



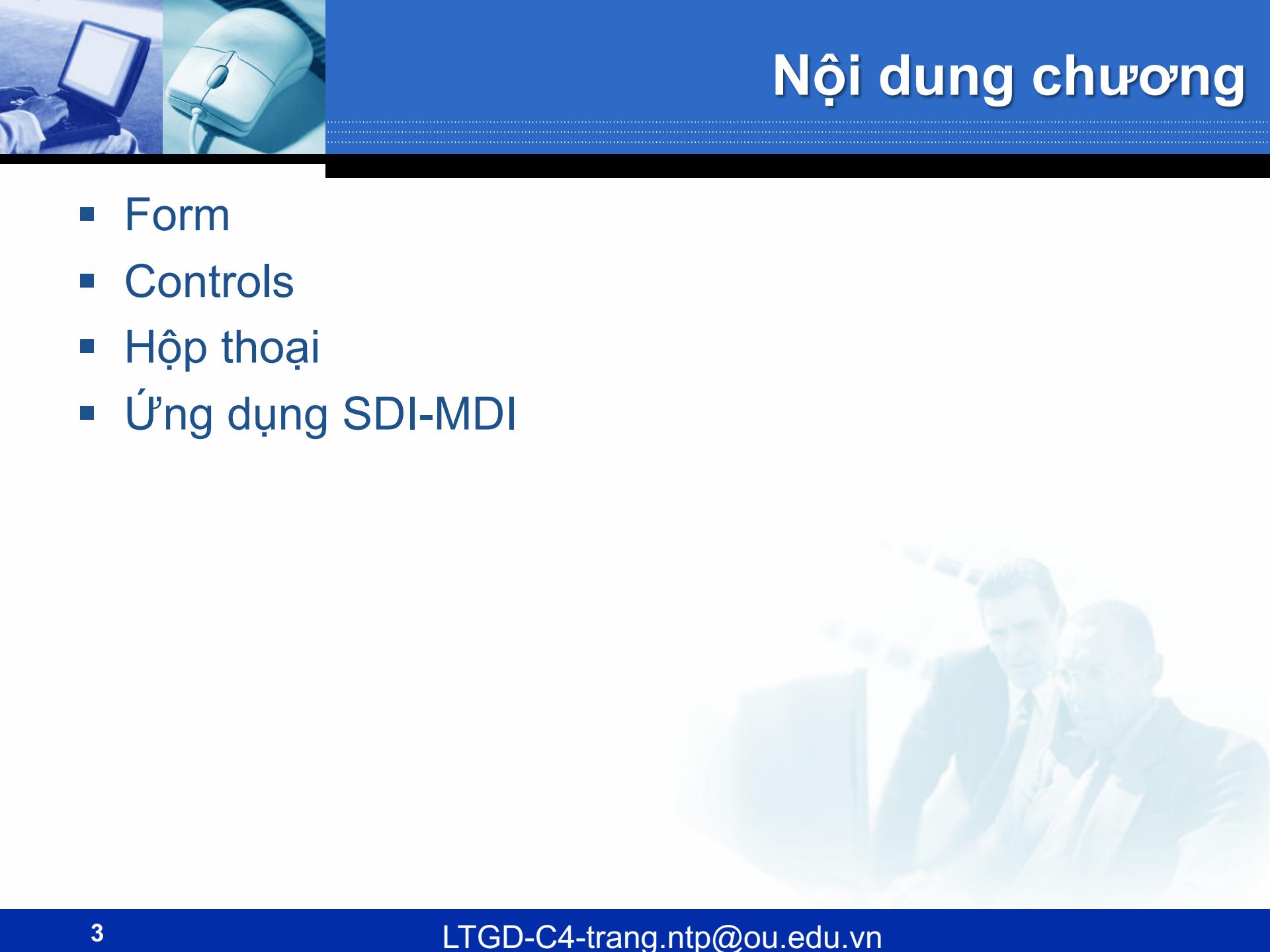
TRƯỜNG ĐẠI HỌC MỞ
TP HỒ CHÍ MINH

BÀI GIẢNG
LẬP TRÌNH GIAO DIỆN



CHƯƠNG 4:
WINDOWS FORM VÀ CÁC
CONTROLS

- Sử dụng đúng và hợp lý các loại giao diện ứng dụng
- Sử dụng thành thạo các thuộc tính của Form và control để xây dựng các ứng dụng đáp ứng yêu cầu ở mức độ từ cơ bản đến nâng cao
- Nắm bắt và xử lý các sự kiện trong Windows Form và các control một cách hợp lý
- Thao tác thành thạo khi chuyển form và truyền dữ liệu giữa các form
- Xây dựng và xử lý thành thạo ứng dụng dạng MDI

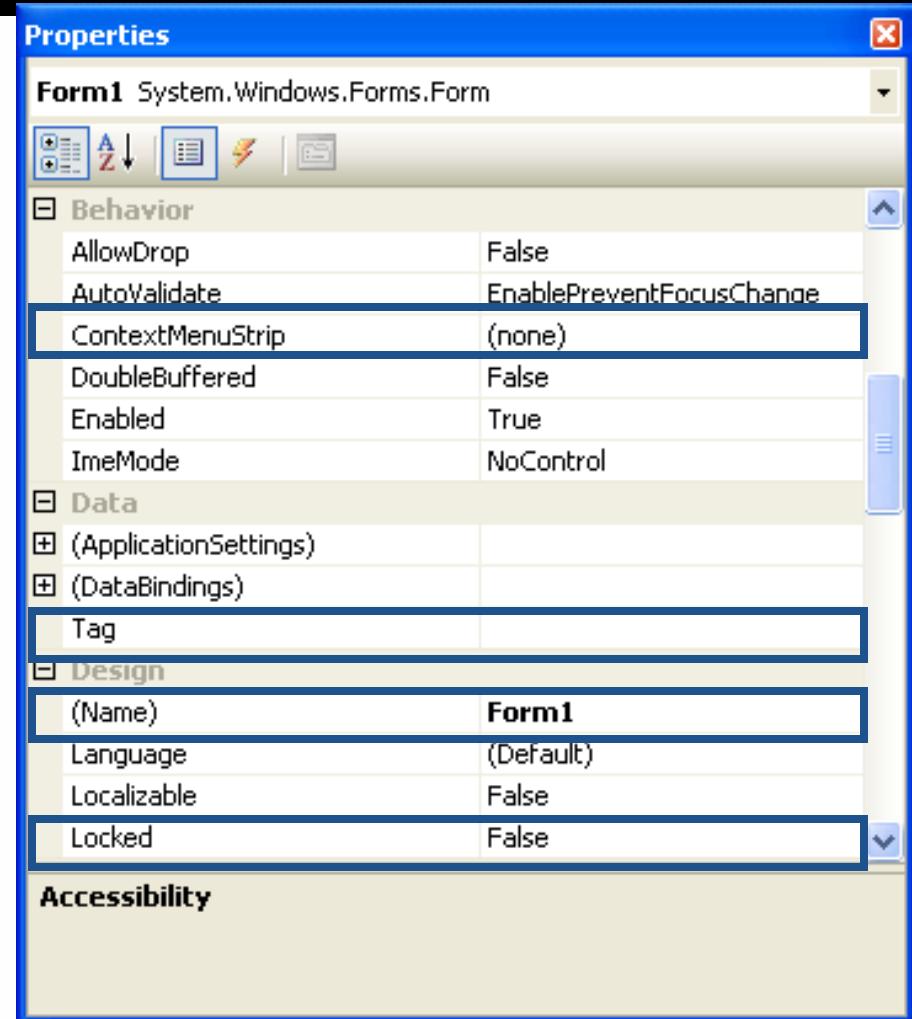
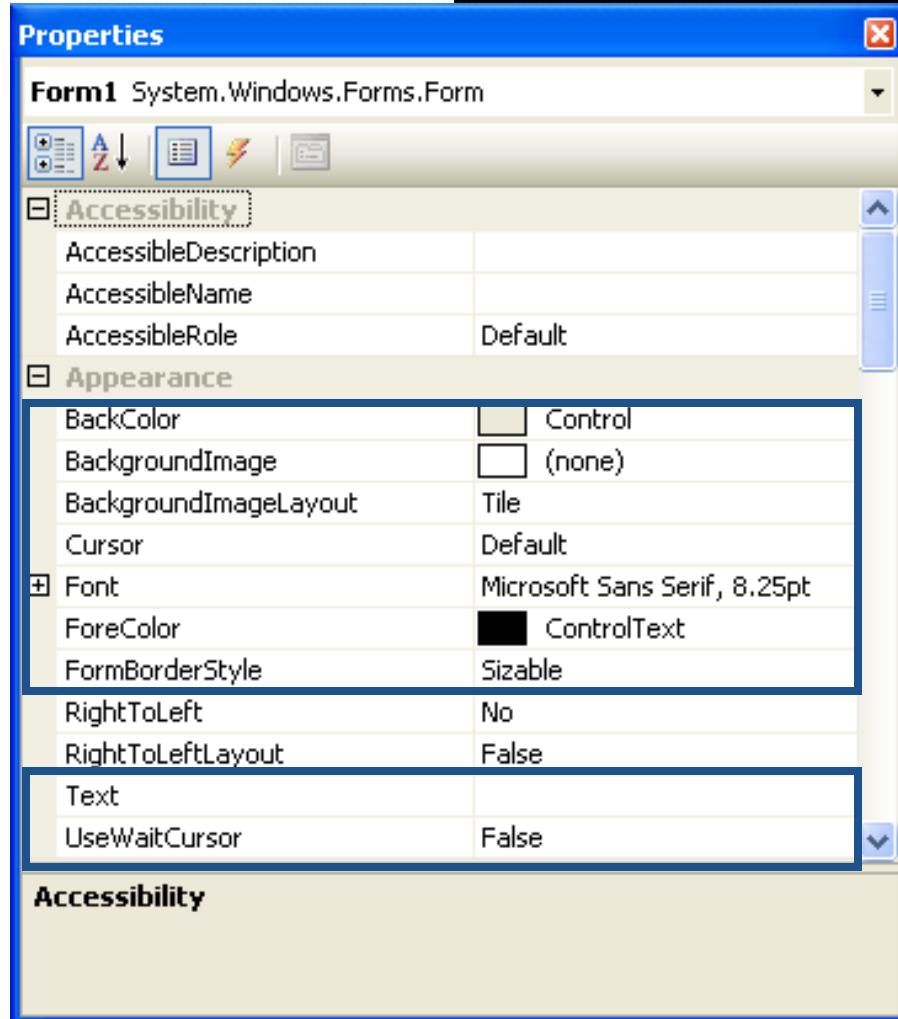


Nội dung chương

- Form
- Controls
- Hộp thoại
- Ứng dụng SDI-MDI

- **Form** thể hiện một cửa sổ (window) hay một dialog box tạo nên giao diện của ứng dụng
- Thông thường tạo custom form bằng cách thừa kế từ Form
- Namespace
 - System.Windows.Form

Properties





Properties

Properties

Form1 System.Windows.Forms.Form

	CausesValidation	True
	AutoSizeMode	Font
	AutoScroll	False
	AutoScrollMargin	0, 0
	AutoScrollMinSize	0, 0
	AutoSize	False
	AutoSizeMode	GrowOnly
	Location	0, 0
	MaximumSize	0, 0
	MinimumSize	0, 0
	Padding	0, 0, 0, 0
	Size	243, 187
	StartPosition	WindowsDefaultLocation
	WindowState	Normal
Accessibility		

Properties

Form1 System.Windows.Forms.Form

	AcceptButton	(none)
	CancelButton	(none)
	KeyPreview	False
	ControlBox	True
	HelpButton	False
	Icon	(Icon)
	IsMdiContainer	False
	MainMenuStrip	(none)
	MaximizeBox	True
	MinimizeBox	True
	Opacity	100%
	ShowIcon	True
	ShowInTaskbar	True
	SizeGripStyle	Auto
	TopMost	False
	TransparencyKey	<input type="checkbox"/>
Accessibility		

▪ Các thuộc tính của Form

- Name: Tên Form.
- Text: Chuỗi hiển thị trên thanh tiêu đề.
- ShowIcon: true/false – hiển thị/không hiển thị icon ở bên trái thanh tiêu đề.
- ShowInTaskBar: true/false – hiển thị/không hiển thị biểu tượng của form trên thanh Taskbar khi form được thực thi.
- Icon: tên tập tin *.ico làm biểu tượng trên thanh tiêu đề của form.
- BackColor: màu nền của form.
- ForeColor: màu của các chuỗi trên các control của form.
- StartPosition: vị trí hiển thị form.
- Opacity: độ rõ của form, mặc định là 100%.

- **Các thuộc tính của Form (tt)**

- WindowsStates: trạng thái của form khi thực thi:
 - Minimized (thu nhỏ).
 - Maximized (phóng to).
 - Normal (trạng thái như thiết kế).
- isMdiContainer: được sử dụng trong ứng dụng MDI.
 - true: form được chọn là MDI form (form cha).
 - false: form bình thường.
- TopMost:
 - true: form nằm chồng lên trên các cửa sổ khác.
 - false: form bình thường.
- FormBorderStyle: kiểu đường viền của form.
- MainMenuStrip: control ToolStrip gắn trên form

▪ Một số phương thức của Form

- Close (): đóng form.
- Hide (): ẩn form.
- Show (): Hiển thị form dạng modeless-dialog (khi form hiển thị, người sử dụng vẫn có thể thao tác được với các thành phần khác trong cùng một ứng dụng).
- ShowDialog (): Hiển thị form dạng modal-dialog (khi form hiển thị, người sử dụng không thể thao tác được với các thành phần khác trong cùng một ứng dụng cho đến khi đóng form).

Events

Properties

Form1 System.Windows.Forms.Form

Action

- Click
- DoubleClick
- MouseCaptureChanged
- MouseClick
- MouseDoubleClick
- ResizeBegin
- ResizeEnd
- Scroll

Appearance

- Paint

Behavior

- ChangeUICues
- ControlAdded
- ControlRemoved
- FormClosed
- FormClosing
- HelpButtonClicked
- HelpRequested
- ImeModeChanged
- InputLanguageChanged
- InputLanguageChanging
- Load
- QueryAccessibilityHelp
- Shown
- StyleChanged
- SystemColorsChanged

Action

Properties

Form1 System.Windows.Forms.Form

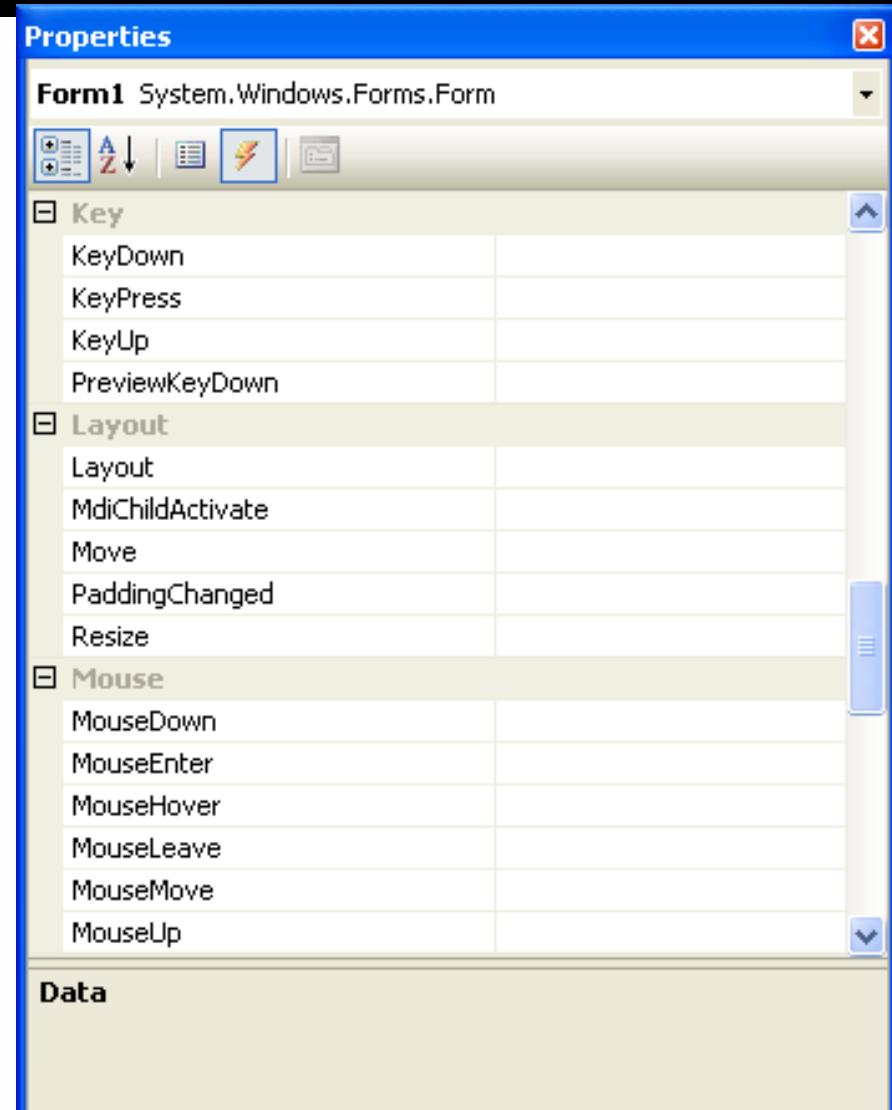
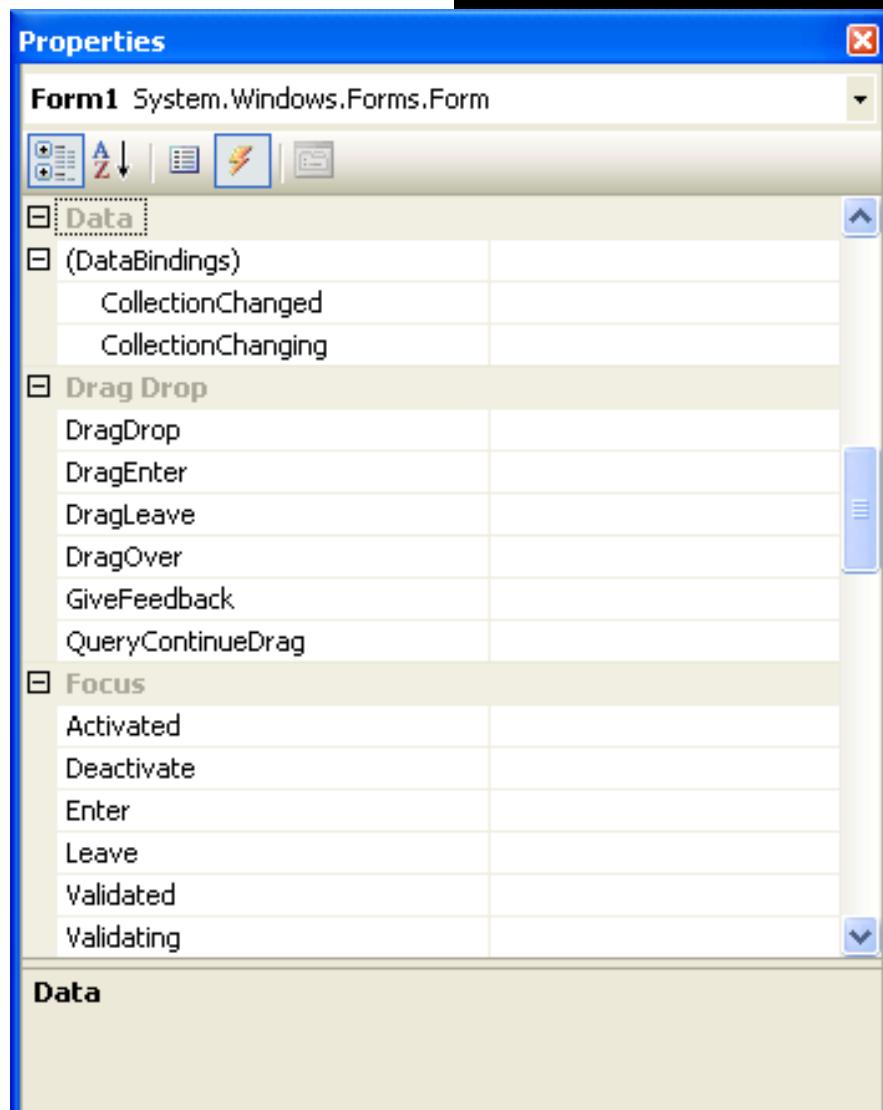
Property Changed

- AutoSizeChanged
- AutoValidateChanged
- BackColorChanged
- BackgroundImageChanged
- BackgroundImageLayoutChanged
- BindingContextChanged
- CausesValidationChanged
- ClientSizeChanged
- ContextMenuStripChanged
- CursorChanged
- DockChanged
- EnabledChanged
- FontChanged
- ForeColorChanged
- LocationChanged
- MaximizedBoundsChanged
- MaximumSizeChanged
- MinimumSizeChanged
- ParentChanged
- RegionChanged
- RightToLeftChanged
- RightToLeftLayoutChanged
- SizeChanged
- TextChanged
- VisibleChanged

Leave
Occurs when the control is no longer the active control of the form.



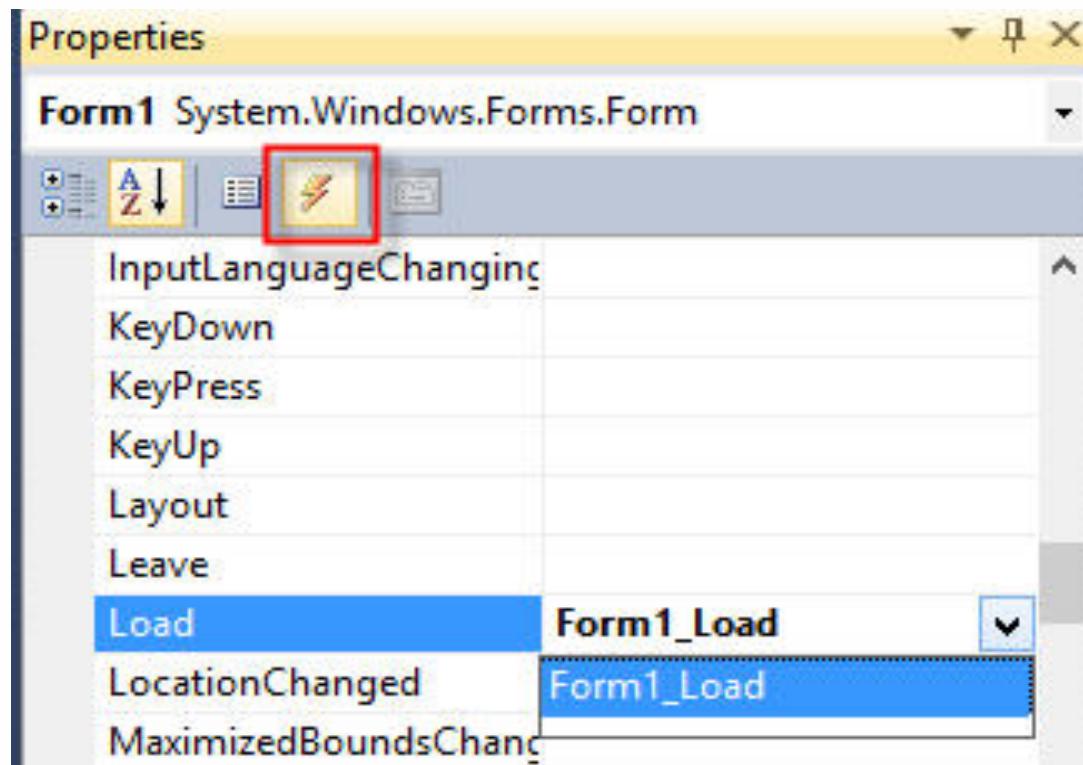
Events



▪ Các sự kiện trên Form

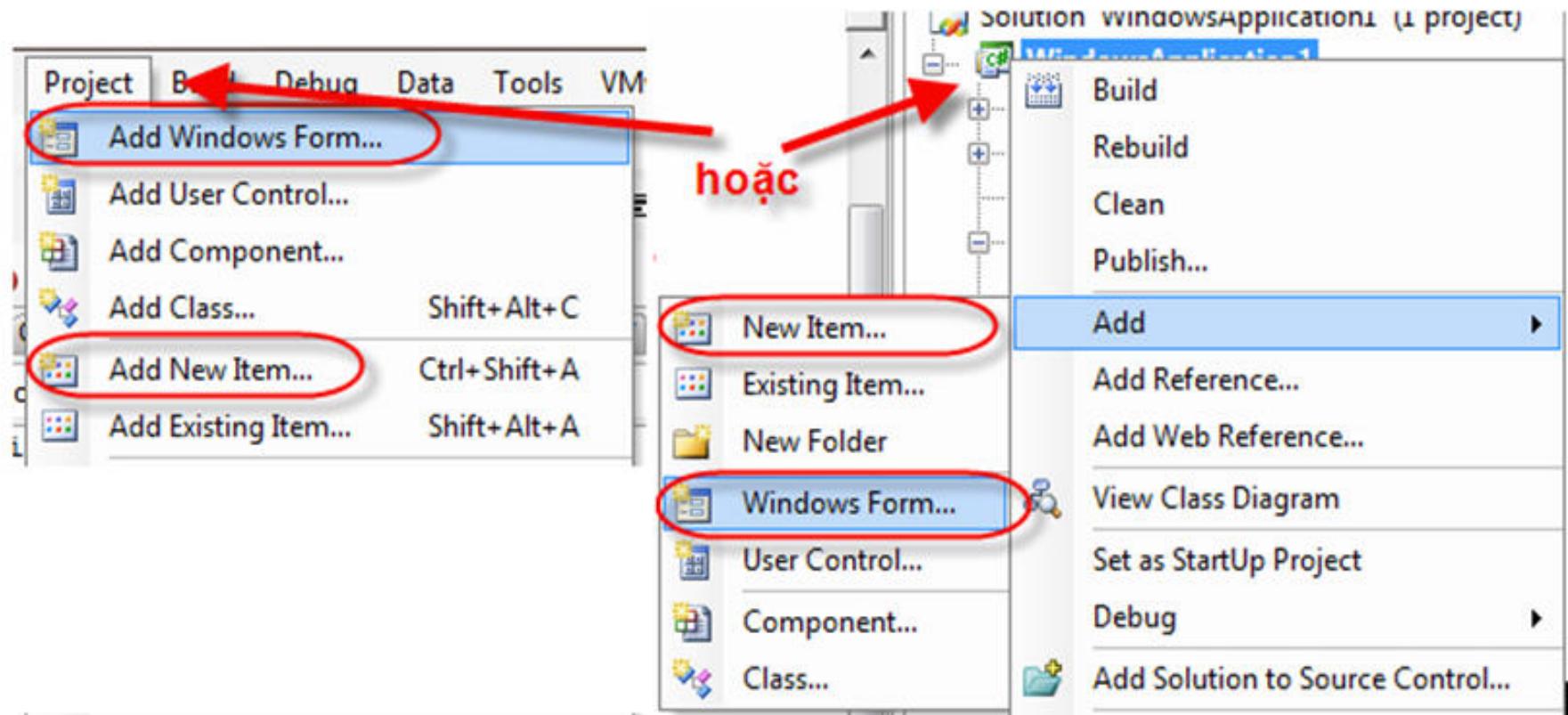
- FormClosed: được gọi tự động khi form đã đóng.
- FormClosing: được gọi tự động khi form đang đóng.
- Click: được gọi tự động khi click chuột lên form.
- Activated: được gọi tự động khi form được kích hoạt bằng mã lệnh hay do tác động của người sử dụng
- Load: được gọi tự động khi form được nạp, dùng để khởi tạo giá trị các thành phần trong form.
- KeyPress, KeyDown, KeyUp: được gọi tự động khi một phím được nhấn trên form.
- Resize: được gọi tự động khi form bị thay đổi kích thước.

- Cài đặt sự kiện trên Form: trong bảng properties, chọn tab Events



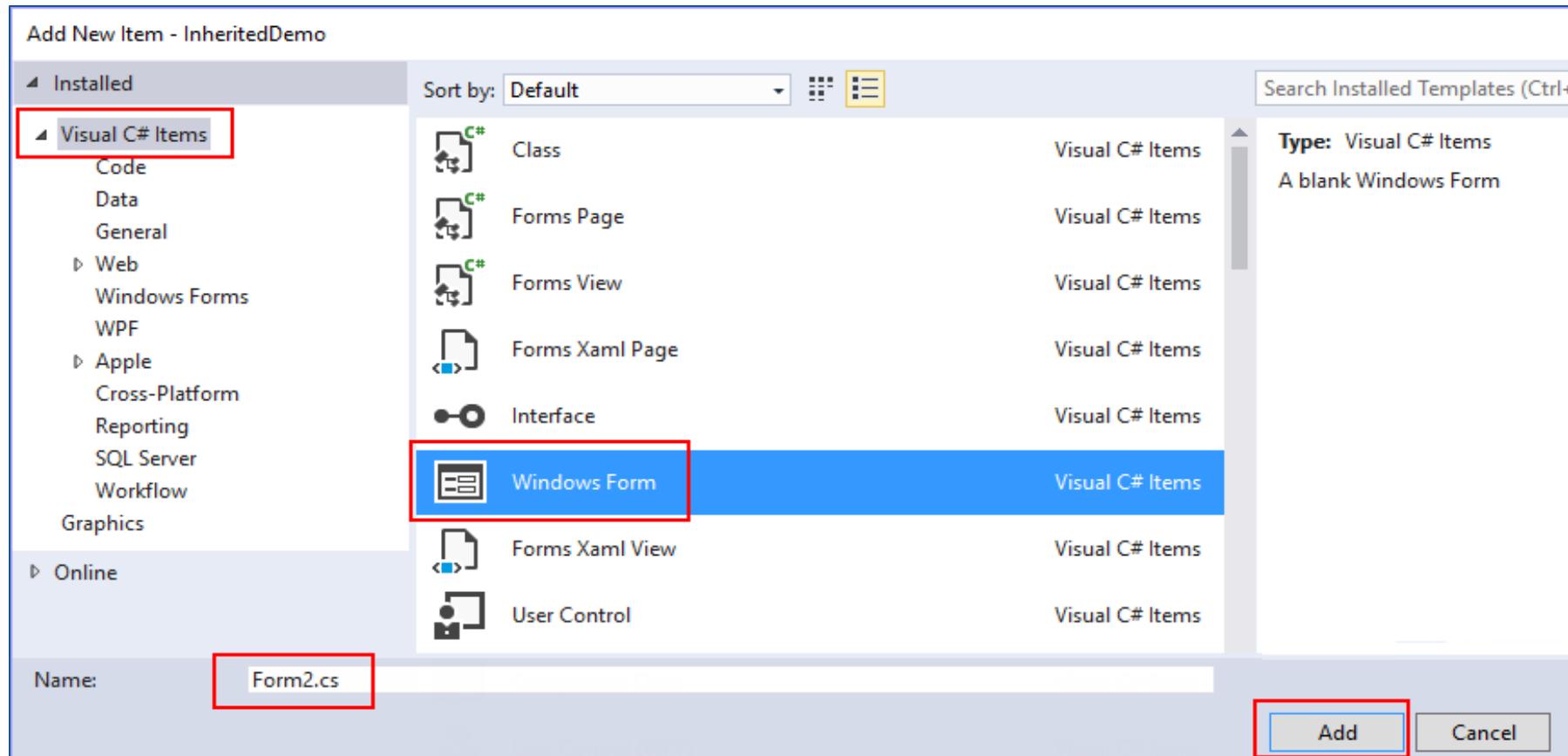


Thêm một form vào project



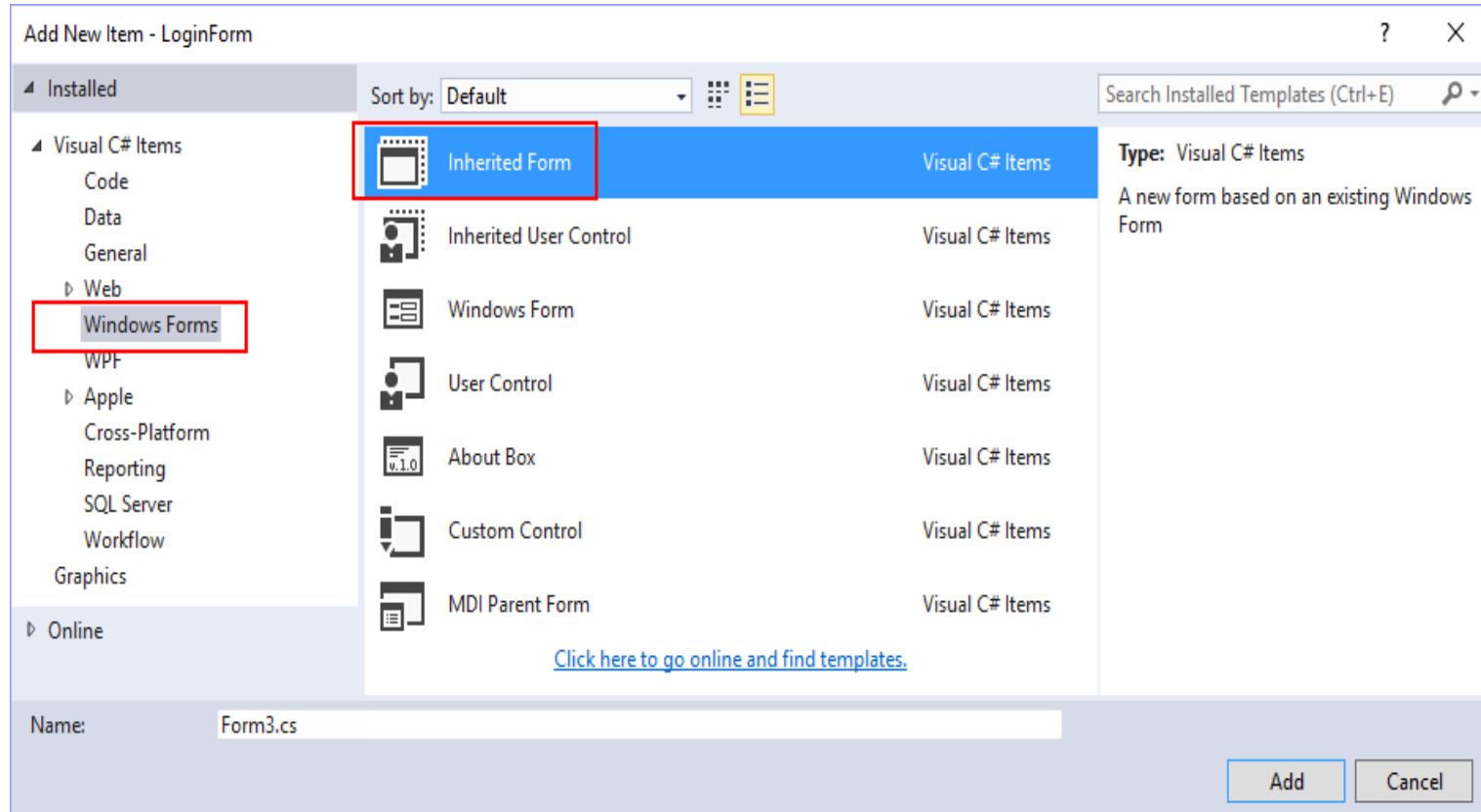


Thêm một form vào project (tt)





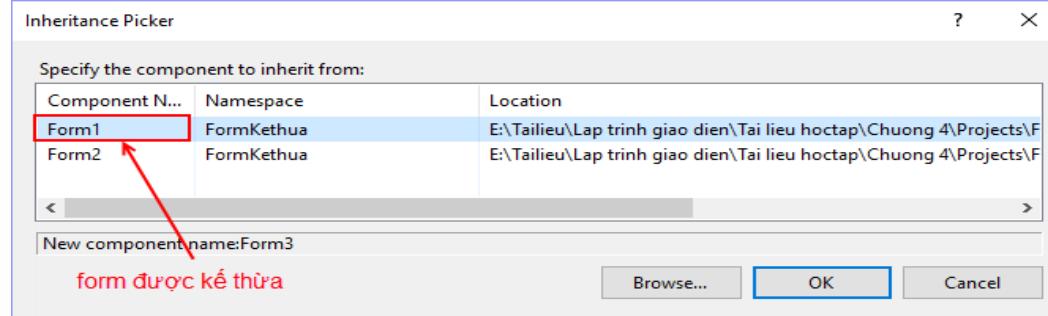
Tạo Form kế thừa (Inherited Form)





Tạo Form kế thừa (tt)

- Chọn Form cha (lớp cơ sở):



- Tạo nhanh form kế thừa:

```
public partial class Form3 : Form1
{
    public Form3()
    {
        InitializeComponent();
    }
}
```



Tạo Form lúc chương trình thực thi

- Tạo đối tượng thuộc class Form
- Xây dựng các thuộc tính cho đối tượng
- Gọi phương thức Show, hoặc ShowDialog

```
Form f = new Form();  
f.Text = "New Form";  
f.BackColor = Color.Red;  
f.StartPosition = FormStartPosition.CenterParent;  
f.ShowDialog();
```



4.2 Controls

- Controls là các thành phần mà ta có thể bổ sung lên form khi thiết kế giao diện cho chương trình.
- Ví dụ: các nút lệnh, các ô cho phép người sử dụng nhập liệu, các loại danh sách lựa chọn, các hộp kiểm, hình ảnh,...
- Các control được tổ chức thành các nhóm nằm trên cửa sổ Toolbox, cho phép người sử dụng kéo thả vào form một cách trực quan.

Control



The image shows a screenshot of the Microsoft Visual Studio .NET IDE interface. It features several floating toolboxes, each listing various Windows Forms controls and components.

- Toolbox (Left):** Shows categories like All Windows Forms, Common Controls, Containers, Menus & Toolbars, Data, Components, Printing, Dialogs, WPF Interoperability, Reporting, General, Pointer, and Shockwave Flash Object.
- Toolbox (Top Left):** Shows All Windows Forms and Common Controls. Under Common Controls, items include Pointer, Button, CheckBox, CheckedListBox, ComboBox, DateTimePicker, Label, LinkLabel, ListBox, ListView, MaskedTextBox, MonthCalendar, NotifyIcon, NumericUpDown, PictureBox, ProgressBar, RadioButton, RichTextBox, TextBox, ToolTip, TreeView, WebBrowser, and Containers.
- Toolbox (Top Middle):** Shows All Windows Forms and Common Controls. Under Common Controls, items include Pointer, FlowLayoutPanel, GroupBox, Panel, SplitContainer, TabControl, TableLayoutPanel, and Containers.
- Toolbox (Bottom Middle):** Shows All Windows Forms and Common Controls. Under Common Controls, items include Pointer, ContextMenuStrip, MenuStrip, StatusStrip, ToolStrip, ToolStripContainer, and Containers.
- Toolbox (Right):** Shows Components. Items listed under Components include Pointer, BackgroundWorker, DirectoryEntry, DirectorySearcher, ErrorProvider, EventLog, FileSystemWatcher, HelpProvider, ImageList, MessageQueue, PerformanceCounter, Process, SerialPort, ServiceController, Timer, and Shockwave Flash Object.
- Toolbox (Far Right):** Shows Printing and Dialogs. Under Printing, items include Pointer, ColorDialog, FolderBrowserDialog, FontDialog, OpenFileDialog, SaveFileDialog, and Shockwave Flash Object.
- Toolbox (Bottom Right):** Shows WPF Interoperability, Reporting, General, Pointer, and Shockwave Flash Object.

- Thêm các controls vào Form:

- Hộp công cụ (Toolbox): cung cấp danh sách các Component liệt kê theo nhóm, cho phép thiết kế giao tiếp với người dùng.

- Hiện ToolBox:

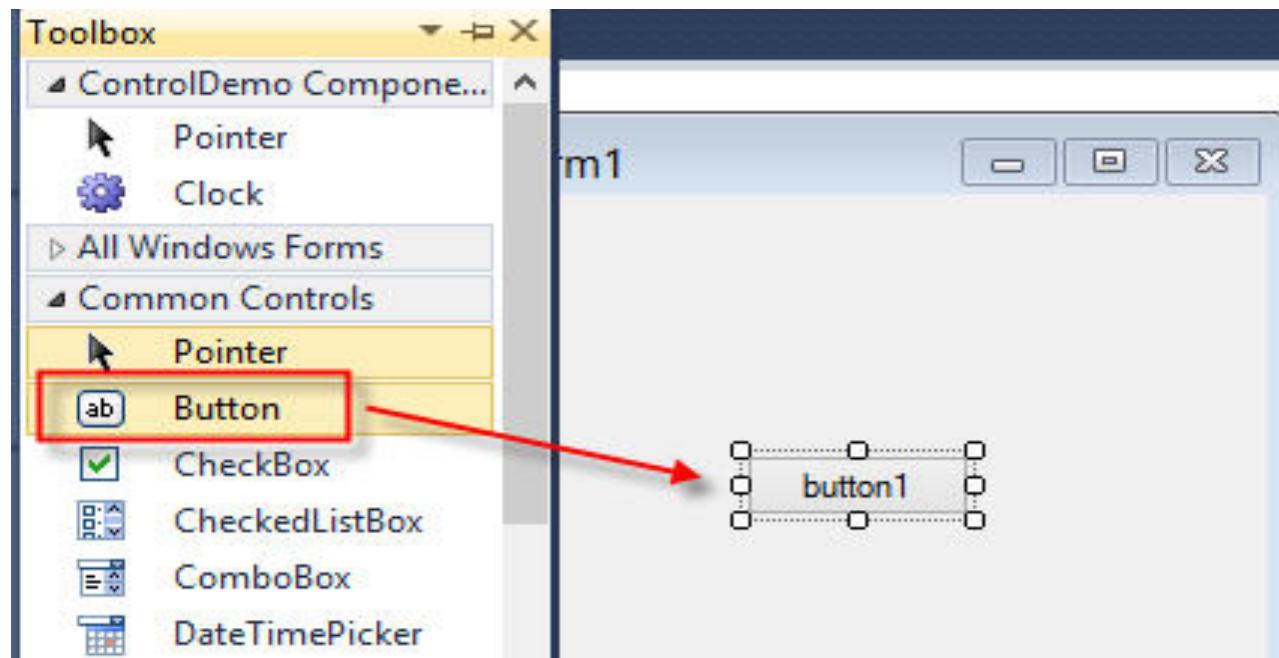
- View, Toolbox

- Chọn biểu tượng trên thanh công cụ

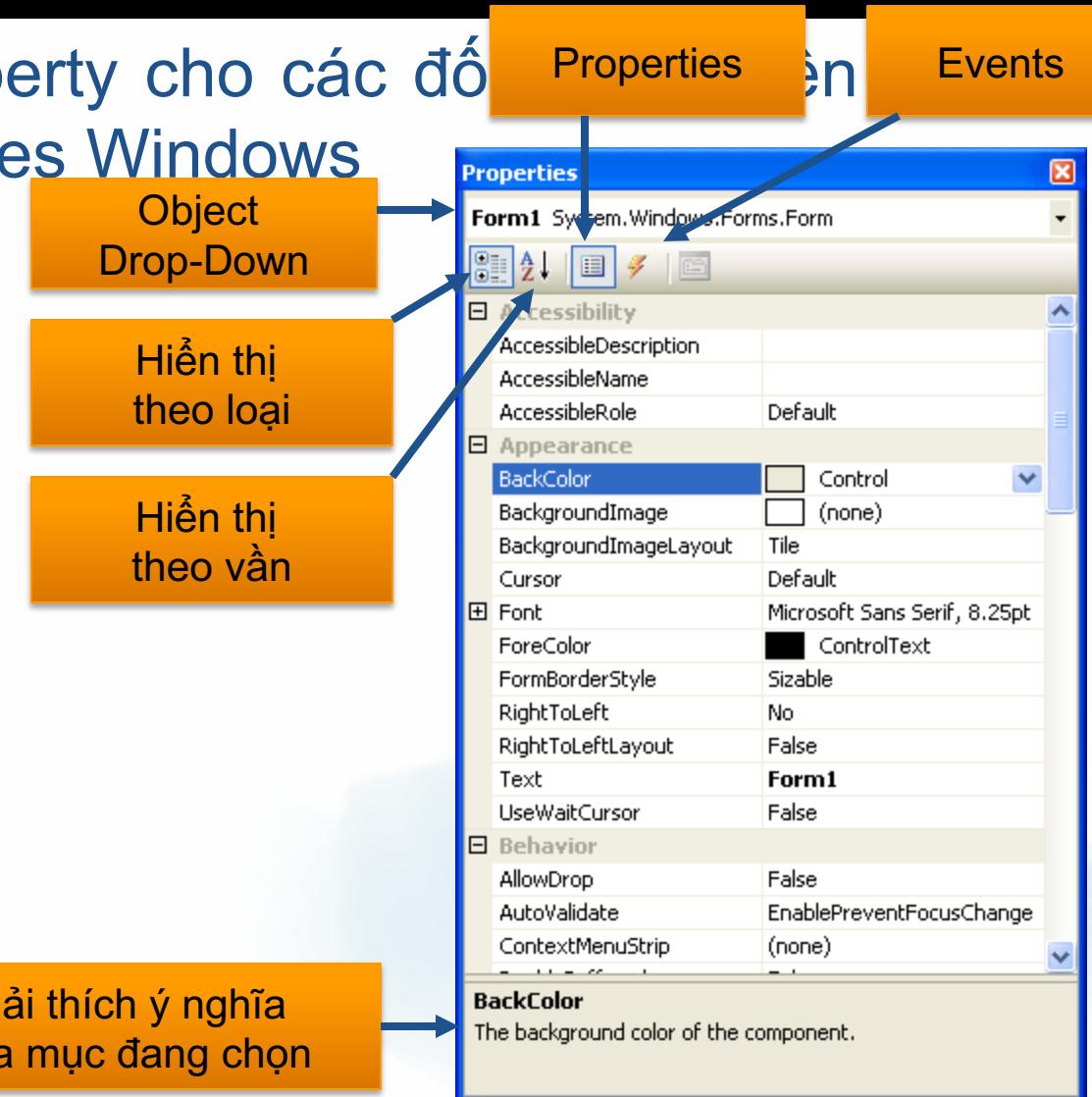
- Ctrl+V



- Thêm các controls vào Form:
 - Kéo thả control từ hộp Toolbox vào Form



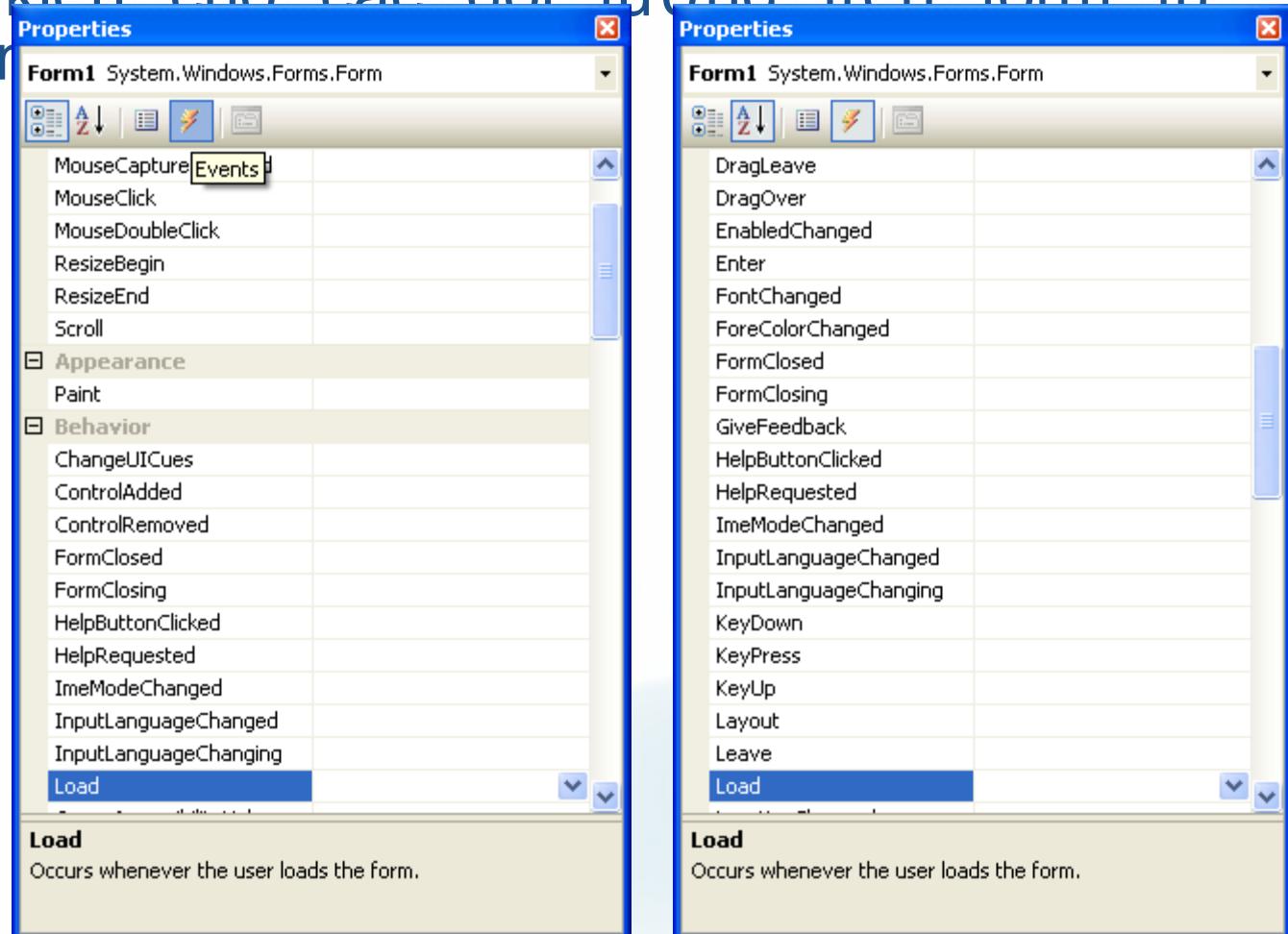
- Thiết lập các Property cho các đối tượng qua Properties Windows



- **Thuộc tính chung của các control:**

- BackColor: Màu nền
- ForeColor: Màu chữ trên control
- Text: Chuỗi hiển thị trên control
- Visible: Ẩn hay hiển thị control
- Name: Tên của control, dùng để truy xuất các thuộc tính của control
- Locked: Khoá không cho di chuyển trên Form
- Enabled: Vô hiệu hoá hay cho phép sử dụng

Bắt các sự kiện cho các đối tượng trên form tùy Properties Window

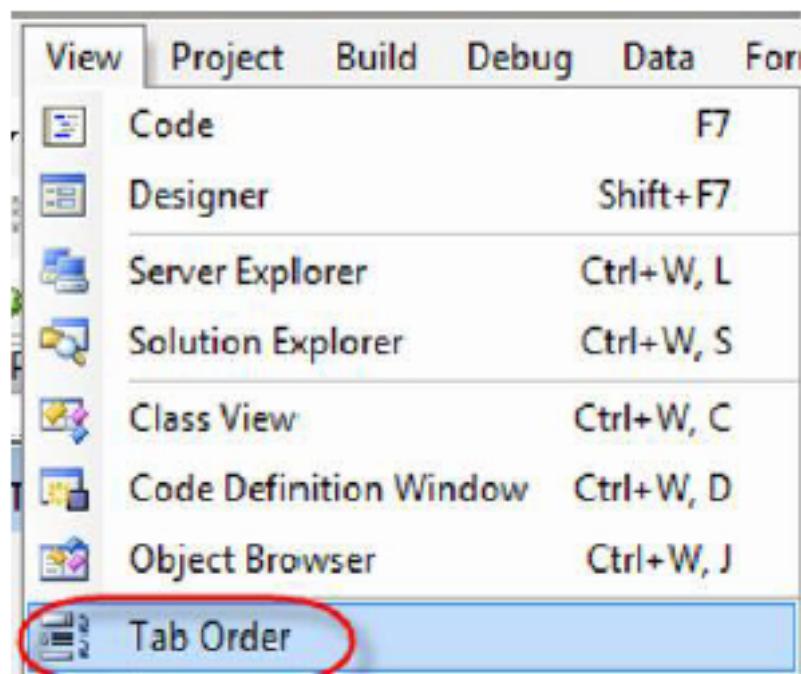


- **Sự kiện chung của các control:**

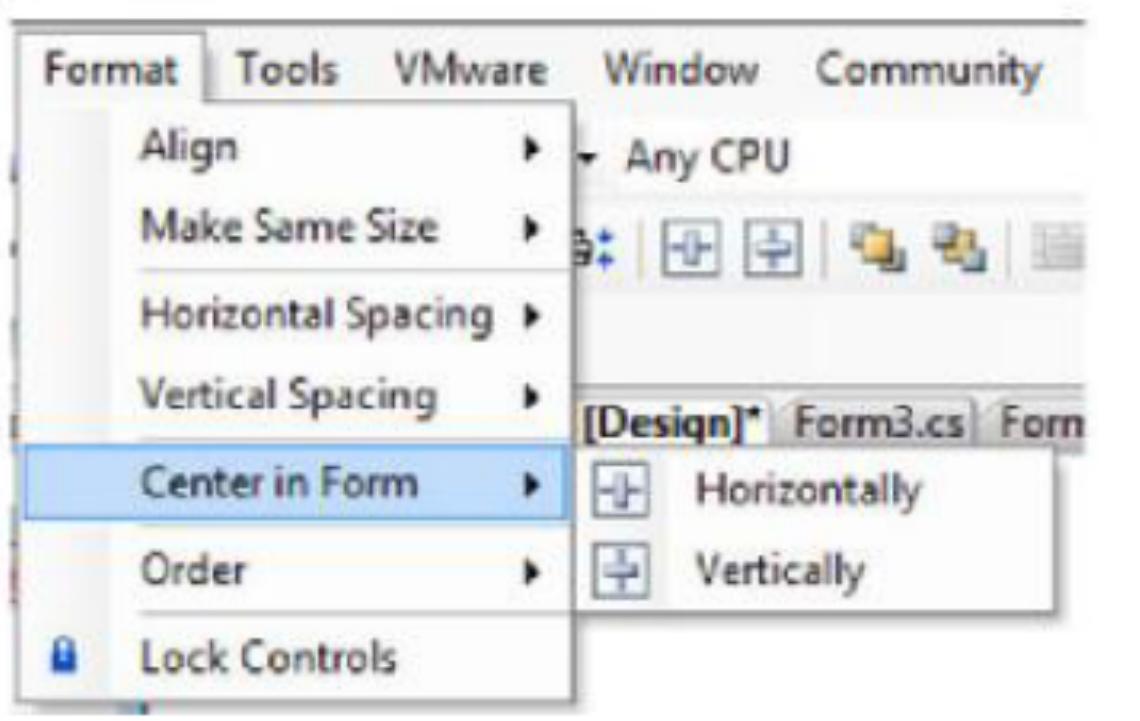
- Click: click chuột lên control.
- MouseMove: di chuyển chuột trên control.
- MouseDown: nhấn chuột trên control.
- MouseUp: nhả chuột sau khi nhấn chuột trên control.
- Move: di chuyển control bằng mã lệnh hay sử dụng chuột.
- SizeChanged: kích thước control được thay đổi bằng mã lệnh hay do người sử dụng.
- Paint: xảy ra khi control được vẽ lại.

- **Xếp thứ tự các control:**
 - View, Tab Order
 - Click chuột lần lượt trên các control để thay đổi số thứ tự (là thứ tự các control trên Form)
- **Sắp xếp các control:**
 - Sử dụng menu Format
 - Sử dụng thanh công cụ
- **Thay đổi thuộc tính các control:**
 - Click chuột phải trên control, chọn Properties
 - Chọn các thuộc tính cần thay đổi trên cửa sổ Properties

Xếp thứ tự các control



Sắp xếp các control



Định vị các control

Sử dụng code thông qua các thuộc tính: Size, location, Top, Left, Width, height

Ví dụ:

```
TextBox1.Location = new Point(100,50);
```

//vị trí điểm góc trên trái của textbox có tọa độ (100,50)

```
TextBox1.Size = new Size(250, 150);
```

// thiết lập kích thước cho textbox với chiều rộng là 250 pixel, chiều cao là 150 pixel

```
TextBox1.Left = 100;
```

```
TextBox1.Top = 50;
```

```
TextBox1.Width = 250;
```

```
TextBox1.Height = 150;
```

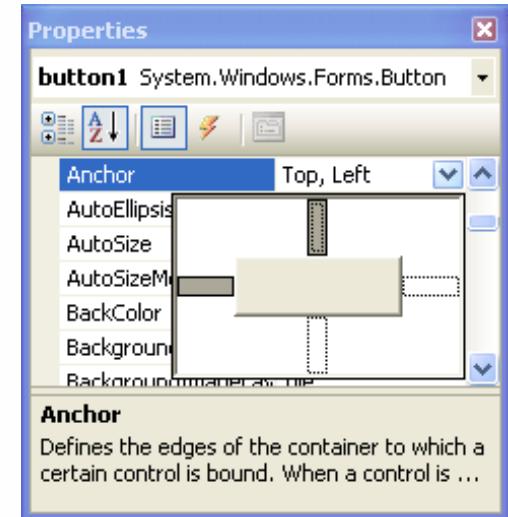




Bố cục các controls trên form

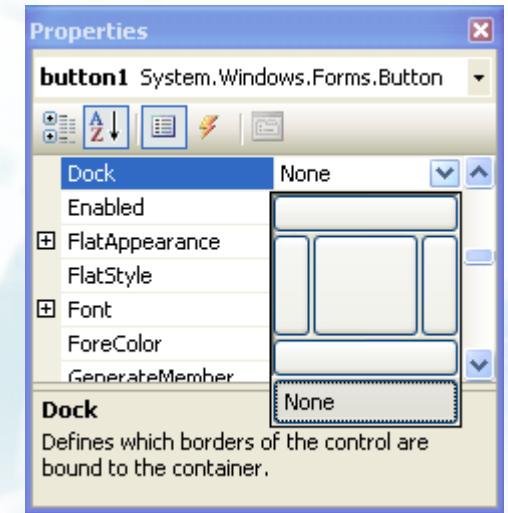
Anchor: Chỉ ra các cạnh của container để biết control sẽ thay đổi kích thước như thế nào khi form thay đổi kích thước của container

- Left, Top, Right, Bottom



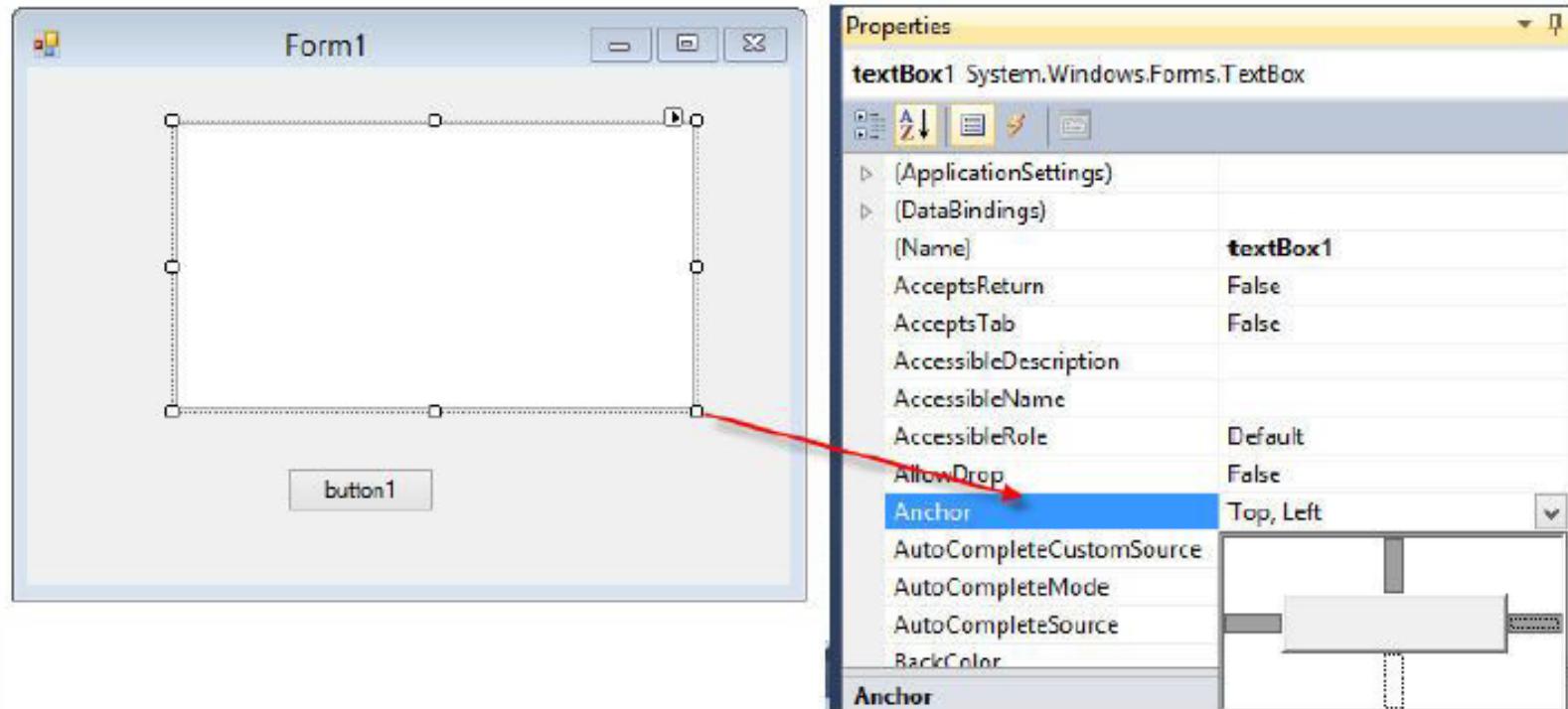
Dock: Chỉ ra các cạnh của control sẽ bám vào container

- Left, Top, Right, Bottom,
- Các cạnh Fill





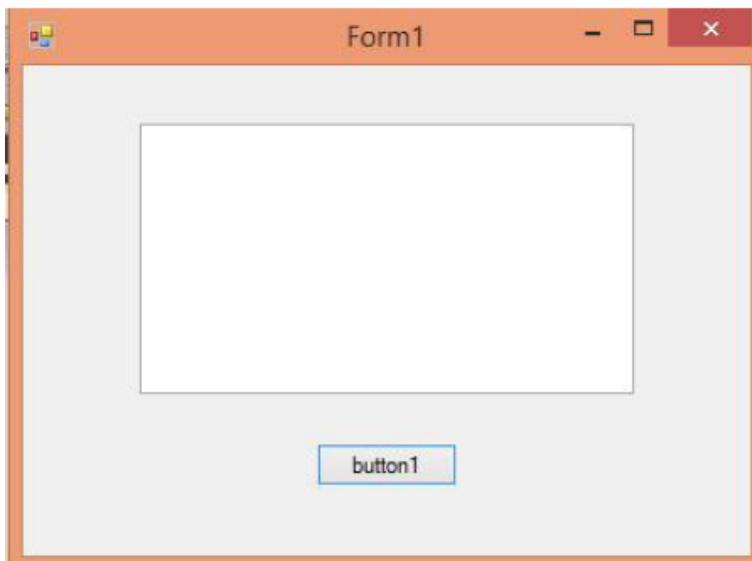
Bố cục các controls trên form



Anchor



Bố cục các controls trên form



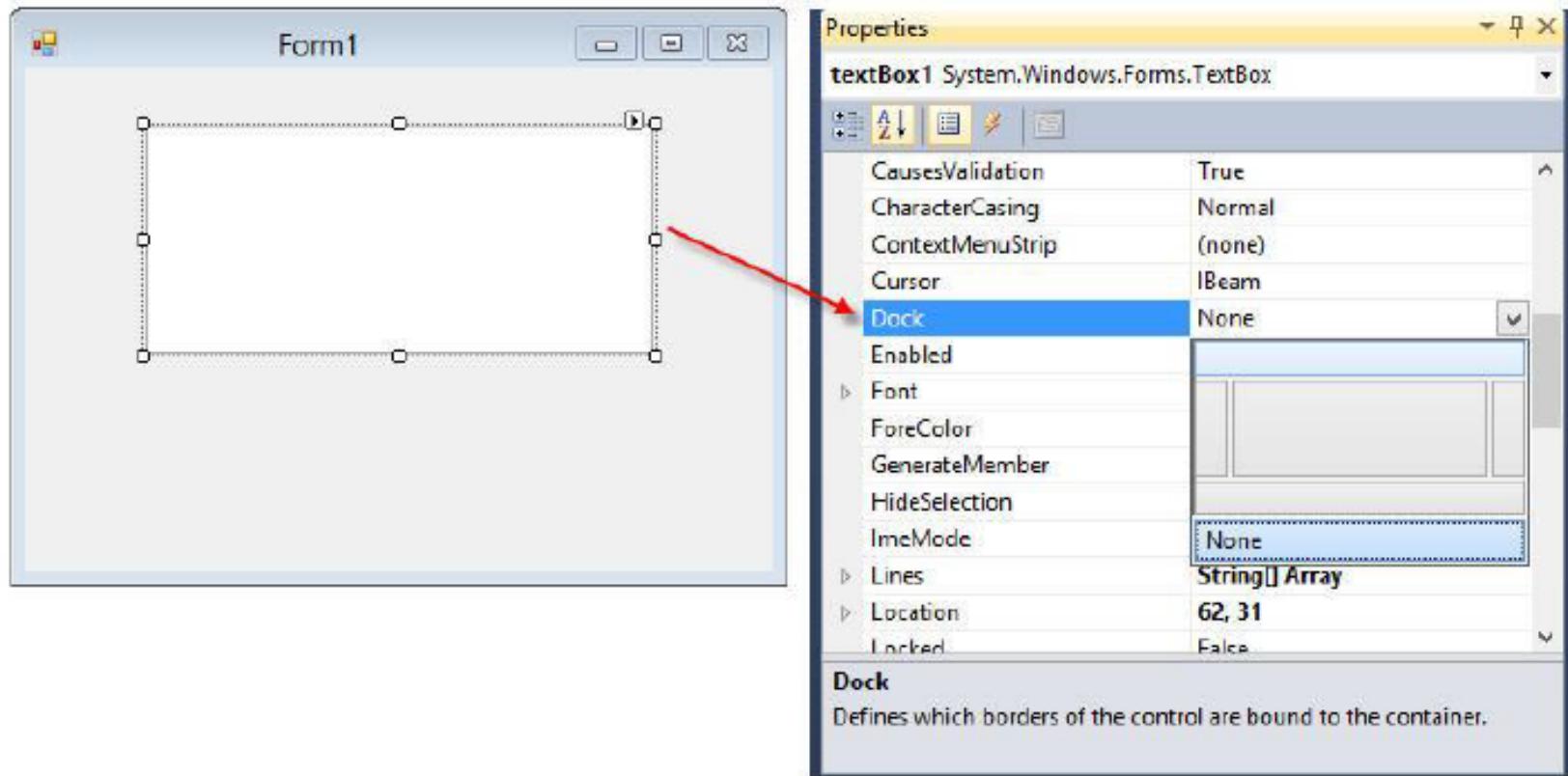
Anchor





Bố cục các controls trên form

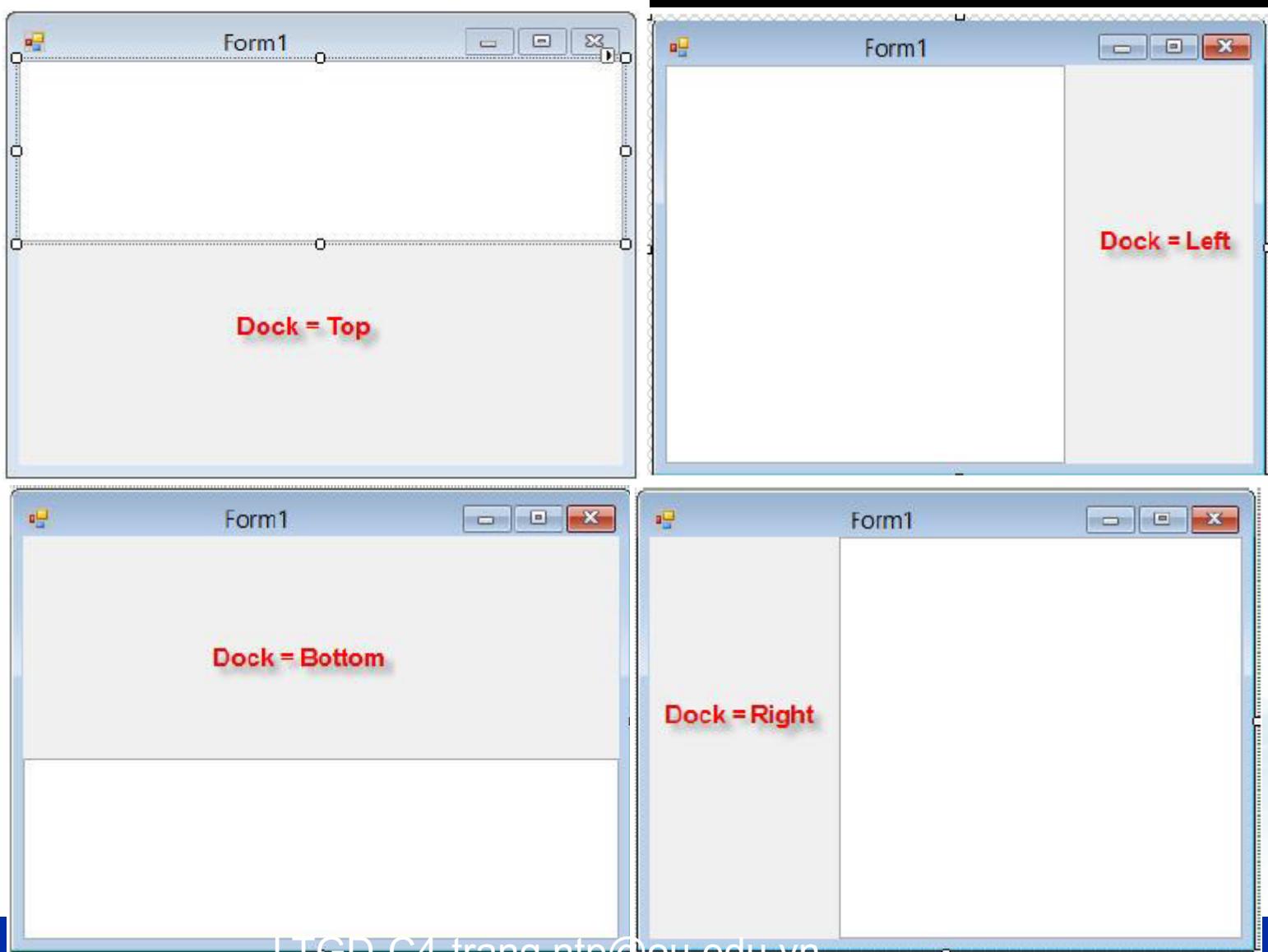
Dock



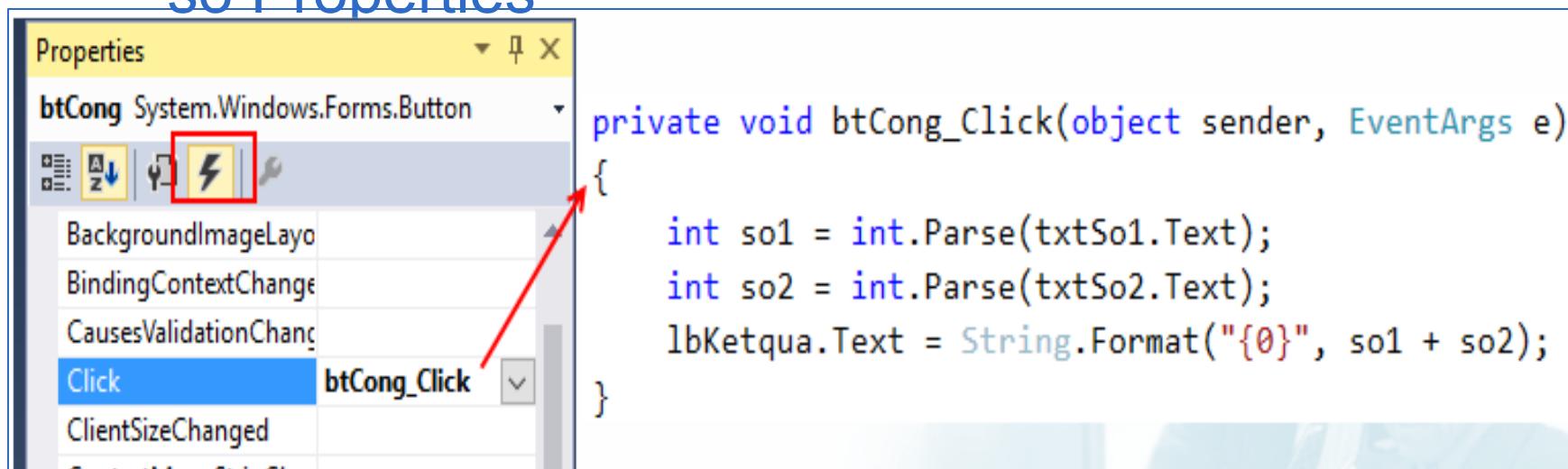


Bố cục các controls trên form

Dock

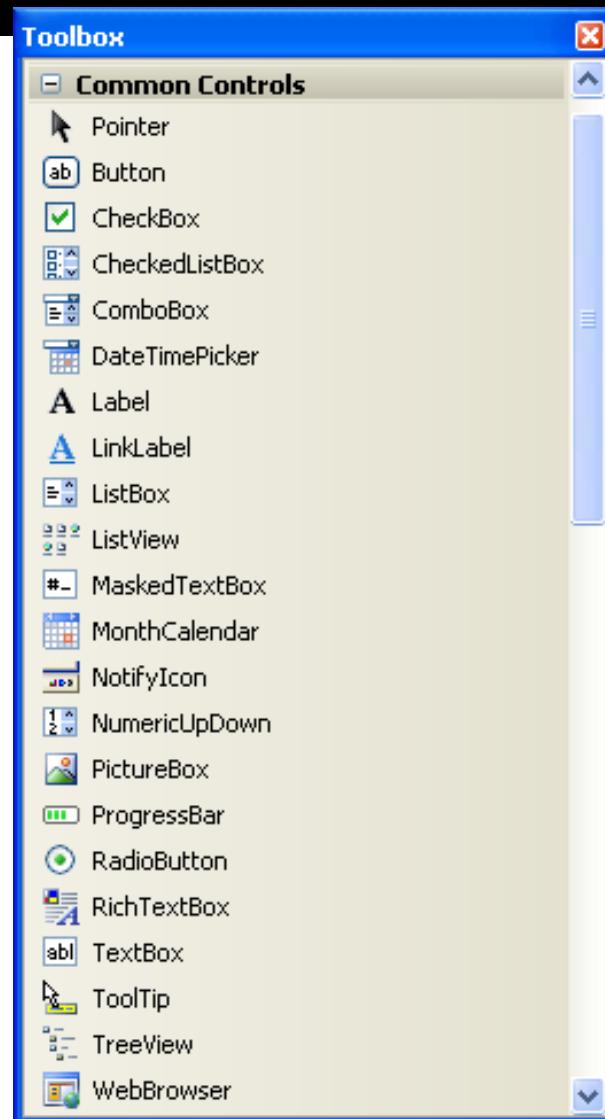


- Xử lý các sự kiện của control:
 - Click chuột phải trên control, chọn Properties
 - Double click lên sự kiện trên tab Event trên cửa sổ Properties

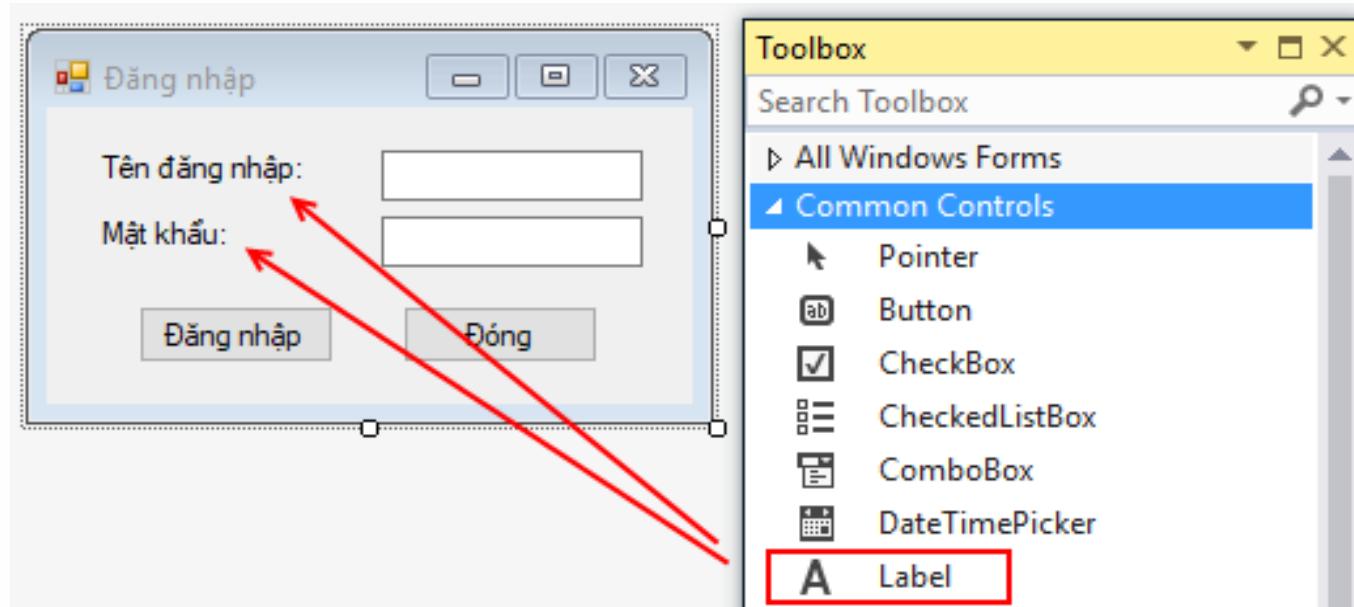




Các control thông dụng

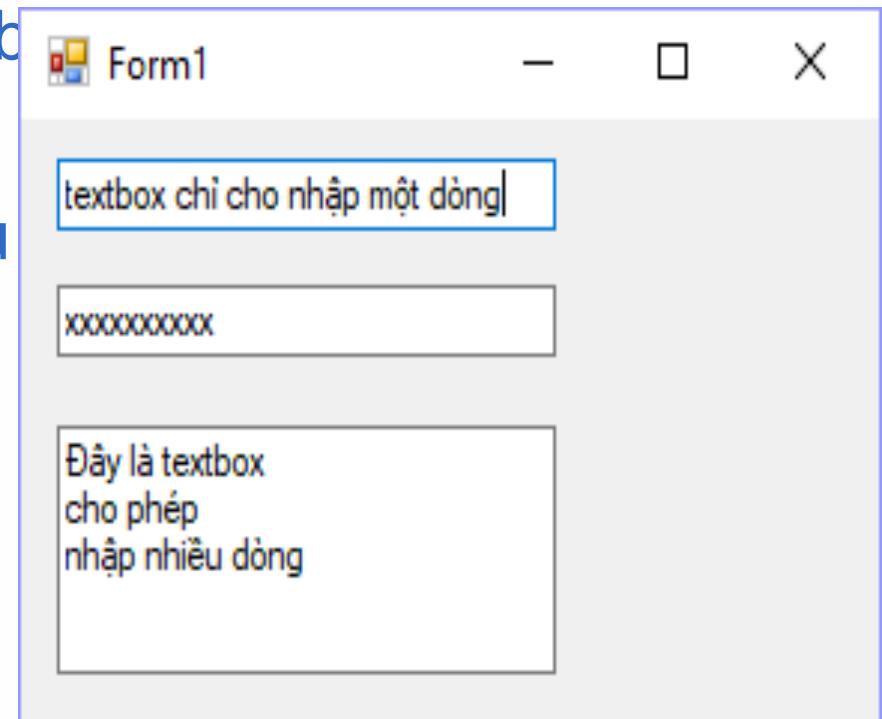


- Trình bày văn bản dạng “tĩnh”, thường được dùng để chú thích cho các control khác hoặc gợi ý cho người sử dụng.



- Các thuộc tính thông dụng:
 - **BorderStyle**: Kiểu đường viền của label.
 - **TextAlign**: Canh chỉnh văn bản trong label.
 - **AutoSize**:
 - true/false - cho / không cho phép label tự động thay đổi kích thước theo độ dài của chuỗi chứa bên trong nó.

- Control được dùng để nhập dữ liệu
- Có ba dạng:
 - nhập một dòng văn bản
 - nhập nhiều dòng
 - nhập dạng mật khẩu



- **Một số thuộc tính:**

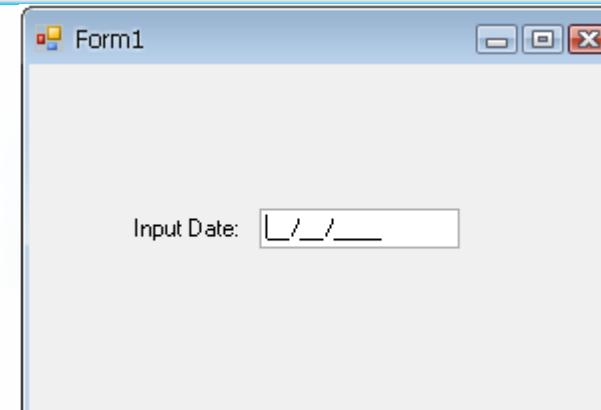
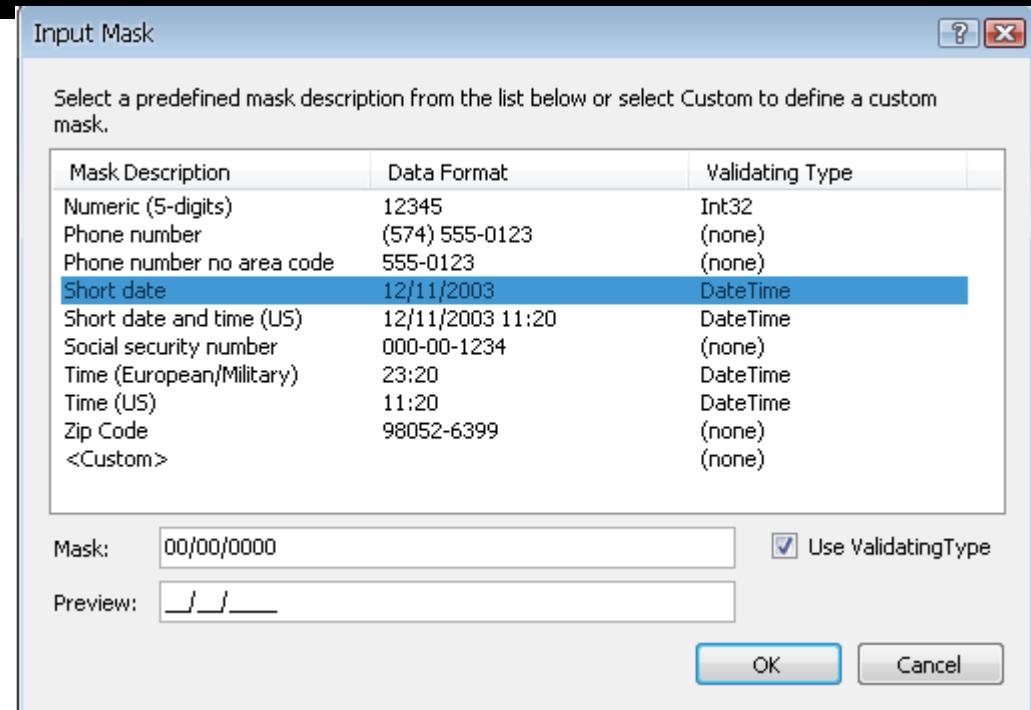
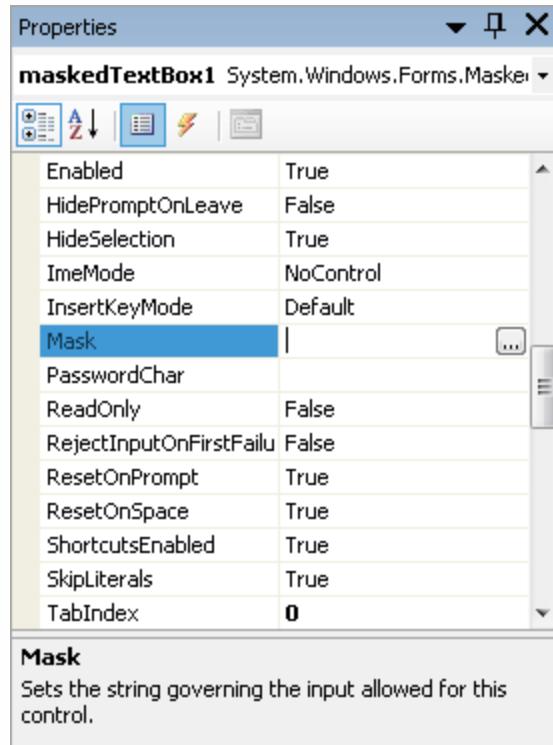
- BorderStyle: kiểu đường viền của textbox.
- CharacterCasing: định dạng kiểu chữ hoa (Upper), chữ thường (Lower) hay mặc định (Normal).
- Maxlength: số ký tự cho phép nhập.
- MultiLine: true/false - cho/không cho phép nhập nhiều dòng.
- PasswordChar: ký tự thay thế ký tự nhập.
- ReadOnly: true/false - khóa/cho phép nhập văn bản.
- ScrollBars: thiết lập thanh cuộn khi MultiLine = true
 - Vertical: thanh cuộn dọc.
 - Horizontal: thanh cuộn ngang,
 - Both: cả 2 thanh cuộn,
 - None: không có thanh cuộn.
- WordWrap: true/false - cho/không cho phép văn bản tự động xuống dòng khi chuỗi nhập dài hơn kích thước của điều khiển.

- Một số sự kiện trên TextBox:
 - **MouseClick**: xảy ra khi click vào TextBox.
 - **MouseDoubleClick**: xảy ra khi double click vào TextBox.
 - **TextChanged**: sự kiện mặc định, xảy ra khi chuỗi trên TextBox bị thay đổi.



TextBox

■ MaskedTextBox



- Bài tập luyện tập
 - Textbox tự động chuyển thành chữ hoa hay chữ thường khi nhập liệu
 - Dùng Textbox để nhập password
 - Hãy giới hạn số ký tự trong Textbox

- Bài tập luyện tập
 - Textbox tự động chuyển thành chữ hoa hay chữ thường khi nhập liệu

```
textBox1.CharacterCasing = CharacterCasing.Upper;
```
 - Dùng Textbox để nhập password

```
textBox1.PasswordChar = '*';
```
 - Hãy giới hạn số ký tự trong Textbox

```
textBox1.MaxLength = 20;
```

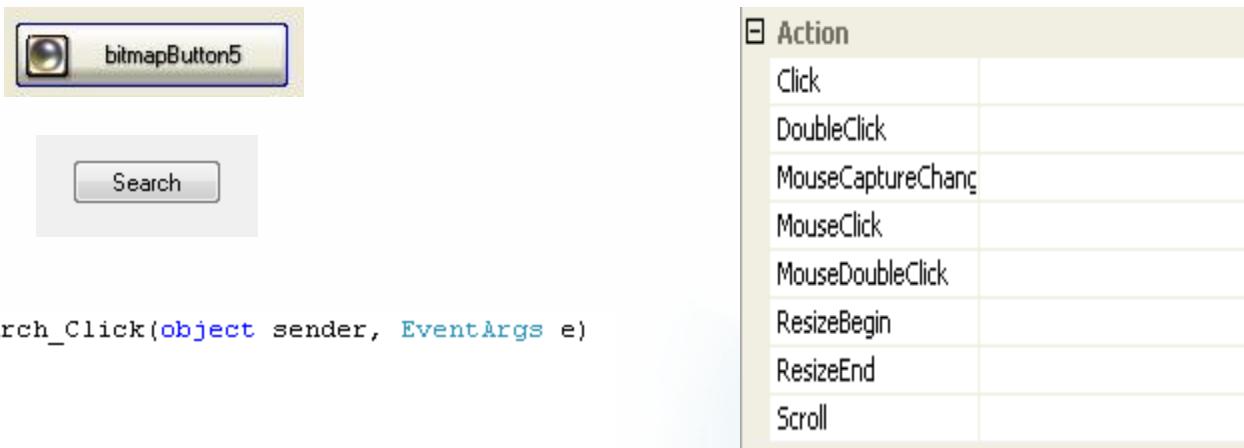
- Bài tập luyện tập
 - Hãy focus đến control tiếp theo khi nhấn enter

```
private void txtTenKhachHang_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
        SendKeys.Send("{TAB}");
}
```



Button

- **Button control:** cho phép user click lên nó để thực hiện một hành động
- Một số property thông dụng
 - Text, Image, TextAlign, ImageAlign, DialogResult



Action
Click
DoubleClick
MouseCaptureChange
MouseClick
MouseDoubleClick
ResizeBegin
ResizeEnd
Scroll

```
private void btSearch_Click(object sender, EventArgs e)
{
    //Code Here
}
```

- Bài tập luyện tập
 - Làm thế nào để click button bằng lập trình
 - Làm thế nào để tự tạo ra một button mới



Button

```
private void btnAddButton_Click(object sender, EventArgs e)
{
    pnButton.Controls.Clear();
    for (int i = 0; i < Int32.Parse(txtNumberControl.Text); i++)
    {
        Button btnRuntime = new Button();
        btnRuntime.BackColor = Color.Red;
        btnRuntime.Location = new System.Drawing.Point
            (pnButton.Width/2-btnRuntime.Width/2,
             i * btnRuntime.Height);
        btnRuntime.Text = "a_" + i;
        btnRuntime.Tag = i;
        btnRuntime.Click += btnRuntime_click;
        pnButton.Controls.Add(btnRuntime);
    }
}

private void btnRuntime_click(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    lblMessage.Text = "Button : "+btn.Text +" was clicked";
}
```

CheckBox

Multi-Select

- Cho phép người sử dụng tại cùng một thời điểm có thể chọn nhiều lựa chọn.
- Các thuộc tính thông dụng:
 - **Checked**: true/false (được chọn/ không chọn)
 - **CheckState**: trạng thái được chọn của checkbox
 - Checked.
 - Unchecked.
 - Indeterminate.
- Sự kiện mặc định: **CheckedChanged**, xảy ra khi người sử dụng thay đổi lựa chọn trên checkbox.
- Ví dụ: xem tài liệu học tập Lập trình giao diện

Events

CheckedChanged – sự kiện phát sinh khi thay đổi trạng thái check

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        // if checkBox1.Checked is true
        label1.Text = "checkBox1 is checked";
    }
}
```



Radio Buttons

Alignment

Left

Center

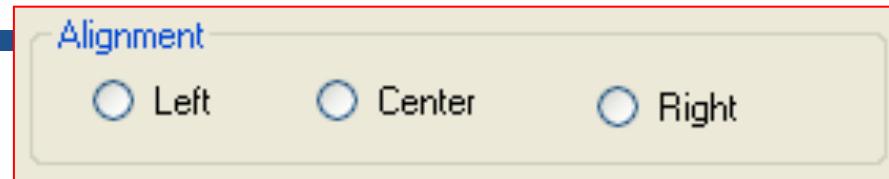
Right

eckbox, tuy nhiên

- Các button trong cùng nhóm chỉ có một button được check tại một thời điểm
- Một nhóm: Các radio button được đặt trong cùng container – thường là panel hay group box



Radio Buttons



iết button có được check

- Event **CheckedChanged** – Sự kiện phát sinh khi check box được check hay không được check



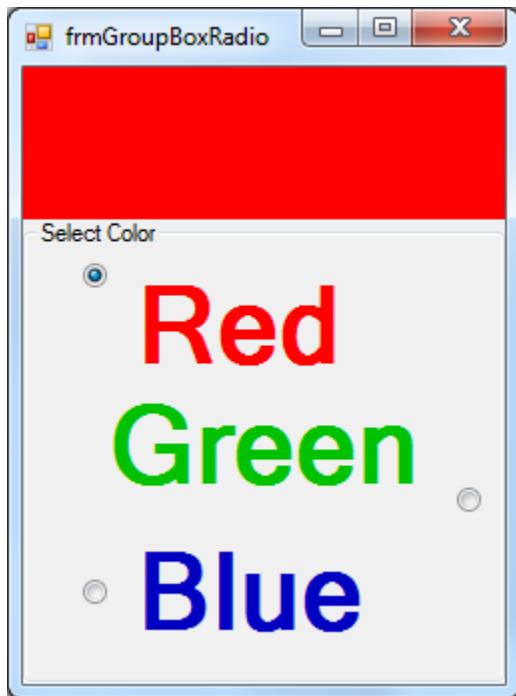
Radio Buttons

Red
 Green

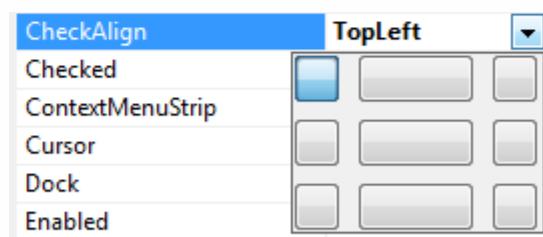
```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton1.Checked)
    {
        // if radioButton1.Checked is true
        label1.Text = "radioButton1 is selected";
    }
}
```



Radio Buttons

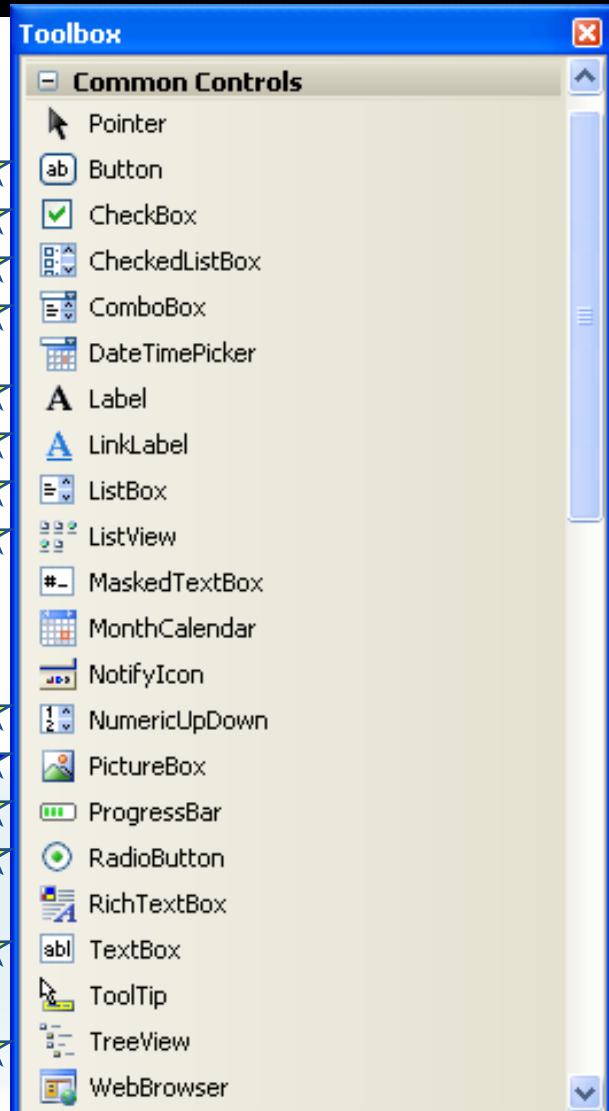


```
private void frmGroupBoxRadio_Load  
(object sender, EventArgs e)  
{  
    radRed.CheckedChanged +=  
        rad_CheckedChanged;  
    radGreen.CheckedChanged +=  
        rad_CheckedChanged;  
    radBlue.CheckedChanged +=  
        rad_CheckedChanged;  
}
```



```
private void rad_CheckedChanged  
(object sender, EventArgs e)  
{  
    RadioButton rad = (RadioButton)sender;  
    pnColor.BackColor = rad.ForeColor;  
}
```

PictureBox



Properties

- **Mage:** Cho phép thiết lập ảnh lúc *thiết kế* hoặc *runtime*.
- **BorderStyle:** quy định kiểu khung
 - *None*: không có border.
 - *FixedSingle*: khung đơn
 - *Fixed3D*: Một dung dạng 3D
- **SizeMode:**
 - *Normal*: hiển thị ảnh như lúc thiết kế.
 - *StretchImage*: tự động giãn ảnh cho vừa với không gian hiển thị của khung.
 - *AutoSize*: kích thước tự động.
 - *CenterImage*: Ảnh được canh giữa.

Ví dụ:

```
Bitmap myJpeg = new Bitmap(@"E:\coffee.jpg");
pictureBox1.Image = (Image)myJpeg;
pictureBox1.Image = Image.FromFile(@"E:\coffee.jpg");
```

- Nạp ảnh từ file

```
string path = "...";
Bitmap bitmap = new Bitmap(path);
pic.Image = bitmap;

pic.SizeMode=PictureBoxSizeMode.StretchImage;
```



PictureBox và Bitmap

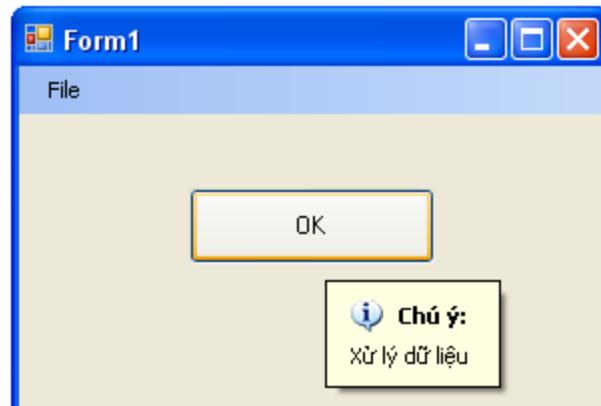
- Bài tập luyện tập
 - Viết chương trình xử lý ảnh cơ bản, gồm những chức năng cơ bản sau
 - Chọn file ảnh, Lưu ảnh
 - Quay ảnh
 - Hiện thị thông tin của ảnh
 - Cách hiện thị ảnh (SizeMode)
 - Lấy màu tại vị trí chuột trên ảnh



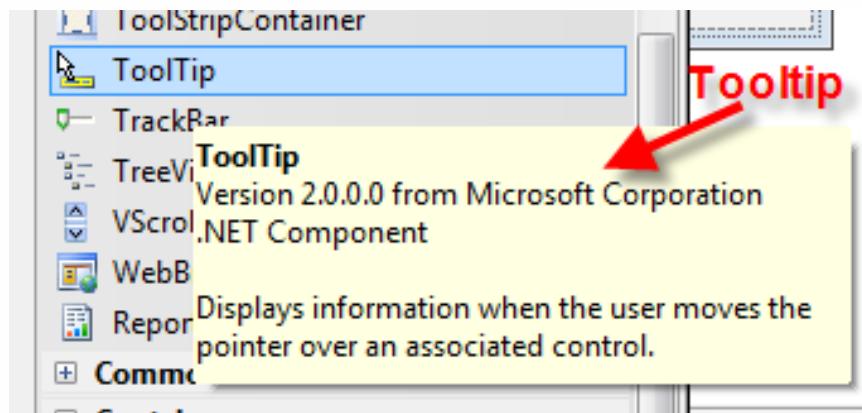
PictureBox và Bitmap

```
//lấy đường dẫn đến thư mục hình  
path = Application.StartupPath + @"\TenTMHinh\";  
//load file hình  
pictureBox1.Image = Image.FromFile(path+"tenFile");
```

- ToolTip một công cụ trợ giúp user hiểu rõ các control trên form. ToolTip sẽ hiển thị một text khi user rê chuột vào control
- Thuộc tính:
 - TooltipTitle: chuỗi tiêu đề
 - ToolTipIcon: biểu tượng kèm theo



- Dùng để hiển thị chú thích cạnh các control khi đưa trỏ chuột đến control
- Một số thuộc tính:
 - TooltipTitle: chuỗi tiêu đề Tooltip
 - ToolTipIcon: biểu tượng hiển thị kèm theo chuỗi





NumericUpDown

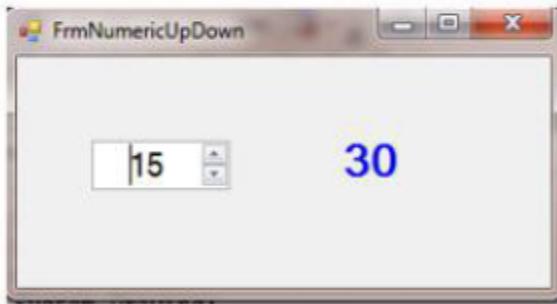
- NumericUpDown control cho phép chọn một số (nguyên/thực) trong miền giới hạn
- Properties
 - Maximum, Minimum – Giá trị lớn nhất và nhỏ nhất có thể chọn
 - Increment – Bước nhảy mỗi lần click
 - DecimalPlaces – Số chữ số lẻ
 - Value – Giá trị hiện tại của control
- Methods
 - void DownButton()
 - void UpButton()
- Events
 - ValueChanged – Sự kiện xảy ra khi value thay đổi





NumericUpDown

```
private void numericUpDown1_ValueChanged(object sender, EventArgs e)
{
    lbNumber.Text = Convert.ToString(
        Convert.ToInt32( numericUpDown1.Value)*2);
}
```





HScrollBar và VScrollBar

- HScrollBar: Thanh cuộn ngang
- VScrollBar: Thanh cuộn dọc
- Properties
 - Value: int
 - Maximum: int
 - Minimum: int
- Events
 - Scroll

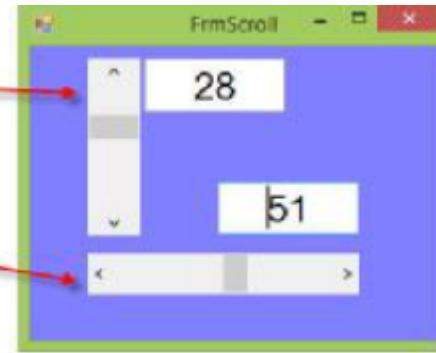




HScrollBar và VScrollBar

```
private void vScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
    textBox1.Text = vScrollBar1.Value.ToString();
}

private void hScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
    textBox2.Text = hScrollBar1.Value.ToString();
}
```





HScrollBar và VScrollBar

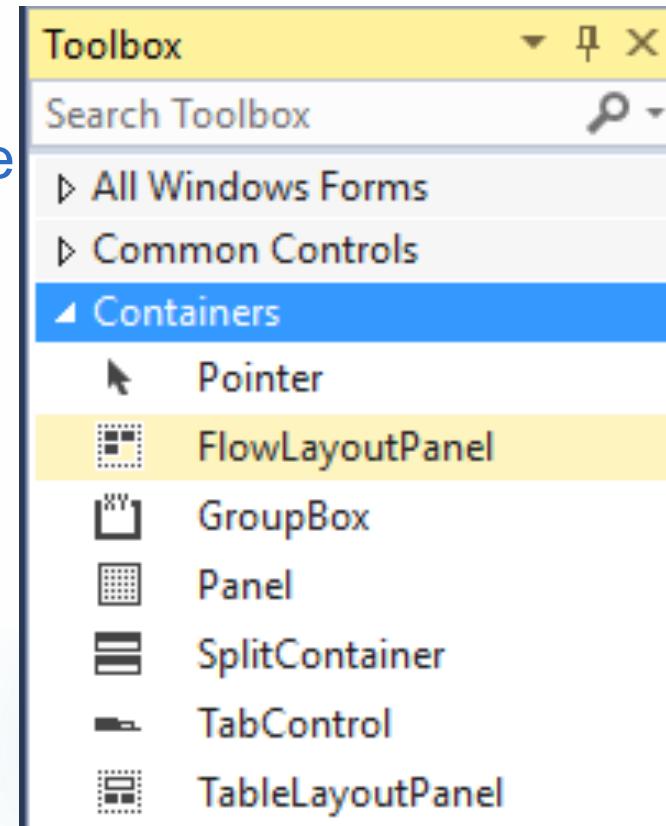
- Bài tập luyện tập
 - Viết chương trình chỉnh sửa màu có giao diện như sau





Container Control

- Là các loại control có thể chứa các điều khiển khác.
- Bao gồm:
 - GroupBox
 - Panel, FlowLayoutPanel, TableLayoutPanel
 - TabControl
 - SplitContainer





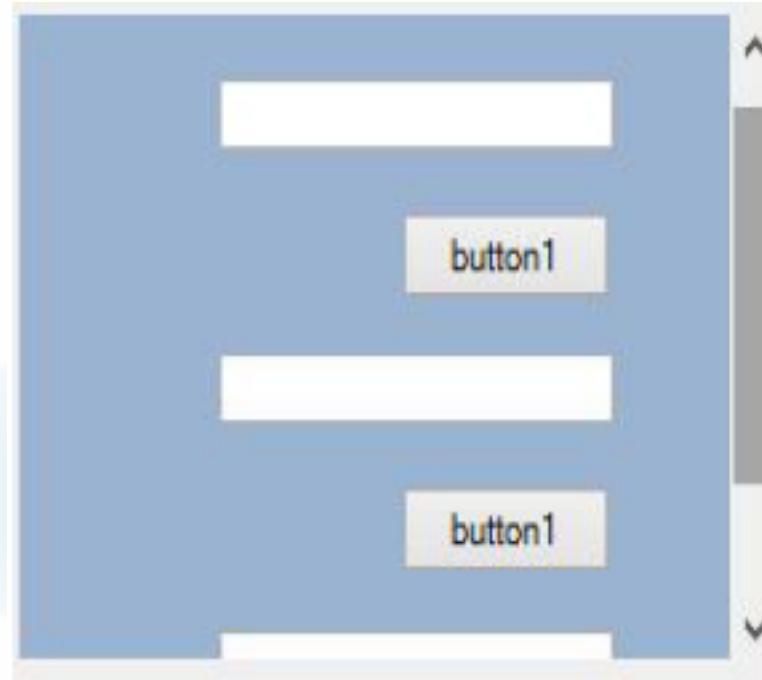
GroupBox



quanh nhóm các control

- Property Text – gán Text label cho GroupBox
- Control có thể được thêm vào
 - Đặt control vào trong GroupBox khi thiết kế
 - Viết code

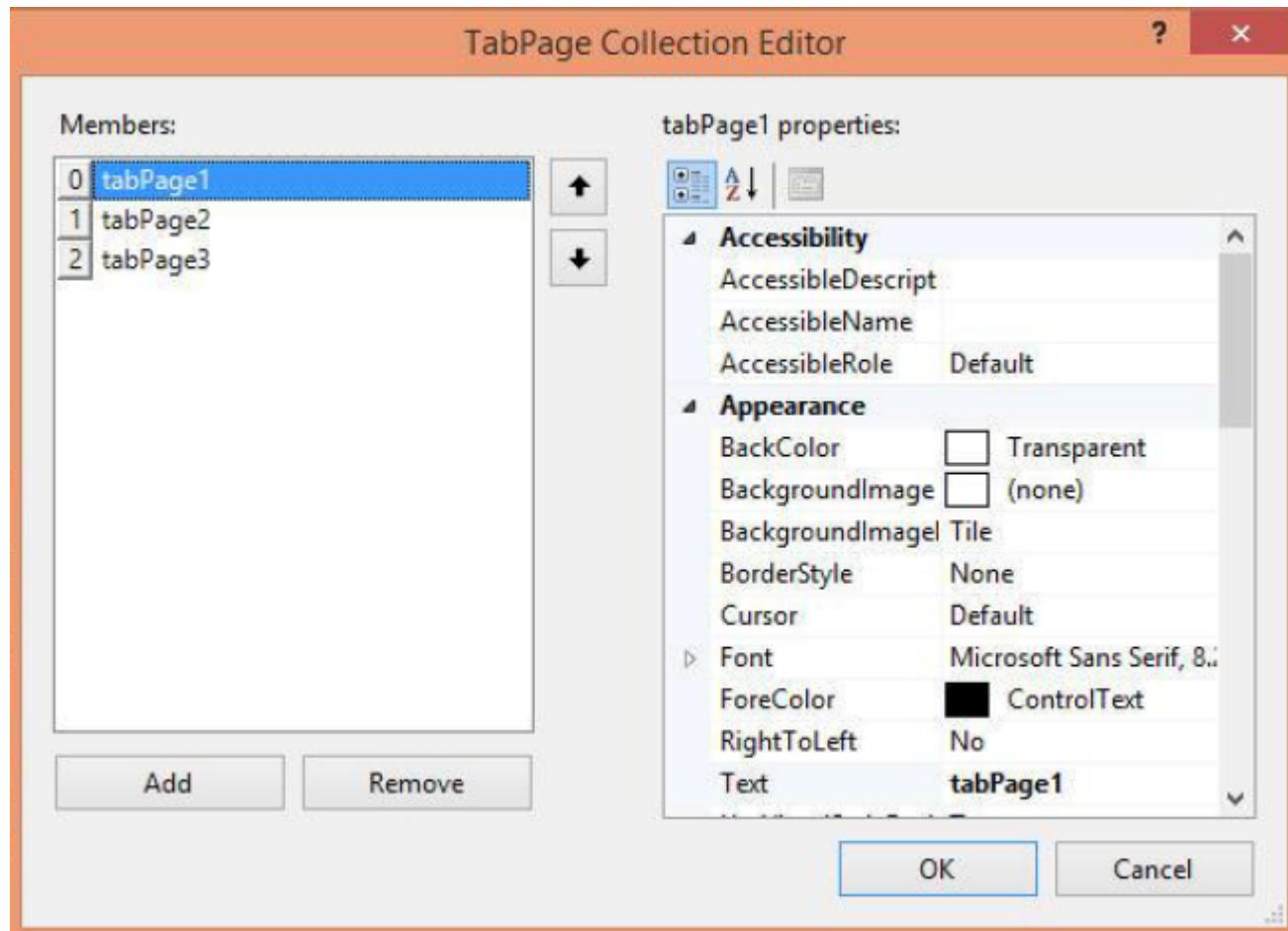
- Panel giống GroupBox nhưng không có text label
- Panel chứa các control như GroupBox
- Có thanh cuộn (scrollbar)



- Là một loại control cho phép thể hiện nhiều trang trên một form duy nhất, các control có cùng nhóm chức năng thường được sắp xếp trong cùng một trang.
- Mỗi trang chứa các control thành một nhóm, có chứa tiêu đề trang.
- Click vào mỗi tiêu đề để chuyển qua lại giữa các trang.
- Để thêm, xóa và hiệu chỉnh một trang trong TabControl, ta truy cập vào thuộc tính **TabPage** và thao tác trong cửa sổ **TabPage Collection Editor**

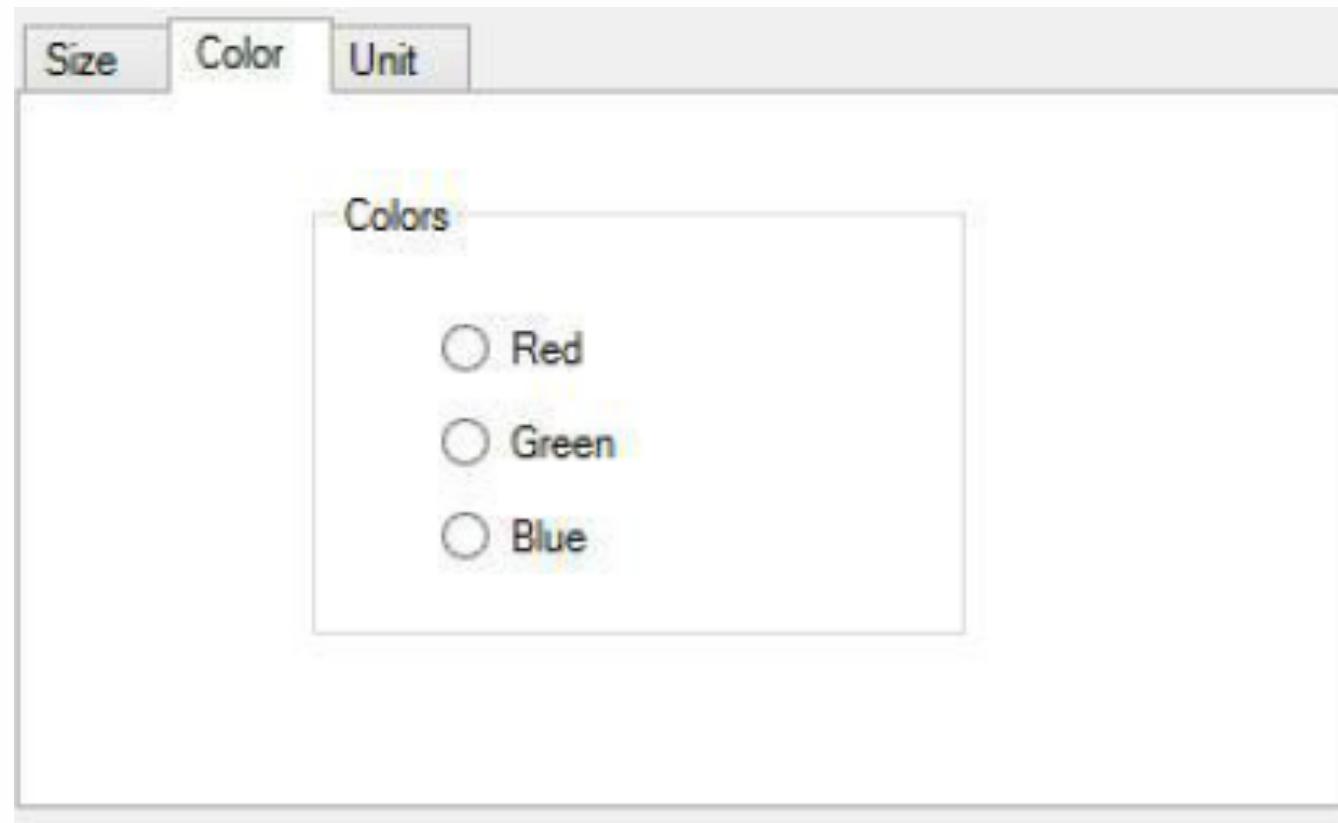


TabControl





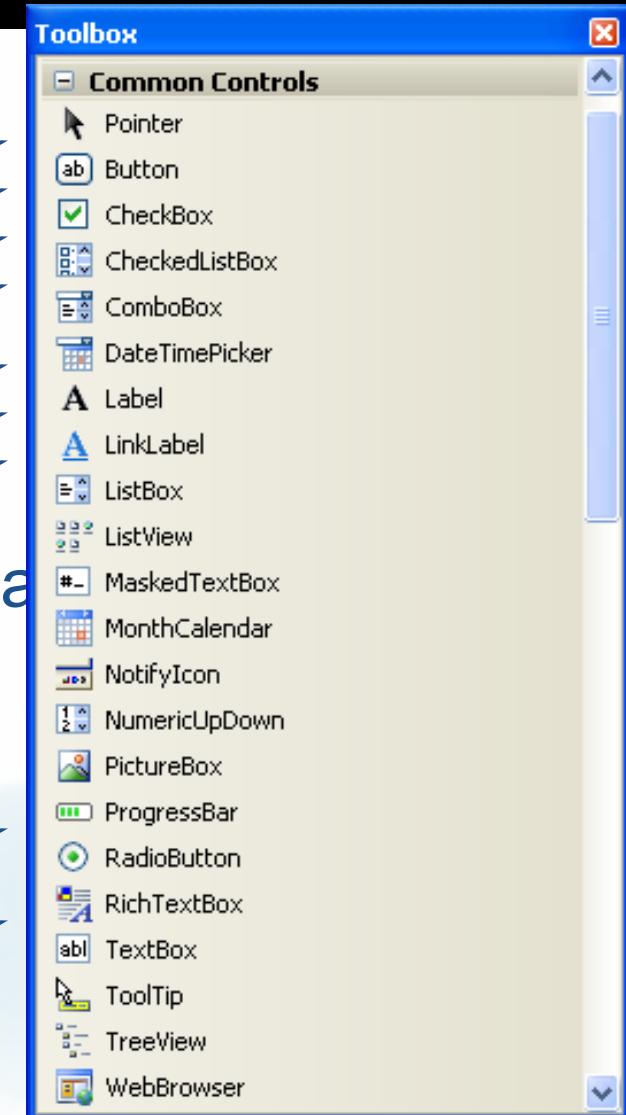
TabControl





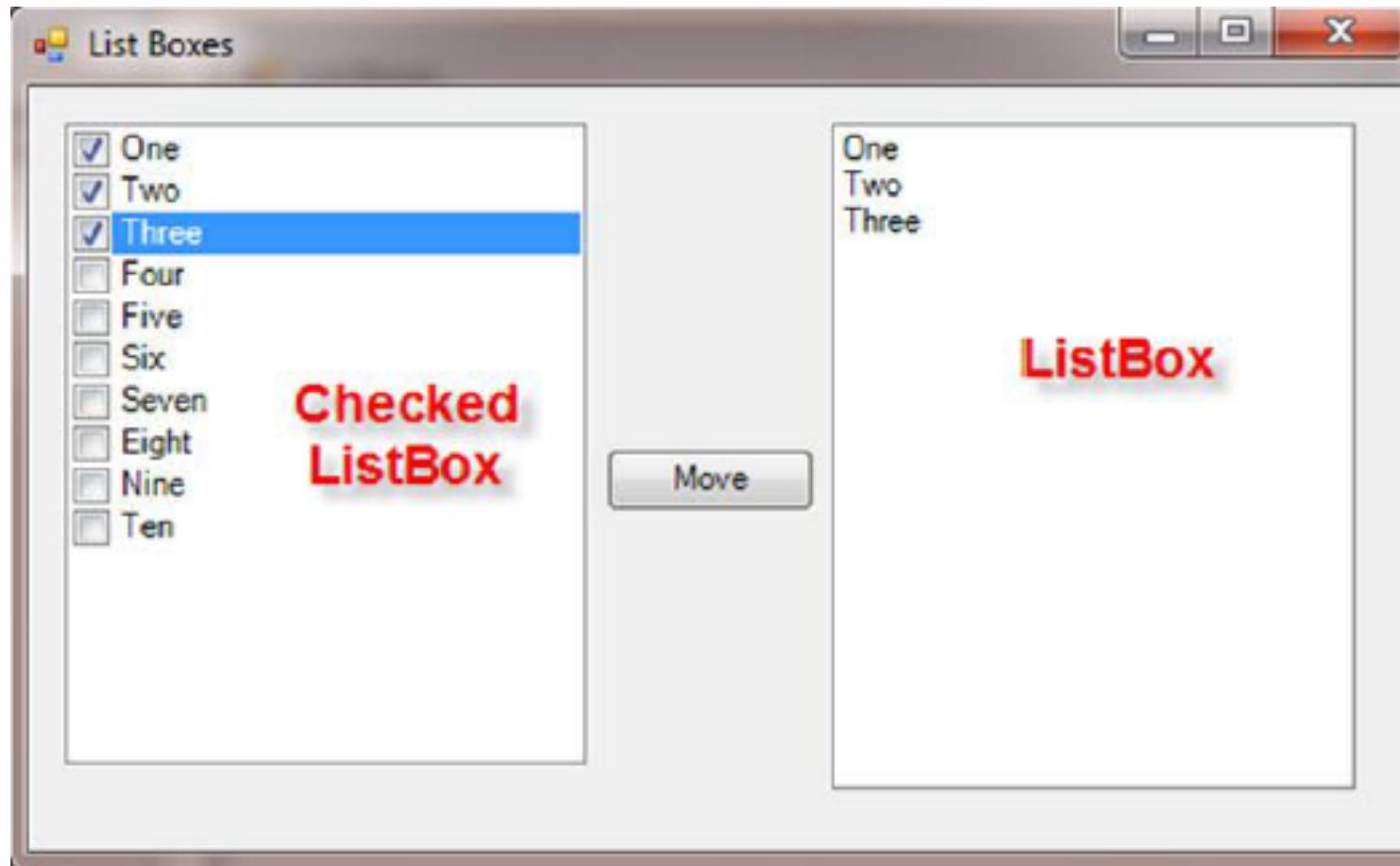
Control dạng danh sách

- Có 3 list controls
 - ListBox
 - CheckedListBox
 - ComboBox
- Lớp cơ sở: thừa kế từ lớp trừu tượng ListControl





Control dạng danh sách



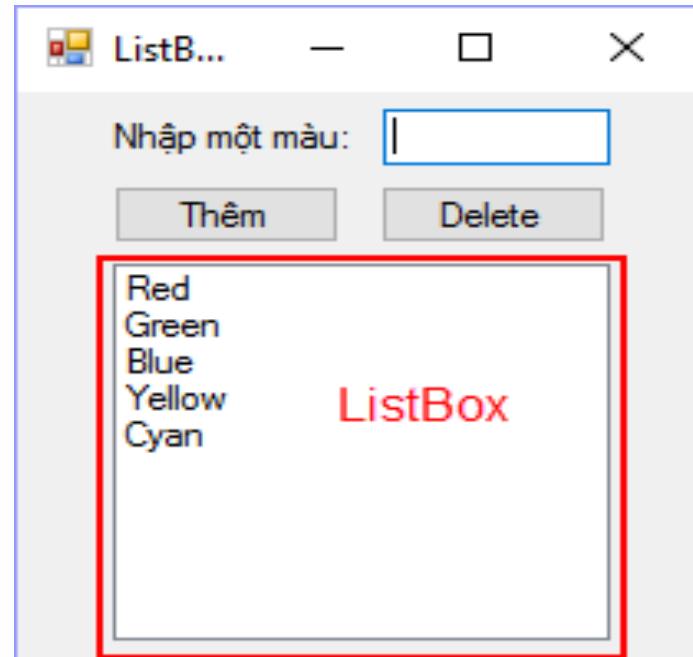
- Properties
 - Items – danh sách các item có trong list
 - Sorted = true: tự động sắp xếp theo từ điển
 - SelectedIndex, SelectedItem, SelectedItems và Text – cung cấp các cách khác nhau để truy cập các mục đã chọn
 - SelectionMode: None, One, MultiSimple, MultiExtended
 - CheckedItems (CheckedListBox)
- Methods
 - int FindString(string s) – tìm chuỗi s có trong list hay không
- Events
 - *SelectedIndexChanged*



List Controls

- Thêm item vào item list
 - `listName.Items.Add("")`;
 - `listName.Items.AddRange(String [])`;
- Chèn item vào item list
 - `listName.Items.Insert(index, data)`;
- Xóa:
 - `listName.Items.Remove(data)`;
 - `listName.Items.RemoveAt(index)`;
 - `listName.Items.Clear()`;
- Tìm kiếm
 - `listName.Items.IndexOf(object obj)`;

- Control hiển thị danh sách các phần tử là các chuỗi văn bản
- Cho phép chọn một hoặc nhiều phần tử.



- Một số thuộc tính cơ bản:
 - Items: danh sách các phần tử trong ListBox
 - SelectedIndex: vị trí phần tử được chọn.
 - SelectedItem: phần tử được chọn.
 - SelectedItems: danh sách các phần tử được chọn.
 - SelectionMode
 - None: không cho phép chọn.
 - One: chỉ cho phép chọn một phần tử.
 - MultiSimple: chọn nhiều phần tử, không giữ phím Ctrl, Shift.
 - MultiExtended: chọn nhiều phần tử, giữ phím Ctrl, Shift.
 - Sorted: cho phép xếp thứ tự tự động.

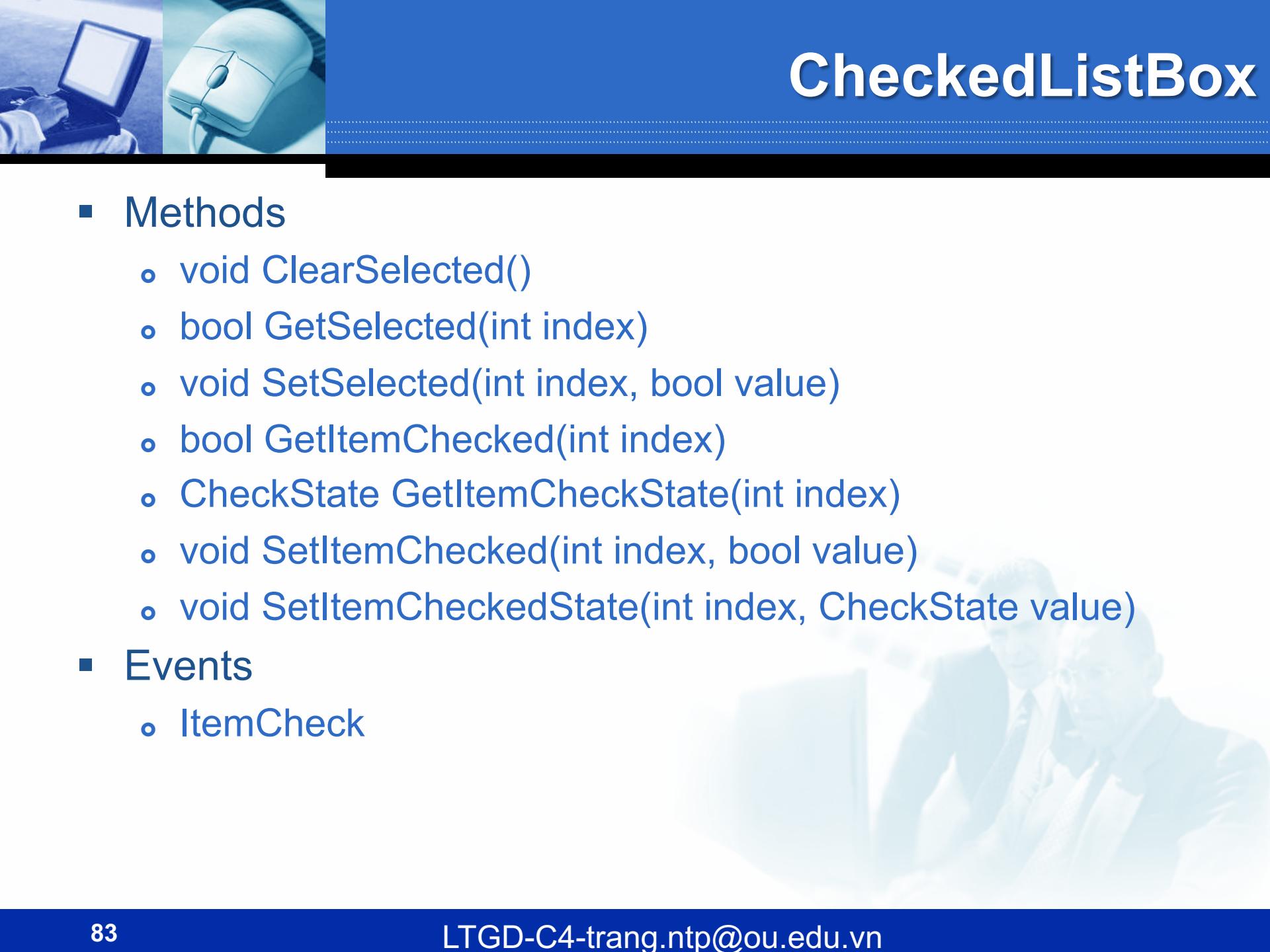
- Một số phương thức:
 - ClearSelected: bỏ chọn tất cả.
 - FindString: Tìm một chuỗi trong ListBox bắt đầu bởi một chuỗi cần tìm.
 - FindStringExact: như FindString, nhưng tìm chính xác.
 - GetSelected: trả về phần tử được chọn.
 - SetSelected: chọn một phần tử.

- Thêm phần tử vào ListBox: sử dụng các phương thức Add, Insert:
 - `list_name.Items.Add (text)`
 - `list_name.Items.AddRange (array)`
 - `list_name.Items.Insert (index, text)`
- Xóa phần tử trong ListBox: sử dụng các phương thức Remove và RemoveAt:
 - `list_name.Items.RemoveAt (index)`
 - `list_name.Items.Remove (delstring)`



CheckedListBox

- **CheckedListBox control – Hiện danh sách các checkbox**
- **Thuộc tính:**
 - BorderStyle: Kiểu đường viền
 - MultiColumn: Trình bày danh sách phần tử trên điều khiển có nhiều cột
 - ColumnWidth: Chiều rộng của mỗi cột
 - Sorted: Nếu chọn True thì danh sách tăng dần theo giá trị
 - SelectedValue: Gán hay lấy giá trị ứng phần tử kiểu object đang chọn
 - SelectItem: Gán hay lấy giá trị object đang chọn
 - SelectIndex: Gán hay lấy chỉ mục ứng với phần tử đang chọn
 - CheckedItems: Trả về tập phần tử được Check
 - Items: Tập các phần tử có điều khiển



CheckedListBox

- Methods

- void ClearSelected()
- bool GetSelected(int index)
- void SetSelected(int index, bool value)
- bool GetItemChecked(int index)
- CheckState GetItemCheckState(int index)
- void SetItemChecked(int index, bool value)
- void SetItemCheckedState(int index, CheckState value)

- Events

- ItemCheck

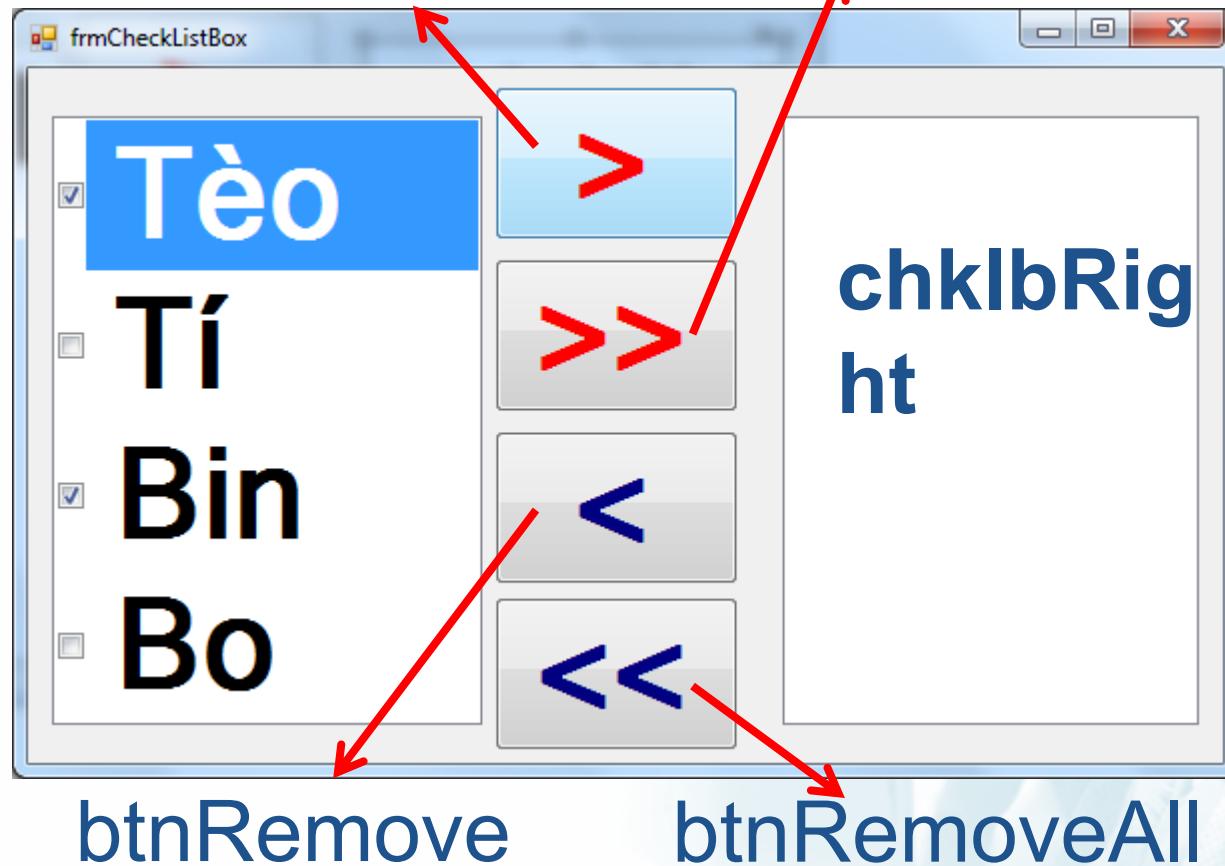


CheckedListBox

chkIbLeft

btnAdd

btnAddAll





CheckedListBox

Dùng **Items/AddRange** để thêm dữ liệu

```
private void frmCheckListBox_Load  
    (object sender, EventArgs e)
```

```
{    chklbLeft.Items.Add("Tèo");  
    chklbLeft.Items.Add("Tí");  
    chklbLeft.Items.Add("Bin");  
    chklbLeft.Items.Add("Bo");  
}
```

Hoặc

```
chklbLeft.Items.AddRange(new string[] {  
    "Tèo", "Tí", "Bin", "Bo"});
```

Th1:

```
CheckedListBox.CheckedIndexCollection  
indexCollection = chklbLeft.CheckedIndices;  
string strChecked = "";  
foreach (int i in indexCollection)  
{  
    strChecked += i + ":";  
}  
MessageBox.Show(strChecked);
```

TH2:

```
CheckedListBox.CheckedItemCollection  
items = chk1bLeft.CheckedItems;  
string strChecked = "";  
foreach (string s in items)  
{  
    strChecked += s + ";"  
}  
MessageBox.Show(strChecked);
```

TH3:

```
string strChecked = "";  
for (int i = 0; i <  
     chk1bLeft.Items.Count; i++)  
{  
    if (chk1bLeft.GetItemChecked(i))  
    {  
        //Process Item checked here  
    }  
}
```



Ví dụ CheckBoxList

```
private void btnAdd_Click  
(object sender, EventArgs e)  
{  
    foreach(int i in chk1bLeft.CheckedIndices)  
    {  
        chk1bRight.Items.Add(chk1bLeft.Items[i]);  
    }  
    foreach (string s in chk1bRight.Items)  
    {  
        chk1bLeft.Items.Remove(s);  
    }  
}
```



Ví dụ CheckBox

```
private void btnAddAll_Click  
(object sender, EventArgs e)  
{  
    chk1bRight.Items.AddRange  
        (chk1bLeft.Items);  
    chk1bLeft.Items.Clear();  
}
```



Ví dụ CheckBoxList

```
private void btnRemove_Click  
(object sender, EventArgs e)  
{  
    foreach (string s in  
            chk1bRight.CheckedItems)  
        chk1bLeft.Items.Add(s);  
    foreach(string s in  
            chk1bLeft.Items)  
        chk1bRight.Items.Remove(s);  
}
```



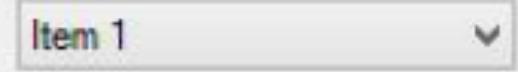
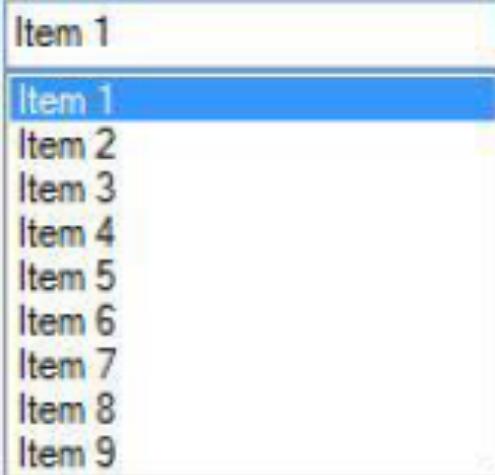
Ví dụ CheckBox

```
private void btnRemoveAll_Click  
(object sender, EventArgs e)  
{  
    chk1bLeft.Items.AddRange  
        (chk1bRight.Items);  
  
    chk1bRight.Items.Clear();  
}
```

- **ComboBox control** cho phép hiển thị danh sách các mục dạng drop down để user chọn
- ComboBox=LISTBOX+TEXTBOX
- Properties
 - MaxDropDownItems, DropDownStyle
- Methods
 - void Select(int start, int length)
 - void SelectAll()
- Events
 - DropDown



ComboBox

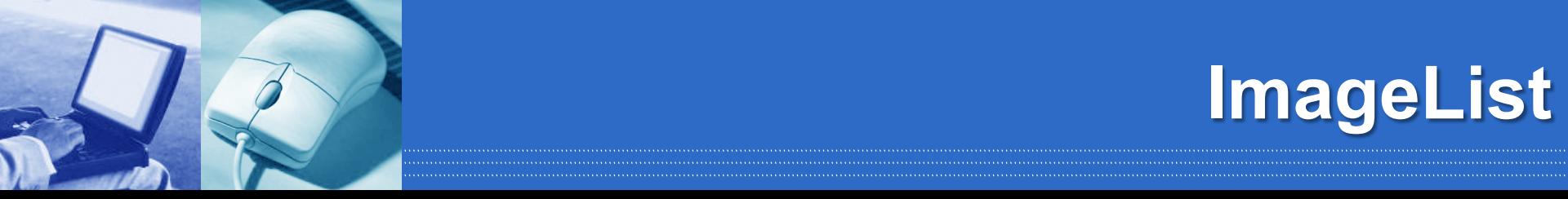


**DropDownStyle
= DropDownList**

**DropDownStyle
= DropDownList**

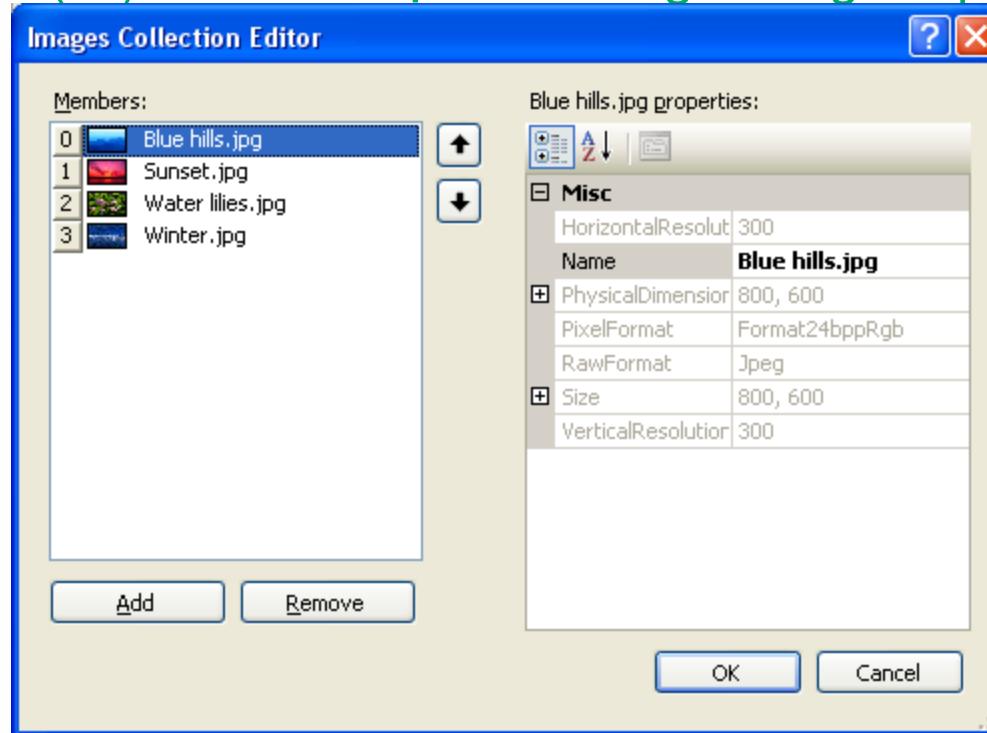
DropDownStyle = Simple

- ImageList là một kiểu collection đặc biệt chứa các ảnh có kích thước và độ sâu màu được xác định trước
- Các control khác nếu có hỗ trợ dùng ImageList thì dùng các ảnh trong ImageList thông qua chỉ mục
 - ListView, TreeView, ToolBar, Button, ...
- Properties
 - Images – collection các ảnh
 - ImageSize – kích thước của các ảnh
 - TransparentColor – Định nghĩa màu trong suốt
- Method
 - Draw(...) Vẽ ảnh lên bề mặt GDI+



ImageList

- Thêm/xóa/sắp xếp các ảnh trong ImageList
 - Cách 1: Dùng Designer để thêm các ảnh (bmp, jpg, ico, ...) vào
 - Click (...) kế bên thuộc tính Image trong Properties



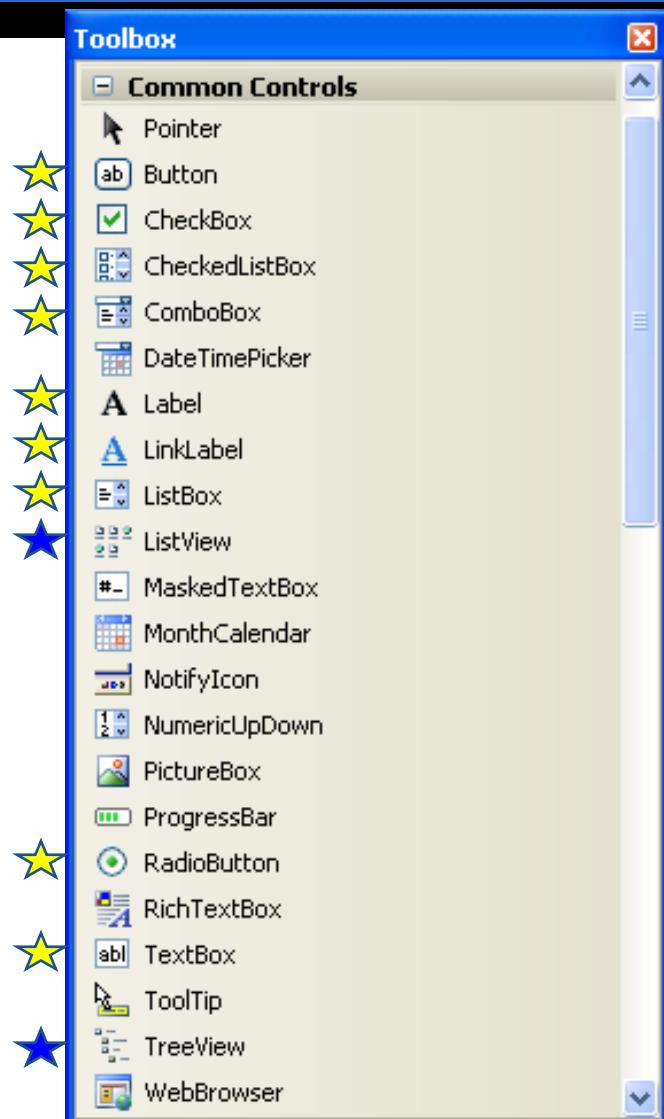
- Thêm/xóa/sắp xếp các ảnh trong ImageList
 - Chú ý: các file ảnh được thêm vào resource file khi dùng designer

Cách 2: Nạp các file ảnh từ file vào ImageList

```
ImageList iconImage = new ImageList();  
  
// Cấu hình ImageList  
iconImages.ColorDepth = System.Windows.Forms.ColorDepth.Depth8Bit;  
iconImages.ImageSize = new System.Drawing.Size(16, 16);  
  
// Lấy các file trong thư mục hiện tại  
string[] iconFiles = Directory.GetFiles(Application.StartupPath,  
                                         "*.ico");  
  
// Thêm vào ImageList  
foreach (string iconFile in iconFiles)  
{  
    Icon newIcon = new Icon(iconFile);  
    iconImages.Images.Add(newIcon);  
}
```



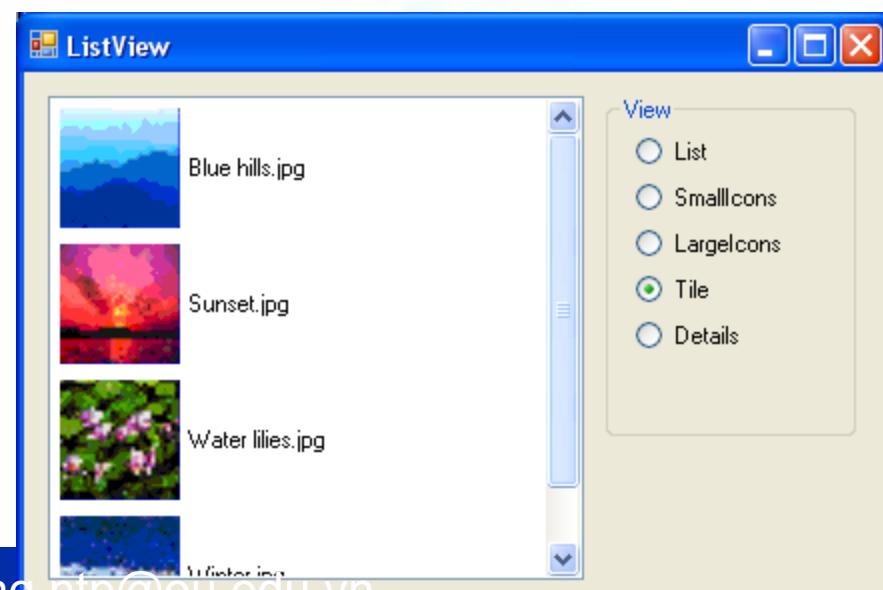
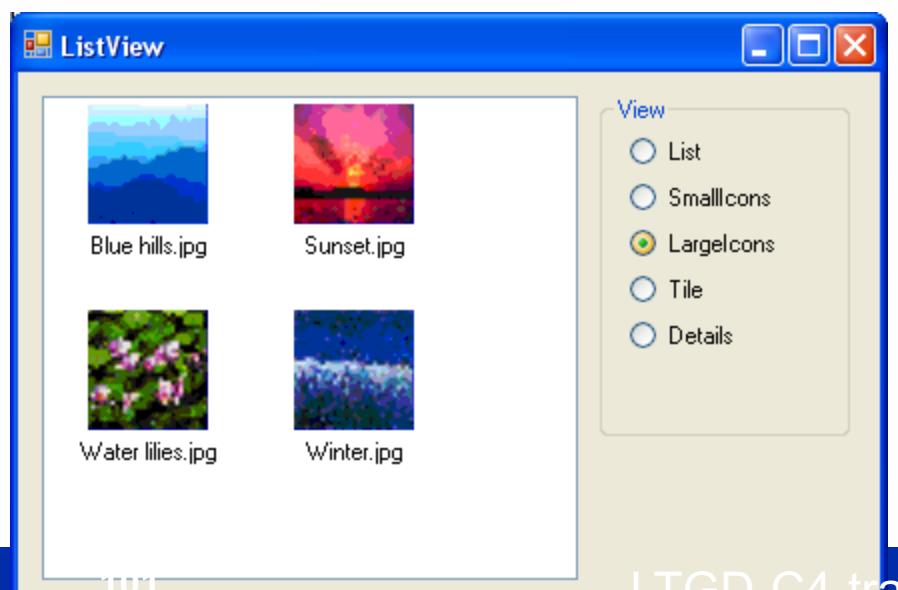
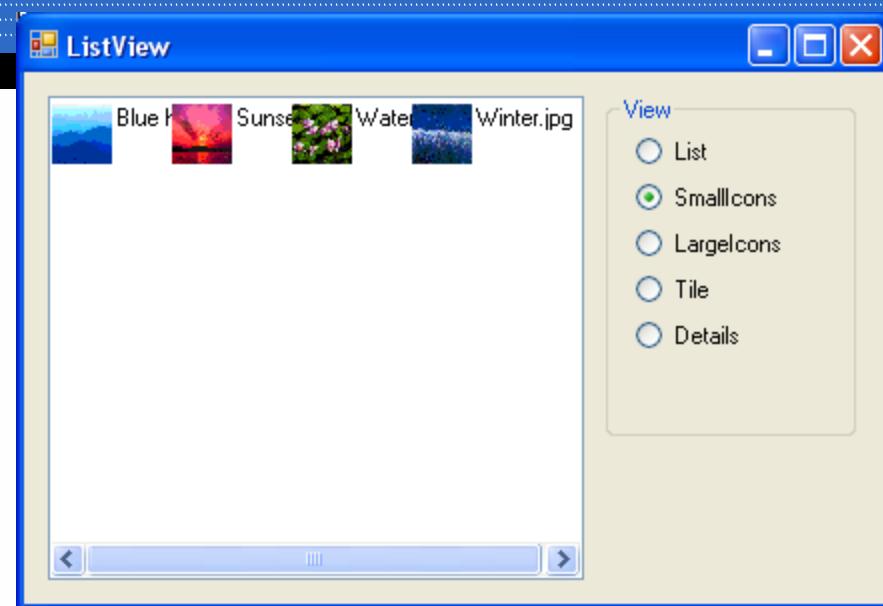
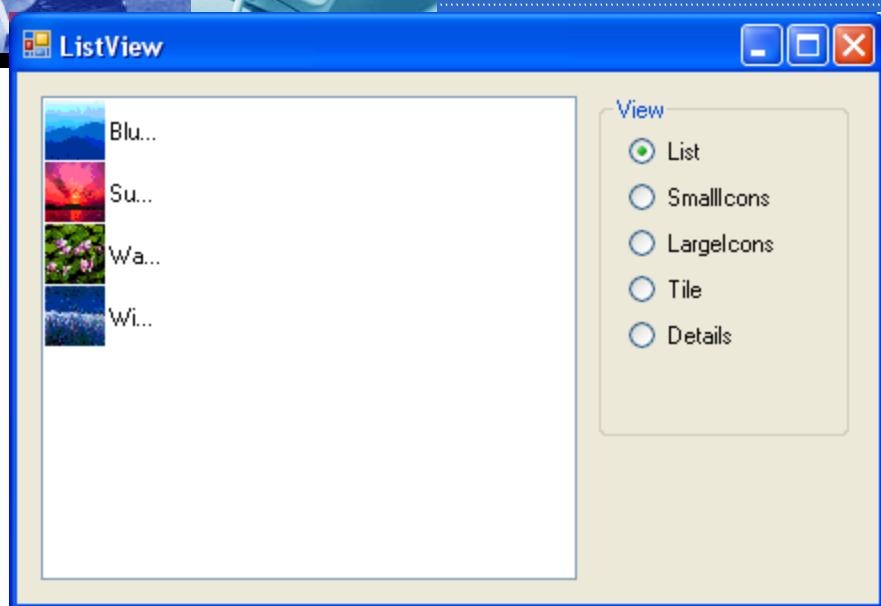
ListView



- Listview là control dùng để trình bày danh sách các mục mà có thể trình bày bằng 1 trong 4 cách khác nhau
 - Các item trong ListView là một đối tượng ListViewItem

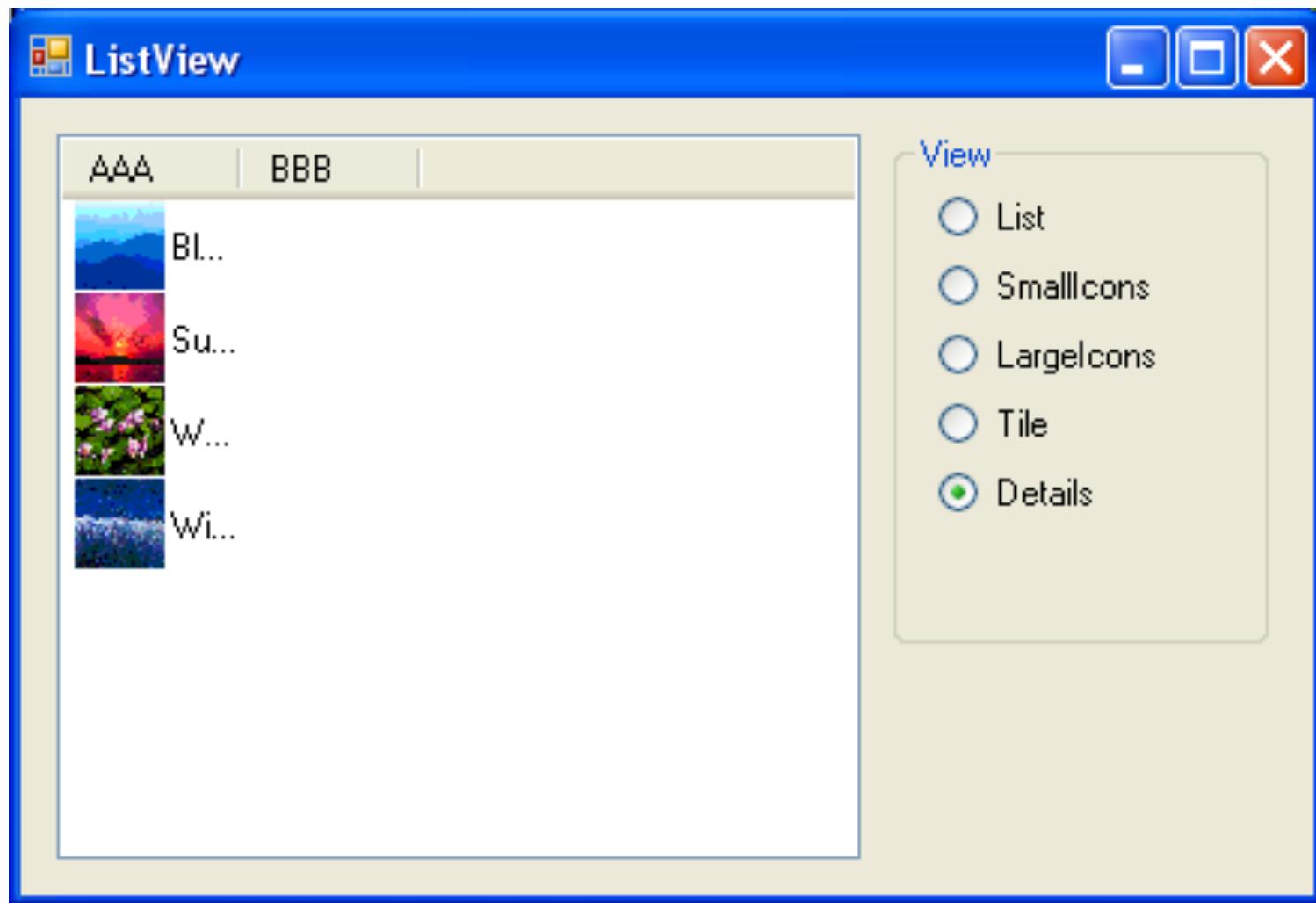


ListView





ListView



- Một số thuộc tính cơ bản:
 - Columns: danh sách các cột trong ListView.
 - Items: danh sách các phần tử (hàng) trong ListView.
 - LargeImageList: ImageList, chứa các biểu tượng hiển thị trong ListView ở chế độ View=LargeIcon.
 - SmallImageList: ImageList, chứa các biểu tượng hiển thị trong ListView ở chế độ View=SmallIcon.
 - View: chế độ hiển thị của ListView.
 - SelectedItems: danh sách các phần tử được chọn trong ListView.

- Thiết kế ListView trong môi trường trực quan: sử dụng cửa sổ ListView Tasks.
- Thêm cột vào ListView bằng code:

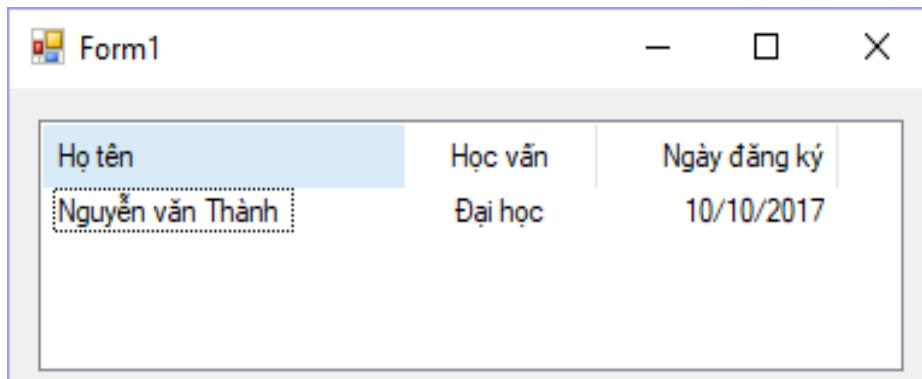
```
listView1.Columns.Add ("ColumnName", Width,  
Alignment);
```

- ColumnName: chuỗi trên tiêu đề cột.
- Width: độ rộng cột, mặc định bằng với kích thước chuỗi.
- Alignment: canh lề chuỗi trên tiêu đề cột:
 - HorizontalAlignment.Left (mặc định)
 - HorizontalAlignment.Right
 - HorizontalAlignment.Center



ListView (tt)

- Thêm các phần tử vào ListView bằng code:
 - Mỗi phần tử là một đối tượng ListViewItem.
 - Tạo đối tượng ListViewItem, đó là phần tử có chuỗi hiển thị ở cột đầu tiên
 - Các chuỗi hiển thị ở các cột kế tiếp lần lượt là các đối tượng SubItem.
- Ví dụ: xem tài liệu học tập Lập trình giao diện



- **Thêm một item đơn**
 - ListViewItem item = new ListViewItem(string);
 - listView.Items.Add(item);
- **Thêm item có ảnh**
 - Tạo imagelist
 - Gán imagelist cho LargeImageList/SmallImageList
 - item.ImageIndex=chỉ số hình trong imagelist
- **Thêm item có nhiều subItem**
 - string[] data = {string1, string2, ...};
 - ListViewItem item = new ListViewItem(data);
 - listView.Items.Add(item);

- Nhận dữ liệu trên ListView
 - ListView item = lstSignature.SelectedItems[0];
 - string sub1 = item.SubItems[0].Text;
 - string sub2 = item.SubItems[1].Text;

- Tạo detail
 - View: Detail
 - GridLines: true
 - Columns: Thêm các column

- Xóa một phần tử trong ListView:

- Xóa phần tử ListViewItem:
`listView1.Items.Remove(item);`
 - Xóa phần tử theo vị trí:
`listView1.Items.RemoveAt(index);`
 - Trong đó:
 - item là một đối tượng ListViewItem.
 - index là vị trí phần tử trong ListView.

- Thay đổi chế độ hiển thị của ListView:
`listView1.View = View.Details;`



ListView

btnHeader

btnAdd

btnRemove

frmListView

ID	Name	Birthday

Load Header

Add One Item

Remove Items

lvDataDn

ID:

Name:

Birthday:

```
private void btnLoadHeader_Click
    (object sender, EventArgs e)
{
    ColumnHeader hdID = new ColumnHeader();
    hdID.Text = "ID";
    ColumnHeader hdName = new ColumnHeader();
    hdName.Text = "Name";
    ColumnHeader hdBirthday = new ColumnHeader();
    hdBirthday.Text = "Birthday";
    lvDataDn.View = View.Details;
    hdID.Width = 100;
    hdName.Width = 140; hdBirthday.Width = 140;
    lvDataDn.Columns.Clear();
    lvDataDn.Columns.AddRange(new ColumnHeader[] {
        hdID,hdName,hdBirthday});
}
```



ListView

```
private void btnAdd_Click(object sender, EventArgs e)
{
    ListViewItem itemID = new ListViewItem();
    itemID.Text = txtID.Text;
    ListViewItem.ListViewSubItem itemName = new
        ListViewItem.ListViewSubItem();
    itemName.Text = txtName.Text;
    itemID.SubItems.Add(itemName);
    ListViewItem.ListViewSubItem itemBirthday = new
        ListViewItem.ListViewSubItem();
    itemBirthday.Text = dateBirthDay.Value.Day + "/"
        + dateBirthDay.Value.Month + "/" +
        dateBirthDay.Value.Year;
    itemID.SubItems.Add(itemBirthday);
    lvDataDn.Items.Add(itemID);
}
```

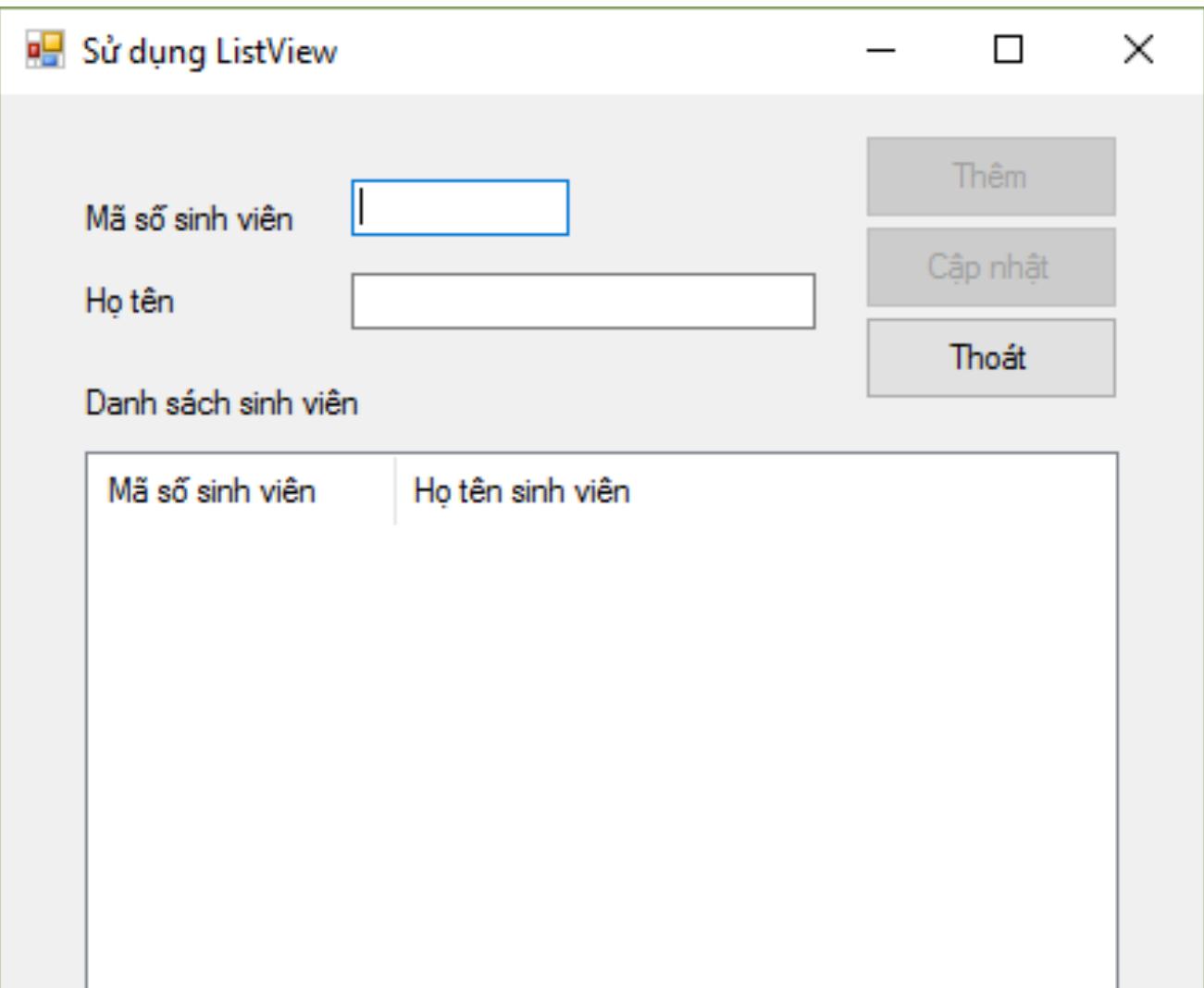


ListView

```
private void btnRemove_Click(object sender,
                           EventArgs e)
{
    ListView.SelectedListViewItemCollection
        list = lvDataDn.SelectedItems;
    foreach (ListViewItem item in list)
        lvDataDn.Items.Remove(item);
}
```



Bài tập



Nút **Thêm** chỉ Enable khi nhập đầy đủ thông tin
Nút **Cập nhật** chỉ Enable khi chọn một dòng nào đó trong danh sách



Bài tập

Sử dụng ListView

Mã số sinh viên	TH15004	Thêm
Họ tên	Lâm Ngọc Toàn	Cập nhật
Thoát		

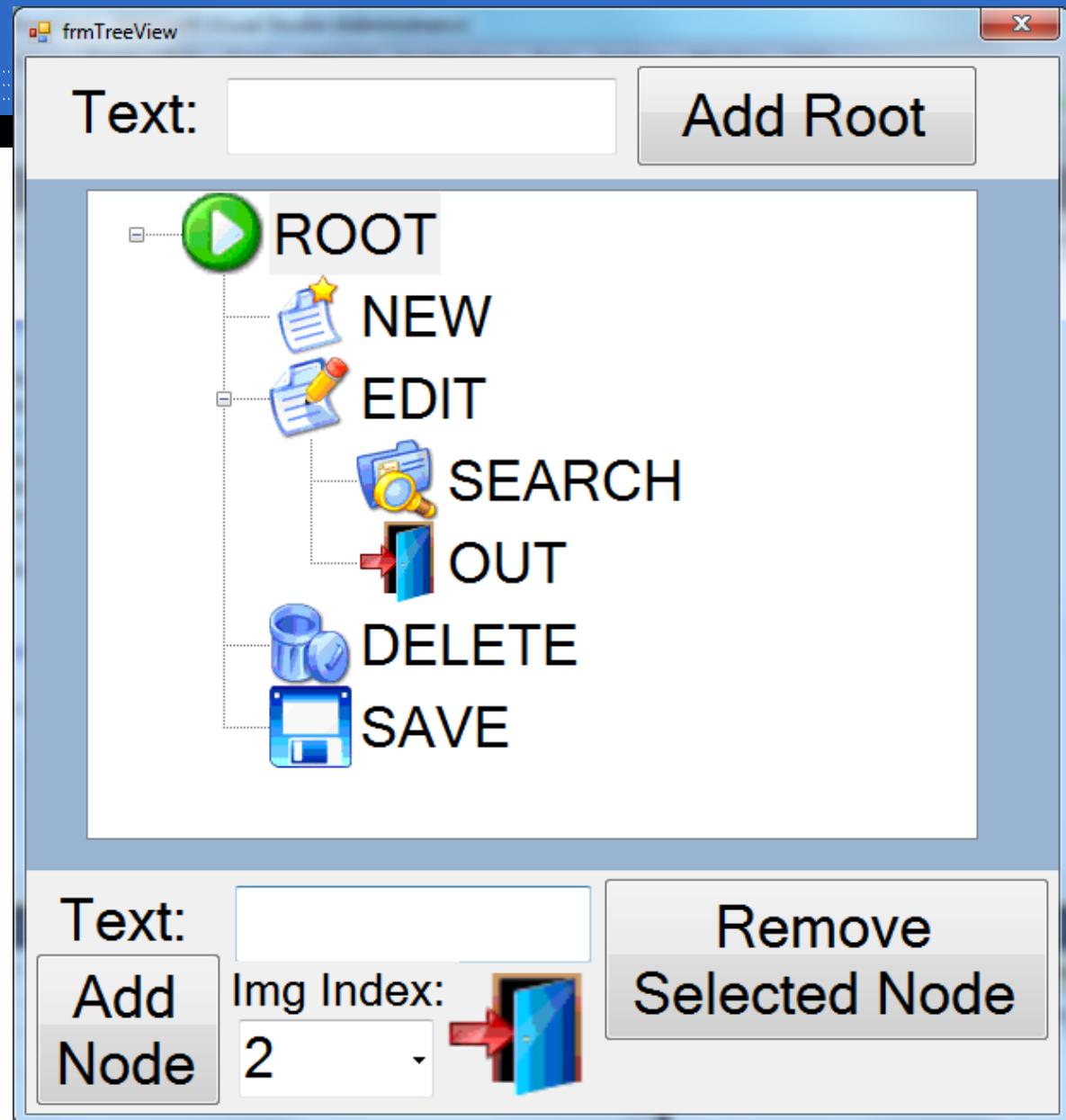
Danh sách sinh viên

Mã số sinh viên	Họ tên sinh viên
TH150001	Nguyễn Thanh Tuấn
TH15004	Lâm Ngọc Toàn
TH15007	Lý Nguyễn Anh Tâm
TH15015	Phan Hữu Đạt

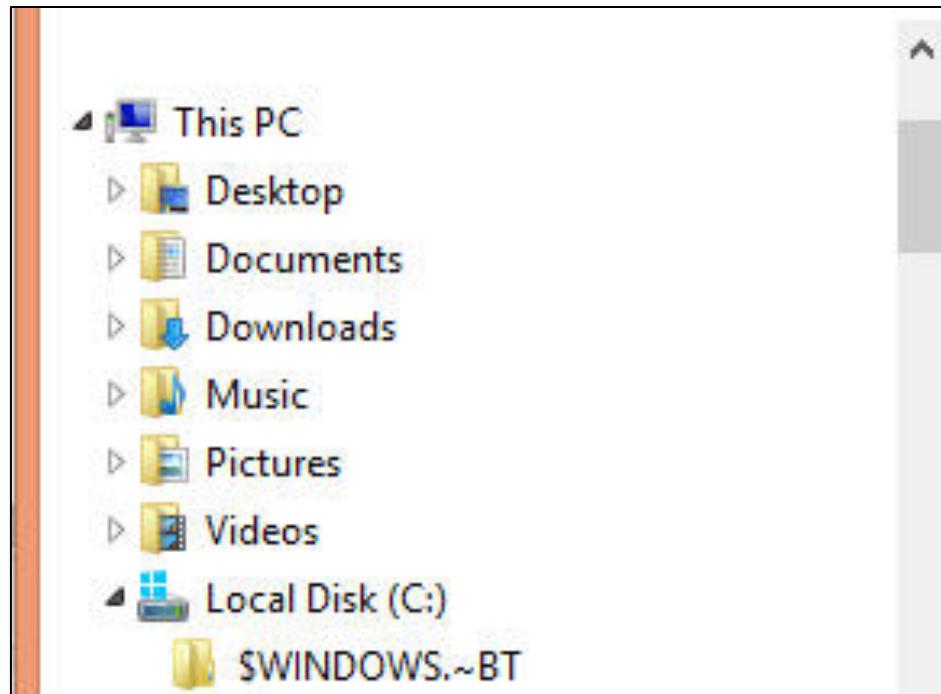
thông tin của sinh viên
sửa thông tin trên



Tree View



- Trình bày danh sách phân tử phân cấp theo từng node (tương tự Windows Explorer)



- Một số thuộc tính cơ bản:
 - Nodes: danh sách các phần tử trong TreeView.
 - ImageList: ImageList, chứa các biểu tượng hiển thị trong TreeView.
 - TreeView chỉ sử dụng duy nhất một đối tượng ImageList.
 - SelectedNode: node đang được chọn.
 - TopNode: node cao nhất trong TreeView.

- Thêm một node
 - Cách 1: tree.Nodes.Add(string);
 - Cách 2:
 - `TreeNode newNode = new TreeNode();`
 - `newNode.Text = "...";`
 - `newNode.Tag = "...";`
 - `tree.Nodes.Add(newNode);`

- Cấu trúc TreeView
 - Các nodes có thể lồng nhiều cấp
 - Thêm node con:
 - Tìm node cha
 - Thêm node con vào node cha
 - Ví dụ 1
- ```
TreeNode node;
node = tree.Nodes.Add("cha");
node.Nodes.Add("con");
```

- Cấu trúc TreeView

- Ví dụ 2:

```
TreeNode[] nodes = new TreeNode[n];
nodes[0] = new TreeNode("cha");
nodes[0].Nodes.Add("con");
...
tree.Nodes.AddRange(nodes);
```

- Duyệt các phần tử của TreeView: 2 kỹ thuật cơ bản
  - Dùng kỹ thuật đệ quy

```
void XuLyCacNode(TreeNodeCollection nodes)
{
 foreach (TreeNode node in nodes)
 {
 // Xử lý node
 XuLyCacNode (node.Nodes)
 }
}
```

- Duyệt các phần tử của TreeView: 2 kỹ thuật cơ bản
  - Dùng các properties của node
    - Parent
    - FirstNode
    - LastNode
    - PrevNode
    - NextNode

- Duyệt các phần tử của TreeView: 2 kỹ thuật cơ bản
  - Dùng các properties của node

```
void XuLyCacNode(TreeNode node)
{
 do
 {
 // Xử lý node
 if (node.Nodes.Count>0)
 XuLyCacNode(node.FirstNode);
 node = node.NextNode;
 } while (node != null);
}
```

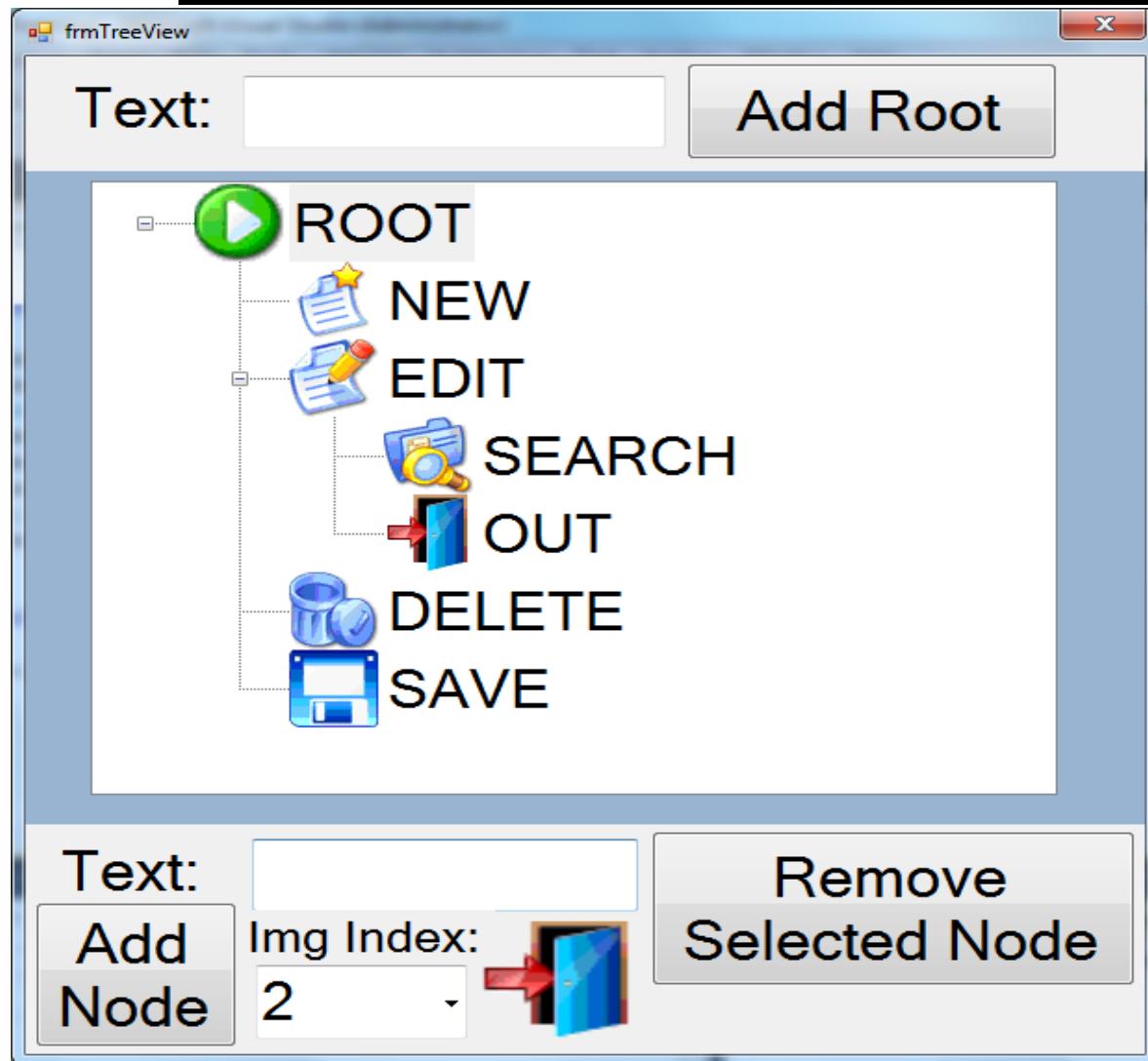
- Thao tác trên TreeView
  - Xóa node: Remove()
    - Tham chiếu đến node cần xóa: Xóa node
    - Tìm vị trí node: xóa node tại vị trí đó
  - Xóa các node: Clear()
  - Tìm kiếm vị trí: IndexOf()

- Node ảnh
  - Ảnh cho toàn TreeView
    - `tree.ImageList = imageList;`
    - `tree.ImageIndex = vị trí`
  - Ảnh cho từng node
    - `tree.ImageList = imageList;`
    - `node.ImageIndex = vị trí`
    - `node.SelectedImageIndex = vị trí`

- Expand và collapse
  - Tree
    - ExpandAll()
    - CollapseAll()
  - Node
    - Collapse(), Expand(), ExpandAll(), Toggle().

- Một số properties khác của TreeView
  - CheckBoxes = bool
  - FullRowSelect = bool
  - Indent = number
  - ShowLines, ShowPlusMinus, và ShowRootLines
  - Sorted = bool
- Một số properties khác của TreeNode
  - Checked = bool
  - IsSelected = bool
  - IsExpanded = bool

# Bài tập



```
private void frmTreeView_Load
 (object sender, EventArgs e)
{
 cboImageIndex.Items.Clear();
 for (int i = 0; i < imgList.Images.Count;i++)
 {
 cboImageIndex.Items.Add(i);
 }
 tvSample.ImageList = imgList;
}
```

```
private void btnAddRoot_Click
 (object sender, EventArgs e)
{
 tvSample.Nodes.Clear();
 TreeNode rootNode = new TreeNode(txtRoot.Text);
 rootNode.ImageIndex =
 Int32.Parse(cboImageIndex.Text);
 rootNode.SelectedImageIndex = rootNode.ImageIndex;
 tvSample.Nodes.Add(rootNode);
 txtRoot.Text = "";
}
```



# Bài tập

```
private void
cboImageIndex_SelectedIndexChanged(object sender,
 EventArgs e)
{
 picShow.Image=imgList.Images[Int32.Parse(
 cboImageIndex.Text)];
}

private void btnRemove_Click(object sender,
 EventArgs e)
{
 TreeNode tNode = tvSample.SelectedNode;
 tvSample.Nodes.Remove(tNode);
}
```



# Bài tập

ComboListBox

Nhập 1 chuỗi:

Chọn màu:

Insert

Phần tử 1  
Phần tử 3  
Phần tử 5

==>

<==

Remove

Clear All

Exit

Phần tử 2  
Phần tử 4



# Bài tập

```
private void MainForm_Load(object sender, EventArgs e){}
private void btnL2R_Click(object sender, EventArgs e){}
private void btnRemove_Click(object sender, EventArgs e){}
private void btnClearAll_Click(object sender, EventArgs e){}
```



# Bài tập

```
private void MainForm_Load(object sender, EventArgs e)
{
 lsbLeft.Items.AddRange(new string[] { "Phần tử 1", "Phần tử 3",
 "Phần tử 5" });
 lsbRight.Items.AddRange(new string[] { "Phần tử 2", "Phần tử 4" });

 cbbColor.Items.AddRange(new string[] { "White", "Yellow", "Lime",
 "Aqua" });
}
```

```
private void btnL2R_Click(object sender, EventArgs e)
{
 foreach (int i in lsbLeft.SelectedIndices)
 {
 lsbRight.Items.Add(lsbLeft.Items[i]);
 lsbLeft.Items.Remove(lsbLeft.Items[i]);
 }
}

Cách 2

foreach (string s in lsbRight.Items)
{
 lsbLeft.Items.Remove(s);
}

}
```



# Bài tập

```
private void btnRemove_Click(object sender, EventArgs e)
{
 if (lsbRight.SelectedItem != null)
 {
 lsbRight.Items.Remove(lsbRight.SelectedItem);
 }
}

private void btnClearAll_Click(object sender, EventArgs e)
{
 lsbLeft.Items.Clear();
 lsbRight.Items.Clear();
}
```



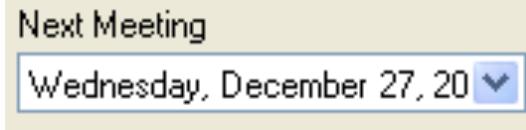
# Month Calendar

- Là control có dạng quyển lịch.
- Thuộc tính:
  - SelectionStart, SelectionEnd, TodayDate
- Biến cố mặc định: DateChanged



# DateTImePicker và MonthCalender

## DateTImePicker



Chọn một ngày

## MonthCalender

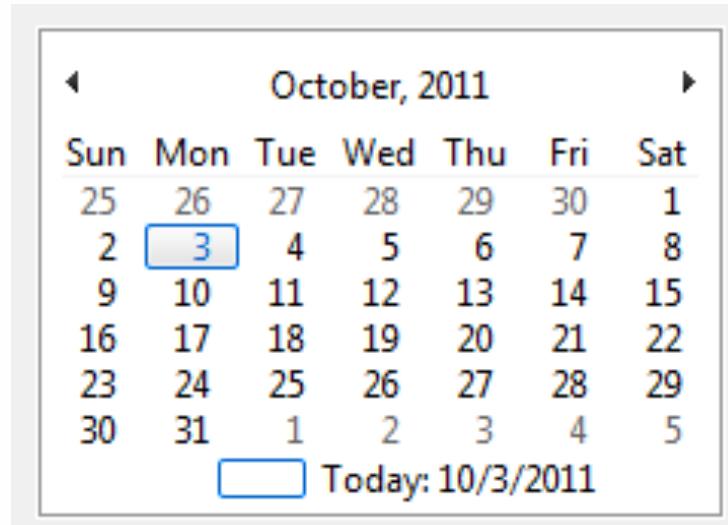


Chọn một vùng ngày



# MonthCalendar

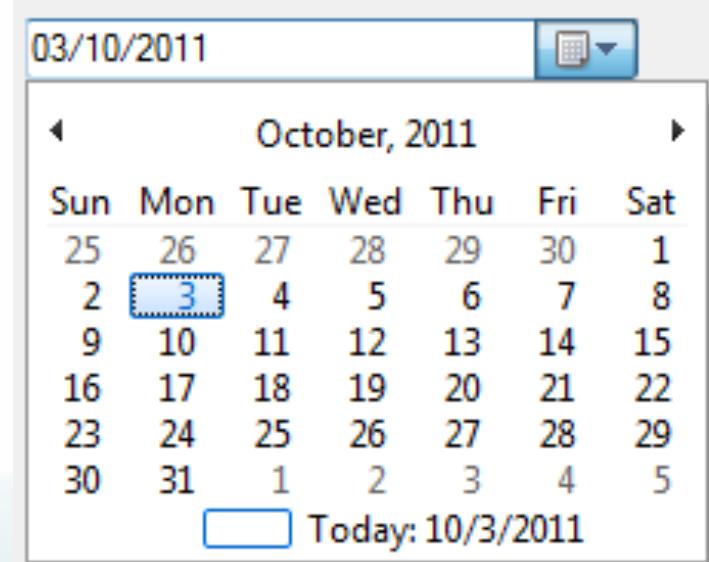
- Thuộc tính
  - SelectionStart: ngày bắt đầu danh sách chọn
  - SelectionEnd: ngày cuối danh sách chọn
  - TodayDate: ngày hiện tại (trên máy tính)
- Biến cố mặc định: DateChanged



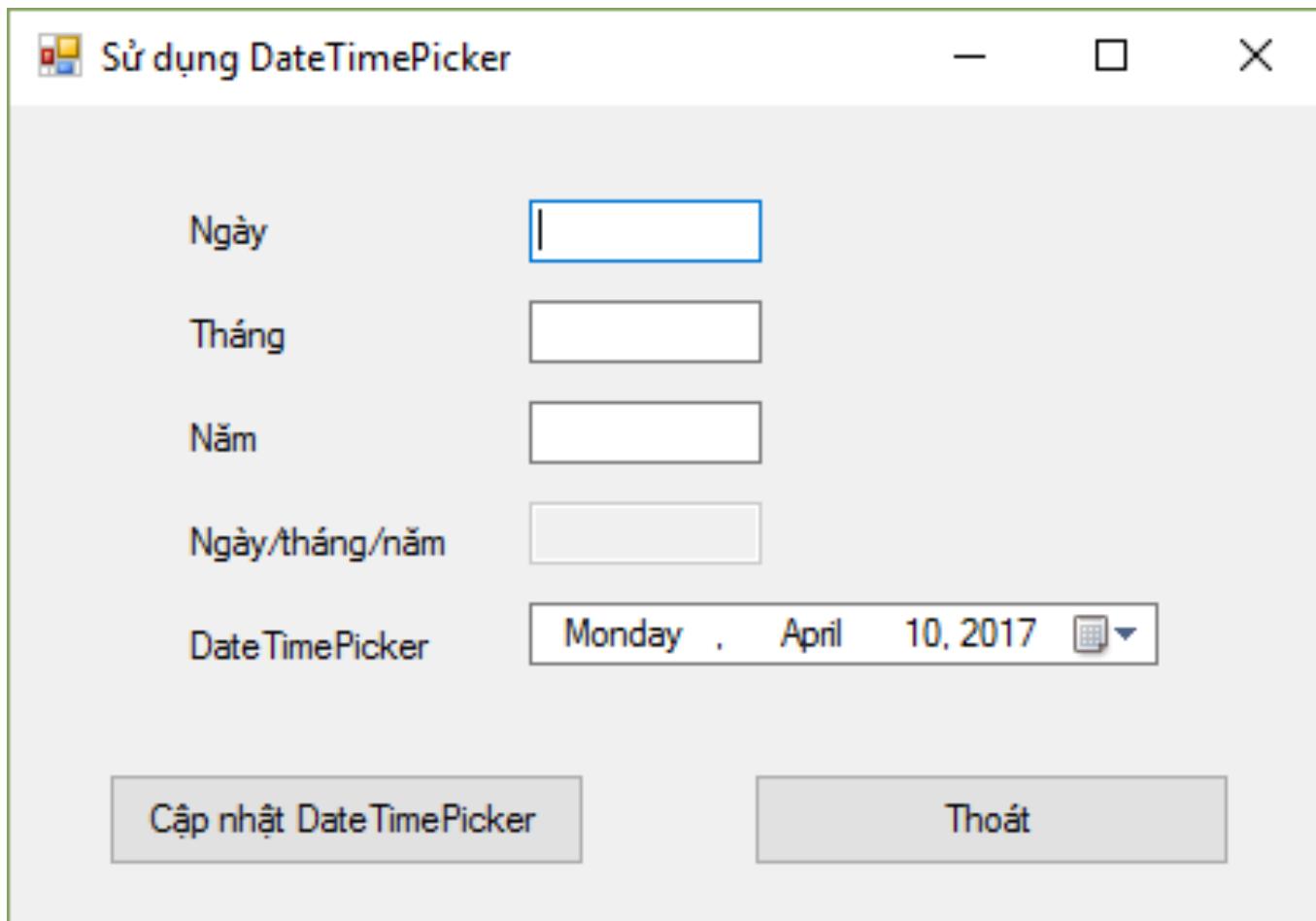


# Date Time Picker

- Kết hợp của hai control ComboBox + MonthCalendar.
- Không chiếm nhiều diện tích trên giao diện và tính năng sử dụng linh hoạt hơn MonthCalendar.
- Các thuộc tính:
  - Format: định dạng hiển thị
    - long, short, time, custom
  - CustomFormat:
    - xem thêm [MSDN Online](#)
  - MaxDate: giá trị ngày lớn nhất
  - MinDate: giá trị ngày nhỏ nhất
  - Value: giá trị ngày hiện tại đang chọn
- Sự kiện mặc định: ValueChanged



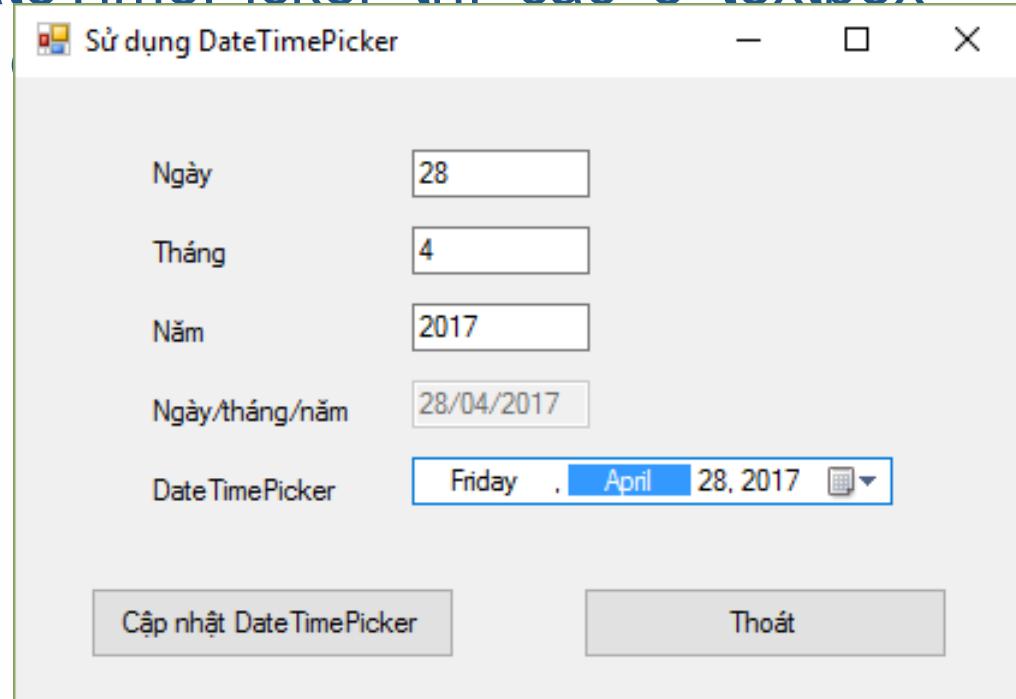
## Thiết kế giao diện như sau





# Bài tập ...

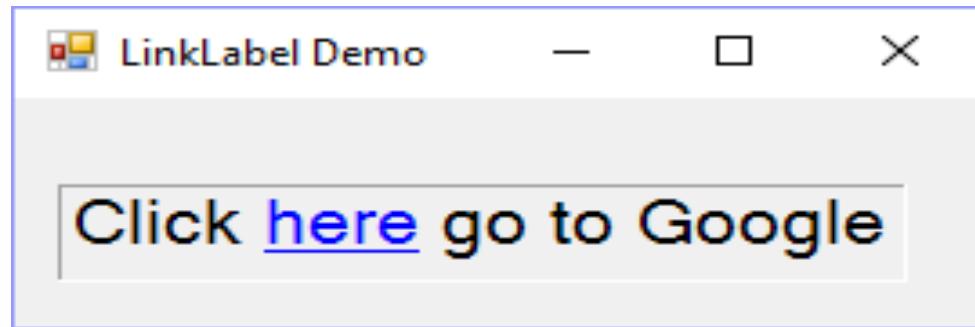
1. Nút **Cập nhật DateTimePicker** dùng để cập nhật DateTimePicker từ các ô textbox ngày, tháng và năm
2. Khi thay đổi trong DateTimePicker thì các ô textbox ngày, tháng và năm sẽ





# LinkLabel

- Chứa liên kết đến một URL, text file,...



- Timer sinh ra sự kiện Tick theo định kỳ
- Properties
  - Interval: int – thời gian sự kiện xảy ra (tính bằng millisecond)
  - Enabled: bool – Cho phép timer start hay stop
- Methods
  - void Start() – Start timer
  - void Stop() – Stop timer
- Events
  - Tick



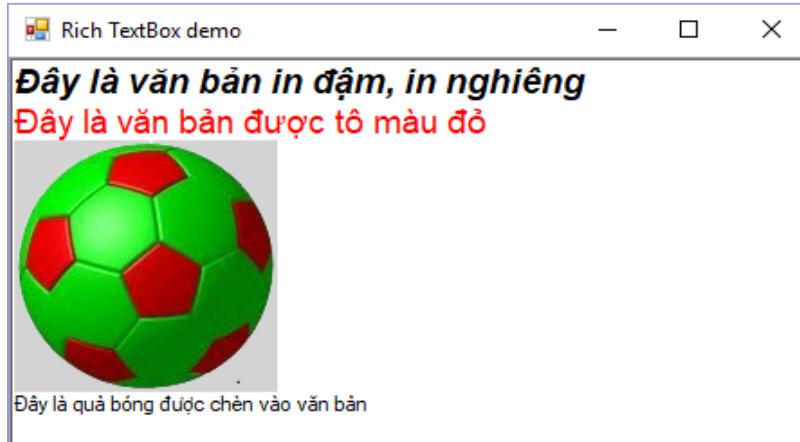
# RichTextBox

- Tương tự như TextBox, nhưng cung cấp nhiều khả năng định dạng hơn
- Cho phép nhập văn bản, hình ảnh hoặc tải nội dung từ một file .txt, .rtf, .docx,...
- Một số thuộc tính cơ bản:
  - SelectedText: chuỗi văn bản được chọn trên RichTextBox.
  - SelectionFont: font chữ áp dụng cho phần văn bản được chọn.
  - SelectionColor: màu chữ áp dụng cho phần văn bản được chọn.
  - CanFocus: true/false - RichTextBox có/không nhận focus.
  - CanPaste: true/false - RichTextBox có/không thể paste.
  - CanSelect: true/false - RichTextBox có/không cho phép chọn văn bản..
  - CanUndo: true/false - RichTextBox có/không thể undo.



# RichTextBox (tt)

- Một số phương thức cơ bản :
  - Copy, Cut: sao chép, cắt dữ liệu được chọn trong RichTextBox lưu vào Clipboard.
  - Paste: dán dữ liệu từ Clipboard vào RichTextBox.
  - SaveFile: lưu dữ liệu trong RichTextBox ra file.
- Ví dụ: xem tài liệu học tập Lập trình giao diện





# RichTextBox

```
private void Form1_Load(object sender, EventArgs e)
{
 Font arial = new Font("Arial", 16, FontStyle.Bold | FontStyle.Italic);
 richTextBox1.SelectionFont = arial;
 richTextBox1.SelectedText = "Đây là văn bản in đậm, in nghiêng\n";
 richTextBox1.SelectionFont = new Font("Arial", 16);
 richTextBox1.SelectionColor = Color.Red;
 richTextBox1.SelectedText = "Đây là văn bản được tô màu đỏ\n";
 InsertBitmap();
 richTextBox1.SelectionFont = arial;
 richTextBox1.SelectedText = "\nĐây là quả bóng được chèn vào văn bản\n";
}
private void InsertBitmap()
{
 Bitmap bmp = new Bitmap(@"E:\ball.jpg");
 Clipboard.SetDataObject(bmp);
 DataFormats.Format format = DataFormats.GetFormat(DataFormats.Bitmap);
 if (richTextBox1.CanPaste(format))
 richTextBox1.Paste(format);
}
```



## 4.7 User control

- Là control do người sử dụng tự định nghĩa.
- Được sử dụng như các control thông thường khác bằng cách kéo thả từ cửa sổ Toolbox.
- UserControl được tạo ra còn có thể được sử dụng trong các ứng dụng khác.
- Tạo UserControl dùng trong một ứng dụng:
  - **Project → Add User Control**
  - UserControl được tạo ra giống như một form nhưng không có tiêu đề.
  - Thiết kế giao diện và thao tác trên UserControl như với một form thông thường.



# User control (tt)

- Tạo User control sử dụng trong ứng dụng khác
  - Tạo ứng dụng loại Class Library project.
  - Sau khi biên dịch thành công, ứng dụng tạo ra một file có phần mở rộng là dll.
  - Trong ứng dụng khác, click chuột phải trên cửa sổ Toolbox, chọn Choose item..., chọn file dll nói trên, click nút OK để hoàn tất.
  - UserControl sẽ hiển thị trên cửa sổ Toolbox như các control khác.



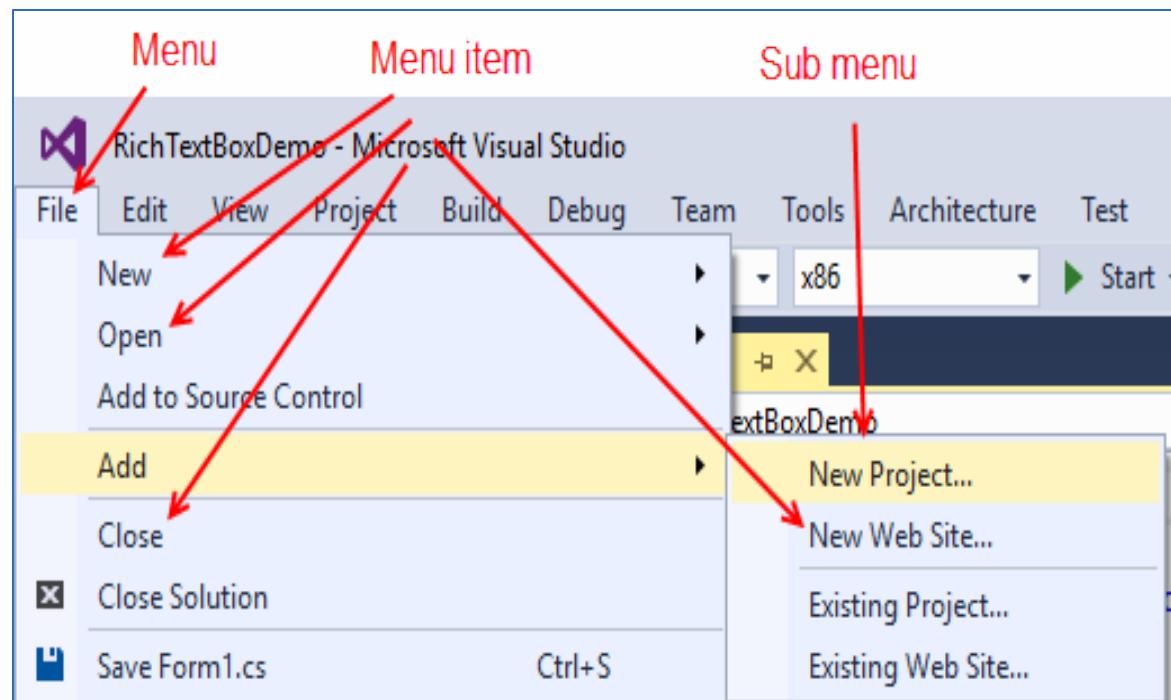
## 4.8 Thêm các control lúc chương trình thực thi

- Khai báo và tạo đối tượng control muốn thêm vào form.
- Thiết lập các thuộc tính cho đối tượng:
  - vị trí: Location, X, Y
  - kích thước: Size, Width, Height
  - ...
- Khai báo sự kiện cho control nếu cần.
- Thêm đối tượng control vào danh sách Controls của Form hoặc Control chứa (Panel, GroupBox, TabControl...).

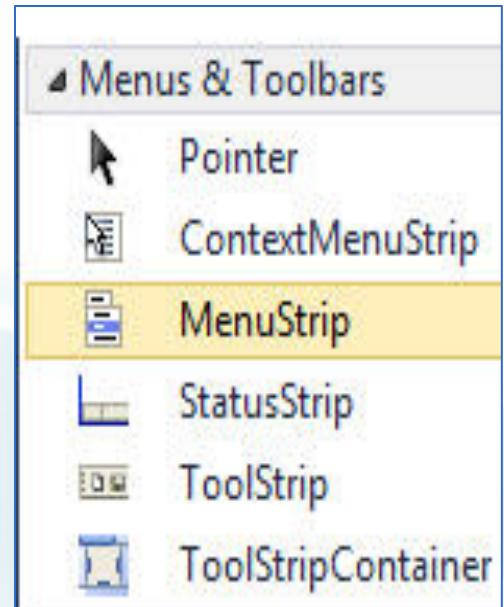


## 4.9 Menu

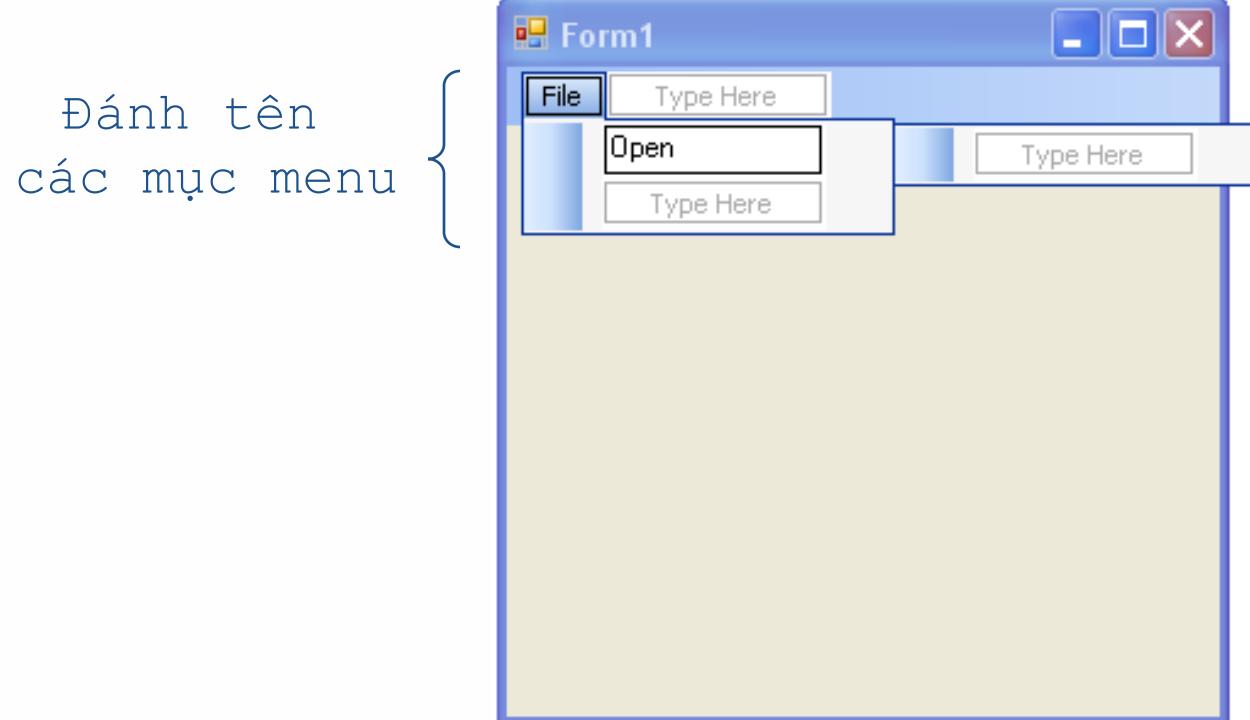
- Control cho phép tổ chức các chức năng xử lý của ứng dụng theo nhóm.
- Menu được tổ chức phân cấp
  - Menu
  - Sub menu
  - Menu item



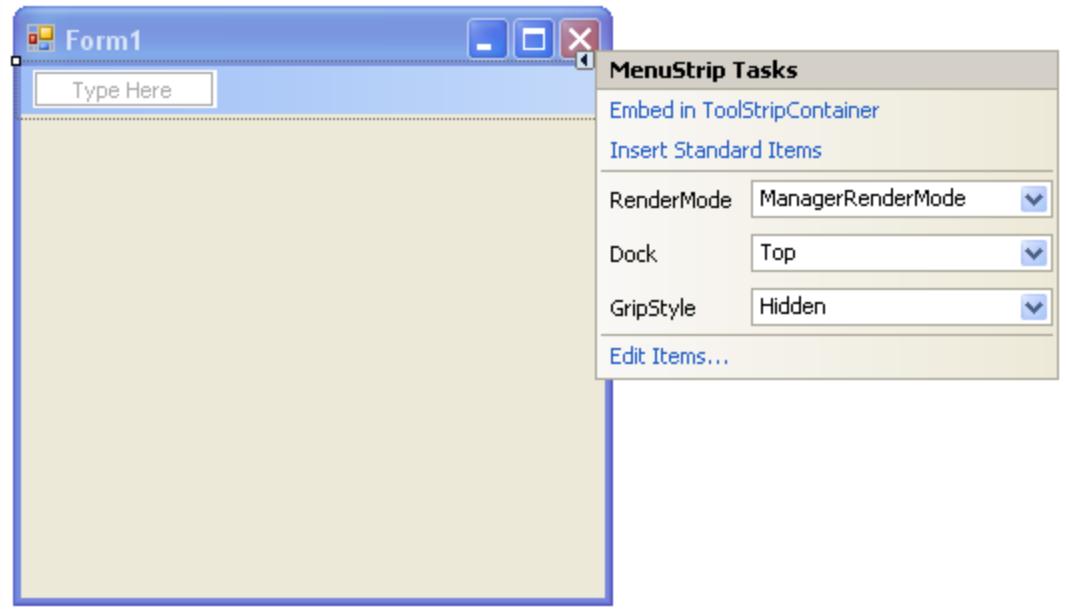
- Tạo menu: sử dụng điều khiển ToolStrip
- Một số thuộc tính của Menu
  - Text: chuỗi hiển thị
  - Shortcut Keys: phím nóng kết hợp với menu
  - Image: hình ảnh hiển thị trên menu
  - AutoTooltip,...
- Các loại menu item
  - MenuItem
  - ComboBox
  - TextBox
  - Separator
- Sự kiện mặc định: Click



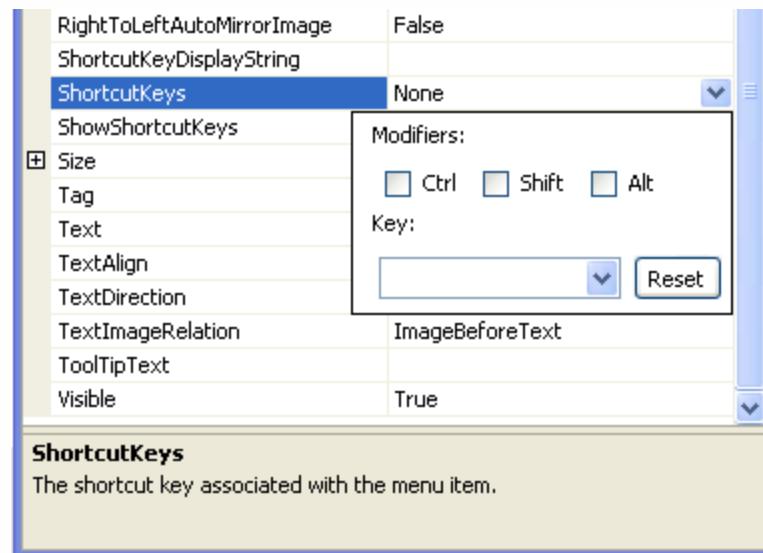
- Tự thiết kế menu
  - HotKey: thêm ký tự & trước ký tự muốn làm hot key



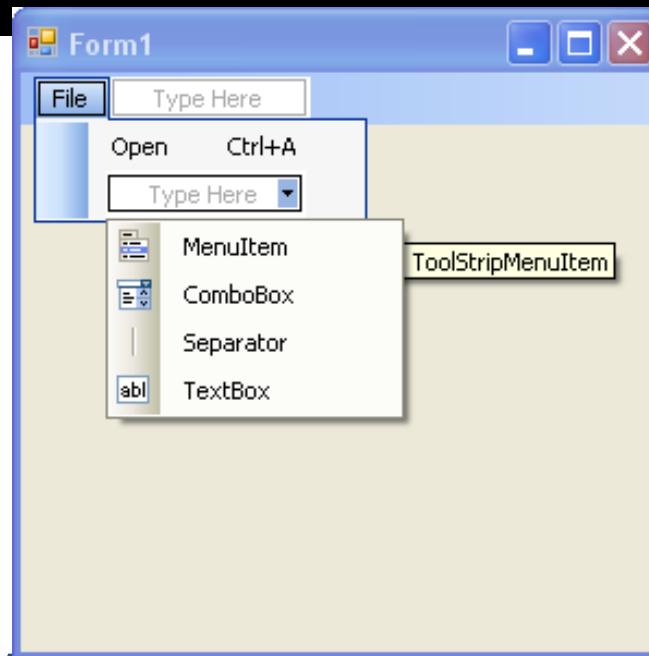
- Thêm menu chuẩn
  - Click vào ký hiệu tam giác → Insert Standard Items



- Tạo Shortcut key
  - Chọn mục menu
  - Vào properties thiết lập ShortcutKeys



- Các loại menu item
  - MenuItem
  - ComboBox
  - TextBox
  - Separator



- Chú ý: Click mũi tên xuống tại nơi muốn tạo menu item để chọn các loại menu item

- Xử lý sự kiện:
  - Xử lý sự kiện cho từng mục menu item
    - Click
    - DoubleClick
    - CheckedChanged
    - CheckStateChanged



# Tóm tắt cấu trúc menu

**MainMenu**

**MenuItem**

**Menu Property**

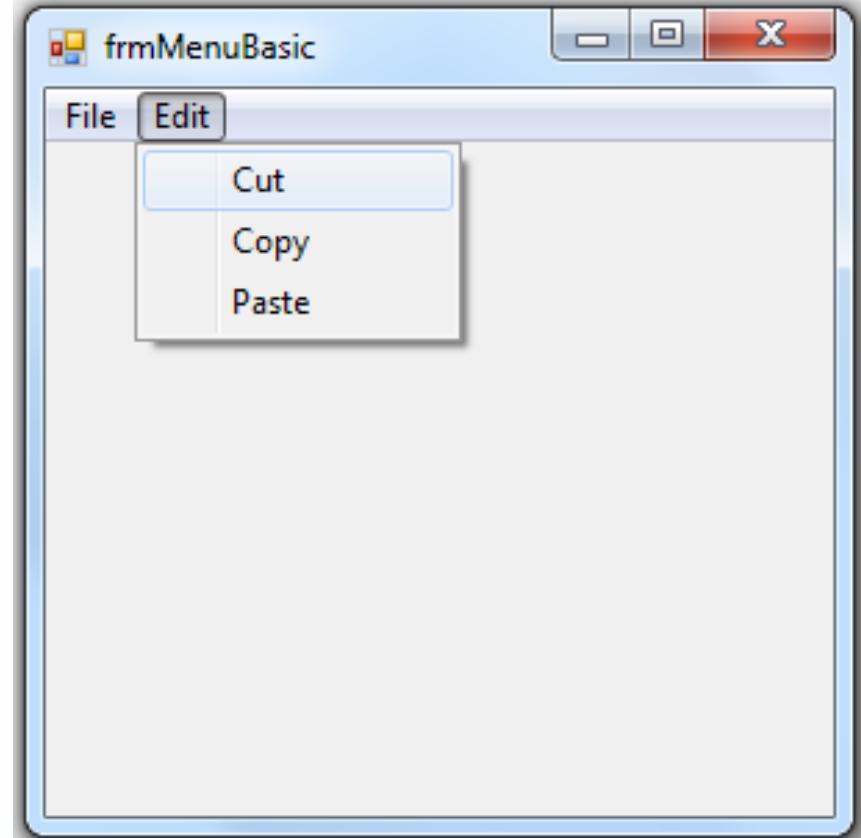
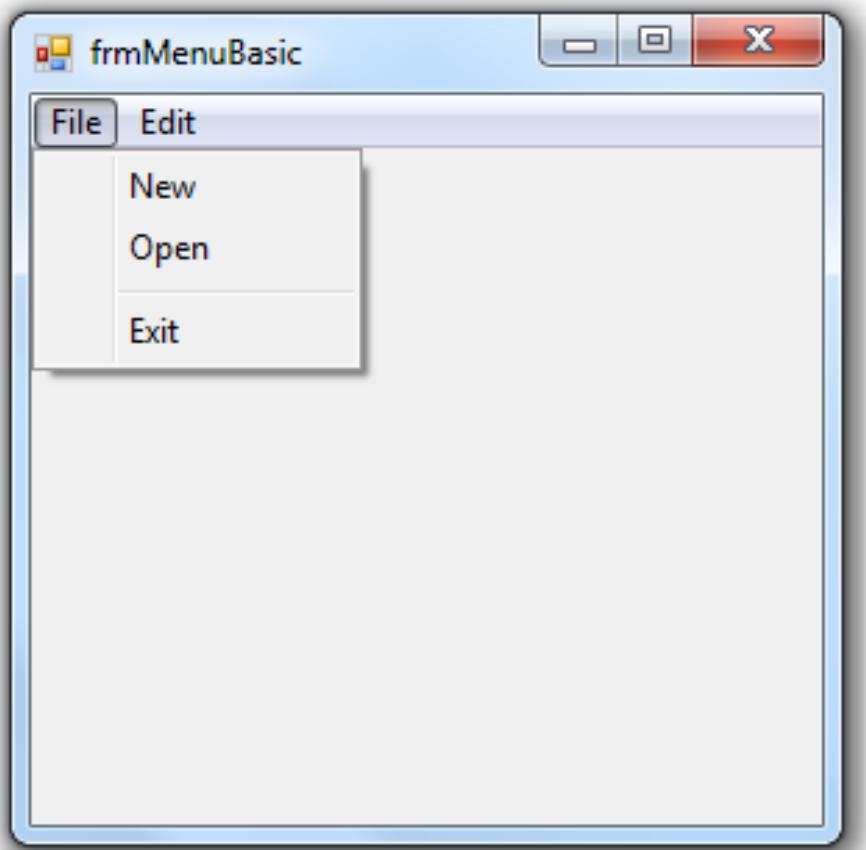
**MenuStrip**

**ToolStripMenuItem**

**MainMenuStrip  
Property**



# Tạo menu (runtime)





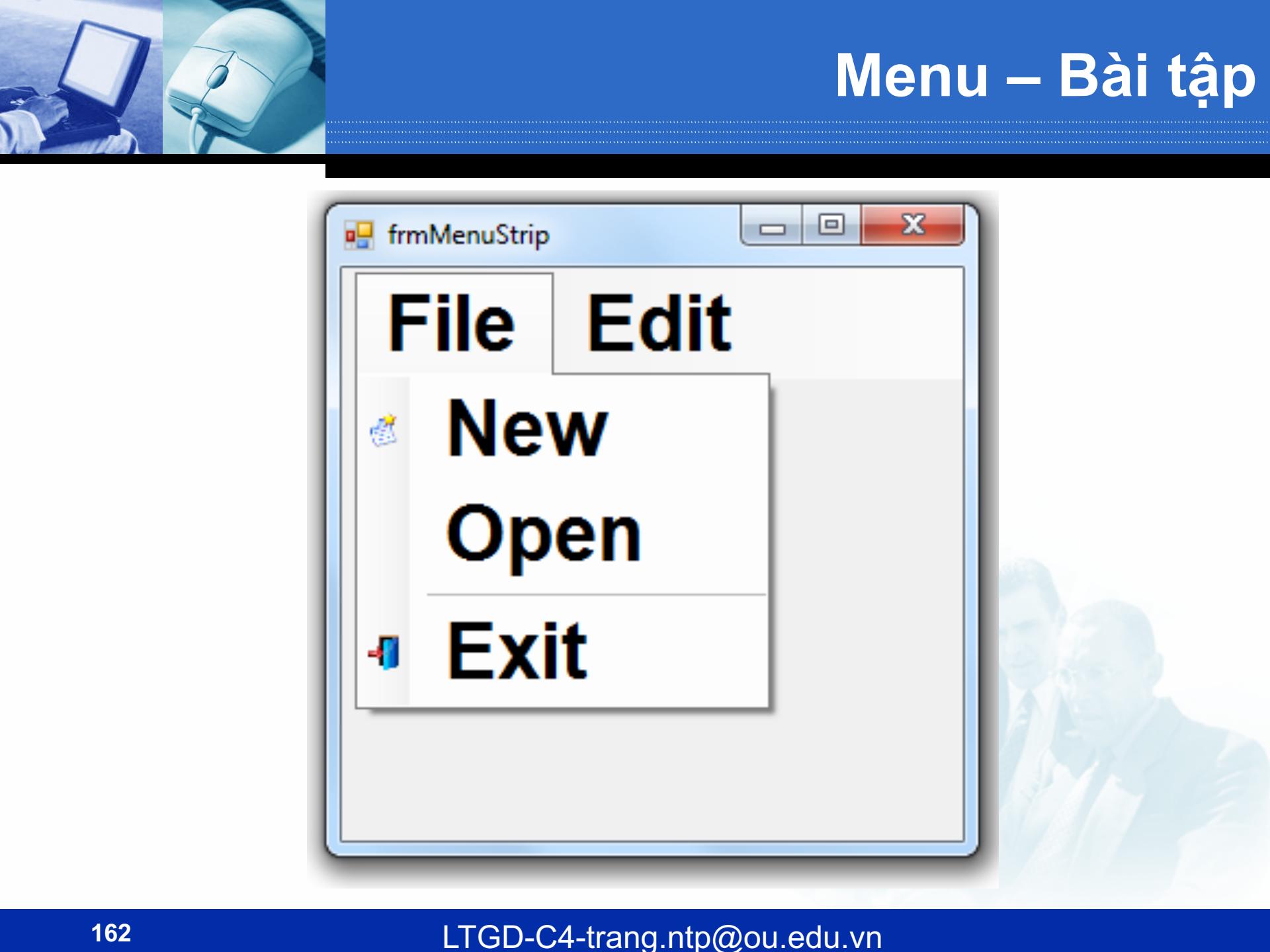
# Tạo menu (runtime)

```
private MainMenu mainMenuBar;
private MenuItem menuFile, menuEdit,
menuFileNew, menuFileOpen, menuFileExit,
menuEditCut, menuEditCopy, menuEditPaste;
private void createMenu()
{mainMenuBar = new MainMenu();
this.Menu = mainMenuBar;
menuFile=new MenuItem("File");
menuFileNew = new MenuItem("New");
menuFileOpen = new MenuItem("Open");
menuFileExit = new MenuItem("Exit");
menuFile.MenuItems.Add(menuFileNew);
menuFile.MenuItems.Add(menuFileOpen);
```

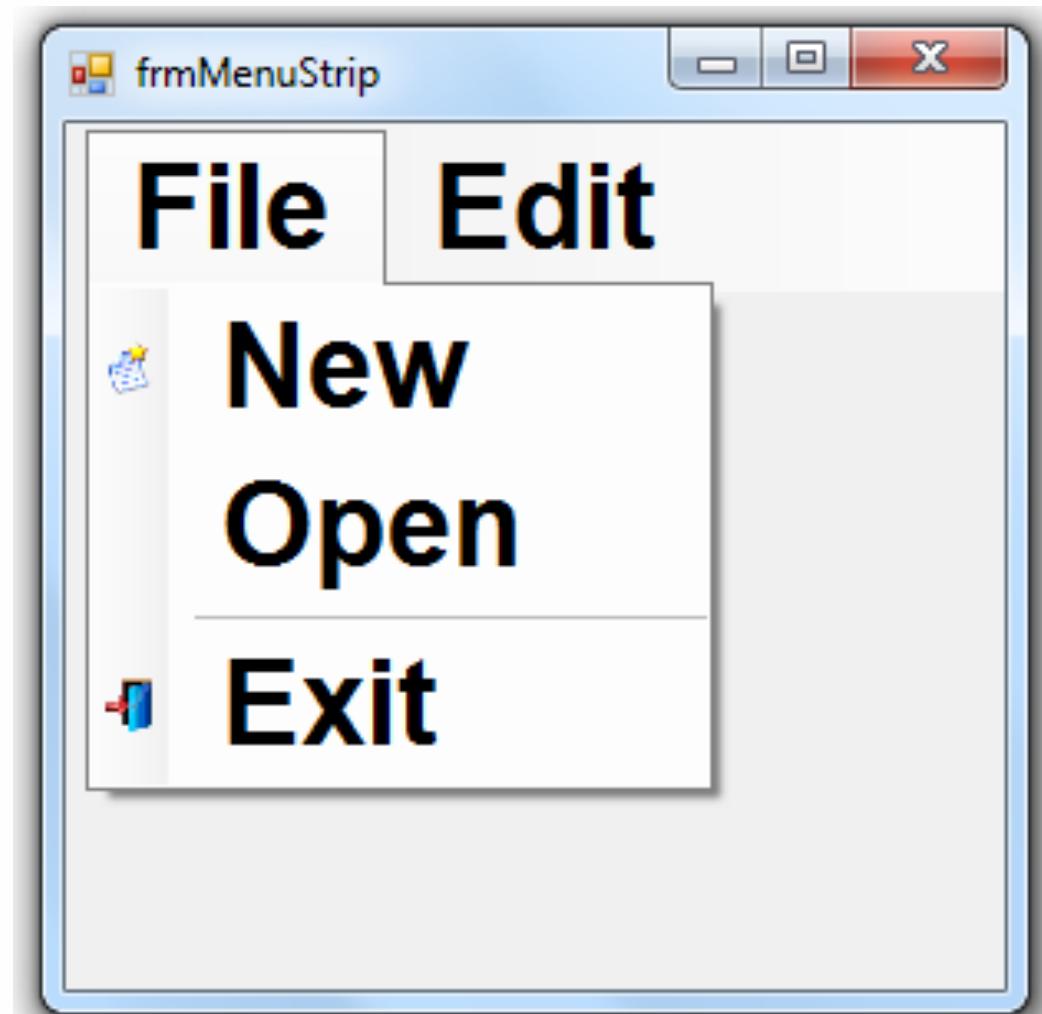


# Tạo menu (runtime)

```
menuFile.MenuItems.Add("-");
menuFile.MenuItems.Add(menuFileExit);
mainMenuBar.MenuItems.Add(menuFile);
menuEdit = new MenuItem("Edit");
menuEditCut = new MenuItem("Cut");
menuEditCopy = new MenuItem("Copy");
menuEditPaste = new MenuItem("Paste");
menuEdit.MenuItems.AddRange(new MenuItem[]
 { menuEditCut, menuEditCopy, menuEditPaste});
mainMenuBar.MenuItems.Add(menuEdit);
}
```



# Menu – Bài tập



# Designer



The screenshot shows the Windows Forms Designer interface. On the left is the Toolbox, which contains various Windows Forms controls like TextBox, ToolTip, TreeView, etc., under the Containers category, and Menus & Toolbars category, where **MenuStrip** is highlighted with a red box. In the center is the **frmMenu.cs [Design]** window, showing a single **menuStrip1** control on the form. On the right is the Properties window, which displays properties for **menuStrip1**. The **Items** property is selected and highlighted with a red box. A blue arrow points from the **Items** property in the Properties window towards the **Items** collection in the code editor area below.

frmMenu.cs [Design] X

Properties

menuStrip1 System.Windows.Forms.MenuStrip

Items (Collection)

LayoutStyle HorizontalStackWith

Location 0, 0

Locked False

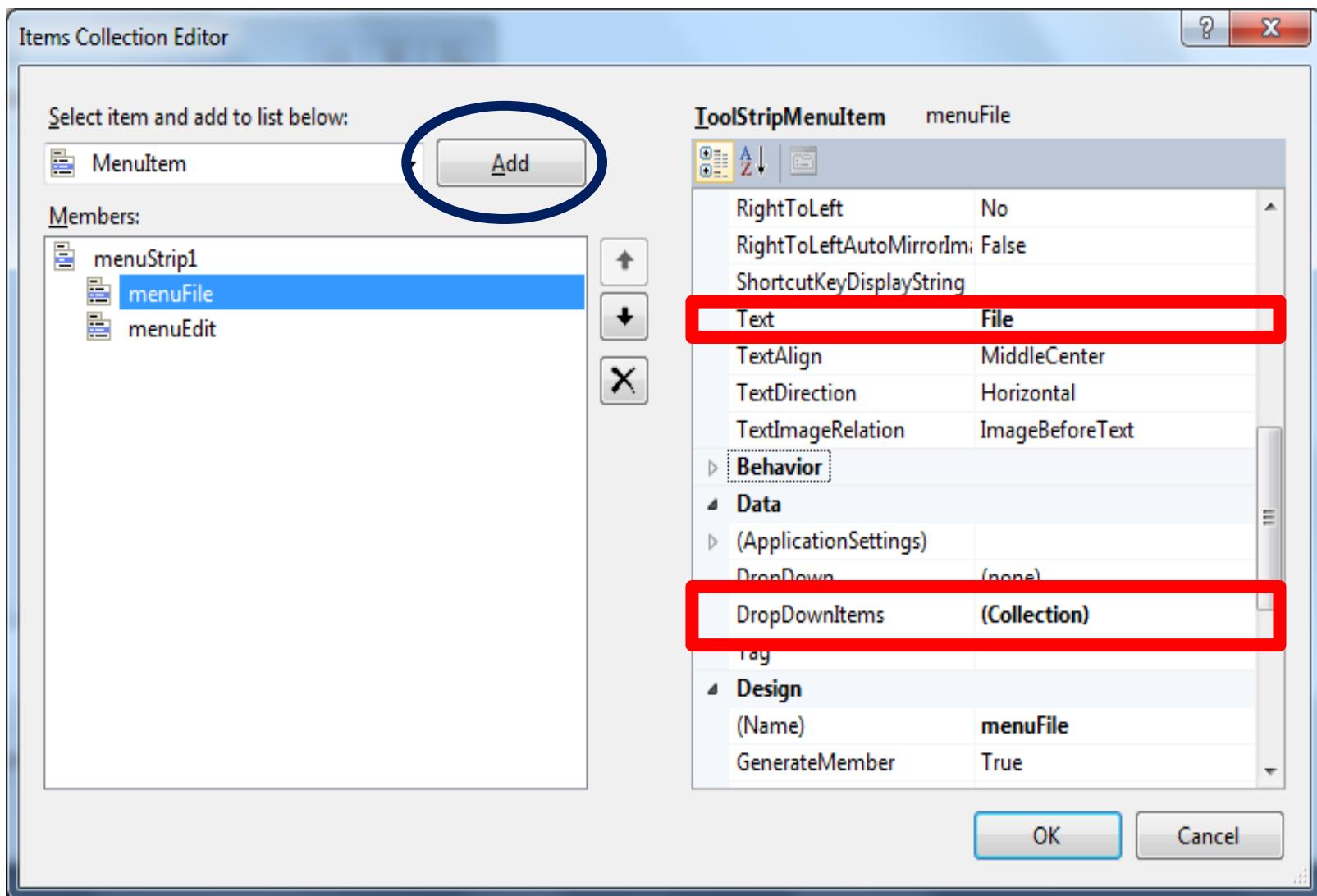
Margin 0, 0, 0, 0

Items

Collection of items to display on the ToolStrip.

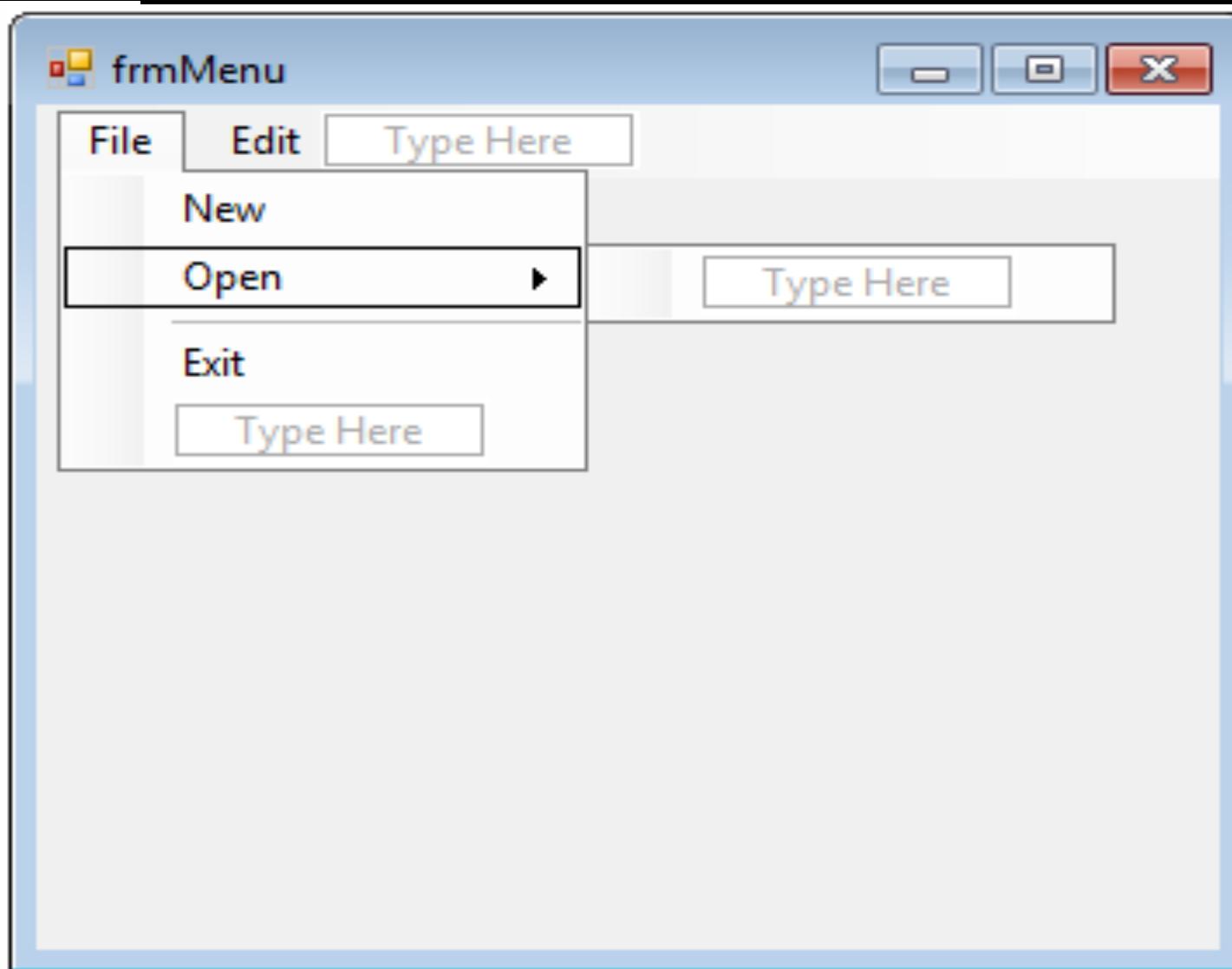
**Click button để  
thêm MenuItem**

# Thêm MenuItem





# Bài tập



- Tạo ImageList chứa các icon.
- Tạo MenuStrip (container) và gán cho MainMenuStrip.
- Tạo ToolStripMenuItem: cấp bậc các menultem. (Item và DropDownList)



# Menu – Ví dụ

```
private ImageList ImageList1;
private ToolStripMenuItem menuFile, menuEdit,
menuFileNew, menuFileOpen, menuFileExit,
menuEditCut, menuEditCopy, menuEditPaste;
```



# Menu – Ví dụ

```
imageList1 = new ImageList();
// Cấu hình ImageList
imageList1.ImageSize = new System.Drawing.Size(16, 16);
// Lấy các file trong thư mục hiện tại
string[] iconFiles =
 Directory.GetFiles(Application.StartupPath, "*.ico");
foreach (string iconFile in iconFiles)
{
 Icon newIcon = new Icon(iconFile);
 imageList1.Images.Add(newIcon);
}
```

```
private void createMenu()
{menuBar = new MenuStrip();
menuBar.Font = new Font("arial", 36,
FontStyle.Bold, GraphicsUnit.Pixel);
this.MainMenuStrip = menuBar;
menuFile = new ToolStripMenuItem("File");
menuFileNew = new ToolStripMenuItem("New");
menuFileNew.Image = imageList1.Images[0];
```



# Menu – Ví dụ

```
menuFileOpen = new ToolStripMenuItem("Open");
ToolStripSeparator sp = new
ToolStripSeparator();
menuFileExit = new ToolStripMenuItem("Exit");
menuFileExit.Image = imageList1.Images[1];
menuFile.DropDownItems.Add(
menuFileNew);
menuFile.DropDownItems.Add(
menuFileOpen);
```



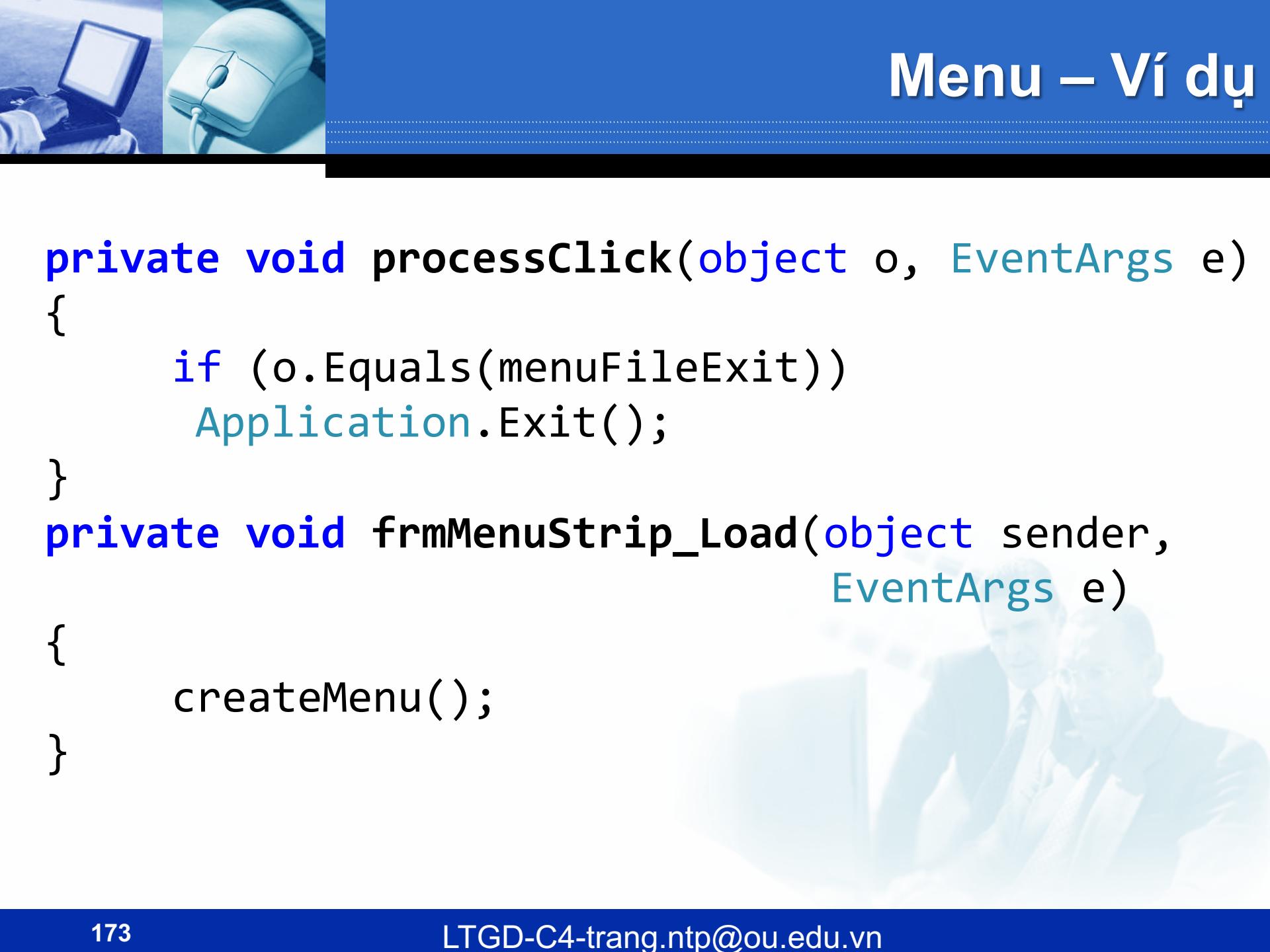
# Menu – Ví dụ

```
menuFile.DropDownItems.Add(sp);
menuFile.DropDownItems.Add(
menuFileExit);
menuEdit = new ToolStripMenuItem("Edit");
menuEditCut = new ToolStripMenuItem("Cut");
menuEditCopy = new ToolStripMenuItem("Copy");
menuEditPaste = new
ToolStripMenuItem("Paste");
```



## Menu – Ví dụ

```
menuEdit.DropDownItems.AddRange(new
ToolStripItem[] { menuEditCut, menuEditCopy,
menuEditPaste});
menuBar.Items.AddRange(new ToolStripItem[] {
menuFile, menuEdit});
this.Controls.Add(menuBar);
attachEvents();
}
private void attachEvents()
{menuFileExit.Click += processClick; }
```



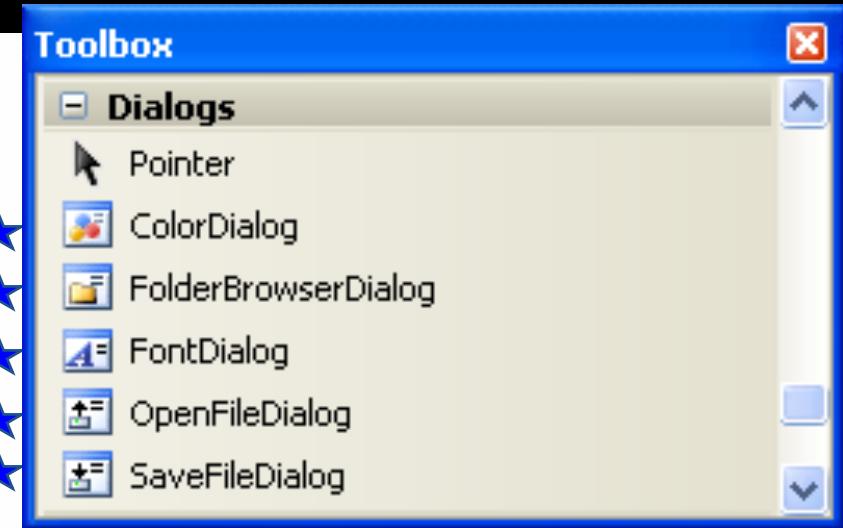
# Menu – Ví dụ

```
private void processClick(object o, EventArgs e)
{
 if (o.Equals(menuFileExit))
 Application.Exit();
}
private void frmMenuStrip_Load(object sender,
 EventArgs e)
{
 createMenu();
}
```



# Các dialog thông dụng

- Các loại Dialog
  - Custom Dialog Boxes
  - Common Dialog Boxes
    - OpenFileDialog
    - SaveFileDialog
    - FontDialog
    - ColorDialog
    - FolderBrowserDialog
    - PageSetUpDialog
    - PrintPreviewDialog
    - PrintDialog





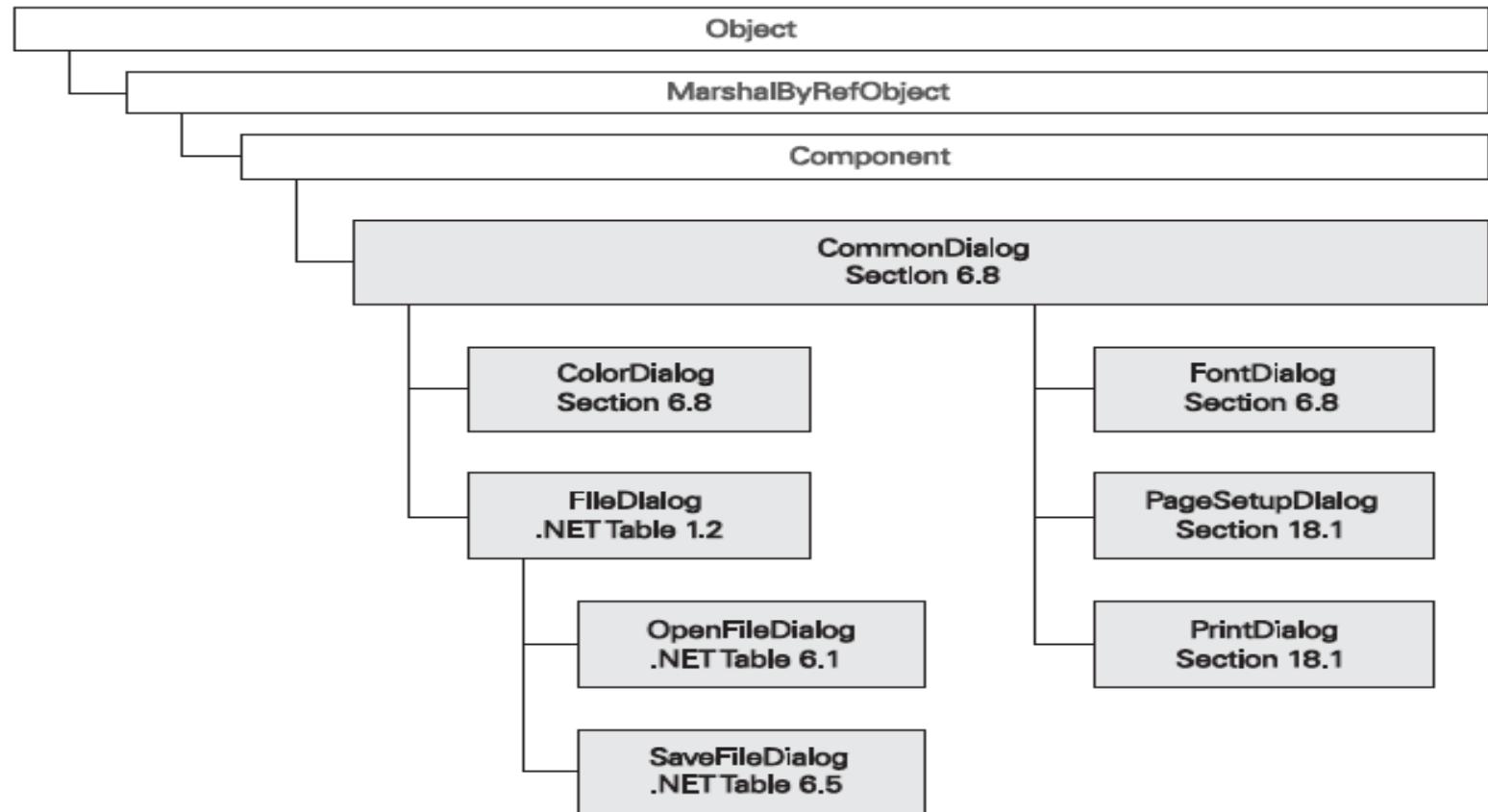
## 4.10 Các hộp thoại thông dụng của Windows

- Là những lớp hộp thoại được thiết kế cho các mục đích sử dụng khác nhau như:
  - Hộp thoại mở file, lưu file
  - Hộp thoại chọn font chữ
  - Hộp thoại chọn màu
  - Hộp thoại chọn thư mục
  - Hộp thoại in ấn
  - ....



# Các hộp thoại thông dụng của Windows (tt)

## *Common dialogs*





# Sử dụng các hộp thoại

- Trong thiết kế:
  - Kéo thả vào từ thanh Toolbox
  - Thiết lập các thuộc tính trong cửa sổ Properties
  - Gọi phương thức ShowDialog, kiểm tra kiểu trả về của phương thức ShowDialog để xử lý:
    - DialogResult.OK: chấp nhận thao tác
    - DialogResult.Cancel: hủy thao tác



# Sử dụng các hộp thoại (tt)

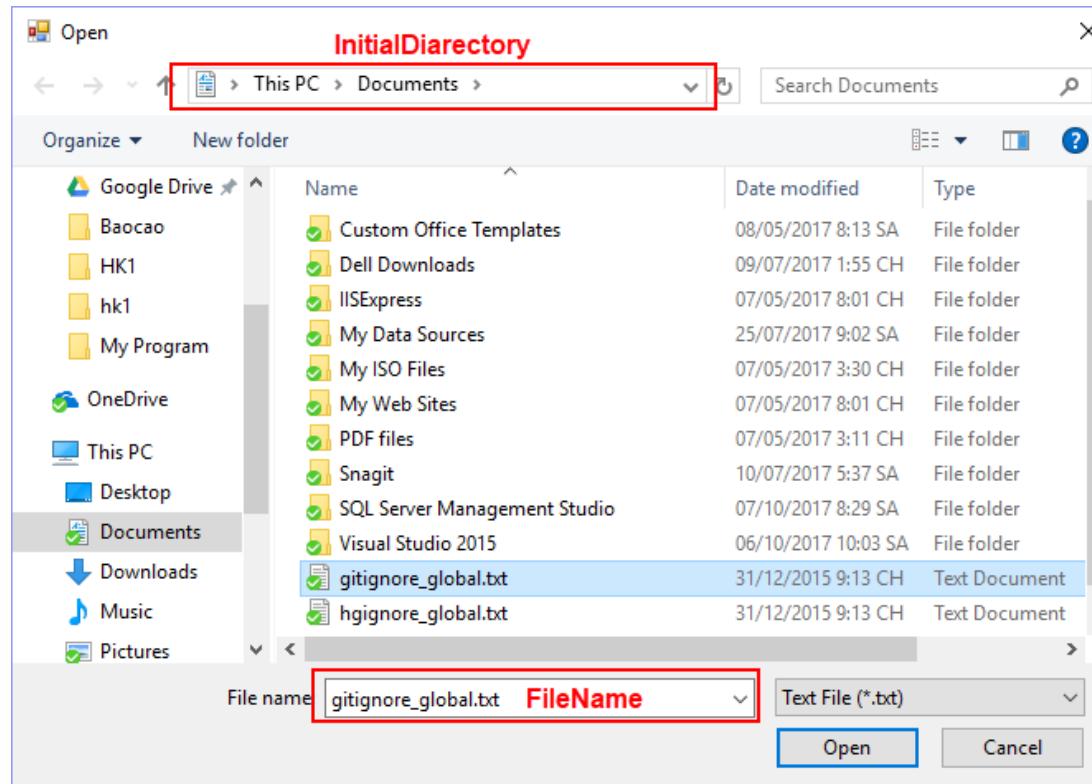
- Sử dụng code:
  - Khai báo đối tượng của lớp hộp thoại
  - Thiết lập các thuộc tính cho đối tượng
  - Gọi phương thức ShowDialog, kiểm tra kiểu trả về của phương thức ShowDialog để xử lý:
    - DialogResult.OK: chấp nhận thao tác
    - DialogResult.Cancel: hủy thao tác
  - Ví dụ:

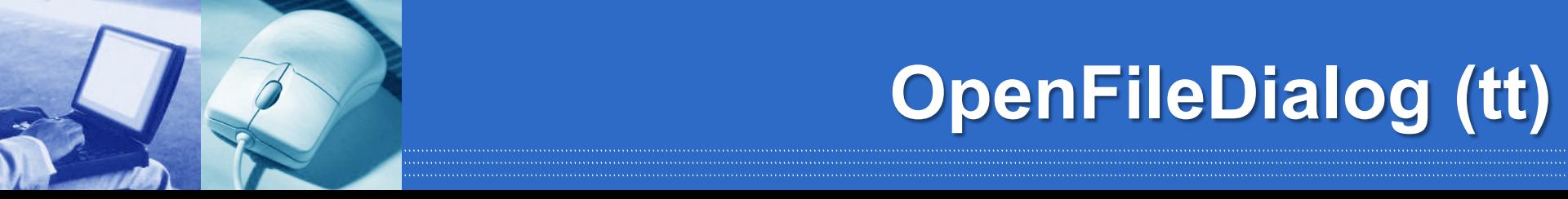
```
ColorDialog dlg =new ColorDialog();
if(dlg.ShowDialog()==DialogResult.OK) {....}
```



# OpenFileDialog

- Hộp thoại cho phép chọn và mở file
- Lớp OpenFileDialog kế thừa từ lớp OpenFileDialog





# OpenFileDialog (tt)

- Các thuộc tính cơ bản:

- Filter: chuỗi quy định loại file được hiển thị trong danh sách các file cho phép người sử dụng chọn.
- Multiselect: true/false: cho phép chọn nhiều file hoặc chỉ chọn một file, mặc định là false.
- FileName: file được chọn kiểu string (sử dụng trong trường hợp thuộc tính Multiselect = false)
- FileNames: danh sách các file được chọn kiểu string (sử dụng trong trường hợp thuộc tính Multiselect = true).



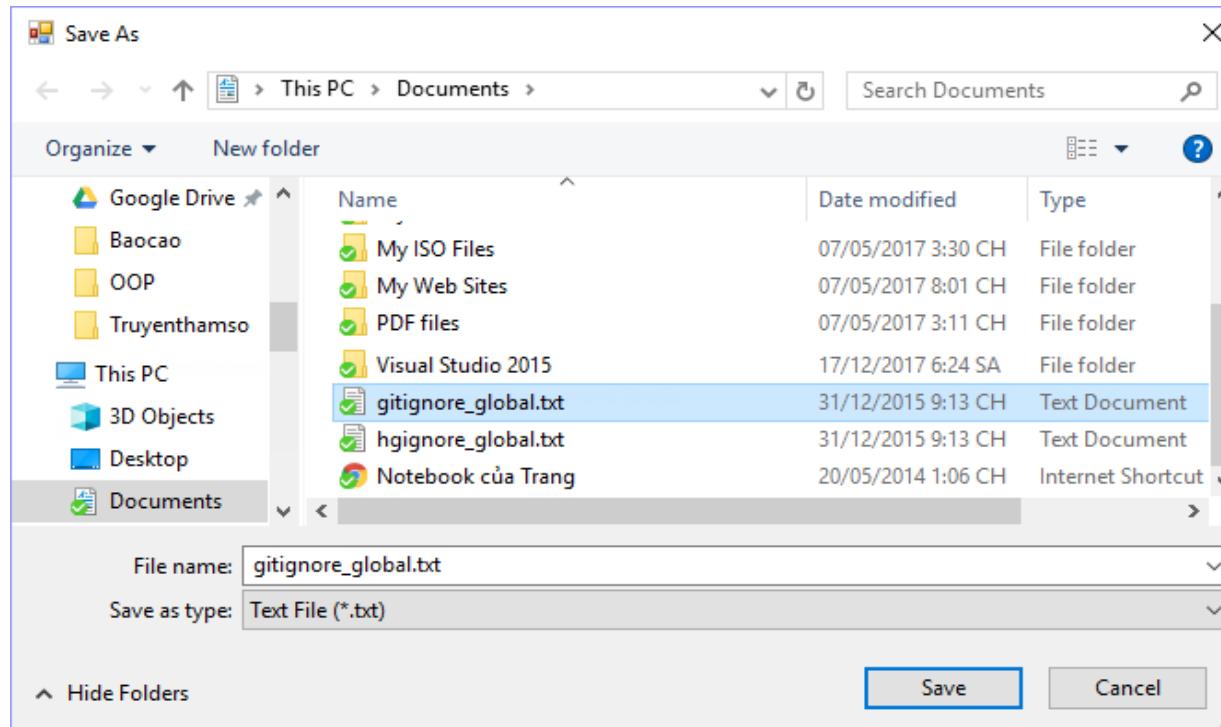
# OpenFileDialog (tt)

- Cách sử dụng hộp thoại OpenFileDialog:
  - Kéo biểu tượng OpenFileDialog từ cửa sổ Toolbox vào Form hoặc khai báo và tạo đối tượng bằng code.
  - Thiết lập các thuộc tính cho hộp thoại.
  - Gọi phương thức ShowDialog, truy xuất thuộc tính FileName hoặc FileNames của đối tượng hộp thoại để xử lý.



# SaveFileDialog

- Tương tự OpenFileDialog, kế thừa từ lớp FileDialog, là hộp thoại cho phép lưu file lên đĩa, không có thuộc tính Multiselect

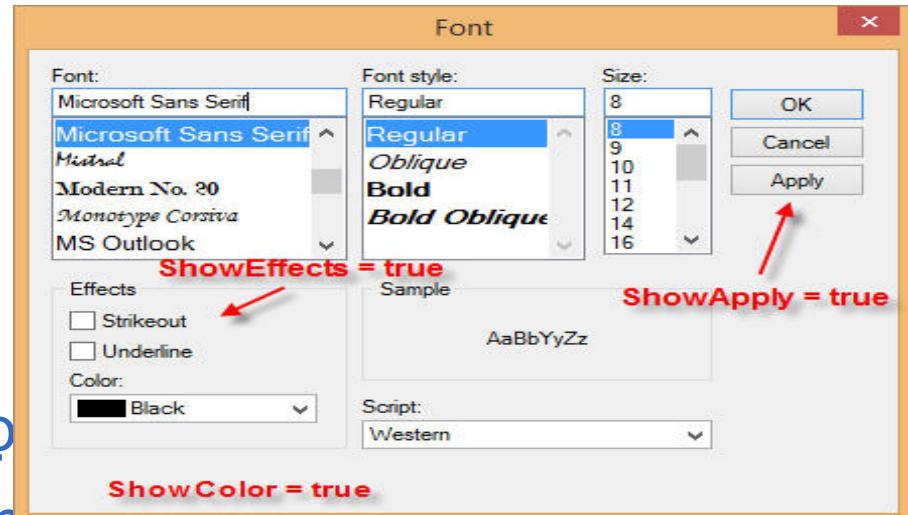




# FontDialog

- Hộp thoại cho phép chọn font chữ.

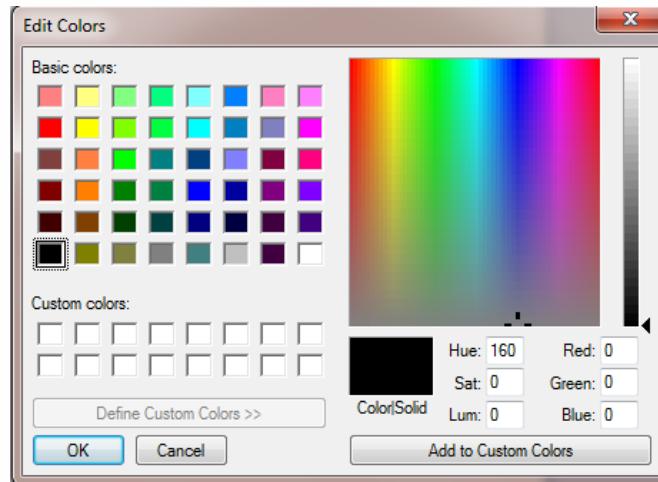
- Các thuộc tính cơ bản:
  - Font: font chữ được chọn
  - Color: màu chữ được chọn.
  - ShowEffects: true/false - hiển thị/không hiển thị khung Effects.
  - ShowApply: true/false - hiển thị/không hiển thị nút Apply.
  - ShowColor: true/false - hiển thị/không hiển thị combobox chọn màu chữ.





# ColorDialog

- Họp thoại cho phép chọn màu
- Các thuộc tính
  - AllowFullOpen: true/false - cho/không cho phép người sử dụng định nghĩa một màu tùy chọn.
  - FullOpen: true/false - cho/không cho phép mở hộp thoại màu dạng đầy đủ.





# Các dialog thông dụng: ColorDialog

- Properties
  - Color
  - AllowFullOpen: bool
  - FullOpen: bool
- Methods
  - DialogResult ShowDialog()
  - void Reset()



# Các dialog thông dụng: ColorDialog

```
private void btChangeForeColor_Click(object sender, EventArgs e)
{
 ColorDialog dlg = new ColorDialog();
 dlg.FullOpen = true;
 if (dlg.ShowDialog() == DialogResult.OK)
 {
 txtView.ForeColor = dlg.Color;
 }
}
```



# Các dialog thông dụng: FolderBrowserDialog

- FolderBrowserDialog dùng để chọn thư mục có trên máy





# Các dialog thông dụng: FolderBrowserDialog

- Properties
  - Description: string
  - RootFolder: Environment.SpecialFolder
  - SelectedPath: string
  - ShowNewFolderButton: bool
- Methods
  - DialogResult ShowDialog()
  - void Reset()

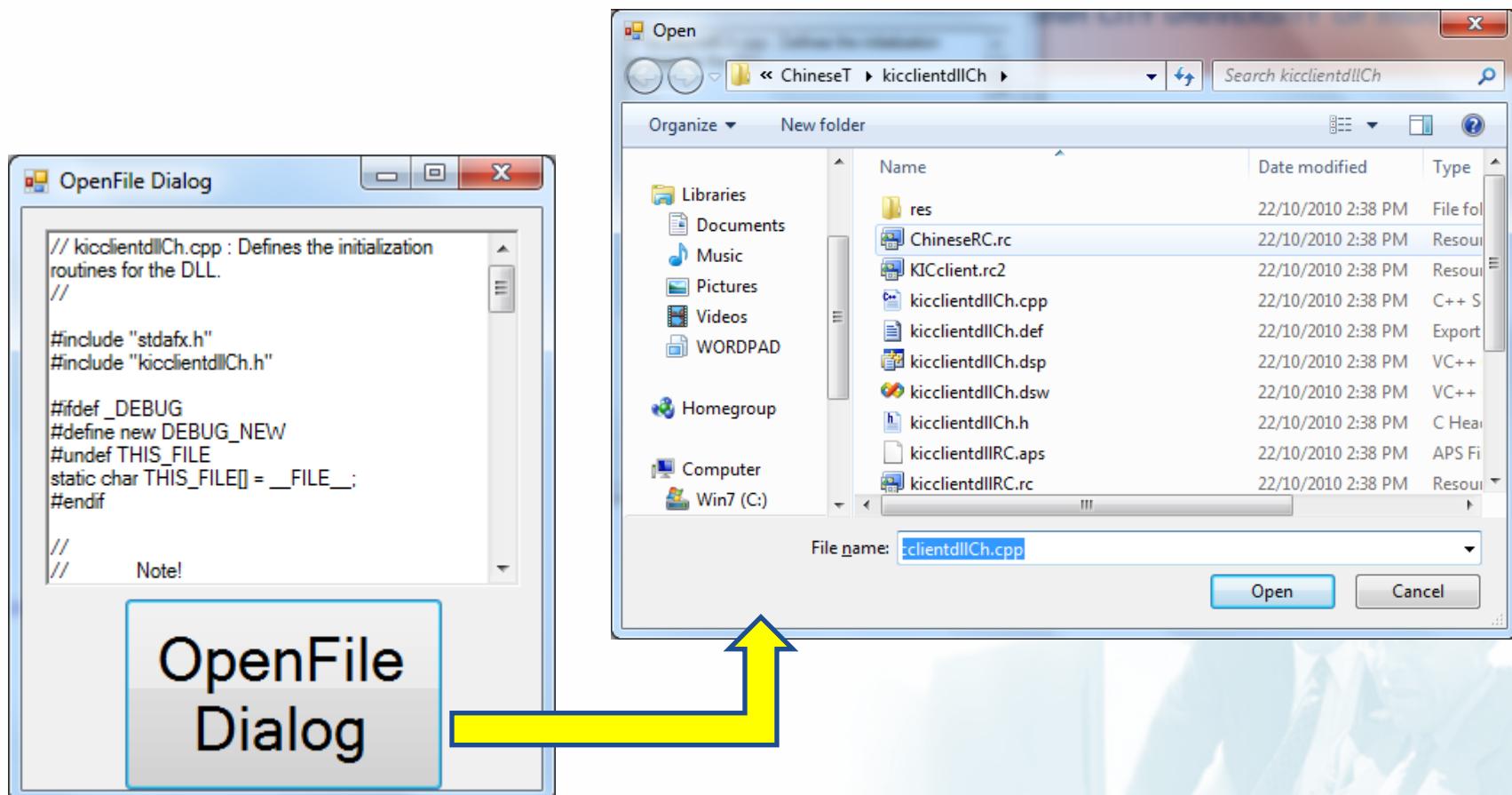


# Các dialog thông dụng: FolderBrowserDialog

```
FolderBrowserDialog dlg = new FolderBrowserDialog();
dlg.Description = "Select a folder...";
dlg.ShowNewFolderButton = true;
if (dlg.ShowDialog() == DialogResult.OK)
 label1.Text = dlg.SelectedPath;
```

↑  
**Thư mục được chọn**

# Bài tập



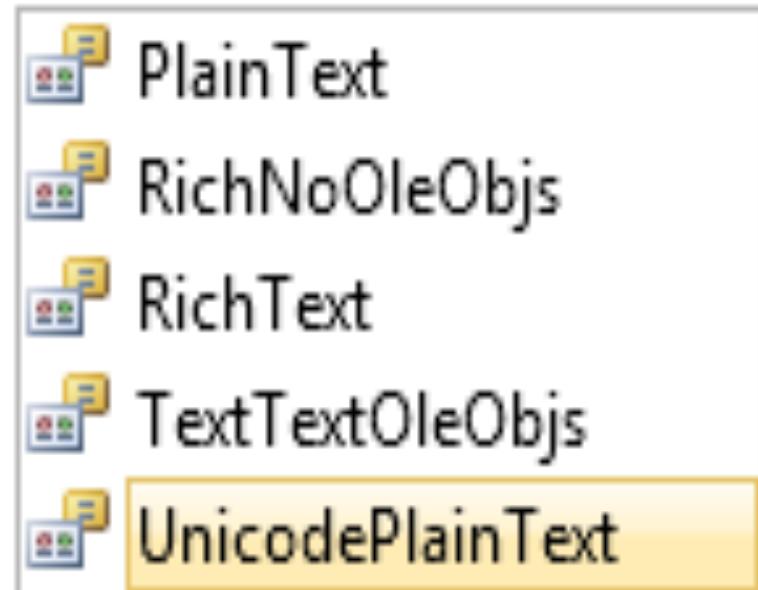
Khai báo một OpenFileDialog

Khai báo các định dạng file (Filter)

Load file (LoadFile(Thuộc tính, Loại file))

```
richTxtFile.LoadFile(openDlg.FileName,
RichTextBoxStreamType.);
```

Chọn loại file  
tương ứng



```
 OpenFileDialog openDlg = new OpenFileDialog();
openDlg.Filter = "(*.txt)|*.txt|(All)|*.*";
if (openDlg.ShowDialog() == DialogResult.OK)
{
 rtFile.LoadFile(openDlg.FileName,
 RichTextBoxStreamType.PlainText);
}
```

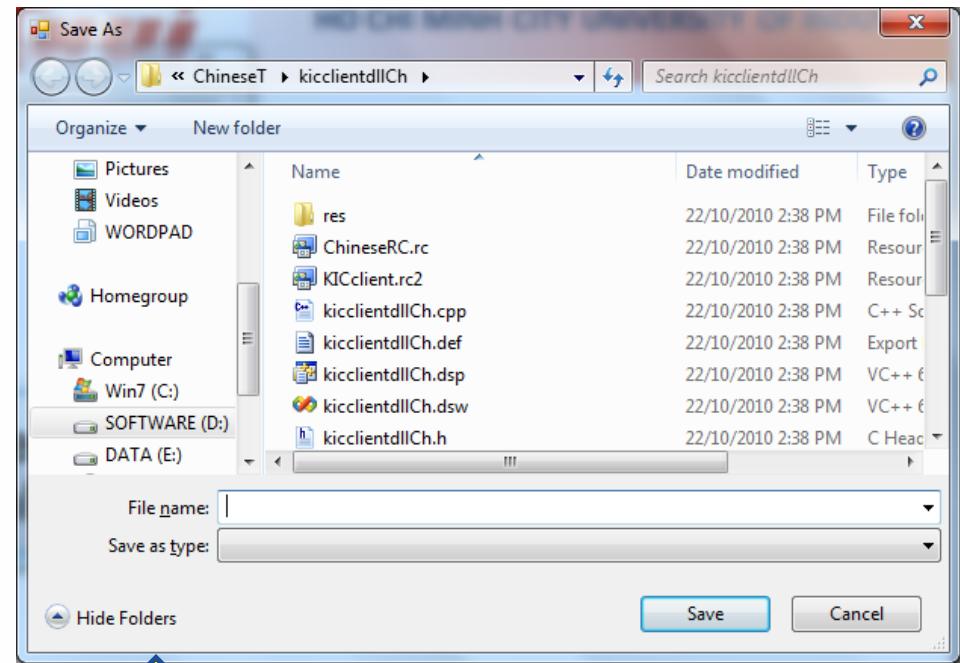
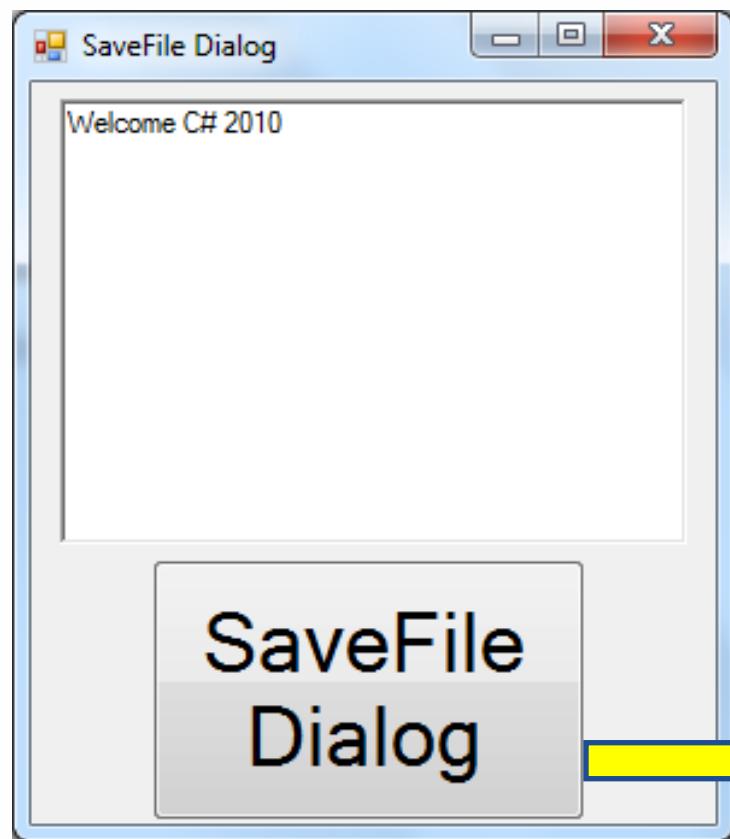


# Đọc file - StreamFile

```
using System.IO;

if (openDlg.ShowDialog() == DialogResult.OK)
{
 //Open to read
 Stream stream = openDlg.OpenFile();
 StreamReader reader = new
 StreamReader(stream);
 richTxtFile.Text = reader.ReadToEnd();
 reader.Close();
}
```

# Lưu file



Khai báo một SaveFileDialog

Khai báo các định dạng file (Filter)

Lưu file (SaveFile)(Thuộc tính, Loại file)



# Lưu file

```
SaveFileDialog saveDlg = new SaveFileDialog();
saveDlg.Filter = "(*.txt)|*.txt|(All)|*.*";
if (saveDlg.ShowDialog() == DialogResult.OK)
{
 rtFile.SaveFile(saveDlg.FileName, RichTextBoxStreamType.PlainText);
}
```



## Lưu file - StreamFile

```
if (saveDlg.ShowDialog() == DialogResult.OK)
{
 Stream stream = saveDlg.OpenFile(); //Open to
 Write
 StreamWriter writer = new StreamWriter(stream);
 writer.WriteLine(rtFile.Text);
 writer.Close();
}
```



# Ứng dụng SDI - MDI

- SDI (Single Document interface): là ứng dụng mà tại thời điểm, ta chỉ có thể làm việc với một tài liệu duy nhất.
  - NotePad
  - Paint
- MDI (Multi Document interface): là ứng dụng mà tại thời điểm, ta có thể làm việc với nhiều tài liệu.
  - Word
  - Excel



- Các thao tác thường gặp trong ứng dụng có nhiều form:
  - Truyền dữ liệu giữa các form
  - Chuyển form



# SDI – Truyền dữ liệu giữa Form

- Sử dụng phương thức khởi tạo

```
public partial class Form1 : Form
{
 ...
 private void btOpenForm2_Click(object sender,
 EventArgs e)
 {
 Form2 f2 = new Form2(txtMessage.Text);
 f2.Show();
 }
}
```

```
public partial class Form2 : Form
{
 private string strText;
 public Form2()
 {
 InitializeComponent();
 }
 public Form2(string s):this()
 {
 strText = s;
 lbText.Text = strText ;
 }
}
```



# SDI – Truyền dữ liệu giữa Form

- Sử dụng Property

```
public partial class Form3 : Form
{
 private string strText;
 public string TEXT
 {
 get { return strText; }
 set { strText = value; }
 }
 private void Form3_Load(object sender, EventArgs e)
 {
 lbText.Text = strText;
 }
}
```

```
public partial class Form1 : Form
{
 private void btOpenForm3_Click(object sender,
 EventArgs e)
 {
 Form3 f3 = new Form3();
 f3.TEXT = txtMessage.Text;
 f3.Show();
 }
}
```





# SDI – cách chuyển Form

- Các phương thức mở/đóng form:
  - Show(): Khi form đang hiển thị, có thể thao tác với các form khác trong cùng ứng dụng.
  - ShowDialog():
    - Form được gọi, hiển thị dạng modal dialog. Người sử dụng phải thao tác với form, không thể thao tác với các form khác trong cùng ứng dụng trước khi đóng form.
    - Trả về đối tượng DialogResult, dùng để đáp ứng xác nhận hành động của người sử dụng.
  - Hide(): ẩn form
  - Mở form:
    - Tạo đối tượng form đã tồn tại
    - Gọi Show() hoặc ShowDialog()



# SDI – cách chuyển Form

- Ví dụ: Mở Form 2 từ Form 1 và ẩn Form 1

```
Form2 f = new Form2()
f.Show();
This.Hide();
```



# SDI – cách chuyển Form

- Sử dụng Property của đối tượng để chuyển form

```
public partial class Form5 : Form
{
 Form form;
 public Form FORM
 {
 get { return form; }
 set { form = value; }
 }
 private void btOpenForm1_Click
 (object sender, EventArgs e)
 {
 Close();
 form.Show();
 }
}
```

```
public partial class Form1 : Form
{
 private void btOpenForm5_Click(
 object sender, EventArgs e)
 {
 Hide();
 Form5 f5 = new Form5();
 f5.FORM = this;
 f5.Show();
 }
}
```



# SDI – cách chuyển Form

- Sử dụng đối tượng static

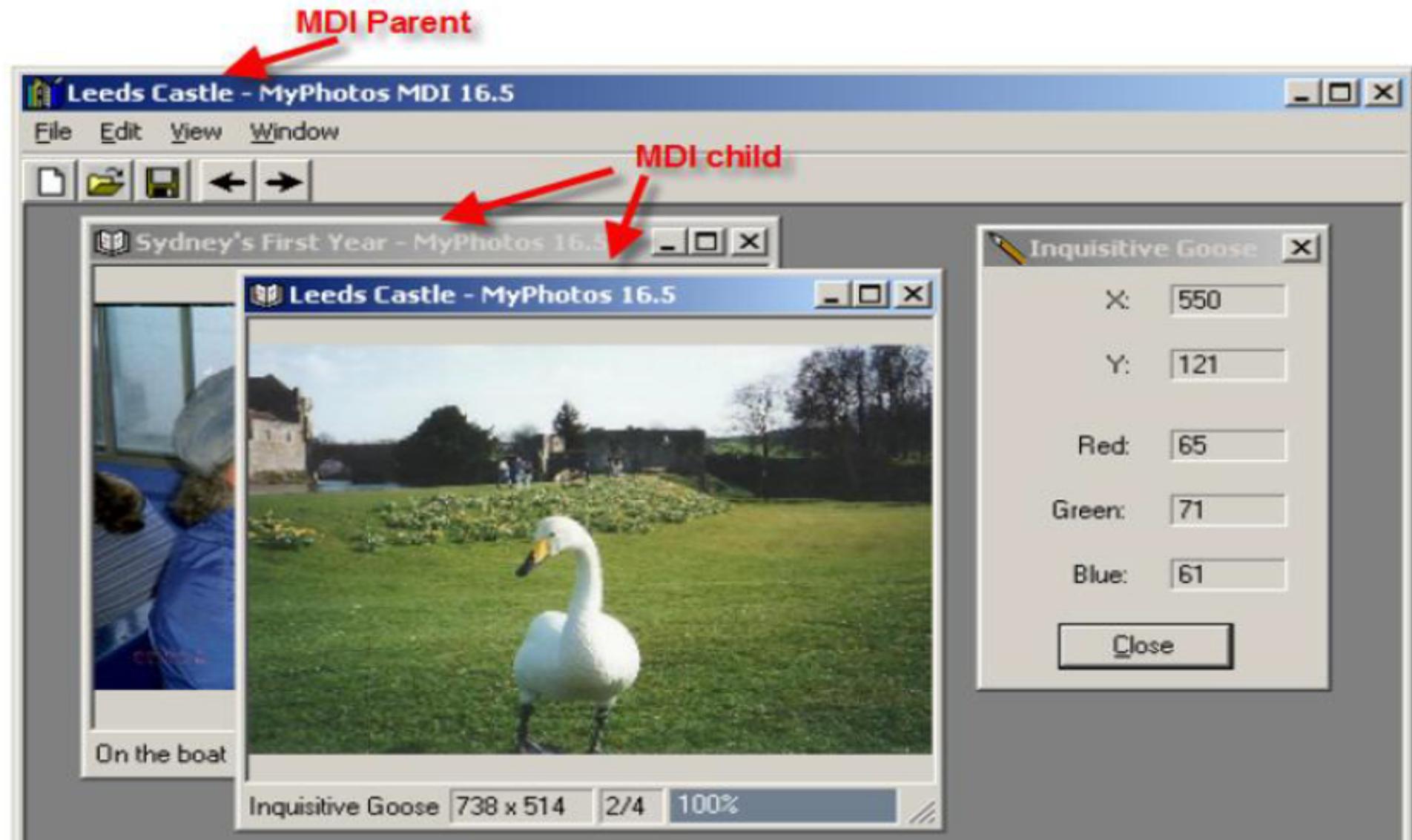
```
public partial class Form1 : Form
{
 public static Form1 Form_1;
 private void btOpenForm6_Click
 (object sender, EventArgs e)
 {
 Form_1 = this;
 Form_1.Hide();
 Form6 f6 = new Form6();
 f6.Show();
 }
}
```

```
public partial class Form6 : Form
{
 public Form6()
 {
 InitializeComponent();
 }
 private void btOpenForm1_Click
 (object sender, EventArgs e)
 {
 Close();
 Form1.Form_1 .Show();
 }
}
```



- Các thành phần cơ bản:
  - Main Form.
  - Child Form.
  - Menu

# MDI





## ▪ Main Form :

- Form chứa các form khác
- Thiết lập thuộc tính `isMdiContainer = true`
- Sử dụng code

```
Form frm=new Form();
Frm. isMdiContainer=true;
Frm.Show();
```



- Child Form :

- Form nằm trong MDI Form
- Khai báo thuộc tính MdiParent ứng với MDI Form
- Sử dụng code

```
Form2 Frm = new Form2();
Frm.MdiParent = this;
Frm.Show();
```

- Từ khóa **this** chỉ form hiện hành.



- Thuộc tính:
  - IsMdiChild: True/False, cho biết có phải là Form con.
  - MdiParent: Đối tượng Form, xác định là Form cha
  - ActiveMdiChild: Trả về form con đang active.
  - IsMdiContainer: True/False
  - MdiChildren: Danh sách các form con
- Sắp xếp form con:
  - Sử dụng phương thức LayoutMdi().
  - MdiLayout.Cascade
  - MdiLayout.TileHorizontal
  - MdiLayout.Vertical



- Duyệt các Child Form

```
public void closeAllChildrenForm()
{
 foreach (Form frms in this.MdiChildren)
 {
 frms.Close();
 }
}
```





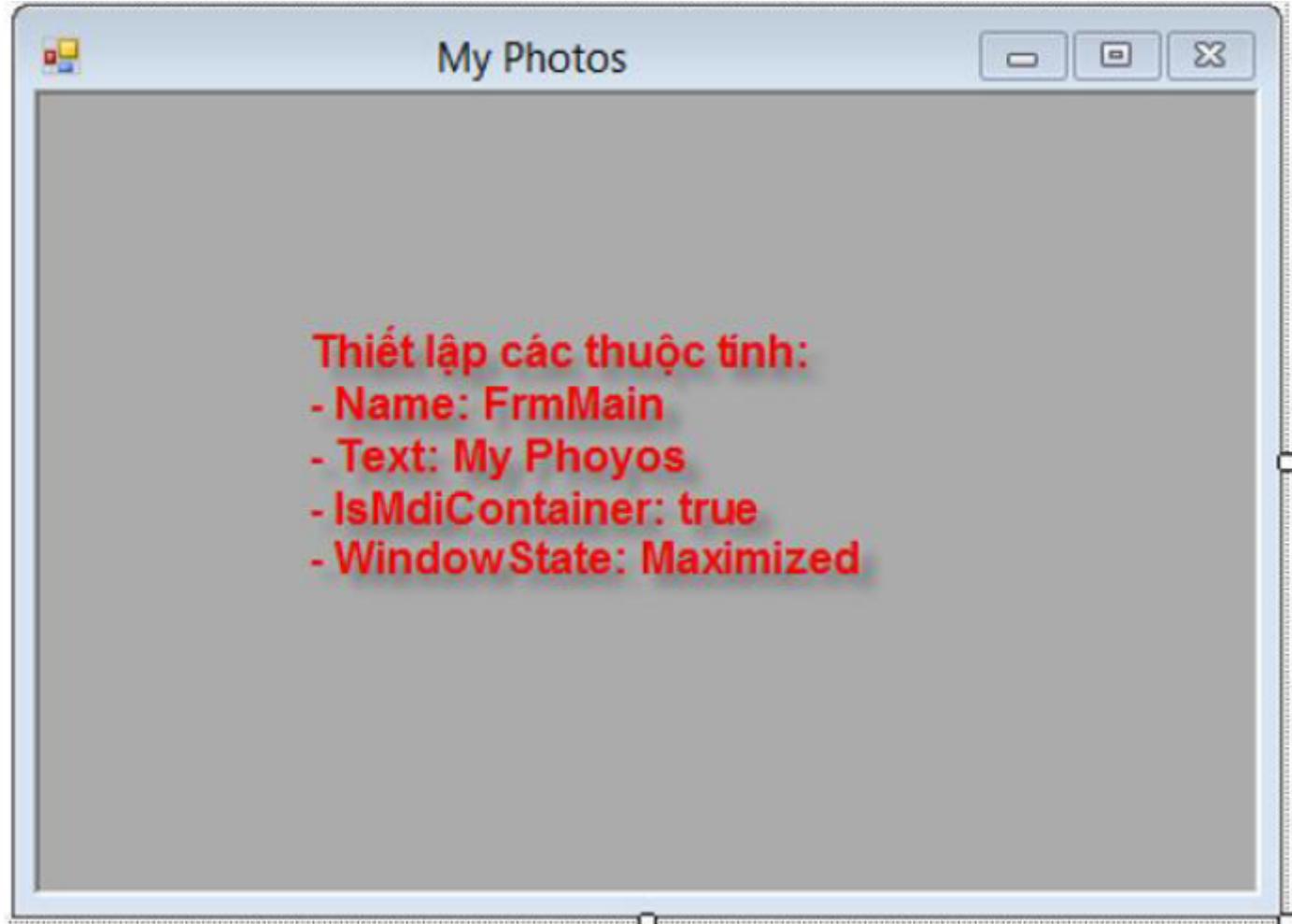
- Các bước tạo ứng dụng MDI:

- Tạo form cha.
- Bổ sung một **New menu item** để gọi các form con.
- Trộn và sắp xếp các menu của form cha và form con.
- Gọi form con trong sự kiện **click** của menu **New**



# Ví dụ tạo ứng dụng MDI

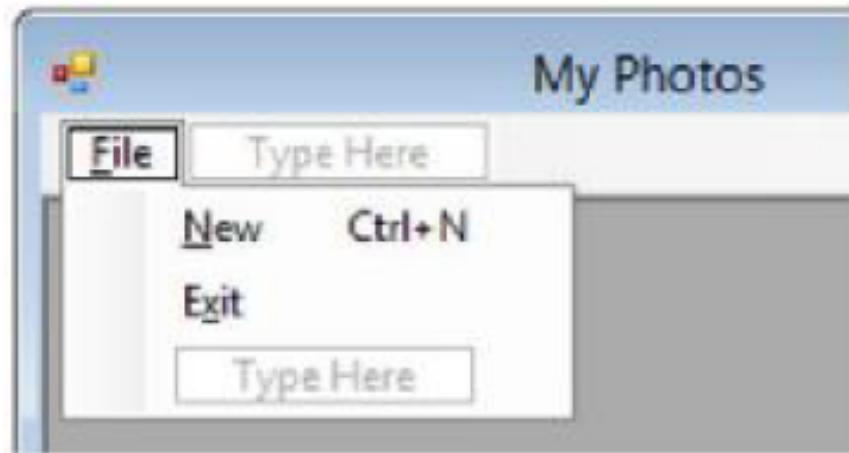
- Tạo Form cha





# Ví dụ tạo ứng dụng MDI

- Thêm điều khiển ToolStrip



- Thêm form mới làm form con (FrmChildForm)
- Thêm vào FrmChildForm một PictureBox (picImage)



# Ví dụ tạo ứng dụng MDI

- Sự kiện Load form

```
private void FrmChildForm_Load(object sender, EventArgs e)
{
 string pathImg = Application.StartupPath + @"\cat.jpg";
 picImage.Image = Image.FromFile(pathImg);
}
```

file ảnh trong thư mục Debug



# Ví dụ tạo ứng dụng MDI

- Sự kiện menuExit\_Click và menuNew\_Click

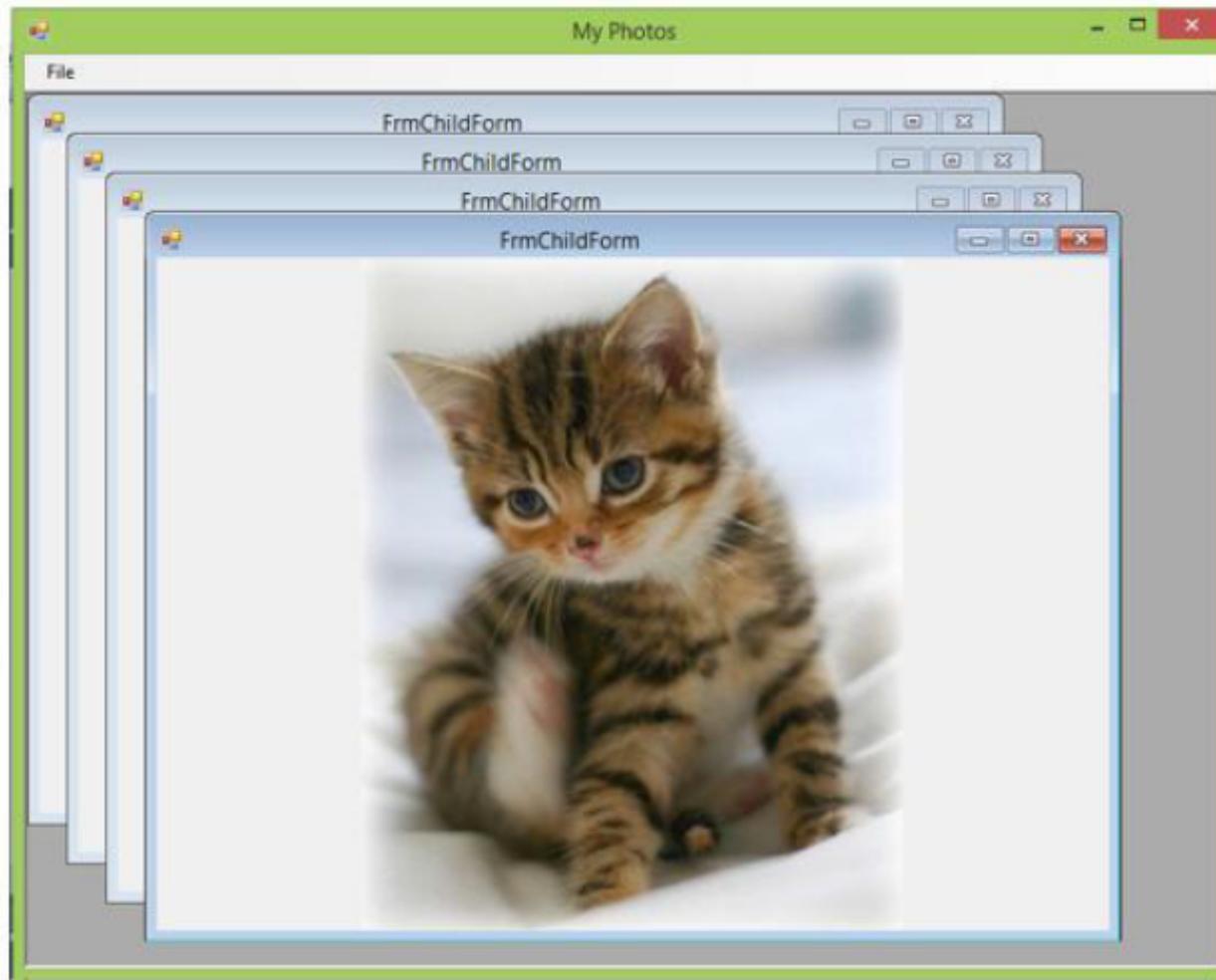
```
private void menuExit_Click(object sender, EventArgs e)
{
 Close();
}
```

```
private void menuNew_Click(object sender, EventArgs e)
{
 FrmChildForm f = new FrmChildForm();
 f.MdiParent = this;
 f.Show();
}
```



# Ví dụ tạo ứng dụng MDI

- Kết quả sau khi biên dịch.





# Lớp MessageBox

- Message Box hiện một thông báo hay một hướng dẫn cho user
- Lớp MessageBox chỉ chứa một phương thức tĩnh duy nhất: Show(...)

```
DialogResult Show(string text, string caption,
 MessageBoxButtons buttons,
 MessageBoxIcon icon,
 MessageBoxDefaultButton defaultButton,
 MessageBoxOptions options);
```

- Namespace
  - System.Windows.Forms
- Assembly
  - System.Windows.Forms (System.Windows.Forms.dll)



# Lớp MessageBox

- Các Enumerations

- MessageBoxButtons
- MessageBoxIcon
- MessageBoxDefaultButton
- MessageBoxOptions
- DialogResult

```
public enum MessageBoxButtons
{
 OK,
 OKCancel,
 AbortRetryIgnore,
 YesNoCancel,
 YesNo,
 RetryCancel
}
```

```
public enum MessageBoxIcon
{
 Asterisk = 0x40,
 Error = 0x10,
 Exclamation = 0x30,
 Hand = 0x10,
 Information = 0x40,
 None = 0,
 Question = 0x20,
 Stop = 0x10,
 Warning = 0x30
}
```



# Lớp MessageBox

```
public enum MessageBoxOptions
{
 DefaultDesktopOnly = 0x20000,
 RightAlign = 0x80000,
 RtlReading = 0x100000,
 ServiceNotification = 0x200000
}
```

```
public enum MessageBoxDefaultButton
{
 Button1 = 0,
 Button2 = 0x100,
 Button3 = 0x200
}
```



# Lớp MessageBox

```
public enum DialogResult
{
 None,
 OK,
 Cancel,
 Abort,
 Retry,
 Ignore,
 Yes,
 No
}
```



# Lớp MessageBox

**MessageBox.Show("Hello Tèo");**

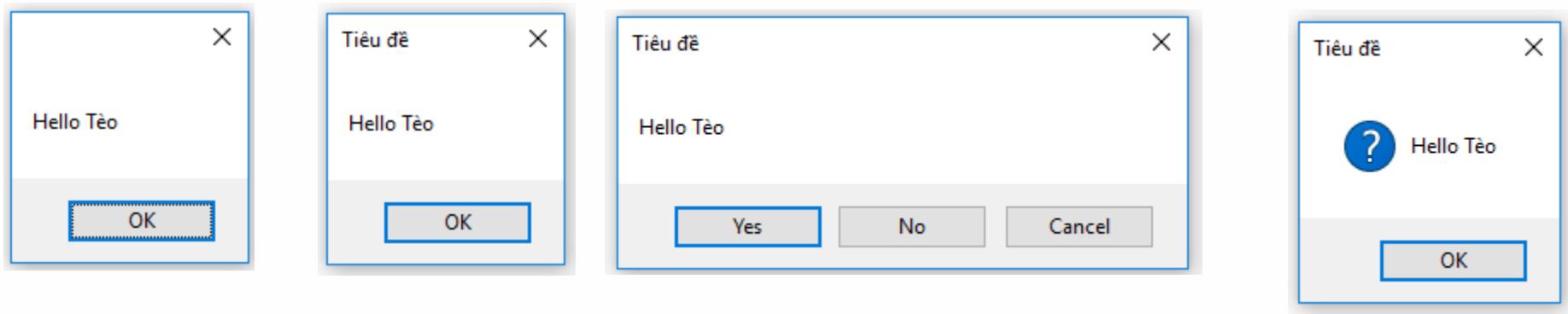
**MessageBox.Show("Hello Tèo", "Title");**

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

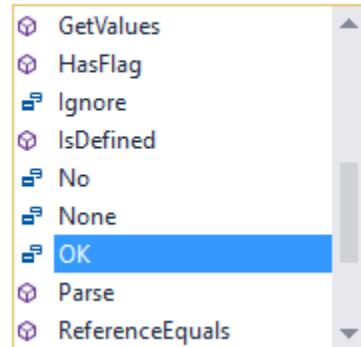
**MessageBox.Show("Hello Tèo", "Title",  
MessageBoxButtons.OK,  
MessageBoxIcon.Question);**



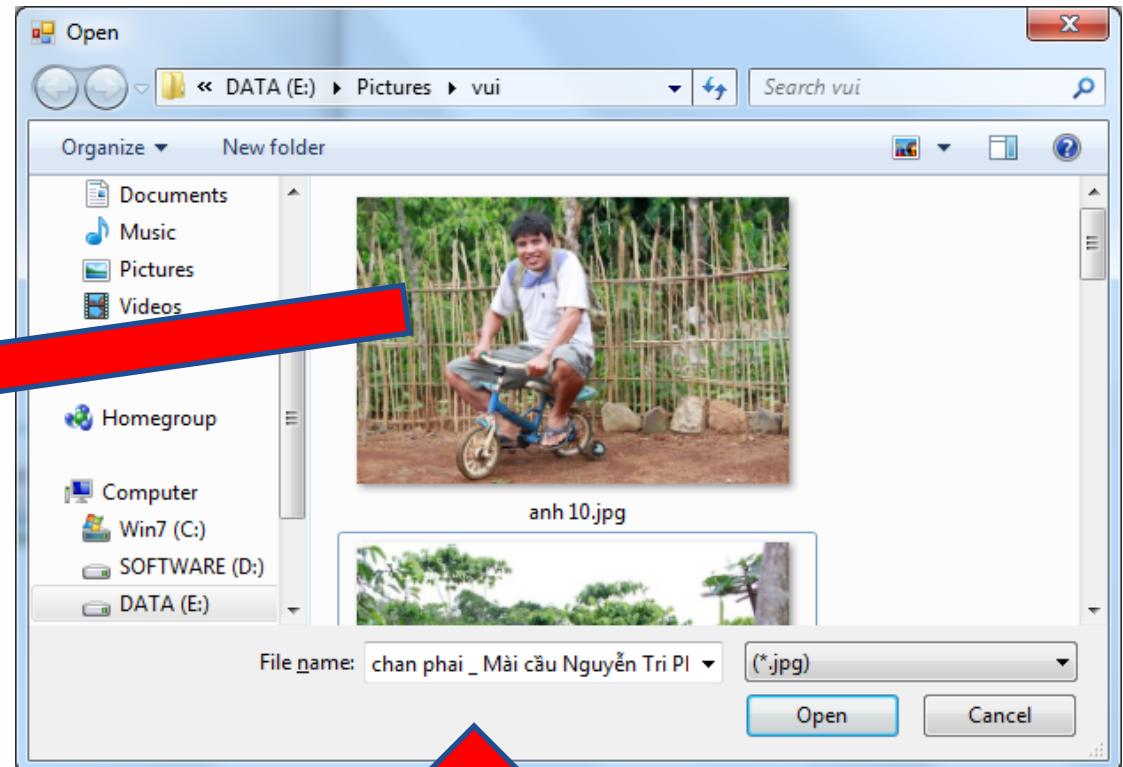
# Lớp MessageBox



```
DialogResult dR= MessageBox.Show("Hello Tèo", "Tiêu đề", MessageBoxButtons.OK, MessageBoxIcon.Question);
if (dR==DialogResult.)
```



# Bài tập



```
private void btnOpenPic_Click
(object sender, EventArgs e)
{
 OpenFileDialog fileOpenDlg = new OpenFileDialog();
 fileOpenDlg.Filter = "(*.jpg)|*.jpg|(*.doc)|*.doc";
 if (fileOpenDlg.ShowDialog() == DialogResult.OK)
 {
 //picDemo.Image = new Bitmap(fileOpenDlg.OpenFile());
 //Or
 picDemo.Image = Image.FromFile(fileOpenDlg.FileName);
 //Or
 //picDemo.BackgroundImage = new Bitmap(fileOpenDlg.OpenFile());
 }
}
```



# Bài tập - TrackBar

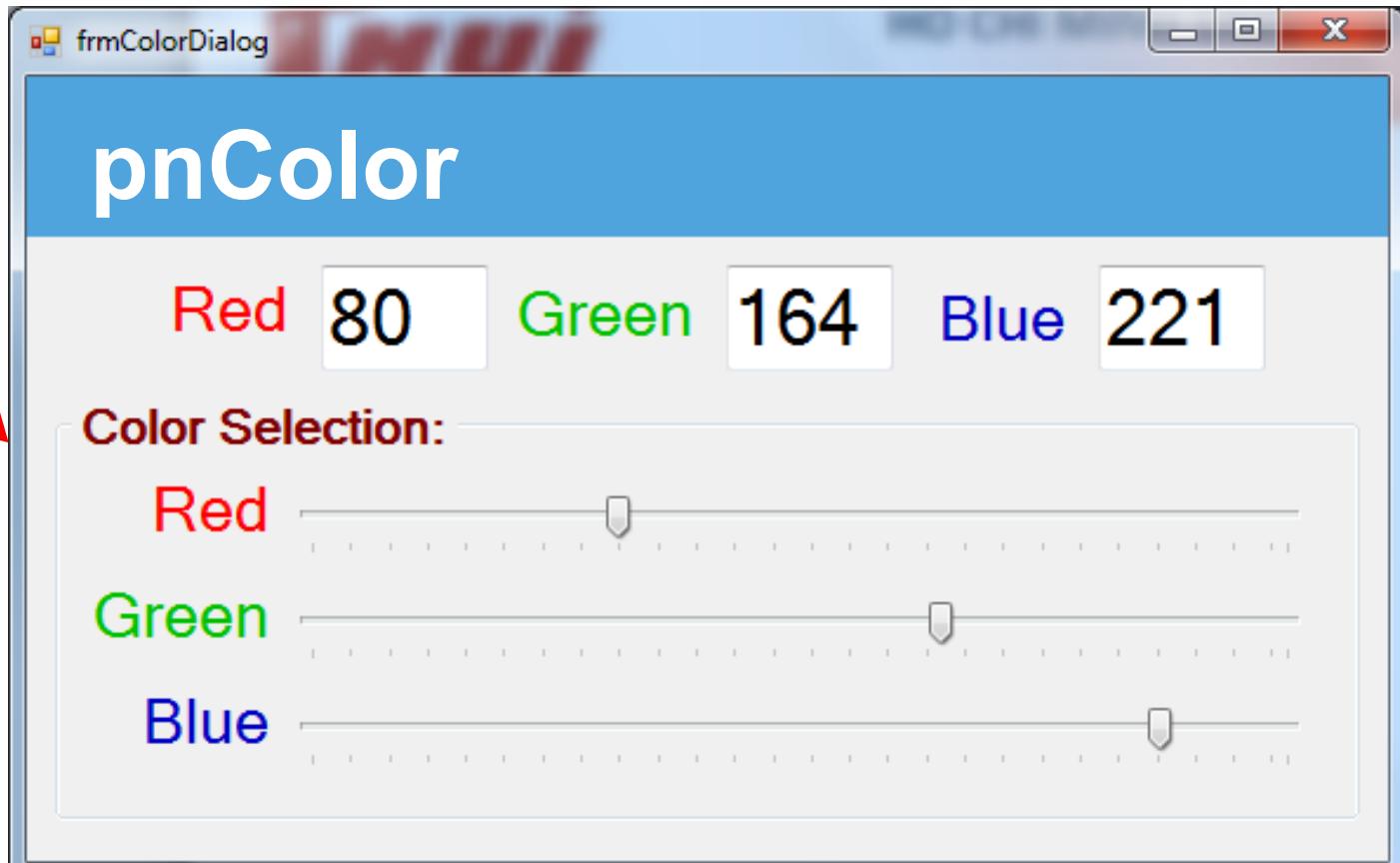
Toolbox

- ToolTip
- TrackBar**
- TreeView
- VScrollBar
- WebBrowser

trackBarRed  
txtRed

trackBarGreen  
txtGreen

trackBarBlue  
txtBlue



The frmColorDialog window displays the title "frmColorDialog" and the form name "pnColor". It shows current color values: Red 80, Green 164, and Blue 221. Below these are three trackbars labeled "Color Selection:" with arrows pointing right, and text labels "Red", "Green", and "Blue" aligned with their respective trackbars.



# TrackBar

The screenshot shows the Windows Forms Properties window for a track bar control named "trackBarRed". The properties listed are:

|                   |             |
|-------------------|-------------|
| Maximum           | 254         |
| MaximumSize       | 0, 0        |
| Minimum           | 0           |
| MinimumSize       | 0, 0        |
| Modifiers         | Private     |
| Orientation       | Horizontal  |
| RightToLeft       | No          |
| RightToLeftLayout | False       |
| Size              | 441, 45     |
| SmallChange       | 1           |
| TabIndex          | 1           |
| TabStop           | True        |
| Tag               |             |
| TickFrequency     | 10          |
| TickStyle         | BottomRight |

**Maximum**  
The maximum value for the position of the slider on the TrackBar.



# TrackBar

```
private void setColor(){
 int nRed = trackBarRed.Value;
 int nGreen = trackBarGreen.Value;
 int nBlue = trackBarBlue.Value;
 txtRed.Text = nRed+"";
 txtBlue.Text = nBlue + "";
 txtGreen.Text = nGreen+"";
 pnColor.BackColor = Color.FromArgb(nRed,
 nGreen,nBlue);}
trackBarBlue.Scroll += processTrackBar;
private void processTrackBar(object sender,
 EventArgs e)
{setColor();}
```