**Object-Oriented Analysis and Design (Spring, 2012)**

Student Information System (SIS)

**Đề tài: Xây dựng website quản lý nhà trọ**

**GVHD: ThS Võ Phương Bình**

**Danh sách thành viên**

# Menu

[Menu 2](#_Toc164519178)

[1. Hệ thống 5](#_Toc164519179)

[1.1. Vấn đề liên quan đến hệ thống 6](#_Toc164519180)

[1.2. System Context Diagram 7](#_Toc164519181)

[1.3. Tóm tắt tính năng hệ thống 7](#_Toc164519182)

[1.4. Sơ đồ Use Case 8](#_Toc164519183)

[1.5. Danh sách các Use Cases 8](#_Toc164519184)

[1.5.1. Đăng nhập 8](#_Toc164519185)

[1.5.2. ChuTro 9](#_Toc164519186)

[1.5.3. QuanTriVien 9](#_Toc164519187)

[1.6. Non-functional Requirements and Constraints 11](#_Toc164519188)

[1.7. Thuật ngữ 11](#_Toc164519189)

[1.8. Công nghệ và công cụ sử dụng 12](#_Toc164519190)

[2. Domain Class Model 12](#_Toc164519191)

[2.1. Liên kết giữa các thuộc tính 13](#_Toc164519192)

[2.2. Add Attributes 14](#_Toc164519193)

[2.3. Kiến trúc logic 15](#_Toc164519194)

[2. Class Design 16](#_Toc164519195)

[2.1. Use-Case Realizations with GRASP Patterns 16](#_Toc164519196)

[2.1.1. System Sequence Diagram (new) 16](#_Toc164519197)

[2.1.2. Operation Contract 20](#_Toc164519198)

[2.1.2.6. searchByStudentID (new) 23](#_Toc164519199)

[2.1.2.7. searchByName (new) 23](#_Toc164519200)

[2.1.2.8. viewStudentInformation 23](#_Toc164519201)

[2.1.2.9. enterPersonalInformation 24](#_Toc164519202)

[2.1.2.10. enterQualifying 25](#_Toc164519203)

[2.1.2.12. XoaThongTin 26](#_Toc164519204)

[2.1.2.13. showQualifying (new) 26](#_Toc164519205)

[2.1.2.14. enterCourse 26](#_Toc164519206)

[2.1.2.15. deleteCourse (new) 27](#_Toc164519207)

[2.1.2.16. updateCourse (new) 27](#_Toc164519208)

[2.1.2.17. showCourse (new) 27](#_Toc164519209)

[2.1.2.18. enterPaper 28](#_Toc164519210)

[2.1.2.19. deletePaper (new) 28](#_Toc164519211)

[2.1.2.20. updatePaper (new) 28](#_Toc164519212)

[2.1.2.21. showPaper (new) 29](#_Toc164519213)

[2.1.3. Trình tự hoạt động 29](#_Toc164519214)

[2.1.3.1. login 29](#_Toc164519215)

[2.1.3.2. enterPrimaryInformation 30](#_Toc164519216)

[2.1.3.3. deletePrimaryInformation (new) 31](#_Toc164519217)

[2.1.3.4. updatePrimaryInformation (new) 31](#_Toc164519218)

[2.1.3.5. showPrimaryInformation 31](#_Toc164519219)

[2.1.3.6. logout 32](#_Toc164519220)

[Tìm kiếm theo khoảng cách 32](#_Toc164519221)

[Tìm theo giá 33](#_Toc164519222)

[Xóa 35](#_Toc164519223)

[Cập nhật 35](#_Toc164519224)

[2.1.3.7. showQualifying (new) 36](#_Toc164519225)

[2.1.3.8. enterCourse 36](#_Toc164519226)

[4.2. Design Class Model 40](#_Toc164519227)

[5. Implementation Class Model 41](#_Toc164519228)

[5.1. Draw an implementation class diagram 41](#_Toc164519229)

[6. Kết quả chương trình 50](#_Toc164519230)

[6.1. Snapshots of system execution 50](#_Toc164519231)

[6.2. Source Code Listing 50](#_Toc164519232)

[6.2.1. Program.cs 50](#_Toc164519233)

[6.2.2. frmLogin.cs 50](#_Toc164519234)

[6.2.3. StudentInformation.cs 50](#_Toc164519235)

[7. Unit Testing 50](#_Toc164519236)

[7.1. Snapshots of testing result 50](#_Toc164519237)

[7.2. Unit Testing Code Listing 51](#_Toc164519238)

[7.2.1. AdvisorTest.cs 51](#_Toc164519239)

[7.2.2. ManagerTest.cs 53](#_Toc164519240)

# Hệ thống

|  |  |  |
| --- | --- | --- |
| **Revision** | **Description** | **Date** |
| **Iteration I** | | |
| Homework #1 | Problem statement  Software Environments | 19/3/2024 |
| Homework #2 | Tóm tắt các tính năng của hệ thống  Sơ đồ Use case  Ràng buộc phi chức năng | Mar 14, 2012 |
| Homework #3 | *Summary of System Features refinement*  *Non-functional Requirements and Constraints refinement*  Domain Class Diagram Add Associations  Add Attributes | Mar 28, 2012 |
| Homework #4 | *Cover pager refinement*  *Use Cases (UC3) refinement (by adding the* ***Qualifying*** *attribute of Extend Information)* Logical Architecture  System Sequence Diagram Operation Contract Operation Sequence Diagram  Design Class Model | April 16, 2012 |
| Homework #5 | *Logical Architecture refinement*  *System Sequence Diagram refinement* | April 26, 2012 |

|  |  |  |
| --- | --- | --- |
|  | *Operation Sequence Diagram refinement Design Class Model refinement* Implementation Class Model Programming  Unit Testing |  |
| **Iteration II** | | |
| Homework #6 | *Use Cases refinement. Classes Identified refinement. Bad Classes refinement.*  *Good Classes refinement.*  *Domain Class Diagram refinement* | May 22, 2012 |
| Homework#7 | *System Sequence Diagram refinement Operation Sequence Diagram refinement Design Class Model refinement Implementation Class Model refinement Programming refinement*  *Unit Testing refinement* | June 12, 2012 |

## Vấn đề liên quan đến hệ thống

Hệ thống nhà trọ Đà Lạt là một trang web cho phép

Student Information System (SIS) is a GUI-based application used in the Lab of the university. SIS is used for both advisor and students. The students can manage their personal information, courses, papers, and qualifying. The advisor can add students to the SIS and see their personal information. Therefore, the advisor can control the study progress of his/her students.

* Đăng nhập và đăng xuất khỏi hệ thống
* Cập nhật

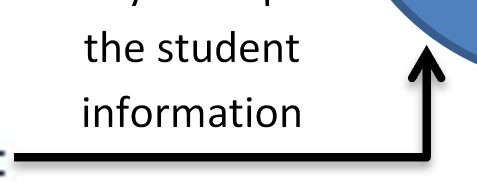
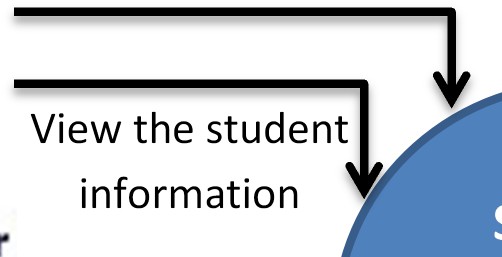
Đối tượng sinh viên có thể:

* Đăng nhập, đăng ký trong hệ thống

## System Context Diagram

View the student

information



Add, edit, and delete users to/in the SIS

Student

Information System (SIS)

Save data

**Database**

Modify and update

Load data

***Figure 1.1 – System Context Diagram***

the student

information

## Tóm tắt tính năng hệ thống

|  |  |
| --- | --- |
| **Feature ID** | **Feature Description** |
| **FEA-01** | **Đăng nhập và đăng ký** |
| **FEA-02** | **Đối tượng đến thuê: sinh viên có thể tìm kiếm thông tin, liên hệ với chủ trọ** |
| **FEA-03** | **User can be checked his/her role (as advisor or student) on**  **log in the system.** |
| **FEA-04** | **Advisor can find and choose a student to view the student**  **information and the study progress.** |
| **FEA-05** | **Student can modify, update one more the student**  **information such as courses, papers, qualifying, and so on.** |

## Sơ đồ Use Case

**A diagram of a network

Description automatically generated**

***Hình 1.2 - Use Case Diagram***

## Danh sách các Use Cases

### Đăng nhập

|  |  |
| --- | --- |
| **UC1: Đăng nhập** | |
| **Scope** | Student Information System (SIS). |
| **Level** | User-goal |
| **Primary Actor** | Advisor |
| **Stakeholders and Interests** | * SinhVien: mong muốn thông tin của họ được cập nhật khi đã quyết định đến thuê. |
| **Luồng sự kiện chính** | 1. Hệ thống hiển thị màn hình đăng nhập 2. Người dùng nhập username, password 3. Nếu thành công, hệ thống thể hiện việc đăng nhập đã hoàn tất 4. Kết thúc use case |
| **Luồng sự kiện phụ** | 1a Mật khẩu không hợp lệ (khi nhập sai username hoặc password   * Bắt nhập lại thông tin, đi kèm đó là thông báo nhập sai và yêu cầu nhập lại * Quay lại bước 2 trong luồng sự kiện chính   1b. Quên mật khẩu: khi người dùng chọn chức năng quên mật khẩu trên màn đăng nhập   * Hiển thị màn hình |

### ChuTro

|  |  |
| --- | --- |
| **Use Case UC2: ChuTro** | |
| **Scope** |  |
| **Level** | User-goal |
| **Primary Actor** | Advisor |

Hg

### QuanTriVien

|  |  |
| --- | --- |
| **Use Case UC3: QuanTriVien** | |
| **Scope** | Truy cập toàn bộ hệ thống |
| **Level** | User-goal |
| **Primary Actor** | SinhVien và ChuTro |
| **Stakeholders and Interests** | * Advisor: wants students modify the extend information of the student information exactly. * Students: wants to modify the extend information of the student information easily. |
| **Preconditions** | Student is identified and authenticated. |
| **Success Guarantee** | Student’s modified information is saved into database exactly. |
| **Main Success Scenario** | 1. Student starts a new transaction (log in). 2. Student selects and enters which information need to modify such as personal information, courses, papers, and qualifying. |

## Non-functional Requirements and Constraints

**Hghgg**

|  |  |  |
| --- | --- | --- |
| **NFR ID** | **Category** | **Description** |
| **NFA-01** | **Performance** | The student wants to complete the information  modifying easily and quickly. |
| **NFA-02** | **Usability** | The advisor and student will be worked with a clear display of the SIS. Therefore:   * Text should be easily visible (as large as possible). * Avoid colors associated with common forms   of color blindness. |
| **NFA-03** | **Reliability** | Advisor can see exactly information modified by  students. |
| **NFA-04** | **Adaptability** | SIS has a unique rule for all students: only modified  or update oneself; do not allow delete users. |

## Thuật ngữ

|  |  |
| --- | --- |
| **Term** | **Definition or Description** |
| **SinhVien** | Đối tượng sinh viên |
| **ChuTro** | Những người cho thuê trọ |
| **QuanTriVien** | Quản trị viên |
| **CSDL** | Cơ sở dữ liệu |
| **Save data** | Save advisor or student information into database. |
| **Load data** | Get advisor or student information from database. |
| **Assign role** | Assign the role to users as advisor or student. |

|  |  |
| --- | --- |
| **Study progress** | Includes courses and papers which student finished. |
| **Field** | An attribute of the student information (e.g. name,  papers, and so on). |
| **Criteria** | The requirements of a class (Master or Ph.D) which a  student must finish to get the degree relatively. |

## Công nghệ và công cụ sử dụng

* Dự án được viết bằng ngôn ngữ C#
* Môi trường phát triển: Microsoft Visual Studio 2022

# Domain Class Model

* 1. **Domain Class Diagram Showing Only Concepts List conceptual classes from UC1, UC2, and UC3:**

Advisor

Student Information

Qualifying

Class

Student

Primary Information

Course

Progress (new)

Login

Extend Information

Paper

Point (new)

Password

Study Progress

Criteria (new)

**List bad conceptual classes (\*):**

* **Login:** only is an operation of advisor or student.
* **Password:** only is an attribute of advisor or student.
* **Study Progress:** only is a few attributes of student information.
* **Class:** only is an attribute of Primary Information.
* **(new) Progress:** only is a few attributes of Extend Information

Advisor

Student Information

Qualifying

Class (\*)

Student

Primary Information

Course

**Progress(\*) (new)**

Login (\*)

Extend Information

Paper

**Point(\*) (new)**

Password (\*)

Study Progress (\*)

**Criteria (new)**

**List good conceptual classes:**

Advisor

Student Information

Qualifying

**Criteria (new)**

Student

Primary Information

Course

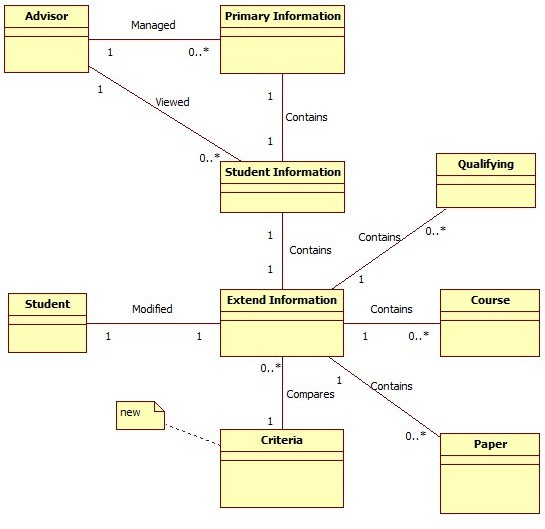
Extend Information

Paper

## Liên kết giữa các thuộc tính

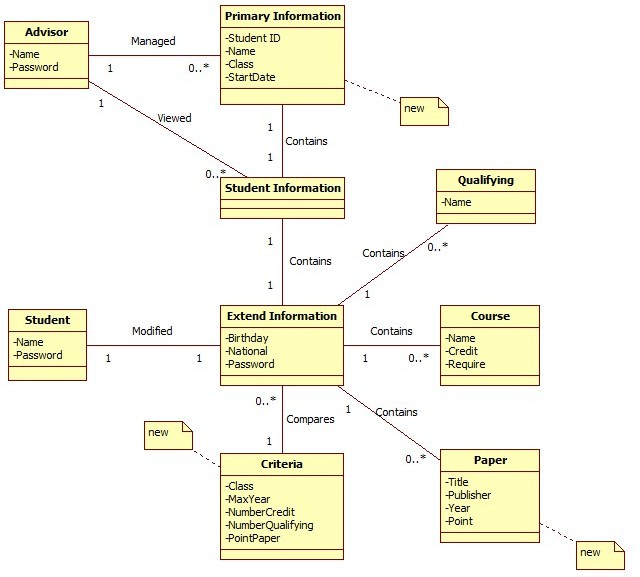
|  |
| --- |
| **Mô tả các quan hệ** |
| Một **sinh viên** có thể đến thuê tại một phòng trong một khoảng thời gian nhất định. |
| Một **chủ trọ** có thể cho nhiều sinh viên thuê tại một khu trọ  Có thể có nhiều khu trọ |
| One Student Information **Contains** only one Primary Information. |
| One Student Information **Contains** only one Extend Information. |

|  |
| --- |
| One Extend Information **Contains** zero or more than one Qualifying |
| One Extend Information **Compares** to one Criteria**.** One Criteria **is Compared**  by zero or more than one Extend Information. |



***Figure 2.1 Domain Class Model with Associations***

## Add Attributes

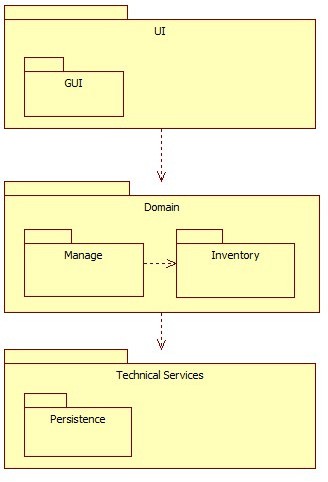


***Figure 2.2 Domain Class Model with Associations and Attributes***

## Kiến trúc logic

**Descriptions**

* **UI**
  + **GUI:** use Graphic User Interface (GUI) of the C# language to perform the User Interface (UI) of software.
  + **Manage:** the main package of software represents domain concepts for managing student information.
* **Domain**
  + **Inventory:** the package for accessing to a particular set of persistent data.
* **Technical Data**
* **Persistence:** the package that provide access to databases.

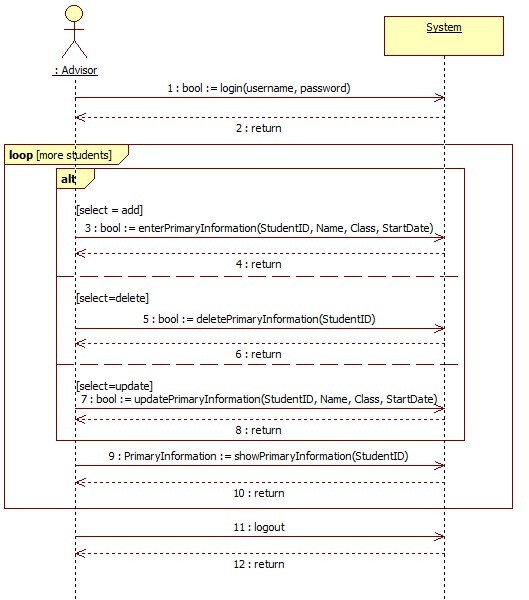


***Figure 3.1 Logical Architecture in UML package diagram***

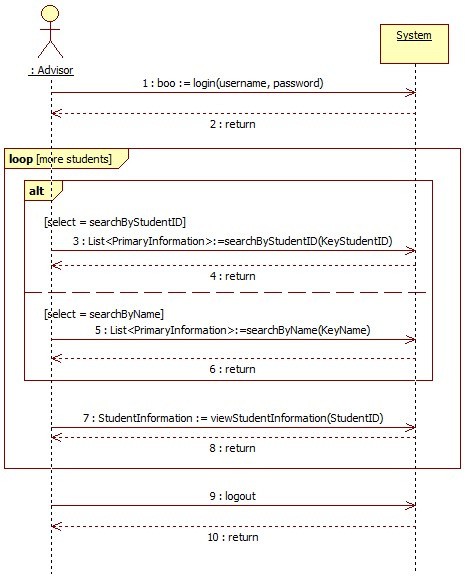
# Class Design

## Use-Case Realizations with GRASP Patterns

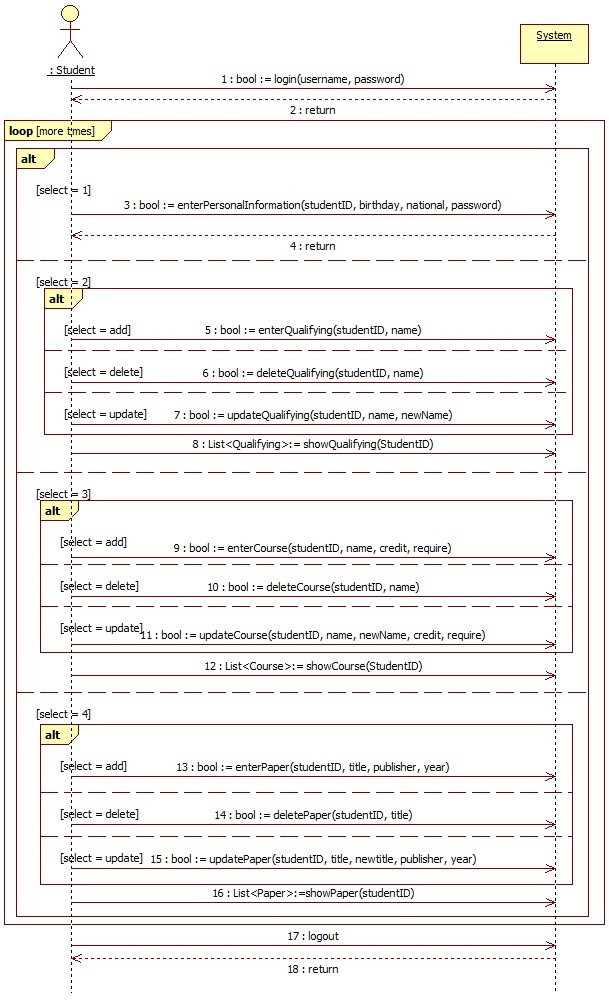
### System Sequence Diagram (new)



***Figure 4.1 Sequence Diagram of use case Manage Student***



***Figure 4.2 Sequence Diagram of use case View Information***



***Figure 4.3 Sequence Diagram of use case Manage Information***

### Operation Contract

|  |  |
| --- | --- |
| **Contract ID** | **Operation Name** |
| CO-01 | Đăng nhập |
| CO-02 | Đăng ký |
| CO-03 | deletePrimaryInformation (new) |
| CO-04 | updatePrimaryInformation (new) |
| CO-05 | showPrimaryInformation |
| CO-06 | Đăng xuất |
| CO-07 | TimTheoKhoangCach() |
| CO-08 | TimTheoGia() |
| CO-09 | TimTheoViTri() |
| CO-10 | enterPersonalInformation |
| CO-11 | enterCourse |
| CO-12 | deleteCourse (new) |
| CO-13 | updateCourse (new) |
| CO-14 | showCourse (new) |
| CO-15 | enterQualifying |
| CO-16 | deleteQualifying (new) |
| CO-17 | updateQualifying (new) |
| CO-18 | showQualifying (new) |
| CO-19 | enterPaper |
| CO-20 | ThemThongTin(new) |
| CO-21 | XoaThongTin(new) |
| CO-22 |  |

#### Đăng nhập

|  |  |
| --- | --- |
| **Operation** | login(userName: string, password: string) |
| **Cross References** | Use Case: Manage Student, View Information,  Manage Information |

|  |  |
| --- | --- |
| **Preconditions** | None |
| **Post-conditions** | * Một user (Sinh viên hoặc chủ cho thuê, quản trị). * Được liên kết với thông tin cho sinh viên was associated with Primary Information, Student Information (for Advisor) and   Extend Information (for Student). |

#### Đăng ký

|  |  |
| --- | --- |
| **Operation** | enterPrimaryInformation (studentID: string, name:  string, class: string, startDate: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The advisor instance was created (logined). |
| **Post-conditions** | * A Primary Information *pi* was created. * *pi.StudentID* became studentID. * *pi.Name* became name. * *pi.Class* became class. * A Student Information *si* was created and *si*   contains *pi*. |

#### deletePrimaryInformation

|  |  |
| --- | --- |
| **Operation** | deletePrimaryInformation (studentID: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The advisor instance was created (logined). |
| **Post-conditions** | A Primary Information *pi*, which has *pi.StudentID*  equal to studentID, is destroyed.  A Student Information *si,* which contains *pi,* is destroyed. |

|  |  |
| --- | --- |
| **Operation** | updatePrimaryInformation (studentID: string,  name: string, class: string, startDate: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The advisor instance was created (logined). |
| **Post-conditions** | A Primary Information *pi*, which has *pi.StudentID*  equal to studentID, is updated:   * *pi.StudentID* became studentID. * *pi.Name* became name. * *pi.Class* became class. * *pi.StartDate* became startDate.   A Student Information *si* was updated with *si*  contains *pi*. |

#### showPrimaryInformation

|  |  |
| --- | --- |
| **Operation** | showPrimaryInformation (studentID: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The Primary Information instances were created  and had the same studentID. |
| **Post-conditions** | None |

#### Đăng xuất

|  |  |
| --- | --- |
| **Operation** | LogOut(userName: string, password: string) |
| **Cross References** | Use case: |

|  |  |
| --- | --- |
| **Điều kiện tiên quyết** | User đang đăng nhập hệ thống |

# searchByStudentID (new)

|  |  |
| --- | --- |
| **Operation** | searchByStudentID (keyStudentID: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The Primary Information instances were created  and had the StudentID like keyStudentID. |
| **Post-conditions** | None |

# searchByName (new)

|  |  |
| --- | --- |
| **Operation** | searchByName (keyName: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The Primary Information instances were created  and had the Name like keyName. |
| **Post-conditions** | None |

# viewStudentInformation

|  |  |
| --- | --- |
| **Operation** | viewStudentInformation (studentID: string) |
| **Cross References** | Use Case: View Information |
| **Preconditions** | * The advisor instance was created (logined). * The Student Information instance was created and had the same studentID. |
| **Post-conditions** | None |

# enterPersonalInformation

|  |  |
| --- | --- |
| **Operation** | enterPersonalInformation (studentID: string,  birthday: string, national: string, password: string) |
| **Cross References** | Use Case: Manage Information |
| **Preconditions** | * The student instance was created (logined). * A Extend Information instance was created. |
| **Post-conditions** | * A Extend Information instance *ei* was created or updated. * *ei.Birthday* became birthday. * *ei.National* became national. * *ei.Password* became password. |

# enterQualifying

|  |  |
| --- | --- |
| **Operation** | enterQualifying (studentID: string, name: string) |
| **Cross References** | Use Case: Manage Information |
| **Preconditions** | * The student instance was created (logined). * A Extend Information instance was created. |
| **Post-conditions** | * A Qualifying instance *q* was created or updated with studentID. * *q.Name* became name. |

#### ThemThongTin

|  |  |
| --- | --- |
| **Operation** | deleteQualifying (studentID: string, name: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The student instance was created (logined). |
| **Post-conditions** | A Qualifying *qi*, which has *qi.StudentID* equal to studentID and *qi.Name* equal to name, is updated:  - *qi.StudentID* became studentID. |

|  |  |
| --- | --- |
|  | - *qi.Name* became newName. |

# XoaThongTin

|  |  |
| --- | --- |
| **Operation** | updateQualifying (studentID: string, name: string,  newName: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The student instance was created (logined). |
| **Post-conditions** | A Qualifying *qi*, which has *qi.StudentID* equal to studentID and *qi.Name* equal to name, is updated:   * *qi.StudentID* became studentID. * *qi.Name* became newName. |

# showQualifying (new)

|  |  |
| --- | --- |
| **Operation** | showQualifying (studentID: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The Qualifying instances were created and had the  same studentID. |
| **Post-conditions** | None |

# enterCourse

|  |  |
| --- | --- |
| **Operation** | enterCourse (studentID: string, name: string, credit:  number, require: boolean) |
| **Cross References** | Use Case: Manage Information |
| **Preconditions** | * The student instance was created (logined). * A Extend Information instance was created. |
| **Post-conditions** | - A Course instance *c* was created or updated |

|  |  |
| --- | --- |
|  | with studentID.   * *c.Name* became name. * *c.Credit* became credit. * *c.Require* became require. |

# deleteCourse (new)

|  |  |
| --- | --- |
| **Operation** | deleteCourse (studentID: string, name: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The student instance was created (logined). |
| **Post-conditions** | A Course *ci*, which has *ci.StudentID* equal to  studentID and *ci.Name* equal to name, is destroyed. |

# updateCourse (new)

|  |  |
| --- | --- |
| **Operation** | updateCourse (studentID: string, name: string,  newName:string, credit: number, require:bool) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The student instance was created (logined). |
| **Post-conditions** | A Course *ci*, which has *ci.StudentID* equal to studentID and *ci.Name* equal to name, is updated:   * *ci.StudentID* became studentID. * *ci.Name* became newName. * *ci.Credit* became credit. * *ci.Require* became require. |

# showCourse (new)

|  |  |
| --- | --- |
| **Operation** | showCourse (studentID: string) |

|  |  |
| --- | --- |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The Course instances were created and had the  same studentID. |
| **Post-conditions** | None |

# enterPaper

|  |  |
| --- | --- |
| **Operation** | enterPaper (studentID: string, title: string,  publisher: string, year: string) |
| **Cross References** | Use Case: Manage Information |
| **Preconditions** | * The student instance was created (logined). * A Extend Information instance was created. |
| **Post-conditions** | * A Paper instance *p* was created or updated with studentID. * *c.Title* became title. * *c.Publisher* became publisher. * *c.Year* became year. |

# deletePaper (new)

|  |  |
| --- | --- |
| **Operation** | deletePaper (studentID: string, title: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The student instance was created (logined). |
| **Post-conditions** | A Paper *pi*, which has *pi.StudentID* equal to  studentID and *ci.Title* equal to title, is destroyed. |

# updatePaper (new)

|  |  |
| --- | --- |
| **Operation** | updatePaper (studentID: string, title: string, |

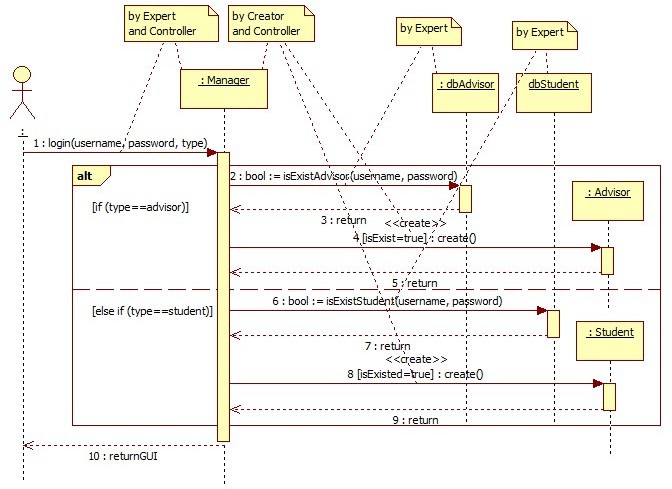
|  |  |
| --- | --- |
|  | newTitle:string, publisher: string, year: string,  point:number) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The student instance was created (logined). |
| **Post-conditions** | A Paper *pi*, which has *pi.StudentID* equal to studentID and *ci.Title* equal to name, is updated:   * *ci.StudentID* became studentID. * *ci.Title* became newTitle. * *Ci.Publisher* became publisher. * *ci.Year* became year. * *ci.Point* became point. |

# showPaper (new)

|  |  |
| --- | --- |
| **Operation** | showPaper (studentID: string) |
| **Cross References** | Use Case: Manage Student |
| **Preconditions** | The Paper instances were created and had the same  studentID. |
| **Post-conditions** | None |

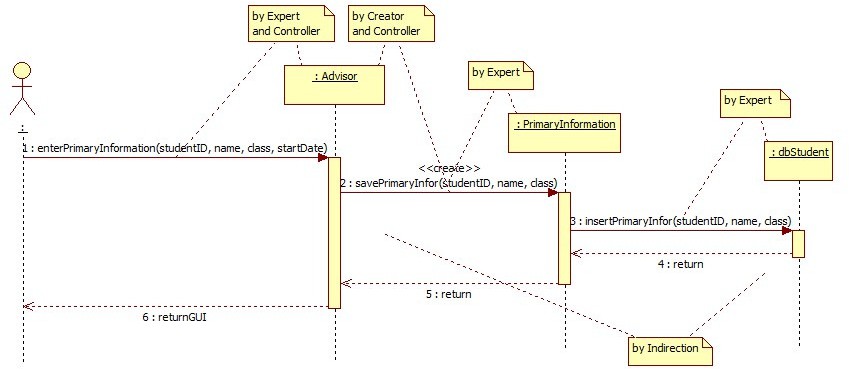
### Trình tự hoạt động

# login



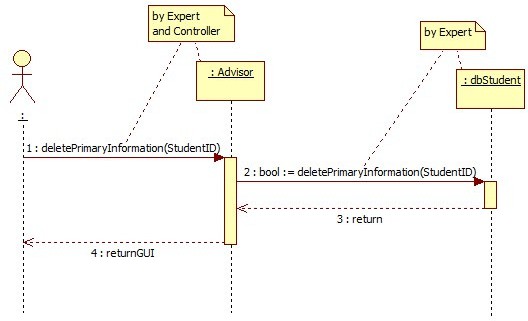
***Figure 4.4 Sequence Diagram of operator: login***

# enterPrimaryInformation



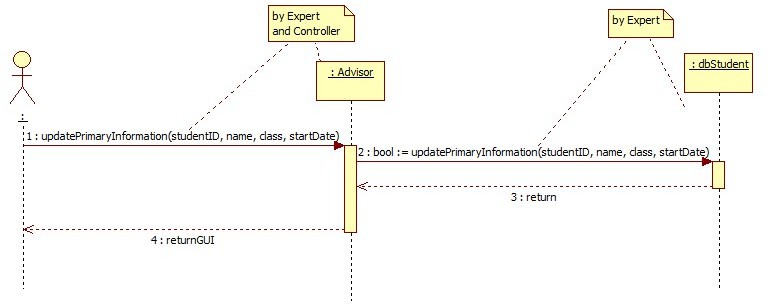
***Figure 4.5 Sequence Diagram of operator: enterPrimaryInformation***

# deletePrimaryInformation (new)



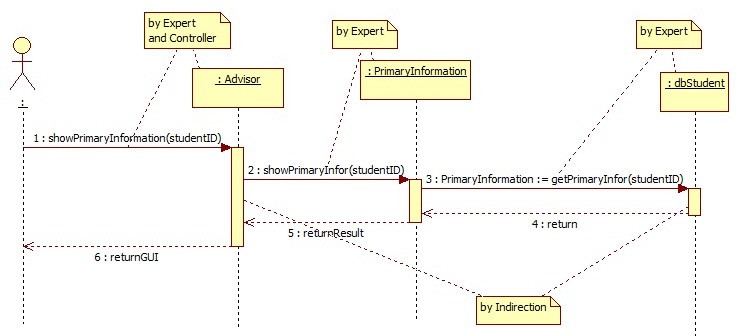
***Figure 4.6 Sequence Diagram of operator: deletePrimaryInformation***

# updatePrimaryInformation (new)



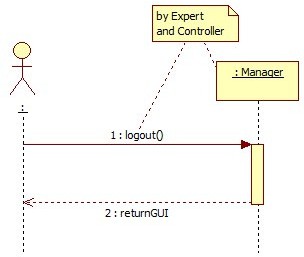
***Figure 4.7 Sequence Diagram of operator: updatePrimaryInformation***

# showPrimaryInformation



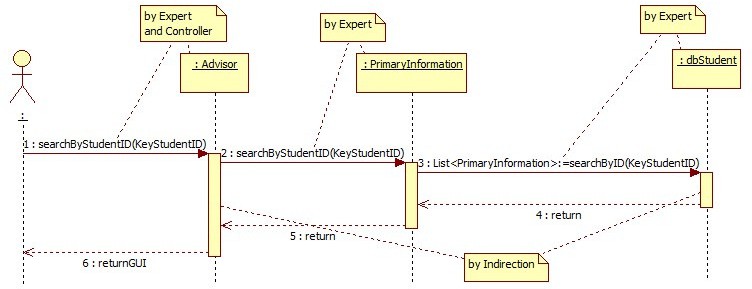
***Figure 4.8 Sequence Diagram of operator: showPrimaryInformation***

# logout



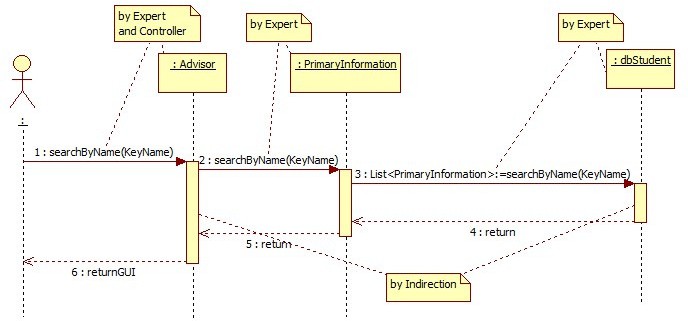
***Figure 4.9 Sequence Diagram of operator: logout***

### Tìm kiếm theo khoảng cách

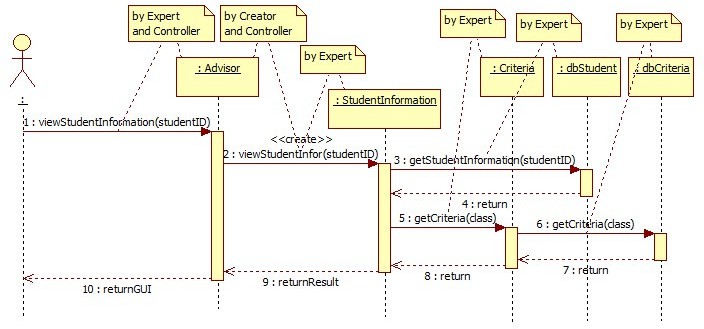


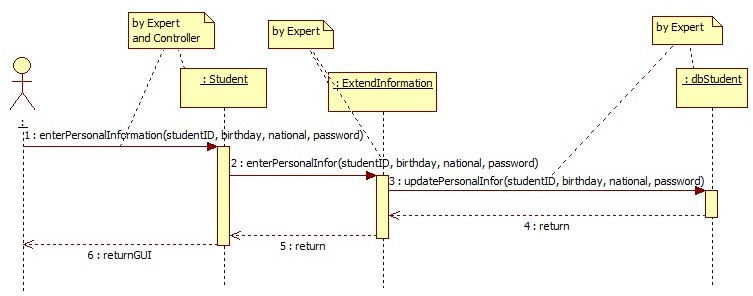
***Figure 4.10 Sequence Diagram of operator: searchByStudentID***

### Tìm theo giá

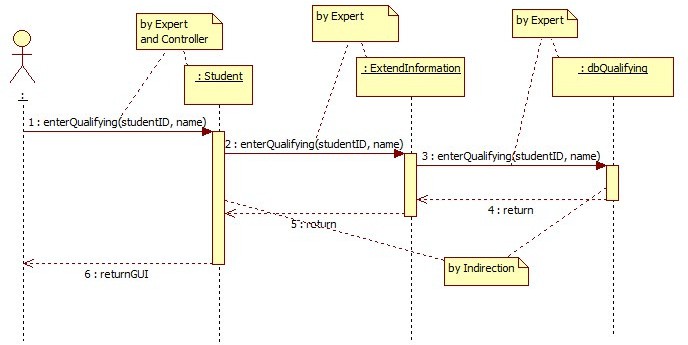


***Figure 4.11 Sequence Diagram of operator: searchByName***



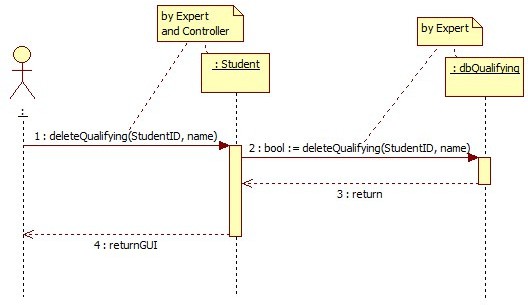
***Figure 4.12 Sequence Diagram of operator: viewStudentInformation***

***Figure 4.13 Sequence Diagram of operator: enterPersonalInformation***



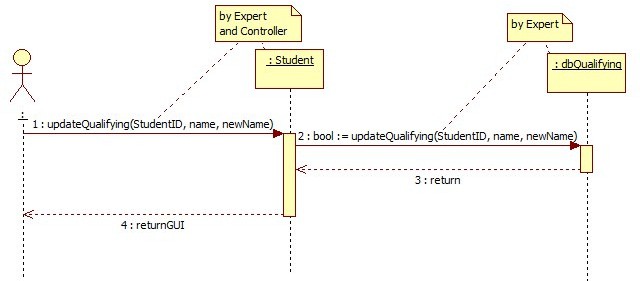
***Figure 4.14 Sequence Diagram of operator: enterQualifying***

### Xóa



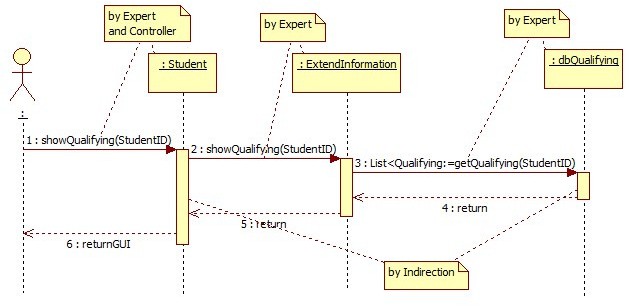
***Figure 4.15 Sequence Diagram of operator: deleteQualifying***

### Cập nhật



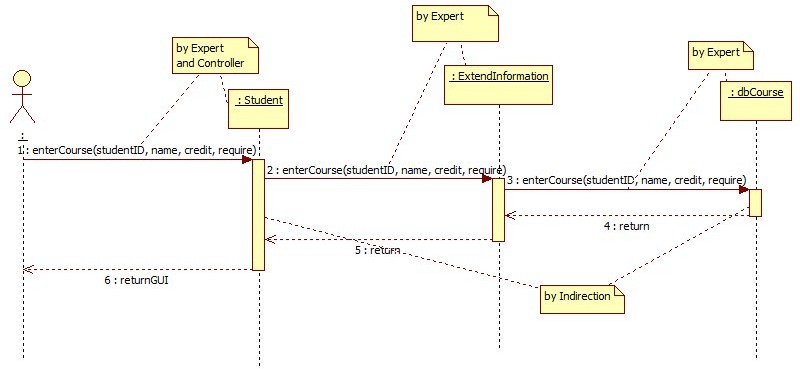
***Figure 4.16 Sequence Diagram of operator: updateQualifying***

# showQualifying (new)

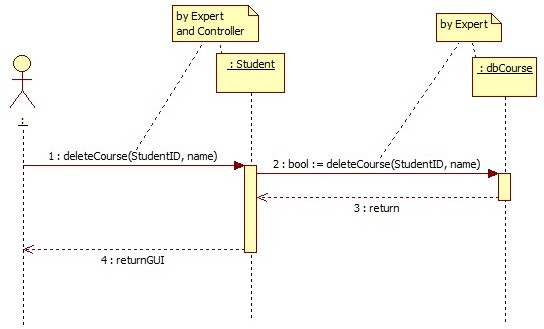


***Figure 4.17 Sequence Diagram of operator: showQualifying***

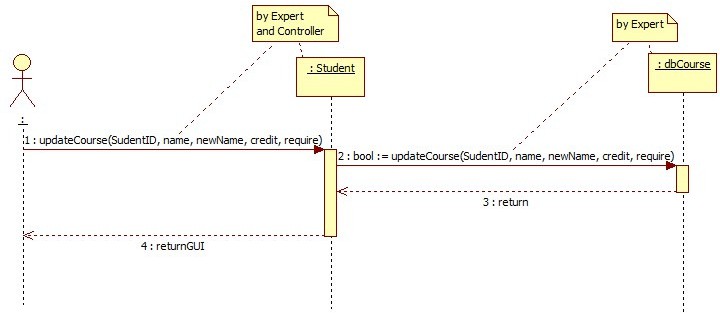
# enterCourse



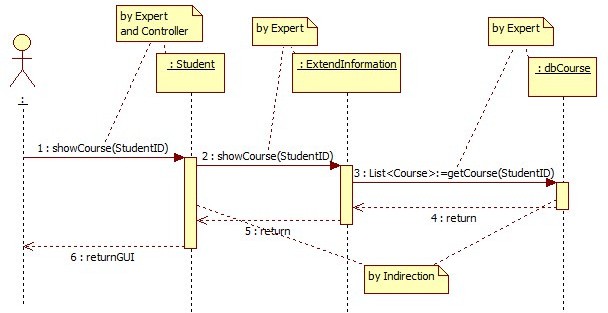
***Figure 4.18 Sequence Diagram of operator: enterCourse***

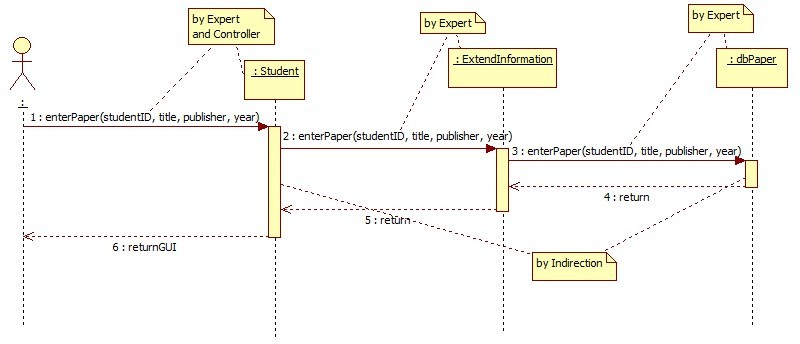


***Figure 4.19 Sequence Diagram of operator: deleteCourse***

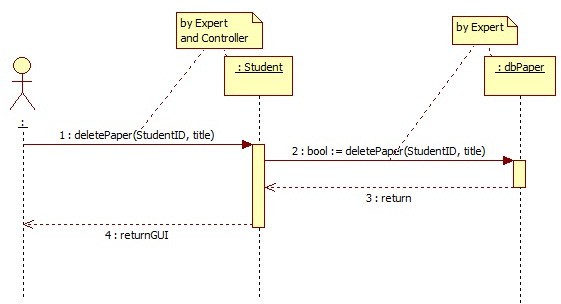


***Figure 4.20 Sequence Diagram of operator: updateCourse***

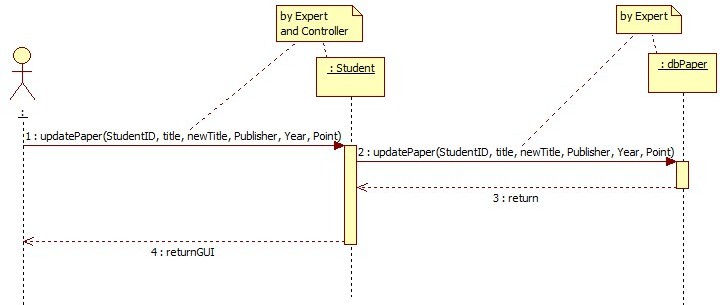


***Figure 4.21 Sequence Diagram of operator: showCourse***

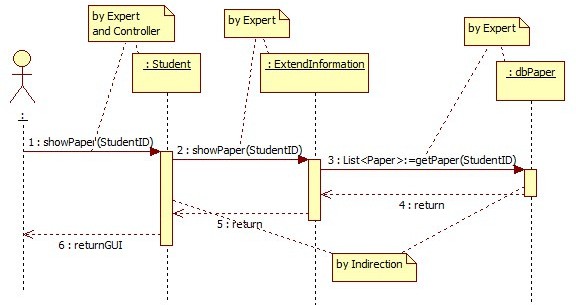
***Figure 4.22 Sequence Diagram of operator: enterPaper***



***Figure 4.23 Sequence Diagram of operator: deletePaper***

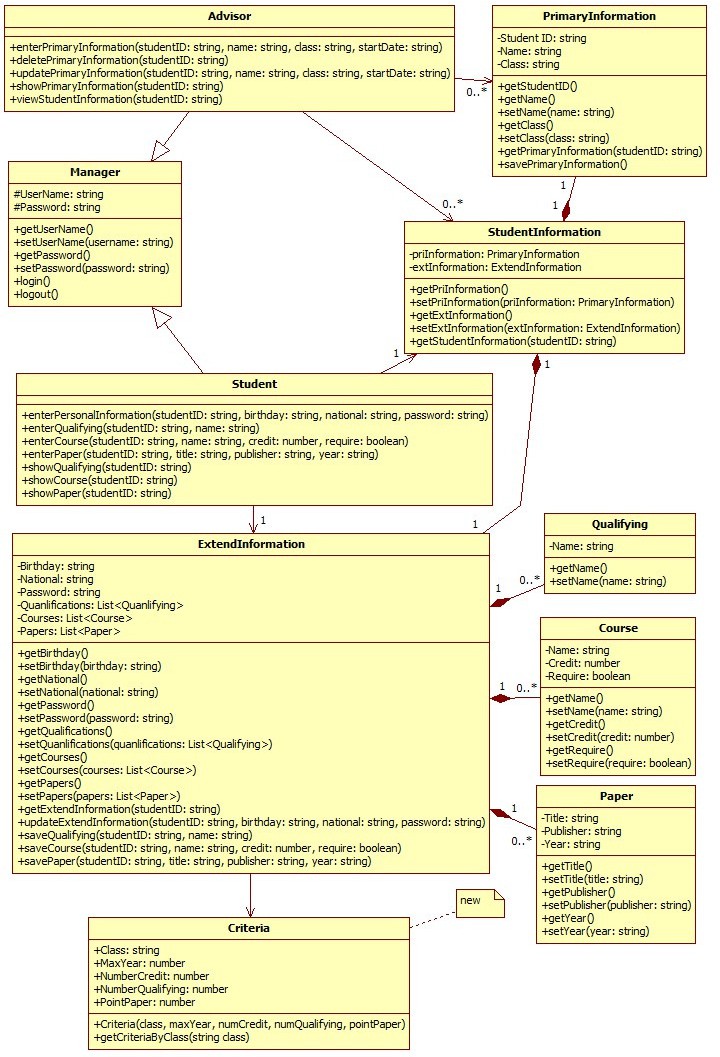


***Figure 4.24 Sequence Diagram of operator: updatePaper***



***Figure 4.25 Sequence Diagram of operator: showPaper***

# Design Class Model



***Figure 4.11 Design Class Model (new)***

# Implementation Class Model

# Draw an implementation class diagram

***Figure 5.1 Implementation Class Diagram (new)***

* 1. **Show the difference between implementation class model and design class model (new)**

**Table 5.2.1: Comparison with design and implementation class**

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Method | Design | Imp. |
| **Advisor** | enterPrimaryInformation() | **Yes** | **Yes** |
| deletePrimaryInformation() | **Yes** | **Yes** |
| updatePrimaryInformation() | **Yes** | **Yes** |
| showPrimaryInformation() | **Yes** | **Yes** |
| viewStudentInformation() | **Yes** | **Yes** |
| **Course** | Course() | **No** | **Yes** |
| deleteCourse() | **No** | **Yes** |
| insertCourse() | **No** | **Yes** |
| updateCourse() | **No** | **Yes** |
| **Criteria** | **Criteria()** | **Yes** | **Yes** |
| **getCriteriaByClass()** | **Yes** | **Yes** |
| **ExtendInformation** | deleteCourse() | **No** | **Yes** |
| deletePaper() | **No** | **Yes** |
| deleteQualifying() | **No** | **Yes** |
| ExtendInformation() | **No** | **Yes** |
| getExtendInformation() | **Yes** | **Yes** |
| insertCourse() | **Yes** | **Yes** |
| insertPaper() | **Yes** | **Yes** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | insertQualifying() | **Yes** | **Yes** |
| updateCourse() | **No** | **Yes** |
| updatePaper() | **No** | **Yes** |
| updatePersonalInformation() | **Yes** | **Yes** |
| updateQualifying() | **No** | **Yes** |
| **Manager** | Login() | **Yes** | **Yes** |
| Logout() | **Yes** | **Yes** |
| Manager() | **No** | **Yes** |
| **Paper** | deletePaper() | **No** | **Yes** |
| insertPaper() | **No** | **Yes** |
| Paper() | **No** | **Yes** |
| updatePaper() | **No** | **Yes** |
| **PrimaryInformation** | getPrimaryInformation() | **Yes** | **Yes** |
| PrimaryInformation() | **No** | **Yes** |
| savePrimaryInformation() | **Yes** | **Yes** |
| **Qualifying** | deleteQualifying() | **No** | **Yes** |
| insertQualifying() | **No** | **Yes** |
| Qualifying() | **No** | **Yes** |
| updateQualifying() | **No** | **Yes** |
| **Student** | deleteCourse() | **No** | **Yes** |
| deletePaper() | **No** | **Yes** |
| deleteQualifying() | **No** | **Yes** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | enterCourse() | **Yes** | **Yes** |
| enterPaper() | **Yes** | **Yes** |
| enterPersonalInformation() (new) | Yes | Yes |
| enterQualifying() | **Yes** | **Yes** |
| showExtendInformation() | **Yes** | **Yes** |
| Student() | **No** | **Yes** |
| updateCourse() | **No** | **Yes** |
| updatePaper() | **No** | **Yes** |
| updateQualifying() | **No** | **Yes** |
| **StudentInformation** | getStudentInformation() | **Yes** | **Yes** |
| StudentInformation() | **No** | **Yes** |
| **Program** | Main() | **No** | **Yes** |
| **frmAbout** | frmAbout() | **No** | **Yes** |
| **frmLogin** | frmLogin() | **No** | **Yes** |
| btLogin\_Click() | **No** | **Yes** |
| btExit\_Click() | **No** | **Yes** |
| **frmConnectDatabase** | btConnect\_Click() | **No** | **Yes** |
| btExit\_Click() | **No** | **Yes** |
| cbAuthenticatedSQL\_CheckedChanged() | **No** | **Yes** |
| frmConnectDatabase() | **No** | **Yes** |
| **Persistence** | connectDatabaseByAuthenticatedWindows() | **No** | **Yes** |
| connectDatabaseByAuthenticatedSQLServer() | **No** | **Yes** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | closeConnectDatabase() | **No** | **Yes** |
| querySelect() | **No** | **Yes** |
| queryUpdate() | **No** | **Yes** |
| **dbAdvisor** | getAdvisorTable() | **No** | **Yes** |
| findAdvisorTable() | **No** | **Yes** |
| isExistAdvisorTable() | **No** | **Yes** |
| **dbCourse** | getCourseTable() | **No** | **Yes** |
| insertCourseTable() | **No** | **Yes** |
| deleteCourseTable() | **No** | **Yes** |
| updateCourseTable() | **No** | **Yes** |
| **dbCriteria** | **getCriteriaByClass()** | **No** | **Yes** |
| **dbPaper** | getPaperTable() | **No** | **Yes** |
| insertPaperTable() | **No** | **Yes** |
| deletePaperTable() | **No** | **Yes** |
| updatePaperTable() | **No** | **Yes** |
| **dbQualifying** | getQualifyingTable() | **No** | **Yes** |
| insertQualifyingTable() | **No** | **Yes** |
| deleteQualifyingTable() | **No** | **Yes** |
| updateQualifyingTable() | **No** | **Yes** |
| **dbStudent** | getStudentTable() | **No** | **Yes** |
| getPrimaryStudentTable() | **No** | **Yes** |
| getPrimaryInformation() | **No** | **Yes** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | getPrimaryInformation() | **No** | **Yes** |
| getExtendInformation() | **No** | **Yes** |
| findStudentTableByStudentID() | **No** | **Yes** |
| findStudentTableByName() | **No** | **Yes** |
| findStudentTable() | **No** | **Yes** |
| isExistStudentTable() | **No** | **Yes** |
| insertPrimaryStudentTable() | **No** | **Yes** |
| deletePrimaryStudentTable() | **No** | **Yes** |
| updatePrimaryStudentTable() | **No** | **Yes** |
| updatePersonalStudentTable() | **No** | **Yes** |
| **MainForm** | aboutToolStripMenuItem\_Click() | **No** | **Yes** |
| AddCoursesToListview() | **No** | **Yes** |
| AddPapersToListview() | **No** | **Yes** |
| AddPrimaryStudentToListview() | **No** | **Yes** |
| AddQualificationsToListview() | **No** | **Yes** |
| AddStudentIDToCombobox() | **No** | **Yes** |
| AddStudentInformationToForm() | **No** | **Yes** |
| btAddStudent\_Click() | **No** | **Yes** |
| btCancelStudent\_Click() | **No** | **Yes** |
| btCourseAdd\_Click() | **No** | **Yes** |
| btCourseCancel\_Click() | **No** | **Yes** |
| btCourseDelete\_Click() | **No** | **Yes** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | btCourseUpdate\_Click() | **No** | **Yes** |
| btDeleteStudent\_Click() | **No** | **Yes** |
| btPaperAdd\_Click() | **No** | **Yes** |
| btPaperCancel\_Click() | **No** | **Yes** |
| btPaperDelete\_Click() | **No** | **Yes** |
| btPaperUpdate\_Click() | **No** | **Yes** |
| btPersonalStudentCancel\_Click() | **No** | **Yes** |
| btPersonalStudentUpdate\_Click() | **No** | **Yes** |
| btQualifyingAdd\_Click() | **No** | **Yes** |
| btQualifyingCancel\_Click() | **No** | **Yes** |
| btQualifyingDelete\_Click() | **No** | **Yes** |
| btQualifyingUpdate\_Click() | **No** | **Yes** |
| btUpdate\_Click() | **No** | **Yes** |
| cbViewStudentID\_SelectedIndexChanged() | **No** | **Yes** |
| cbViewStudentID\_TextChanged() | **No** | **Yes** |
| ClearCourseForm() | **No** | **Yes** |
| ClearPaperForm() | **No** | **Yes** |
| ClearPersonalStudentForm() | **No** | **Yes** |
| ClearPrimaryStudentForm() | **No** | **Yes** |
| ClearQualifyingForm() | **No** | **Yes** |
| ClearViewStudentForm() | **No** | **Yes** |
| closeConnectionToolStripMenuItem\_Click() | **No** | **Yes** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | ConnectSQLServerToolStripMenuItem\_Click() | **No** | **Yes** |
| exitToolStripMenuItem\_Click() | **No** | **Yes** |
| loginToolStripMenuItem\_Click() | **No** | **Yes** |
| logoutToolStripMenuItem\_Click() | **No** | **Yes** |
| lvCourse\_SelectedIndexChanged() | **No** | **Yes** |
| lvPaper\_SelectedIndexChanged() | **No** | **Yes** |
| lvPrimaryStudent\_SelectedIndexChanged() | **No** | **Yes** |
| lvQualifying\_SelectedIndexChanged() | **No** | **Yes** |
| MainForm() | **No** | **Yes** |
| qualifyingToolStripMenuItem\_Click() | **No** | **Yes** |
| setConnectResult() | **No** | **Yes** |
| setEnableLogin() | **No** | **Yes** |
| setEnableLogout() | **No** | **Yes** |
| setInitForAdvisor() | **No** | **Yes** |
| setInitForStudent() | **No** | **Yes** |
| setLoginResult() | **No** | **Yes** |

**Table 5.2.2: Summary of implementation class/method changed**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Number of added** | **Number of removed** | **Number of modified** |
| **Class** | 12 | 0 | 3 |
| **Method** | 78 | 0 | 2 |

* 1. **Calculate Line of Code (new)**

**Table 5.3.1: Line of Code of classes**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Number of methods** | **Line of Code in Class**  **(without comment)** |
| **1** | Program | 1 | 19 |
| **2** | MainForm | 31 | 603 |
| **3** | frmAbout | 1 | 19 |
| **4** | frmLogin | 3 | 46 |
| **5** | frmConnectDatabase | 3 | 62 |
| **6** | Advisor | 5 | 57 |
| **7** | Manager | 6 | 92 |
| **8** | Student | 5 | 79 |
| **9** | PrimaryInformation | 8 | 86 |
| **10** | StudentInformation | 5 | 77 |
| **11** | ExtendInformation | 23 | 183 |
| **12** | Qualifying | 5 | 51 |
| **13** | Course | 6 | 84 |
| **16** | dbStudent | 13 | 151 |
| **17** | Persistence | 5 | 88 |
| **18** | dbAdvisor | 3 | 35 |
| **19** | **dbCriteria** | **1** | **32** |
| **20** | dbCourse | 4 | 49 |

|  |  |  |
| --- | --- | --- |
| **SUM** | **150** | **2096** |

# Kết quả chương trình

## Snapshots of system execution

***Figure 6.1 Main form of SIS***

***Figure 6.2 Connect to SQL Server***

***Figure 6.3 Login to SIS as an Advisor or a Student***

***Figure 6.4 GUI of manage Student for the Advisor logged in***

***Figure 6.5 GUI of View Student Information for the Advisor logged in***

***Figure 6.6 GUI of Modify Personal Information for the Student logged in***

***Figure 6.6 GUI of Modify Qualifying for the Student logged in***

***Figure 6.6 GUI of Modify Course for the Student logged in***

## Source Code Listing

### Program.cs

### frmLogin.cs

### StudentInformation.cs

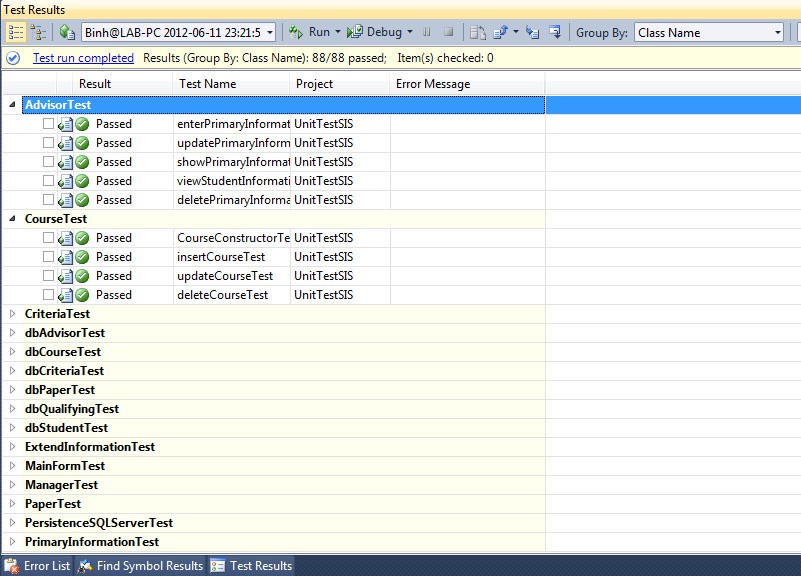
using System.Collections.Generic; using System.Linq;

using System.Text;

using SIS.Domain.Inventory;

# Unit Testing

## Snapshots of testing result



## Unit Testing Code Listing

### AdvisorTest.cs

using SIS.Domain.Manage;

using Microsoft.VisualStudio.TestTools.UnitTesting; using System;

///A test for enterPrimaryInformation

///</summary> [TestMethod()]

public void enterPrimaryInformationTest()

{

Advisor target = new Advisor(); string studentID = "100599000"; string name = "TestName"; string rank = "Master";

bool expected = true; bool actual;

actual = target.enterPrimaryInformation(studentID, name, rank); Assert.AreEqual(expected, actual);

}

/// <summary>

///A test for updatePrimaryInformation

///</summary> [TestMethod()]

public void updatePrimaryInformationTest()

{

Advisor target = new Advisor(); string studentID = "100599000"; string name = "newTestName"; string rank = "Master";

bool expected = true; bool actual;

actual = target.updatePrimaryInformation(studentID, name, rank); Assert.AreEqual(expected, actual);

}

/// <summary>

///A test for showPrimaryInformation

///</summary> [TestMethod()]

public void showPrimaryInformationTest()

{

Advisor target = new Advisor(); string studentID = "100599000";

PrimaryInformation expected = new PrimaryInformation("100599000", "newTestName", "Master");

PrimaryInformation actual;

actual = target.showPrimaryInformation(studentID); Assert.AreEqual(expected.StudentID, actual.StudentID.Trim()); Assert.AreEqual(expected.Name, actual.Name.Trim()); Assert.AreEqual(expected.Rank, actual.Rank.Trim());

}

/// <summary>

///A test for viewStudentInformation

///</summary> [TestMethod()]

public void viewStudentInformationTest()

{

Advisor target = new Advisor(); string studentID = "100599000";

StudentInformation actual;

actual = target.viewStudentInformation(studentID); Assert.AreEqual(studentID, actual.StudentID.Trim());

}

/// <summary>

///A test for deletePrimaryInformation

///</summary> [TestMethod()]

public void deletePrimaryInformationTest()

{

Advisor target = new Advisor(); string studentID = "100599000";

bool expected = true; bool actual;

actual = target.deletePrimaryInformation(studentID); Assert.AreEqual(expected, actual);

}

}

}

### ManagerTest.cs

/// <summary>

///A test for updateQualifying

///</summary> [TestMethod()]

public void updateQualifyingTest()

{

Student target = new Student(); string studentID = "100599008"; string name = "TestName"; string newName = "newTestName"; bool expected = true;

bool actual;

actual = target.updateQualifying(studentID, name, newName); Assert.AreEqual(expected, actual);

}

/// <summary>

///A test for deleteQualifying

///</summary> [TestMethod()]

public void deleteQualifyingTest()

{

Student target = new Student(); string studentID = "100599008"; string name = "newTestName"; bool expected = true;

bool actual;

actual = target.deleteQualifying(studentID, name); Assert.AreEqual(expected, actual);

}

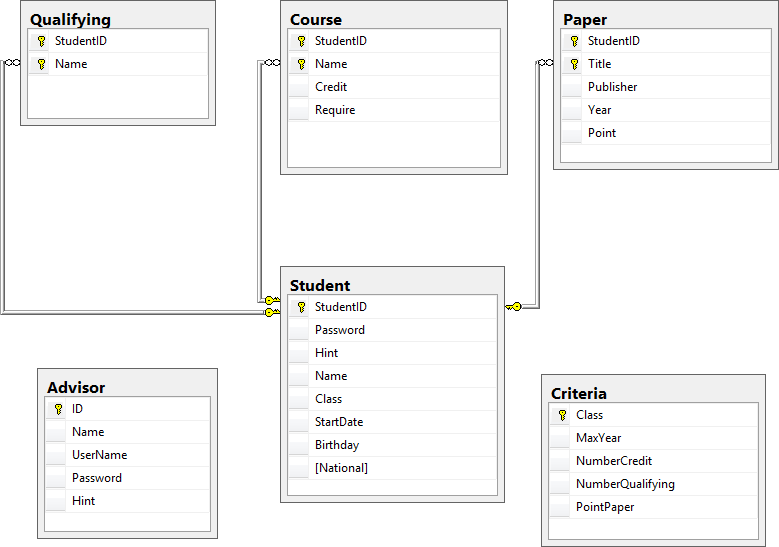
}

}

# 

**Appendix**

**SQL Server Database Tables**



**Test Data**

