**MỤC LỤC**

TRƯỜNG ĐẠI HỌC QUY NHƠN

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

**Nguyễn Thị Loan**

**Tài liệu hướng dẫn thực hành**

**LẬP TRÌNH TRÊN WINDOWS**

**Quy Nhơn - 2015**

**HƯỚNG DẪN THỰC HÀNH CHƯƠNG 1**

**Phần 1. Thực hành theo hướng dẫn**

1. **Chương trình đầu tiên**

#### Tóm tắt

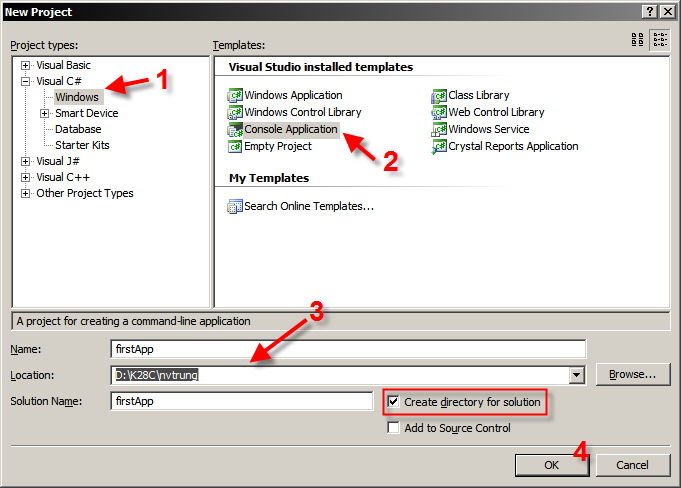
Bài thực hành này giúp bạn làm quen với môi trường Visual Studio .NET và các thao tác nhập xuất cơ bản thông qua giao diện bàn phím. Cụ thể, chương trình yêu cầu người sử dụng nhập hai số, sau đó in ra màn hình tổng, tích và thương của hai số này.

#### Kỹ thuật

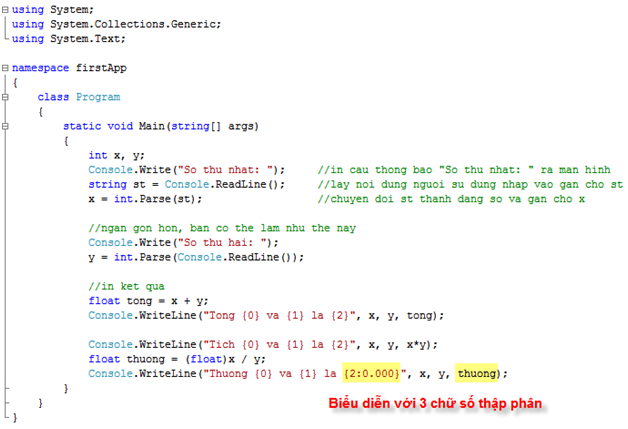
* Làm quen với môi trường Visual Studio .NET. Cấu trúc một solution, project và các tài nguyên có liên quan
* Cách thức sử dụng thư viện MSDN để tra cứu, hướng dẫn
* Sử dụng thao tác nhập xuất cơ bản

#### Trình tự thực hiện

1. Khởi động Microsoft Visual Studio 2005. Nhấn Ctrl + Shift + N hoặc chọn menu tương ứng là File → New → Project để tạo mới một project



1. Chọn loại ứng dụng cần phát triển là Visual C# → Console Application. Chọn thư mục chứa project và đặt tên cho project. Về mặt thực chất, Visual Studio coi project thuộc về một solution nào đó, và một solution có thể chứa nhiều project. Tuy nhiên, trong nhiều “bài toán” đơn giản (như ví dụ của chúng ta chẳng hạn), một solution chỉ có 1 project.
2. Đặt tên cho project là firstApp. Sau khi nhấn nút OK, hãy khảo sát xem cấu trúc của thư mục chứa solution của chúng ta.
3. Gõ mã lệnh như minh họa vào trong phần mã nguồn của tập tin Program.cs



1. Sử dụng MSDN để tra cứu các thông tin bạn chưa biết về:
   1. Console và các phương thức ReadLine(), WriteLine() của nó
   2. Cách chuyển đổi kiểu chuỗi thành số, ví dụ như int.Parse()
2. Nhấn Ctrl + F5 để thực hiện chạy chương trình. Sau đó quan sát cấu trúc thư mục của solution, cho biết sự thay đổi của nó so với khi mới được tạo ra ở bước 3.
3. Thử thay đổi kết câu lệnh

float thuong = (float)x / y;

thành

float thuong = x / y;

rồi chạy chương trình, quan sát kết quả và rút ra kết luận.

1. Sử dụng thêm các cấu trúc lệnh khác để tinh chỉnh hoạt động của chương trình (xử lý phép chia cho 0, …)
2. **Module hóa chương trình**

#### Tóm tắt

Viết chương trình nhập vào một số nguyên N từ bàn phím. Sau đó

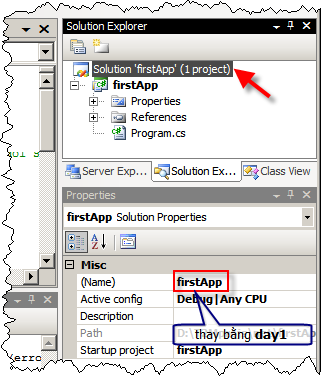
1. In ra màn hình giá trị N!.
2. Nhập thêm một số nguyên K từ bàn phím. Sau đó in ra CKN = N!/(K!\*(N-K)!)

#### Kỹ thuật được trình bày

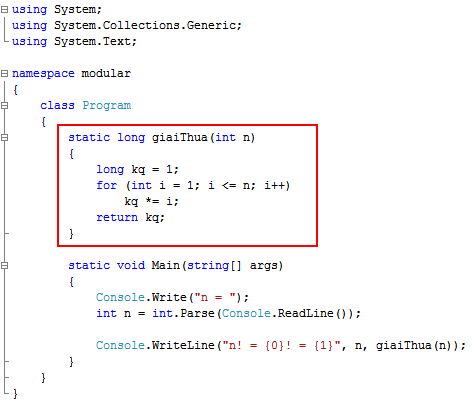
* Cấu trúc, cách quản lý logic và vật lý, cách làm việc của solution và project
* Thực hiện chia nhỏ ứng dụng để chuyên môn hóa các phần
* Cơ bản về các kiểu phương thức trong một lớp

#### Trình tự thực hiện

1. Mở solution đã làm ở Bài thực hành 1.1. Chỉnh sửa tên của solution từ “firstApp” thành “day1” cho có ý nghĩa. Xem cấu trúc thư mục của solution sau khi thay đổi.



1. Thêm một project vào solution này bằng menu lệnh File → Add → New project… . Tương tự như cách tạo mới project ở bài thực hành trước, chọn thể loại project là Console Application. Đặt tên cho project mới là “modular”.
2. Quan sát cấu trúc cây thư mục của solution trong cửa sổ Solution Explorer và cả trong Windows Explorer. Để ý rằng, trong cửa sổ Solution Explorer, project firstApp được tô đậm. Điều này có nghĩa, firstApp đóng vai trò là “Startup project”. Khi nhấn Ctrl + F5 thì project này sẽ được gọi thực thi chứ không phải là project modular mà ta mới tạo ra. Trong cửa sổ Solution Explorer, nhắp phải chuột lên “modular”, trong menu hiện ra, chọn menu lệnh “Set as Startup project” để thiết lập lại startup project cho solution.
3. Việc nhập n, tính n! rồi in kết quả bạn hoàn toàn có thể thực hiện được bằng các câu lệnh đơn giản. Tuy nhiên, để tăng tính rõ ràng và tái sử dụng, bạn nên tạo ra một phương thức để hỗ trợ việc tính toán n!. Xem mã lệnh bên dưới



1. Chạy thử chương trình để xem kết quả. Hãy để ý rằng, khai báo phương thức giaiThua là static long giaiThua(int n). Thử xóa static trong khai báo này rồi chạy lại chương trình.
2. Bằng cách tạo ra phương thức long giaiThua() như trên, chúng ta có thể giải quyết được vấn đề tính Ckn một cách dễ dàng. Lời gọi để tính Ckn như sau: GiaiThua(n)/(GiaiThua(n-k)\*GiaiThua(k))
3. Hãy tạo ra một phương thức để tính tổ hợp chập k của n phần tử (bạn tự quyết định các tham số và kiểu dữ liệu trả về).

**Phần 2. Bài tập tự làm**

Hoàn thiện những yêu cầu đã nêu ở trên

**HƯỚNG DẪN THỰC HÀNH CHƯƠNG 2**

**Phần 1. Thực hành theo hướng dẫn**

1. **Tạo thư viện sử dụng chung**

#### Tóm tắt

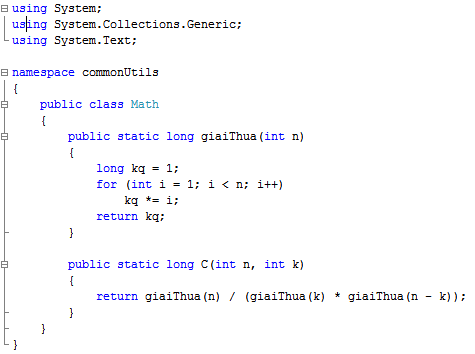
Trong thực tế, một *ứng dụng* có thể có khả năng thực thi hoặc chỉ đơn thuần là thư viện để chứa các chức năng, lớp đối tượng. Bài thực hành này hướng dẫn bạn tạo thư viện chứa các phương thức thường dùng. Với mục đích minh họa, thư viện này chỉ chứa 2 hàm tiện ích giúp tính giai thừa và tổ hợp chập. Sau khi biên dịch, bạn sẽ có được một file nhị với phần mở rộng là DLL. Thư viện này, khi cần, sẽ được tham chiếu đến trong các ứng dụng khác.

#### Kỹ thuật

* Tạo loại ứng dụng loại thư viện

#### Trình tự thực hiện

1. Tạo mới một project, đặt tên là commonUtils (common utilities - các tiện ích dùng chung). Chú ý chọn loại ứng dụng cần tạo là Class Library
2. Mặc định Visual Studio sẽ tạo ra trong namespace CommonUtils một lớp tên là Class1. Đổi tên lớp này lại thành Math. Sau đó cài đặt các phương thức như sau:



1. Rõ ràng, đây không phải là một chương trình để chạy như các ứng dụng bạn đã viết trước đó - class Math không có phương thức static public Main() – tức là bạn không thể nhấn Ctrl + F5 để chạy chương trình. Biên dịch project này bằng menu lệnh *Build → Build commonUtils*. Kết quả, bạn sẽ có một thư viện commonUtils.dll trong thư mục bin\Release hoặc bin\Debug của project tùy theo cách chọn chế độ biên dịch. Thư viện này sẽ được dùng để tham chiếu đến trong các ứng dụng cần nó.

#### Mở rộng

Bổ sung các phương thức thường dùng khác vào thư viện, chẳng hạn như phương thức xác định xem một số có phải là nguyên tố hay không, phương thức hoán đổi giá trị của hai số cho trước, …

1. **Tam giác Pascal**

#### Tóm tắt

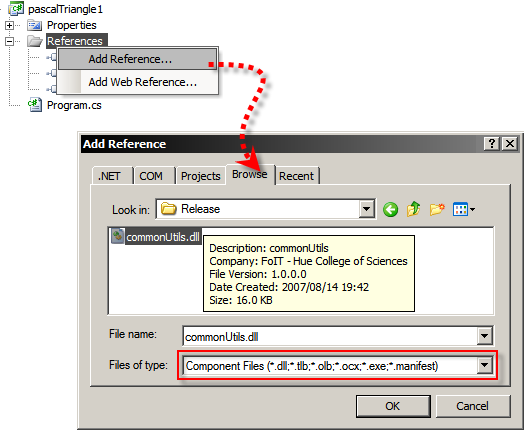
Viết chương trình nhập một số nguyên N từ bàn phím, sau đó in ra màn hình N dòng đầu tiên của tam giác Pascal.

#### Kỹ thuật

* Sử dụng thư viện có sẵn

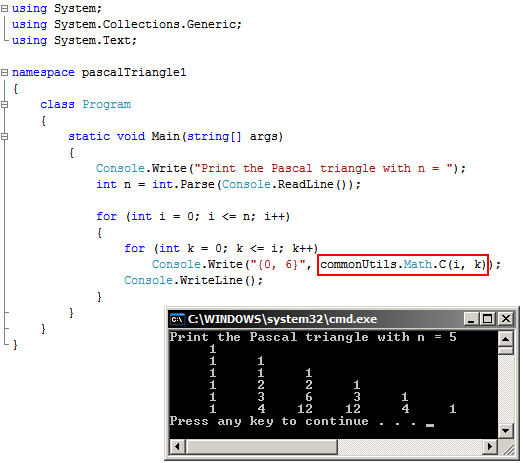
#### Trình tự thực hiện

1. Tạo mới một ứng dụng kiểu Console Application. Đặt tên project là pascalTriangle1
2. Thực hiện bổ sung tham khảo đến thư viện commonUtils bằng cách:
   * Nhắp phải chuột vào project pascalTriangle1 trong cửa sổ Solution Explorer
   * Trong menu hiện ra, chọn Add Reference…



Trong tab Browse của hộp thoại Add Reference, tìm đến thư viện commonUtils.dll đã tạo ra trước đó ➔ Dễ thấy rằng thư viện được tham khảo không chỉ có dạng DLL mà có thể có các dạng khác

1. Hoàn thiện phần mã nguồn có sử dụng tham chiếu đến thư viện vừa bổ sung như hình dưới:



#### Mở rộng

Hãy tự rút ra những ghi chú cần thiết về việc:

* Khai báo phương thức C(int n, int k) trong commonUtils là public static long C(int n, int k), static, public ở đây có ý nghĩa gì, có thể thay thế hoặc bỏ đi?
* Tương tự cho phương thức giaiThua(int n);
* Tại sao trong quá trình sử dụng phương thức C() lại phải ghi đầy đủ là commonUtils.Math.C()? Chỉ cần ghi Math.C() có được không?

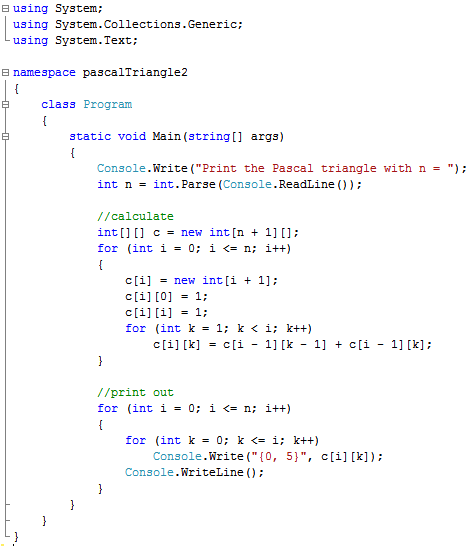
1. **Tam giác Pascal – array version**

#### Tóm tắt: Sử dụng array để xây dựng tam giác Pascal như Bài thực hành 1.4.

#### Kỹ thuật: Sử dụng array

#### Trình tự thực hiện

1. Tạo mới một project kiểu Console Application với tên là pascalTriangle2
2. Sử dụng các tính chất C00 = Ckk = 1, Cnk = Cn-1k-1 + Cn-1k , ta sẽ xây dựng một jagged array từ nhỏ đến lớn. Chi tiết như phần mã nguồn phía dưới:

****

#### Mở rộng

Có thể dùng array nhiều chiều trong trường hợp này không? Nếu có thì có sự khác nhau nào so với dùng jagged array?

**Phần 2. Bài tập tự làm**

Hoàn thiện những yêu cầu đã nêu ở trên

**HƯỚNG DẪN THỰC HÀNH CHƯƠNG 3**

**Phần 1. Thực hành theo hướng dẫn**

1. **Quản lý sinh viên**

#### Tóm tắt

Viết chương trình quản lý sinh viên của một trường. Sinh viên có thể học các chuyên ngành Công nghệ Thông tin, Vật lý, Ngữ văn. Mỗi chuyên ngành tương ứng có các môn học khác nhau.

* Sinh viên khoa Công nghệ Thông tin phải học 3 môn Pascal, C# và SQL.
* Sinh viên khoa Vật lý phải học 4 môn: Cơ học, Điện học, Quang học, Vật lý hạt nhân.
* Sinh viên khoa Văn phải học 2 môn Văn học cổ điển và Văn học Hiện đại

Chương trình cho phép nhập danh sách sinh viên, sau đó in danh sách sinh viên cùng với điểm trung bình của họ ra màn hình.

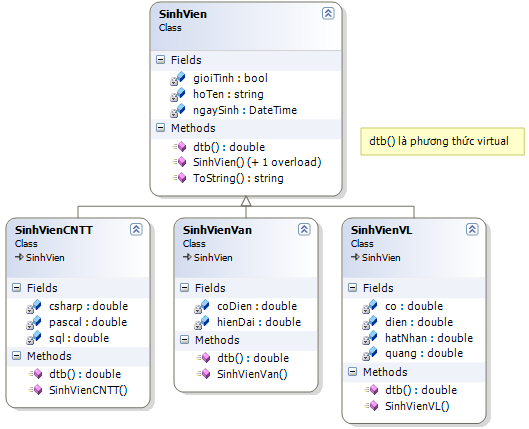
In ra danh sách những sinh viên có điểm trung bình cao trên 5.0 ra màn hình. Thông tin hiển thị có dạng Họ tên, Chuyên ngành đào tạo, Điểm trung bình.

#### Kỹ thuật

* Truy xuất tập tin có định dạng cho trước
* Sử dụng một phương thức của lớp String
* Các kỹ thuật hướng đối tượng được sử dụng trong bài toán thực tế

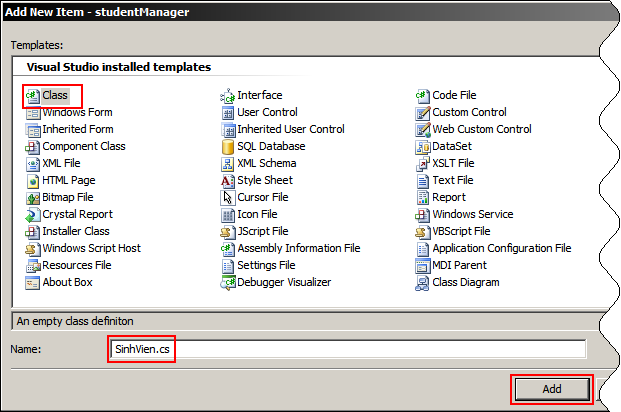
#### Trình tự thực hiện

1. Trước khi tiến hành cài đặt, ta khảo sát qua sơ đồ lớp được sử dụng. Với những mô tả khá rõ ràng trong yêu cầu bài toán, ta có được cái nhìn tổng quan về các lớp như sau:

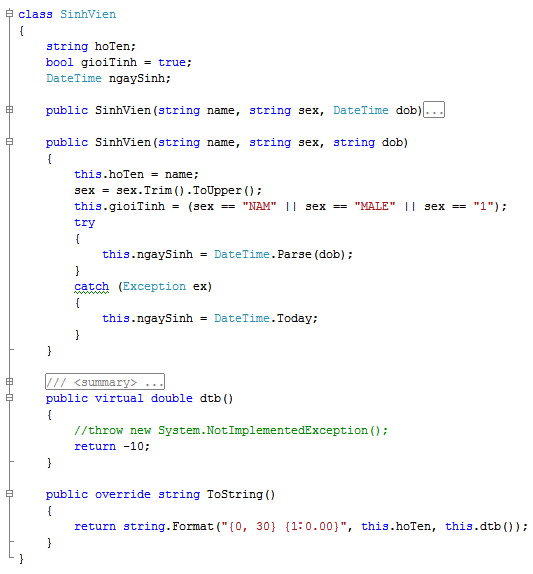


Lưu ý: phương thức dtb() được cài đặt là virtual để chúng ta có thể override một cách cụ thể, chi tiết hơn trong các lớp kế thừa từ class SinhVien. Phương thức ToString() được cài đặt override từ lớp object để sử dụng trong việc in “nội dung” của đối tượng.

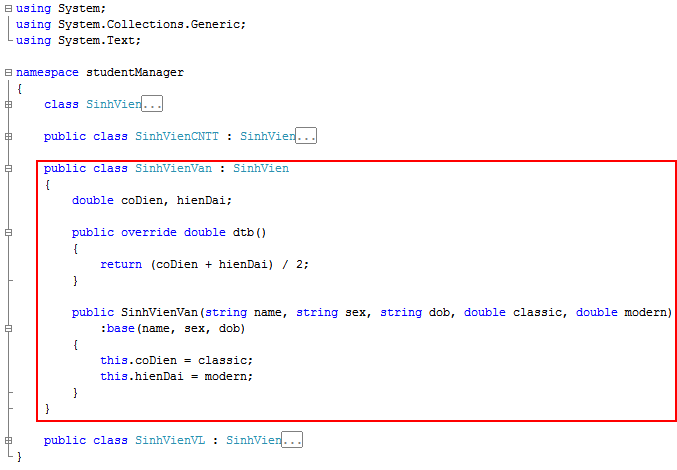
1. Tạo mới một project kiểu Console Application với tên là studentManager
2. Tại cây phân cấp Solution Explorer nhắp phải chuột và chọn Add →New Item… Trong hộp thoại hiện ra, chọn tạo mới class SinhVien.cs



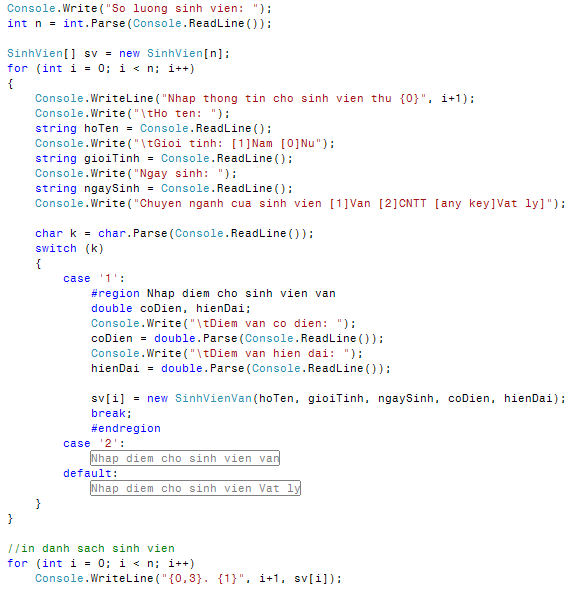
1. Cài đặt các thành phần cơ bản cho lớp SinhVien



1. Bổ sung thêm các class SinhVienCNTT, SinhVienVan, SinhVienVL theo phân tích thiết kế lớp từ trước. Dưới đây là phần mô tả cài đặt cho lớp SinhVienVan. Hai lớp còn lại SinhVienCNTT, SinhVienVL được cài đặt một cách tương tự.



1. Trong phần chương trình (tập tin Program.cs) ta thực hiện yêu cầu bài toán như sau:



#### Yêu cầu thêm

* In ra 3 sinh viên có điểm trung bình cao nhất trường.
* Chỉnh sửa để người sử dụng có thể nhập danh sách mà không biết trước số lượng sinh viên (sử dụng vòng lặp while, do, …)
* Chỉnh sửa để có thể nhập dữ liệu các sinh viên từ file.

**Phần 2. Bài tập tự làm**

Hoàn thiện những yêu cầu thêm

**HƯỚNG DẪN THỰC HÀNH CHƯƠNG 4**

**Phần 1. Thực hành theo hướng dẫn**

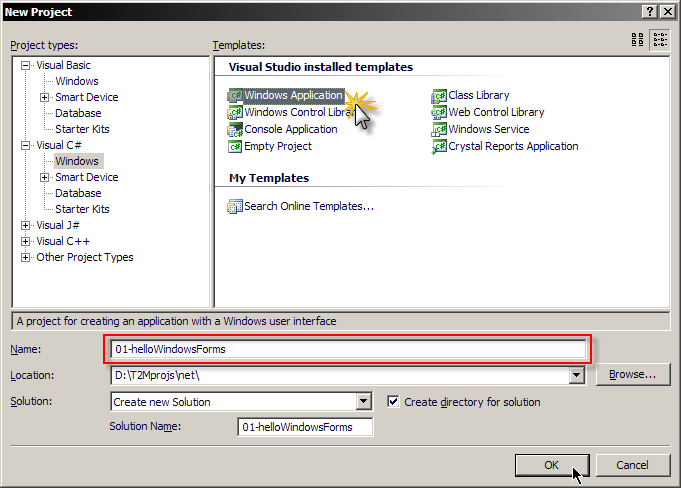
1. **helloWinForms**

#### Kỹ thuật

* Cấu trúc và cơ chế hoạt động của một project Windows Form Application.
* Cơ chế xử lý sự kiện của các Control trong một Windows Form
* Một số phương thức, thuộc tính, sự kiện quan trọng của các điều khiển trong một Windows Form.

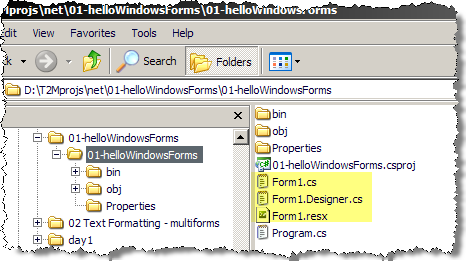
#### Trình tự thực hiện

1. Tạo mới một ứng dụng kiểu Windows Form Application với tên là   
   01-helloWindowsForm như hình vẽ



1. Theo mặc định, một solution với một project được tạo ra. Project này có một lớp Form1.

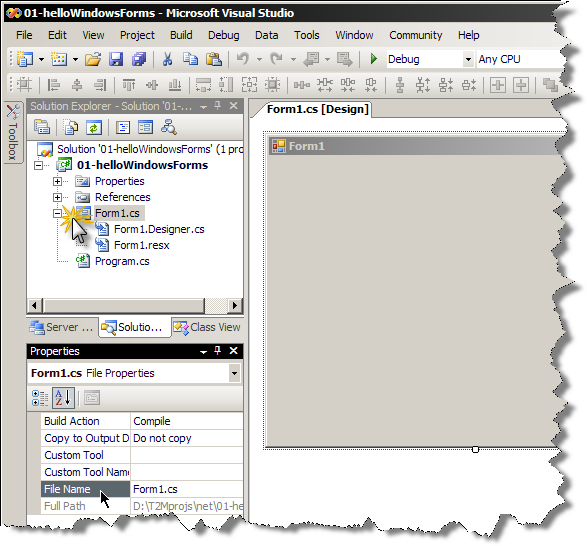
Khảo sát nội dung của project trong Windows Explorer, chúng ta sẽ thấy cấu trúc của thư mục và các tập tin tương tự như hình dưới:



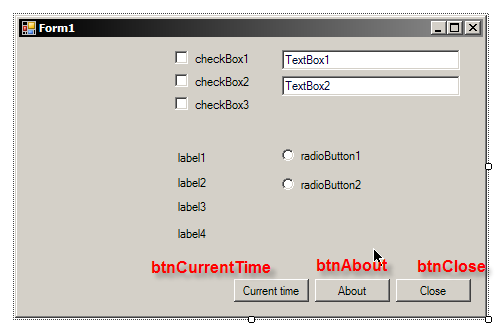
Có thể thấy, mỗi Form được tạo ra tương ứng với 3 tập tin có tiếp đầu ngữ là giống nhau, lấy ví dụ là Form1

* Form1.Designer.cs: chứa các mã lệnh do Form Designer tự sinh ra tương ứng với các thao tác do người sử dụng *kéo thả các Control từ ToolBox vào bề mặt Form* hay *thực hiện các thiết lập đối với các Control*.
* Form1.cs: chứa phần mã lệnh và khai báo thêm do người sử dụng cài đặt.
* Form1.resx: chứa các mô tả, khai báo về các tài nguyên được sử dụng trong Form.

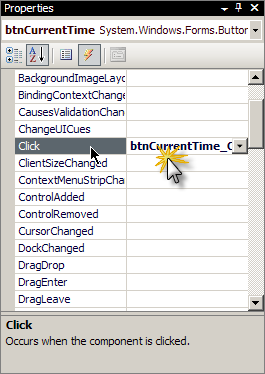
1. Quan sát cấu trúc của solution hay project bằng cách khảo sát cửa sổ Solution Explorer:



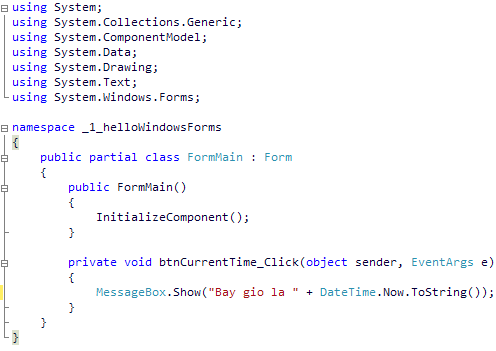
1. Từ cửa sổ Solution Explorer, đổi tên tập tin Form1.cs thành FormMain.cs. Để ý rằng, cả ba tập tin liên quan đến Form1 đều được thay đổi theo một cách đồng bộ.
2. Thiết kế giao diện cho FormMain như hình vẽ



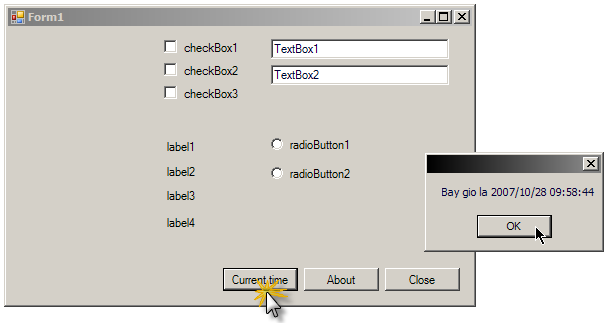
1. Bước tiếp theo, chúng ta sẽ thực hiện cài đặt phương thức xử lý sự kiện Click của nút bấm btnCurrentTime:
   1. Chọn điều khiển nút bấm btnCurrentTime trong cửa số thiết kế Form.
   2. Ở trang Event trong cửa sổ Properties, nhắp đúp chuột vào sự kiện Click (xem hình vẽ dưới). Form Designer sẽ sinh ra phương thức xử lý sự kiện có tên mặc định là btnCurrentTime\_Click(…). (Phương thức xử lý sự kiện được mặc định đặt tên là <tênĐiềuKhiển>\_<TênSựKiện>)



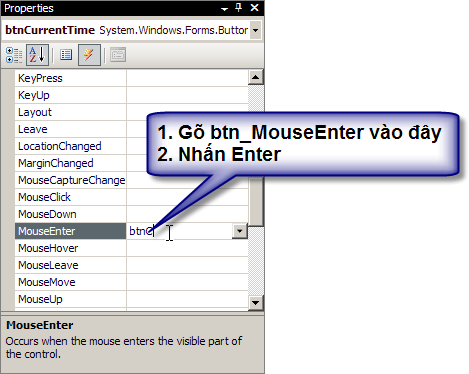
Soạn thảo phần mã lệnh cho phương thức này như sau:



1. Thực hiện chạy chương trình, khi nhấn vào nút bấm btnCurrentTime, một hộp thông báo được hiển thị ra như hình vẽ



1. Thực ra chúng ta có thể tự đặt tên cho phương thức xử lý sự kiện. Chẳng hạn, để cài đặt phương thức xử lý sự kiện MouseEnter cho nút bấm btnCurrentTime, trong cửa sổ Properties ở trang Events, tìm đến mục MouseEnter và:
   1. Nhập vào tên phương thức xử lý sự kiện: btn\_MouseEnter
   2. Nhấn Enter
   3. FormDesigner sẽ tạo ra phương thức với tên tương ứng



* 1. Tiến hành cài đặt mã lệnh cho phương thức xử lý sự kiện trên như sau:

private void btn\_MouseEnter(object sender, EventArgs e)

{

btnCurrentTime.ForeColor = Color.Red;

}

1. Tương tự, chúng ta cài đặt tiếp phương thức xử lý sự kiện MouseLeave cho nút bấm btnCurrentTime như sau

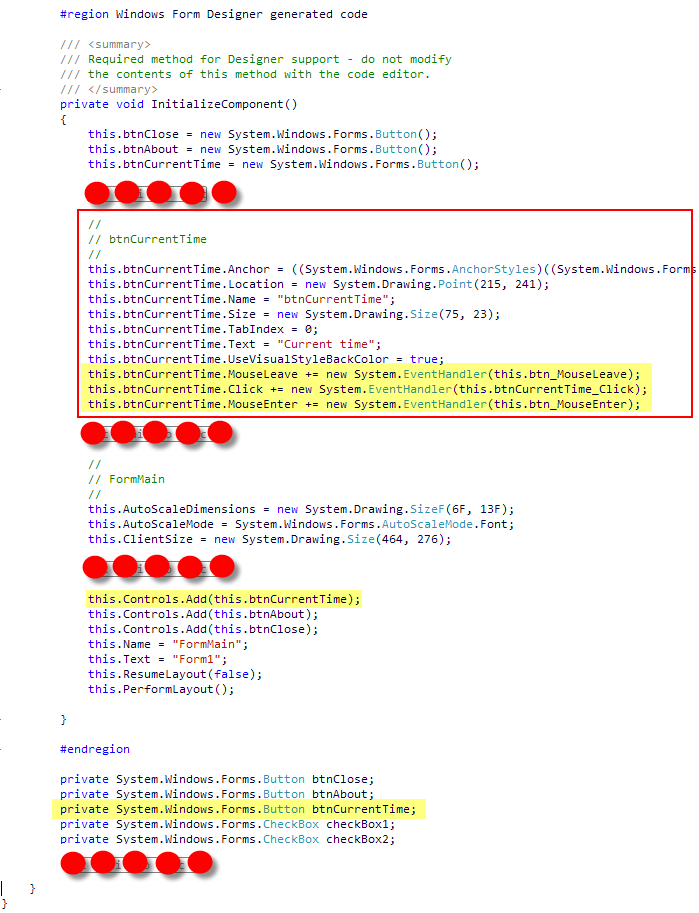
private void btn\_MouseLeave(object sender, EventArgs e)

{

btnCurrentTime.ForeColor = SystemColors.ControlText;

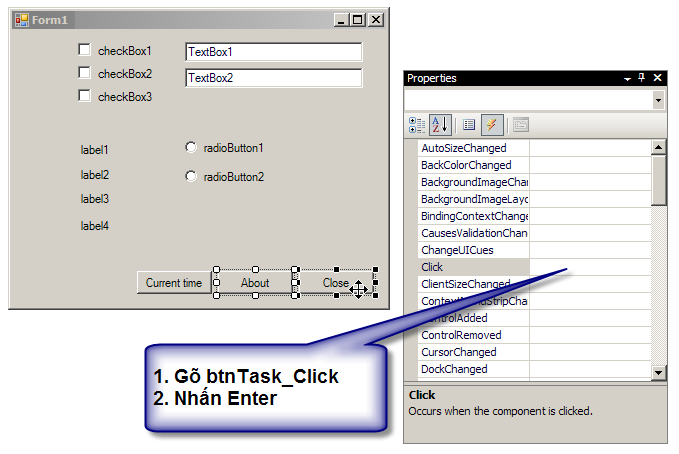
}

1. Chạy chương trình và quan sát kết quả: Điều khiển nút bấm btnCurrentTime sẽ có hiệu ứng *mouse hover* khá ấn tượng: khi rê con trỏ chuột vào nút bấm btnCurrentTime, màu chữ của nó sẽ đổi sang màu đỏ; màu chữ của nút bấm trở thành bình thường (màu ControlText) khi con trỏ chuột rê ra khỏi nút bấm.
2. Để tìm hiểu kỹ hơn bản chất của việc gắn kết phương thức xử lý sự kiện, chúng ta nhắp đúp chuột vào FormMain.Designer.cs trong cửa sổ Solution Explorer để xem phần nội dung được sinh ra bởi Form Designer:



Chú ý những phần được tô sáng trong hình vẽ nói trên; từ đó suy ra được bản chất của việc gắn kết phương thức xử lý sự kiện trong khi thiết kế.

1. Đóng file nội dung FormMain.Designer.cs lại. Các bước tiếp theo sẽ minh họa cách thức dùng chung một phương thức xử lý sự kiện cho nhiều đối tượng khác nhau.
2. Trong cửa sổ thiết kế của FormMain, thực hiện
   1. Chọn cả hai đối tượng btnClose và btnAbout
   2. Trong trang Events của cửa sổ Properties, gõ tên phương thức xử lý sự kiện Click cho cả hai điều khiển nút bấm này là **btnTask\_Click** rồi nhấn Enter (xem hình vẽ)



1. Thực hiện cài đặt mã lệnh cho phương thức này như sau:

private void btnTask\_Click(object sender, EventArgs e)

{

if (sender == btnClose)

this.Close();

else if (sender == btnAbout)[[1]](#footnote-1)

MessageBox.Show("Day la chuong trinh minh hoa", "Thong bao");

}

Trong phương thức trên, chúng ta sử dụng đối số sender để nhận biết điều khiển nào phát sinh sự kiện. Chúng ta cũng có thể thực hiện như thế này:

private void btnTask\_Click(object sender, EventArgs e)

{

string stTask = (sender as Button).Text; [[2]](#footnote-2)

if (stTask == "Close")

this.Close();

else if (stTask == "About")

MessageBox.Show("Day la chuong trinh minh hoa", "Thong bao");

}

1. Bây giờ, chúng ta tinh chỉnh thêm để chương trình hỗ trợ hiệu ứng *mouse hover* *cho tất cả các điều khiển* trong form:
   1. Sửa lại phần mã nguồn cho 2 phương thức xử lý sự kiện btn\_MouseEnter và btn\_MouseLeave như sau:

private void btn\_MouseEnter(object sender, EventArgs e)

{

(sender as Control).ForeColor = Color.Red;

}

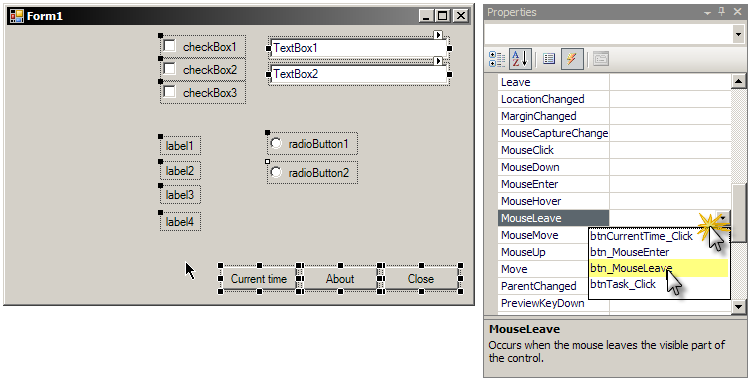
private void btn\_MouseLeave(object sender, EventArgs e)

{

(sender as Control).ForeColor = SystemColors.ControlText;

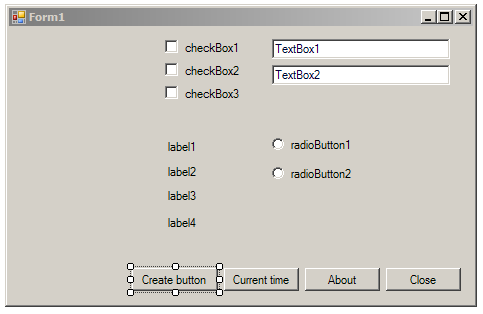
}

* 1. Trong phần FormDesigner, chọn tất cả các đối tượng trên bề mặt Form.
  2. Trong cửa sổ Properties, chọn phương thức xử lý sự kiện MouseLeave cho *tất cả các đối tượng đang chọn* là btn\_MouseLeave (xem hình vẽ)

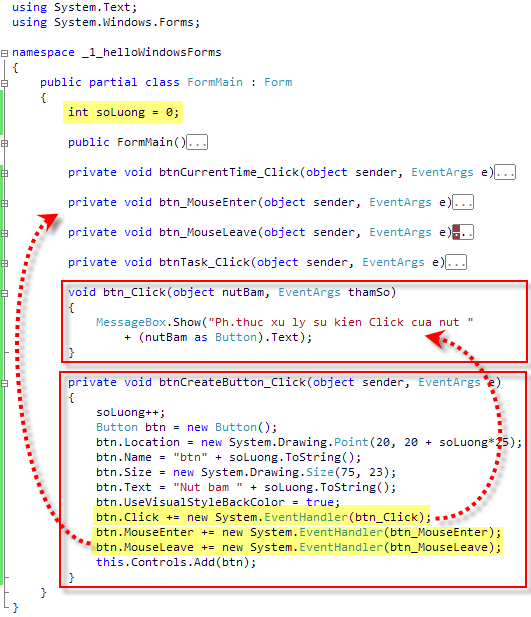


* 1. Làm tương tự để gán phương thức xử lý sự kiện MouseEnter cho tất cả các điều khiển nói trên là btn\_Enter.
  2. Chạy chương trình để xem hiệu ứng: khi rê con trỏ chuột qua các điều khiển, font chữ của chúng sẽ được đổi thành màu đỏ.

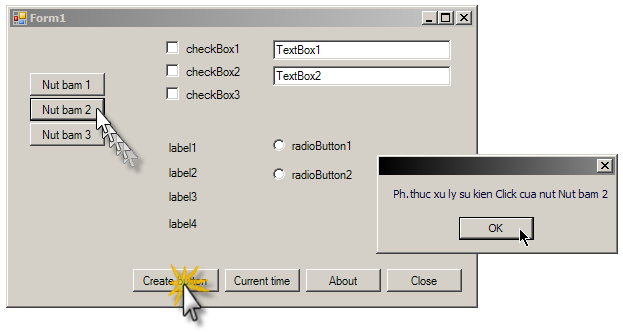
1. Trong bước 11, chúng ta đã biết được cách thức đưa một thành phần điều khiển vào giao diện của một Windows Form thông qua mã lệnh (bằng cách tìm hiểu phần mã sinh ra bởi Form Designer). Bây giờ, chúng ta sẽ áp dụng để thực hiện thêm các điều khiển vào Form và gán phương thức xử lý sự kiện cho chúng trong thời gian thực thi chương trình
   1. Bổ sung vào Form một nút bấm btnCreateButton



* 1. Cài đặt phương thức xử lý sự kiện Click cho nút bấm này như sau:



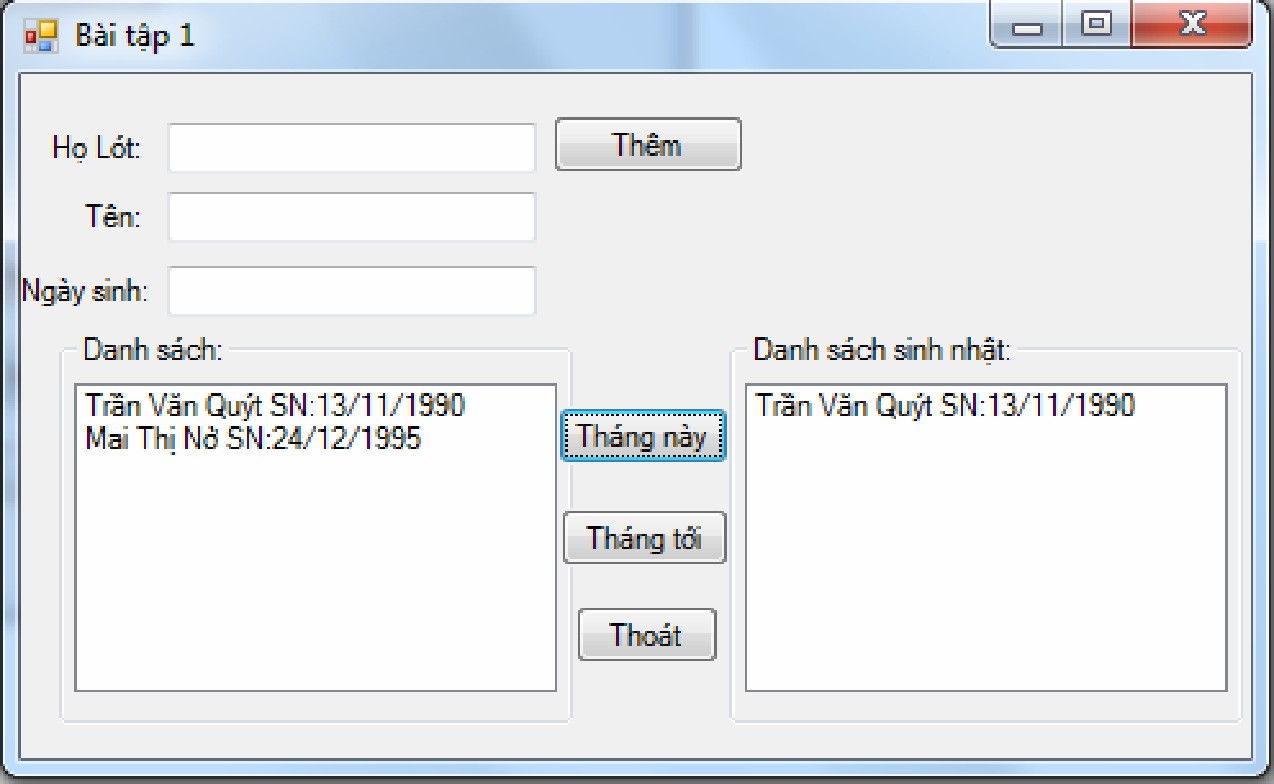
* 1. Chạy chương trình và quan sát kết quả.



#### Mở rộng

* Hãy tìm hiểu ý nghĩa của việc cài đặt mã lệnh ở bước 15.a: (sender as Control). Có thể sử dụng phép ép kiểu nào khác không? Tại sao?
* Điều chỉnh trong giao diện chương trình, trong đó có một số điều khiển (Label, TextBox, RadioButton, CheckBox hoặc Button) sử dụng màu khác với màu mặc định (là SystemColors.ControlText). Khi đó, hiệu ứng *mouse hover* hoạt động không đúng nữa. Hãy chỉnh sửa chương trình để khắc phục phát sinh này.

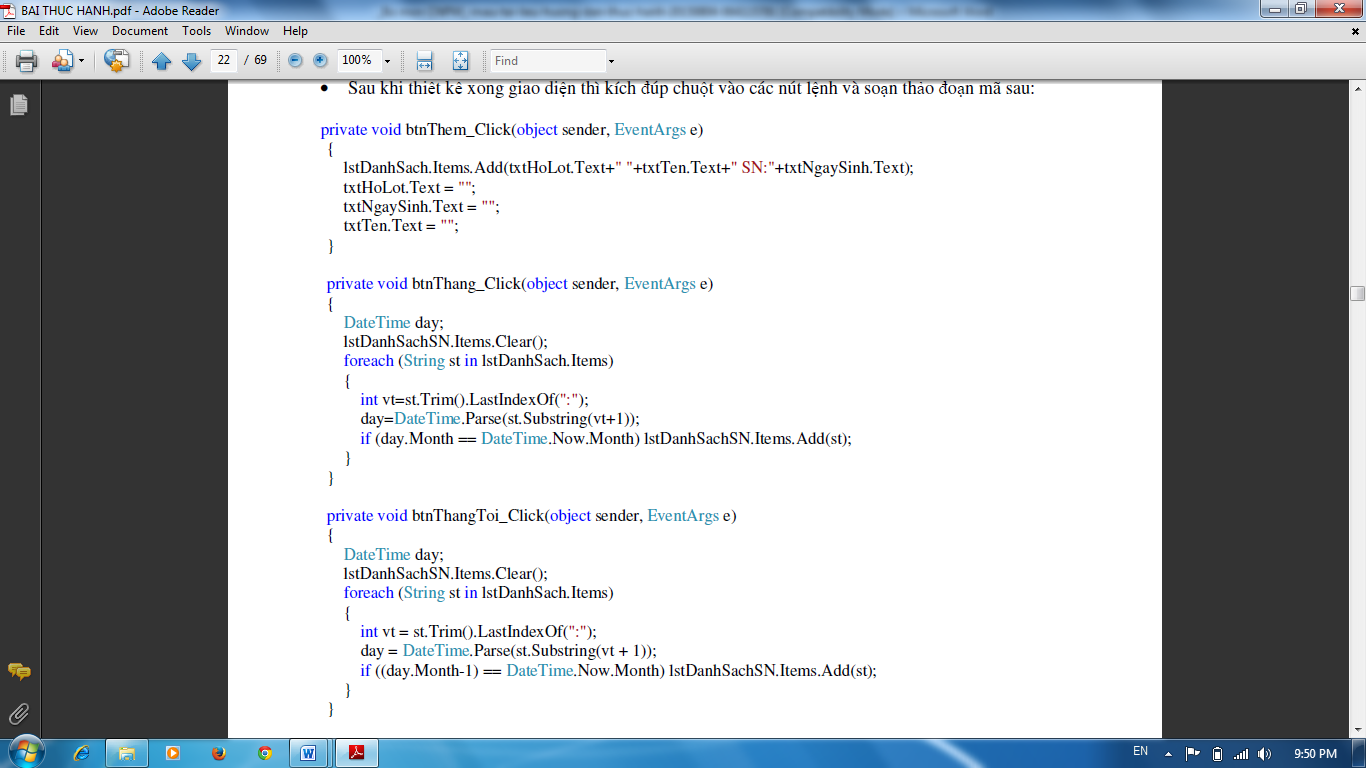
1. **usingControls**



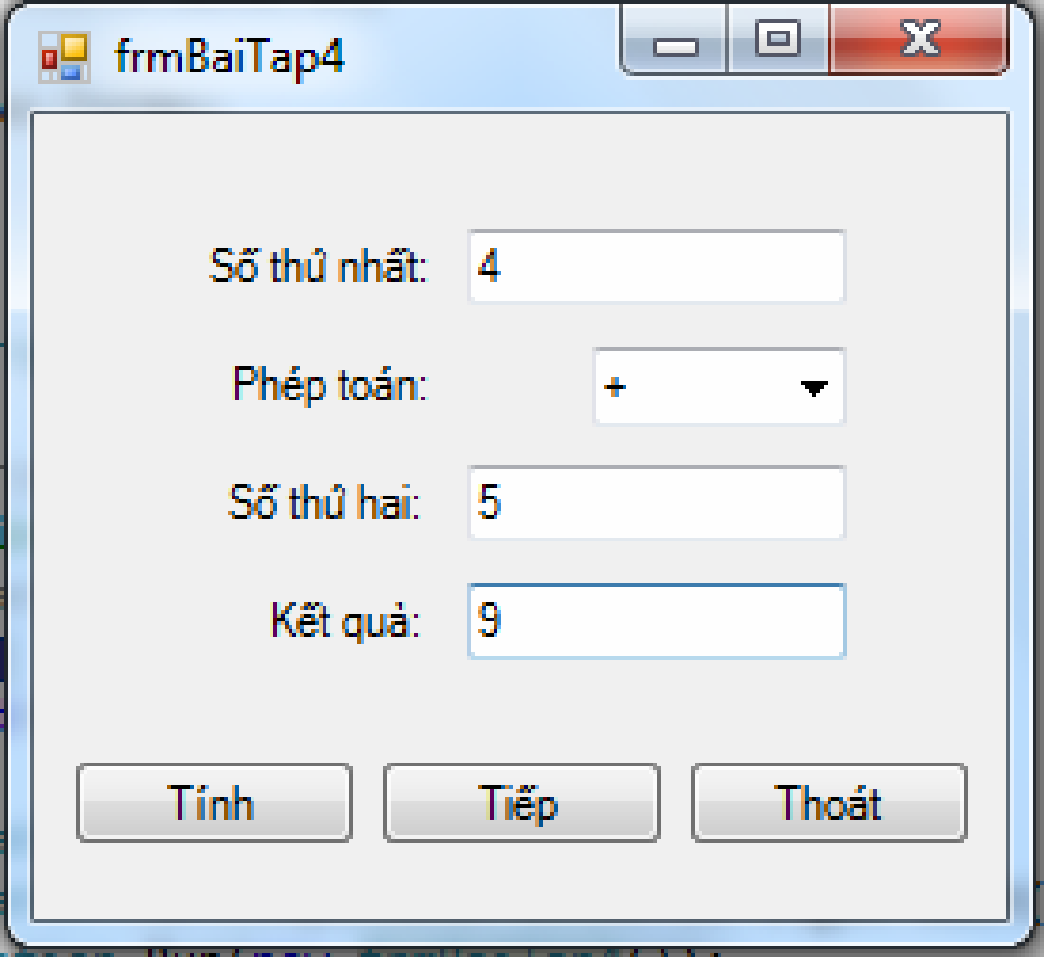
***Yêu cầu:***

* Khi chạy chương trình, nhập thông tin gồm Họ Lót, Tên, Ngày sinh vào các TextBox tương ứng
* Kích nút “Thêm” thì các thông tin vừa nhập sẽ thêm vào vào ListBox “Danh sách”
* Kích nút “Tháng này” thì hiển thị tên, ngày sinh của những người có sinh nhật ở tháng hiện tại ở- Listbox “Danh sách sinh nhật”.
* Kích nút “Kết thúc” thì kết thúc chương trình.

***Hướng dẫn:***

* Viết code cho các nút lệnh như sau:
* 

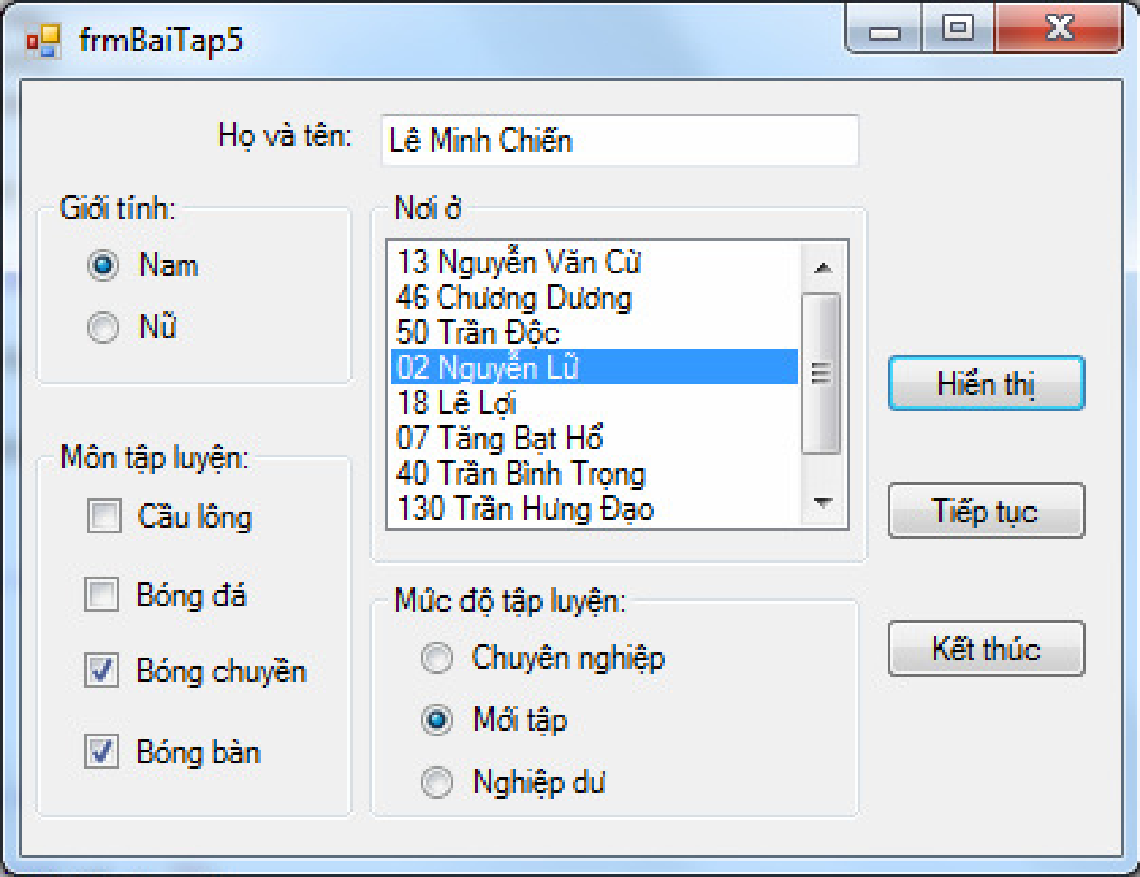
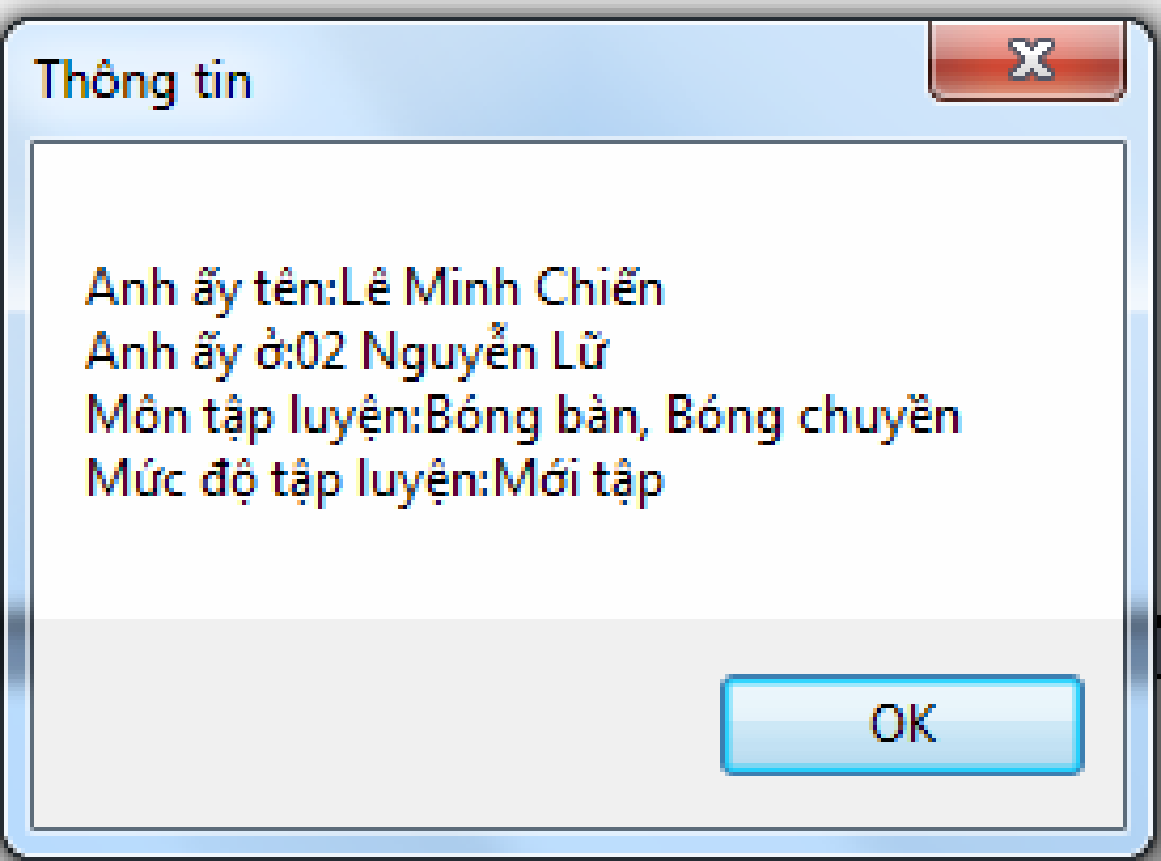
**Phần 2. Bài tập tự làm**



***Yêu cầu:***

* Khi chạy chương trình nhập vào số thứ nhất, chọn phép toán, nhập số thứ 2 và nhấn nút lệnh “Tính” thì chương trình sẽ hiển thị kết quả ở ô textbox kết quả
* Khi nhấnn nút “Tiếp” thì xóa trắng các textbox, textbox số thứ nhất nhận tiêu điểm

**Bài 4**

** **

***Yêu cầu:***

* Khi chạy chương trình nhập các thông tin một người, sau đó nhấn nút lệnh “Hiển thị” thì thông tin được đưa ra Messagebox
* Khi nhấn nút “tiếp tục” sẽ xóa các textbox

**HƯỚNG DẪN THỰC HÀNH CHƯƠNG 4 (tt)**

**Phần 1. Thực hành theo hướng dẫn**

1. **Tổng hợp các control**

#### Tóm tắt

Xây dựng chương trình điền thông tin cá nhân như minh họa



#### Kỹ thuật được trình bày

* Giới thiệu một ứng dụng WinForms cơ bản
* Cách thức lưu file với nội dung tiếng Việt
* Các thành phần điều khiển cơ bản: Button, Label, TextBox, PictureBox, Timer, …
* Nạp một ảnh từ file

#### Trình tự thực hiện

1. Tạo mới một project loại Windows Application, đặt tên là usingControls
2. Theo mặc định, một lớp Form1 được sinh ra. Chỉnh sửa các thuộc tính của Form1 với các giá trị như bảng dưới:

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ghi chú** |
| Name | FormMain |  |
| Text | Hello WinForms | Tiêu để của cửa sổ |
| FormBorderStyle | FixedSingle | Kích thước của cửa sỗ sẽ không được thay đổi khi chạy chương trình |
| MaximizeBox | False | Vô hiệu hóa nút Maximize của cửa sổ |

Chú ý: những thuộc tính có thay đổi giá trị so với mặc định sẽ được hiển thị trong cửa sổ Properties dưới dạng chữ in đậm

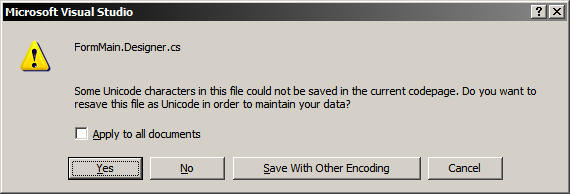
1. Thiết kế giao diện của form như minh họa. Với mỗi điều khiển đưa vào form, nếu dự định truy xuất nó trong phần mã nguồn khi lập trình thì hãy đặt tên nó thay vì để như tên mặc định.



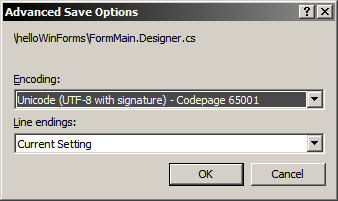
Chỉnh sửa thuộc tính của một số đối tượng như sau:

|  |  |  |
| --- | --- | --- |
| **Điều khiển** | **Thuộc tính** | **Giá trị** |
| **dtpDOB** | Format | Custom |
|  | CustomFormat | dd/MM/yyyy |
| **txtOther** | Enable | False |
| **lblInfo** | Font | Chọn font thích hợp, in đậm |
| **picImage** | SizeMode | StretchImage |
| **lblName** | BackColor | Transparent (Web) |
| **tmrScroll** | Interval | 120 |

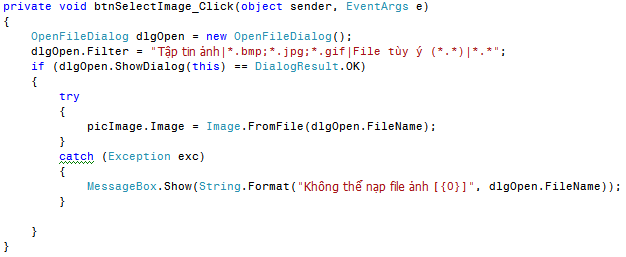
1. Nhấn Ctrl + S để lưu nội dung project. Do chúng ta có sử dụng ký tự tiếng Việt trong Form nên Visual Studio có hiển thị hộp thoại để yêu cầu chỉ định bảng mã lưu ký tự:



Nhấn nút “Save With Other Encoding” để chọn bảng mã thích hợp – sau đó bạn có thể chọn cách lưu theo UTF8 như hình dưới (cũng có thể chọn tùy chọn Unicode – Codepage 1200):

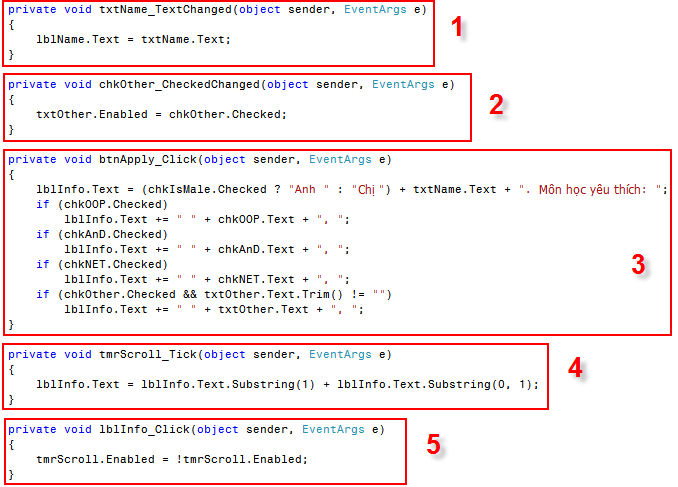


1. Cài đặt phần mã lệnh cho sự kiện Click của nút bấm btnSelectImage như sau:



Khi nhấn vào nút này, một hộp thoại sẽ hiện ra cho phép chọn ảnh. Chỉ các tập tin có phần mở rộng là BMP, JPG, GIF mới được hiển thị để lựa chọn. Điều này được thiết lập thông qua thuộc tính Filter của đối tượng dlgOpen (thuộc lớp OpenFileDialog).

1. Khi người dùng gõ tên vào txtName thì nội dung của lblName cũng thay đổi theo. Muốn vậy, ta cài đặt mã lệnh cho sự kiện TextChanged của txtName như (1) – xem minh họa code ở dưới
2. Đối tượng txtOther chỉ được sử dụng (Enabled) khi mà chkOther được check vào, do đó ta cũng cài đặt mã lệnh cho sự kiện CheckChanged của chkOther như (2)
3. Khi nhấn nút “Cập nhật” thì nội dung của lblInfo được cập nhật theo như phần mã lệnh cài đặt cho sự kiện Click của btnApply (3)
4. Người sử dụng có thể bật tắt chế độ cuộn nội dung dòng chữ lblInfo bằng cách nhấn chuột vào nó. Cài đặt mã lệnh cho sự kiện Click của lblInfo như (5)
5. Để cuộn nội dung dòng chữ, cài đặt mã lệnh cho sự kiện Tick của tmrScroll như (4)



1. **textFormat**

#### Tóm tắt

Xây dựng chương trình thể hiện định dạng cho đoạn văn bản tĩnh (Label)

#### Kỹ thuật

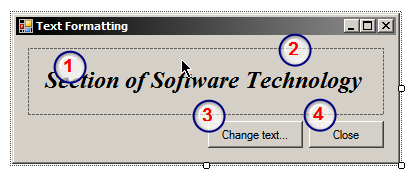
* Cách sử dụng Font, FontStyle trong ứng dụng Windows Form
* Truy xuất các thành phần dữ liệu giữa các Form
* Sử dụng cửa sổ dạng Dialog trong chương trình

#### Trình tự thực hiện

1. Tạo mới một ứng dụng loại Windows Applications, đặt tên là textFormat
2. Theo mặc định, một lớp Form1 được sinh ra. Chỉnh sửa các thuộc tính của Form1 với các thuộc tính giá trị như hình dưới:

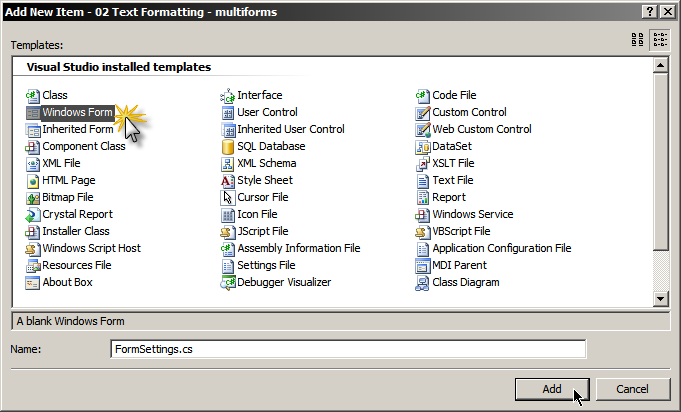
|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ghi chú** |
| Name | FormMain |  |
| Text | Text formatting | Tiêu để của cửa sổ |
| FormBorderStyle | FixedSingle | Kích thước của cửa sỗ sẽ không được thay đổi khi chạy chương trình |
| MaximizeBox | False | Vô hiệu hóa nút Maximize của cửa sổ |

1. Thiết kế giao diện cho FormMain như hình dưới đây



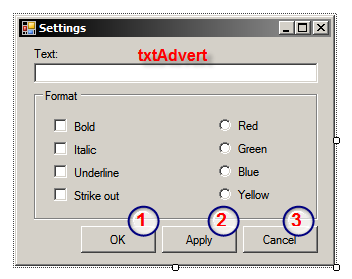
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Giá trị** | **Ghi chú** |
| 1 | **Name** | lblAdvert | Label |
| 2 | **Name** | pnlAdvert | Panel |
| 3 | **Name**  **Text** | btnChangeText  ChangeText… | Button |
| 4 | **Name**  **Text** | btnClose  Close | Button |

1. Bổ sung vào Project thêm một WindowForm bằng cách chọn menu lệnh Project → Add Windows Forms…



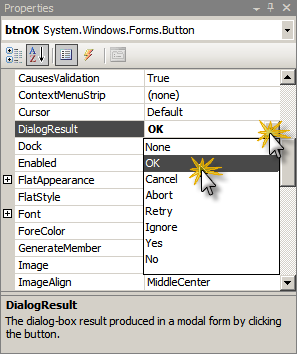
Chọn tên file của Windows Form cần tạo là FormSettings.cs như hình trên.

1. Thiết kế giao diện cho FormSettings vừa được tạo ra như hình dưới.



Trình tự đặt các điều khiển vào form có thể thực hiện như sau:

* Điều khiển TextBox đặt tên là txtAdvert.
* Điều khiển Frame với thuộc tính Text là “Format”
* 4 CheckBox có tên là chkB, chkI, chkU, chkS với thuộc tính Text được thiết lập tương ứng là “Bold”, “Italic”, “Underline” và “Strike out”
* 4 RadioButton có tên là rbRed, rbGreen, rbBlue, rbYellow với thuộc tính Text được thiết lập tương ứng với 4 tên màu “Red”, “Green”, “Blue” và “Yellow”
* 3 nút bấm btnOK, btnApply, btnCancel. Sau đó thiết lập thuộc tính DialogResult của các nút bấm btnOK, btnCancel lần lượt là OK và Cancel



1. Tiến hành cài đặt mã lệnh cho nút bấm btnChangeText ở FormMain như sau:

private void btnChangeText\_Click(object sender, EventArgs e)

{

FormSettings frm = new FormSettings();

if (frm.ShowDialog() == DialogResult.OK)

{

lblAdvert.Text = “You’ve clicked at OK button!”;

}

}

1. Thực hiện chạy chương trình và quan sát kết quả. Với việc thiết lập thuộc tính DialogResult cho 2 nút bấm btnOK và btnCancel như bước 5 ở trên, ta không cần cài đặt mã lệnh cho 2 nút này mà vẫn có hiệu ứng frm (một thể hiện của FormSettings) bị đóng khi một trong 2 nút này được nhấn. Hơn nữa, chúng ta sẽ biết được người sử dụng đã nhấn vào nút nào bằng cách kiểm tra kết quả của hàm frm.ShowDialog() như trên.

Trong Bước tiếp theo, chúng ta sẽ làm cho hai form này tương tác dữ liệu được với nhau trong thời gian thực thi.

1. Trong FormMain, tạo ra một thuộc tính AdvertText kiểu string như sau:

public string AdvertText

{

get

{

return lblAdvert.Text;

}

set

{

lblAdvert.Text = value;

}

}

1. Trong FormSettings, chúng ta thêm khai báo biến thành phần và sửa lại phương thức khởi dựng của lớp:

public partial class FormSettings: Form

{

FormMain formMain;

public FormSettings(FormMain frmMain)

{

InitializeComponent();

this.formMain = frmMain;

}

private void FormSettings\_Load(object sender, EventArgs e)

{

this.txtAdvert.Text = formMain.AdvertText;

}

. . . . .

. . . . .

}

1. Cài đặt phương thức xử lý sự kiện Click của nút bấm btnApply như sau

public void btnApply\_Click(object sender, EventArgs e)

{

formMain.AdvertText = this.txtAdvert.Text;

}

Để ý rằng, bạn phải chỉnh sửa modifier của phương thức bntApply\_Click thành public thay vì private như FormDesigner sinh ra theo mặc định. Điều này cho phép phương thức này có thể được triệu gọi từ ngoài lớp FormSettings như ở bước 11 dưới đây.

1. Chỉnh sửa lại phương thức xử lý sự kiện Click của nút bấm btnChangeText ở FormMain như sau:

private void btnChangeText\_Click(object sender, EventArgs e)

{

FormSettings frm = new FormSettings(this);

if (frm.ShowDialog() == DialogResult.OK)

{

frm.btnApply\_Click(null, null);

}

}

1. Chạy thử chương trình và quan sát kết quả. Tóm lại, chúng ta đã thực hiện những bước sau đây để có thể thay đổi dữ liệu từ 2 form:
   1. Tạo property (AdvertText) có modifier kiểu public cho phần dữ liệu muốn truy xuất (lblAdvert.Text)
   2. Tìm cách truyền tham chiếu của form chứa property nói trên (FormMain) đến form muốn truy xuất (FormSettings)
2. Tiếp theo, chúng ta sẽ tạo ra property tên AdvertForeColor có kiểu Color trong lớp FormMain để thực hiện thay đổi màu sắc của lblAdvert:

public Color AdvertForeColor

{

get

{ return lblAdvert.ForeColor; }

set

{

lblAdvert.ForeColor = value;

}

}

1. Cập nhật lại phương thức xử lý sự kiện Click của nút bấm btnApply và sự kiện Load trong lớp FormSettings:

private void FormSettings\_Load(object sender, EventArgs e)

{

this.txtAdvert.Text = formMain.AdvertText;

rbRed.Checked = formMain.AdvertForeColor == Color.Red;

rbGreen.Checked = formMain.AdvertForeColor == Color.Green;

rbBlue.Checked = formMain.AdvertForeColor == Color.Blue;

rbYellow.Checked = formMain.AdvertForeColor == Color.Yellow;

}

public void btnApply\_Click(object sender, EventArgs e)

{

formMain.AdvertText = this.txtAdvert.Text;

if (rbRed.Checked)

formMain.AdvertForeColor = Color.Red;

else if (rbGreen.Checked)

formMain.AdvertForeColor = Color.Green;

else if (rbBlue.Checked)

formMain.AdvertForeColor = Color.Blue;

else

formMain.AdvertForeColor = Color.Yellow;

}

1. Như vậy, trong các bước trên, chúng ta đã tạo ra một property kiểu Color và đồng bộ hóa nó với 4 điều khiển RadioButton. Những bước tiếp theo chúng ta tạo ra thêm một property khác để thay đổi định dạng font chữ cho lblAdvert. Trước hết, chúng ta cần biết
   1. Các thuộc tính như Bold, Italic, .. của lblAdvert.Font là chỉ đọc. Thế nên không thể thực hiện phép gán để thay đổi lblAdvert.Font.Bold được
   2. Việc thay đổi tính chất Bold, Italic, … của một đối tượng Font được thực hiện bằng cách tạo mới đối tượng Font (tham khảo thêm MSDN để biết 13 hàm nạp chồng để khởi tạo một đối tượng Font)
   3. Một trong các hàm nạp chồng khá đơn giản mà chúng ta có thể sử dụng để tạo một đối tượng Font có cú pháp là:



Trong đó, family sẽ là tên font, emSize là kích cỡ font, style là kiểu font. Giá trị của style sẽ là sự tổng hợp theo phép toán or của các giá trị FontStyle.Bold, FontStyle.Italic, FontStyle.Underline và FontStyle.Strikeout.

1. Tạo property AdvertFontFormat trong lớp FormMain như sau:

public bool[] AdvertFontFormat

{

get

{

Font f = lblAdvert.Font;

return (new bool[] {f.Bold, f.Italic, f.Underline, f.Strikeout});

}

set

{

if (value.Length == 4)

{

FontStyle fs = FontStyle.Regular;

if (value[0])

fs = fs | FontStyle.Bold;

if (value[1])

fs = fs | FontStyle.Italic;

if (value[2])

fs = fs | FontStyle.Underline;

if (value[3])

fs = fs | FontStyle.Strikeout;

lblAdvert.Font = new Font(lblAdvert.Font.Name,

lblAdvert.Font.Size, fs);

}

}

}

Property AdvertFontFormat có kiểu dữ liệu là một mảng bool gồm 4 phần tử mà thứ tự của chúng sẽ tương ứng biểu diễn tính chất Bold, Italic, Underline và Strikeout của một FontStyle.

1. Tiếp đến, chúng ta sẽ cập nhật các phương thức cần thiết trong lớp FormSettings để sử dụng thuộc tính AdvertFontFormat vừa tạo như sau:

private void FormSettings\_Load(object sender, EventArgs e)

{

this.txtAdvert.Text = formMain.AdvertText;

rbRed.Checked = formMain.AdvertForeColor == Color.Red;

rbGreen.Checked = formMain.AdvertForeColor == Color.Green;

rbBlue.Checked = formMain.AdvertForeColor == Color.Blue;

rbYellow.Checked = formMain.AdvertForeColor == Color.Yellow;

this.chkB.Checked = formMain.AdvertFontFormat[0];

this.chkI.Checked = formMain.AdvertFontFormat[1];

this.chkU.Checked = formMain.AdvertFontFormat[2];

this.chkS.Checked = formMain.AdvertFontFormat[3];

}

public void btnApply\_Click(object sender, EventArgs e)

{

formMain.AdvertText = this.txtAdvert.Text;

if (rbRed.Checked)

formMain.AdvertForeColor = Color.Red;

else if (rbGreen.Checked)

formMain.AdvertForeColor = Color.Green;

else if (rbBlue.Checked)

formMain.AdvertForeColor = Color.Blue;

else

formMain.AdvertForeColor = Color.Yellow;

formMain.AdvertFontFormat = new bool[] {

chkB.Checked, chkI.Checked, chkU.Checked, chkS.Checked };

}

1. Chạy chương trình để xem kết quả.

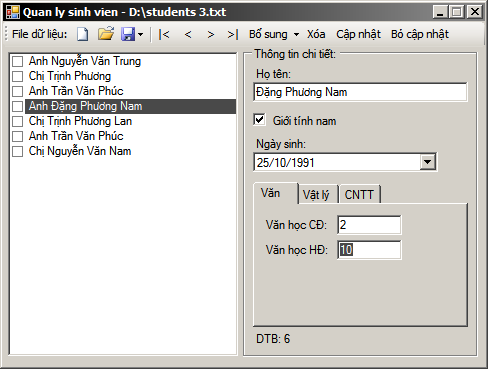
#### Mở rộng

* Qua quá trình cài đặt, có thể thấy, Form Designer của Visual Studio .NET không hỗ trợ việc tạo mảng điều khiển giống như ở Visual Basic. Tuy nhiên, chúng ta có thể mô phỏng mảng điều khiển bằng cách tạo ra các Property hoặc là Indexer để ánh xạ đến các điều khiển.

1. **Quản lý sinh viên - WinForms version**

#### Tóm tắt

Thực hiện lại chương trình quản lý sinh viên như bài thực hành 1.7 nhưng giao tiếp với người sử dụng thông qua giao diện Windows.



Chương trình có thể tạo mới, mở và lưu tập tin chứa dữ liệu về các sinh viên có định dạng như sau:

Nguyễn Văn Trung|True|1981/10/25 00:00:00|9|10|10

Trịnh Phương|False|1984/11/20 00:00:00|4|9

Trần Văn Phúc|True|1999/10/01 00:00:00|6|6.5|6.5|8

Đặng Phương Nam|True|1991/10/25 00:00:00|2|10

Trịnh Phương Lan|False|1994/11/20 00:00:00|2|9

Theo định dạng lưu này, mỗi dòng thông tin sẽ chứa các thông tin Họ tên, Giới tính, Ngày sinh, sau đó là các cột điểm. Nếu có 3 (tương ứng 2, 4) cột điểm thì sinh viên đó thuộc chuyên ngành CNTT (tương ứng Văn, Vật lý). (Xem lại Bài thực hành 1.7 để biết thêm về mô tả thông tin sinh viên).

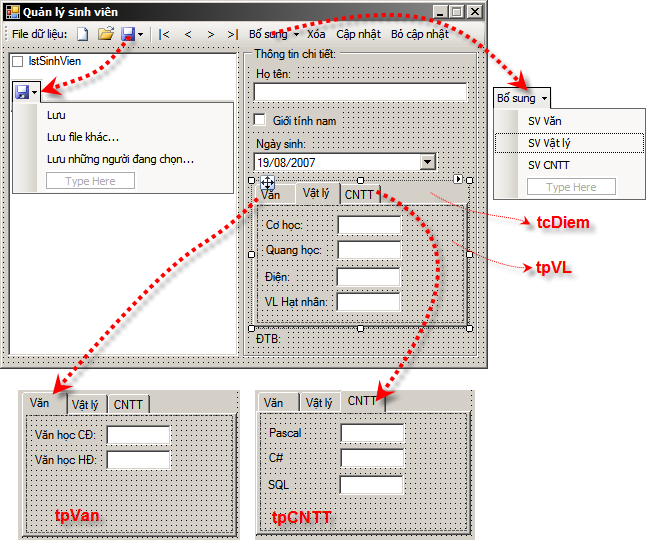
Khi người sử dụng chọn một sinh viên trong danh sách (danh sách này chỉ hiển thị tên sinh viên) thì các thông tin chi tiết về sinh viên này sẽ được hiển thị và cho phép chỉnh sửa tương ứng (xem hình vẽ). Chú ý rằng, tùy theo chuyên ngành của sinh viên, tên các môn học sẽ được hiển thị một cách hợp lý.

#### Kỹ thuật

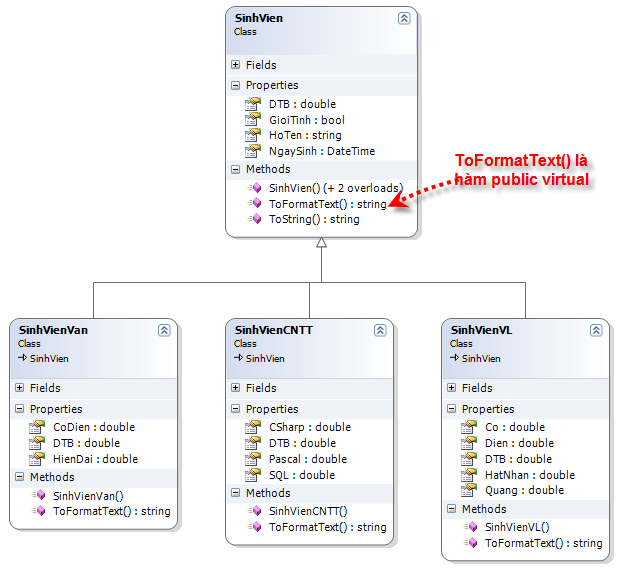
* Sử dụng kỹ thuật hướng đối tượng trên ứng dụng GUI
* Truy xuất file có định dạng quy ước sẵn

#### Trình tự thực hiện

1. Tạo form có giao diện như minh họa. Đặt tên các thành phần cho hợp lý



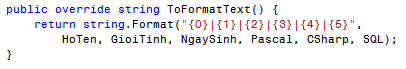
1. Thiết kế các lớp đối tượng sinh viên như sơ đồ lớp được mô tả ở dưới. Các phương thức, thuộc tính của các lớp này đã được xây dựng trong các bài thực hành trước, ở đây ta chỉ thảo luận thêm về hàm ToFormatText() ở các lớp:



Trong lớp SinhVien, đặc tả hàm này như là một hàm ảo và không cho sử dụng trực tiếp từ thể hiện thuộc lớp SinhVien như sau:

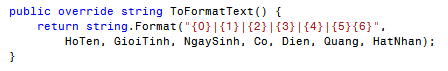


Đối với lớp SinhVienCNTT, hàm này được override lại thành:

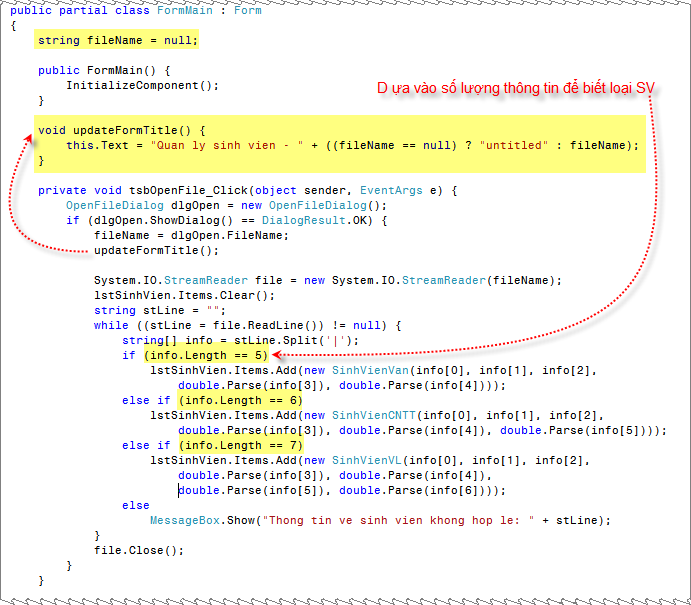


Tương tự, đối với lớp SinhVienVan và SinhVienVL, hàm này lần lượt được cài đặt như sau:



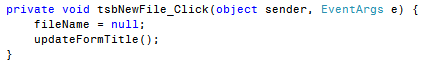


1. Xử lý sự kiện mở file dữ liệu, nạp vào listBox:
   * Bổ sung biến thành phần fileName để chứa tên file dữ liệu
   * Bổ sung thêm phương thức void updateFormTitle()
   * Thêm sự kiện Click cho nút Open như sau:



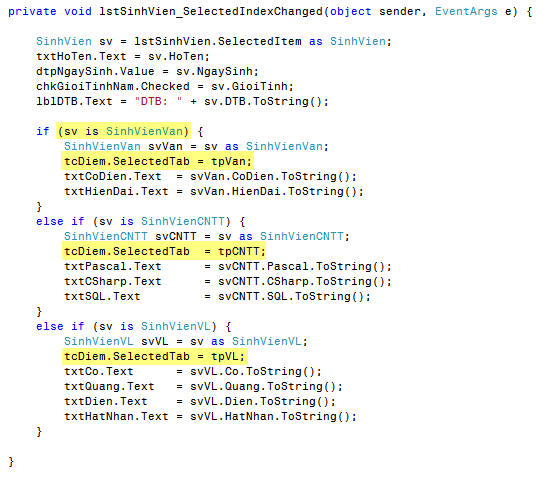
Chú ý rằng, các lớp SinhVienVan, SinhVienCNTT, SinhVienVL đều phải override hàm ToString() để được hiển thị ngắn gọn trong hộp danh sách.

1. Xử lý sự kiện tạo mới file dữ liệu
   1. Thêm phương thức xử lý sự kiện Click của nút New

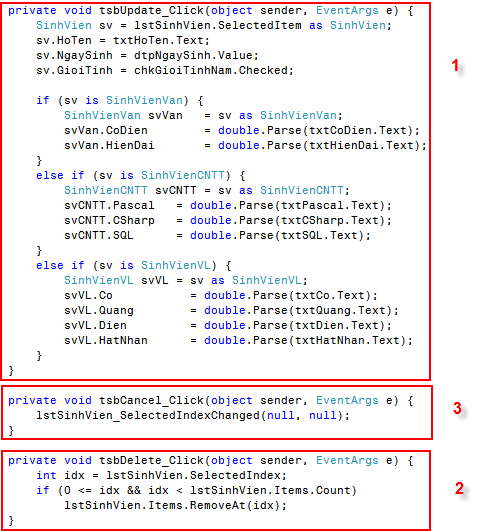


* 1. Gán phương thức xử lý sự kiện này cho sự kiện Load của Form

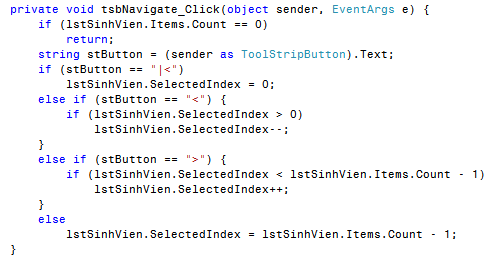
1. Khi người sử dụng chọn 1 sinh viên trong lstSinhVien, thông tin chi tiết về sinh viên này sẽ được hiển thị. Viết phương thức xử lý sự kiện SelectedIndexChanged của lstSinhVien như sau:



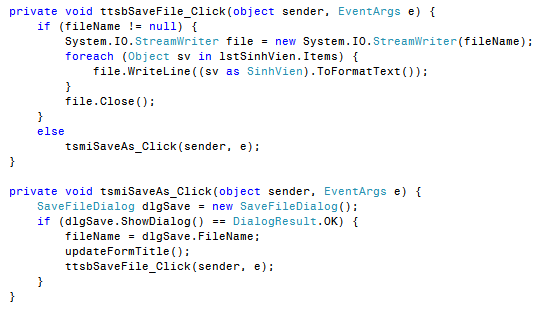
1. Viết phương thức xử lý sự kiện Click của nút “Cập nhật”, “Xóa” và “Bỏ cập nhật” như (1), (2), (3):



1. Cài đặt phương thức xử lý sự kiện Click cho 4 nút “|<”, “<”, “>”, “>|” như sau:



1. Tiếp đến, cài đặt phương thức xử lý sự kiện Click của nút bấm Save và Save as…



#### Mở rộng

* Thêm phần xử lý lỗi cho chương trình
* Bổ sung chức năng Tạo mới Sinh viên

**Phần 2. Bài tập tự làm**

1. **myCalculator**

#### Tóm tắt

Xây dựng chương trình mô phỏng một máy tính điện tử đơn giản.

#### Kỹ thuật

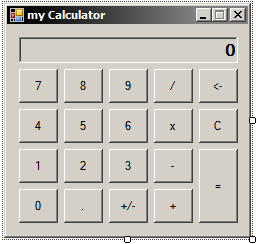
* Xây dựng ứng dụng GUI với WinForms
* Cách thức cài đặt, gắn kết, xử lý events với một thành phần giao diện
* Tạo hiệu ứng transparent với Form

#### Trình tự thực hiện

1. Tạo mới một project loại Windows Application, đặt tên là myCalculator
2. Theo mặc định, một lớp Form1 được sinh ra. Chỉnh sửa các thuộc tính của Form1 với các giá trị như bên dưới:

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ghi chú** |
| Name | FormMain |  |
| Text | my Calculator | Tiêu để của cửa sổ |
| FormBorderStyle | FixedSingle | Kích thước của cửa sỗ sẽ không được thay đổi khi chạy chương trình |
| MaximizeBox | False | Vô hiệu hóa nút Maximize của cửa sổ |

1. Tạo giao diện cho chương trình như hình minh họa



Sau khi đã tạo ra được giao diện với các điều khiển có vị trí, kích thước hợp lý, bạn nên *cố định chúng.* Thực hiện điều này bằng cách nhắp phải chuột lên bề mặt của FormMain, chọn Lock Controls. Điều chỉnh các thuộc tính của đối tượng “màn hình hiển thị” như sau:

|  |  |  |
| --- | --- | --- |
| **Đối tượng** | **Thuộc tính** | **Giá trị** |
| **“Màn hình hiển thị”** | **Name** | txtScreen |
|  | **Text** | 0 |
|  | **Font** | Chọn font thích hợp |
|  | **ReadOnly** | True |
|  | **TextAlign** | Right |

1. Trước khi bắt tay vào viết mã lệnh, bạn phải hình dung cách thức làm việc của chương trình cũng như việc chuẩn bị các kiểu đối tượng, dữ liệu.

Giao diện của chương trình chỉ là các nút bấm và “màn hình” hiển thị kết quả. Do chỉ là máy tính đơn giản nên chương trình chỉ hỗ trợ các phép tính 2 ngôi cơ bản. Có thể phân biệt nút bấm thuộc các nhóm:

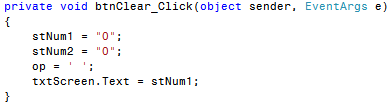
1. Nút bấm chữ số 0, 1, …, 9, ký hiệu dấu chấm thập phân, nút đảo dấu, nút xóa trái → xác định hai toán hạng tham gia vào phép tính.
2. Nút bấm phép tính +, -, x, / → xác định phép tính
3. Nút = → để kích hoạt việc thực hiện phép tính và hiển thị kết quả
4. Nút C → khởi tạo một phép toán mới

Trình tự các bước sử dụng của máy tính thông thường như sau: Người sử dụng “nhập vào” số thứ nhất, xác định phép toán, sau đó “nhập vào” số thứ hai, cuối cùng nhấn dấu =

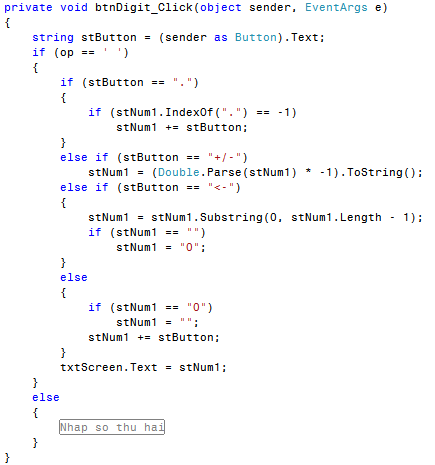
Khái niệm “nhập vào” ở đây có nghĩa là người sử dụng dùng liên tiếp các nút bấm thuộc nhóm (1). Chú ý rằng dấu chấm thập phân chỉ xuất hiện 1 lần trong 1 số hạng. Việc xác định phép toán có thể xác định nhiều lần *giữa hai lần nhập hai số hạng*. Tóm lại, khi người sử dụng nhấn một nút bấm trong nhóm (1), ta luôn biết được anh ta đang “nhập số liệu” cho số hạng nào: nếu chưa xác định phép toán thì có nghĩa anh ta đang nhập cho số thứ nhất, ngược lại thì đang nhập cho số thứ hai.

Điều khác nữa cần phải tính đến, đó là lưu trữ dữ liệu như thế nào? Trong bài toán này, việc sử dụng string để lưu trữ các số hạng là hợp lý cho việc nhập số. Phép tính sẽ được lưu trữ bằng một ký tự.

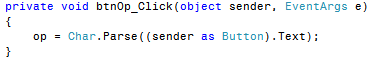
1. Ngay trong đầu phần mã nguồn partial class FormMain: Form, bổ sung khai báo các biến
   1. stNum1, stNum2 kiểu string để đại diện cho hai chuỗi số hạng.
   2. op kiểu char để đại diện cho phép toán (op là viết tắt của operator)
2. Cài đặt phần mã lệnh cho sự kiện FormLoad và sự kiện click của nút C như sau:



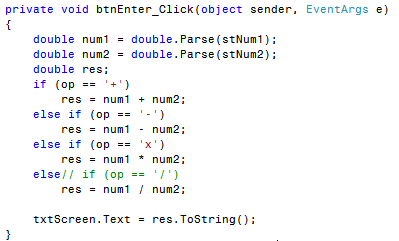
1. Các nút bấm trong nhóm (1) được cài đặt phần mã lệnh tương ứng với sự kiện click như sau:



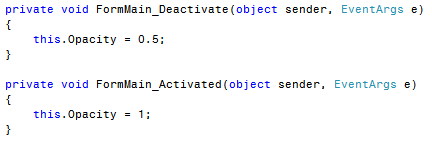
1. Phương thức xử lý sự kiện click của nút bấm thuộc nhóm (2) – xác định phép tính



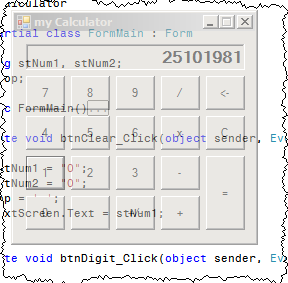
1. Cuối cùng, phương thức xử lý sự kiện click của nút bấm ‘=’ được cài đặt như sau:



1. Đến đây, các tính năng cơ bản của máy tính đã được cài đặt. Chúng ta khảo sát thêm một số hiệu ứng đặc biệt của lớp đối tượng Form:
   1. Thiết lập thuộc tính TopMost bằng True để trong thời gian thực thi, cửa sổ chương trình luôn luôn nổi.
   2. Bổ sung phương thức xử lý sự kiện Activated và Deactive cho FormMain. Hai sự kiện này được kích hoạt tương ứng khi người dùng chuyển focus từ vào ứng dụng (Activated) hay ra khỏi ứng dụng (Deactive)



* 1. Minh họa khi chạy chương trình: khi người sử dụng chuyển focus ra khỏi cửa sổ ứng dụng, cửa sổ này sẽ bị mờ đi 50%.



#### Mở rộng

* Cài đặt thêm các tính năng khác cho máy tính, chẳng hạn hỗ trợ phép tính 1 ngôi như căn bậc 2, nghịch đảo, phần trăm,… Chú ý phần xử lý lỗi (ví dụ như chia cho 0, căn bậc hai với số âm…)
* Hỗ trợ để người sử dụng có thể nhập dữ liệu bằng bàn phím.

1. **myNotePAD**

#### Tóm tắt

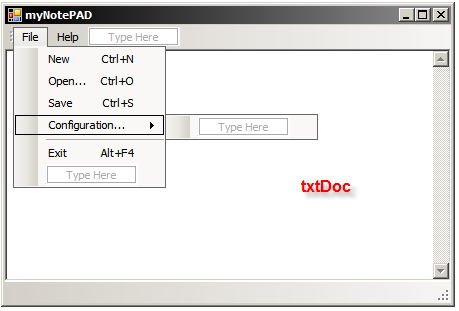
Chương trình có các chức năng cơ bản của một chương trình soạn thảo văn bản đơn giản (như Ms Windows NotePAD): Tạo mới, Soạn thảo, Lưu lên đĩa, Mở từ đĩa. Chương trình cũng cho phép người sử dụng thiết lập các cấu hình phụ như: font chữ thể hiện (loại font, kích thước, màu sắc, ...).

#### Kỹ thuật

* Sử dụng menu
* Xây dựng ứng dụng với nhiều form
* Lấy danh sách font hệ thống
* Cách liên hệ giữa các đối tượng thành phần thuộc các form, module khác nhau

#### Trình tự thực hiện

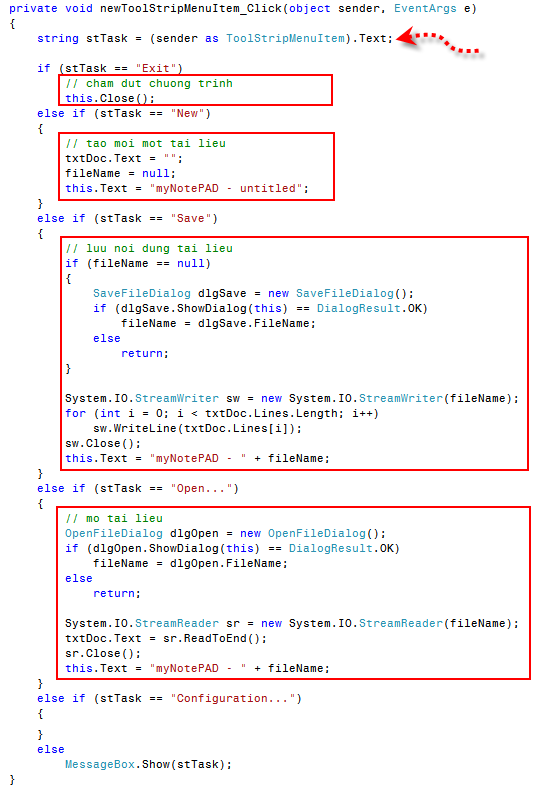
1. Tạo mới một project kiểu Windows Application, đặt tên là myNotePAD
2. Đổi thuộc tính Name của form thành FormMain
3. Bổ sung các thành phần StatusBarStrip, MenuStrip với các menu lệnh như minh họa.



1. Bổ sung thêm thành phần TextBox với tên gọi là txtDoc. Thiết lập các thuộc tính của txtDoc như sau:

|  |  |
| --- | --- |
| **Thuộc tính** | **Giá trị** |
| Multiline | True |
| Dock | Fill |

1. Khai báo một biến fileName kiểu string ở đầu phần mô tả lớp. Biến này sẽ lưu tên file đang được soạn thảo. Sau đó cài đặt phương thức xử lý sự kiện cho tất cả các ToolStripMenuItem như sau:



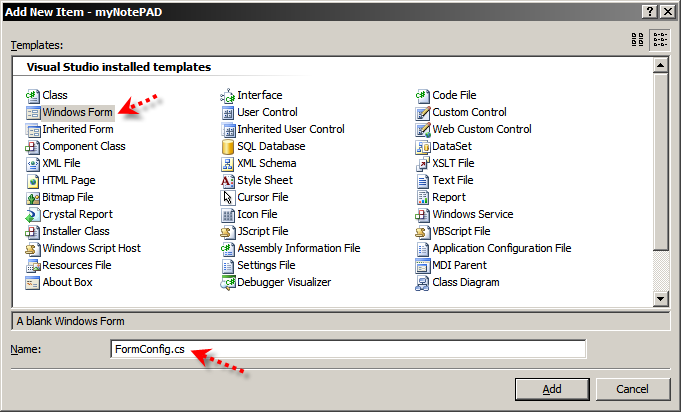
✍ Phần xử lý cho “else if (stTask == “Configuration…”)…” sẽ được cài đặt sau.

✍Chú ý rằng thuộc tính Text của các ToolStripMenuItem được sử dụng để viết các câu lệnh if ở trên. Vậy nên, bạn phải nhớ chính xác các thuộc tính Text của chúng để viết mã lệnh đúng.

✍Thực ra, bạn có thể viết riêng rẽ cho từng phương thức xử lý sự kiện Click của từng ToolStripMenuItem, tuy nhiên theo ý kiến của riêng người viết giáo trình này, cách gộp chung như ở trên giúp bạn tập trung mã lệnh hơn.

1. Bước tiếp theo chúng ta cài đặt phần mã lệnh cho phép người sử dụng thay đổi font hiển thị tài liệu:

Bổ sung một Form vào project bằng cách nhấn tổ hợp phím Ctrl + Shift + A. Một hộp thoại như sau hiện ra:



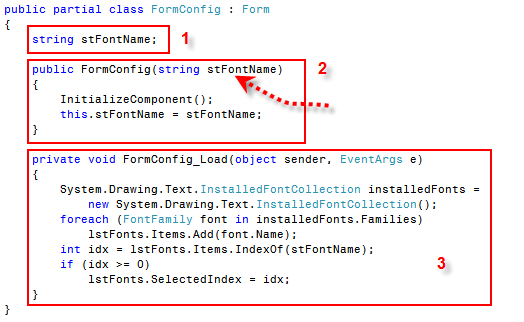
Chọn loại đối tượng cần thêm là Windows Form, tên file là “FormConfig.cs”, nhấn Add.

Sau thao tác này, Visual Studio sẽ tạo ra một file trong đó có khai báo lớp FormConfig (trùng tên với tên file mà bạn cung cấp). Nếu muốn, chúng ta có thể đổi tên lớp của form này.

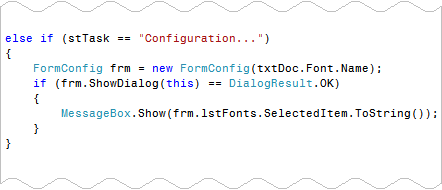
1. FormConfig sẽ làm nhiệm vụ hiển thị danh sách các fonts hiện có trong hệ thống để người sử dụng chọn. Thiết kế giao diện của FormConfig như sau:

|  |  |
| --- | --- |
|  | Dự kiến, khi form này bắt đầu nạp nó sẽ hiển thị danh sách tất cả các fonts đang được cài đặt trong hệ thống vào lstFonts. Font đang được sử dụng ở txtDoc (ở FormMain) sẽ được tô sáng trong lstFonts. Cuối cùng, khi người sử dụng nhấn nút OK, font được chọn trong lstFonts sẽ được sử dụng cho txtDoc. Như vậy, đã có sự sử dụng biến thành phần giữa hai forms: FormMain và FormConfig. |

1. Trước hết, thiết lập thuộc tính DialogResult của hai nút OK và Cancel là OK và Cancel. Có thể thiết lập thêm các thuộc tính Anchor của các đối tượng để có giao diện hợp lý.
2. Thiết lập thuộc tính Modifiers của lstFonts là Public. Bằng cách này, chúng ta có thể truy xuất đến thành phần lstFonts từ bên ngoài.
3. Bổ sung biến thành phần stFontName kiểu string vào khai báo FormConfig, xem (1).
4. Sửa phương thức khởi dựng của lớp FormConfig như (2).
5. Viết mã cho phương thức xử lý sự kiện Load của form như (3).



1. Hoàn thiện phần gọi FormConfig ở FormMain như sau:



#### Mở rộng

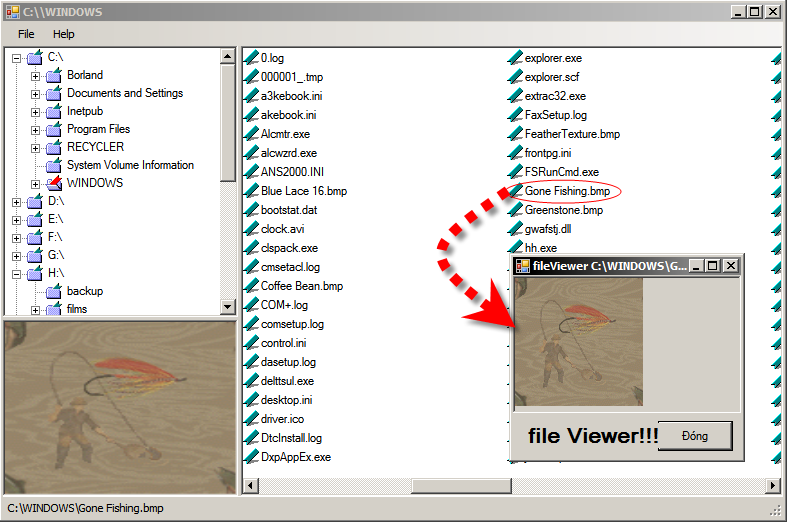
* Cài đặt thêm các chức năng thường dùng khác đối với ứng dụng soạn thảo văn bản

Sửa lại ứng dụng theo giao diện MDI

1. **myFileViewer**

#### Tóm tắt

Trong bài thực hành này, bạn sẽ được tiếp cận một số kỹ thuật nâng cao như: kế thừa form, thành phần điều khiển TreeView, ListView, ... Ứng dụng cuối cùng được tạo ra sẽ là ứng dụng quản lý tập tin kiểu ACDSee.

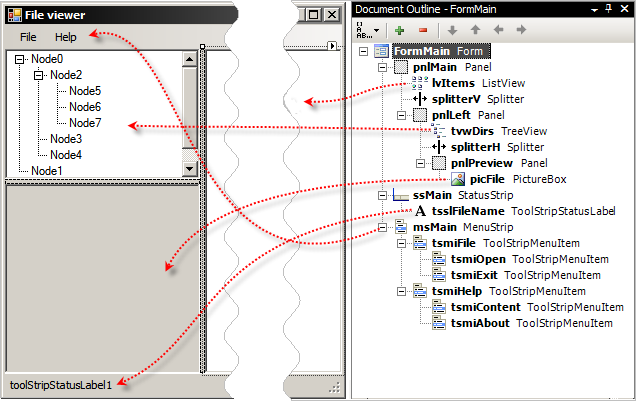


#### Kỹ thuật

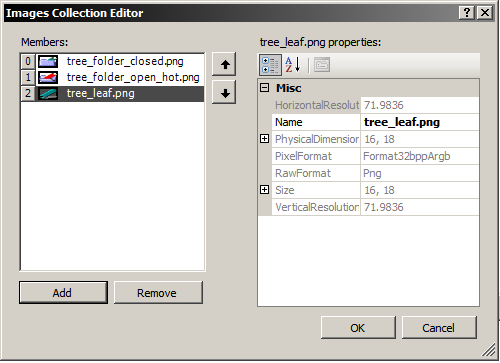
* Thừa kế form
* Làm quen với các component, các thư viện nâng cao

#### Trình tự thực hiện

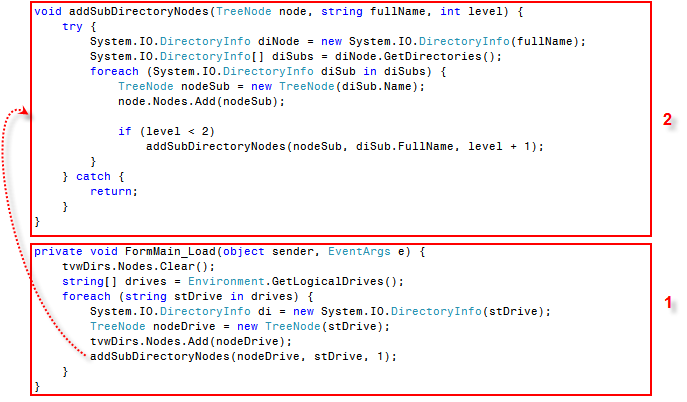
1. Tạo mới một ứng dụng kiểu Windows Application với tên là fileViewer. Đặt tên cho form chính là FormMain.
2. Thiết kế giao diện ban đầu cho FormMain như hình vẽ. Khi chạy chương trình, vùng bên tay trái sẽ hiển thị cây thư mục (đối tượng tvwDirs). Mỗi khi chọn 1 thư mục ở cây, nội dung của nó được hiển thị ở lvwItems bên phải. Ngay phía dưới tvwDirs là vùng “preview” nội dung của file được chọn trong lvwItems.



1. Thêm vào Form một đối tượng ImageList có tên là imageList1 với các ảnh như hình dưới:

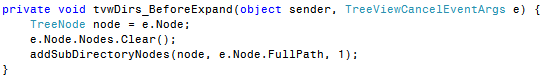


1. Gán imageList1 cho thuộc tính ImageList của tvwDirs và lvItems.
2. Viết mã lệnh cho sự kiện Load của FormMain như (1)



Trước hết, lấy danh sách các ổ đĩa của hệ thống. Ứng với mỗi đĩa như vậy, ta sẽ thêm tất cả các thư mục con cấp 1 của ổ đĩa (hãy giải thích?) bằng cách gọi phương thức (2). Để ý rằng (2) sẽ được gọi đệ quy.

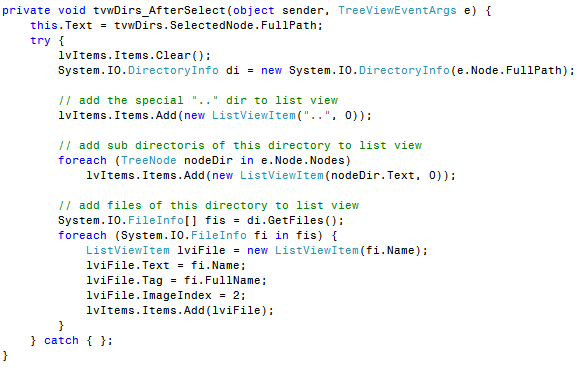
1. Khi người dùng nhấn vào dấu “+” của một node trên treeview để mở nhánh này ra, ta phải hiển thị được tất cả các thư mục con của nó. Việc này được chuẩn bị qua phương thức xử lý sự kiện BeforeExpand của treeview tvwDirs:



1. Khi người sử dụng chọn một node trên cây thì nội dung của thư mục tương ứng (bao gồm tất cả thư mục và files chứa trong nó) sẽ được hiển thị trong listview lvItems.

Để thuận tiện cho những thao tác khác sau này của ứng dụng, chúng ta quy ước, trong lvItems, phần tử đại diện cho file sẽ có ImageIndex là 2, còn phần tử đại diện cho thư mục sẽ có ImageIndex là 0.

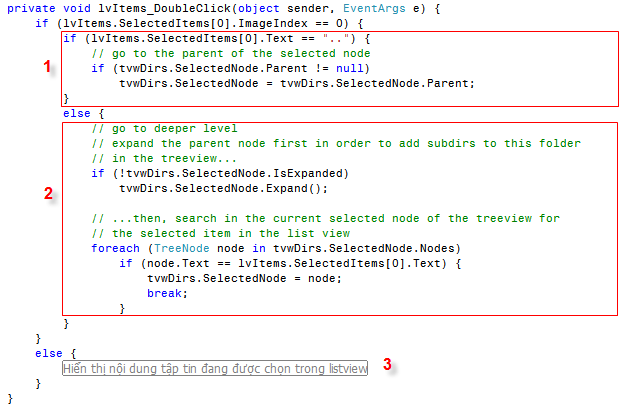
Chúng ta xử lý sự kiện AfterSelect của tvwDirs như sau:



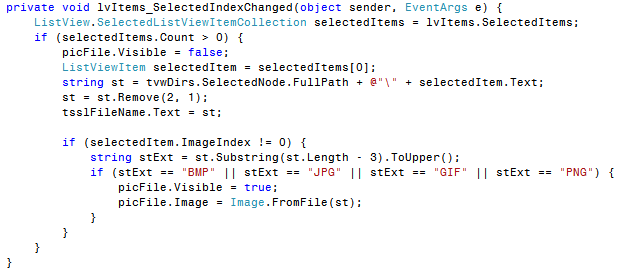
Chú ý rằng, ngoài các thư mục và tập tin, ta thêm vào một “thư mục” đặc biệt, thư mục “..” đại diện cho thư mục cha của thư mục hiện tại.

1. Tiếp đến, chúng ta viết mã cho sự kiện nhấn đúp chuột vào một phần tử trong listview. Có 3 khả năng xảy ra:
2. người sử dụng nhấn đúp vào phần tử là thư mục “..” → chuyển đến thư mục trên cấp của thư mục hiện tại
3. người sử dụng nhấn đúp vào phần tử thư mục bình thường → chọn thư mục đó làm thư mục hiện tại
4. người sử dụng nhấn đúp vào phần tử là một file → hiển thị nội dung của file này ra một cửa sổ riêng một cách thích hợp theo nội dung của nó.

Trong phần code dưới đây, ta chỉ quan tâm đến 2 khả năng đầu. Khả năng còn lại sẽ được hoàn thiện ở các bước sau.

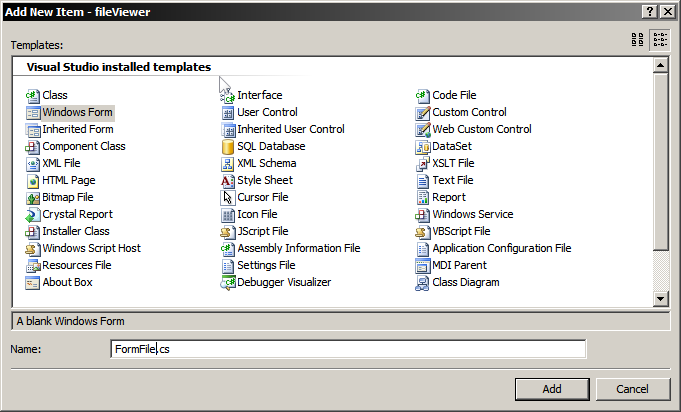


1. Tiếp theo, ta sẽ cài đặt thêm chức năng Preview nội dung của phần tử file ảnh đang được chọn trong listview lvItems. Sự kiện SelectedIndexChanged của lvItems được xử lý như sau:

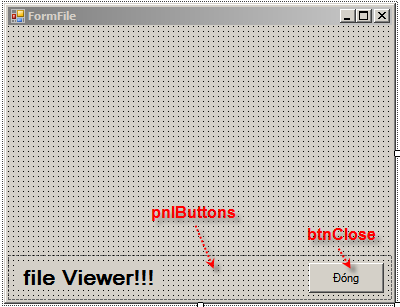


1. Phần còn lại của bài thực hành sẽ hướng dẫn bạn cài đặt phần xử lý khả năng (3) trong bước 8. Giả sử rằng, ứng dụng của chúng ta chỉ hỗ trợ hiển thị ra cửa sổ 2 loại nội dung: ảnh và văn bản. Như thế, để hiển thị mỗi loại nội dung, chúng ta cần 1 loại form khác nhau. Tuy thế, cả hai loại form này cũng có một số thuộc tính và hành vi chung. Ta xác định mối quan hệ của chúng qua sơ đồ lớp như sau:

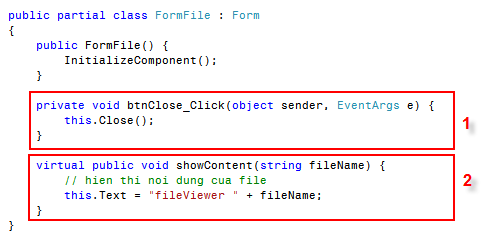
Tạo mới một Form để hiển thị nội dung file như sau:



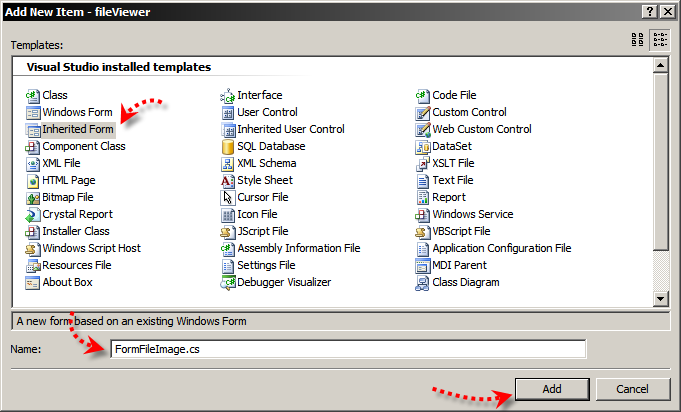
1. Thiết kế giao diện cho FormFile như sau:
   * Đặt một Panel tên pnlButtons neo phía dưới Form (Dock = Bottom).
   * Trên pnlButtons, đặt nút btnClose vào vị trí thích hợp.
   * Có thể trang trí thêm cho pnlButtons, ví dụ đặt một Label trong nó như hình minh họa.
   * Phần trống còn lại của form dự kiến sẽ dùng để hiển thị nội dung của file.



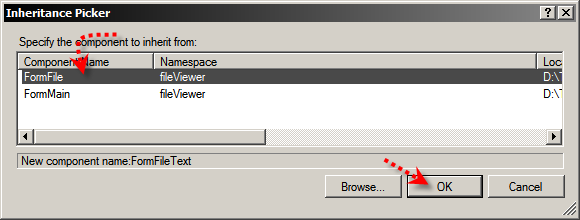
1. Cài đặt mã lệnh cho form như sau
   * Cài đặt phương thức xử lý sự kiện Click của nút btnClose (1)
   * Bổ sung thêm hàm virtual showContent() vào lớp (2). Dự kiến hàm này sẽ làm nhiệm vụ hiển thị nội dung của file: tùy theo file ảnh hay file văn bản mà có cách hiển thị khác nhau, nhưng thao tác chung nhất là hiển thị đường dẫn của file lên tiêu đề cửa sổ.



1. Lưu và Build ứng dụng. (rất quan trọng cho các bước sau!)
2. Bây giờ, ta bổ sung thêm một Windows Form vào project. Tuy nhiên khác với các lớp cửa sổ ta đã bổ sung trước đó, lần này ta sẽ bổ sung một Windows Form kế thừa từ một lớp ta đã thiết kế. Cụ thể là FormFileImage kế thừa từ FormFile.

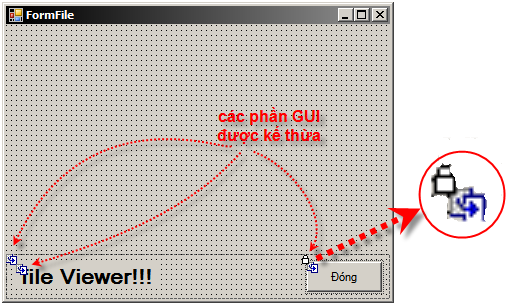


1. Visual Studio sẽ hiển thị hộp thoại hỏi lớp mà FormFileImage được kế thừa:



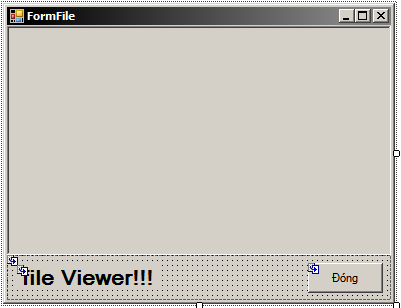
Chọn FormFile làm Form để kế thừa rồi nhấn OK

1. FormFileImage được tạo ra với giao diện như dưới đây:

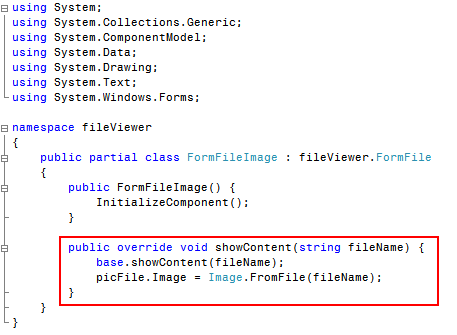


Để ý, các thành phần được kế thừa sẽ từ lớp cha có mũi tên nhỏ bên góc trái. Hơn nữa, những thành phần thuộc loại Private thì sẽ có thêm biểu tượng ổ khóa, bạn không thể thay đổi thuộc tính của nó trong lớp kế thừa. Ví dụ, nếu muốn chỉnh sửa thuộc tính Text của nút btnClose trong FormFileImage, bạn phải thiết lập btnClose ở FormFile thuộc loại Protected hoặc Public.

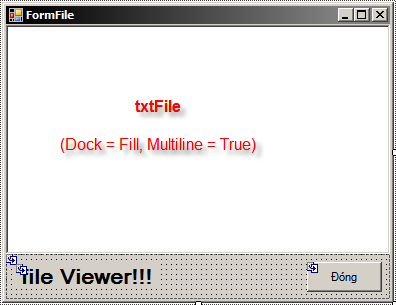
1. Bây giờ, bổ sung thêm một PictureBox có tên là pic vào FormFileImage. Thiết lập thuộc tính Dock của nó là Fill. Xem hình minh họa

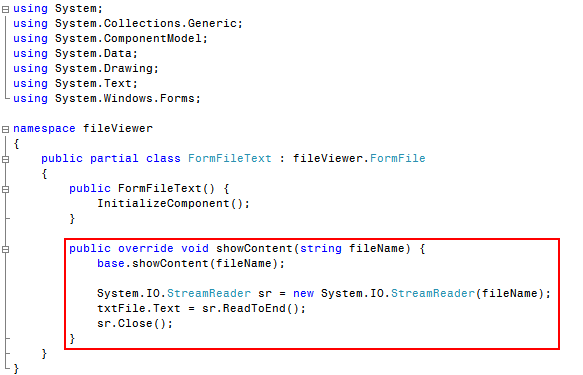


1. Trong phần mã lệnh cho FormFileImage, ta override hàm showContent như sau:

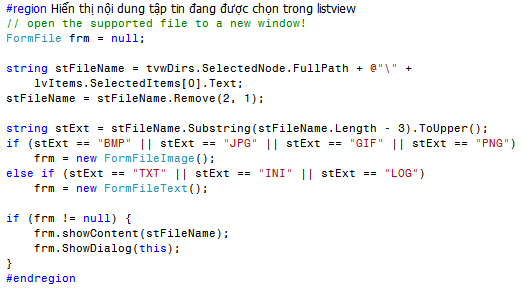


1. Tương tự như các bước tạo FormFileImage, ta tạo thêm một Windows Form có tên là FormFileText kế thừa từ FormFile có giao diện và phần cài đặt mã lệnh như sau:





1. Cuối cùng, ta hoàn chỉnh chức năng (3) đã mô tả ở Bước 8 bẳng cách sửa lại phương thức xử lý sự kiện DoubleClick của lvItems như sau:



#### Mở rộng

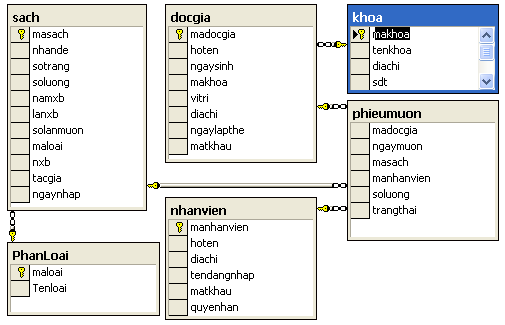
* Bổ sung thêm chức năng preview cho file văn bản
* Cài đặt chức năng cho các menu

**HƯỚNG DẪN THỰC HÀNH CHƯƠNG 5**

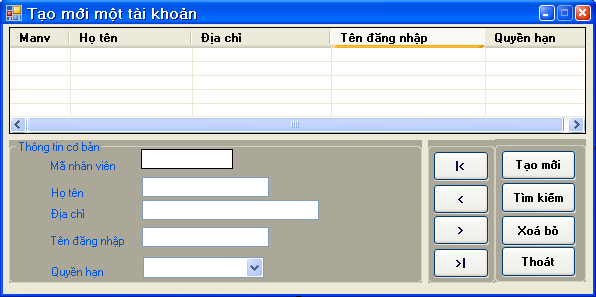
**Phần 1. Thực hành theo hướng dẫn**

1. **Sử dụng ADO.NET, Đối tượng Connection, Command và DataReader**

Giả sử ta đã có cơ sở dữ liệu **quanlythuvien** trong SQL Server có quan hệ như sau:

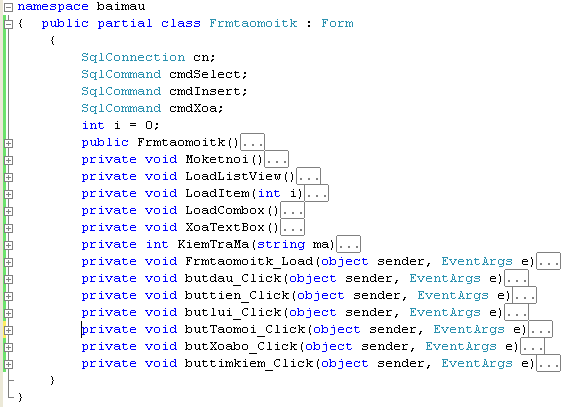


Thiết kế Form để tạo mới 1 tài khoản như sau (làm việc trên bảng nhanvien):



ListView1

Frmtaomoitk sử dụng các trường, phương thức và sự kiện sau:



#### Các điều khiển

|  |  |
| --- | --- |
| **Tên điều khiển** | **Thuộc tính** |
| Form | **Name**: Frmtaomoitk  **Text**:Tạo mới một tài khoản sử dụng chương trình |
| listView | **Name**:listView1  **Columns**: Add thêm 4 cột: Họ tên, Địa chỉ, Tên đăng nhập và Quyền hạn  View: Details  GridLines:True |
| groupBox | **Name:**groupBox1  Text: Thông tin cơ bản |
| Label | Tạo ra 5 label để hiển thị: Mã nhân viên, Họ tên, Địa chỉ, Tên đăng nhập và quyền hạn. |
| TextBox | Tạo ra 4 TextBox lần lượt với các tên: txtmanv, txthoten, txtdiachi, txttendangnhap |
| Button | Tạo 8 button lần lượt với các tên butdau, butlui, buttien, butcuoi, buttaomoi, buttimkiem, butxoabo,butthoat |

#### Các trường:

|  |  |
| --- | --- |
| **Tên trường** | **Ý nghĩa** |
| Cn | Dùng để kết nối đến cơ sở dữ liệu quanlythuvien |
| cmdSelect | sqlCommand sử dụng câu lệnh select để hiển thị và tìm kiếm |
| cmdInsert | sqlCommand sử dụng câu lệnh Insert để tạo thêm 1 tài khoản |
| cmdXoa | sqlCommand sử dụng câu lệnh Delete để xóa 1 tài khoản |
| I | Tài khoản thứ i |

#### Các phương thức

+ Hàm dựng **Frmtaomoitk** để tạo giao diện

public Frmtaomoitk()

{

InitializeComponent();

}

+ Phương thức Moketnoi(): Kiểm tra đường kết nối, nếu đang mở thì đóng lại, sau đó mở lại đường kết nối

private void Moketnoi()

{

if (cn.State == ConnectionState.Open)

cn.Close();

cn.Open();

}

+ Phương thức **LoadListView**: Lấy dữ liệu của bảng nhanvien nạp dữ liệu lênlistView1. Phương thức này được gọi khi thay đổi dữ liệu trong bảng nhận viên như nhâp thêm hoặc xóa đi 1 nhân viên. Sử dụng 2 đối tượng SqlCommand, SqlDataReader

private void LoadListView()

{

Moketnoi();

cmdSelect = new SqlCommand("select \* from nhanvien", cn);

SqlDataReader r = cmdSelect.ExecuteReader();

listView1.Items.Clear(); // Xóa tất cả dữ liệu trong listView1

while (r.Read())

{

string[] st = new string[5];

st[0] = r[0].ToString();

st[1] = r[1].ToString();

st[2] = r[2].ToString();// Không hiển thị mật khẩu, nên không có r[3]

st[3] = r[4].ToString();

st[4] = r[5].ToString();

ListViewItem lv = new ListViewItem(st);

listView1.Items.Add(lv);

}

cmdSelect.Dispose();

}

+ Phương thức **LoadItem**: Lấy dữ liệu từ dòng thứ i của listView1 đưa vào txtmanv, txthoten, txtdiachi, txttendangnhap và comboBox1. Phương thức này được gọi khi di chuyển qua lại thông tin của các nhân viên.

private void LoadItem(int i)

{

txtmanv.Text = listView1.Items[i].Text;

txthoten.Text = listView1.Items[i].SubItems[1].Text;

txtdiachi.Text = listView1.Items[i].SubItems[2].Text;

txttendangnhap.Text = listView1.Items[i].SubItems[3].Text;

comboBox1.Text = listView1.Items[i].SubItems[4].Text;

}

+ Phương thức **LoadCombox**: Đưa dữ liệu vào cho comboBox1. Giả sử chỉ có 3 quyền hạn: admin, sinhvien và Thuthu. Phương thức này được gọi khi vừa nạp Form lên

private void LoadCombox()

{

comboBox1.Items.Add("Admin");

comboBox1.Items.Add("Sinhvien");

comboBox1.Items.Add("ThuThu");

comboBox1.Text = "Admin";

}

+ Phương thức **XoaTextBox**: Xóa hết dữ liệu trong các textBox, phương thức này được goi khi nhập thêm 1 tài khoản.

private void XoaTextBox()

{

txtmanv.Clear();

txthoten.Clear();

txtdiachi.Clear();

txttendangnhap.Clear();

txtmanv.Focus();

}

+ Phương thức**KiemTraMa:** Kiểm tra xem có mã nhân viên nào bằng với ma hay không. Phương thức này được gọi khi nhập thêm 1 tài khoản

private int KiemTraMa(string ma)

{

Moketnoi();

cmdSelect = new SqlCommand("select count(\*) from nhanvien where manhanvien='"+ma.Trim()+"'");

cmdSelect.Connection = cn;

return (int)cmdSelect.ExecuteScalar();

}

+ Sự kiện **Frmtaomoitk\_Load:** Tạo và mở ra đường kết nối đến cơ sở dữ liệu quanlythuvien, tên máy chủ nhha, sử dụng cơ chế xác thực kiểu Windows, nạo dữ liệu vào cho các điều khiển.

private void Frmtaomoitk\_Load(object sender, EventArgs e)

{

try

{

cn = new SqlConnection("Data Source=nhha;Initial Catalog=quanlythuvien; Trusted\_Connection=yes");

cn.Open();

}

catch (Exception loi)

{

MessageBox.Show("Không thể kết nối được");

}

LoadListView(); //Nạp dữ liệu vào cho listView1

i = 0;

LoadItem(i);// Nạp dữ liệu vào cho các textBox và comboBox1

LoadCombox();

}

+ Sự kiện **butdau\_Click:** Nạp dữ liệu của dòng đầu tiên từ listView1 vào cho các textBox và comboBox

private void butdau\_Click(object sender, EventArgs e)

{ i = 0;

LoadItem(i);

}

+ Sự kiện **buttien\_Click:** Nạp dữ liệu của dòng tiếp theo từ listView1 vào cho các textBox và comboBox

private void buttien\_Click(object sender, EventArgs e)

{

i++;

if (i == listView1.Items.Count) i = listView1.Items.Count - 1;

LoadItem(i);

}

+ Sự kiện **butlui\_Click:**

private void butlui\_Click(object sender, EventArgs e)

{

i--;

if (i < 0) i = 0;

LoadItem(i);

}

+ Sự kiện **butcuoi\_Click**:

private void butcuoi\_Click(object sender, EventArgs e)

{

i = listView1.Items.Count - 1;

LoadItem(i);

}

+ Sự kiện **butTaomoi\_Click:** Được sử dụng để thêm 1 tài khoản (1 nhân viên), nút butTaomoi có 2 trạng thái tạo mới và lưu. Nếu người sử dụng kích vào nút Tạo mới sẽ chuyển sang trạng thái là lưu và ngược lại.

private void butTaomoi\_Click(object sender, EventArgs e)

{

if (butTaomoi.Text.Equals("Tạo mới"))

{

XoaTextBox();

butTaomoi.Text = "Luu";

}

else

*// Kiểm tra xem mã nhân viên này có hay chưa ?*

if (KiemTraMa(txtmanv.Text)==1)

{

MessageBox.Show("Mã này đã có");

txtmanv.Clear();

txtmanv.Focus();

}

else

{

string ma = txtmanv.Text;

string hoten = txthoten.Text;

string diachi = txtdiachi.Text;

string tendangnhap = txttendangnhap.Text;

string matkhau = "";// Khi tạo 1 tài khoản thì mật khẩu ban đầu là rỗng

string quyenhan = comboBox1.Text;

Moketnoi();

string sql="insert into nhanvien values(" + "'" + ma + "','" + hoten +"','" +diachi + "','" + tendangnhap + "','" + matkhau + "','" + quyenhan + "')";

cmdInsert = new SqlCommand(sql,cn);

cmdInsert.ExecuteNonQuery();

MessageBox.Show("Đã lưu thành công");

LoadListView(); //Nạp lại dữ liệu mới vào listView1

butTaomoi.Text = "Tạo mới";

cmdInsert.Dispose();

}

}

+ Sự kiện **buttimkiem\_Click:** Khi người sử dụng gõ 1 mã nhân viên vào txtmanv và kích vào nút buttimkiem, nếu tìm thấy mã nhân viên này sẽ hiển thị kết quả lên các textBox và comboBox

private void buttimkiem\_Click(object sender, EventArgs e)

{

Moketnoi();

string sql = "select \* from nhanvien where manhanvien='" + txtmanv.Text + "'";

cmdSelect = new SqlCommand(sql,cn);

SqlDataReader dr = cmdSelect.ExecuteReader();

if (dr.Read())// Đã tìm thấy

{

txtmanv.Text = dr[0].ToString();

txthoten.Text = dr[1].ToString();

txtdiachi.Text = dr[2].ToString();

txttendangnhap.Text = dr[4].ToString();

comboBox1.Text = dr[5].ToString();

}

else

MessageBox.Show("Không tìm thấy");

}

+ Sự kiện **butXoabo\_Click:** Xóa nhân viên có mã nhân viên ở txtmanv

private void butXoabo\_Click(object sender, EventArgs e)

{

DialogResult dr = MessageBox.Show("Chắc chắn xóa hay không ?",

"Thông báo", MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes) // Nếu người sử dụng chọn nút yes

{

Moketnoi();

string Sql = "delete from nhanvien where manhanvien='" + txtmanv.Text + "'";

cmdXoa = new SqlCommand(Sql,cn);

if (cmdXoa.ExecuteNonQuery() == 1)

{

MessageBox.Show("Xóa thành công");

LoadListView();

LoadItem(0);

}

else

MessageBox.Show("Không tồn tại mã nhân viên " + txtmanv.Text);

cmdXoa.Dispose();

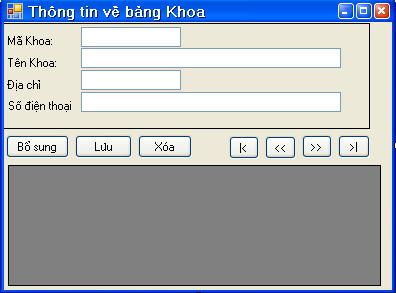
}

}

1. **DataSet và DataBinding**

Ví dụ này sử dụng cơ sở dữ liệu **quanlythuvien** như trong bài thực hành 4.3.4

Thiết kế form **frmkhoa** để nhập, xóa, lưu và duyệt qua các mẫu tin trong bảng khoa như sau:



**frmkhoa** sử dụng các trường, phương thức và sự kiện sau:



#### Các điều khiển

|  |  |
| --- | --- |
| **Tên điều khiển** | **Thuộc tính** |
| Form | **Name**: **frmKhoa**  **Text**: Thông tin về bảng Khoa |
| Label | Tạo ra 4 lable với các **Text**: Mã Khoa, Tên Khoa, Địa chỉ, Số điện thoại |
| TextBox | Tạo ra 4 TextBox với các Name: txtMakhoa, txtTenKhoa, txtDiachi, txtSodienthoai |
| Button | Tạo ra 7 Button với các Name: butBosung, butLuu, butXoa, butFirst, butPre, butNext, butLast |
| dataGridView | **Name**: dataGridView1 |

#### Các phương thức

+ Hàm dựng frmKhoa để tạo giao diện:

public frmKhoa()

{

InitializeComponent();

}

+ Phương thức BuocCacDieuKhien(): Buộc dữ liệu vào dataGridView1 và các textBox

private void BuocCacDieuKhien()

{

//Buộc dữ liệu vào dataGridView1

dataGridView1.DataSource = ds;dataGridView1.DataMember = "khoa";

// Buộc dữ liệu vào các textBox

txtMaKhoa.DataBindings.Add("Text", ds, "khoa.makhoa");

txtTenKhoa.DataBindings.Add("Text", ds, "khoa.tenkhoa");

txtdiachi.DataBindings.Add("Text", ds, "khoa.diachi");

txtSodienthoai.DataBindings.Add("Text", ds, "khoa.sdt");

}

+ **Sự kiện**: frmKhoa\_Load() được sử dụng để kết nối dữ liệu, tạo ra DataSet ds chứa toàn bộ dữ liệu của bảng khoa, buộc dữ liệu vào cho các điều khiển và tạo ra 1SqlCommandBuilder cb để quản lý việc nhập thêm, xóa và lưu dữ liệu của SqlDataAdapter dakhoa .

private void frmKhoa\_Load(object sender, EventArgs e)

{

cn.Open(); // Kết nối dữ liệu

cmdkhoa = new SqlCommand("select \* from khoa", cn);

dakhoa = new SqlDataAdapter(cmdkhoa);

dakhoa.Fill(ds, "khoa");

BuocCacDieuKhien();

cb = new SqlCommandBuilder(dakhoa);

}

+ **Sự kiện**: butFirst\_Click: Di chuyển con trỏ về mẫu tin đầu tiên

private void butFirst\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa"].Position = 0;

}

+ **Sự kiện**: butPre\_Click: Di chuyển con trỏ về mẫu tin trước mẫu tin hiện thời

private void butPre\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa"].Position--;

}

+ **Sự kiện**: butNext\_Click: Di chuyển con trỏ đến mẫu tin kế tiếp

private void butNext\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa"].Position++;

}

+ **Sự kiện**: butLast\_Click: Di chuyển con trỏ về mẫu tin cuối cùng

private void butLast\_Click(object sender, EventArgs e)

{

int ViTriMauTinCuoiCung = this.BindingContext[ds, "khoa"].Count - 1;

this.BindingContext[ds, "khoa"].Position = ViTriMauTinCuoiCung;

}

+ **Sự kiện**: butBosung\_Click: Tạo mới một dòng

private void butBosung\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa"].AddNew();

}

+ **Sự kiện**: butLuu\_Click: Di chuyển con trỏ về mẫu tin cuối cùng, nếu có thay đổi trong DataSet ds thì cập nhật lại dữ liệu, việc cập nhật nhờ vào SqlCommandBuilder cb. Các thao tác bổ sung và xóa chỉ được cập nhật vào cơ sở dư liệu khi người sử dụng kích chuột vào nút Lưu

private void butLuu\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa"].EndCurrentEdit();

if (ds.HasChanges() == true)

{

try

{

dakhoa.Update(ds, "khoa");

MessageBox.Show("Da cap nhat");

}

catch (Exception ll)

{

MessageBox.Show(ll.Message);

}

}

}

+ **Sự kiện**: butXoa\_Click: Lấy vị trí của con trỏ hiện thời, sau đó xóa đi mẫu tin này.

private void butXoa\_Click(object sender, EventArgs e)

{

int donghientai;

donghientai = this.BindingContext[ds, "khoa"].Position;

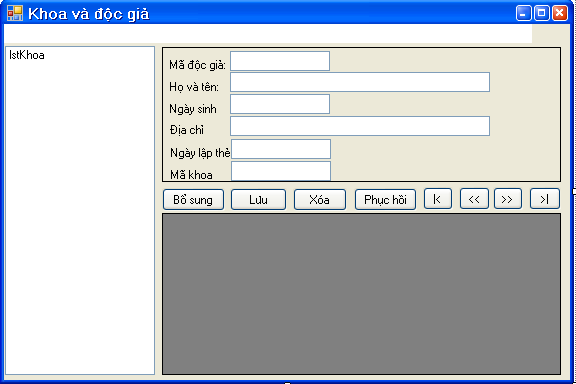
this.BindingContext[ds, "khoa"].RemoveAt(donghientai);

}

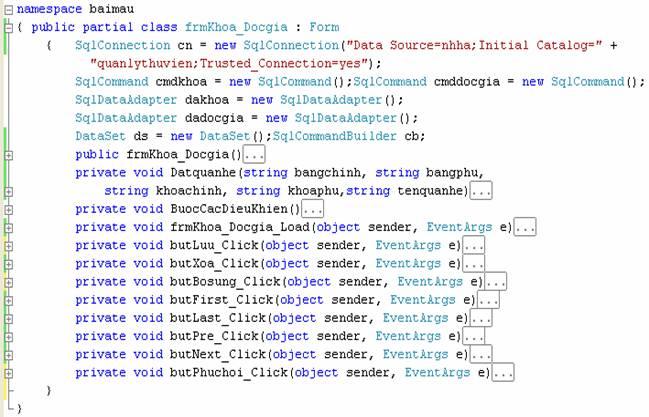
1. **Quan hệ giữa các bảng, DataSet và DataBinding**

Bài thực hành này sử dụng cơ sở dữ liệu **quanlythuvien** liên quan đến hai bảng dữ liệu: Khoa và docgia

Thiết kế form **frmKhoa\_Docgia** để nhập, xóa, lưu, phục hồi và duyệt qua các mẫu tin trong bảng docgia cho từng khoa như sau:



**frmkhoa** sử dụng các trường, phương thức và sự kiện sau:



#### Các điều khiển

|  |  |
| --- | --- |
| **Tên điều khiển** | **Thuộc tính** |
| ListBox | **Name:** lstKhoa |
| Form | **Name**: **frmKhoa\_Docgia**  **Text**: Khoa và độc giả |
| Label | Tạo ra 6 lable với các **Text**: Mã độc giả, Họ và tên, Ngày sinh, Địa chỉ, Ngày lập thẻ và Mã khoa |
| TextBox | Tạo ra 6 TextBox với các Name: txtMadocgia, txtHoten, txtNgaysinh, txtDiachi, txtNgaylapthe, txtMakhoa |
| Button | Tạo ra 8 Button với các Name: butBosung, butLuu, butXoa, butPhuchoi, butFirst (|<), butPre (<<), butNext (>>), butLast (>|) |
| dataGridView | **Name**: |

#### Các phương thức

+ Hàm dựng frmKhoa\_Docgia() để tạo giao diện:

public frmKhoa\_Docgia()

{

InitializeComponent();

}

 + Phương thức Datquanhe bao gồm các tham số: bảng chính, bảng phụ, khóa chính, khóa phụ và tên quan hệ. Phương thức này nạp dữ liệu của 2 bảng: bảng chính và bảng phụ vào DataSet ds, sau đó đặt quan hệ giữa 2 bảng trong DataSet ds.

private void Datquanhe(string bangchinh, string bangphu,string khoachinh, string khoaphu,string tenquanhe)

{

cn.Open();

cmdkhoa = new SqlCommand ("select \* from "+ bangchinh, cn);

dakhoa = new SqlDataAdapter(cmdkhoa);

cmddocgia = new SqlCommand("select \* from " + bangphu, cn);

dadocgia = new SqlDataAdapter(cmddocgia);

ds = new DataSet();

dakhoa.Fill(ds, bangchinh);

dadocgia.Fill(ds, bangphu);

DataColumn chinh=ds.Tables[bangchinh].Columns[khoachinh];

DataColumn phu=ds.Tables[bangphu].Columns[khoaphu];

DataRelation r = new DataRelation(tenquanhe,chinh ,phu );

ds.Relations.Add(r);

}

+ Phương thức BuocCacDieuKhien(): Buộc dữ liệu vào lstKhoa, dataGridView1 và các textBox

private void BuocCacDieuKhien()

{

lstKhoa.DataSource = ds;

lstKhoa.DisplayMember = "khoa.tenkhoa";

dataGridView1.DataSource = ds;

dataGridView1.DataMember = "khoa.khoa\_docgia";

//khoa\_docgia là tên quan hệ của 2 bảng khoa và docgia trong DataSet ds

txtMadocgia.DataBindings.Add("Text", ds, "khoa.khoa\_docgia.madocgia");

txtHoten.DataBindings.Add("Text", ds, "khoa.khoa\_docgia.hoten");

txtNgaysinh.DataBindings.Add("Text", ds, "khoa.khoa\_docgia.ngaysinh");

txtdiachi.DataBindings.Add("Text", ds, "khoa.khoa\_docgia.diachi");

txtNgaylapthe.DataBindings.Add("Text", ds, "khoa.khoa\_docgia.ngaylapthe");

txtMakhoa.DataBindings.Add("Text", ds, "khoa.khoa\_docgia.makhoa");

}

+ **Sự kiện** frmKhoa\_Docgia\_Load: Đặt quan hệ giữa 2 bảng khoa và docgia trong DataSet DataSet ds, tạo ra 1 SqlCommandBuilder để quản lý việc lưu dữ liệu vào cơ sở dữ liệu ,buộc dữ liệu vào các điều khiển trên form:

private void frmKhoa\_Docgia\_Load(object sender, EventArgs e)

{

Datquanhe("khoa", "docgia", "makhoa", "makhoa", "khoa\_docgia");

cb = new SqlCommandBuilder(dadocgia);

BuocCacDieuKhien();

}

 + **Sự kiện**: butFirst\_Click: Di chuyển con trỏ về mẫu tin đầu tiên

private void butFirst\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa.khoa\_docgia"].Position = 0;;

}

+ **Sự kiện**: butPre\_Click: Di chuyển con trỏ về mẫu tin trước mẫu tin hiện thời

private void butPre\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa.khoa\_docgia"].Position--;

}

+ **Sự kiện**: butNext\_Click: Di chuyển con trỏ đến mẫu tin kế tiếp

private void butNext\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa.khoa\_docgia"].Position++;

}

+ **Sự kiện**: butLast\_Click: Di chuyển con trỏ về mẫu tin cuối cùng

private void butLast\_Click(object sender, EventArgs e)

{

int ViTri = this.BindingContext[ds"khoa.khoa\_docgia"].Count - 1;

this.BindingContext[ds, "khoa"].Position = ViTri;

}

+ **Sự kiện**: butBosung\_Click: Tạo mới một dòng

private void butBosung\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa.khoa\_docgia"].].AddNew();

}

+ **Sự kiện**: butLuu\_Click: Di chuyển con trỏ về mẫu tin cuối cùng, nếu có thay đổi trong DataSet ds thì cập nhật lại dữ liệu, việc cập nhật nhờ vào SqlCommandBuilder cb. Các thao tác bổ sung và xóa chỉ được cập nhật vào cơ sở dư liệu khi người sử dụng kích chuột vào nút Lưu

private void butLuu\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa.khoa\_docgia"].EndCurrentEdit();

if (ds.HasChanges() == true)

{

try

{

dakhoa.Update(ds, "docgia");

MessageBox.Show("Da cap nhat");

}

catch (Exception ll) { MessageBox.Show(ll.Message); }

}

}

+ **Sự kiện**: butXoa\_Click: Lấy vị trí của con trỏ hiện thời, sau đó xóa đi mẫu tin này.

private void butXoa\_Click(object sender, EventArgs e)

{

int donghientai;

donghientai = this.BindingContext[ds, "khoa.khoa\_docgia"].Position;

this.BindingContext[ds, "khoa.khoa\_docgia"].RemoveAt(donghientai);

}

+ **Sự kiện**: buttPhuchoi\_Click: Phục hồi lại các thao tác Bổ sung và xóa, dữ liệu chỉ được phục hồi khi chưa lưu vào cơ sở dữ liệu

private void buttPhuchoi\_Click(object sender, EventArgs e)

{

this.BindingContext[ds, "khoa.khoa\_docgia"].CancelCurrentEdit();

ds.RejectChanges();

}

**TÀI LIỆU THAM KHẢO**

1. Thực ra không nhất thiết phải có nhánh else if, chỉ cần else là đủ, bởi vì ở đây chúng ta chỉ áp dụng phương thức này cho hai điều khiển btnClose và btnAbout!. [↑](#footnote-ref-1)
2. Phép chuyển kiểu (sender as Button) trong câu lệnh này là thành công vì cả btnClose và btnAbout đều là các điều khiển kiểu Button [↑](#footnote-ref-2)