

Trường Đại học Công nghiệp Thực phẩm Tp. Hồ Chí Minh

Bộ môn Công nghệ Phần mềm



Lập trình hướng đối tượng

ĐẠI HỌC CHÍNH QUY

Trường Đại học Công nghiệp Thực phẩm Tp. Hồ Chí Minh
Bộ môn Công nghệ Phần mềm



Phần 11&12: MÔ HÌNH THIẾT KẾ HĐT 3 LỚP

ĐẠI HỌC CHÍNH QUY

26/04/2023



MỤC TIÊU

Giúp sinh viên có được những kiến thức về Mô hình Hướng đối tượng 3 lớp và lợi thế của nó trong C#.

NỘI DUNG

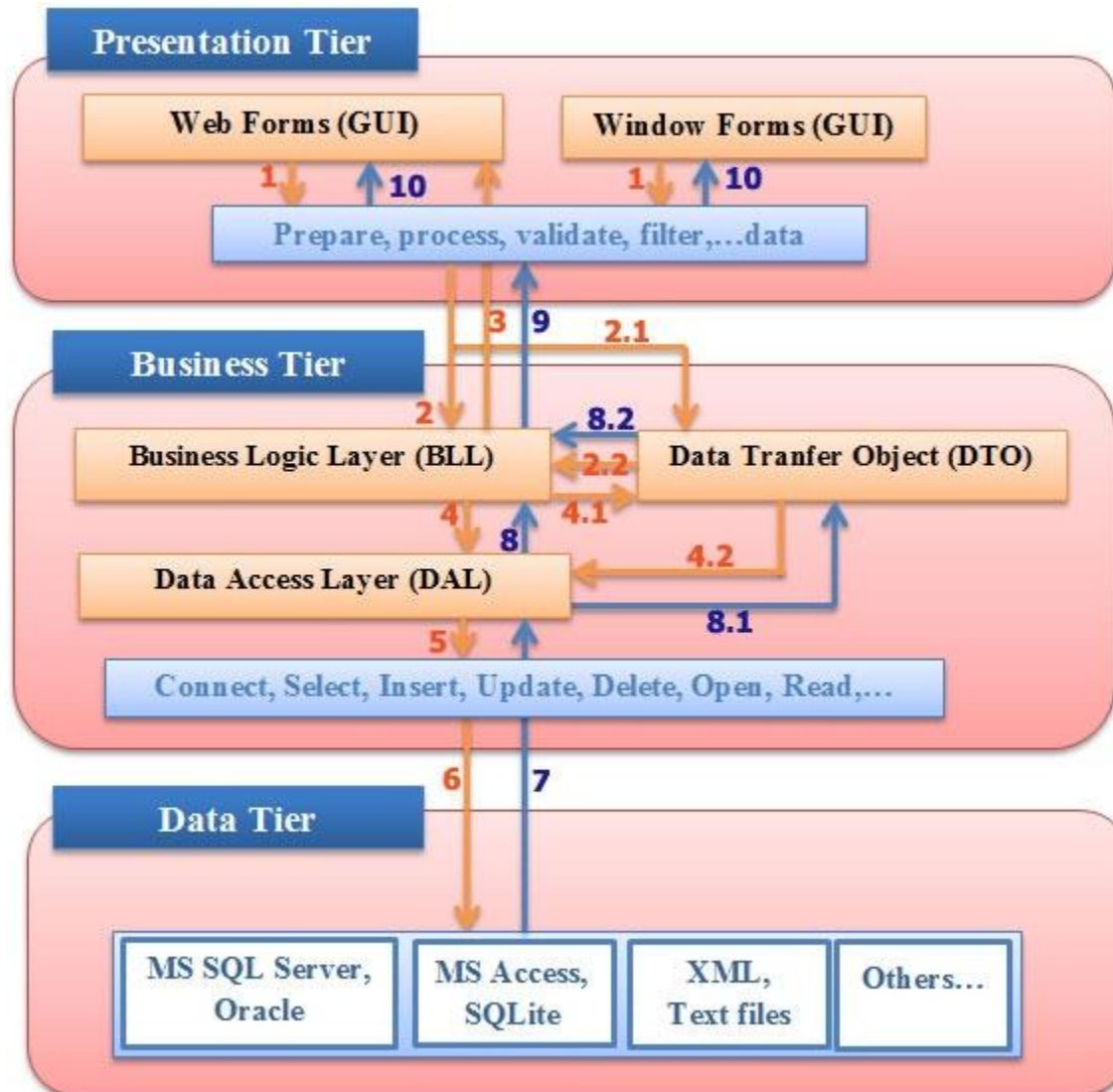
- ❖ Phân biệt giữa 3-Layers và 3-tiers.
- ❖ Mô hình Hướng đối tượng 3 lớp trong C#
- ❖ Lợi thế của mô hình 3 lớp
- ❖ Chi tiết về mô hình 3 lớp
- ❖ Ví dụ minh họa



Phân biệt biệt 3-tiers và 3-Layers

- ❖ Mô hình **3 tầng** (**3-tiers** architecture) là gì?
- ❖ Mô hình **3 lớp** (**3-Layers** architecture) là gì?
- ❖ Sự khác biệt giữa 3-Layers và 3-tiers?
- ➔ Chúng có ý nghĩa như thế nào và tại sao ta phải dùng mô hình 3-Layers để viết theo mô hình này trong ứng dụng quản lý .NET?

MÔ HÌNH 3-Tiers và 3-Layers



Mô hình 3 tầng (3-tiers)

- ❖ **3-Tiers có tính vật lý** (physical): là mô hình client-server (mỗi tier có thể đặt chung 1 nơi hoặc nhiều nơi, kết nối với nhau qua Web services, WCF, Remoting...). Như hình vẽ trên ta thấy 3 tầng rõ rệt:
- **Presentation tier:** bao gồm các thành phần xử lý giao diện Graphic User Interface (GUI).
 - **Business tier:** gồm các thành phần Business Logic Layer (BLL), Data Access Layer (DAL) và Data Transfer Object (DTO).
 - **Data tier:** lưu trữ dữ liệu, là các hệ quản trị CSDL như MS SQL Server, Oracle, SQLite, MS Access, XML files, Text files,...

Mô hình 3 tầng (3-tiers)

❖ Ưu điểm:

- Dễ dàng mở rộng, thay đổi quy mô của hệ thống: Khi cần tải lớn, người quản trị có thể dễ dàng thêm các máy chủ vào nhóm, hoặc lấy bớt ra trong trường hợp ngược lại.

❖ Nhược điểm:

- Việc truyền dữ liệu giữa các tầng sẽ chậm hơn vì phải truyền giữa các tiến trình khác nhau (IPC), dữ liệu cần phải được đóng gói → truyền đi → mở gói trước khi có thể dùng được.
- Việc phát triển ứng dụng phức tạp hơn.



Mô hình 3 lớp (3-Layers)

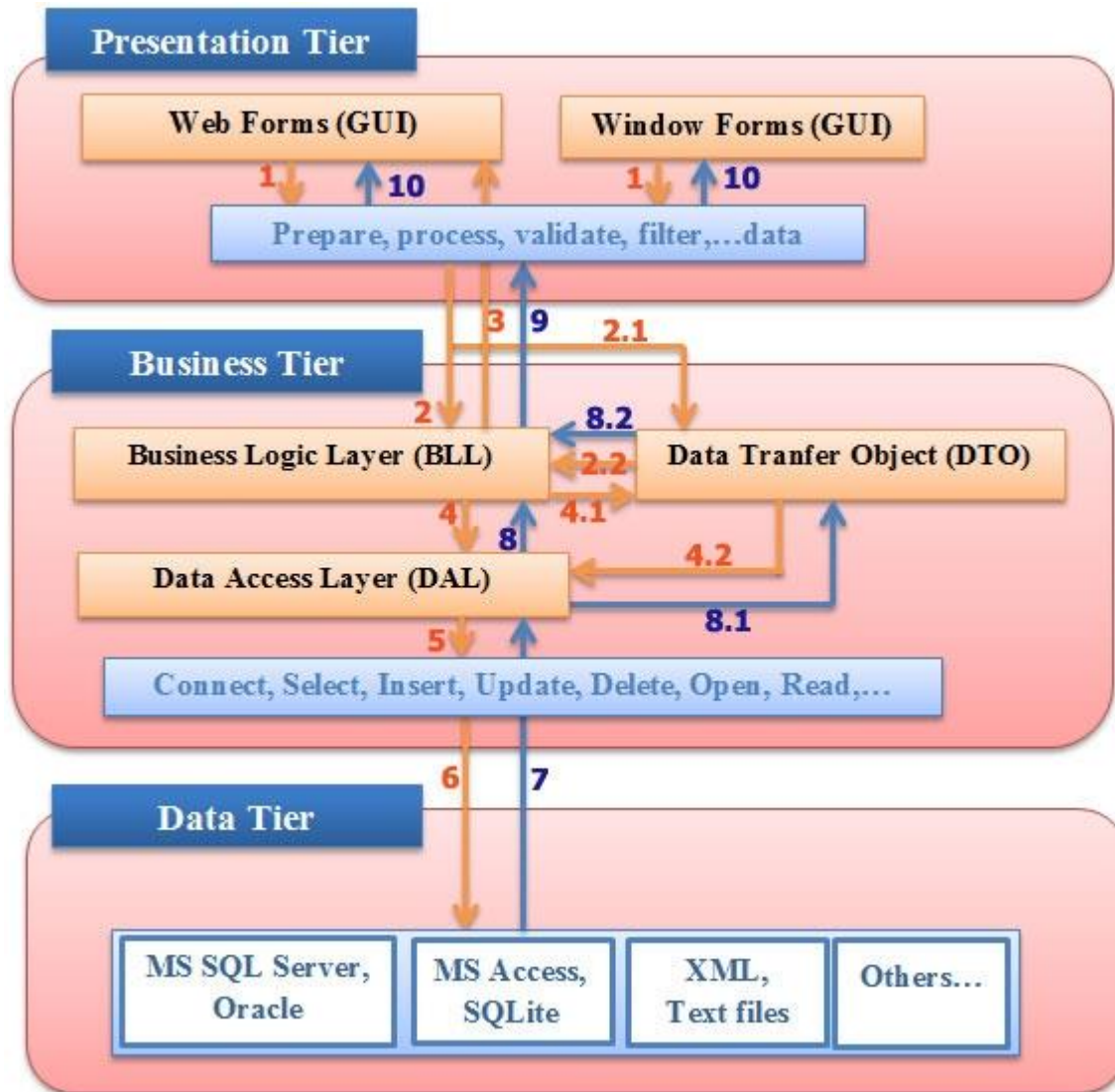
- ❖ Không như 3-Tiers có tính vật lý, 3-Layers **có tính logic** (mỗi layer có 1 công việc) và là 1 thành phần của 3-Tiers. Gồm 3 lớp chính:
 - **Graphic User Interface (GUI)**: Thành phần giao diện dạng *Console* hay *Forms* của chương trình tương tác với người sử dụng.
 - **Business Logic Layer (BLL)**: Xử lý các nghiệp vụ của chương trình như tính toán, xử lý hợp lệ và toàn vẹn về mặt dữ liệu.
 - **Data Access Layer (DAL)**: Tầng giao tiếp với các hệ quản trị CSDL, XML files, Text files,...



Mô hình 3 lớp (3-Layers)

- ❖ Trong 1 số trường hợp vì lượng thông tin gửi nhiều ta có thể dùng **Data Transfer Object (DTO)** để chuyển đổi tượng hoặc danh sách đối tượng giữa các tầng với nhau cho tiện dụng.

Mô hình 3 lớp (3-Layers)



Mô hình 3 lớp (3-Layers)

Theo như hình vẽ: mô tả sự hoạt động của mô hình 3-Layers gồm 2 tiến trình như sau:

❖ Các đường **mũi tên màu cam** biểu diễn cho 1 tiến trình giao tiếp từ **GUI** đến **DAL**.

- (1) Người sử dụng tác động lên **GUI** yêu cầu hiển thị thông tin lên màn hình. Tại đây **GUI** sẽ kiểm tra yêu cầu của người dùng nhập có hợp lệ hay không, nếu không hợp lệ sẽ thông báo cho người dùng.



Mô hình 3 lớp (3-Layers)

- Ngược lại nếu yêu cầu hợp lệ thì sẽ được gửi trực tiếp đến **BLL** (2) hoặc thông qua lớp object **DTO** hỗ trợ luân chuyển (2.1 & 2.2), tại đây **BLL** sẽ xử lý nghiệp vụ về yêu cầu của người dùng, nếu:
 - (i) Yêu cầu không hợp lệ hoặc tự xử lý yêu cầu không cần phải truy vấn CSDL thì **BLL** sẽ gửi thông tin về **GUI** (3) và **GUI** sẽ hiển thị kết quả cho người dùng.
 - (ii) Yêu cầu hợp lệ và xử lý yêu cầu cần phải truy vấn CSDL thì **BLL** sẽ gửi yêu cầu đến trực tiếp đến **DAL** (4) hoặc thông qua lớp object **DTO** (4.1 & 4.2), nhờ **DAL** giao tiếp với hệ quản trị CSDL (5) để lấy (đọc) hoặc thêm, xóa, sửa dữ liệu.



Mô hình 3 lớp (3-Layers)

- **DAL** sẽ giao tiếp hệ quản trị CSDL (**6**) với các truy xuất dữ liệu (*sử dụng công nghệ ADO, LINQ to SQL, NHibernate, Entity Framework*), XML files, Text files, ...

Mô hình 3 lớp (3-Layers)

❖ Tiếp tục thực hiện *tiến trình thứ 2* được biểu diễn bằng các **mũi tên màu xanh**:

- Sau khi **DAL** thực hiện giao tiếp với hệ quản trị CSDL hoặc XML thì sẽ trả kết quả truy vấn về cho **DAL** (**7**),
- sau đó **DAL** sẽ gửi thông tin về dữ liệu vừa lấy trực tiếp sang cho **BLL** (**8**) hoặc *thông qua lớp object DTO* (**8.1** & **8.2**) xử lý tiếp nghiệp vụ với yêu cầu đã gửi từ trước, sau khi xử lý xong nghiệp vụ, thì **BLL** sẽ gửi thông tin đến **GUI** (**9**), và **GUI** sẽ hiển thị thông báo và kết quả yêu cầu lên màn hình (**10**).

Ví dụ - Mô hình 3 lớp

- ❖ Mô tả hoạt động của mô hình 3-Layers với ứng dụng liệt kê danh sách điểm sinh viên?
- ➔ Từ màn hình giao diện quản lý sinh viên: có 1 yêu cầu nhập mã lớp học để hiển thị danh sách sinh viên.
 - Nếu người dùng nhập đúng mã lớp (*hợp lệ*) thì chương trình sẽ thực hiện lệnh liệt kê danh sách sinh viên theo yêu cầu.
 - Nếu người dùng nhập đúng mã lớp không hợp lệ thì chương trình sẽ báo lỗi.

Ví dụ - Mô hình 3 lớp

- (1) Người dùng nhập mã lớp trên **GUI** và ấn phím Enter.
- (2) **GUI** kiểm tra yêu cầu hợp lệ và gửi mã lớp sang **BLL** xử lý yêu cầu hiển thị danh sách điểm sinh viên.
- (4) Tại **BLL** vì yêu cầu từ **GUI** khá đơn giản nên **BLL** sẽ không xử lý gì mà sẽ gửi mã lớp sang **DAL** lấy danh sách điểm.
- (5) Tại **DAL** sau khi đã nhận được yêu cầu lấy danh sách điểm với mã lớp nhận được, **DAL** sẽ tương tác với hệ quản trị CSDL (6) qua các lệnh mở tập tin, kết nối, truy vấn,... để lấy được danh sách điểm (7) với mã lớp yêu cầu, **DAL** tiếp tục gửi danh sách này sang **BLL** để xử lý (7).
- (8) Tại **BLL** sau khi nhận được danh sách điểm từ **DAL** gửi sang, **BLL** thực hiện nghiệp vụ của mình bằng cách tính điểm trung bình, kết luận đậu/rớt của từng sinh viên (tất cả xử lý về mặt nghiệp vụ), sau đó gửi danh sách điểm đã xử lý sang **GUI** để hiển thị (9).
- (9) Một lần nữa **GUI** có thể kiểm tra tính hợp lệ của dữ liệu và hiển thị thông tin và thông báo lên màn hình cho người dùng (10).



Cách tổ chức mô hình trên.NET

Với mỗi tầng (**DAL**, **BLL**) ta tạo 1 Project mới kiểu Class Library, sau khi **build** ra các dll như: BLL.dll, DTO.dll, DAL.dll. Khi đó:

- Tầng **GUI** là project chính chương trình, vì đặc điểm GUI chỉ thấy **BLL** nên ta sẽ ***add references*** BLL.dll, DTO.dll từ tab project vào GUI.
- Tầng **BLL** chỉ thấy được **DAL**, ta tiếp tục ***add references*** DAL.dll, DTO.dll vào BLL.
- Tầng **DAL** giao tiếp được với CSDL, ta ***add references*** DTO.dll vào DAL. Sử dụng các namespace data provider để tương tác với hệ quản trị CSDL, XML files, Text files,...



Một số kinh nghiệm khi sử dụng

- ❖ Các thao tác trên control như: kiểm tra nhập hợp lệ, ẩn hiện các control, và các xử lý thông tin trên control thì ta đặt các hàm xử lý ngay trên **GUI**.
- ❖ Các thao tác trên các dữ liệu cơ bản như: List, Array List, Object, DataTable, string, int, long, float,... ta xử lý ngay chính tầng nghiệp vụ **BLL**, vì bản chất khi thay đổi hệ quản trị hay các platform thì BLL không thay đổi.

Một số kinh nghiệm khi sử dụng

- ❖ Các thao tác với CSDL như: truy vấn, kết nối, đóng kết nối,... ta xử lý trong **DAL**.
- ❖ Khi có nhu cầu thay đổi hệ quản trị CSDL, ta chỉ cần thay đổi DAL phù hợp với hệ quản trị mới, giữ nguyên BLL, GUI và build lại project.
- ❖ Khi có nhu cầu chuyển đổi qua lại giữa ứng dụng **Web Forms** hoặc **Window Forms** ta chỉ cần thay GUI, giữ nguyên DAL, BLL và build lại project.



Lợi thế của mô hình 3 lớp

- ❖ **Việc phân chia thành từng lớp:** giúp cho code được tường minh hơn. Nhờ vào việc chia ra từng lớp đảm nhận các chức năng khác nhau và riêng biệt như giao diện, xử lý, truy vấn thay vì để tất cả lại một chỗ. Nhằm giảm sự kết dính.
- ❖ **Dễ bảo trì khi được phân chia:** một thành phần của hệ thống sẽ dễ thay đổi. Việc thay đổi này có thể được cô lập trong 1 lớp, hoặc ảnh hưởng đến lớp gần nhất mà không ảnh hưởng đến cả chương trình.



Lợi thế của mô hình 3 lớp

- ❖ **Dễ phát triển, tái sử dụng:** khi chúng ta muốn thêm một chức năng nào đó thì việc lập trình theo một mô hình sẽ dễ dàng hơn vì chúng ta đã có chuẩn để tuân theo. Và việc sử dụng lại khi có sự thay đổi giữa hai môi trường (Winform sang Webfrom và ngược lại) thì chỉ việc thay đổi lại lớp GUI.
- ❖ **Dễ bàn giao:** Nếu mọi người đều theo một quy chuẩn đã được định sẵn, thì công việc bàn giao, tương tác với nhau sẽ dễ dàng hơn và tiết kiệm được nhiều thời gian.



Lợi thế của mô hình 3 lớp

❖ **Dễ phân phối khối lượng công việc:** Mỗi một nhóm, một bộ phận sẽ nhận một nhiệm vụ trong mô hình 3 lớp. Việc phân chia rõ ràng như thế sẽ giúp các lập trình viên kiểm soát được khối lượng công việc của mình.



Ví dụ về mô hình 3 lớp

Ứng dụng **mô hình 3 lớp** để hiển thị danh sách sinh viên đã cho trong file XML.

...

<StudentList>

<Student>

<StudentID>12345</StudentID>

<LastName>Nguyễn Thị Tú</LastName>

<FirstName>Trinh</FirstName>

<Phone>0933726351</Phone>

<Email>ntttrinh@gmail.com</Email>

<AverageScore>6.27</AverageScore>

</Student>

...

</StudentList>

Ví dụ về mô hình 3 lớp

Khi xây dựng mô hình 3 lớp:

- ❖ Cần một **Solution** riêng (VD: **Demo_3Layers**).
- ❖ Cần 4 **project** khác nhau để làm nên 4 **lớp**, với tên các **Project** được đặt như sau:
 - Lớp GUI: **GUI_*** (VD: **GUI_QLSinhVien**)
 - Lớp Business: **BLL_*** (VD: **BLL_QLSinhVien**)
 - Lớp Data Access: **DAL_*** (VD: **DAL_QLSinhVien**)
 - Lớp DTO: **DTO_*** (VD: **DTO_QLSinhVien**)

Ví dụ về mô hình 3 lớp

Bên trong **4 lớp** như trên, khi đặt tên cho các file ***.cs** cần có các hậu tố như sau:

Ví dụ ta có một đối tượng tên là **Student** thì:

- ❖ Lớp GUI: ***GUI** (VD: **StudentGUI.cs**)
- ❖ Lớp Business: ***BLL** (VD: **StudentBLL.cs**)
- ❖ Lớp Data Access: ***DAL** (VD: **StudentDAL.cs**)
- ❖ Lớp DTO: ***DTO** (VD: **StudentDTO.cs**)

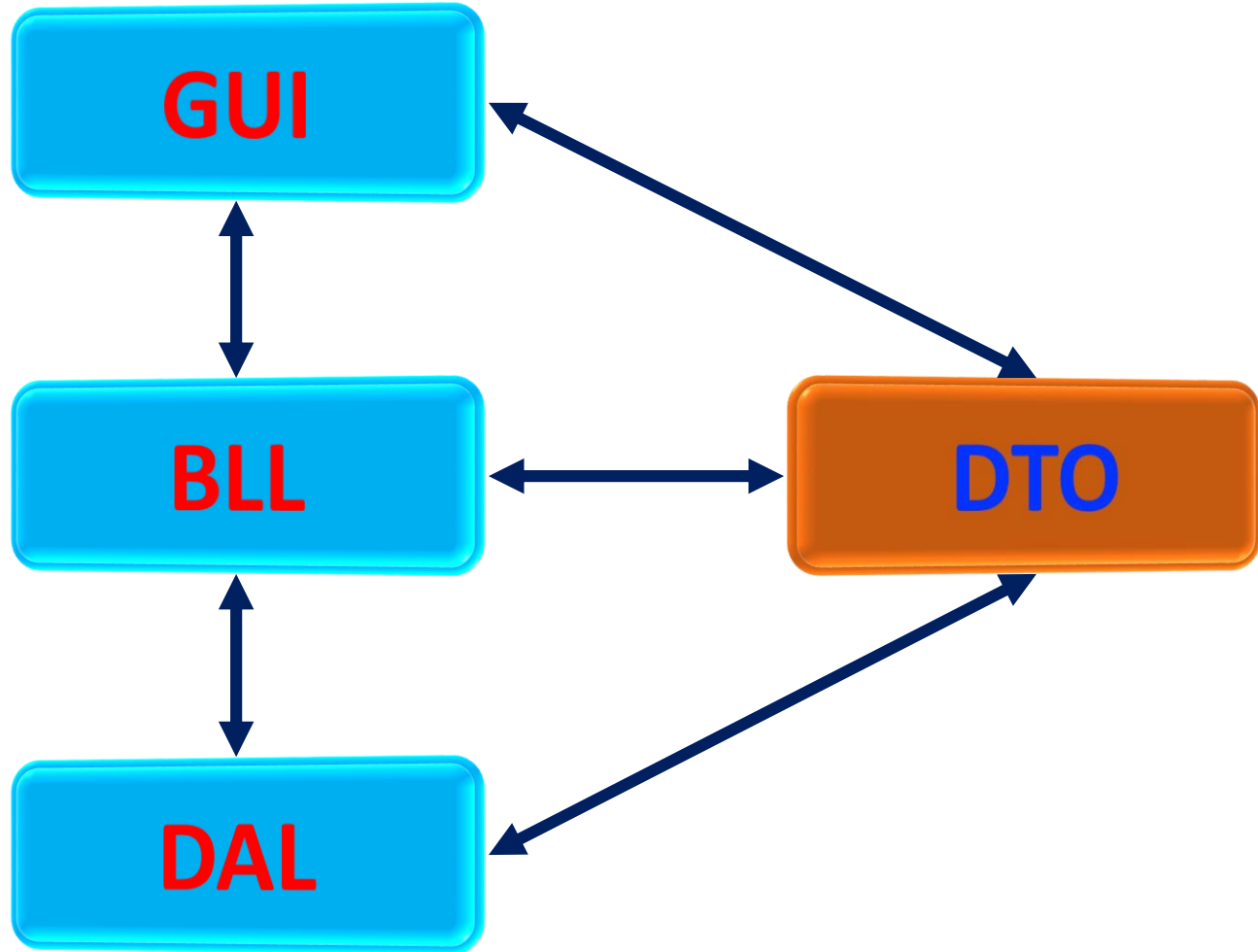


Ví dụ về mô hình 3 lớp

Ta có sơ đồ liên kết trong Mô hình 3 lớp hoạt động như sau:

- ❖ **GUI Layer:** Liên kết tới **BLL** và **DTO**.
- ❖ **Business Layer:** Liên kết tới **DAL** và **DTO**.
- ❖ **Data Access Layer:** Chỉ liên kết tới **DTO**.

Ví dụ về mô hình 3 lớp





Cách tạo project và liên kết 3 lớp

Bây giờ chúng ta bắt đầu tạo:

- ❖ Đối với Project **GUI**: Tạo theo dạng **Console Application**.
- ❖ Đối với 3 Project **DTO**, **BLL** và **DAL**: Tạo theo dạng **Class Library**.

Cách tạo Solution và project GUI

New Project

Recent

Installed

Templates

- Visual C++
- LightSwitch
- Other Languages
 - Visual Basic
 - Visual C#
 - SQL Server
 - Visual F#
 - JavaScript
- Other Project Types
- Samples

Online

.NET Framework 4.5 Sort by: Default

	Windows Forms Application	Visual C#
	WPF Application	Visual C#
	Console Application	Visual C#
	ASP.NET Web Forms Application	Visual C#
	Class Library	Visual C#
	Portable Class Library	Visual C#
	Blank App (XAML)	Visual C#
	ASP.NET MVC 3 Web Application	Visual C#
	ASP.NET MVC 4 Web Application	Visual C#

Search Installed Templates (Ctrl+E)

Type: Visual C#

A project for creating a command-line application

Name: GUI_QLSinhVien

Location: D:\DriveGoogle_2000\LTHDT\Bien soan_CSharp\Demo\ Browse...

Solution: Create new solution

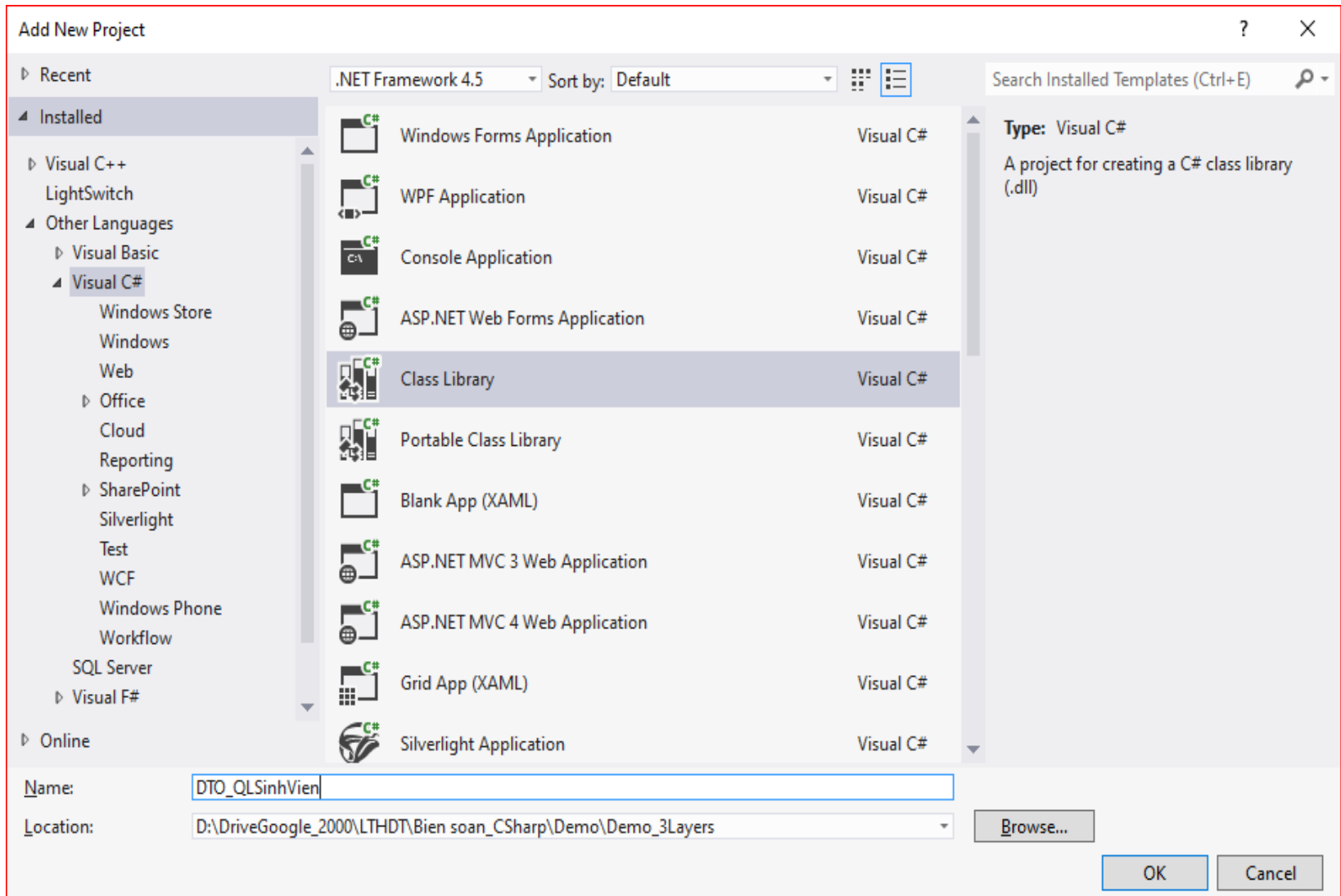
Solution name: Demo_3Layers

☒ Create directory for solution

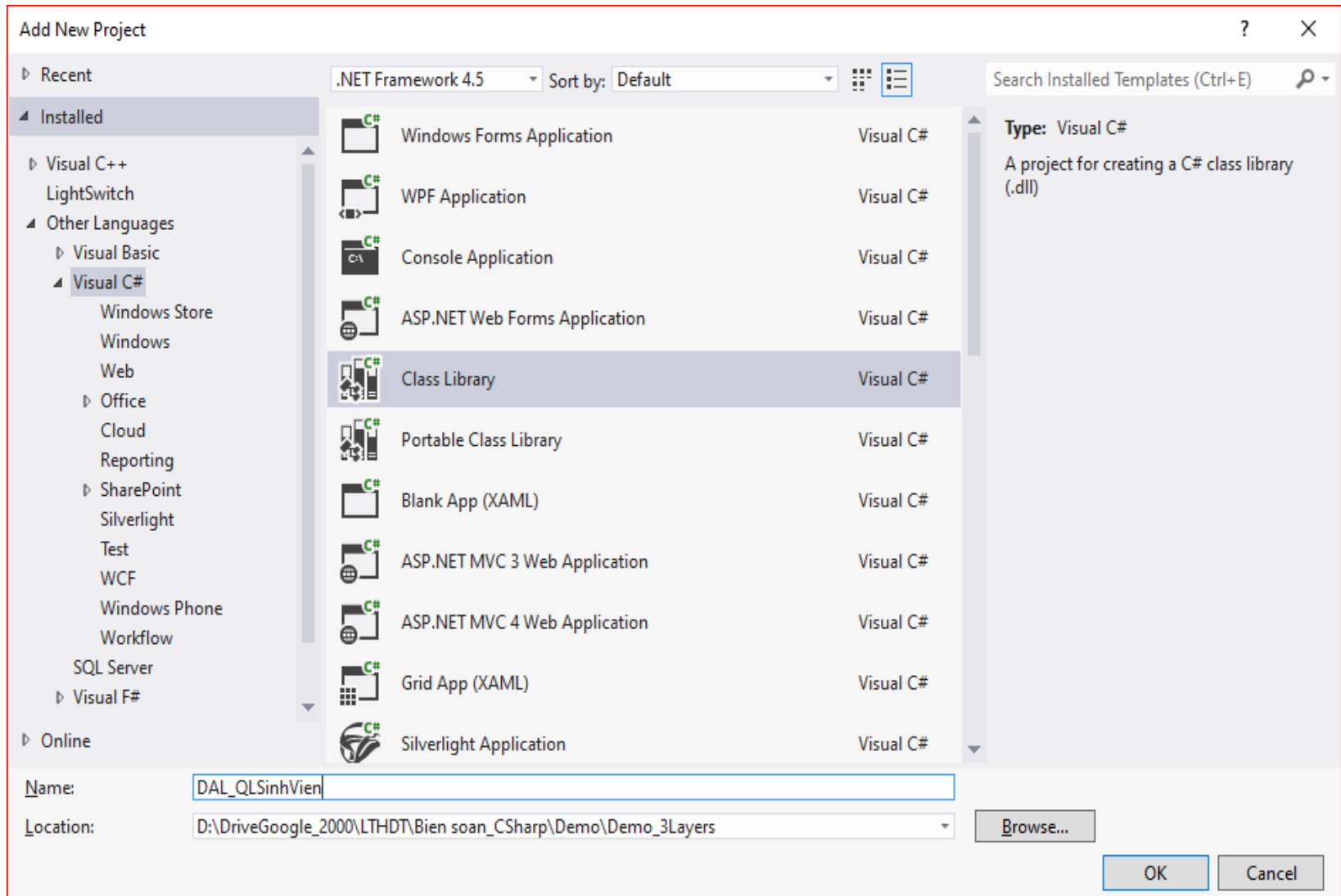
☐ Add to source control

OK Cancel

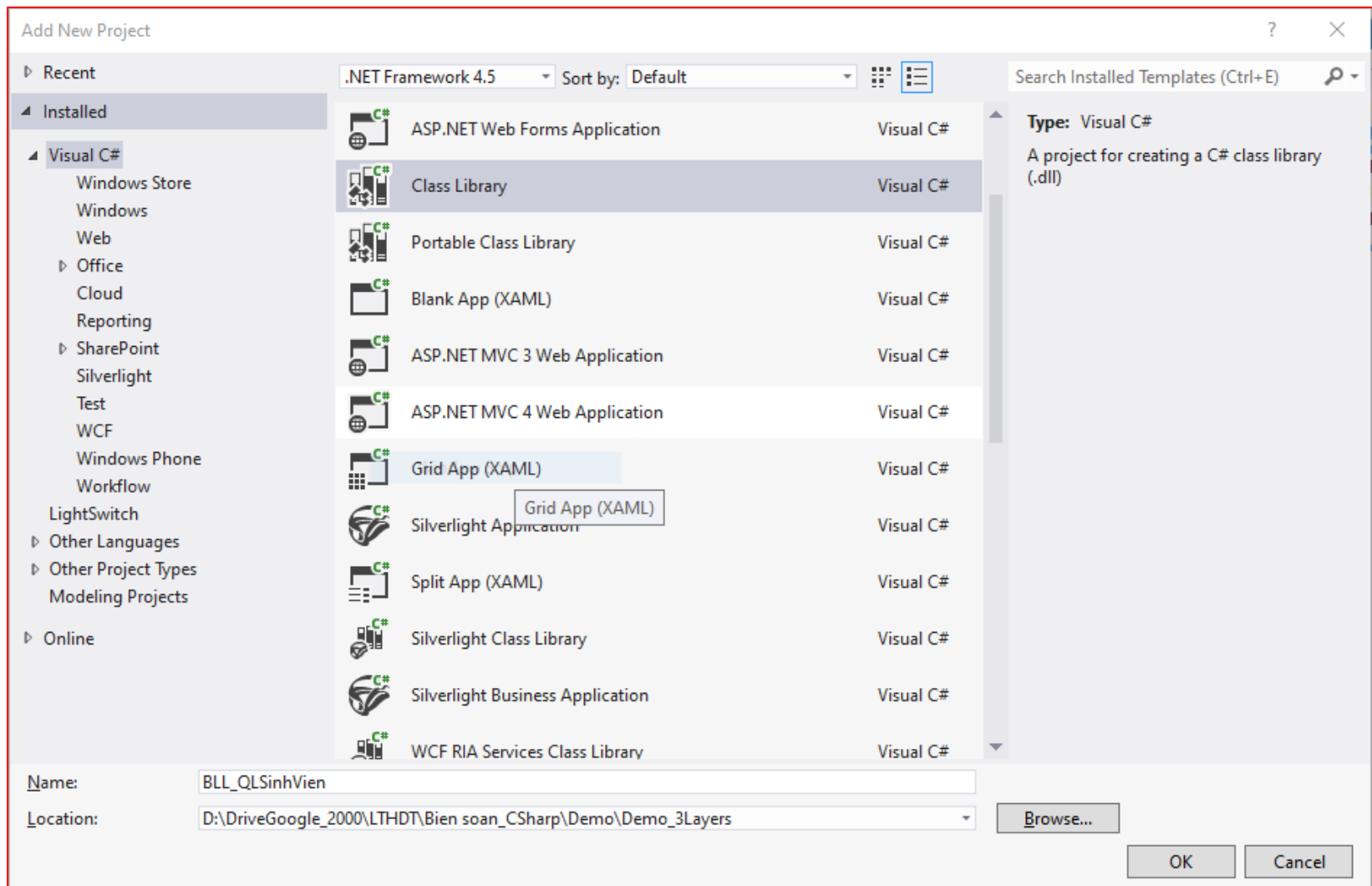
Cách tạo project DTO



Cách tạo project DAL

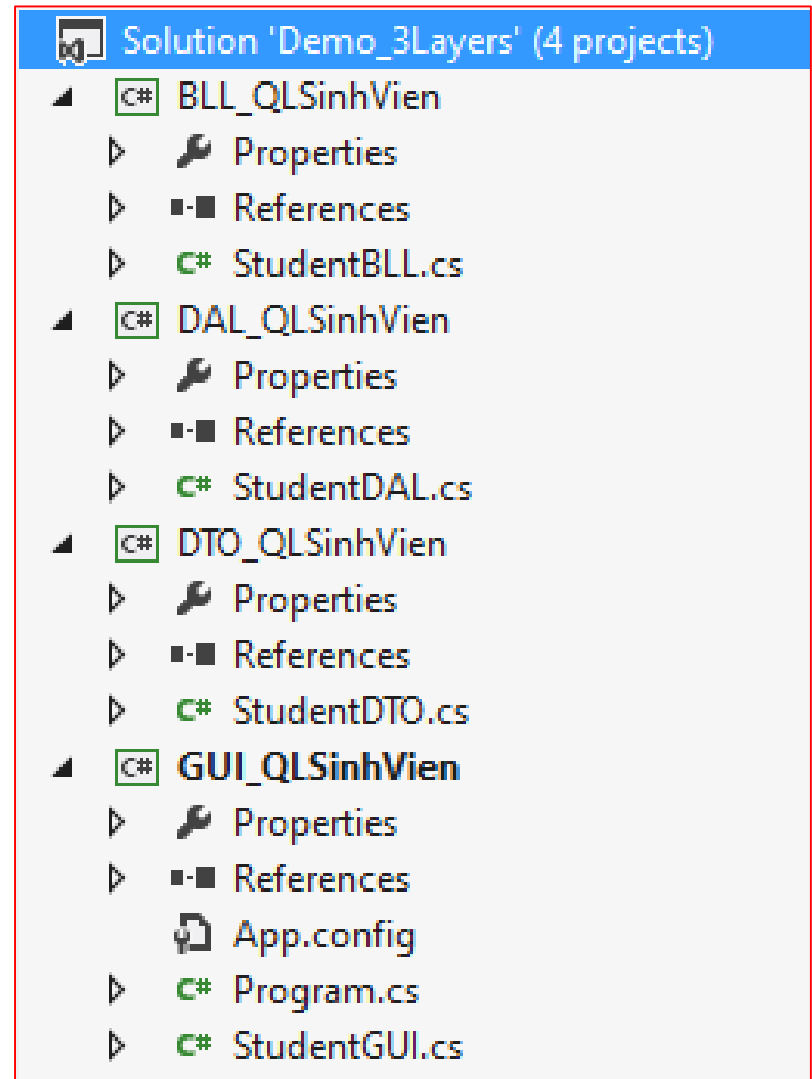


Cách tạo project BLL



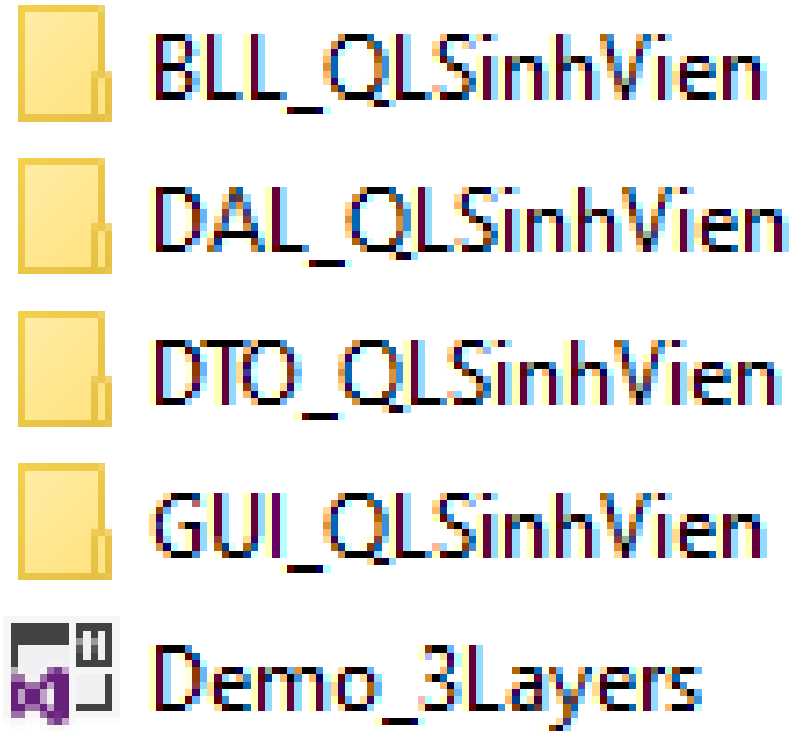
Kết quả sau khi tạo

❖ Và khi tạo xong, ta sẽ có **4 project** như hình bên (Các file mặc định **class1.cs** có thể bị xóa bỏ *hoặc* đổi tên cho phù hợp với yêu cầu).



Kết quả sau khi tạo

❖ Sau khi tạo xong chúng ta có cây Folder trên thiết bị lưu trữ như sau:



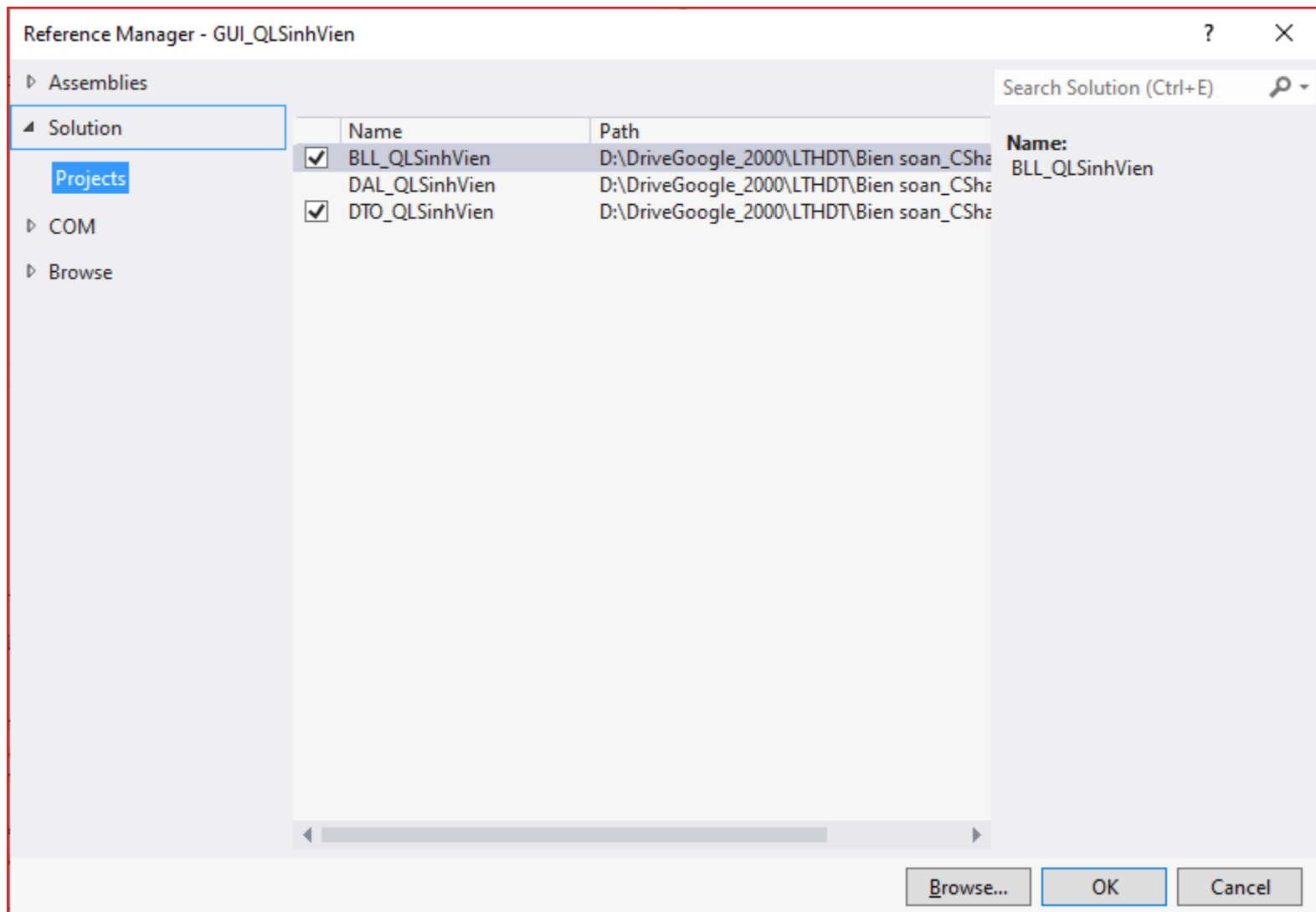


Cách liên kết các lớp lại với nhau

Bây giờ chúng ta sẽ tạo các liên kết, bắt đầu làm với project **GUI**.

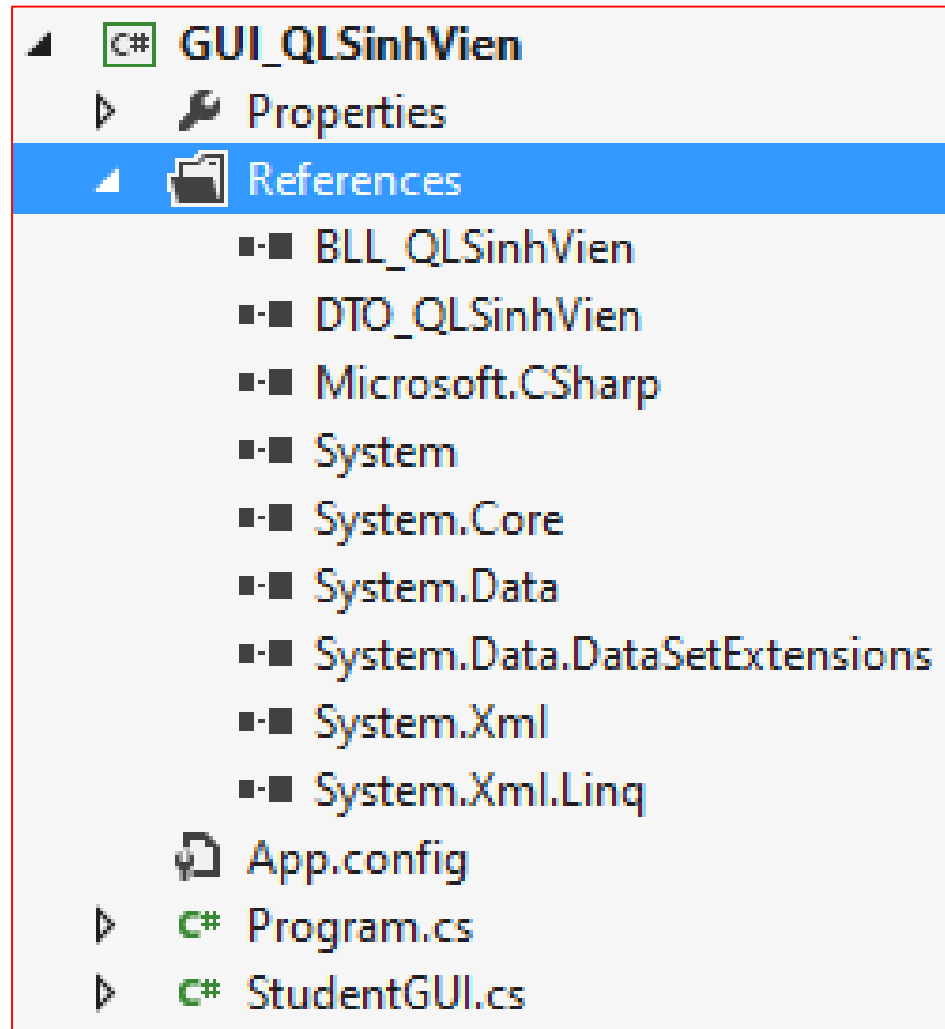
- ❖ Đầu tiên ta chọn chuột phải vào **References** → **Add Reference**.
- ❖ Chọn **Solution** → **Projects**
- ❖ Ta sẽ đánh dấu chọn **2 lớp** mà lớp **GUI** có thể liên kết tới là **BLL** và **DTO**.
- ❖ **OK**

Cách liên kết các lớp lại với nhau



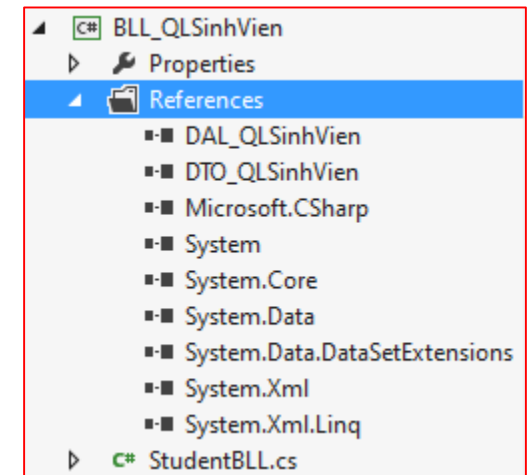
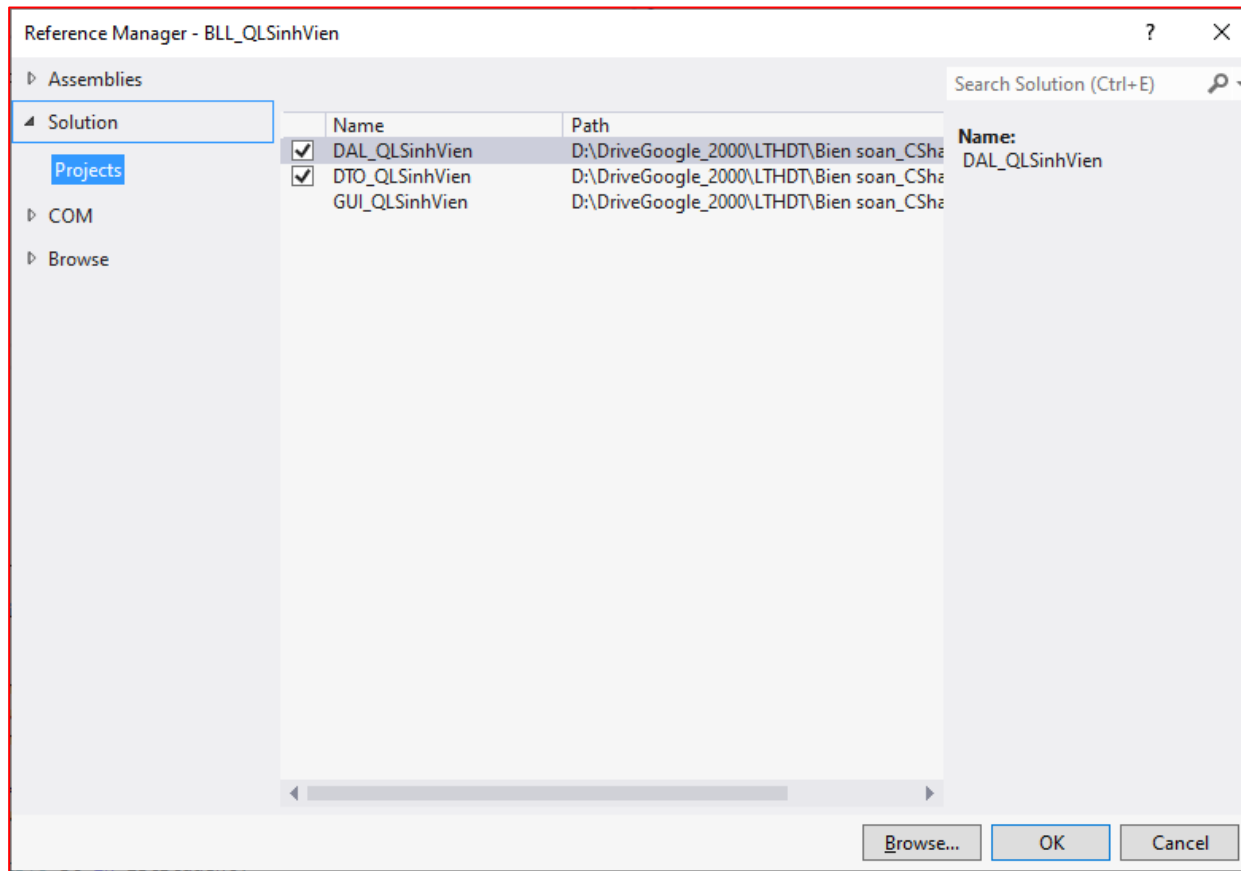


Cách liên kết các lớp lại với nhau



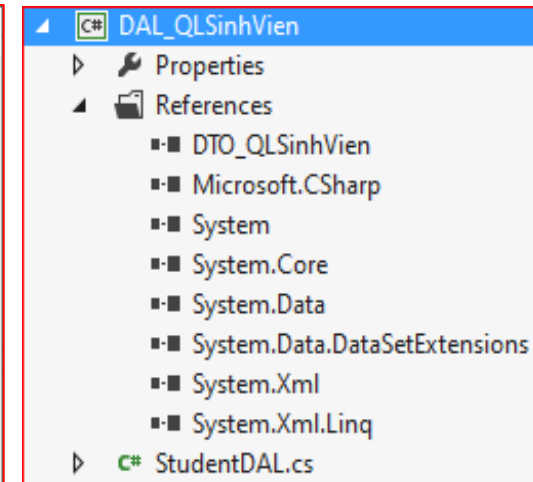
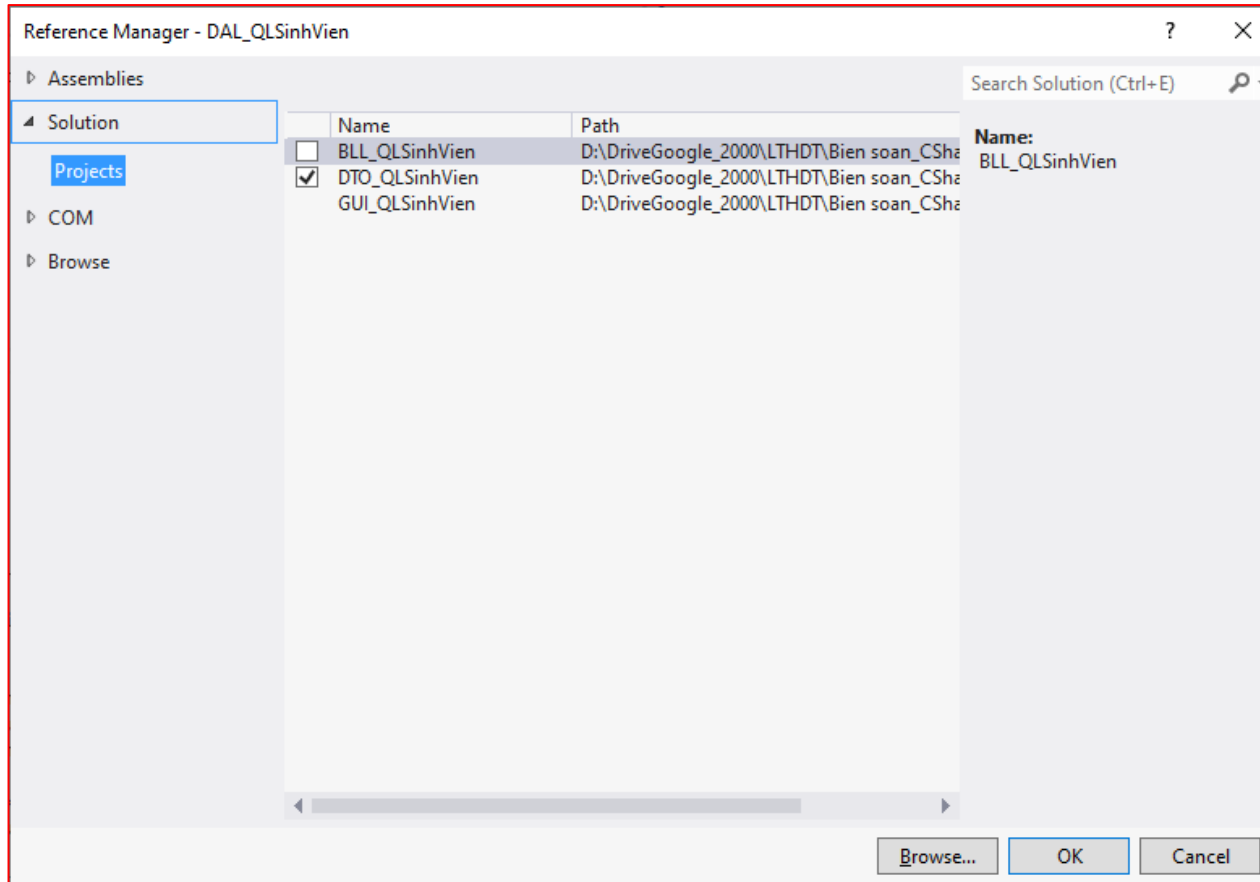
Cách liên kết các lớp lại với nhau

Làm tương tự với Lớp **BLL** như sau:



Cách liên kết các lớp lại với nhau

Làm tương tự với Lớp **DAL** như sau:



Xây dựng DTO

❖ Trong Project **DTO_QLSinhVien** ta tạo file **StudentDTO.cs** (*Class file*) có nội dung như sau:

```
namespace DTO_QLSinhVien
{
    public class StudentDTO
    {
        #region Attributes

        private String _studentID;

        public String StudentID
        {
            get { return _studentID; }
            set { _studentID = value; }
        }
    }
}
```

Xây dựng DTO

```
private String _firstName; //Tên

public String FirstName
{
    get { return _firstName; }
    set { _firstName = value; }
}

private String _lastName; //Họ và đệm

public String LastName
{
    get { return _lastName; }
    set { _lastName = value; }
}

private String _phone;

public String Phone
{
    get { return _phone; }
    set { _phone = value; }
}
```

```
private String _email;

public String Email
{
    get { return _email; }
    set { _email = value; }
}

private double _averageScore;

public double AverageScore
{
    get { return _averageScore; }
    set { _averageScore = value; }
}

#endregion
```

Xây dựng DTO

#region Methods

```
/* === Constructor === */
```

```
public StudentDTO()
```

```
{  
    this.StudentID = "";  
    this.LastName = ""; //Họ và đệm  
    this.FirstName = ""; //Tên  
    this.Phone = "";  
    this.Email = "";  
    this.AverageScore = 0;  
}
```

```
public StudentDTO(string id, string lname, string fname, string phone, string email, double avgScore)
```

```
{  
    this.StudentID = id;  
    this.LastName = lname; //Họ và đệm  
    this.FirstName = fname; //Tên  
    this.Phone = phone;  
    this.Email = email;  
    this.AverageScore = avgScore;  
}
```

Xây dựng DTO

```
//====Input/output
public string toString()
{
    string kq = StudentID + "\t\t" + LastName + "\t" + FirstName;

    if (LastName.Length <= 15)
        kq += "\t\t" + Phone;
    else if (LastName.Length <= 22)
        kq += "\t" + Phone;
    else
        kq += Phone;

    kq += "\t" + Email;

    if (Email.Length <= 15)
        kq += "\t\t" + AverageScore;
    else if (Email.Length <= 22)
        kq += "\t" + AverageScore;

    return kq;
}

#endregion
}
```

Xây dựng DAL

❖ Trong Project **DAL_QLSinhVien** ta tạo file **StudentDAL.cs** (Class file) có nội dung như sau:
Tham chiếu đến **System.Xml** và **DTO_QLSinhVien**.

```
using System.Xml;

using DTO_QLSinhVien;

namespace DAL_QLSinhVien
{
    public class StudentDAL
    {
        #region Attributes

        List<StudentDTO> _lstStudent = new List<StudentDTO>();

        #endregion
    }
}
```



Xây dựng DAL

```
#region Methods
```

```
public StudentDAL()...
```

```
internal List<StudentDTO> lstStudent  
{  
    get { return _lstStudent; }  
    set { _lstStudent = value; }  
}
```

```
public List<StudentDTO> readFile(string fileName)  
{  
    //Thiết lập để hiển thị tiếng việt trên cửa sổ Console.  
    Console.InputEncoding = UnicodeEncoding.Unicode;  
    try  
    {  
        XmlDocument read = new XmlDocument();  
        read.Load(fileName);  
        XmlNodeList nodeList = read.SelectNodes("StudentList/Student");  
        foreach (XmlNode node in nodeList)  
        {
```



Xây dựng DAL

```
//Mỗi node chứa thông tin của một sinh viên
StudentDTO st = new StudentDTO();
st.StudentID = node["StudentID"].InnerText;
st.FirstName = node["FirstName"].InnerText;
st.LastName = node["LastName"].InnerText;
st.Phone = node["Phone"].InnerText;
st.Email = node["Email"].InnerText;
st.AverageScore = double.Parse(node["AverageScore"].InnerText);
lstStudent.Add(st); //Thêm st vào danh sách
    }
    return lstStudent;
}
catch (Exception e)
{
    return null;
}
}

#endregion
}
}
```

Xây dựng BLL

- ❖ Trong Project **BLL_QLSinhVien** ta tạo file **StudentBLL.cs** (Class file) có nội dung như sau: Tham chiếu đến **DAL_QLSinhVien** và **DTO_QLSinhVien**.

```
using DTO_QLSinhVien;
using DAL_QLSinhVien;

namespace BLL_QLSinhVien
{
    public class StudentBLL
    {
        #region Attributes

        StudentDAL studentDAL = new StudentDAL();

        #endregion
    }
}
```


Xây dựng BLL

```
#region Methods
```

```
public StudentBLL()  
{  
}
```

```
public List<StudentDTO> getStudentList()  
{  
    return studentDAL.readFile("../Data/StudentList.xml");  
}
```

```
#endregion
```

```
}
```

```
}
```

Xây dựng GUI

- ❖ Trong Project **GUI_QLSinhVien** ta tạo file **StudentGUI.cs** (Class file) có nội dung như sau: Tham chiếu đến **BLL_QLSinhVien** và **DTO_QLSinhVien**.

```
using DTO_QLSinhVien;
using BLL_QLSinhVien;

namespace GUI_QLSinhVien
{
    public class StudentGUI
    {
        #region Attributes

        StudentBLL studentBLL = new StudentBLL();

        #endregion
    }
}
```

Xây dựng GUI

```
#region Methods
public void showStudentList()
{
    Console.OutputEncoding = UnicodeEncoding.Unicode;
    try
    {
        Console.WriteLine("\t\t\t\t\tSTUDENT LIST");
        string kq = "StudentID\tFull name\t\t\tPhone\t\tE-mail\t\tAverage Score";
        Console.WriteLine(kq);

        List<StudentDTO> lstStudent = new List<StudentDTO>();
        lstStudent = studentBLL.getStudentList();

        foreach (StudentDTO st in lstStudent)
        {
            Console.WriteLine(st.toString());
        }
    }
    catch (Exception e)
    {
    }
}
#endregion
}
```

Xây dựng GUI

File **Program.cs** trong **GUI_QLSinhVien** có nội dung như sau:

```
namespace GUI_QLSinhVien
{
    class Program
    {
        public static void testStudentList()
        {
            StudentGUI studentGUI = new StudentGUI();
            studentGUI.showStudentList();

            Console.ReadKey();
        }

        static void Main(string[] args)
        {
            testStudentList();
        }
    }
}
```

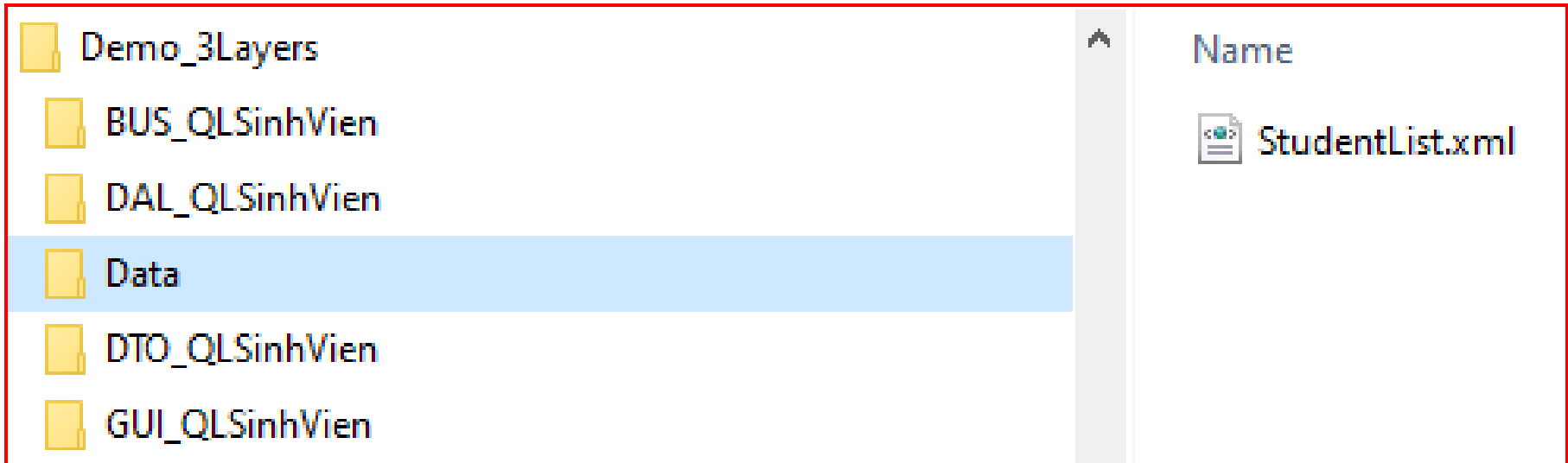


Nội dung file StudentList.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<StudentList>
  <Student>
    <StudentID>12345</StudentID>
    <LastName>Nguyễn Thị Tú</LastName>
    <FirstName>Trinh</FirstName>
    <Phone>0933726351</Phone>
    <Email>ntttrinh@gmail.com</Email>
    <AverageScore>6.27</AverageScore>
  </Student>
  <Student>
    <StudentID>34251</StudentID>
    <LastName>Lê Hoàng Diệp</LastName>
    <FirstName>Thảo</FirstName>
    <Phone>0873234567</Phone>
    <Email>lhdthao@gmail.com</Email>
    <AverageScore>6.83</AverageScore>
  </Student>
  ...

```

Bổ sung file **StudentList.xml** vào Folder **Data** như sau:





Thực thi chương trình

Kết quả sau khi chạy chương trình như sau:

STUDENT LIST					
StudentID	Full name		Phone	E-mail	Average Score
12345	Nguyễn Thị Tú	Trinh	0933726351	ntttrinh@gmail.com	6.27
34251	Lê Hoàng Diệp	Thảo	0873234567	lhdthao@gmail.com	6.83
24351	Đặng Lê Nguyên	Vũ	0345225334	dlnvu@gmail.com	8.25
14355	Phạm Nhật	Vượng	0914661145	pnvuong@gmail.com	5.67
84365	Đoàn Nguyên	Đức	0987245132	dnduc@gmail.com	6.73

