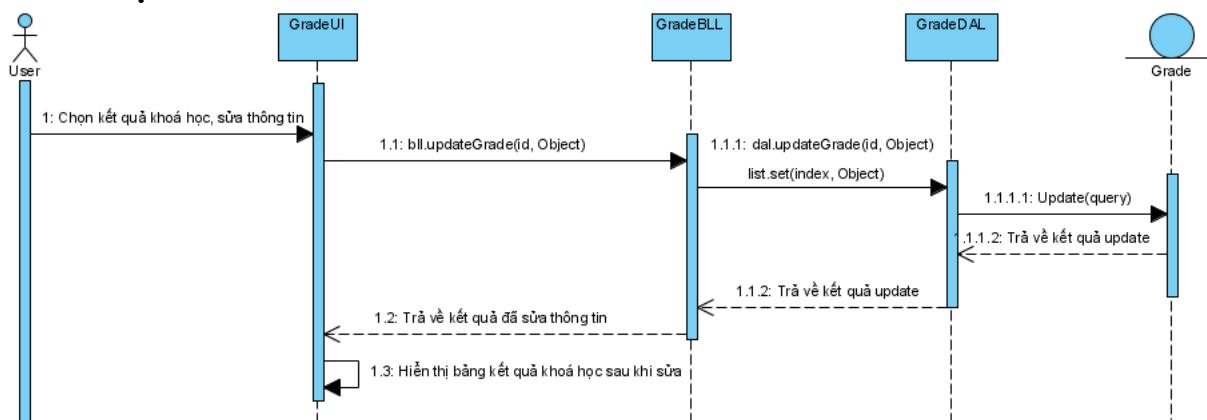
    //if (model.getRowCount() == 0)
    model = new DefaultTableModel(header, 0);
}

}

```

#### 2.4.4. Xử lý 3: Sửa kết quả khoá học

Sơ đồ tuần tự:



Code:

DAL:

```

public void updateGrade(int id, Grade g) throws Exception {
    HashMap<String, Object> Updatevalues = new HashMap<String, Object>();
    Updatevalues.put("EnrollmentID", g.getEnrollmentID());
    Updatevalues.put("CourseID", g.getCourseID());
    Updatevalues.put("StudentID", g.getStudentID());
    Updatevalues.put("Grade", g.getGrade());
    try {
        this.Update("studentgrade", Updatevalues, "EnrollmentID='" + id + "'");
    } catch (SQLException ex) {
        System.out.println("Khong the cap nhat Grade vao database !!!");
    }
}

```

BLL:

```

public void updateGrade(int id, Grade g) throws Exception {
    for (int i = 0; i < listGrade.size(); i++) {
        if (listGrade.get(i).getEnrollmentID() == g.getEnrollmentID()) {
            try {
                listGrade.set(i, g);
                data.updateGrade(id, g);
            } catch (Exception e) {
                System.out.println("Khong the cap nhat Grade vao database !!!");
            }
            return;
        }
    }
}

```

UI:

```

private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int eid = Integer.parseInt(txtEID.getText());
        int cid = Integer.parseInt(txtCourse.getText());
        int sid = Integer.parseInt(txtStuID.getText());
        float grade = Float.parseFloat(txtGrade.getText());
        Grade g = new Grade();
        g.setEnrollmentID((int) eid);
        g.setCourseID((int) cid);
        g.setStudentID((int) sid);
        g.setGrade((float) grade);
        gradeBLL.updateGrade(eid, g);
        reloadData();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Can't update Grade ",
                "Error Message", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

private void refreshData() {
    try {
        gradeBLL.loadDSGrade();
        insertHeader();
        outModel(model, GradeBLL.getListGrade());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Không Thể Load Database",
            "Thông Báo Lỗi", JOptionPane.ERROR_MESSAGE);
    }
}

private void outModel(DefaultTableModel model, ArrayList<Grade> grade)
{
    Vector data;
    model.setRowCount(0);
    for (Grade g : grade) {
        data = new Vector();
        data.add(g.getEnrollmentID());
        data.add(g.getCourseID());
        data.add(g.getStudentID());
        data.add(g.getGrade());
        model.addRow(data);
    }
    tblGrade.setModel(model);
}

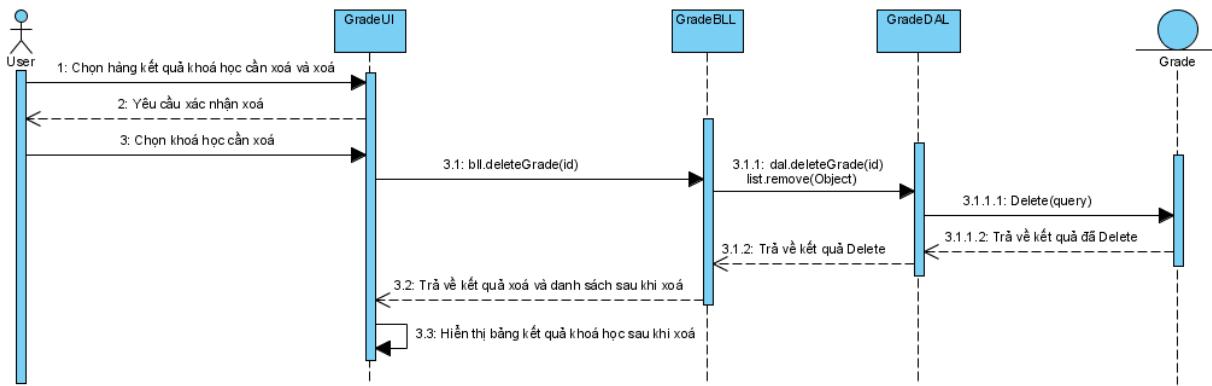
private void insertHeader() {
    Vector header = new Vector();
    header.add("Mã ghi danh");
    header.add("Mã khoá học");
    header.add("Mã học viên");
    header.add("Điểm");

    //if (model.getRowCount() == 0)
    model = new DefaultTableModel(header, 0);
}

```

#### 2.4.5. Xử lý 4: Xoá kết quả khoá học

*Sơ đồ tuần tự:*



Code:

DAL:

```

public void deleteGrade(int enrollmentID) {
    try {
        this.Delete("studentgrade", "EnrollmentID=" + enrollmentID + "'");
    } catch (Exception e) {
        System.out.println("Khong the xoa Grade vao database !!!");
    }
}
  
```

BLL:

```

public void deleteGrade(int enrollmentID) throws Exception {
    for (Grade g : listGrade) {
        if (g.getEnrollmentID() == enrollmentID) {
            try {
                listGrade.remove(g);
                data.deleteGrade(enrollmentID);
            } catch (Exception e) {
                System.out.println("Khong the xoa Grade vao database");
            }
            return;
        }
    }
}
  
```

UI:

```

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    int confirm = JOptionPane.showConfirmDialog(null, "Bạn có chắc chắn muốn xoá kết quả này?", "Thông báo", JOptionPane.YES_NO_OPTION);
    if (confirm == 0) {
        try {
            int eid = Integer.parseInt(txtEID.getText());
            gradeBLL.deleteGrade(eid);
            reloadData();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Can't delete Grade ", "Error Message", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        return;
    }
}
  
```

```

private void refreshData() {
    try {
        gradeBLL.loadDSGrade();
        insertHeader();
        outModel(model, GradeBLL.getListGrade());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Không Thể Load Database",
            "Thông Báo Lỗi", JOptionPane.ERROR_MESSAGE);
    }
}

private void outModel(DefaultTableModel model, ArrayList<Grade> grade)
{
    Vector data;
    model.setRowCount(0);
    for (Grade g : grade) {
        data = new Vector();
        data.add(g.getEnrollmentID());
        data.add(g.getCourseID());
        data.add(g.getStudentID());
        data.add(g.getGrade());
        model.addRow(data);
    }
    tblGrade.setModel(model);
}

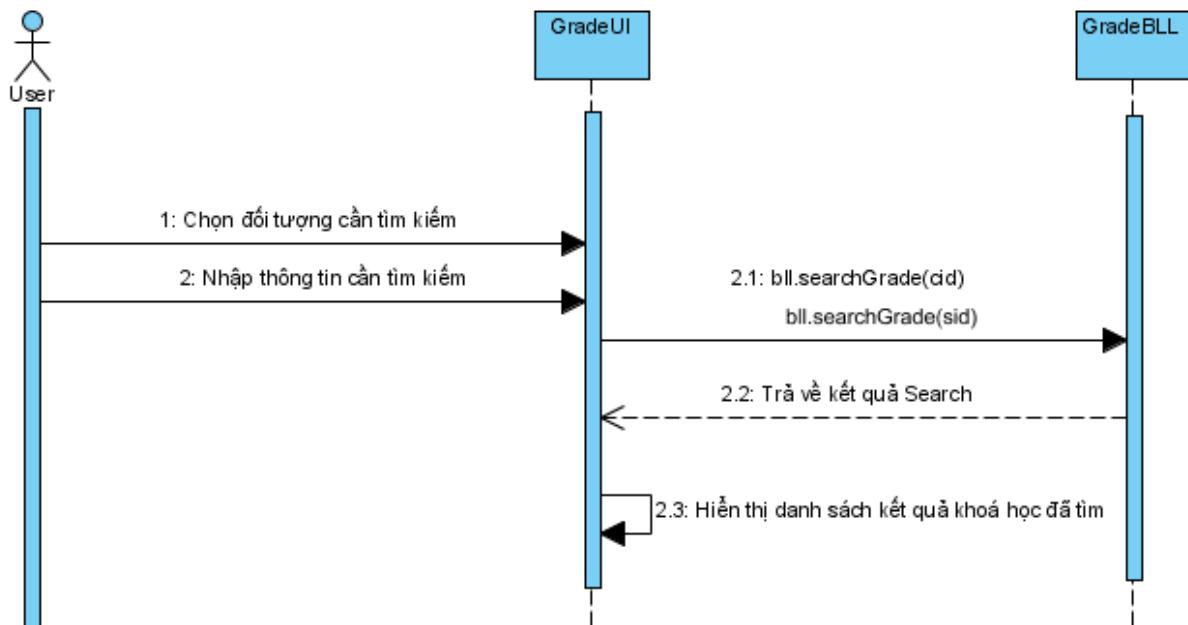
private void insertHeader() {
    Vector header = new Vector();
    header.add("Mã ghi danh");
    header.add("Mã khoá học");
    header.add("Mã học viên");
    header.add("Điểm");

    //if (model.getRowCount() == 0)
    model = new DefaultTableModel(header, 0);
}

```

#### 2.4.6. Xử lý 5: Lọc và tìm kiếm kết quả khoá học

*Sơ đồ tuần tự:*



Code:

BLL:

```

public ArrayList<Grade> searchGradeWithCourseID(int courseID) {
    ArrayList<Grade> search = new ArrayList<>();

    for (Grade g : listGrade) {
        if (g.getCourseID() == courseID) {
            search.add(g);
        }
    }
    return search;
}

public ArrayList<Grade> searchGradeWithStudentID(int studentID) {
    ArrayList<Grade> search = new ArrayList<>();

    for (Grade g : listGrade) {
        if (g.getStudentID() == studentID) {
            search.add(g);
        }
    }
    return search;
}
  
```

UI:

```

private void btTimMouseClicked(java.awt.event.MouseEvent evt) {
    String a = cbbTim.getSelectedItem().toString();
    ArrayList<Grade> ds = new ArrayList<>();
    ArrayList<Grade> dsl = new ArrayList<>();
    ds = gradeBLL.getListGrade();
    if ((a.equals("StudentID") && txTim.getText().isEmpty())
        || (a.equals("CourseID") && txTim.getText().isEmpty())) {
        outModel(model, ds);
    }
    else if (a.equals("StudentID")) {
        dsl = gradeBLL.searchGradeWithStudentID(Integer.parseInt(txTim.getText()));
        outModel(model, dsl);
    }
    else if (a.equals("CourseID")) {
        dsl = gradeBLL.searchGradeWithCourseID(Integer.parseInt(txTim.getText()));
        outModel(model, dsl);
    }
}

```

```

private void outModel(DefaultTableModel model, ArrayList<Grade> grade)
{
    Vector data;
    model.setRowCount(0);
    for (Grade g : grade) {
        data = new Vector();
        data.add(g.getEnrollmentID());
        data.add(g.getCourseID());
        data.add(g.getStudentID());
        data.add(g.getGrade());
        model.addRow(data);
    }
    tblGrade.setModel(model);
}

```

```

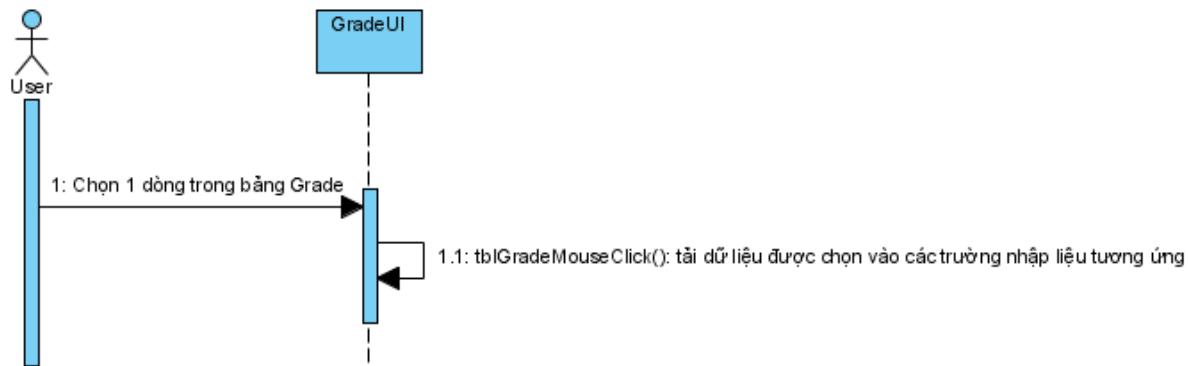
private void insertHeader() {
    Vector header = new Vector();
    header.add("Mã ghi danh");
    header.add("Mã khóa học");
    header.add("Mã học viên");
    header.add("Điểm");

    //if (model.getRowCount() == 0)
    model = new DefaultTableModel(header, 0);
}

```

#### 2.4.7. Xử lý 6: Hiển thị 1 dòng (record)

Sơ đồ tuần tự:



Code:

UI:

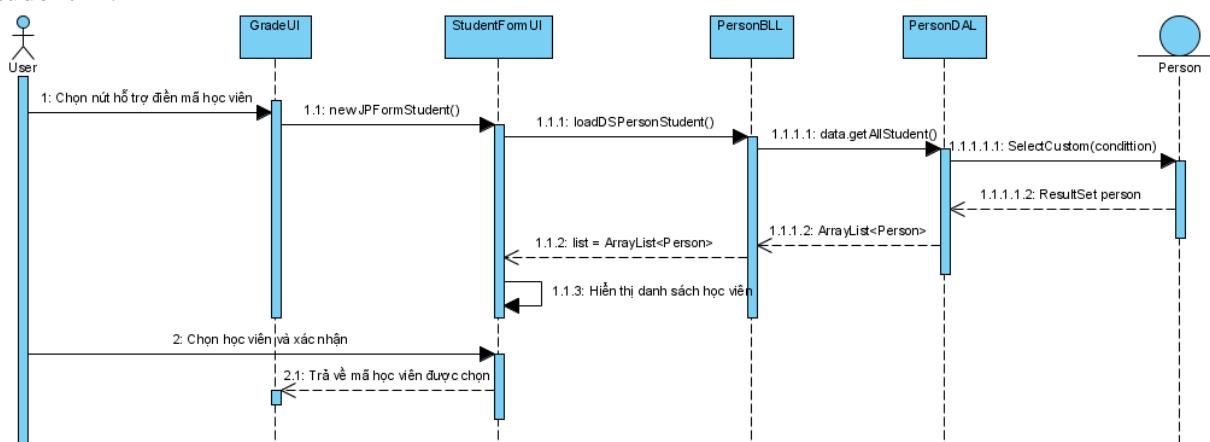
```

private void tblGradeMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int i = tblGrade.getSelectedRow();
    if (i >= 0) {
        if (tblGrade.getRowSorter() != null) {
            i = tblGrade.getRowSorter().convertRowIndexToModel(i);
        }
        txtEID.setText(tblGrade.getModel().getValueAt(i, 0).toString());
        txtCourse.setText(tblGrade.getModel().getValueAt(i, 1).toString());
        txtStuID.setText(tblGrade.getModel().getValueAt(i, 2).toString());
        txtGrade.setText(tblGrade.getModel().getValueAt(i, 3).toString());
    }
}
  
```

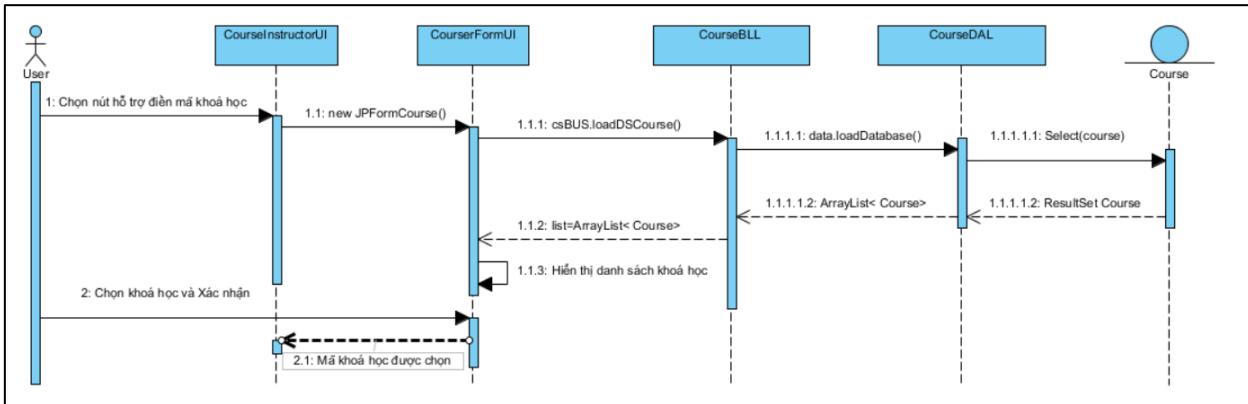
#### 2.4.8. Xử lý 7: Hiển thị bảng chọn nhanh

Sơ đồ tuần tự:

StudentID:



CourseID:



**Code:**

CourseID:

UI:

Chức năng chọn mã khoá học thông qua hộp thoại list Course

```

private void btnSelectCourseActionPerformed(java.awt.event.ActionEvent evt) {
    JPFormCourse tbcs;
    try {
        tbcs = new JPFormCourse();
        int courseID = tbcs.getCourseID();
        txtCourse.setText(String.valueOf(courseID));
    } catch (Exception ex) {
        Logger.getLogger(QuanLyKetQuaKhoaHoc.class.getName()).log(Level.SEVERE, null, ex);
    }
}
  
```

Đọc danh sách khoá học rồi tải lên Table và hàm lấy mã Course sau khi chọn Xác nhận

```

public void LoadDSCourse() throws Exception {
    Vector header = new Vector();
    header.add("CourseID ");
    header.add("Title");
    header.add("Credits");
    header.add("Department");
    if (model.getRowCount() == 0) {
        model = new DefaultTableModel(header, 0);
    }
    if (courseBLL.getListCourse() == null) {
        courseBLL.loadDSCourse("DESC");// lấy dữ liệu các khoá học
    }
    ArrayList<Course> listCourse = new ArrayList<>();
    listCourse = courseBLL.getListCourse();
    outModel(model, listCourse);// truyền data vào table
}
  
```

```

public void outModel(DefaultTableModel model, ArrayList<CourseDTO> listCourse) {
    Vector row;
    model.setRowCount(0);
    for (CourseDTO cs : listCourse) {
        row = new Vector();
        row.add(cs.getCourseID());
        row.add(cs.getTitle());
        row.add(cs.getCredits());
        row.add(cs.getDepartmentID());
        model.addRow(row);
    }
    tbKH.setModel(model);
}

```

Hàm lấy mã khoá học sau khi chọn xác nhận

```

public int getCourseID() {
    int personID = Integer.parseInt(txCourseID.getText());
    return personID;
}

```

BLL:

```

public void loadDSCourse(String orderby) throws Exception {

    if (listCourse == null) {
        listCourse = new ArrayList<Course>();
    }
    listCourse = data.loadDatabase(orderby); // gọi Layer DAL hàm đọc data từ CSDL
}

```

DAL:

```

public ArrayList<Course> loadDatabase() throws Exception
{
    ArrayList<Course> listCourse = new ArrayList<>();
    try {
        ResultSet rs = this.Select("course");
        while(rs.next())
        {
            Course csin = new Course(
                rs.getInt("CourseID") , rs.getString("Title"),
                rs.getInt("Credits") , rs.getInt("DepartmentID"));
            listCourse.add(csin);
        }
        rs.close();
        this.Close(); // đóng kết nối
    } catch (SQLException ex) {
        System.out.println("Không thể load database Course: "+ex);
    }

    return listCourse;
}

```

PersonID

Chức năng chọn mã học viên thông qua hộp thoại list Person.

```

private void btnSelectStudentActionPerformed(java.awt.event.ActionEvent evt) {
    JPFormStudent tbps;
    try {
        tbps = new JPFormStudent();
        int personID = tbps.getPersonID();
        txtStuID.setText(String.valueOf(personID));
    } catch (Exception ex) {
        Logger.getLogger(QuanLyKetQuaKhoaHoc.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Đọc

danh sách học viên rồi tải lên Table và hàm lấy mã học viên sau khi chọn Xác nhận

```

public void LoadDSKH() throws Exception {
    Vector header = new Vector();
    header.add("PersonID");
    header.add("FirstName");
    header.add("LastName");
    header.add("HireDate");
    header.add("EnrollmentDate");
    if (model.getRowCount() == 0) {
        model = new DefaultTableModel(header, 0);
    }
    if (psBLL.getListPersonStudent() == null) {
        psBLL.loadDSPersonStudent();
    }
    ArrayList<Person> listPerson = new ArrayList<>();
    listPerson = psBLL.getListPersonStudent();
    outModel(model, listPerson);
}

public void outModel(DefaultTableModel model, ArrayList<PersonDTO> listPerson) {
    Vector row;
    model.setRowCount(0);
    for (PersonDTO ps : listPerson) {
        row = new Vector();
        row.add(ps.getPersonID());
        row.add(ps.getFirsrname());
        row.add(ps.getLastname());
        row.add(ps.getHireDate());
        row.add(ps.getEnrollmentDate());
        model.addRow(row);
    }
    tbKH.setModel(model);
}

```

Hàm lấy mã học viên sau khi chọn xác nhận

```

public int getPersonID() {
    int personID = Integer.parseInt(txStudentID.getText());
    return personID;
}

```

BLL:

```

public void loadDSPersonStudent() throws Exception {
    if (listPersonStudent == null) {
        listPersonStudent = new ArrayList<Person>();
    }
    listPersonStudent = data.getAllStudent();
}

```

DAL:

```
public ArrayList<Person> getAllStudent() throws Exception {
    ArrayList<Person> dssach = new ArrayList<>();
    try {

        ResultSet rs = this.SelectCustom("person as ps",
            "ps.PersonID,ps.Lastname, ps.Firstname,ps.HireDate,ps.EnrollmentDate",
            "ps.EnrollmentDate IS NOT NULL ORDER BY PersonID DESC");
        while(rs.next())
        {
            Person psDTO=new Person(
                rs.getInt("PersonID"),rs.getString("Lastname"), rs.getString("Firstname")
                rs.getDate("HireDate"),rs.getDate("EnrollmentDate")
            );
            dssach.add(psDTO);
        }
        rs.close();
        this.Close(); //dong ket noi;

    } catch (SQLException ex) {
        System.out.println("Khong the load database Person: "+ex);
    }

    return dssach;
}
```

## Chương 3. SOURCE CODE KẾT NỐI CSDL, HƯỚNG DẪN CÀI ĐẶT CHƯƠNG TRÌNH VÀ LINK CHÚA SOURCE CODE ĐỒ ÁN

### 3.1. Hướng dẫn cài đặt chương trình

\*\*\* Các file source code nằm trong thư mục src/

1. Các thư viện mở rộng hỗ trợ nằm ở thư mục src/libs. (cần import đầy đủ các file .jar)
2. Tạo database “shool2” và import file “school2.sql” trong folder “db” vào phpadmin trên XAMPP .
3. Import tất cả các thư viện trong thư mục /libs/ .
4. Mở IDE NetBeans ( hoặc Eclipse ) để import project .
5. Build project sau đó tiến hành Run application để sử dụng chương trình.

### 3.2. Link Chứa Source Code Đồ Án

Link source code: [minhtrung0110/quanlykhoahoc \(github.com\)](https://github.com/minhtrung0110/quanlykhoahoc)

❖ Hết ❖