**Windows Form**

**C#**

Mục lục

[1. Form 4](#_Toc42596077)

[2. Button 7](#_Toc42596078)

[3. Checkbox 9](#_Toc42596079)

[4. RadioButton 9](#_Toc42596080)

[5. DataGirdView 9](#_Toc42596081)

[5.1 Add Column: Thêm Cột mới vào bảng. 10](#_Toc42596082)

[5.2 Edit Column: Chỉnh sửa thuộc tính cột 10](#_Toc42596083)

[6. TextBox 11](#_Toc42596084)

[7. GroupBox 12](#_Toc42596085)

[8. PictureBox 12](#_Toc42596086)

[9. ToolStrip 13](#_Toc42596087)

[10. TreeView 13](#_Toc42596088)

[11. Label 13](#_Toc42596089)

[12. ComboBox 14](#_Toc42596090)

[13. ListBox 14](#_Toc42596091)

[14. CheckListBox 14](#_Toc42596092)

[15. ListView 15](#_Toc42596093)

[15.1 Thêm một ListView Control vào ứng dụng. 15](#_Toc42596094)

[15.2 Thay đổi chế độ xem. 16](#_Toc42596095)

[15.3 Add các item vào ListView (Khi ListView không theo cách hiển thị Details). 16](#_Toc42596096)

[15.4 Add các cột vào ListView. 17](#_Toc42596097)

[15.5 Add Sub Item vào ListView. 18](#_Toc42596098)

[15.6 Thêm Style cho SubItem. 20](#_Toc42596099)

[15.7 Xóa Item. 20](#_Toc42596100)

[15.8 Liên kết hình ảnh với danh sách các Item. 21](#_Toc42596101)

[15.9 Thêm CheckBox vào trước mỗi item trong ListView 23](#_Toc42596102)

[16. Panel 24](#_Toc42596103)

[16.1 Thêm Panel Control vào Form. 24](#_Toc42596104)

[16.2 Hiển thị và ẩn bảng điều khiển. 25](#_Toc42596105)

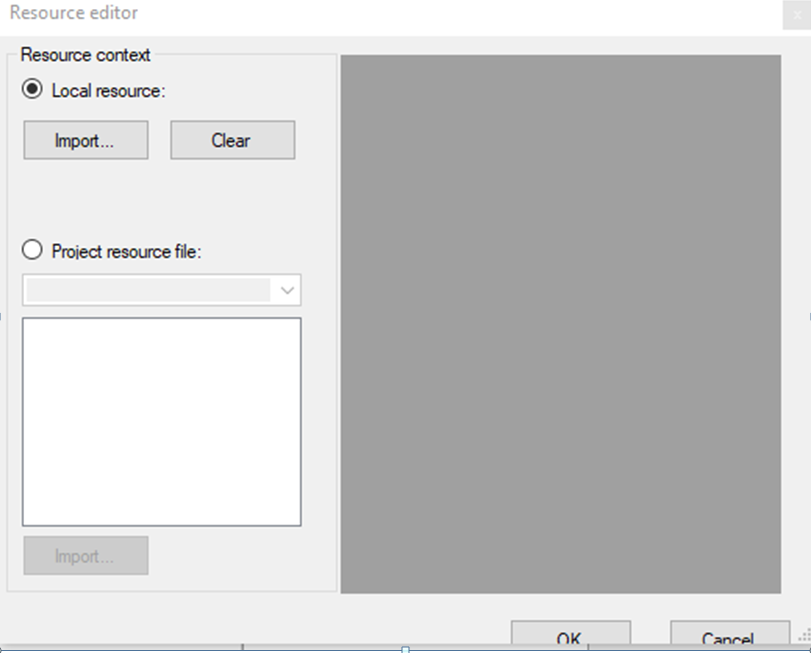
[17. FlowLayout 25](#_Toc42596106)

[18. MenuStrip 25](#_Toc42596107)

[19. StatusBar 26](#_Toc42596108)

[20.MessageBox 26](#_Toc42596109)

# Form

* **BackColor**: Dùng để chỉnh màu nền của form.
* **BackColorImage**: Dùng để thêm ảnh làm nền cho Form. Nếu dùng hình ảnh có sẵn trên máy thì chọn Local resource và bấm Import sau đó chọn hình (Có thể xem hình nền nhìn như thế nào form ở bên phải). Dùng Clear để xóa hình đã chọn. 
* **BackGroundImageLayout**: Hình nền sẽ được bố trí như thế nào.
* ***None***: Hình sẽ được đặt ở góc trên bên trái của form (Giữ nguyên kích cỡ hình).
* ***Title***: Hình sẽ được đặt theo từng ô. Nếu Form mà có kích cỡ lớn hơn hình thì hình sẽ bị lặp lại.
* ***Center***: Hình được đặt ở trung tâm của form.
* ***Stretch***: Hình sẽ được resize lại làm sao cho nó vừa khít và bao phủ toàn bộ form mặc kệ hình nhìn có đẹp hay không.
* ***Zoom***: Hình sẽ được resize để toàn bộ hình đều có thể nhìn được đầy đủ trong form.
* **Cursor**: Chỉnh con trỏ chuột khi rê vào form sẽ nhìn như thế nào (Bàn tay, Đồng hồ, ?,….).
* **Font**: Chỉnh font chữ trong form.
* **ForeColor**: Chỉnh màu chữ trong form.  
  \*Lưu ý: Font và ForeColor khi chỉnh ở form sẽ ảnh hưởng tới tất cả các thành phần trong form nhưng chỉ những thành phần chưa được chỉnh riêng Font và ForeColor cho nó.
* **FormBorderStyle**: Chỉnh kiểu form và liệu nó có thể resize được khi Run hay không.
* **Text**: Chỉnh tên form ở góc trên cùng bên trái.
* **UseWaitCusor**: Nếu true thì khi rê chuột vào form hoặc các thành phần của nó thì chuột sẽ hiện đang load.
* **AllowDrop**: Có cho phép kéo thả những thứ khác vào form hay không.
* **Auto Validate**: Khi rê chuột hoặc click ra khỏi form thì những thành phần trong form sẽ được validate (Xác định xem những gì điền bên trong có đúng theo yêu cầu không ).
* **ContextMenuStrip**: Khi nhấn chuột phải trong form thì sẽ xuất hiện một menuStrip.
* **DoubleBuffer**: Tăng bộ nhớ đệm form lên gấp đôi để giảm độ lag và chập chờn nếu như đồ họa trong form thay đổi liên tục.
* **Enable**: Cho phép người dùng sử dụng thành phần này hay không. Nếu dùng cho form thì nó sẽ vô hiệu toàn bộ thành phần trong form.
* (**Name**): Đây là tên biến của Form được dùng để gọi Form đó trong code. Ví dụ: Nếu Name là Form1 thì để gọi hoặc khai báo Form1 trong Code:   
  From1 form = new Form1();  
  \*Lưu ý: Trong các thành phần của một Form (Label, Button,…)thì nếu như muốn đổi Name trong code thì không phải chỉ đổi ở .Name là đủ. Ví dụ:  
  - Mình có một Button với Name:   
  Button button1 = new Button();  
  button1.Name = “button1”;  
  - Nếu như mình đổi Name của button1 bằng cách:  
  button1.Name=”button1234”; (Sai hoàn toàn)  
  Vì nếu làm như trên thì khi bạn bật qua Design View nó sẽ tự reset Name của button1 lại như cũ trong code.  
  - Cách để đổi Name của button1 thành 1234 là phải đổi toàn bộ những chỗ có button1 trong code:  
  Button button1234 = new Button();  
  button1234.Name = “button1234”;
* **Language**: Ngôn ngữ sử dụng. Sẽ có 2 chế độ chính là Default và tất cả ngôn ngữ khác. Cách sử dụng là:
* Khi ở chế độ Default thì bạn có thể thêm các thành phần khác vào trong Form.
* Khi bạn đổi sang một ngôn ngữ nào đó thì lúc này bạn không thể nào thay đổi cấu trúc form được nữa (Thêm, Sửa, Xóa thành phần) nhưng bạn có thể chỉnh các đặc tính của những thành phần đó thành cái thích hợp cho ngôn ngữ bạn đang chọn.  
  Ví dụ: Khi ở chế độ Default thì mình có thêm một Button với chữ in trên nó là Hello. Nhưng khi mình đổi sang Tiếng Việt thì mình sẽ chữ đó thành Xin Chào. Lúc này nếu như khi đổi về Default thì chữ in trên Button lại tên là Hello còn khi sang tiếng Việt thì nó là Xin Chào.
* **Localizable**: Nếu set cái này về true thì bạn mới có thể sử dụng Language.
* **Locked**: Nếu bạn chỉnh cái này là true thì bạn sẽ không thể nào resize lại Form trong Designer View được nữa nhưng nó sẽ không ảnh hưởng đến việc bạn có thể resize khi Run Form hay không.
* **Causes** **Validation**: Các thành phần mà có cái này là true thì khi bạn rê chuột vào hoặc click vào nó sẽ kích hoạt một số Validation được cài đặt sẵn.
* **AutoScaleMode**: Form và các thành phần của chương trình sẽ được resize lại như thế nào khi độ phân giải màn hình thay đổi hoặc font chữ thay đổi.
* **AutoScroll**: Nếu như mà các control vượt quá kích cỡ form thì thanh trượt sẽ tự động xuất hiện.
* **AutoScrollMargin**: Khoảng cách giữa các control trong form khi sử dụng thanh trượt.
* **AutoScrollMinSize**: Độ dài tối thiểu mà thanh trượt có thể trượt.
* **AutoSize**: Cho phép các thành phần hoặc form có thể tự resize nếu như kích cỡ của những thứ nó chứa thay đổi. Ví dụ như nếu như text của một Button quá lớn để có thể hiển thị toàn bộ trên button thì button sẽ tự thay đổi kích cỡ để chứa toàn bộ text đó.
* **AutoSizeMode**: Khi nào thì các thành phần sẽ tự thay đổi kích cỡ. Có 2 chế độ:
* GrowOnly: Chỉ thay đổi kích cỡ nếu như cần phải tăng kích cỡ để chứa đủ.
* GrowAndShrink: Tự động thay đổi để vừa với những gì chứa bên trong dù tăng hay giảm kích cỡ.
* **Location**: Tọa độ của góc trên cùng bên trái của thành phần so với vật chứa nó.
* **MaximumSize**: Kích cỡ tối đa mà Form có thể kéo giãn ra. (Chiều rộng, chiều dài)
* **MinimumSize**: Kích cỡ tối thiểu mà Form có thể thu nhỏ. (Chiều rộng, chiều dài)
* **Pading**: Các thành phần bên trong Form sẽ cách khỏi 4 cạnh của Form bao nhiêu.
* **Size**: Kích cỡ của Form. (Chiều rộng, Chiều dài).
* **StartPosition**: Khi form xuất hiện thì sẽ xuất hiện ở vị trí như thế nào.
* **WindowState**: Quyết định tình trạng khi xuất hiện của Form:
* Normal: Bình thường.
* Maximized: Khi Form xuất hiện thì nó sẽ luôn ở kích cỡ tối đa. Nếu kích cỡ tối đa là 0,0 thì nó sẽ bao phủ cả màn hình.
* Minimized: Form sẽ xuất hiện dưới dạng icon trên Task Bar của bạn. Và khi bạn bấm vào icon đó thì form sẽ xuất hiện ở kích cỡ nhỏ nhất.
* **Accept** Button: Cài đặt một button vào đây thì khi bạn bấm Enter nó cũng coi như bạn bấm button đó.
* **Cancel** Button: Cài đặt một button vào đây thì khi bạn bấm ESC nó cũng coi như bạn bấm button đó.
* **KeyPreView** (Chưa rõ)
* **ControlBox**: Nếu true thì trên góc bên phải form sẽ có những nút quen thuộc như dấu X để tắt Form hay là hình vuông để phóng to form hoặc dấu trừ để thu form vào Task Bar.
* **HelpButton** (Chưa rõ).
* **Icon**: Biểu tượng của form (Góc trái trên cùng).
* **IsMdiContainer**(Chưa rõ).
* **MainMenuStrip**(Chưa rõ).
* **MaximizedBox**: Quyết định xem nút phóng to form ở trên cùng góc phải có hoạt động hay không. (Hình vuông)
* **MinimizedBox**: Quyết định xem nút thu nhỏ form thành icon trên Task Bar ở trên cùng góc phải của Form có hoạt động hay không. (Dấu trừ)
* **Opacity**: Độ trong suốt của Form.
* **ShowIcon**: Icon của Form có xuất hiện ở góc trái trên cùng Form hay không.
* **ShowInTaskBar**: Khi bấm nút dấu trừ thì liệu Form có xuất hiện ở TaskBar của máy tính hay không.
* **SizeGripStyle** (Chưa rõ).
* **TopMost**: Nếu true thì Form luôn luôn xuất hiện đè lên những form khác mà cái thuộc tính này là False.
* **Transparency**: Chọn một màu mà khi màu đó xuất hiện trên Form nó sẽ trở nên trong suốt.

*(Từ lúc này trở đi nếu như có bất kỳ thuộc tính hoặc Tab nào mà mình không nhắc tới trong các Component dưới đây là bởi vì nó giống trên Form.)*

# Button

* FlatStyle: Kiểu dáng và hình dạng của nút khi rê chuột qua và click chuột vào. Có 4 lựa chọn:
* Flat
* Standard
* System
* PopUp
* FlatAppearance: Nếu như bạn muốn tự chỉnh màu sắc và hình dạng nút khi rê chuột qua và click chuột vào thay vì chọn những kiểu có sẵn trên FlatStyle thì sử dụng thuộc tính này. Lưu ý là để sử dụng thuộc tính này thì FlatStyle phải chọn Flat. Thuộc tính có những chỉnh sửa sau:
* BorderColor: Màu viền bên ngoài của nút.
* BorderSize: Kích cỡ viền bên ngoài của nút.
* MouseDownBackColor: Màu nút khi nút được click vào.
* MouseOverBackColor: Màu nút khi nút được rê chuột qua.
* Image: Hình ảnh hiện trên nút.
* ImageAlign: Vị trí của hình ảnh trên nút.
* ImageIndex, Key, List: Cả 3 thuộc tính này liên quan đến ImageList và vì trong tài liệu này không nhắc tới nó nên mình sẽ không nói ở đây.
* Text: Chữ hiện trên nút.
* TextAlign: Chữ sẽ ở vị trí nào trên nút.
* TextImageRelation: Ảnh và nút trên hình sẽ hiện như thế nào. Có các chế độ sau:
* OverLay: Chữ đè lên hình.
* ImageAboveText: Hình sẽ hiện lên trước sau đó tới chữ (Hiện theo hàng dọc).

|  |
| --- |
| Hình |
| Chữ |

* TextAboveImage: Chữ sẽ hiện lên trước sau đó tới hình (Hiện theo hàng dọc).

|  |
| --- |
| Chữ |
| Hình |

* ImageBeforeText: Hình sẽ hiện lên trước sau đó tới chữ (Hiện theo hàng ngang và chữ sẽ bị hiện theo hàng dọc).

|  |  |
| --- | --- |
| Hình | Chữ |

* TextBeforeImage: Chữ sẽ hiện lên trước sau đó tới Hình (Hiện theo hàng ngang và chữ sẽ bị hiện theo hàng dọc).

|  |  |
| --- | --- |
| Chữ | Hình |

* UseMemonic: Cho phép sử dụng phím tắt trên bàn phím để bấm nút. Cách sử dụng:
* Set thuộc tính này về true.
* Ở thuộc tính Text thêm & vào trước chữ cái muốn làm phím tắt. Ví dụ: &button1 thì phím b trên bàn phím sẽ được sử dụng làm phím tắt cho nút này.
* UseVisualStyle: (Chưa rõ)
* AutoElipisis: Nếu true thì khi chữ trên nút không thể hiện đầy đủ trên nút do quá dài thì khi rê chuột vào nút sẽ xuất hiên một dòng chú thích hiện toàn bộ nội dung của chữ.
* DialogResult: (Chưa Rõ).
* TabStop: Nếu true thì có thể dùng nút tab để di chuyển đến nút.
* TabIndex: Thứ tự của nút khi sử dụng tab. Ví dụ: Button1 có TabIndex là 0, Button 2 là 2, Button 3 là 1 thì khi bấm Tab 3 lần thứ tự di chuyển sẽ làn Button1 > Button3 > Button2.
* UseCompatibileTextRendering: Xem liệu chữ hiện trên nút có phải tự điều chỉnh để phù hợp với các phiên bản trước của hệ điều hành Window hay không.
* Visible: Nút có thể được thấy bởi người dùng hay không.
* GenerateMember:
* Modifier: Nếu như GenerateMember là true thì chúng ta có thể sử dụng Modifier để chỉnh sử Modifier của nút (Private, Public, Protected…).
* Anchor: (Chưa rõ).
* Dock: Chọn vị trí để của nút ở trong container của nó.
* \*Lưu ý: Khi sử dụng thuộc tính này nó sẽ làm thay đổi kích cỡ nút để phù hợp vị trí bạn chọn.

# Checkbox

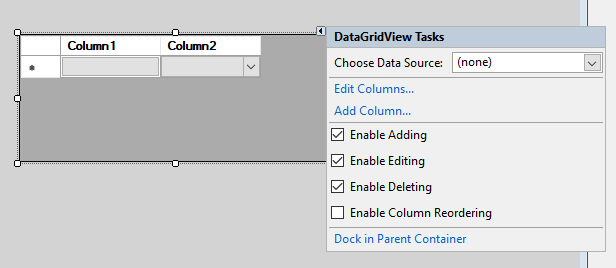
Hầu hết thuộc của CheckBox đều giống như Button. Nhưng có một thuộc tính đáng chú ý:

**ThreeState**: cho phép CheckBox có 3 trạng thái thay vì 2.

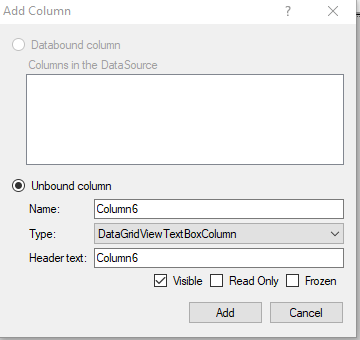
# RadioButton

Tất cả thuộc tính đều như Button. Để nhóm các RadioButton lại thì ta dùng GroupBox (Bắt buộc).

# DataGirdView

Dùng để hiện hiện dữ liệu dưới dạng bảng. (Phần này mình chỉ giải thích cách sử dụng cơ bản mà thôi).

## Add Column: Thêm Cột mới vào bảng.

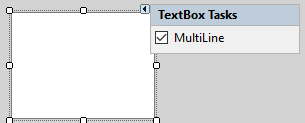


* Name: Cách sử dụng giống với Name trên Form.
* Type: Loại Cột.
* TextBoxColumn: Cột chứa các Text Box.
* ButtonColumn: Cột chứa các Button.
* CheckBoxColumn: Cột chứa các CheckBox.
* ComboBoxColumn: Cột chứa các ComboBox.
* ImageColumn: Cột chứa các Image.
* LinkColumn: Cột chứa các Đường dẫn.
* Header: Tên Cột.
* Visible: Liệu Cột có thể thấy được hay không.
* ReadOnly: Liệu Dữ Liệu trong cột có thay đổi được hay không.
* Frozen: Cột có bị đóng băng hay không. Sử dụng khi nếu như có quá nhiều cột nên nó vượt quá kích cỡ của DataGridView và bạn luôn cần phải thấy một cột nào đó khi Scroll.

## Edit Column: Chỉnh sửa thuộc tính cột

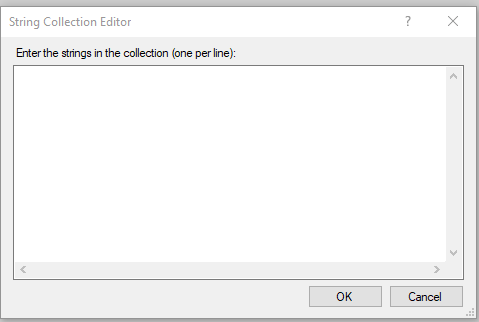
* Enable Adding: Cho phép người dùng thêm dữ liệu mới vào hay không.
* Enable Editing: Cho phép người dùng sửa dữ liệu hay không.
* Enable Deleting: Cho phép người dùng xóa dữ liệu hay không.
* Dock In Parent Container: Cho DataGridView bao phủ cả container đang chứa nó.

# TextBox



\*Nếu chọn MutiLine thì TextBox sẽ có nhiều hàng.

* Line: Thêm sẵn những dòng text vào Textbox (Một dòng mỗi hàng).



* ScrollBar: Xác định xem text Box này sẽ có những thanh trượt nào(Chỉ khi MutiLine):
* Vertical: Thanh trượt Dọc.
* Horizontal: Thanh trượt ngang. Nếu như WordWarp là true thì than trượt này sẽ không xuất hiện.
* Both: Cả hai thanh.
* None: Không sử dụng thanh trượt.
* Text: Giống như Lines.
* Text Align: Chữ sẽ ở phía nào của Textbox.
* AcceptReturn: Nếu như là true thì khi bấm Enter trong TextBox Mutilne sẽ là xuống dòng. Còn nếu False thì khi Enter sẽ kích hoạt nút mặc định của Form (Thuộc tính AcceptButton trên Phần Form) và để xuống dòng phải dùng CTRL+ENTER nhưng nếu như Form ko có nút mặc định thì dù thuộc tính này giá trị như thế nào thì Enter cũng sẽ là xuống dòng.
* AcceptTab: Có cho phép sử dụng nút tab trong TextBox hay không.
* CharacterCasing: Khi nhập chữ vào thì tất cả sẽ biến thành chữ hoa hoặc chữ thường hoặc không thay đổi tùy theo lựa chọn.
* HideSelection: Nếu true thì khi rê chuột hoặc click ra chỗ khác thì các chữ đã tô đen sẽ không còn tô đen. Nếu False thì các chữ đó sẽ vẫn tô đen.
* MaxLength: Số ký tự tối đa mà TextBox có thể chấp nhạn.
* Mutiline: TextBox có phải là Mutiline hay không. (Nhiều dòng)
* PasswordChar: Chọn một ký tự làm ký tự mật khẩu mà khi nhập thì tất cả ký tự sẽ sẽ trở thành ký tự đó. (Chỉ dùng cho Single Line)
* ReadOnly: Liệu chữ trong text box có thể bị chỉnh sửa hay không.
* TabStop: Tương tự trên Button.
* TabIndex: Tương tự trên Button.
* UseSystemPasswordChar: Sử dụng ký tự mật khẩu mặc định của hệ thống.
* Visible: Tương tự button.
* WordWarp: Khi chữ nhập vào vượt quá chiều dài của TextBox Mutiline thì có tự xuống dòng hay không.

# GroupBox

\*Hầu hết các thuộc tính đều tương tự Form.  
\*Cái này như là một container dùng để nhóm các thành phần lại nhằm mục đích Logic. Ví dụ như mình có GroupBox tên là cấu hình và bên trong nó sẽ là các RadioButton của các lựa chọn cấu hình như (Thấp, Cao,…).

# PictureBox

* BorderStyle: Kiểu viền ngoài của control.
* ErrorImage: hình ảnh hiển thị khi có lỗi xảy ra trong quá trình load ảnh.
* Image: hinh ảnh hiển thị trong PictureBox.
* ImageLocation: Url của hình.
* InitialImage: Hình ảnh hiển thị trong quá trinh load ảnh.
* SizeMode: Xác định kiểu hiển thị của hình, có các giá trị sau:
* Normal: Là kiểu mặc định, hiển thị hình ảnh từ góc trên bên trái đến hết chiều dài và chiều rộng của PictureBox, phần thừa của hình sẽ không hiển thị.
* StrecthImage: Hình ảnh vừa khít với tỉ lệ của PictureBox.
* Zoom: Hinh ảnh được thu phóng vừa với PictureBox nhưng tỉ lệ hình giữ nguyên.
* WaitOnLoad: Quá trinh load hình có đồng bộ không, giá trị mặc định là false.

# ToolStrip

* AllowDrop: Nếu giá trị của AllowDrop là true thì thực hiện việc kéo thả và sắp xếp sẽ được xử lý theo code ta viết.
* AllowItemReorder: Nếu giá trị là true thì kéo thả và sắp xếp tự động.
* AllowMerge: Nếu true cho phép một vài control như MenuStrip, ToolStripDropdownMenu,… vv có thể hợp lại làm một.
* CausesValidation: Set giá trị true sẽ hiển thị thông báo xác nhận khi nhấp chọn các control có yêu cầu xác nhận.
* DefaultDropDownDirection: Vị trí xổ xuống của dropdown list so với ToolStrip. Với cú pháp của vị trí: ToolStripDropDownDirection.[vị trí].
* DisplayedItems: Trả về tập con các item đang hiển thị trong ToolStrip.
* Hscroll: (true) Hiển thị thanh cuộn ngang.
* Orientation: Trả về hướng của Panel( Horiziontal/Vertical).
* Renderer: Sử dụng bộ kết xuất nào từ ToolStripRenderer.
* ShowItemToolTips: Hiển thị các tip hay không .
* TextDirection: ToolStripTextDirection.[hướng]: viết text theo chiều nào.
* Vscroll: Thanh cuộn dọc.

# TreeView

* HasChildren: Nếu có node con, getter trả về true.
* Indent: Khoảng cách thụt lề mỗi bậc.
* Nodes: Lấy tập hợp các node của TreeView.
* PathSeparator: Dấu phân cách trong đường dẫn của các node.
* SelectedNode: Get/set node đang chọn.
* ShowLines: Hiển thị đường nối giữa các node.
* ShowPlusMinus: Hiển thị dấu (+), (-) kế bên các node có node con.
* Sorted: Sắp xếp các node.
* TopNode: Trả về node đầu tiên hiển thị đầy đủ.
* Visible: Node và các node con có hiển thị ra hay không.

# Label

* FlatStyle: Get/set kiểu nổi của label
* FlatStyle.Flat: Dạng phẳng.
* FlatStyle.PopUp: Phẳng nhưng khi rê chuột đến sẽ chuyển sang 3D.
* Standard: Dạng 3D.
* System: Tùy theo hệ điều hành.
* UseCompatibleTextRendering: Dùng GDI+ hay GDI để kết xuất text. Nên để mặc định là false (GDI).

# ComboBox

* AutoCompleteCustomSource: Nhập các chuỗi vào làm option cho ComboBox
* AutoCompleteMode: Tùy chọn cho ComboBox tự hoàn thiện từ khóa
* Append: Thêm các ký tự vào để tạo thành chuỗi gần giống ký tự nhập vào.
* None: Tắt tính năng tự hoàn thiện.
* Suggest: Tạo ra 1 dropdown list gồm các chuỗi bắt đầu bằng các ký tự đã nhập .
* SuggetAppend: Kết hợp Suggest và Append.
* AutoCompleteSource: Xác định kiểu source dùng cho autocomplete.
* FormatString: Định dạng cách các giá trị được hiển thị.
* IntegralHeight:(true) Tự thay đổi kích thước của ComboBox để tránh hiển thị không hết các item.
* MaxDropDownItem: số item có thể hiển thị tối đa trong dropdown list.

# ListBox

* AllowSelection:(bool) listbox cho phép chọn các item.
* ColumnWidth: Chiều dài của cột trong multicolumn listbox.
* DataSource: Các obj như DataSet, Array để làm dữ liệu.
* HorizontalExtended: Độ dài thanh cuộn ngang có thể cuộn.
* HorizontalScrollBar: Hiển thị thanh cuộn ngang.
* MultiColumn: ListBox có hỗ trợ nhiều cột không.
* ScrollAlwaysVisible: Thanh cuộn dọc luôn hiện hay không.

# CheckListBox

* CheckedIndices: Tập hợp các vị trí đã đánh dấu trong CheckedListBox.
* CheckedItems: Tập hợp các item đã đánh dấu trong CheckedListBox.
* CheckOnClick: (true) Khi item được chọn thì checkbox sẽ đánh dấu.
* Items: Tập hợp các item trong CheckedListBox.
* ThreeDCheckBox:(true) Checkbox phẳng.

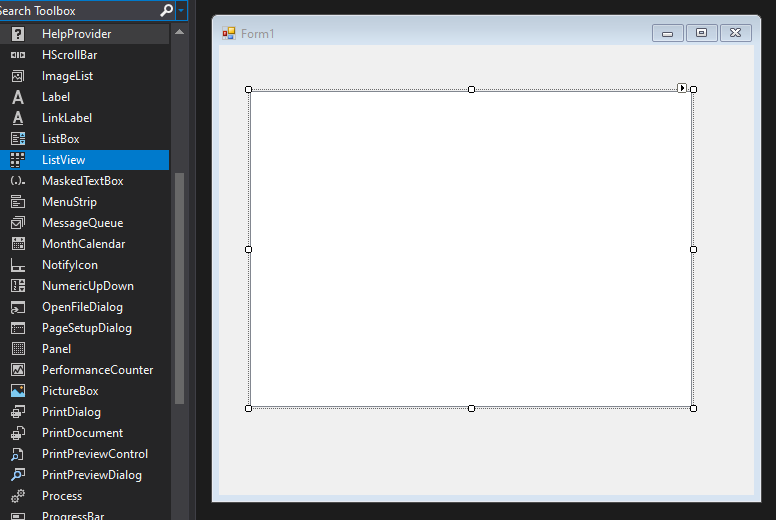
# ListView

ListView là một control dung để hiển thị một danh sách các item các biểu tượng.

ListView Control là là một control có nguồn gốc từ Listbox Control, tuy nhiên ListView có nhiều tùy biến hơn cho các lập trình viên. Sau khi chạy sẽ có giao diện như trên.

## Thêm một ListView Control vào ứng dụng.

Có 2 cách thêm một ListView vào ứng dụng:

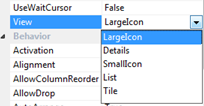
* Kéo thả ListView từ ToolBox vào Form
* Sử dụng code để add vào Form

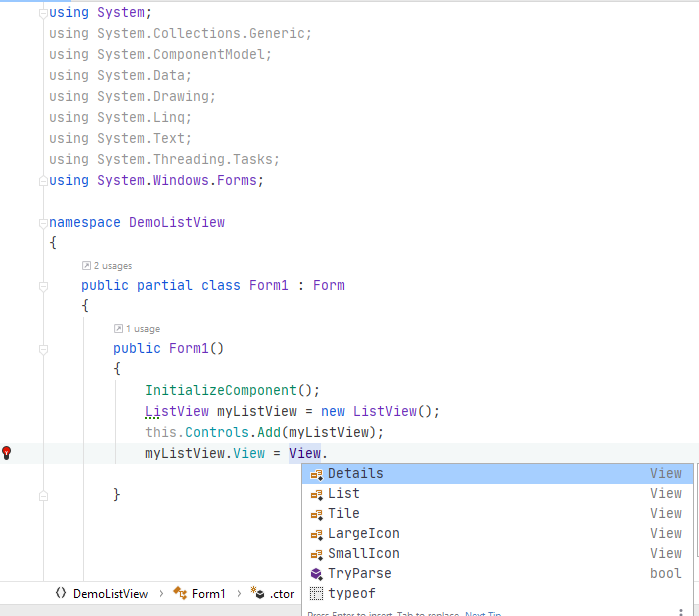
ListView myListView = new ListView(); // Khai báo một ListView control.

myListView.Size = new System.Drawing.Size(390, 100); // Kích thước hiển thị

this.Controls.Add(myListView); // Add ListView control vừa khai báo vào Form

## Thay đổi chế độ xem.

* Tùy chỉnh thuộc tính View trong cửa sổ Properties của Listview. Sẽ có 4 thuộc tính hiển thị để chúng ta lựa chọn:  LargeIcon, Details, SmallIcon, List Tile.
* Sử dụng code để tùy chỉnh thuộc tính trong View:



## Add các item vào ListView (Khi ListView không theo cách hiển thị Details).

* Sử dụng thuộc tính Items trong cửa sổ Properties. Khi click vào button … ở thuộc tính Items. Thì cửa sổ như hình dưới sẽ hiện ra để bạn add item vào.
* Mỗi Item add sẽ có các thuộc tính như: Text, ForeColor, Text, ImageIndex…
* Chúng ta cũng có thể viết code để add các item vào ListView với mục đích tương tự cách làm trên. Ví dụ:

myListView.Items.Add("Công Nghệ Thông Tin");

myListView.Items.Add("Bách Khoa");

myListView.Items.Add ("Khoa Học Tự Nhiên");

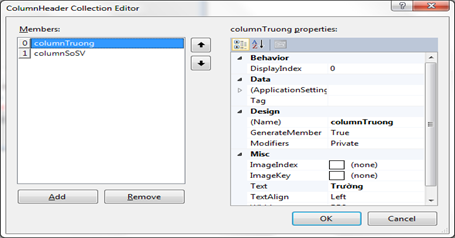
myListView.Items.Add("Nhân Văn");

myListView.Items.Add("Kinh Tế - Luật");

Và kết quả:

## Add các cột vào ListView.

* Chúng ta cũng có thể thực hiện một cách đơn giản như cách add các items ở trên:

[](http://thanhcuong.files.wordpress.com/2011/04/image17.png)

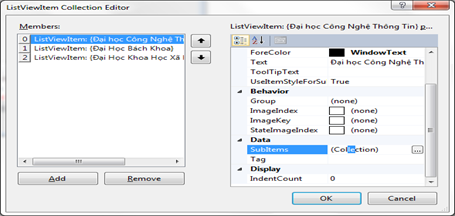
* Hoặc cũng có thể sử dụng code:

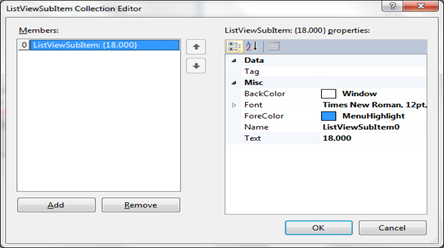
myListView.Columns.Add("Tên Trường", 200);

myListView.Columns.Add("Số lượng sinh viên", 100);

\*Lưu ý: Để có thể hiển thị các columns thì chúng ta phải chọn chế độ xem là Details

## Add Sub Item vào ListView.

* [](http://thanhcuong.files.wordpress.com/2011/04/image19.png)Sử dụng giống như cách add các items trong phần 3 đã trình bày. Ở chúng ta click vào thuộc tính SubItem một cửa sổ mới sẽ hiện ra khá giống với cửa sổ add items:
* Giờ chúng ta có thể add các item con cho item chính một cách bình thường giống như khi add item chính.

[](http://thanhcuong.files.wordpress.com/2011/04/image20.png)

* Chúng ta cũng có thể sử dụng code để add các giá trị con cho item như:

// Add subitem

ListViewItem cntt = new ListViewItem("Công Nghệ Thông Tin");

ListViewItem.ListViewSubItem svcntt = new ListViewItem.ListViewSubItem(cntt, "3.000 sinh viên");

cntt.SubItems.Add(svcntt);

myListView.Items.Add(cntt);

ListViewItem bk = new ListViewItem("Bách Khoa");

ListViewItem.ListViewSubItem svbk = new ListViewItem.ListViewSubItem(bk, "18.00 sinh viên");

bk.SubItems.Add(svbk);

myListView.Items.Add(bk);

ListViewItem khtn = new ListViewItem("Khoa Học Tự Nhiên");

ListViewItem.ListViewSubItem svkhtn = new ListViewItem.ListViewSubItem(khtn, "20.000 sinh viên");

khtn.SubItems.Add(svkhtn);

myListView.Items.Add(khtn);

ListViewItem nv = new ListViewItem("Khoa Học Xã Hội & Nhân Văn");

ListViewItem.ListViewSubItem svnv = new ListViewItem.ListViewSubItem(nv, "15.000 sinh viên");

nv.SubItems.Add(svnv);

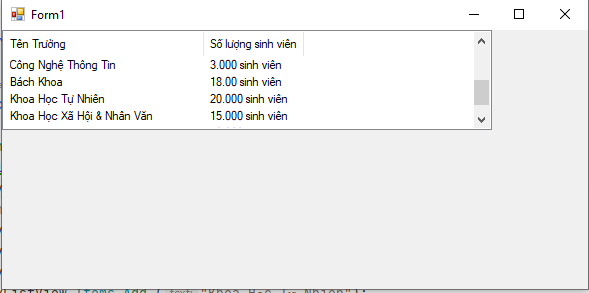
myListView.Items.Add(nv);

ListViewItem ktl = new ListViewItem("Kinh Tế - Luật");

ListViewItem.ListViewSubItem svktl = new ListViewItem.ListViewSubItem(ktl, "10.000 sinh viên");

ktl.SubItems.Add(svktl);

myListView.Items.Add(ktl);



## Thêm Style cho SubItem.

* Trong chế độ xem Details, chúng ta muốn thêm các hiển thị khác nhau của cách SubItems từ Item cha chúng ta sử dụng thuộc tính UseItemStyleForSubItem = true; Như vậy chúng ta sẽ xác định được các kiểu khác nhau cho các subitems.
* Ví dụ:

ListViewItem ktl = new ListViewItem("Kinh Tế - Luật");

ListViewItem.ListViewSubItem svktl = new ListViewItem.ListViewSubItem(ktl, "10.000 sinh viên");

ktl.SubItems.Add(svktl);

myListView.Items.Add(ktl);

ktl.UseItemStyleForSubItems = true;

## Xóa Item.

* Việc xóa các item của ListView được thực hiện rất đơn giản.
* Để xóa toàn bộ các item trong ListView có tên là myListView ta thực hiện lệnh:

myListView.Clear();

* Để xóa item nào ta gọi phương thức Remove():

ListViewItem cntt = new ListViewItem("Công Nghệ Thông Tin");

cntt.Remove();

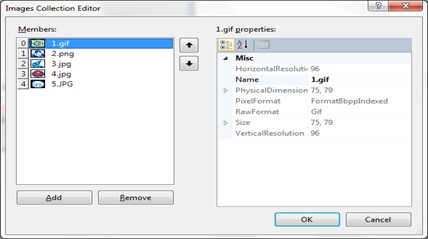
* Xóa item ở vị trí thứ a trong ListView ta sử dụng phương thức RemoveAt():

myListView.Items.RemoveAt(2);

## Liên kết hình ảnh với danh sách các Item.

* Đây là một tính năng làm đẹp cho ListView. Để liên kết các items trong danh sách chúng ta cần phải có một imageList với một tập hợp các ảnh. Điều này được thực hiện trong trong phương thức ListView.Items.Add(…), sử dụng đối số imageIndex – là chỉ mục liên kết với hình ảnh trong imageList.
* Đầu tiên kéo một imageList từ Toolbox vào Form (tên mặc định sẽ là imageList1)

[](http://thanhcuong.files.wordpress.com/2011/04/image22.png)

* [](http://thanhcuong.files.wordpress.com/2011/04/image23.png)Trong thuộc tính Images của imageList1 sẽ được sử dụng để add hình ảnh vào imageList1 như:
* Bây giờ ta sử dụng đối số imageIndex trong phương thức add item vào listview để liên kết hình ảnh với imageList1:

myListView.SmallImageList = imageList1; // Liên kết danh sách hình ảnh nhỏ với imageList1

ListViewItem cntt = new ListViewItem("Công Nghệ Thông Tin", 0);

ListViewItem.ListViewSubItem svcntt = new ListViewItem.ListViewSubItem(cntt, "3.000 sinh viên");

cntt.SubItems.Add(svcntt);

myListView.Items.Add(cntt);

ListViewItem bk = new ListViewItem("Bách Khoa", 1);

ListViewItem.ListViewSubItem svbk = new ListViewItem.ListViewSubItem(bk, "18.00 sinh viên");

bk.SubItems.Add(svbk);

myListView.Items.Add(bk);

ListViewItem khtn = new ListViewItem("Khoa Học Tự Nhiên",2);

ListViewItem.ListViewSubItem svkhtn = new ListViewItem.ListViewSubItem(khtn, "20.000 sinh viên");

khtn.SubItems.Add(svkhtn);

myListView.Items.Add(khtn);

ListViewItem nv = new ListViewItem("Khoa Học Xã Hội & Nhân Văn", 3);

ListViewItem.ListViewSubItem svnv = new ListViewItem.ListViewSubItem(nv, "15.000 sinh viên");

nv.SubItems.Add(svnv);

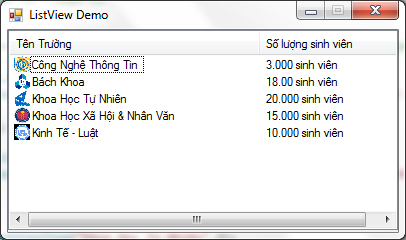
myListView.Items.Add(nv);

ListViewItem ktl = new ListViewItem("Kinh Tế - Luật", 4);

ListViewItem.ListViewSubItem svktl = new ListViewItem.ListViewSubItem(ktl, "10.000 sinh viên");

ktl.SubItems.Add(svktl);

myListView.Items.Add(ktl);

\*Lưu ý đối số thứ 2 trong phương thức add item chính là chỉ mục tham chiếu tới hình ảnh trong imageList1.

## Thêm CheckBox vào trước mỗi item trong ListView

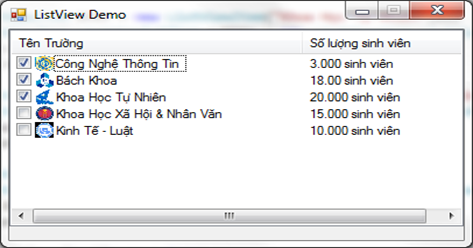
* Đôi khi chúng ta cần một checkbox đứng trước mỗi item trong danh sách của Listview ví dụ như: khi chọn các món hàng trong danh sách các hàng hóa… Ta thực hiện phương thức sau đây:

myListView.CheckBoxes = true;

* Bây giờ chúng ta có thể xử lý sự kiện item nào được chọn bằng cách thêm xử lý trong sự kiện

myListView.ItemChecked += new ItemCheckedEventHandler(myListView\_ItemChecked);

* Hình ảnh minh họa ListView khi thêm checkbox:

[](http://thanhcuong.files.wordpress.com/2011/04/image25.png)

# Panel

## Thêm Panel Control vào Form.

Có 2 cách để thêm Panel vào form:

* Kéo thả từ Toolbox vào Form
* Thêm bằng code:

Panel dynamicPanel = newPanel();

dynamicPanel.Location = **new** System.Drawing.Point(26, 12);

dynamicPanel.Name = "Panel1";

dynamicPanel.Size = **new** System.Drawing.Size(228, 200);

dynamicPanel.TabIndex = 0;

Đặt thuộc tính cho Panel

* Sử dụng thanh công cụ Properties
* Sử dụng code:

privatevoid CreateButton\_Click(object sender, EventArgs e) {

    Panel dynamicPanel = newPanel();

    dynamicPanel.Location = **new** System.Drawing.Point(26, 12);

    dynamicPanel.Name = "Panel1";

    dynamicPanel.Size = **new** System.Drawing.Size(228, 200);

    dynamicPanel.BackColor = Color.LightBlue;

    TextBox textBox1 = newTextBox();

    textBox1.Location = newPoint(10, 10);

    textBox1.Text = "I am a TextBox5";

    textBox1.Size = newSize(200, 30);

    CheckBox checkBox1 = newCheckBox();

    checkBox1.Location = newPoint(10, 50);

    checkBox1.Text = "Check Me";

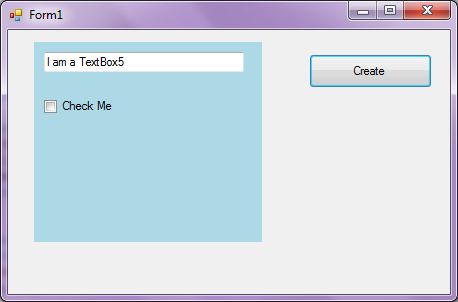
    checkBox1.Size = newSize(200, 30);

    dynamicPanel.Controls.Add(textBox1);

    dynamicPanel.Controls.Add(checkBox1);

    Controls.Add(dynamicPanel);

}



## Hiển thị và ẩn bảng điều khiển.

* dynamicPanel.Visible = **false**;
* dynamicPanel.Visible = **true**;

# FlowLayout

Flow LayoutPanel tương tự như Panel nhưng các control được thêm vào bên trong sẽ được sắp xếp theo vị trí ngay ngắn thay vì ngẫu nhiên như Panel.

# MenuStrip

* MenuStrip cho phép thiết kế hệ thống menu trên Form menu một cấp hay nhiều cấp. Ví dụ hệ thống menu của chương trình Word, Visual,..
* MenuStrip cho phép thiết kế menu với các điều khiển:
* ToolStripSeparator – Gạch phân cách
* ToolStripMenuItem – Menu con
* ToolStripCombobox – Combobox
* ToolStripTextbox – Textbox
* Một số thuộc tính thường dung của MenuStrip:
* TextDirection: chọn hình thức trình bày menu quay ngược, quay 90 độ
* Item: Thêm các menu con, kiểu của menu con
* RightToLeft: nhận một trong hai giá trị Yes hay No – Yes: trình bày từ phải qua trái, No: ngược lại.

# StatusBar

* StatusBar là thanh điều khiển trạng thái, được đặt ở cuối biễu mẫu dung để hiển thị một số văn bản đại diện cho trạng thái của ứng dụng và trạng thái của người dung
* StatusBar không chỉ cung cấp chức năng thanh trạng thái mà còn cung cấp các tính năng để thêm giao diện người dùng phong phú vào thanh trạng thái như một ProgressBar, DropDownButton và SplitButton điều khiển.
* Cách thêm và sử dụng StatusBar cũng như các control cơ bản trên.

# 20.MessageBox

* MessageBox hiển thị thông báo và cho phép người sử dụng tương tác lựa chọn trên cửa sổ.
* Các thông số bên trong Messagebox

|  |  |
| --- | --- |
| **Name** | **Description** |
| Show(String) | Show a message box with text |
| Show(String,String) | Show a message box with text and caption |
| Show(String, String, MessageBoxButton) | Show a message box with text and caption, and button |
| Show(String, String, MessageBoxButton, MessageBoxIcon) | Show a message box with text and caption, button, and icon |

MessageBox.Show("Hello");

MessageBox.Show("Hello", "Title", MessageBoxButtons.YesNoCancel);

MessageBox.Show("Hello", "Title", MessageBoxButtons.OK, MessageBoxIcon.Question);

# Bài tập Demo

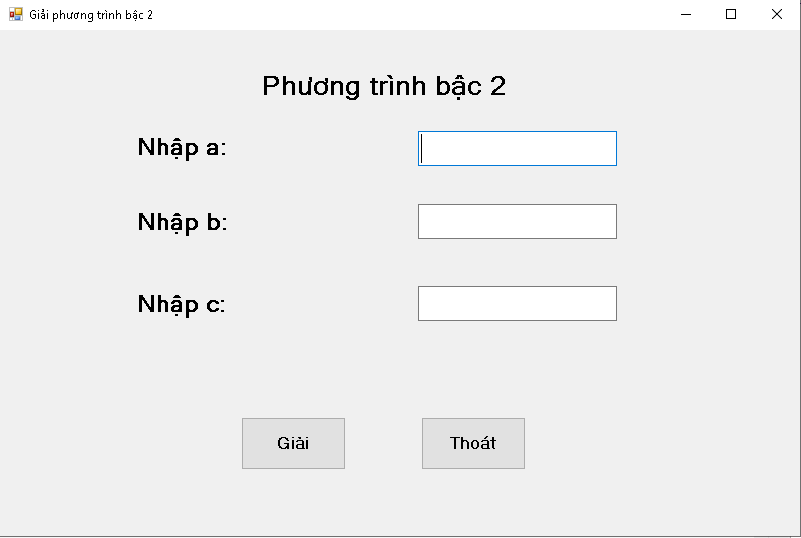
## A screenshot of a cell phone Description automatically generatedVẽ bàn cờ caro

using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace CoCaro  
{  
 public partial class Form1 : Form  
 {  
 public Form1()  
 {  
 InitializeComponent();  
 DrawChessBoard();  
 }  
  
 void DrawChessBoard()  
 {  
 Button oldButton = new Button() {Width = 0, Location = new Point(0, 0)};  
 for (int i = 0; i < Constent.CHESSBOARD\_HEIGHT; i++)  
 {  
 for (int j = 0; j < Constent.CHESSBOARD\_WIDTH; j++)  
 {  
 Button btn = new Button() *// tạo ra đối tượng button* {  
 Width = Constent.C\_WIDTH,  
 Height = Constent.C\_HEIGHT,  
 Location = new Point(oldButton.Location.X + oldButton.Width, oldButton.Location.Y)  
 }  
 ;  
 pnlChessBoard.Controls.Add(btn);  
 oldButton = btn; *// Mỗi lần add lưu tọa độ oldButton lại để sử dụng tiếp* }  
  
 oldButton.Location = new Point(0, oldButton.Location.Y + Constent.C\_HEIGHT);  
 *// Cập nhật lại tọa độ của oldButton mỗi khi xuống dòng* oldButton.Width = 0;  
 oldButton.Height = 0;  
 }  
  
 }  
 }  
}

namespace CoCaro  
{  
 public class Constent  
 {  
 public static int C\_WIDTH=30;  
 public static int C\_HEIGHT=30;  
  
 public static int CHESSBOARD\_HEIGHT = 19;  
 public static int CHESSBOARD\_WIDTH = 19;  
 }  
}

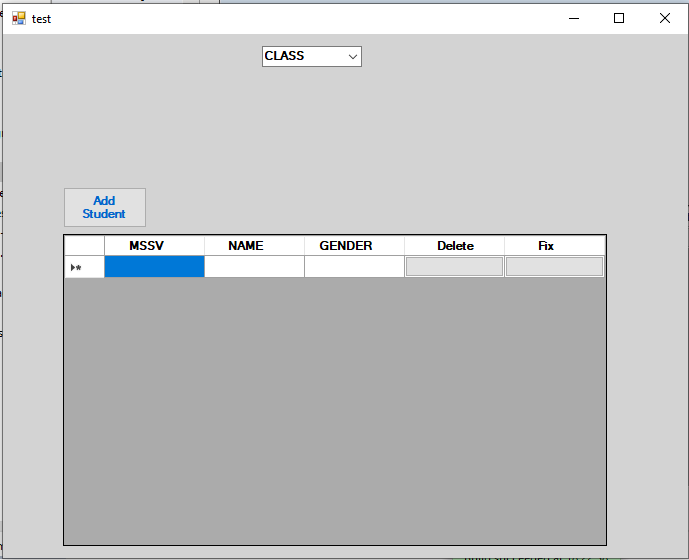
## Giải phương trình bậc 2

using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace PTBac2  
{  
 public partial class Form1 : Form  
 {  
 public Form1()  
 {  
 InitializeComponent();  
 }  
  
 private void button1\_Click(object sender, EventArgs e)  
 {  
 #region Process  
 try  
 {  
  
 double a = double.Parse(textBox1.Text);  
 double b = double.Parse(textBox2.Text);  
 double c = double.Parse(textBox3.Text);  
 double detal = b \* b - 4 \* a \* c;  
 double x, x1, x2;  
  
   
 if (a == 0)  
 {  
 if (b == 0)  
 {  
 if (c == 0)  
 {  
 MessageBox.Show("Phương trình có vô số nghiệm");  
 }  
 else  
 {  
 MessageBox.Show("Phương trinhg vô nghiệm");  
 }  
 }  
 else  
 {  
 MessageBox.Show("Phương trinhg có 1 nghiem x= " + (-c / b));  
 }  
 }  
 else  
 {  
 if (detal < 0)  
 {  
 MessageBox.Show("Phương trinhg vô nghiệm");  
 }  
  
 if (detal == 0)  
 {  
 MessageBox.Show("Phương trình có nghiem kép x1=x2= " + -b / (2 \* a));  
 }  
  
 if (detal > 0)  
 {  
 x1 = (-b + Math.Sqrt(detal)) / (2 \* a);  
 x2 = (-b - Math.Sqrt(detal)) / (2 \* a);  
 MessageBox.Show("Phương trình có 2 nghiệm phân biệt x1= " + x1 + " x2= " + x2);  
 }  
 }   
 }  
   
 catch (Exception exception)  
 {  
 MessageBox.Show("chưa nhập kìa!!!");  
 throw;  
 }  
 #endregion  
 }  
  
 private void button2\_Click(object sender, EventArgs e)  
 {  
 if (MessageBox.Show("Bạn có chắc chán muốn thoát", "Thông báo", MessageBoxButtons.**YesNo**,  
 MessageBoxIcon.**Question**) == DialogResult.**Yes**)  
 {  
 Close();  
 }  
 }  
 }  
}

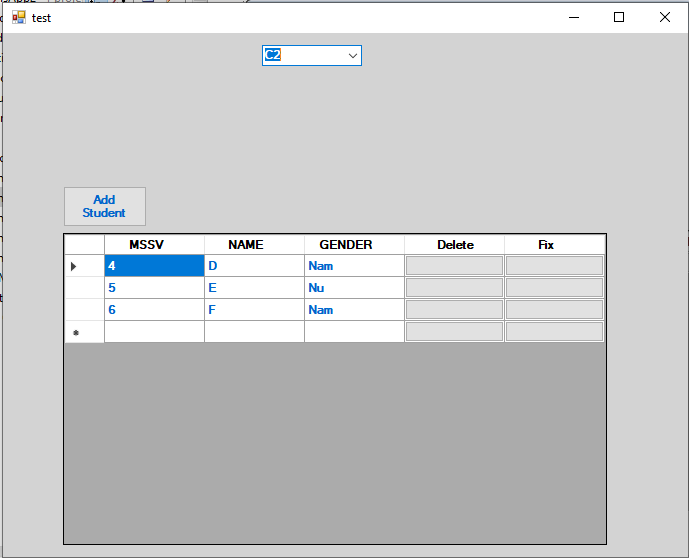


## Quản lý sinh viên

* Danh sách thành phần:
* Combo box: Dùng để chọn lớp.
* DataGridView: Hiện thông tin sinh viên qua 3 cột TextBox (MSSV, Tên, Giới Tính). Cùng với 2 cột Button (Xóa,Sửa).
* Button: Dùng để thêm sinh viên mới.



**BƯỚC 1: Khi mình bấm chọn một lớp trên Combo Box thì DataGridView sẽ hiện thông tin sinh viên trong lớp đó.**



Ta có phương thức load Student dùng để làm cho DataGridView hiện thông tin sinh viên dựa theo ComboBox.

public void loadStudent()

{

dataGridView1.Rows.Clear(); Dòng này dùng để xóa toàn bộ dòng đang hiện trên DataGridView.

String className = comboBox1.SelectedItem.ToString(); Dòng này để xác định xem giá trị hiện tại của Combo Box là gì.

foreach (AClass aClass in sm.AllClassList) { Duyệt từng lớp học trong danh sách các lớp học

if (aClass.ClassName.Equals(className)) { Nếu tên lớp mà trùng với tên lớp được chọn trong Combo Box.

foreach (Student student in aClass.StudentList) { Duyệt qua danh sách SV của lớp dataGridView1.Rows.Add(student.Mssv, student.Name, student.Gender); Thêm thông tin sinh viên vào DataGridView . Mỗi sinh viên 1 hàng.

}

}

}

}

Sau đó ta sẽ cho phương thức loadStudent() chạy khi mà giá trị của ComboBox thay đổi. Ở đây ta sử dụng event SeclectedIndexChanged của Combo Box.

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

loadStudent(); Chạy phương thức loadStudent bên trên.

}

Các bạn nhớ phải gán nó với Combo Box ở bên phía lớp Designer.cs thì nó mới chạy được :

comboBox1.SelectedIndexChanged += new System.EventHandler(this.comboBox1\_SelectedIndexChanged);

**BƯỚC 2: Xử lý xóa và sửa sinh viên.**

Ở trong DataGridView hiện tại có 2 cột button tên là Delete và Fix. Chúng ta sẽ sử dụng event CellContentClick của DataGridView. Event này sẽ kích hoạt khi một ô nhất định trong DataGridView được Click vào.

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

Cách mà event này hoạt động là như một vòng for duyệt qua tất cả Dòng và Cột trong DataGridView.

if (e.ColumnIndex == 4) Nếu như cột là cột thứ 4 (Ở đây là cột Fix) thì tất cả những ô (Cell) mà nằm dưới cột thứ 4 này khi bị click vào sẽ thực hiện cách lệnh dưới đây.

{

fix = new Fix(dataGridView1.Rows[e.RowIndex], sm); Khởi tạo một form dùng để sửa thông tin sinh viên và dữ liệu nhận vào là những ô trong dòng tương ứng (dataGridView1.Rows[e.RowIndex]) của button và chương trình quản lý sinh viên (sm)này

fix.FormClosed += Fix\_FormClosed; Dòng này mình thêm event vào form sửa thông tin sinh viên nhằm để refresh lại DataGridView khi Sửa Thông Tin SV thành công và Form sửa thông tin sinh viên đóng lại.

fix.Show(); Chạy form.

}

if (e.ColumnIndex == 3) Còn nếu cột mà là cột thứ 3 (Cột Delete) thì tất cả những ô (Cell) dưới cột này khi click vào sẽ thực hiện các lệnh bên dưới.

{

String mssv = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString(); Dòng này là lấy mssv từ dòng tương ứng với từng button

DialogResult result = MessageBox.Show("Are you sure you want to delete student " + mssv, "",

MessageBoxButtons.YesNo, MessageBoxIcon.Question); Hiện thông báo rằng bạn có chắc muốn xóa học sinh này.

if (result == DialogResult.Yes) Nếu người dùng bấm vào YES

{

sm.DeleteStudent(mssv); Xóa sinh viên

loadStudent(); Refresh lại danh sách sinh viên trong lớp

}

}

}

Phương thức Fix\_FormClosed nhằm dùng để Refesh thông tin sinh viên trên DataGridView khi Form sửa thông tin sinh viên đóng lại.

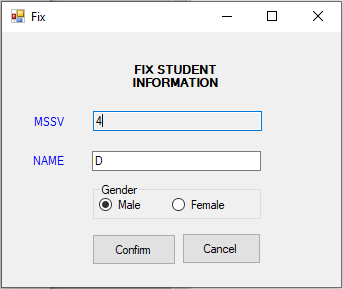
private void Fix\_FormClosed(object sender, FormClosedEventArgs e)

{

loadStudent();

}

Form Fix.cs để sửa thông tin sinh viên. Text Box MSSV thì mình dùng thuộc tính Read Only để không ai có thể thay đổi cái này. Và phài nhớ dùng Group box để nhóm các Radio Button lại.



Contructor của Form Fix.cs:

public Fix(DataGridViewRow information, SchoolManagement sm)

{

InitializeComponent();

this.sm = sm; Chương trình quản lý Sinh viên

this.information = information; Thông tin sinh viên được gửi lên từ DataGridView

textBox1.Text = this.information.Cells["MSSV"].Value.ToString(); Đưa thông tin từ cột MSSV vào textBox1.

textBox2.Text = this.information.Cells["NAME"].Value.ToString(); Đưa thông tin từ cột NAME vào textBox2

if (this.information.Cells["GENDER"].Value.ToString().Equals("Nam")). Kiểm tra thông tin của cột GENDER. Nếu là nam thì radioButton1 sẽ được đánh dấu còn nếu là nữ thì radioButton2 sẽ được đánh dấu.

{

radioButton1.Checked = true;

}

else

{

radioButton2.Checked = true;

}

}

Phương thức validateStudent nhằm mục đích kiểm tra dữ liệu xem coi có hợp lệ hay không trước khi thay đổi thông tin Sinh viên.

public bool validateStudent()

{

if (String.IsNullOrEmpty(textBox2.Text) || String.IsNullOrWhiteSpace(textBox2.Text))

{ Nếu tên sinh viên nhập trong textBox2 là null hoặc không nhập gì cả hoặc chỉ toàn khoảng trắng thì không hợp lệ.

return false;

}

if (!radioButton1.Checked && !radioButton2.Checked)

{ Nếu như cả 2 radioButton ở phần giới tính đều không được đánh đấu thì là không hợp lệ.

return false;

}

return true;

}

Cuối cùng ta thêm sự kiện Click vào cho nút Confirm (Xác Nhận).

private void button1\_Click(object sender, EventArgs e)

{

if (validateStudent()) Kiểm tra xem dữ liệu hợp lệ không

{

String gender = "";

if (radioButton1.Checked)

{

gender = "Nam";

}

if (radioButton2.Checked)

{

gender = "Nu";

}

sm.ChangeStudentInformation(textBox1.Text, textBox2.Text, gender); Đổi thông tin sinh viên

MessageBox.Show("Change Student Information Success"); Hiện thông báo đổi thông tin thành công.

this.Close(); Đóng Form

}

else

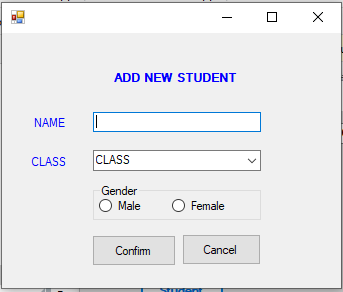
{

MessageBox.Show("Data Input is Invalid"); Nếu dữ liệu không hợp lệ thì thông báo cho người dùng.

}

}

**Bước 3: Nút thêm sinh viên. Nút này chức năng khá tương tự sửa thông tin sinh viên nên mình không nói ở đây.**

****