

CAO ĐẲNG KỸ THUẬT  
CAO THẮNG

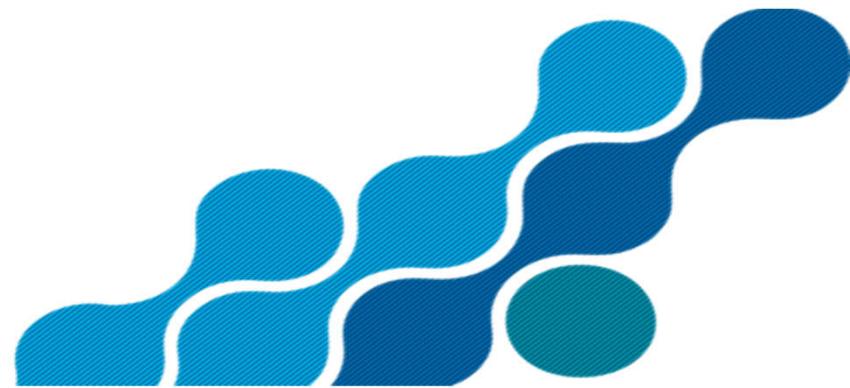


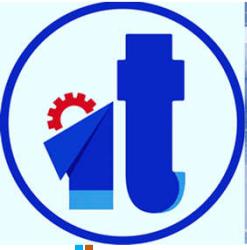
# LẬP TRÌNH TRÊN MÔI TRƯỜNG WINDOWS





## Chương 3: Lập Trình Windows Forms



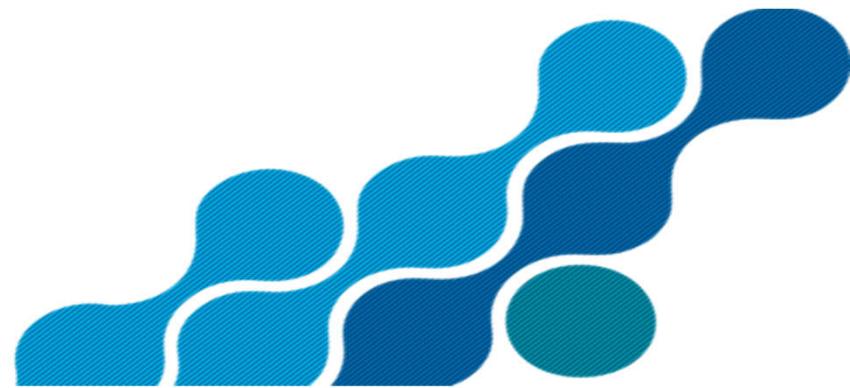


# Nội dung

- Xử lý sự kiện
- Lập trình ứng dụng Windows Forms
- Các controls cơ bản
- Các controls nâng cao



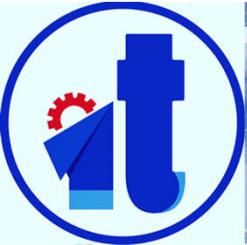
# Xử lý sự kiện



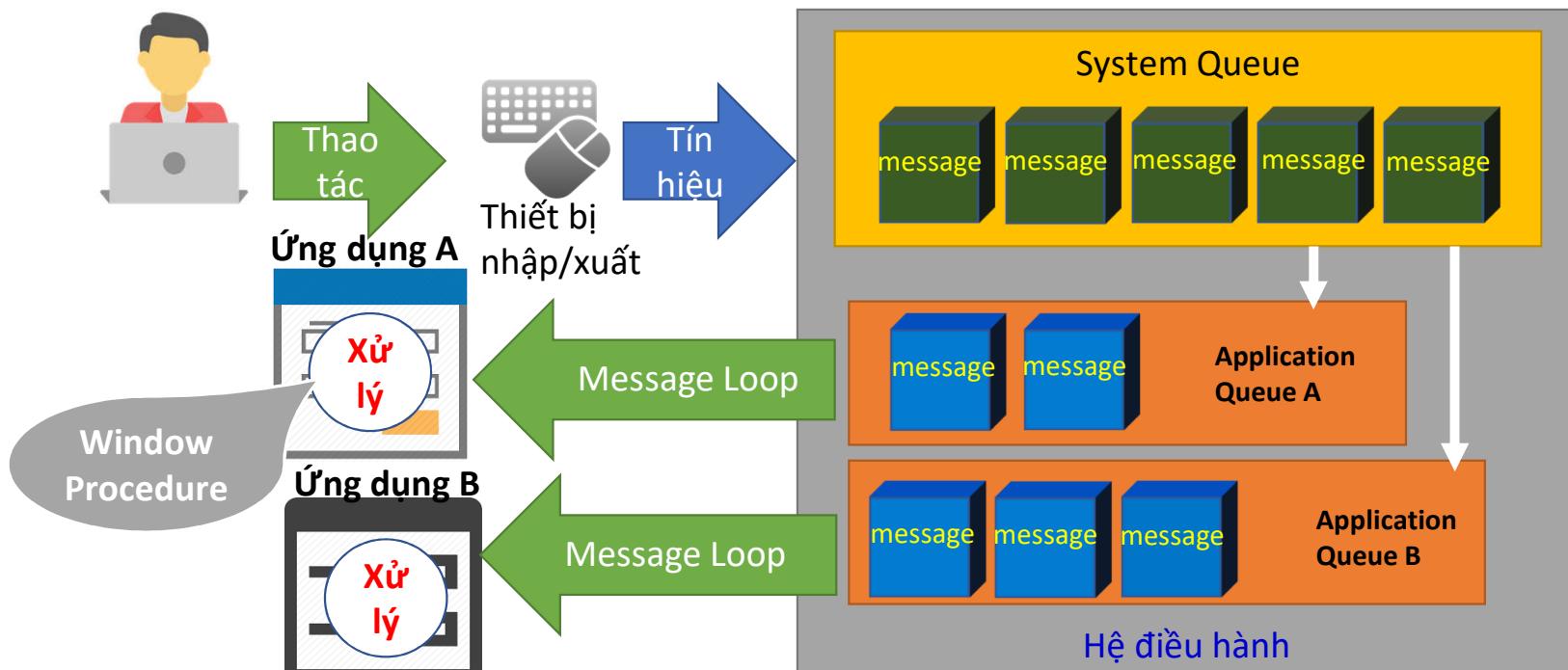


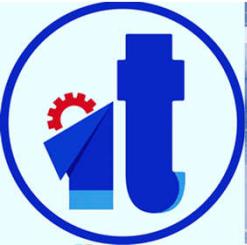
## Quy trình xử lý sự kiện trong windows

- Người sử dụng thao tác trên các ứng dụng/ chương trình thông qua các thiết bị nhập xuất.
- Mỗi thao tác trên các thiết bị nhập xuất → hệ điều hành nhận thao tác → gửi thông điệp để chương trình/ ứng dụng thực hiện một tác vụ nào đó.
- Ứng dụng thực hiện xong tác vụ, trả kết quả về theo hướng ngược lại.



# Quy trình xử lý sự kiện trong windows





# Xử lý sự kiện

## ❑ Message (Thông điệp)

- Là số nguyên được quy ước trước giữa Windows và các ứng dụng
- Các dữ liệu nhập (từ bàn phím, từ chuột, ...) đều được Windows chuyển thành các message và một số thông tin kèm theo message.
- Ví dụ:
  - ✓ 0x0001 WM\_CREATE
  - ✓ 0x0002 WM\_DESTROY
  - ✓ 0x0003 WM\_MOVE
  - ✓ 0x0005 WM\_SIZE
  - ✓ 0x0012 WM\_QUIT



# Xử lý sự kiện

## **System Queue (Hàng đợi hệ thống):**

- Hàng đợi để Windows chứa các message.

## **Application Queue (Hàng đợi ứng dụng):**

- Hàng đợi riêng của các ứng dụng để chứa các message của ứng dụng.
- Windows sẽ tự động phân bố các message từ System Queue đến các Application Queue.

## **Message loop (vòng lặp thông điệp)**

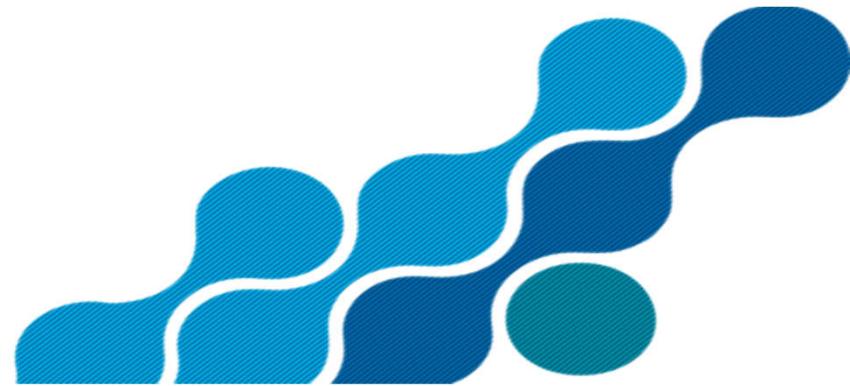
- Mỗi ứng dụng tại một thời điểm có một message loop để lấy các message trong Application Queue về để phân bố cho các cửa sổ (Window) trong Application.

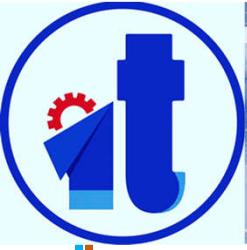
## **Hàm Window Procedure**

- Mỗi cửa sổ (Window) trong Application đều có một hàm Window procedure để xử lý các message do message loop nhận về.



# Lập trình ứng dụng Windows Forms





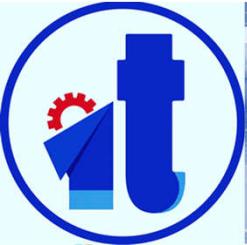
# Lập trình ứng dụng Windows Forms

□ Quy trình:

Thiết kế giao diện

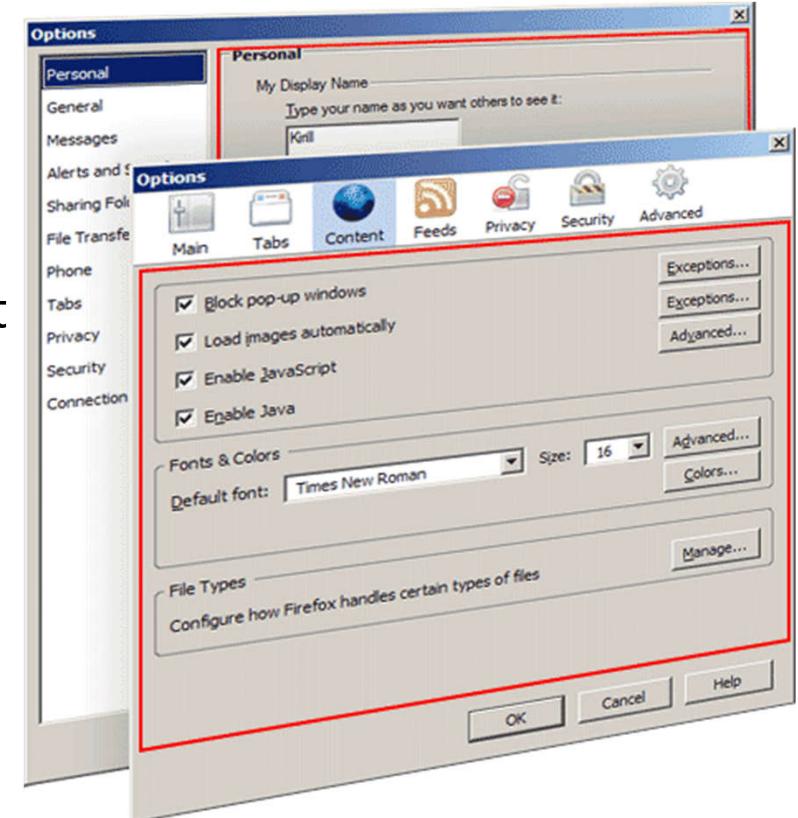
Xử lý các message do Windows gửi đến

Xử lý nghiệp vụ



# Thành phần ứng dụng

- Thành phần của một ứng dụng Windows Forms
  - Application (ứng dụng)
  - Các màn hình/ Forms trong Application
  - Các controls hoặc components trong một Form





# Cấu trúc tổ chức lập trình ứng dụng

- Cấu trúc tổ chức để lập trình ứng dụng Windows Forms

THÀNH PHẦN	CHỨC NĂNG
Properties	Chứa resource và cấu hình
References	Thư viện sử dụng chương trình
App.Config	Cấu hình các tham số Lưu giữ các namespace sử dụng trong chương trình có dạng XML
Program.cs	Chương trình chính để chạy
Forms	Các Form thiết kế trong chương trình



## Lớp Application

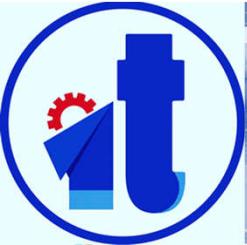
- Cung cấp các phương thức tĩnh và các property tĩnh để quản lý ứng dụng
  - Các phương thức start, stop ứng dụng, xử lý Windows messages,
  - Các property lấy thông tin về ứng dụng
  - Lớp này không thể thừa kế
  - Namespace: System.Windows.Form
  - Assembly: System.Windows.Form (System.Windows.Form.dll)



# Lớp Application

## ❑ Các phương thức, thuộc tính thông dụng

THÀNH VIÊN	LOẠI	MÔ TẢ
<b>Run(Form)</b>	Phương thức	Bắt đầu message loop của ứng dụng
<b>Exit()</b>	Phương thức	dừng message loop
<b>DoEvents()</b>	Phương thức	Xử lý các message trong khi chương trình đang trong vòng lặp
<b>EnableVisualStyles()</b>	Phương thức	Các control sẽ vẽ với kiểu visual nếu control và hệ điều hành hỗ trợ
<b>Restart()</b>	Phương thức	Dừng ứng dụng và tự động restart lại
<b>ExecutablePath</b>	Thuộc tính	Đường dẫn đến file .exe
<b>StartupPath</b>	Thuộc tính	Đường dẫn đến thư mục chứa file .exe
<b>UseWaitCursor</b>	Thuộc tính	Hiện cursor dạng Wait
<b>Idle</b>	Sự kiện	Xuất hiện khi ứng dụng hoàn thành việc xử lý



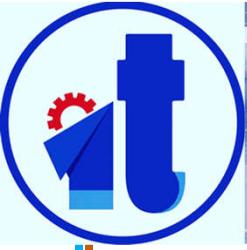
# Components

- Là một thành phần phần mềm
- Lớp **Component** là lớp cơ sở của tất cả các component
- Namespace: System.ComponentModel
- Assembly: System (System.dll)
- Các component trong .NET gồm có các thành viên:
  - Properties (các thuộc tính)
  - Methods (các phương thức)
  - Events (các sự kiện)
  - [Các thành viên protected]
- Các Component không hỗ trợ tương tác với người dùng bằng form giao diện tự nhiên



## Controls

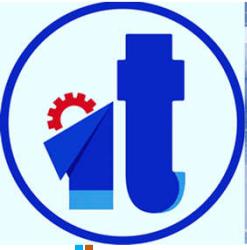
- Là Component có giao diện (hỗ trợ hiển thị khi thiết kế)
- Giao diện ứng dụng gồm một tập các control, giúp người dùng tương tác với ứng dụng
- Cho phép hiển thị dữ liệu (output) hay cho phép nhập dữ liệu vào (input)
- Namespace: System.Windows.Forms
- Assembly: System.Windows.Forms (System.Windows.Forms.dll)



# Controls

## ☐ Một số Controls thông thường

Control	Mô tả	Tiền tố
Form	Giao diện màn hình	frm
Button	Nút	btn
TextBox	Khung nhập liệu	txt
Label	Nhãn tên control	lbl
RadioButton	Tùy chọn một lần	rad
GroupBox	Nhóm đối tượng	grp
CheckBox	Tùy chọn nhiều lần	chk
ListBox	Danh sách đối tượng	Lst



# Controls

## □ Một số Controls thông thường

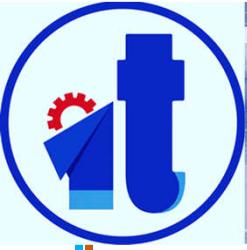
Control	Mô tả	Tiền tố
Combobox	Menu lựa chọn	cbo
PictureBox	Hiển thị hình ảnh	pic
ListView	Danh sách đối tượng	lvw
DataGridView	Danh sách dữ liệu	dgv



# Controls

## ☐ Các thuộc tính chung các controls

THUỘC TÍNH	CHỨC NĂNG
(name)	Tên đại diện cho control, dùng để truy xuất và sử dụng như biến
Achor	Cố định control khi thay đổi kích thước form
BackColor	Màu nền của control
BackgroundImage	Hình nền của control
ContextMenuStrip	Menu khi ấn chuột phải control
Location	Vị trí control đó trên form



# Controls

## □ Các thuộc tính chung các controls

THUỘC TÍNH	CHỨC NĂNG
Dock	Gần giống Anchor nhưng chiếm toàn bộ vùng được thiết lập
Enabled	Cho phép sử dụng control
Font	Nhóm thuộc tính màu, cỡ, kiểu chữ nội dung control
Text	Nội dung miêu tả control
TextAlign	Căn lề nội dung miêu tả control
Visible	Hiển thị control lên form

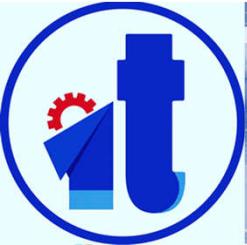
Một đối tượng (**control**) có thể thay đổi thuộc tính trong khung **Properties** hoặc bằng **code**



# Controls

## ❑ Các sự kiện và Phương thức phổ biến

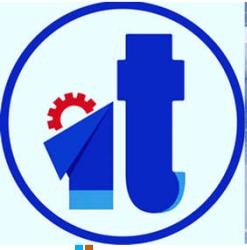
SỰ KIỆN	CHỨC NĂNG
Active	Khởi động Form & focus vào nó
Close	Đóng Form lại
Dispose	Giải phóng mọi tài nguyên đã sử dụng bởi component
Focus	Tập trung vào đối tượng
Hide	Ẩn control khỏi người dùng
OnActive	Khởi động sự kiện Activated (thực hiện hành động sau khi Form được Active)



# Controls

## ❑ Các sự kiện và phương thức phổ biến

SỰ KIỆN	CHỨC NĂNG
OnClick	Khởi động sự kiện Click (thực hiện sau khi control được ấn)
Refresh	Làm mới lại Form/Control, vô hiệu hóa client và lập tức vẽ đối tượng
Show()	Hiển thị control ra màn hình
ShowDialog()	Hiển thị form như một Dialog
ToString()	Hiển thị chuỗi tương ứng



# Controls

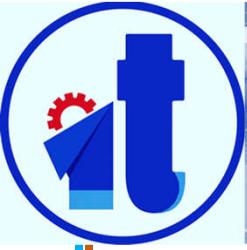
## ❑ Các sự kiện và phương thức phổ biến

SỰ KIỆN	CHỨC NĂNG
Click	Ấn vào control
DoubleClick	Ấn kép vào control
KeyDown	Bắt đầu ấn phím
KeyUp	Đã ấn phím xong
MouseDown	Ấn chuột
MouseUp	Thả chuột
MouseHover	Rê chuột qua control
MouseLeave	Rê chuột khỏi control



# Forms

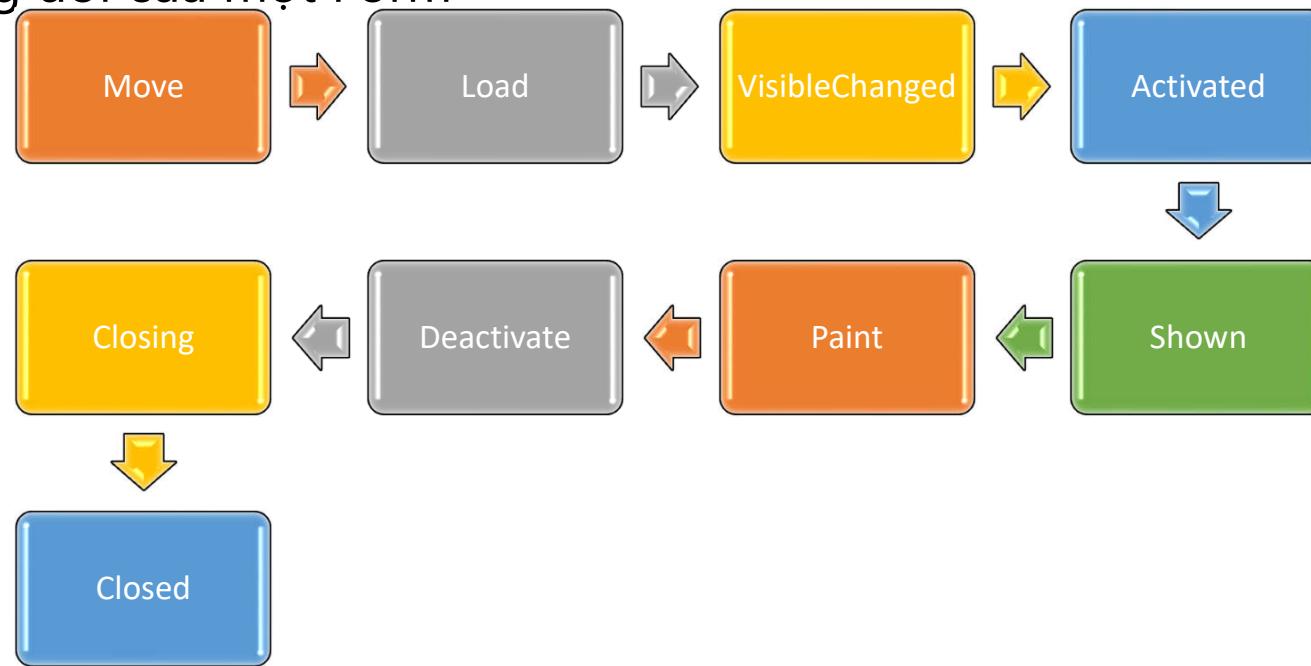
- ❑ **Lớp Form:** thể hiện một cửa sổ (window) hay một dialog box tạo nên giao diện của ứng dụng
  - Thông thường tạo custom form bằng cách thừa kế từ lớp Form
  - Chứa các components hoặc các control
  - Namespace: System.Windows.Form
  - Assembly: System.Windows.Form (System.Windows.Form.dll)
  - Mỗi form sẽ có ba thành phần
    - ✓ File \*.Designer.cs dùng để thiết kế form
    - ✓ File \*.cs dùng để mô tả xử lý của control đó trên form
    - ✓ File \*.resx chứa resource dành cho form
  - Mỗi đối tượng hoặc control hiển thị đều được thiết lập trong file \*.Designer.cs
  - **LƯU Ý:** Khi xóa một control, thiết kế và xử lý của control sẽ đồng thời bị xóa. Xóa bằng không thống nhất → Phát sinh lỗi



# Forms

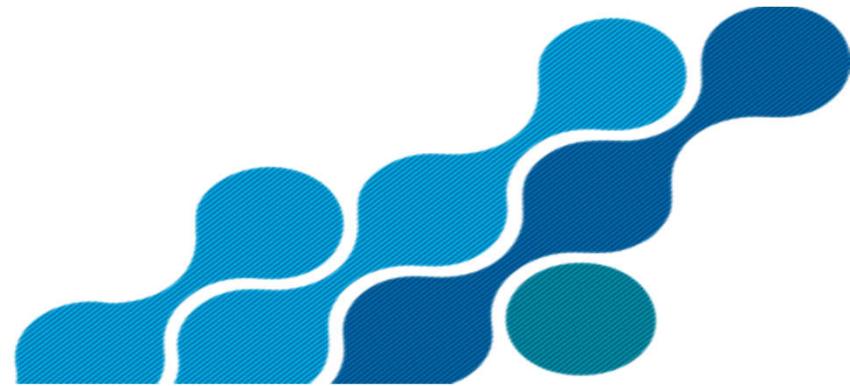
## ☐ Lớp Form:

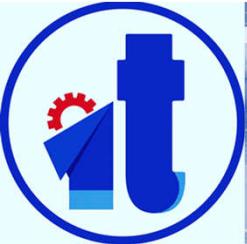
➤ Vòng đời của một Form





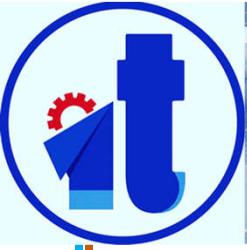
# Các controls cơ bản





## Các controls cơ bản

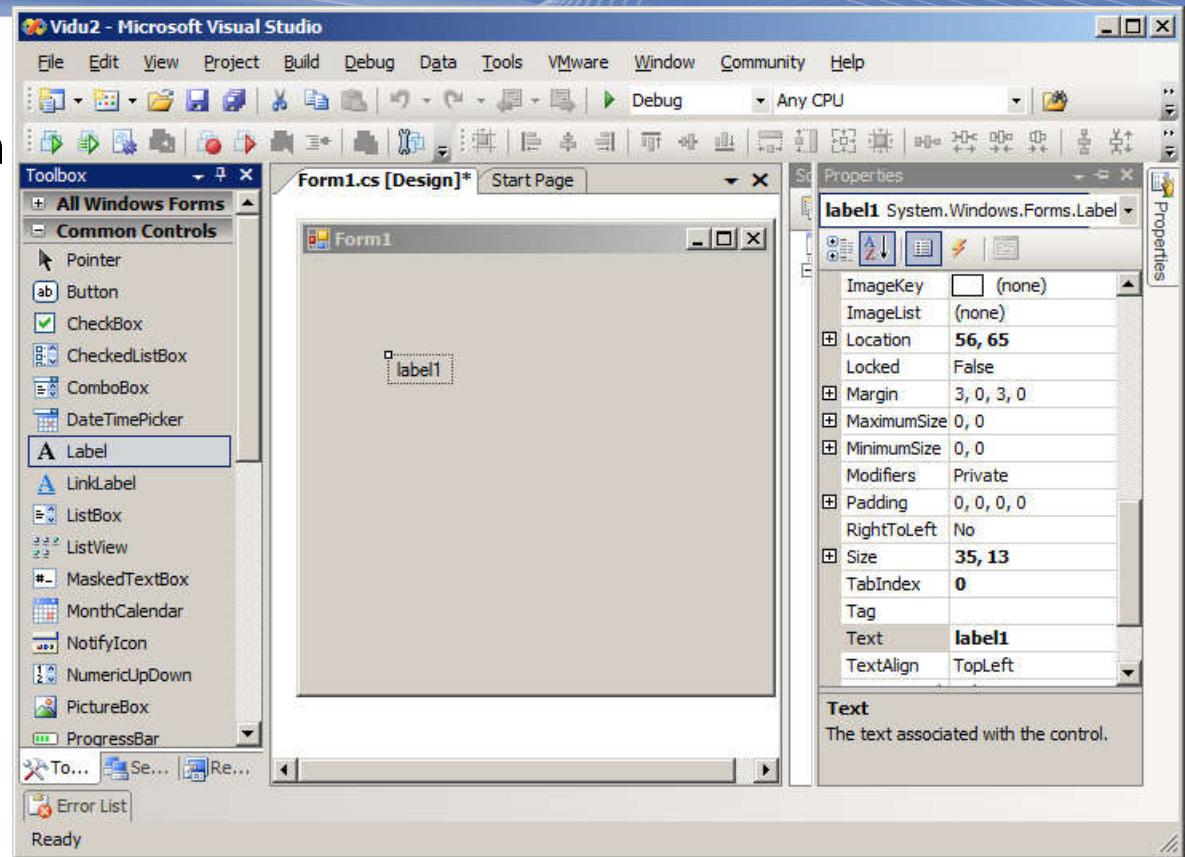
- Label, textbox, button
- CheckBox, RadioButton
- ListBox, CheckedListBox, Combobox
- NumericUpDown, DomainUpDown, TrackBar
- DateTimePicker, MonthCalendar
- ListView, TreeView
- PictureBox, ImageList
- GroupBox, Panel, TabControl

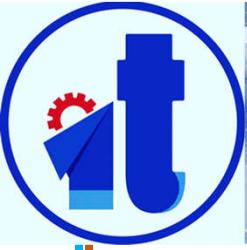


# Label, textbox, button

## □ Label

- Cung cấp chuỗi thông tin chỉ dẫn
- Chỉ đọc
- Được định nghĩa bởi lớp Label
- Dẫn xuất từ Control

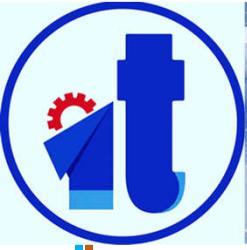




# Label, textbox, button

## Label

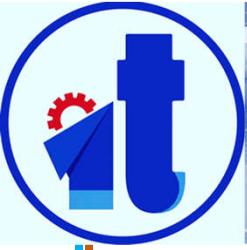
Thuộc tính thường dùng	Điễn giải
Font	Font hiển thị của text
Text	Nội dung text hiển thị
TextAlign	Canh lề text
ForeColor	Màu text
Visible	Trạng thái hiển thị



## Label, textbox, button

### **Textbox**

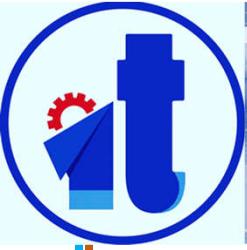
- Vùng cho phép user nhập hoặc hiển thị dữ liệu
- Cho phép nhập dạng Password
- Cho phép nhập nhiều dòng



# Label, textbox, button

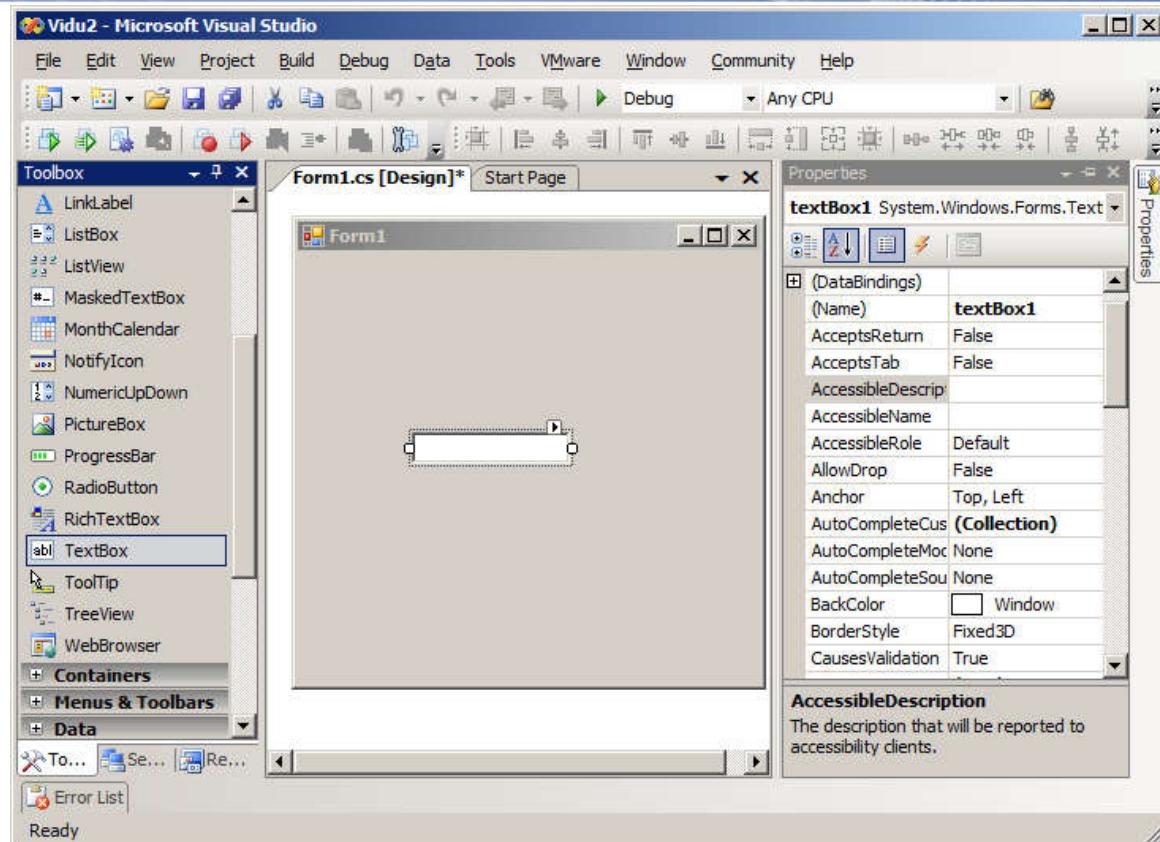
## □ **Textbox**

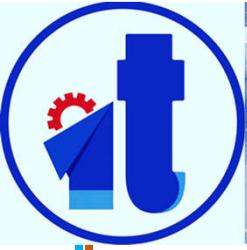
<i>Thuộc tính thường dùng</i>	<i>Điễn giải</i>
<i>AcceptsReturn</i>	Nếu true: nhấn enter tạo thành dòng mới trong chế độ multiline
<i>Multiline</i>	Nếu true: textbox ở chế độ nhiều dòng, mặc định là false
<i>PasswordChar</i>	Chỉ hiển thị ký tự đại diện cho text
<i>ReadOnly</i>	Nếu true: textbox hiển thị nền xám, và ko cho phép nhập liệu, mặc định là false
<i>ScrollBars</i>	Thanh cuộn cho chế độ multiline
<i>Event thường dùng</i>	
<i>TextChanged</i>	Kích hoạt khi text bị thay đổi, trình xử lý được khởi tạo mặc định khi kích đúp vào textbox trong màn hình design view



# Label, textbox, button

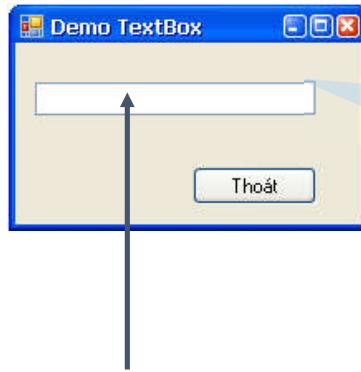
## □ Textbox



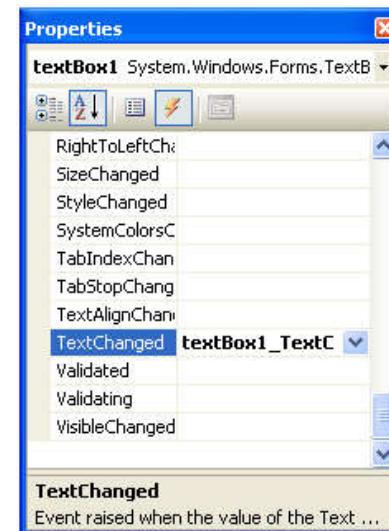


# Label, textbox, button

## ☐ Textbox

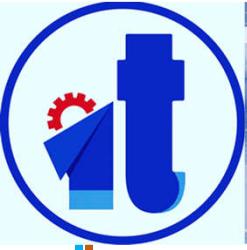


Double click vào  
textbox để tạo event  
handler cho event  
*TextChanged*



```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    string text;
    text = ((TextBox)sender).Text;
    ((TextBox)sender).Text = text.ToUpper();
}
```

Chuyển thành chữ hoa

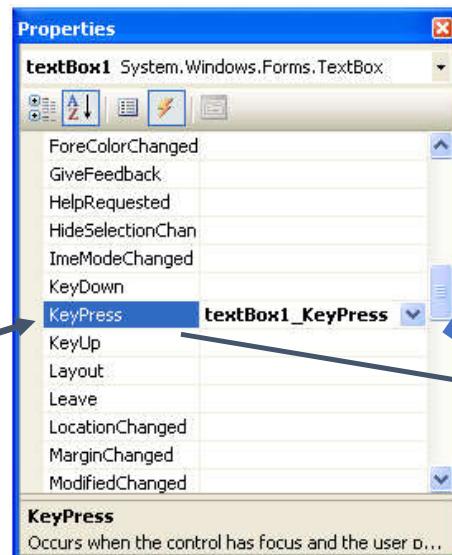


# Label, textbox, button

## ☐ Textbox

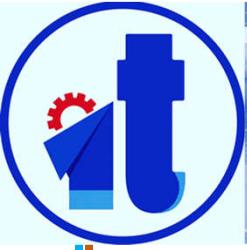


Sự kiện KeyPress



```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!Char.IsDigit(e.KeyChar)) // không là ký tự số
        e.Handled = true; // đã xử lý sự kiện keypress
}
```

Sự kiện phát sinh khi  
textbox nhận focus và  
user nhấn 1 phím

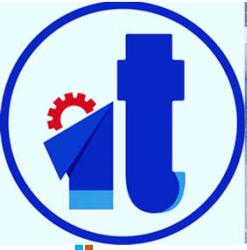


# Label, textbox, button

## **Button**

- Cho phép cài đặt và thực thi một hành động khi người dùng nhấn chuột.
- Dẫn xuất từ ButtonBase

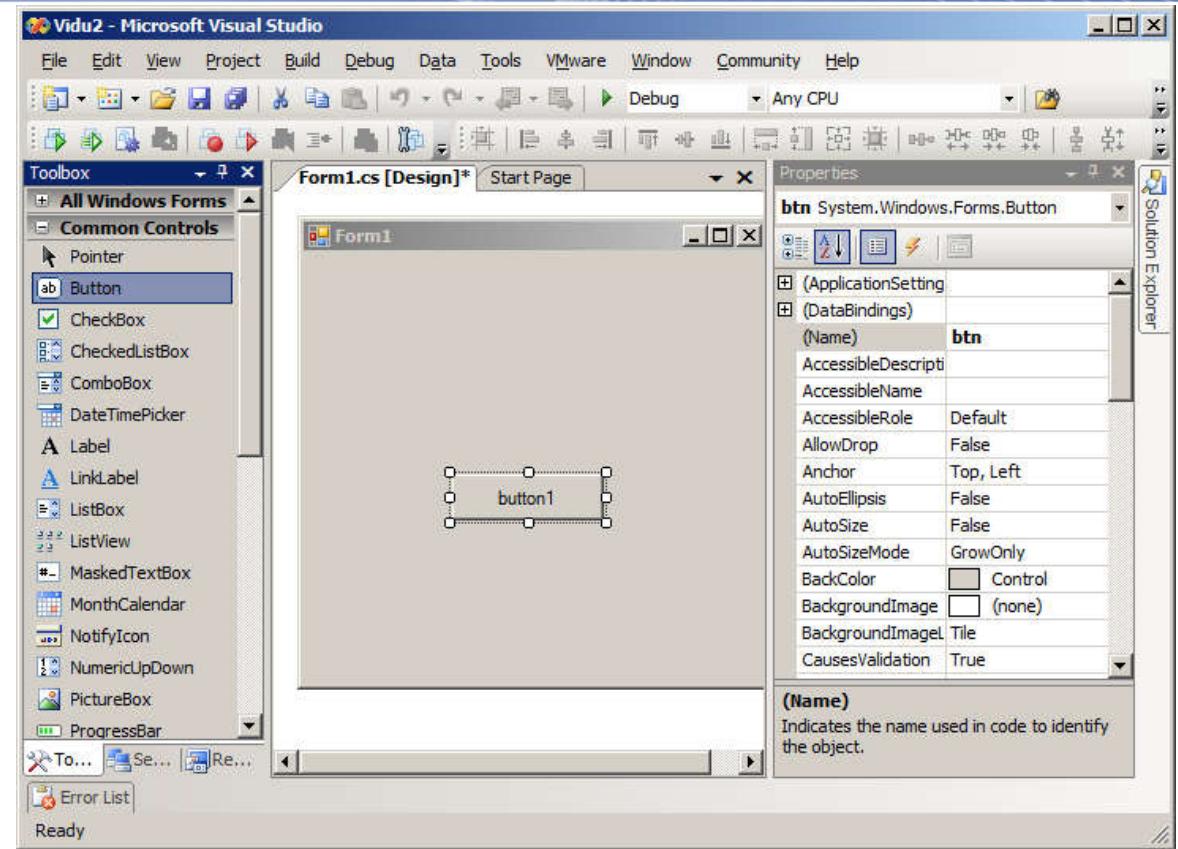
<i>Thuộc tính thường dùng</i>	
Text	Chuỗi hiển thị trên bề mặt button
<i>Event thường dùng</i>	
Click	Kích hoạt khi user kích vào button, khai báo mặc định khi người lập trình kích đúp vào button trong màn hình Design View của Form.



# Label, textbox, button

## □ Button

- Khi click chuột trái vào nút Button thì sẽ thực thi một mệnh lệnh nào đó.
- Để bắt sự kiện khi click chuột trái thì trong Properties chọn thẻ Events và nhập tên hàm trong mục Click





# Label, textbox, button



## Button

Properties

btn System.Windows.Forms.Button	X
A Z	
DataBindings	
AutoSizeChanged	
BackColorChanged	
BackgroundImageChanged	
BackgroundImageLayoutChanged	
BindingContextChanged	
CausesValidationChanged	
Click	btn_Click
ClientSizeChanged	
ContextMenuStripChanged	
ControlAdded	
ControlRemoved	
CursorChanged	
ChangeUICues	
DockChanged	
Click	

Click  
Occurs when the component is clicked.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Vidu2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void btn_Click(object sender, EventArgs e)
        {
            //viết các lệnh xử lý khi thực hiện click chuột trái vào Button
        }
    }
}
```



# Label, textbox, button

- Bài tập: Thiết kế các giao diện sau:

Màn hình đăng nhập

Thông tin tài khoản

Tên tài khoản

Mật khẩu

[Đăng nhập](#)

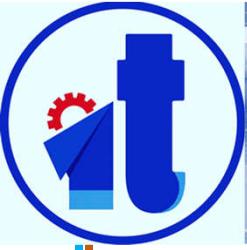
Facebook

Email Address

Password

[Login](#)

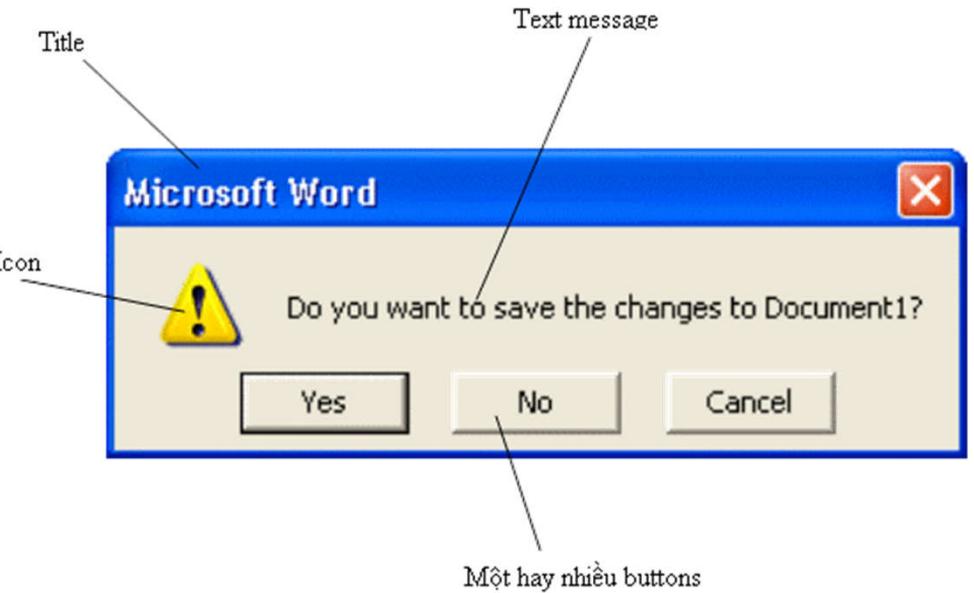
[Forgot password?](#)



# Message Box

- Được dùng để nhắc nhở, cảnh báo, cung cấp thông tin,... và đòi hỏi một phản ứng nào đó từ người dùng.
- 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);
```





# Message Box

## □ Ví dụ

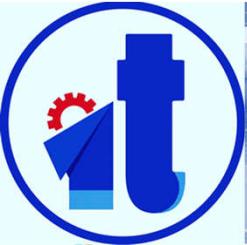
```
// Configure the message box to be displayed
string MessageBoxText = "Do you want to save changes?";
string caption = "Word Processor";
MessageBoxButton button = MessageBoxButton.YesNoCancel;
MessageBoxImage icon = MessageBoxIcon.Warning;

// Display message box
MessageBox.Show(messageBoxText, caption, button, icon);
```

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

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

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

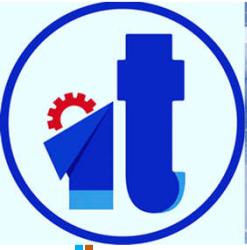


# Message Box

```
// Display message box
MessageBoxResult result = MessageBox.Show(messageBoxText, caption, button, icon);

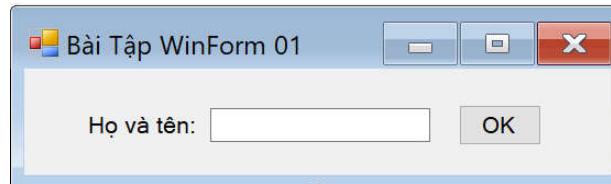
// Process message box results
switch (result)
{
    case MessageBoxResult.Yes:
        // User pressed Yes button
        // ...
        break;
    case MessageBoxResult.No:
        // User pressed No button
        // ...
        break;
    case MessageBoxResult.Cancel:
        // User pressed Cancel button
        // ...
        break;
}
```

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

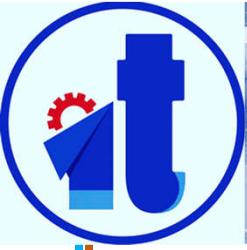


# Bài tập

## 1. Thiết kế giao diện chương trình như hình bên dưới

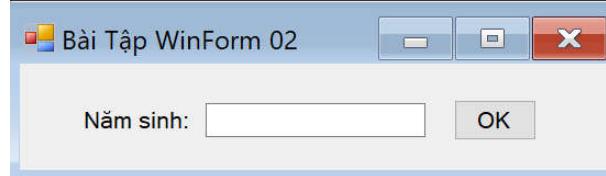


- Nhập **họ và tên** vào Textbox, sau đó nhấn vào Button **OK** thì hiển thị thông báo "**Xin chào ...**" lên MessageBox.
- Lưu ý kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.



# Bài tập

## 2. Thiết kế giao diện chương trình như hình bên dưới



- Nhập **năm sinh** vào Textbox, sau đó nhấn vào Button **OK** thì hiển thị thông báo "**Số tuổi của bạn là: ...**" lên MessageBox.
- Lưu ý:
  - ✓ Chỉ cho nhập số vào textbox
  - ✓ Kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.

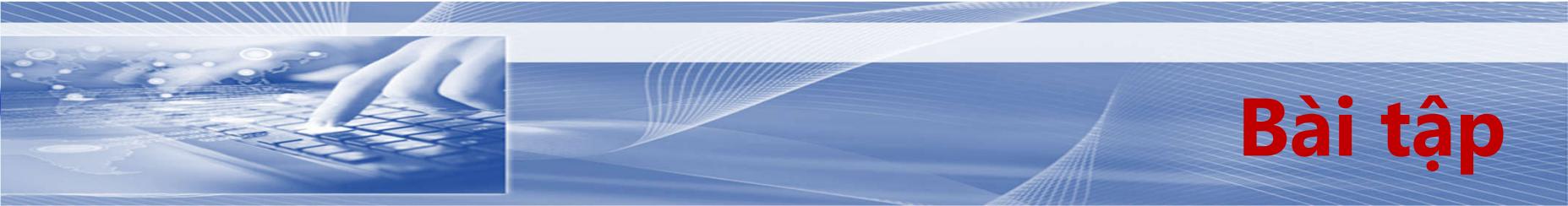


# Bài tập

3. Thiết kế giao diện chương trình như hình bên dưới

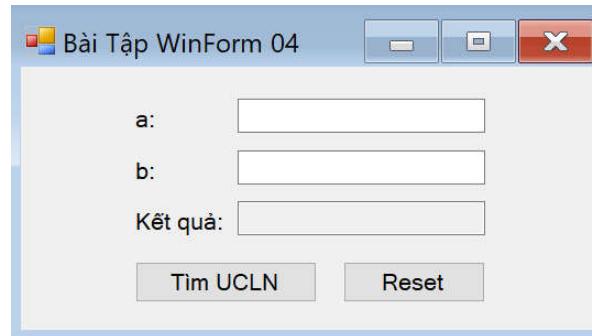


- Nhập **2 số nguyên a, b** vào 2 Textbox, sau đó thực hiện các phép tính khi nhấn vào các Button tương ứng **Cộng** ( $a + b$ ), **Trừ** ( $a - b$ ), **Nhân** ( $a * b$ ), **Chia** ( $a / b$ ) và hiển thị kết quả lên Textbox **Kết quả**.
- Nhấn vào nút **Reset** thì xoá tất cả nội dung trong các Textbox.
- Lưu ý**
  - a và b chỉ cho nhập dạng số (số thực)
  - kiểm tra và thông báo lỗi trên MessageBox nếu chưa nhập dữ liệu.

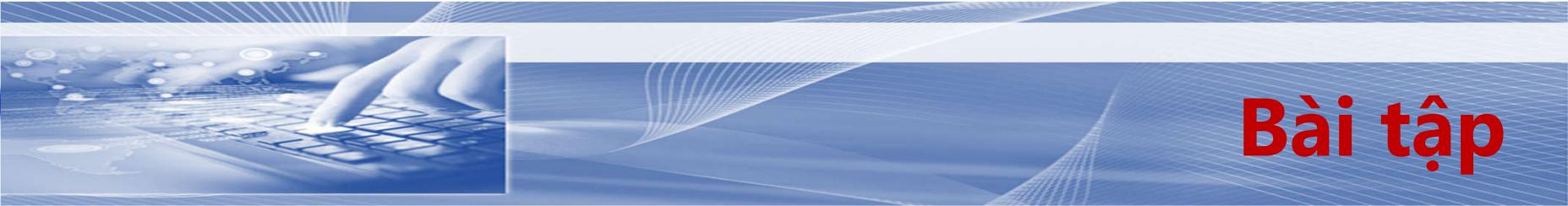


# Bài tập

## 4. Thiết kế giao diện chương trình như hình bên dưới

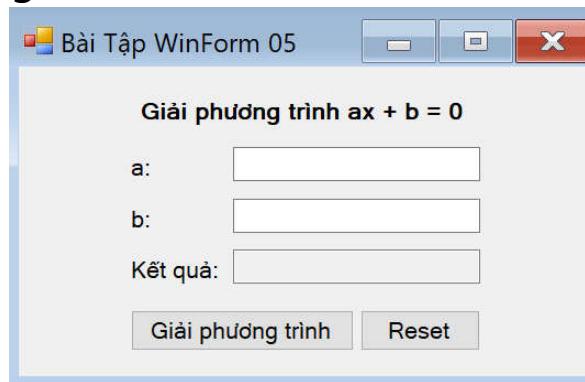


- Nhập **2 số nguyên a, b** vào 2 Textbox, sau đó nhấn vào Button **Tìm UCLN** thì sẽ hiển thị ước chung lớn nhất của 2 số nguyên a, b vào Textbox **kết quả** .
- Nhấn vào nút **Reset** thì xoá tất cả nội dung trong các Textbox.
- Lưu ý**
  - a và b chỉ cho nhập số nguyên
  - kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.

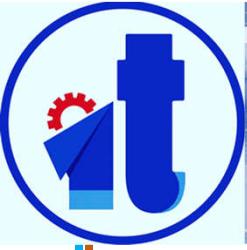


# Bài tập

## 5. Thiết kế giao diện chương trình như hình bên dưới



- Nhập **2 số nguyên a, b** vào 2 Textbox, sau đó nhấn vào Button **Giải phương trình** thì sẽ hiển thị kết quả của việc giải phương trình  $ax + b = 0$  vào Textbox **kết quả**.
- Nhấn vào nút **Reset** thì xoá tất cả nội dung trong các Textbox.
- Lưu ý**
  - a và b chỉ cho nhập dạng số (số thực)
  - kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.



# Bài tập

6. Thiết kế giao diện chương trình như hình bên

Bài Tập WinForm 06

Giải phương trình  $ax^2 + bx + c = 0$

a:

b:

c:

Kết quả:

- ❑ Nhập **3 số nguyên a, b** vào 3 Textbox, sau đó nhấn vào Button **Giải phương trình** thì sẽ hiển thị kết quả của việc giải phương trình  $ax^2 + bx + c = 0$  vào Textbox **kết quả**.
- ❑ Nhấn vào nút **Reset** thì xoá tất cả nội dung trong các Textbox.
- ❑ **Lưu ý**
  - a và b chỉ cho nhập dạng số (số thực)
  - kiểm tra và thông báo lỗi trên MessageBox nếu chưa nhập dữ liệu.



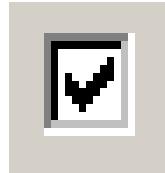
# CheckBox, RadioButton

## CheckBox

- Control mặc định đưa ra một giá trị cho trước và người lập trình có thể:
  - ✓ Chọn giá trị khi **Checked = true**
  - ✓ Không chọn giá trị: **Checked = false**
- Thuộc tính **Text**: hiển thị nội dung bên cạnh checkbox
- Sự kiện thông dụng **CheckedChanged**: phát sinh khi thay đổi trạng thái check

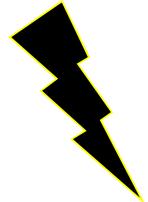
Appearance

Checked



Text

ThreeState



CheckedChanged



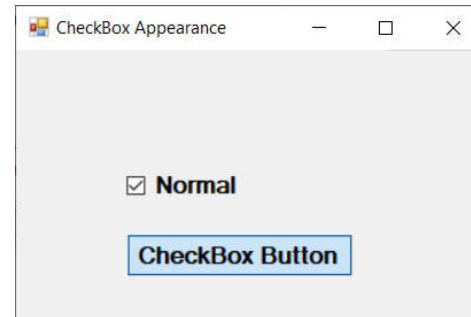
# CheckBox, RadioButton

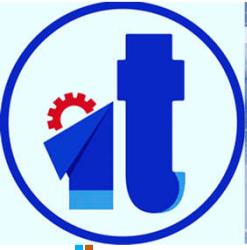
## CheckBox

### ➤ Các dạng khác của CheckBox

- ✓ ThreeState = true: CheckBox có 3 trạng thái
  - Checkstate = Indeterminate: không xác định
  - CheckState= Checked: chọn
  - CheckState= Unchecked: không chọn
  
- ✓ Appearance = Button: CheckBox là một button

*Chưa chọn* →



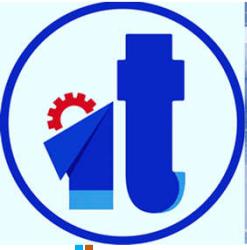


# CheckBox, RadioButton

## RadioButton

- RadioButton 1
- RadioButton 2
- RadioButton 3

- Cho phép user chọn một option trong số nhóm option
- Khi user chọn 1 option thì tự động option được chọn trước sẽ uncheck
- Các radio button chứa trong 1 container (**form**, **GroupBox**, **Panel**, **TabControl**) thuộc một nhóm.
- Lớp đại diện: RadioButton
- Khác với **CheckBox** cho phép chọn nhiều option, còn **RadioButton** chỉ cho chọn một trong số các option.



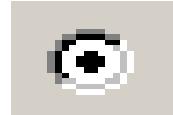
LVHN

# CheckBox, RadioButton

## RadioButton

- RadioButton 1
- RadioButton 2
- RadioButton 3

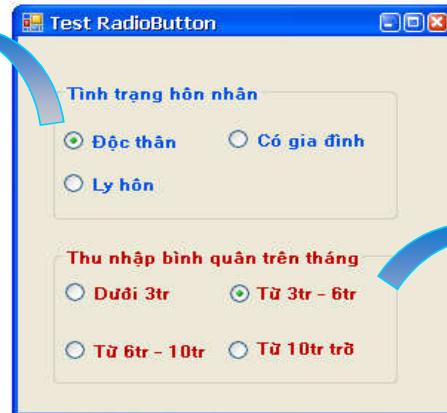
Appearance  
Checked  
**Text**



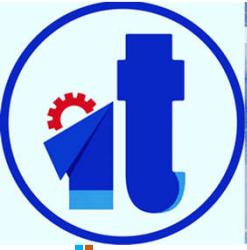
CheckedChanged



Nhóm RadioButton  
thứ 1 chứa trong  
GroupBox1



Nhóm RadioButton  
thứ 2 chứa trong  
GroupBox2



# Bài tập

- Thiết kế giao diện chương trình như hình bên
  - Nhập **Họ tên** vào Textbox và chọn **Sở thích**, sau đó nhấn vào Button **Hiển thị** thì sẽ hiển thị kết quả vào MessageBox.
  - Nhấn vào nút **Reset** thì xoá tất cả nội dung trong Textbox và bỏ check các Checkbox.
  - Lưu ý kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.

Bài Tập WinForm 07

Họ tên: Trần Thanh Tuấn

Sở thích:

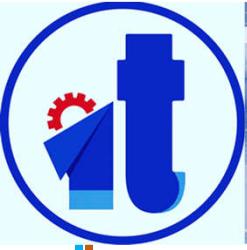
Xem phim  
 Nghe nhạc  
 Đọc sách  
 Đá bóng

Reset      Hiển Thị

Thông Tin

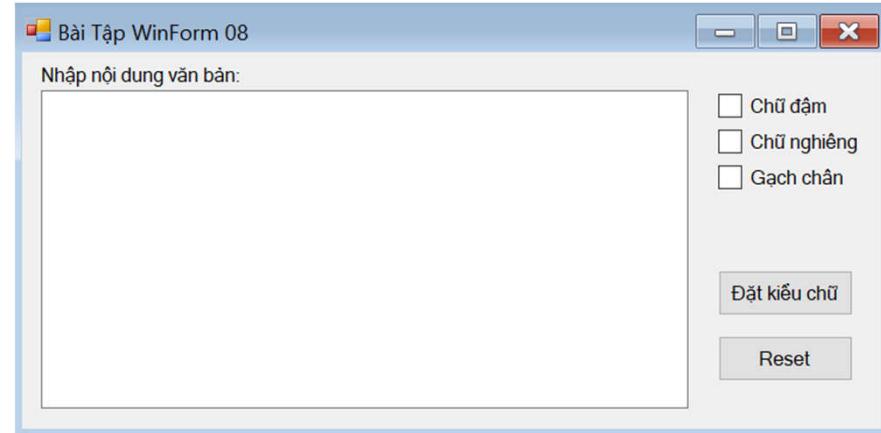
i Trần Thanh Tuấn  
Sở thích:  
- Xem phim  
- Nghe nhạc  
- Đá bóng

OK



# Bài tập

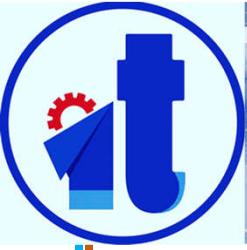
- ❑ Thiết kế giao diện chương trình như hình bên
  - Nhập nội dung vào Textbox, lựa chọn kiểu chữ, sau đó nhấn nút **Đặt kiểu chữ** thì hiển thị nội dung trong Textbox với kiểu chữ đã chọn.
  - Nhấn vào nút **Reset** thì xoá tất cả nội dung trong Textbox.
  - Lưu ý kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.



FontStyle.Bold  
FontStyle.Italic  
FontStyle.Underline  
*(Dùng toán tử ^ để kết hợp nhiều style)*

Gợi ý: thay đổi Font Style cho nội dung Textbox

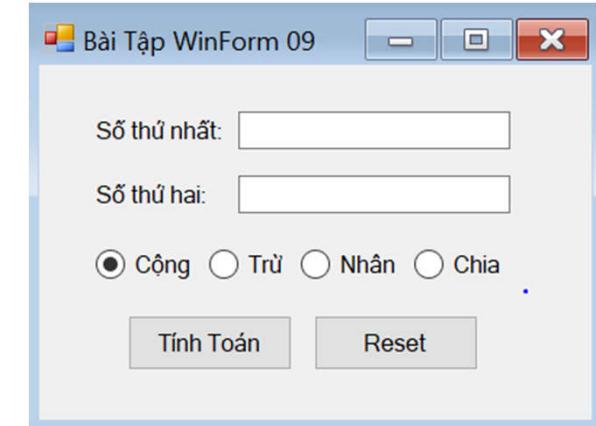
```
<textbox>.Font = new Font(SystemFonts.DefaultFont.FontFamily,  
SystemFonts.DefaultFont.Size, <font_style>);
```

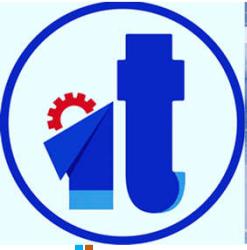


# Bài tập

## ☐ Thiết kế giao diện chương trình như hình bên

- Chọn radio Button:
  - ✓ Cộng / Trừ / Nhân / Chia
- Nhập **2 số nguyên a, b** vào các Textbox, sau đó nhấn vào Button **Tính toán** thì sẽ hiển thị kết quả của biểu thức a <**phép tính được chọn**> b lên MessageBox.
- Nhấn vào nút **Reset** thì xoá tất cả nội dung trong các Textbox.
- Lưu ý kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.





## ListBox, CheckedListBox, Combobox

### □ **ListBox**

- Cung cấp một **danh sách các item** cho phép user chọn
- ListBox cho phép hiển thị scroll nếu các item vượt quá vùng thể hiện của ListBox

### Properties

Items

MultiColumn

**SelectedIndex**

SelectedItem

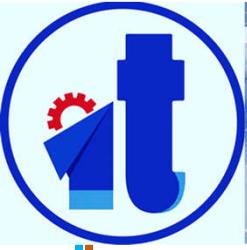
SelectionMode

SelectedItems

Sorted

Text

ListBox



## ListBox, CheckedListBox, Combobox

### □ **ListBox**

#### ➤ Method & Event

##### Method

ClearSelected()

GetSelected()

SetSelected()

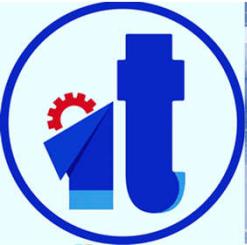
FindString()

##### **ListBox**



SelectedIndexChanged

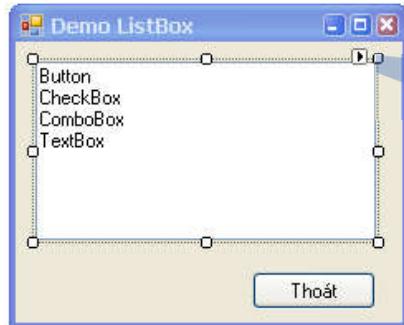
SelectedValueChanged



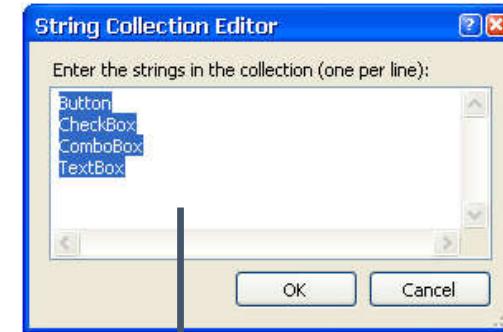
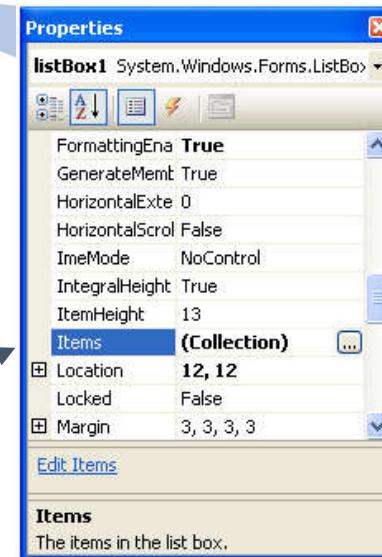
## ListBox, CheckedListBox, Combobox

### ☐ **ListBox**

➤ Thuộc tính **Items** cho phép thêm item vào ListBox



Danh sách item



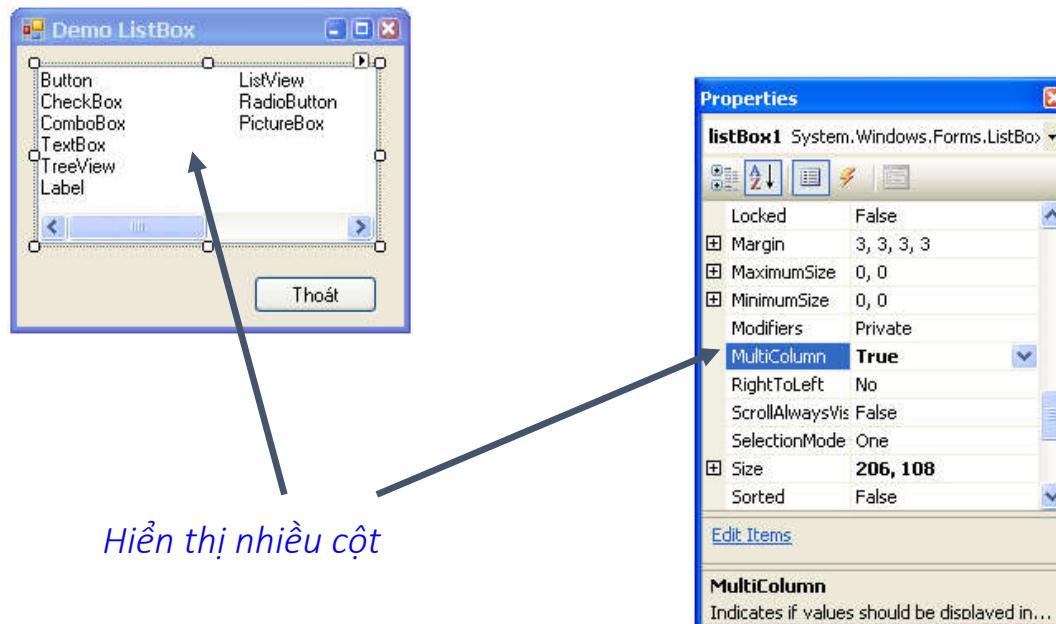
Cho phép thêm item trong  
màn hình thiết kế form



## ListBox, CheckedListBox, Combobox

### ☐ ListBox

- Hiển thị dạng Multi Column

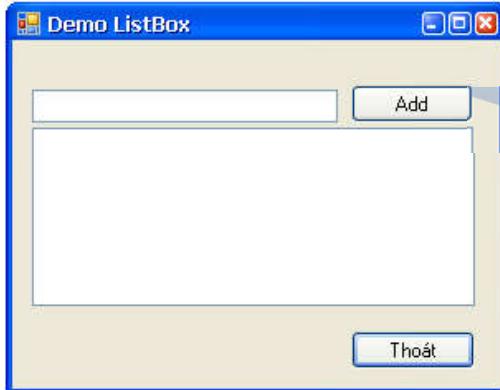




## ListBox, CheckedListBox, Combobox

### ☐ ListBox

#### ➤ Demo



```
private void button2_Click(object sender, EventArgs e)
{
    if (listBox1.Items.IndexOf(textBox1.Text)>=0)
        listBox1.SelectedItem = textBox1.Text;
    else if (textBox1.Text.Length>0)
        listBox1.Items.Add(textBox1.Text);
}
```

Kiểm tra xem chuỗi nhập có trong list box?

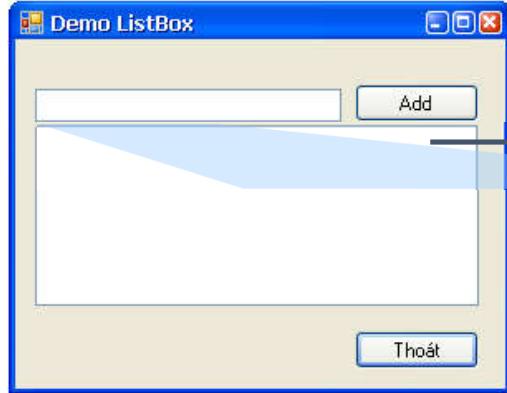
- Nếu có: select item đó
- Ngược lại: thêm chuỗi mới vào list box



## ListBox, CheckedListBox, Combobox

### ☐ ListBox

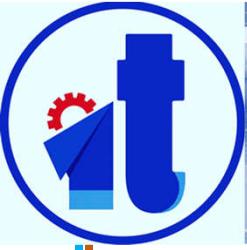
➤ Demo → Sự kiện *SelectedIndexChanged*



*SelectedIndexChanged*

Mỗi khi kích chọn vào item trong  
listbox ⇒ sẽ xóa item được chọn  
tương ứng

```
private void listBox1_SelectedIndexChanged(object sender,
    EventArgs e)
{
    if (listBox1.SelectedIndex >= 0)
        listBox1.Items.RemoveAt(listBox1.SelectedIndex);
}
```



## ListBox, CheckedListBox, Combobox

### ❑ CheckListBox

- Tương tự như list box nhưng mỗi item sẽ có thêm check box.

*Properties*

CheckedItems

CheckedIndices

SelectedItems

SelectedIndices

MultiColumn

Items



*Method*

ClearSelected

SetSelected



*Event*

SelectedIndexChanged

SelectedValueChanged

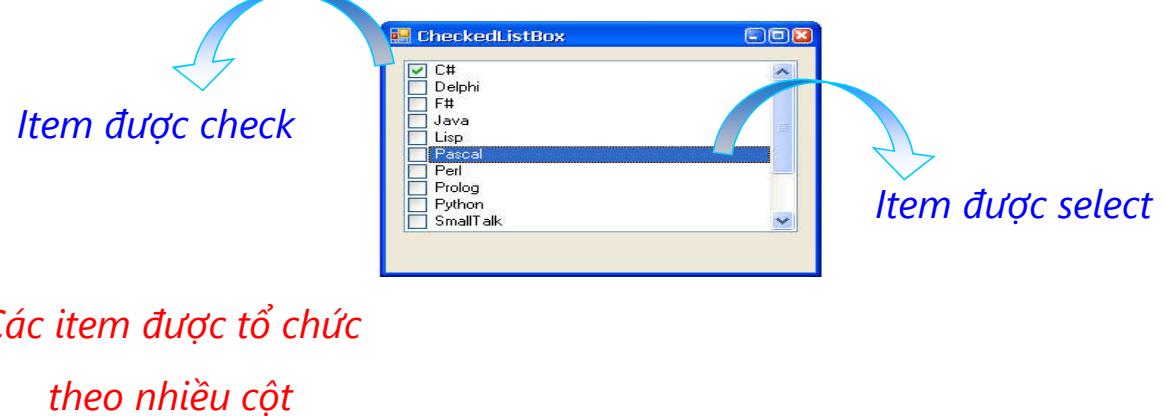
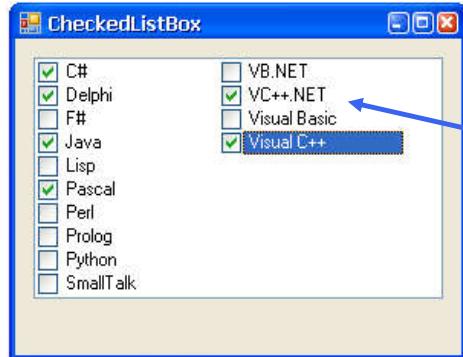
ItemCheck



## ListBox, CheckedListBox, Combobox

### ❑ CheckListBox

- Thuộc tính **Items** lưu trữ danh sách các item
- Có thể bổ sung một item vào thời điểm:
  - ✓ Design time
  - ✓ Run time
- **MultiColumn = true**

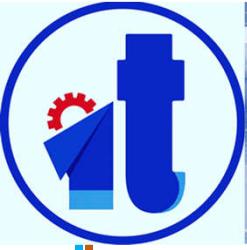


Các item được tổ chức  
theo nhiều cột



Item được select





## ListBox, CheckedListBox, Combobox

### ❑ CheckListBox

➤ Các phương thức phổ biến

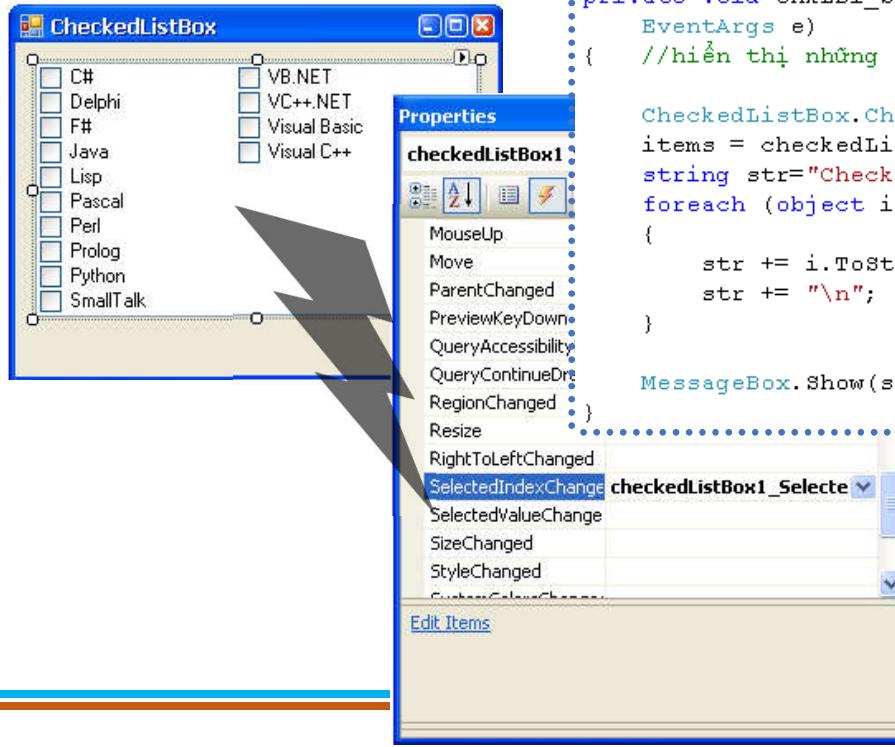
- ✓ void ClearSelected()
- ✓ bool GetSelected(int index)
- ✓ void SetSelected(int index, bool value)
- ✓ bool GetItemChecked(int index)
- ✓ CheckState GetItemCheckState(int index)
- ✓ void SetItemChecked(int index, bool value)
- ✓ void SetItemCheckedState(int index, CheckState value)



## ListBox, CheckedListBox, Combobox

### ❑ CheckedListBox

➤ Sự kiện SelectedIndexChanged



```
private void chkLB1_SelectedIndexChanged(object sender,
EventArgs e)
{
    //hiển thị những item được check

    CheckedListBox.CheckedItemCollection items;
    items = checkedListBox1.CheckedItems;
    string str="Checked items: \n";
    foreach (object i in items)
    {
        str += i.ToString();
        str += "\n";
    }

    MessageBox.Show(str);
}
```



## ListBox, CheckedListBox, Combobox

### □ Combobox

- Kết hợp TextBox với một danh sách dạng drop down
- Cho phép user kích chọn item trong danh sách drop down

Items

DropDownStyle

Text

**ComboBox**

Sorted

MaxDropDownItems

AutoCompleteMode

DropDownHeight

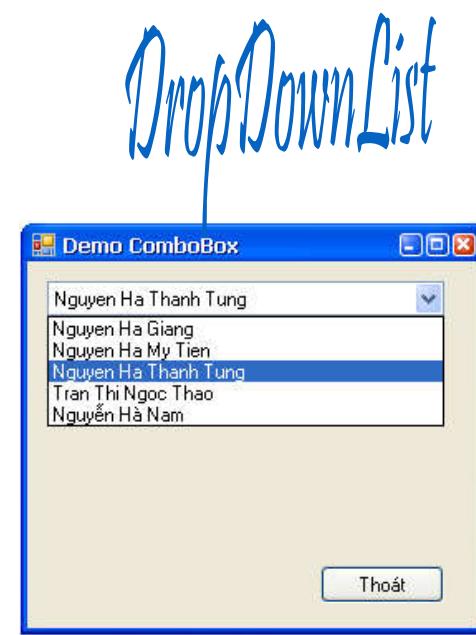
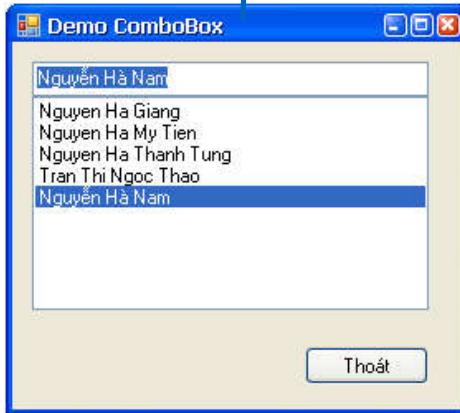


## ListBox, CheckedListBox, Combobox

### □ Combobox

➤ DropDownListStyle

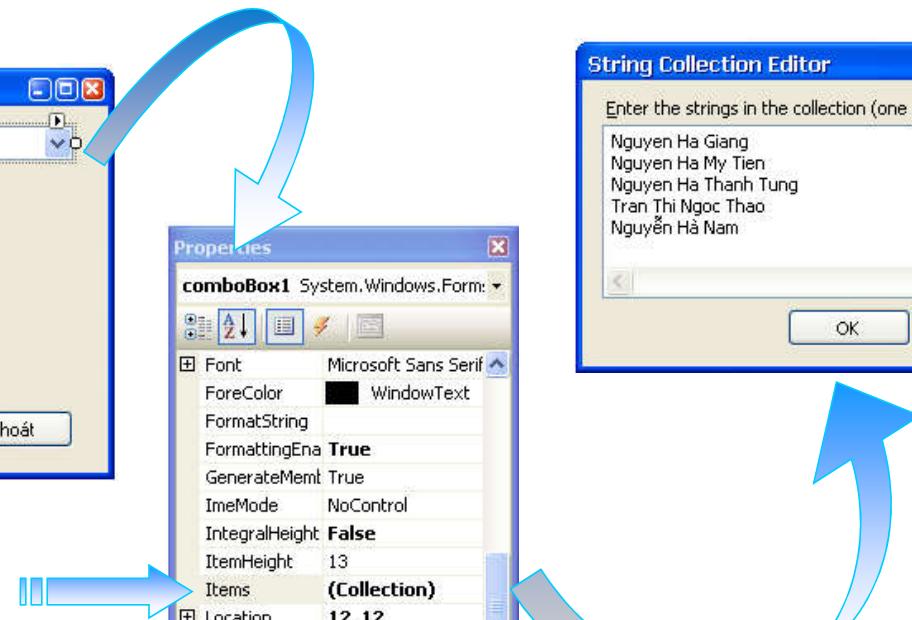
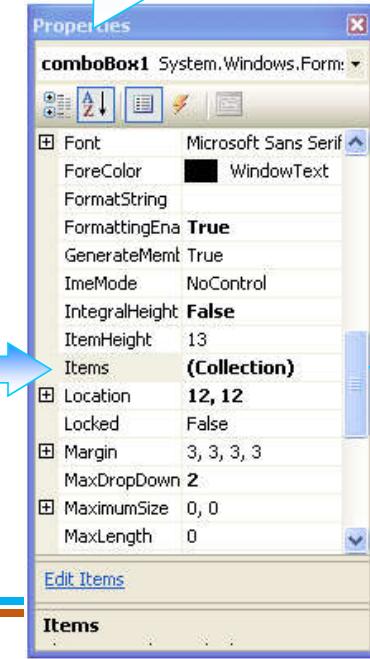
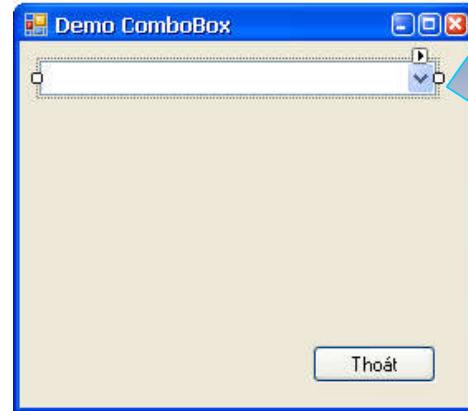
Simple





# Combobox, ListView,TreeView

## □ Combobox ➤ Demo 1

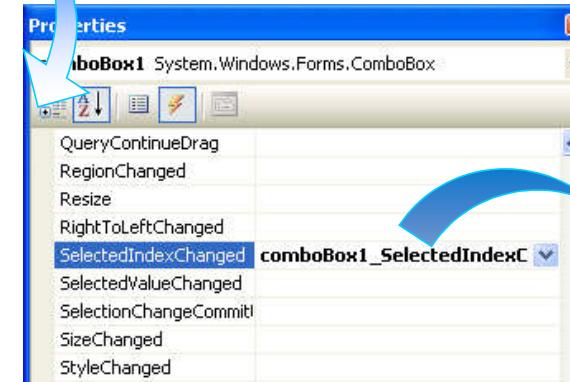
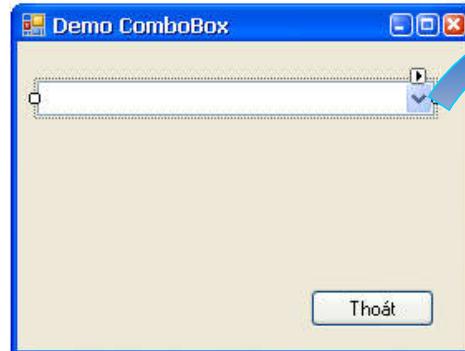


Bổ sung item trong  
màn hình design view



## ListBox, CheckedListBox, Combobox

### □ Combobox ➤ Demo 1



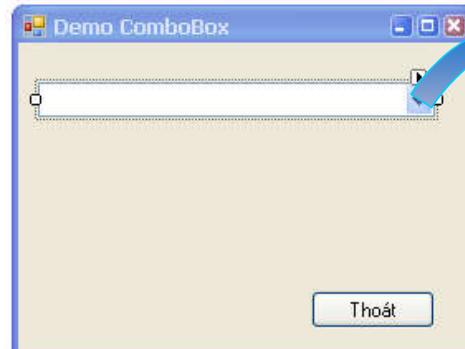
```
private void comboBox1_SelectedIndexChanged(object sender,
    EventArgs e)
{
    string str;
    str = comboBox1.Items[comboBox1.SelectedIndex].ToString();
    MessageBox.Show(str);
}
```

Mỗi khi kích chọn một item  
⇒ hiển thị item được chọn  
trên MessageBox

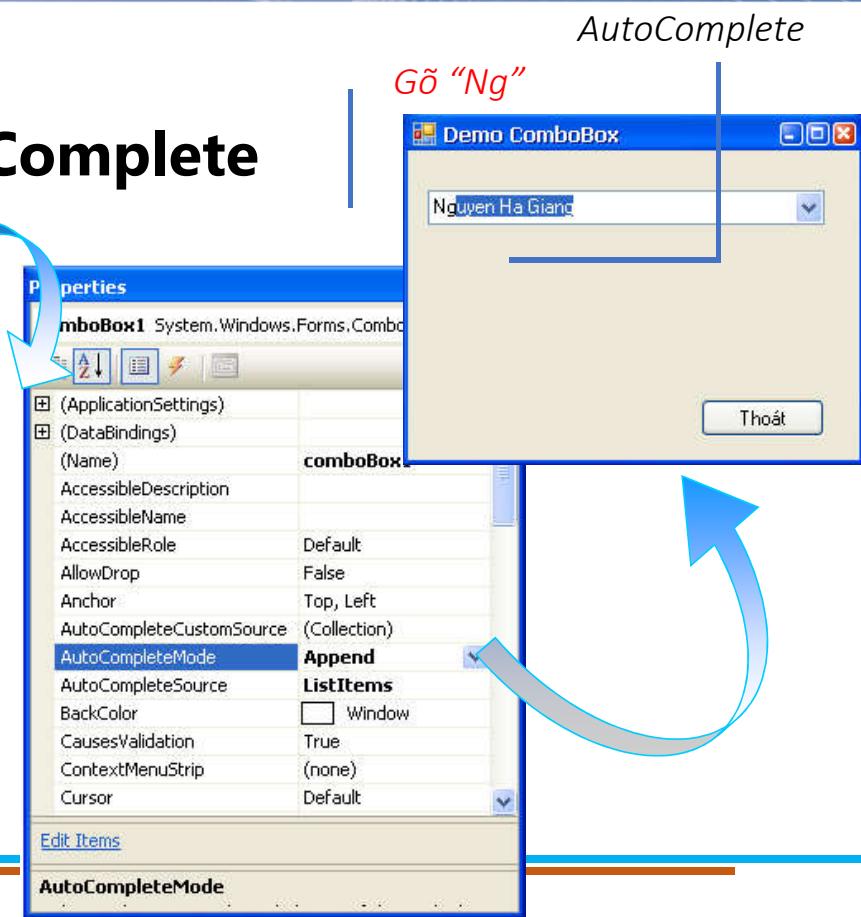


# Combobox

## ➤ Demo 1 → Tính năng AutoComplete



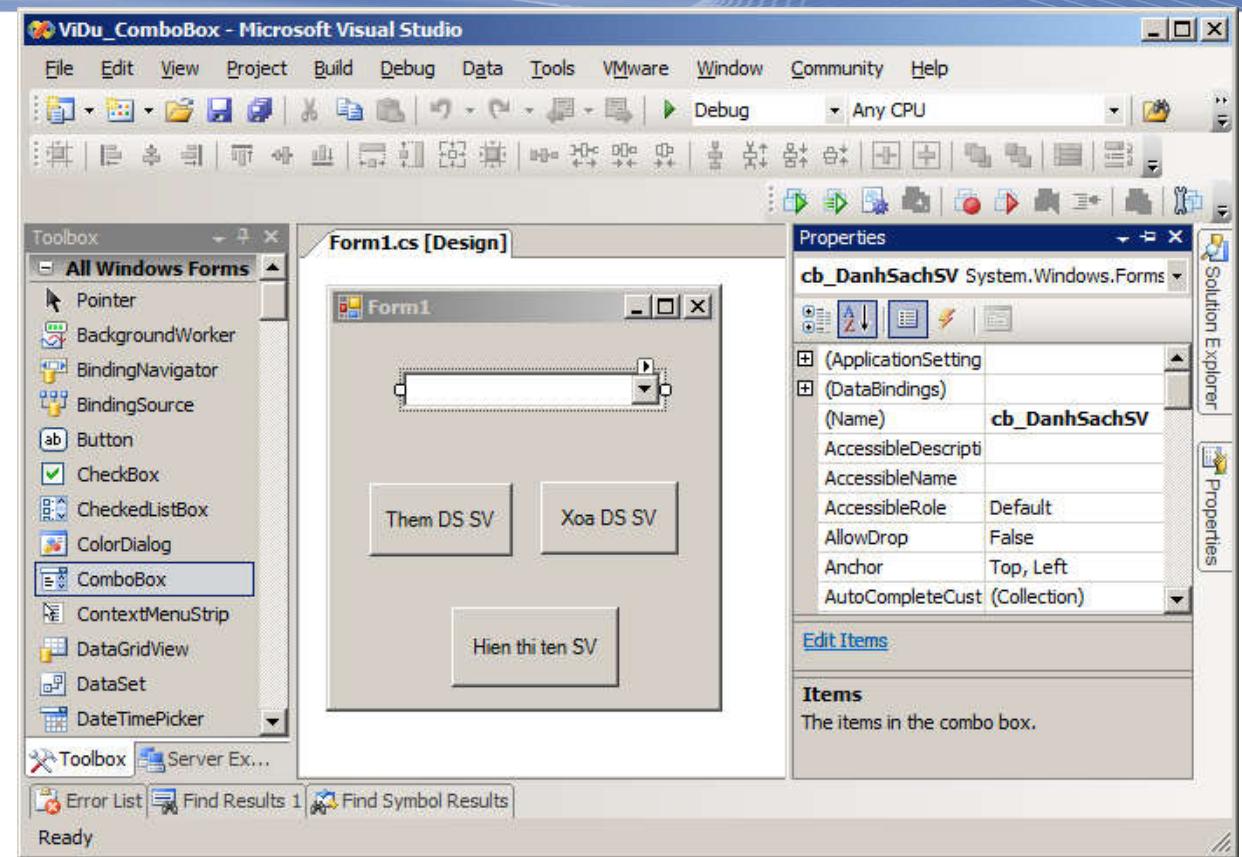
## AutoCompleteMode





## □ Combobox

- Demo 2
- Thiết kế giao diện





## ListBox, CheckedListBox, Combobox

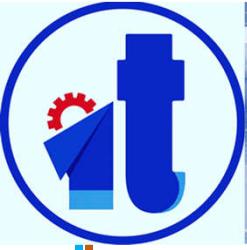
### □ Combobox

- Demo 2
- Code

```
private void btn_ThemDSSV_Click(object sender, EventArgs e)
{
    cb_DanhSachSV.Items.Add("Nguyen Van A");
    cb_DanhSachSV.Items.Add("Le Thi B");
    cb_DanhSachSV.Items.Add("Tran Van C");
}

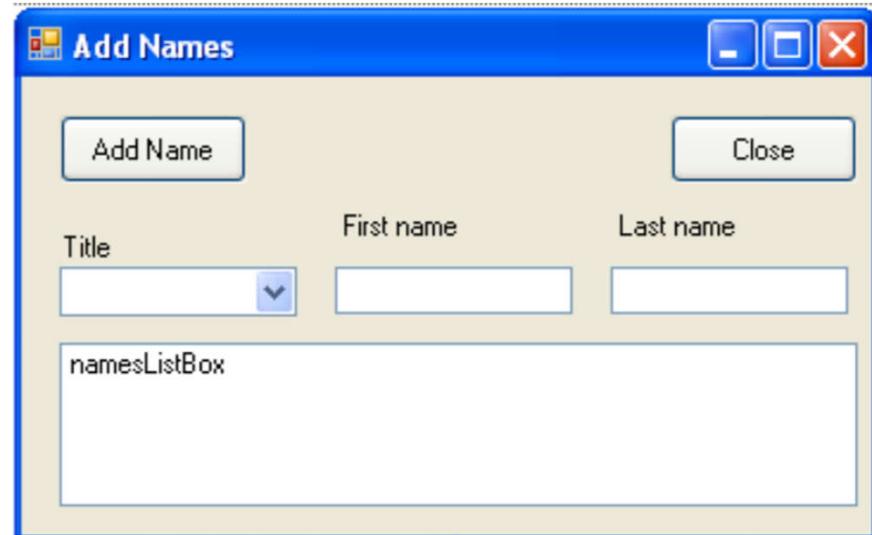
private void btn_XoaDSSV_Click(object sender, EventArgs e)
{
    //xoá toàn bộ danh sách
    cb_DanhSachSV.Items.Clear();
}

private void btn_HienThiDS_Click(object sender, EventArgs e)
{
    string s;
    s = cb_DanhSachSV.SelectedItem.ToString();
    MessageBox.Show(s);
}
```



# Bài tập

- ❑ Thiết kế giao diện chương trình như hình bên
  - Danh sách items trong Combobox  
Title: Dr, Mr, Mrs, Miss
  - Chọn **Title**, nhập **First name** và **Last name** vào Textbox, sau đó nhấn vào Button **Add Name** thì sẽ thêm 1 phần tử có dạng **<Title>. <First name> <Last name>** vào cuối danh sách trong Listbox.
  - Lưu ý kiểm tra và thông báo lên MessageBox nếu chưa nhập dữ liệu.

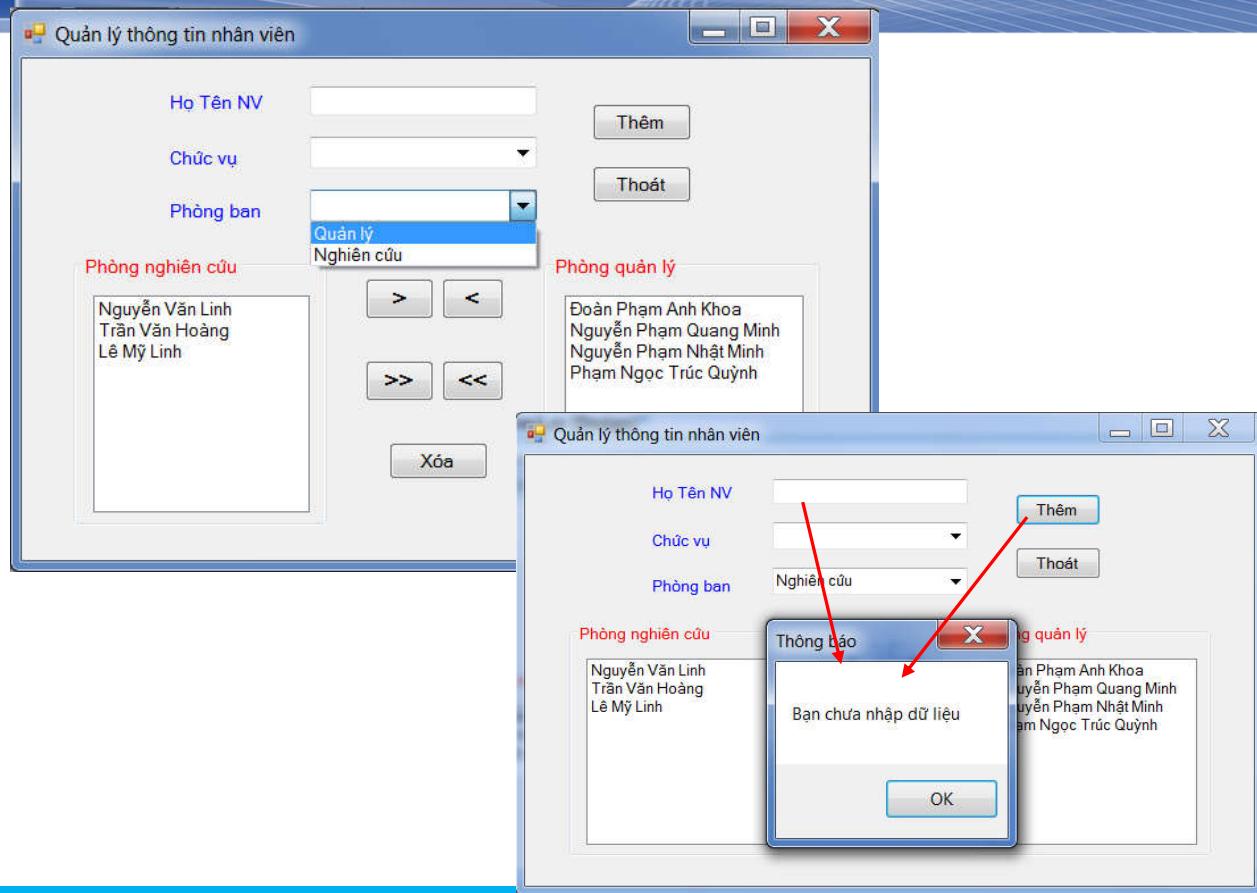


# Bài tập



LHN

- Mặc định phòng ban là phòng nghiên cứu.
- Khi nhấn nút **Thêm**, giá trị nhập tại ô họ tên nhân viên sẽ được cập nhật vào danh sách phòng ban tương ứng được chọn.
- Khi nhấn vào các nút > (ToRight), < (ToLeft), >>(ToRightAll), <<(ToLeftAll) danh sách các đối tượng được chọn sẽ di chuyển tương ứng và mất khỏi danh sách ban đầu.  
(Có hiển thị chú thích: áp dụng ToolTip)
- Khi nhấn nút **xóa** thì đối tượng tương ứng được chọn sẽ xóa khỏi danh sách.
- Khi nhấn nút thoát thì đóng form, thoát khỏi chương trình.





## NumericUpDown, DomainUpDown, TrackBar

### ❑ NumericUpDown



- Cho phép user chọn các giá trị (dạng số) trong khoảng xác định thông qua:
  - ✓ Nút **up & down**
  - ✓ Nhập trực tiếp giá trị
- Các thuộc tính:
  - ✓ 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
- Phương thức:
  - ✓ void DownButton()
  - ✓ void UpButton()
- Sự kiện:
  - ✓ ValueChanged: xảy ra khi thay đổi giá trị



# NumericUpDown, DomainUpDown, TrackBar

## ☐ NumericUpDown

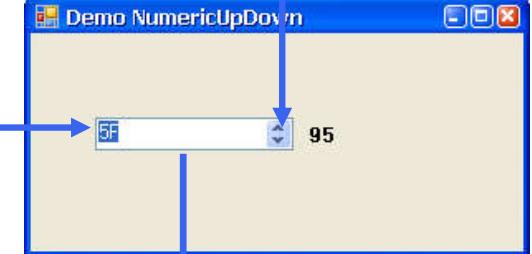
### ➤ Ví dụ

Đoạn code thêm control NumericUpDown

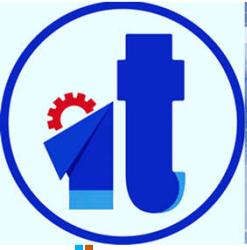
```
public void AddNumericUpDown () {  
    NumericUpDown numUpDn = new NumericUpDown ();  
    numUpDn.Location = new Point (50, 50);  
    numUpDn.Size = new Size (100, 25);  
    numUpDn.Hexadecimal = true; // hiển thị dạng hexa  
    numUpDn.Minimum = 0; // giá trị nhỏ nhất  
    numUpDn.Maximum = 255; // giá trị lớn nhất  
    numUpDn.Value = 0xFF; // giá trị khởi tạo  
    numUpDn.Increment = 1; // bước tăng/giảm  
    // thêm control vào ds control của form  
    Controls.Add (numUpDn);  
}
```

Hiển thị giá trị  
Hexa

Tăng giảm giá trị



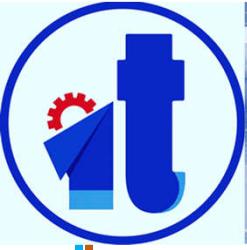
Nhập trực tiếp giá trị



## NumericUpDown, DomainUpDown, TrackBar

### **DomainUpDown**

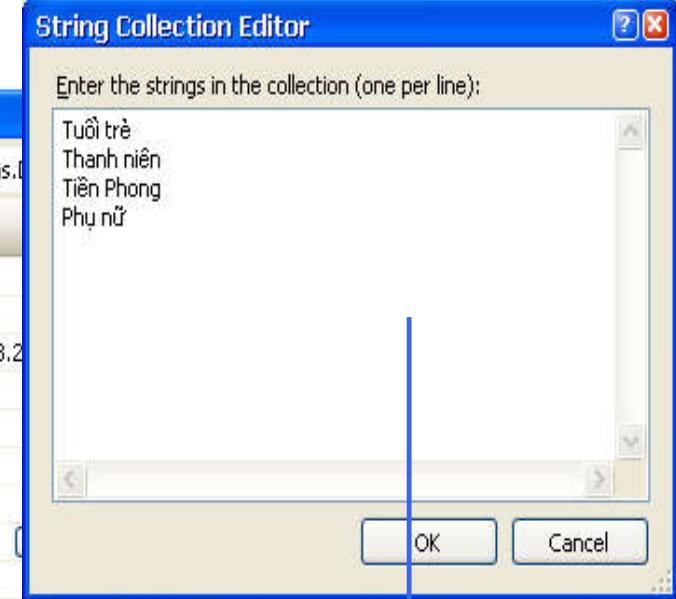
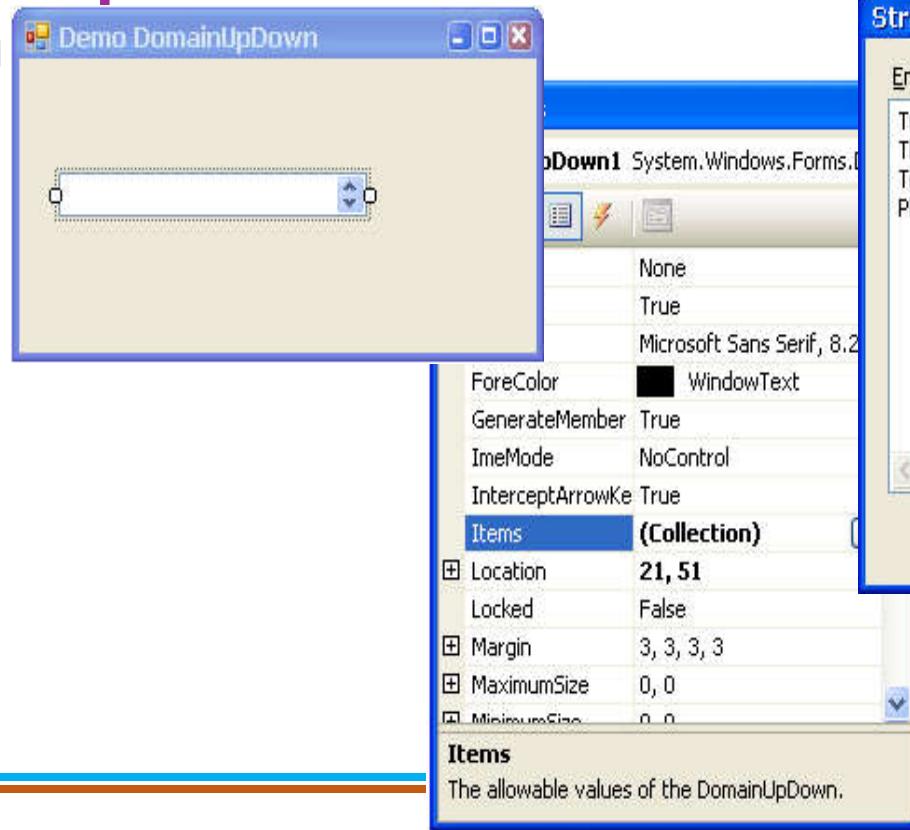
- Cho phép user chọn item trong số danh sách item thông qua:
  - ✓ Button Up & Down
  - ✓ Nhập từ bàn phím.
- Properties
  - ✓ [Items](#): danh sách item
  - ✓ [ReadOnly](#): true chỉ đổi giá trị qua Up & Down
  - ✓ [SelectedIndex](#): chỉ mục của item đang chọn
  - ✓ [SelectedItem](#): item đang được chọn
  - ✓ [Sorted](#): sắp danh sách item
  - ✓ [Text](#): text đang hiển thị trên DomainUpDown
- Một số phương thức:
  - ✓ `voidDownButton()`
  - ✓ `voidUpButton()`
- Event
  - ✓ [SelectedItemChanged](#)



# NumericUpDown, DomainUpDown, TrackBar

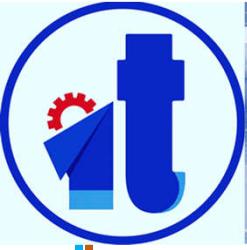
## ☐ DomainUpDown

➤ Ví dụ



String Collection Editor

Cho phép nhập item



## NumericUpDown, DomainUpDown, TrackBar

### **TrackBar**

- Cho phép user thiết lập giá trị (số nguyên) trong khoảng cố định cho trước
- Thao tác qua thiết bị chuột hoặc bàn phím

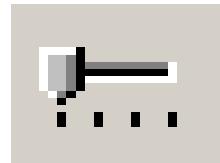
Minimum

Maximum

TickFrequency

TickStyle

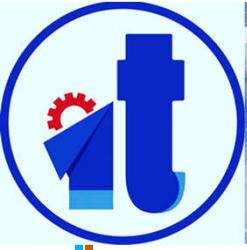
Value



 ValueChanged  
Scroll

Methods

SetRange



# NumericUpDown, DomainUpDown, TrackBar

## ▢ TrackBar

```
public void AddTrackBar() {  
    TrackBar tb1 = new TrackBar();  
    tb1.Location = new Point(10, 10);  
    tb1.Size = new Size(250, 50);  
  
    tb1.Minimum = 0;           → Thiết lập khoảng: 0 - 100  
    tb1.Maximum = 100;          → Số vị trí di chuyển khi dùng  
                                phím mũi tên  
    tb1.SmallChange = 1;          → Số vị trí di chuyển khi  
    tb1.LargeChange = 5;          → dùng phím Page  
    tb1.TickStyle = TickStyle.BottomRight;  
    tb1.TickFrequency = 10;       → Kiểu stick ở bên dưới/bên  
    tb1.Value = 10;                → phải track  
    Controls.Add(tb1);  
}
```

Tạo thể hiện

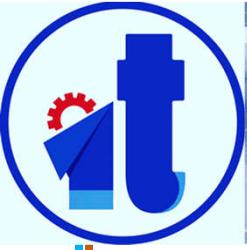
Thiết lập khoảng: 0 - 100

Số vị trí di chuyển khi dùng  
phím mũi tên

Số vị trí di chuyển khi  
dùng phím Page

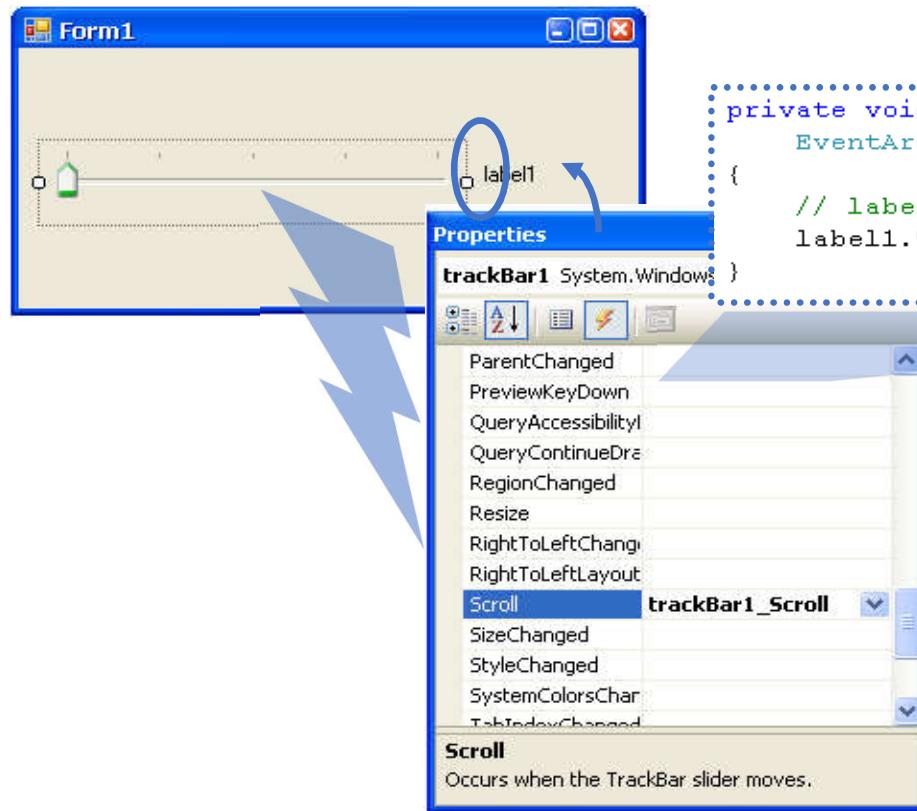
Kiểu stick ở bên dưới/bên  
phải track

Số khoảng cách giữa các  
tick mark



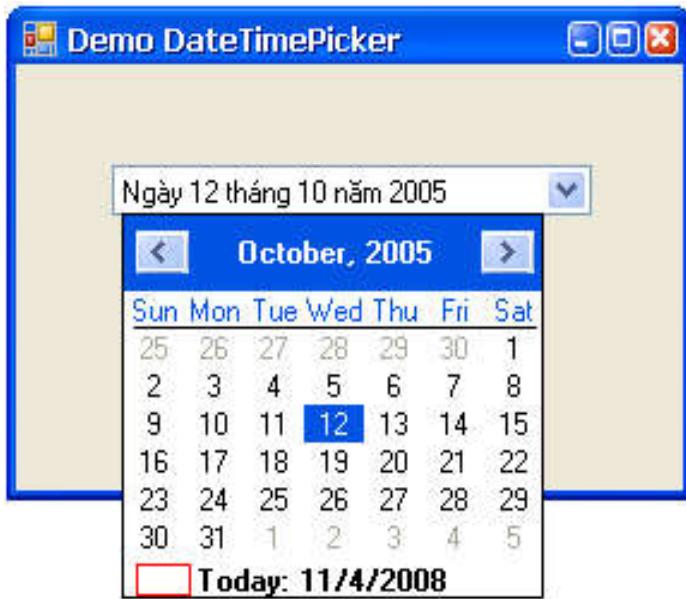
## NumericUpDown, DomainUpDown, TrackBar

### □ TrackBar





## DateTimePicker, MonthCalendar





## DateTimePicker, MonthCalendar

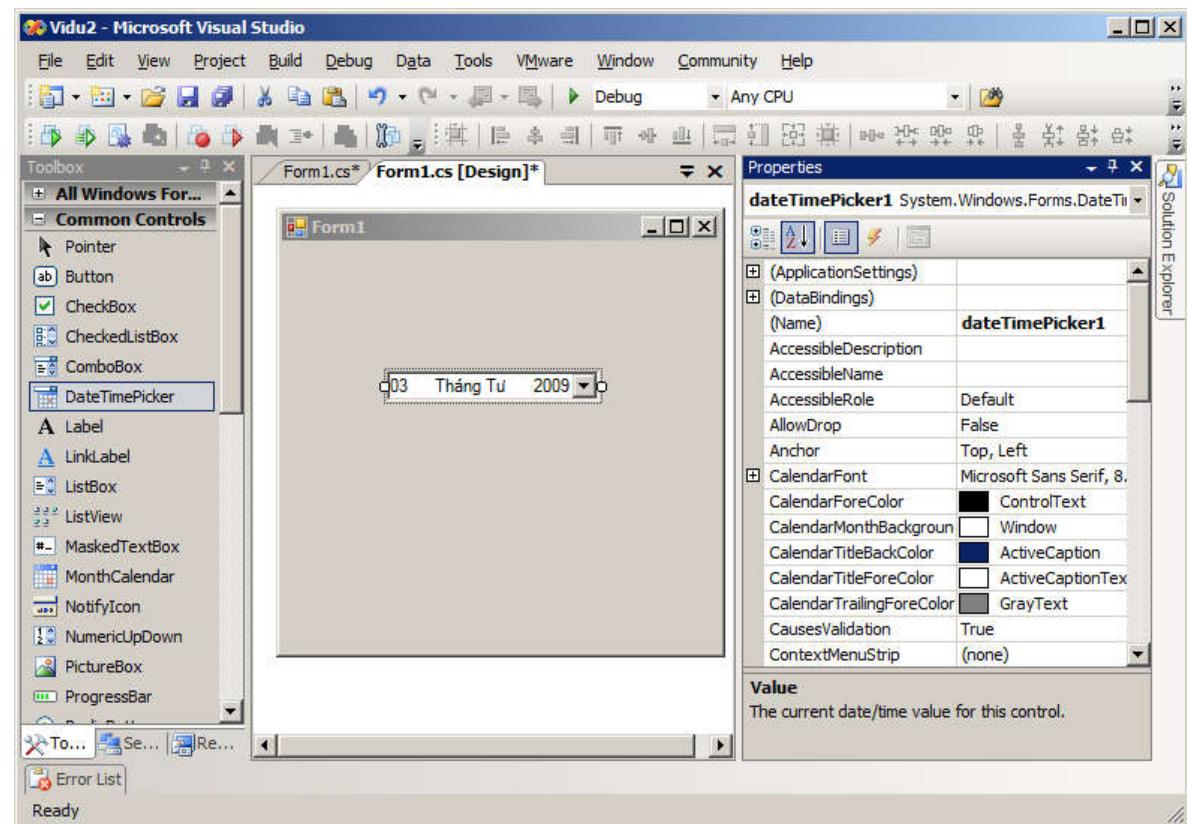
### DateTimePicker

- Cho phép chọn ngày, giờ trong khoảng xác định thông qua giao diện đồ họa dạng calendar
- Kết hợp ComboBox và MonthCalendar
- **Properties**
  - ✓ Format: định dạng hiển thị: long, short, time, custom
  - ✓ CustomFormat:
    - Dd/MM: hiển thị 2 con số của ngày/tháng
    - yyyy: hiển thị 4 con số của năm
    - <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.datetimepicker.customformat?view=netframework-4.8>
  - ✓ MaxDate: giá trị ngày lớn nhất
  - ✓ MinDate: giá trị ngày nhỏ nhất
  - ✓ Value: giá trị ngày hiện tại đang chọn



# DateTImePicker, MonthCalendar

## □ DateTImePicker



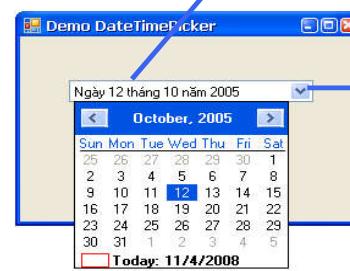


## DatePicker, MonthCalendar

### ☐ DatePicker

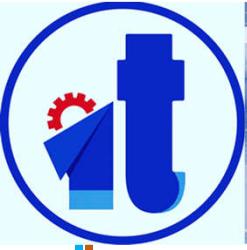
```
private void AddDatePicker() {  
    DateTimePicker DTPicker = new DateTimePicker();  
    DTPicker.Location = new Point(40, 80);  
    DTPicker.Size = new Size(160, 20);  
    DTPicker.DropDownAlign = LeftRightAlignment.Right;  
    DTPicker.Value = DateTime.Now;  
    DTPicker.Format = DateTimePickerFormat.Custom;  
    DTPicker.CustomFormat = "'Ngày' dd 'tháng' MM 'năm' yyyy";  
    this.Controls.Add(DTPicker);  
}
```

Định dạng xuất: 'Ngày' dd 'tháng' MM 'năm' yyyy



Kích drop down để  
hiện thị hộp chọn ngày

Chọn ngày trong  
khoảng cho trước



## DateTimePicker, MonthCalendar

### □ MonthCalendar

➤ Cho phép user chọn một ngày trong tháng hoặc nhiều ngày với ngày bắt đầu và ngày kết thúc.

#### ➤ **Properties:**

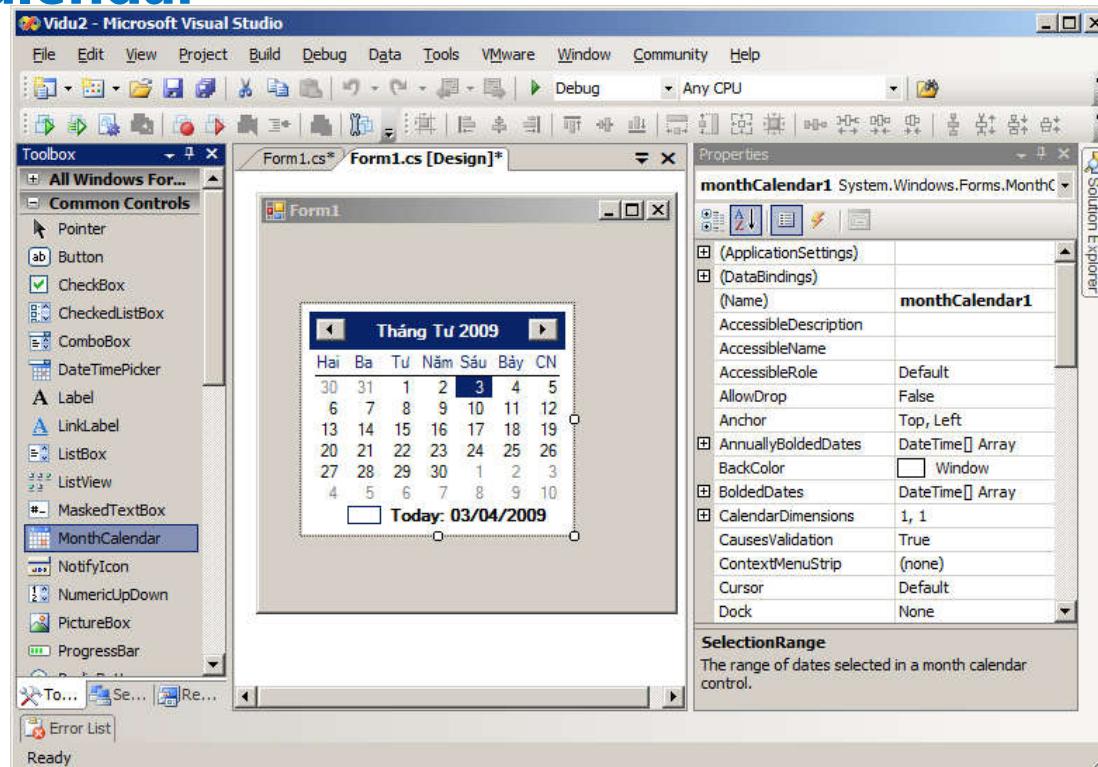
- ✓ MaxDate, MinDate
- ✓ SelectionStart: ngày bắt đầu chọn
- ✓ SelectionEnd: ngày kết thúc

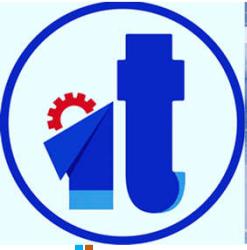




# DateTImePicker, MonthCalendar

## □ MonthCalendar





# ListView, TreeView

## □ **ListView**

- Dạng control phổ biến hiển thị 1 danh sách item
  - ✓ Các item có thể có các item con gọi là subitem
- Windows Explorer hiển thị thông tin Folder, file...
- Có thể hiển thị thông tin theo nhiều dạng thông qua thuộc tính View
  - ✓ Xem dạng chi tiết thông tin
  - ✓ Xem dạng icon nhỏ
  - ✓ Xem dạng icon lớn
  - ✓ Xem dạng tóm tắt,...
- Lớp ListView dẫn xuất từ System.Windows.Forms.Control



## ListView, TreeView

### **ListView**

➤ Các thuộc tính:

Columns

Items

Sorting

GridLines

# ListView

MultiSelect

View

SmallImageList

LargeImageList

FullRowSelect



## ListView, TreeView

### ListView

➤ Các dạng thể hiện của ListView:

*Details*

*Small Icons*

# ListView

*List*

*Large Icons*

*Tile*



# ListView,TreeView

## ❑ ListView

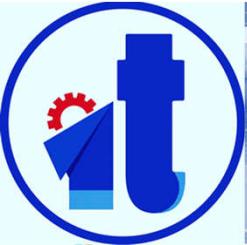
➤ Các dạng thể hiện của ListView:



*Large Icons*

*Mỗi item xuất hiện với 1 icon kích thước lớn  
và một label bên dưới*

Dạng hiển thị mặc định của ListView



# ListView, TreeView

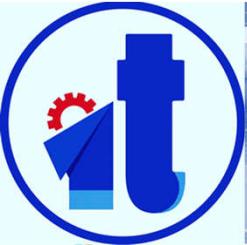
## ❑ ListView

➤ Các dạng thể hiện của ListView:



*Small Icons*

Mỗi item xuất hiện với icon nhỏ và một label bên phải



# ListView, TreeView

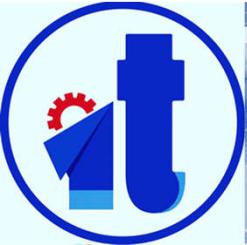
## □ ListView

➤ Các dạng thể hiện của ListView:



List

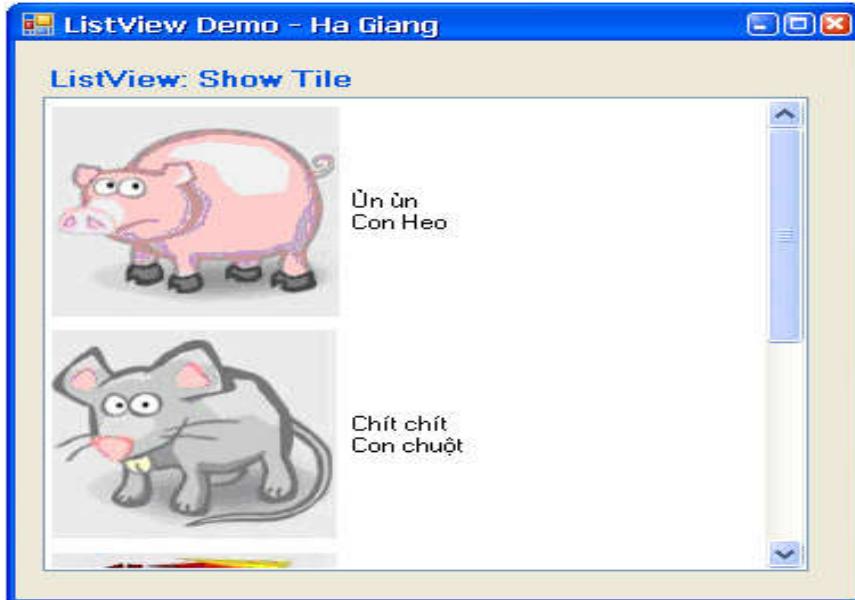
Mỗi item xuất hiện với icon nhỏ với label bên phải, item được sắp theo cột nhưng không có tiêu đề cột



# ListView,TreeView

## ☐ ListView

➤ Các dạng thể hiện của ListView:



Tile

Mỗi item xuất hiện với icon kích thước lớn, bên phải  
có label chứa item và subitem



# ListView,TreeView

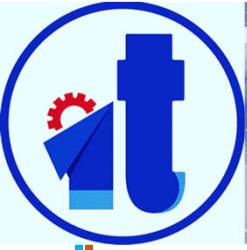
## ☐ ListView

➤ Các dạng thể hiện của ListView:

Hành động	Động vật
	Ùn ùn
	Chít chít
	Ó ó
	Cập cập

Detail

Mỗi item xuất hiện trên một dòng, mỗi dòng có các cột chứa thông tin chi tiết



# ListView, TreeView

## ❑ ListView

➤ Các thuộc tính:

✓ View

○ `View View = View.LargeIcon; // default`

○ Mô tả:

- Cách hiển thị các phần tử trong ListView

- Các giá trị của View là: LargeIcon, SmallIcon, Details, List, Tile

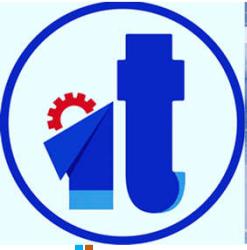
✓ CheckBoxes

○ `bool CheckBoxes = false; // default`

○ Mô tả:

- Hiển thị checkbox bên cạnh mỗi phần tử trong ListView

- Không hiển thị checkbox trong khung nhìn Tile



# ListView, TreeView

## ❑ ListView

➤ Các thuộc tính:

✓ Items

- `ListView.CheckedListViewItemCollection` Items;
- Mô tả: Danh sách các phần tử trong ListView

✓ CheckedItems

- `ListView.CheckedListViewItemCollection` CheckedItems;
- Mô tả:

- Danh sách các phần tử được chọn (check vào checkbox)
- Được sử dụng khi thuộc tính CheckBoxes = true



# ListView, TreeView

## □ **ListView**

➤ Các thuộc tính:

✓ SelectedItems

○ `ListView.SelectedItemsCollection` `SelectedItems`;

○ Mô tả: Danh sách các phần tử được chọn

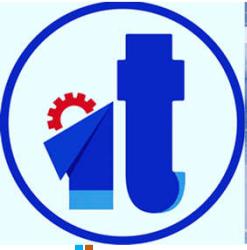
✓ SelectedIndices

○ `ListView.SelectedIndexCollection` `SelectedIndices`;

○ Mô tả:

- Danh sách chỉ số của các phần tử được chọn trong ListView hay còn gọi là mảng chỉ số chọn

- Chỉ số có kiểu dữ liệu là `int`



# ListView, TreeView

## ❑ ListView

➤ Các thuộc tính:

✓ FocusItem

○ `ListViewItem FocusItem;`

○ Mô tả:

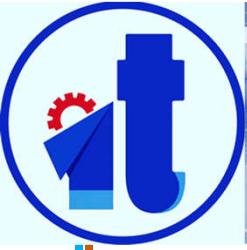
- Phần tử đang được trỏ đến trong ListView
- Ngược lại, FocusItem = null

✓ Columns

○ `ListView.ColumnHeaderCollection Columns;`

○ Mô tả:

- Danh sách các cột trong ListView
- Các cột được hiển thị trong khung nhìn Details



# ListView, TreeView

## ❑ ListView

➤ Các thuộc tính:

✓ FullRowSelect

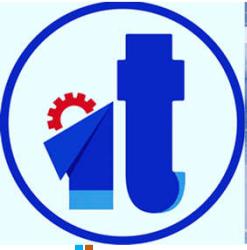
○ `bool FullRowSelect = false; // default`

○ Mô tả: Cho phép tô hết dòng của phần tử được chọn trong ListView (khung nhìn Details)

✓ GridLines

○ `bool GridLines = false; // default`

○ Mô tả: Cho phép hiển thị các đường kẻ quanh các phần tử trong ListView (khung nhìn Details)



# ListView, TreeView

## ❑ ListView

➤ Các thuộc tính:

✓ LabelEdit

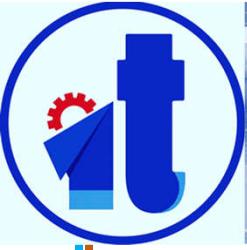
○ `bool LabelEdit = false; // default`

○ Mô tả: Cho chỉnh sửa nội dung của nhãn của phần tử trong ListView khi thực thi (run time)

✓ LargeImageList

○ `ImageList LargeImageList = null; // default`

○ Mô tả: Danh sách các hình ảnh được dùng trong khung nhìn LargeIcon, Tile



# ListView, TreeView

## ❑ ListView

➤ Các thuộc tính:

✓ SmallImageList

○ `ImageList SmallImageList = null; // default`

○ Mô tả: Danh sách các hình ảnh được dùng trong khung nhìn SmallIcon, List, Details

✓ MultiSelect

○ `bool MultiSelect = true; // default`

○ Mô tả: Cho phép chọn nhiều phần tử trong ListView

✓ Scrollable

○ `bool Scrollable = true; // default`

○ Mô tả: Cho phép hiển thị thanh cuộn trên điều khiển ListView

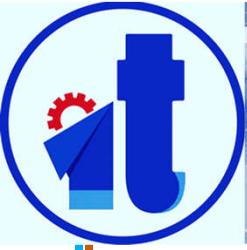


# ListView,TreeView

## □ **ListView**

➤ Các phương thức:

- ✓ Phương thức khởi tạo
  - Cú pháp: `public ListView()`
  - Mô tả: Dùng để tạo một điều khiển ListView
- ✓ `Clear()`
  - Cú pháp: `public void Clear()`
  - Mô tả: Dùng để xóa các cột và các items



# ListView, TreeView

## ❑ ListView

➤ Các sự kiện:

- ✓ **SelectedIndexChanged**: Xảy ra khi mảng chỉ số chọn SelectedIndices thay đổi
- ✓ **Click**: Xảy ra khi click vào điều khiển ListView
- ✓ **DoubleClick**: Xảy ra khi click đúp vào điều khiển ListView



# ListView,TreeView

## □ ListView

### ➤ Các thao tác:

- ✓ Thêm các cột – Detail View
  - Cửa sổ *properties* → *Columns* để tạo.
  - Sử dụng *code* trong chương trình.

```
listView1.Columns.Add("<Tên cột>");
```

- ✓ Thêm các item vào ListView

- Thêm item trong màn hình thiết kế form
  - Thêm item thông qua code

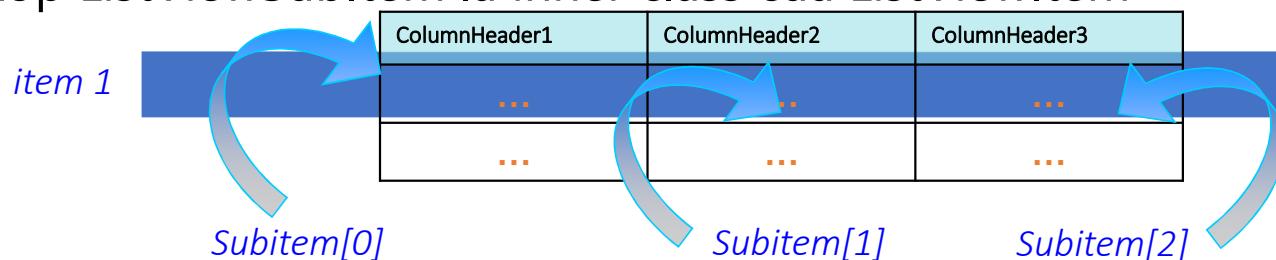


# ListView, TreeView

## ☐ ListView

### ➤ Các lớp định nghĩa Item:

- ✓ [System.Windows.Forms.ListViewItem](#)
- ✓ Mỗi item trong ListView có các item phụ gọi là subitem
  - Lớp [ListViewItem.ListViewSubItem](#) định nghĩa các subitem của ListView
  - Lớp ListViewSubItem là inner class của ListViewItem



```
ListViewItem item1 = new ListViewItem("Marketing");
item1.SubItems.Add("306 NTT - Q.TB");
listView1.Items.Add(item1);
```



# ListView,TreeView

## ☐ ListView

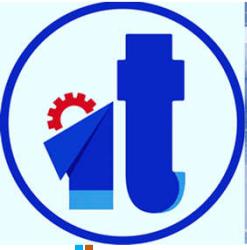
➤ Sự kiện *SelectedIndexChanged*

Properties

- listView1 System.Windows.Forms.ListView
- Resize
- RetrieveVirtualItem
- HeightToLayoutChanged
- WidthToLayoutChanged
- InForVirtualItem
- SelectedIndexChanged**
- SizeChanged
- StyleChanged
- SystemColorsChanged
- TabIndexChanged
- TabStopChanged
- Validated
- Validating
- VirtualItemsSelectionChanged
- VisibleChanged

Edit Items, Edit Columns, Edit Groups

```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    //hiển thị thông tin item được chọn
    ListView.SelectedListViewItemCollection item;
    item = listView1.SelectedItems; // lấy item được chọn
    if (item.Count > 0) // nếu có item được chọn
    {
        string str1 = item[0].Text;
        string str2 = item[0].SubItems[1].Text;
        MessageBox.Show(str1 + "-" + str2);
    }
}
```



# ListView, TreeView

## ☐ **ListView / ListViewItem**

➤ Các thuộc tính:

✓ Checked

- `bool Checked = false; // default`
- `bool Focused = false; // default`

✓ Index

○ `int Index;`

○ Mô tả: Chỉ số của phần tử trong danh sách các phần tử

✓ ImageList

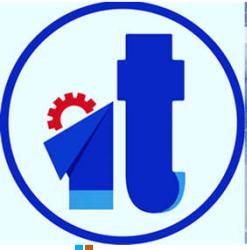
○ `ImageList ImageList;`

○ Mô tả: Danh sách các hình ảnh dùng để hiển thị với các phần tử

✓ ImageIndex

○ `int ImageIndex = -1; // default`

○ Mô tả: Chỉ số của hình ảnh trong ImageList dùng để hiển thị với phần tử



# ListView, TreeView

## ❑ **ListView / ListViewItem**

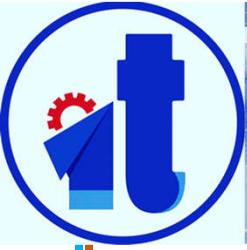
➤ Các thuộc tính:

✓ SubItems

- `ListViewItem.ListViewSubItemCollection SubItems;`
- Mô tả: Danh sách các phần tử con của phần tử

✓ Text

- `string Text;`
- Mô tả: Nội dung văn bản hiển thị của phần tử

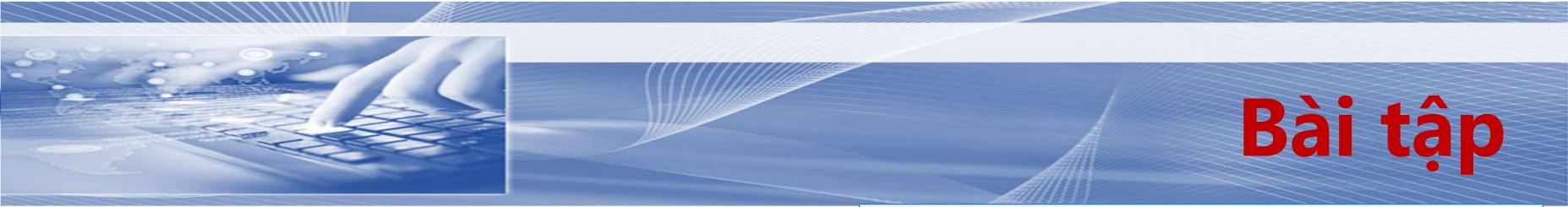
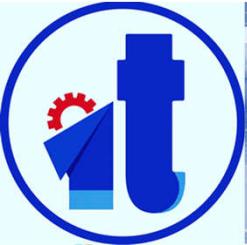


## ListView, TreeView

### ❑ **ListView / ListViewItem**

➤ Các phương thức khởi tạo:

- ✓ `public ListViewItem()`
- ✓ `public ListViewItem(string text)`
- ✓ `public ListViewItem(string text, int imageIndex)`



# Bài tập

Hãy viết chương trình ứng dụng thực hiện các chức năng theo giao diện được thiết kế như sau (Listview):

- Mặc định giới tính là Nam (Nam, Nữ), trạng thái hoạt động (hoạt động, Khóa).
- Người dùng nhập thông tin tài khoản sau đó bấm nút thông tin sẽ được lưu xuống listview
- Khi người dùng click vào một dòng bất kỳ trên listview, dữ liệu của dòng đó được binding lên groupbox thông tin chi tiết:
  - Nếu người dùng chỉnh sửa thông tin và bấm thông tin sẽ được cập nhật xuống listview
  - Người dùng bấm thông tin tài khoản bị xóa khỏi listview
  - Khi người dùng bấm vào thông tin chi tiết trên groupbox về lại trạng thái mặc định (làm trống những điều khiển nhập)

Quản lý tài khoản

Thông tin chi tiết

Tài khoản:	<input type="text"/>	
Họ tên:	<input type="text"/>	
Ngày sinh:	<input type="date" value="Sunday, October 28, 2018"/>	
Giới tính:	<input type="radio"/> Nam	<input type="radio"/> Nữ
Trạng thái:	<input type="checkbox"/> Hoạt động	

Tài khoản	Họ tên	Ngày sinh	Giới tính	Trạng thái



# ListView, TreeView

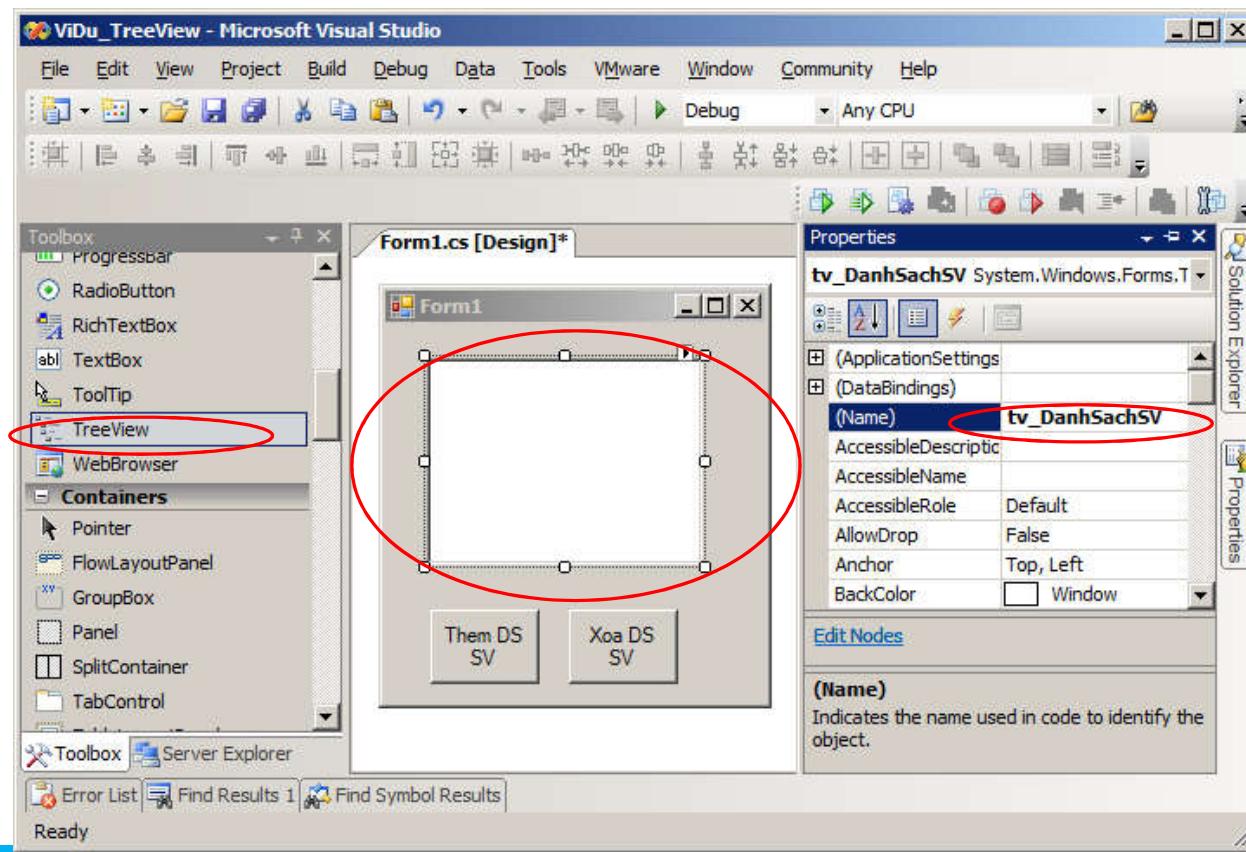
## □ TreeView

- Dạng control hiển thị danh sách các phần tử (node) theo hệ thống phân cấp dạng hình cây
- **Các thuộc tính** thông dụng:
  - ✓ **Nodes**: danh sách các phần tử của cây, mỗi node cũng có danh sách các Nodes con
  - ✓ **SelectedNode**: phần tử (node) đang được chọn
  - ✓ **ShowLines**: hiển thị các đường nối các node theo phân cấp
- **Các phương thức** thông dụng:
  - ✓ CollapseAll()
  - ✓ ExpandAll()
  - ✓ GetNodeAt()
  - ✓ GetNodeCount()



# ListView,TreeView

## TreeView





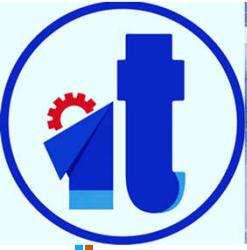
# ListView,TreeView

## ☐ TreeView

```
private void btn_ThemDSSV_Click(object sender, EventArgs e)
{
    tv_DanhSachSV.Nodes.Add("Nguyen van A");
    tv_DanhSachSV.Nodes[0].Nodes.Add("Ngay sinh: 01/01/1990");
    tv_DanhSachSV.Nodes[0].Nodes.Add("Noi sinh: TP HCM");
    tv_DanhSachSV.Nodes.Add("Le thi B");
    tv_DanhSachSV.Nodes[1].Nodes.Add("Ngay sinh: 02/03/1989");
    tv_DanhSachSV.Nodes[1].Nodes.Add("Noi sinh: Ben tre");

    tv_DanhSachSV.Nodes.Add("Tran Van C");
}

private void btn_XoadSSV_Click(object sender, EventArgs e)
{
    //xoá danh sách
    tv_DanhSachSV.Nodes.Clear();
}
```



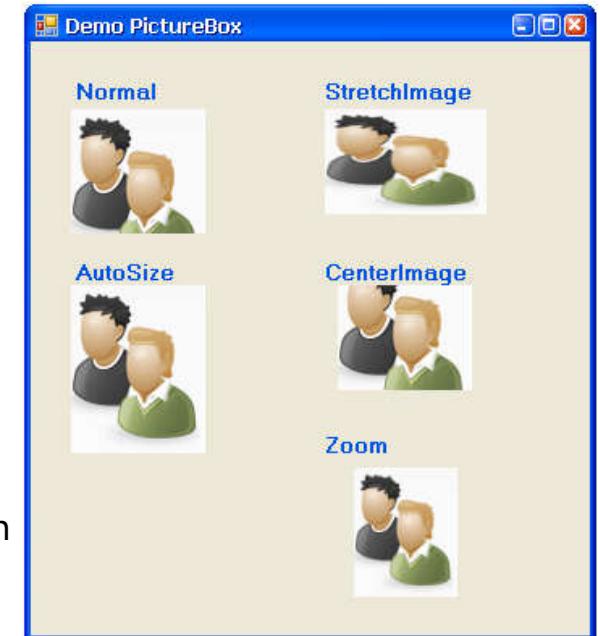
LHN



# PictureBox, ImageList

## ❑ PictureBox

- Sử dụng để hiển thị ảnh dạng bitmap, metafile, icon, JPEG, GIF.
- Sử dụng thuộc tính Image để thiết lập ảnh lúc design hoặc runtime
- **Các thuộc tính:**
  - ✓ **Image:** ảnh cần hiển thị
  - ✓ **SizeMode:** kiểu hiển thị hình ảnh
    - **Normal** (mặc định): hiển thị hình ảnh ở góc trên-trái của PictureBox
    - **StretchImage**: kích thước của hình ảnh = kích thước của PictureBox (kéo giãn hình ảnh vừa khung)
    - **AutoSize**: kích thước của PictureBox = kích thước của hình ảnh
    - **CenterImage**: hiển thị hình ảnh ở giữa PictureBox
    - **Zoom**: phóng to và hiển thị hình ảnh ở giữa PictureBox



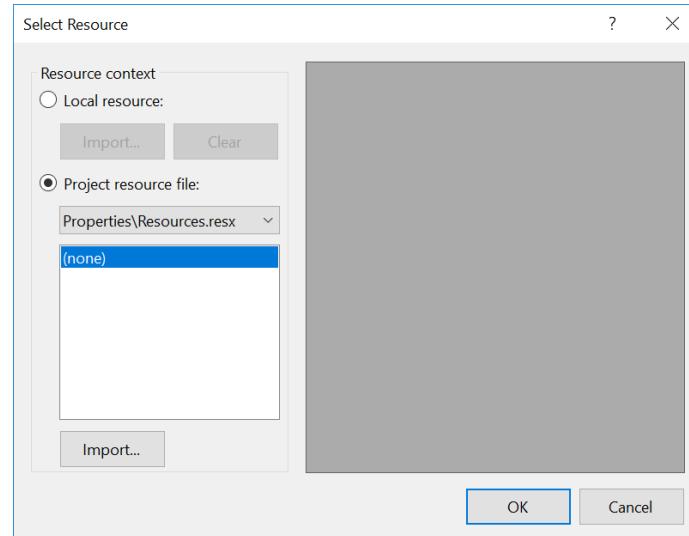
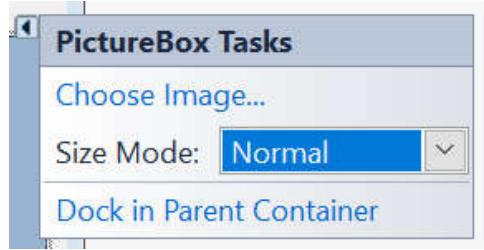


# PictureBox, ImageList

## ❑ PictureBox

➤ Hiển thị hình ảnh

✓ Design:



✓ Code:

```
pictureBox.Image = Image.FromFile(<đường_dẫn>);
```

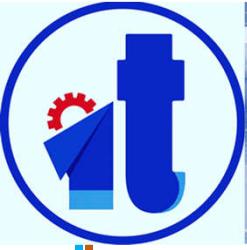
```
pictureBox.SizeMode = PictureBoxSizeMode.<giá_trị>
```



# PictureBox, ImageList

## ❑ ImageList

- Dùng để cung cấp tập hợp những đối tượng image cho các control khác sử dụng
  - ✓ ListView
  - ✓ TreeView
- **Các thuộc tính thường dùng**
  - ✓ *ColorDepth*: độ sâu của màu
  - ✓ *Images*: trả về ImageList.ImageCollection
  - ✓ *ImageSize*: kích thước ảnh
  - ✓ *TransparentColor*: xác định màu là transparent



# PictureBox, ImageList

## **ImageList**

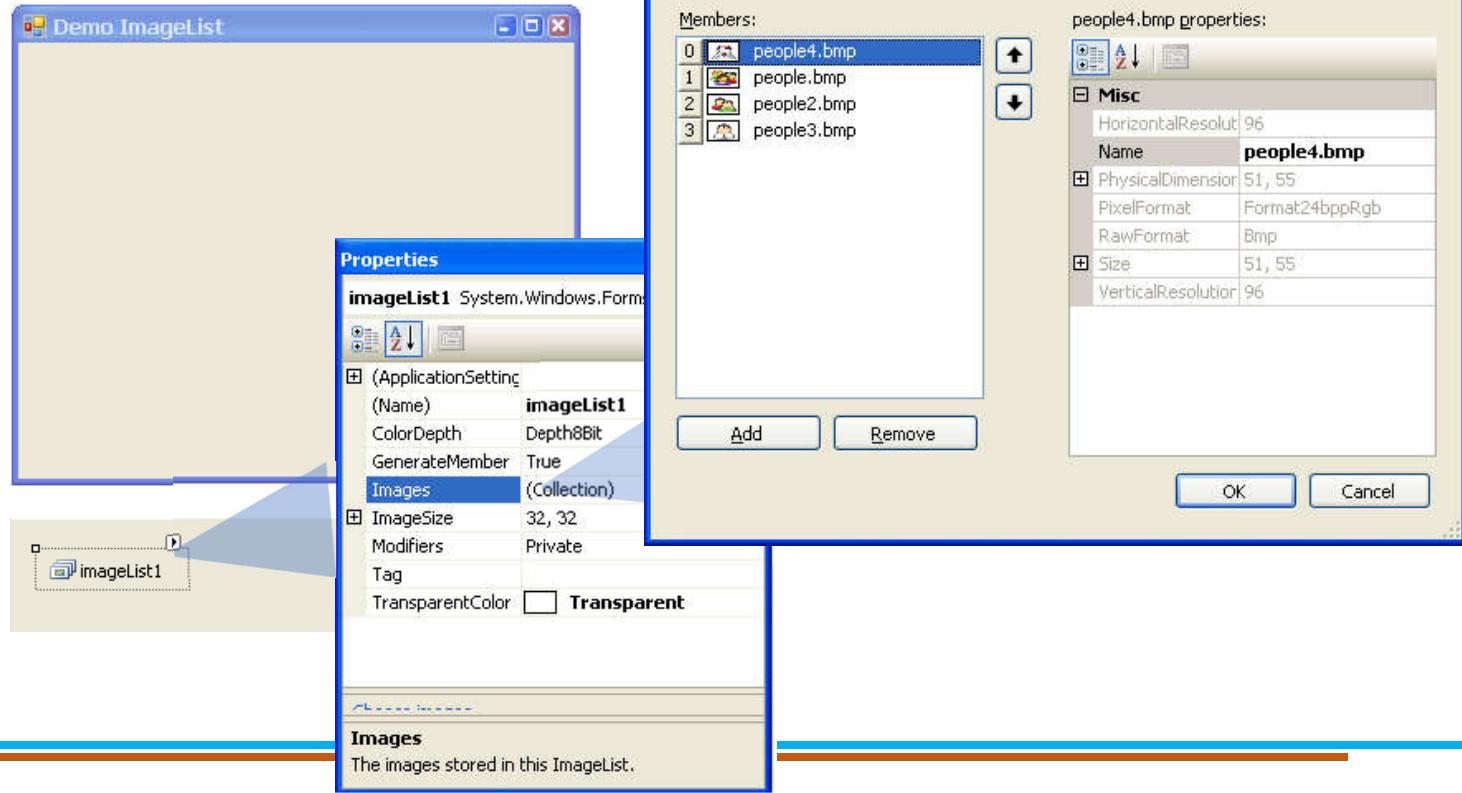
### ➤ Các bước sử dụng ImageList

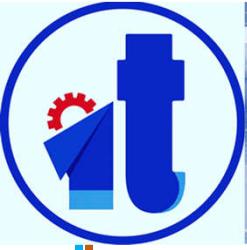
- ✓ Kéo control ImageList từ ToolBox thả vào Form
- ✓ Thiết lập kích thước của các ảnh: ImageSize
- ✓ Bổ sung ảnh vô ImageList qua thuộc tính Image
- ✓ Sử dụng ImageList cho các control
  - Khai báo nguồn là imagelist vừa tạo cho control
    - Thường là thuộc tính ImageList
  - Thiết lập các item/node với ImageIndex tương ứng
    - Việc thiết lập có thể ở màn hình design view hoặc code view



# PictureBox, ImageList

## ❑ ImageList

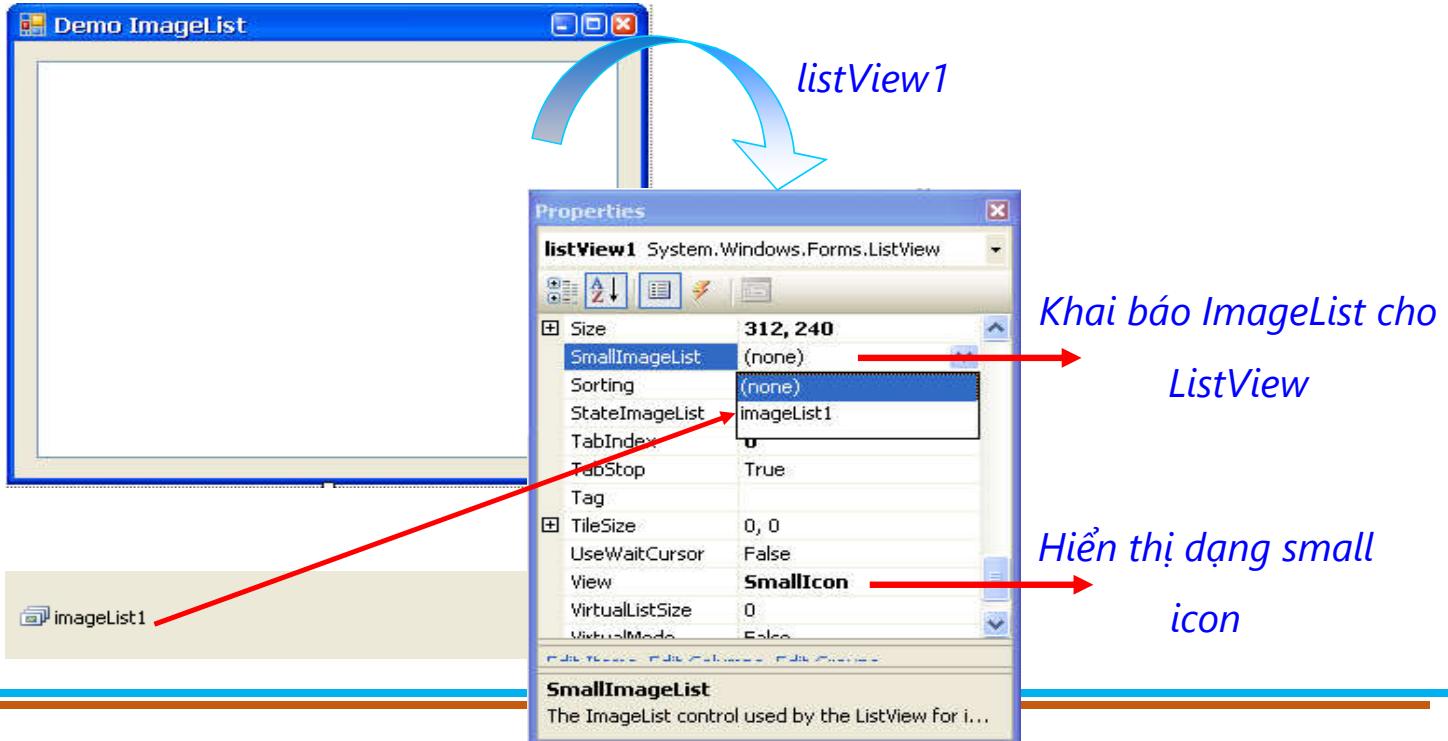




# PictureBox, ImageList

## ☐ ImageList

➤ Sử dụng ImageList trong ListView





# PictureBox, ImageList

## ☐ ImageList

➤ Sử dụng ImageList trong ListView → Thêm vào Item

The screenshot shows the Windows Forms Designer interface. On the left, there's a form titled "Demo ImageList" containing a "ListView1" control. The "Properties" window is open, showing the "listView1" properties. The "Items" property is selected, revealing a collection of four items: "Item 1", "Item 2", "Item 3", and "Item 4". The "ImageList" property is also visible, pointing to an "imageList1" object. In the center, the "ListViewItem Collection Editor" dialog is open, displaying the four items. The "Appearance" tab is selected, showing the "ImageIndex" property for each item. For "Item 1", "Item 2", and "Item 3", the "ImageIndex" is set to 0, 1, and 2 respectively. For "Item 4", the "ImageIndex" is set to 3. A blue arrow points from this dialog to the "ImageIndex" property in the "Properties" window. To the right, a preview window titled "Demo ImageList" shows the ListView control with four items. Each item has a small thumbnail image next to its text label: "Item 1" (blue), "Item 2" (green), "Item 3" (yellow), and "Item 4" (red). A blue arrow points from the preview window to the text below it.

Mỗi item sẽ có ảnh theo đúng thứ tự ImageIndex được khai báo trong ImageList

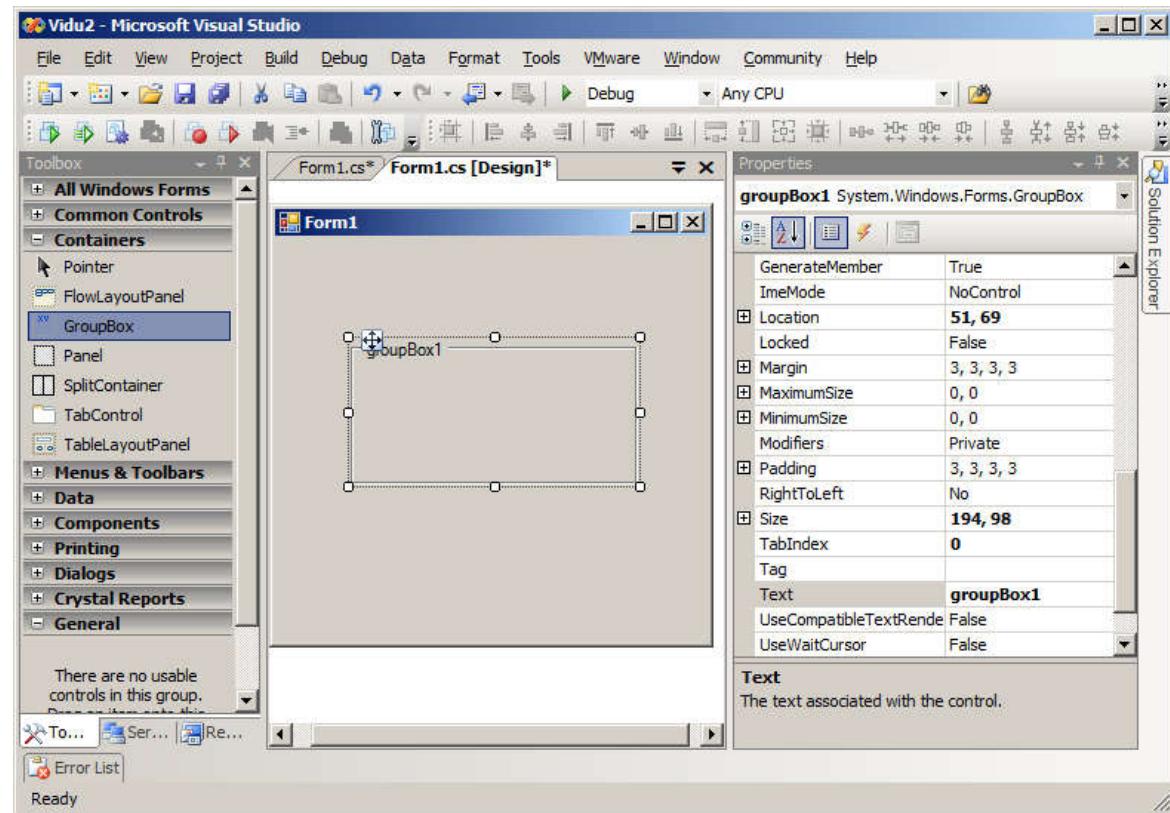
Khai báo image cho item qua ImageIndex

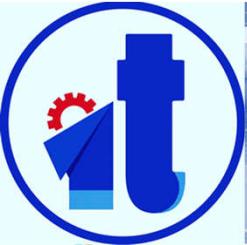


# GroupBox, Panel & TabControl

## □ **GroupBox**

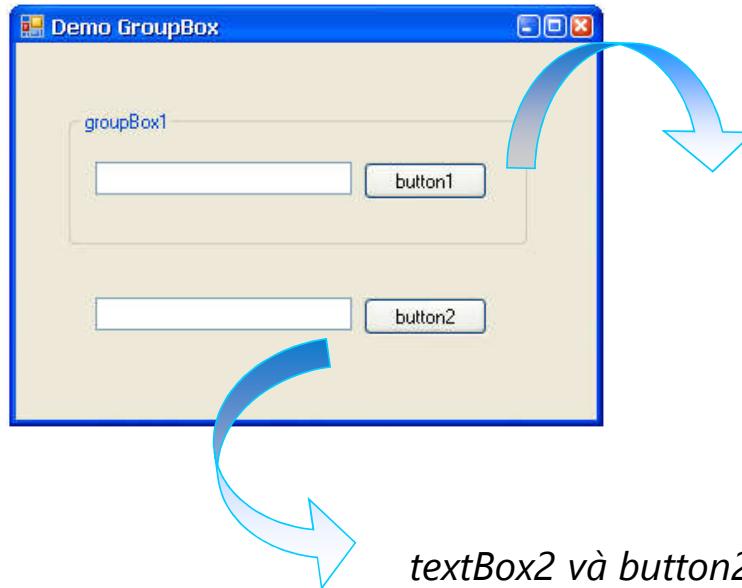
- Hiển thị một khung bao quanh một nhóm controls.
- Có thể hiển thị một tiêu đề.
- Khi xóa một GroupBox thì các control chứa trong nó cũng bị xóa theo.
- Lớp GroupBox kế thừa từ System.Windows.Forms.Control.





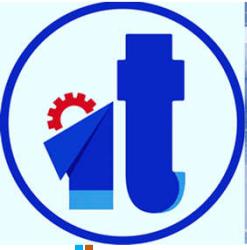
# GroupBox, Panel & TabControl

## **GroupBox**



groupBox1 chứa 2 control  
*textBox1 và button1*

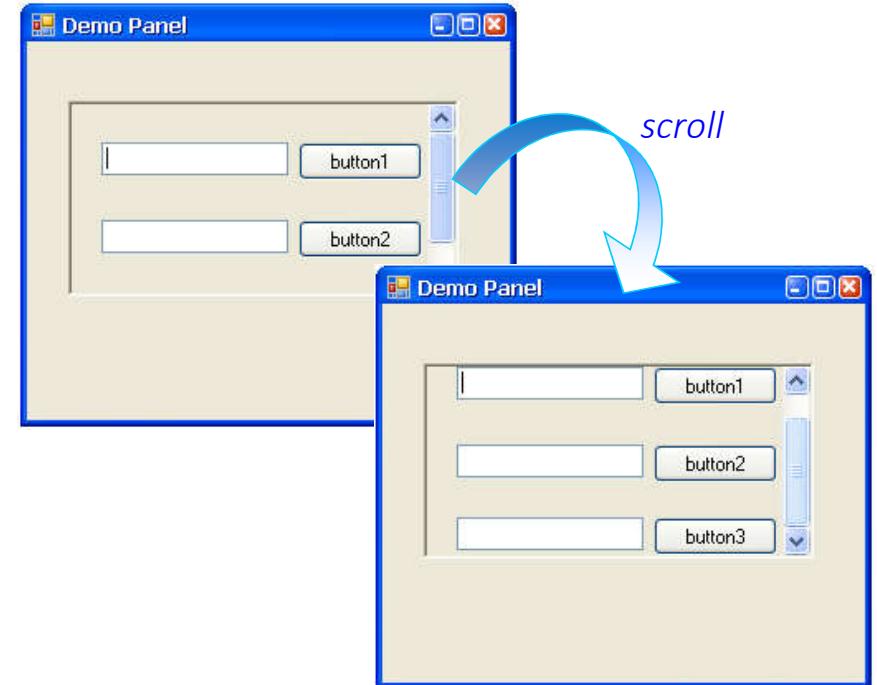
textBox2 và button2 chứa trong  
*Controls của Form*



# GroupBox, Panel & TabControl

## ☐ Panel

- Chứa nhóm các control
- Không có caption
- Có thanh cuộn (scrollbar): Hiển thị được nhiều control khi kích thước panel giới hạn





# GroupBox, Panel & TabControl

## Panel

GroupBox	Mô tả
<i>Thuộc tính thường dùng</i>	
<b>Controls</b>	Danh sách control chứa trong GroupBox.
<b>Text</b>	Caption của GroupBox

Panel	
<i>Thuộc tính thường dùng</i>	
<b>AutoScroll</b>	Xuất hiện khi panel quá nhỏ để hiển thị hết các control, mặc định là false
<b>BorderStyle</b>	Biên của panel, mặc định là None, các tham số khác như Fixed3D, FixedSingle
<b>Controls</b>	Danh sách control chứa trong panel



# GroupBox, Panel & TabControl

## □ TabControl

- Dạng container chứa các control khác
- Cho phép thể hiện nhiều page trên một form
  - ✓ Mỗi page chứa các control tương tự như group control khác.
  - ✓ Mỗi page có tag chứa tên của page
  - ✓ Kích vào các tag để chuyển page.
- Ý nghĩa:
  - ✓ Cho phép thể hiện nhiều control trên một form
  - ✓ Các control có cùng nhóm chức năng sẽ được tổ chức trong một tab (page)



# GroupBox, Panel & TabControl

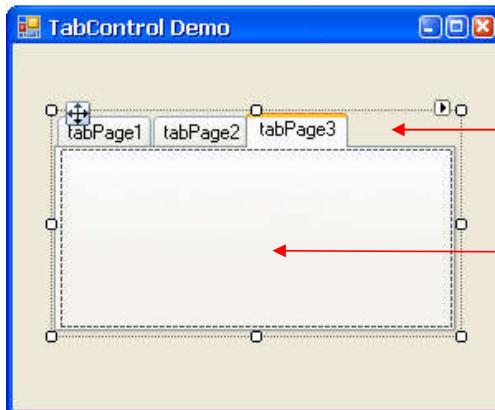
## ☐ TabControl

- Các thuộc tính
  - ✓ TabPages
  - ✓ Appearance

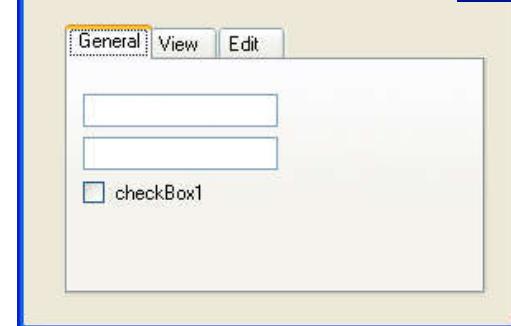
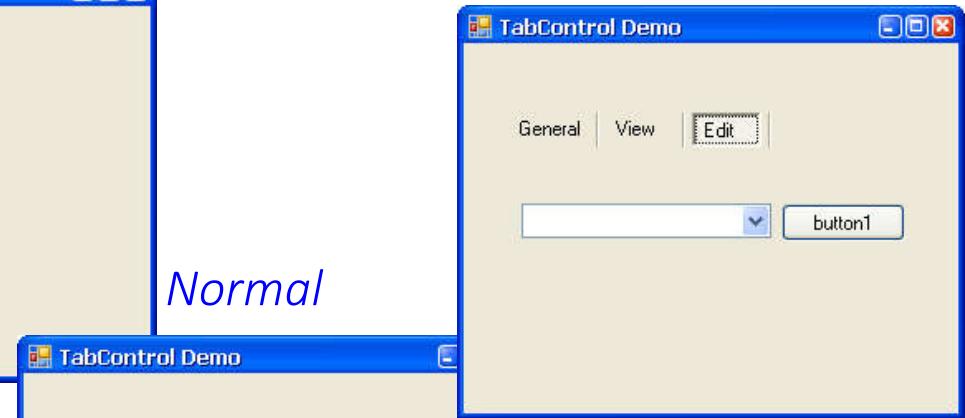
Buttons

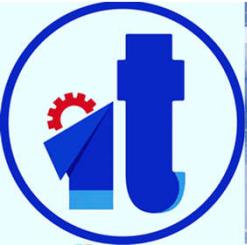


Normal



FlatButton





# GroupBox, Panel & TabControl

## ❑ TabControl

- Thuộc tính, phương thức, sự kiện thường dùng

### *Properties*

TabPage

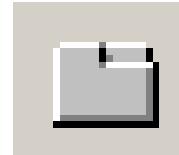
TabCount

SelectedTab

Multiline

SelectedIndex

### TabControl



### *Method*

SelectTab

DeselectTab

### *Event*

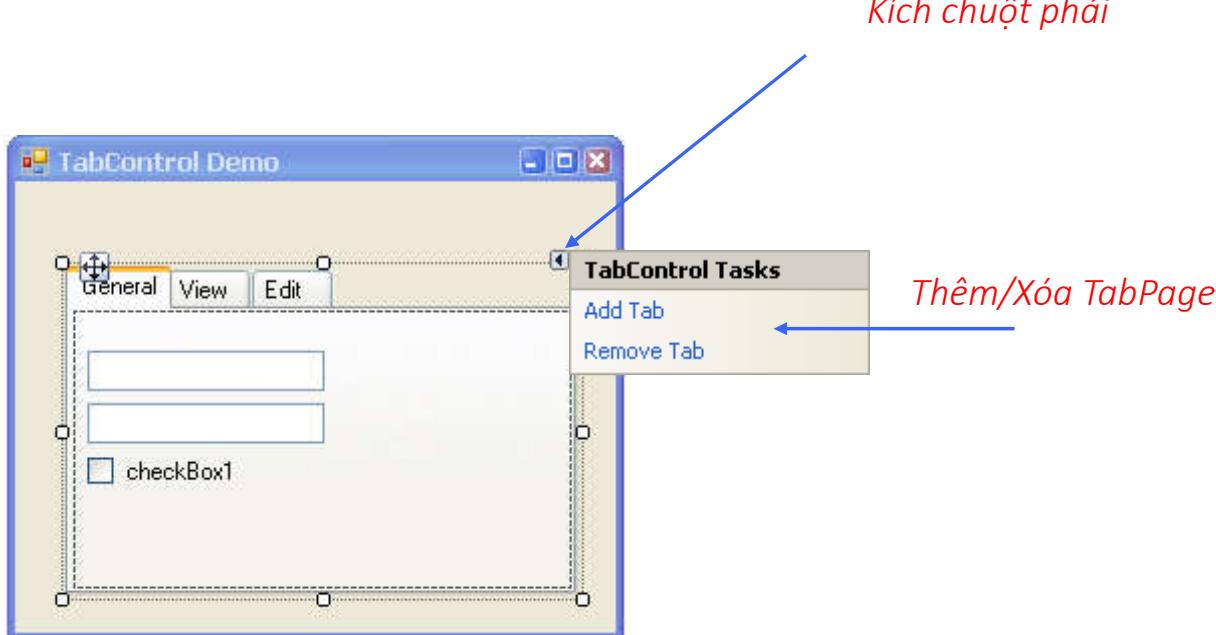
SelectedIndexChanged



# GroupBox, Panel & TabControl

## ❑ TabControl

- Thêm và xóa tabpage bằng Design

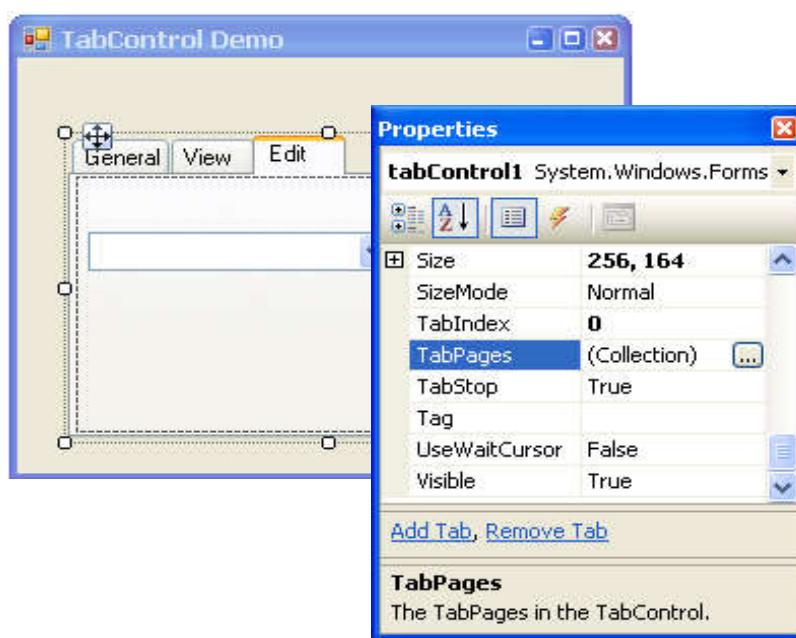


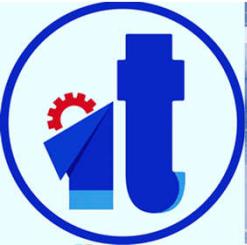


# GroupBox, Panel & TabControl

## ☐ TabControl

- Chính sửa các tabpages

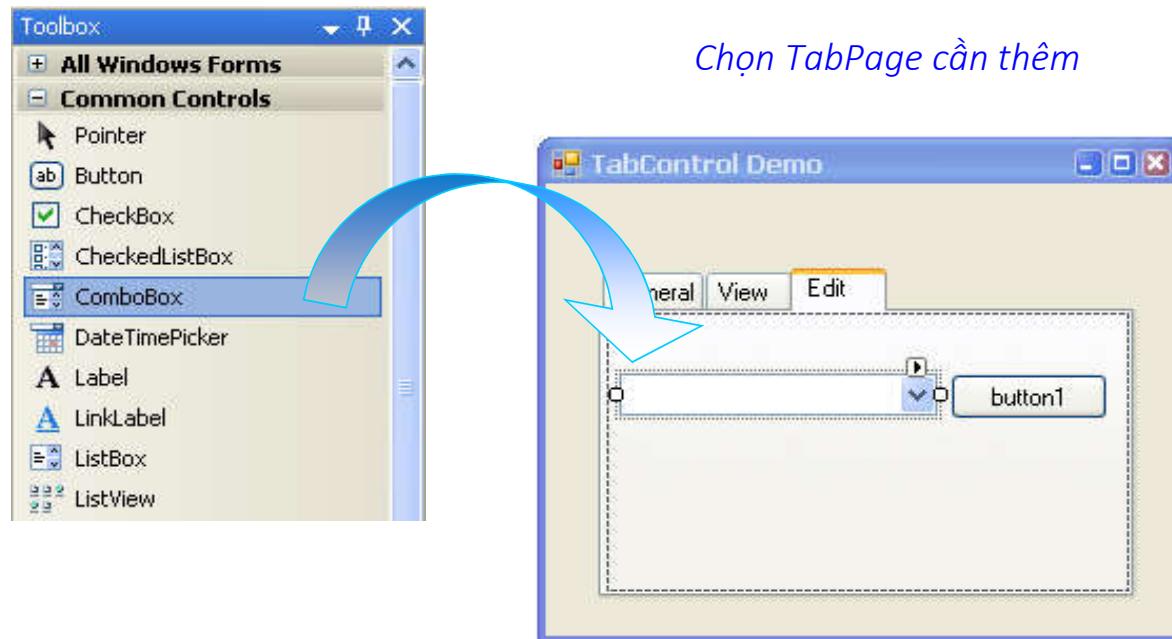


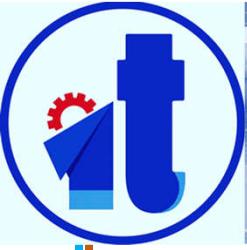


# GroupBox, Panel & TabControl

## ❑ TabControl

- Thêm control vào tabpage





LVHN



# Bài tập

**Hãy viết chương trình ứng dụng thực hiện các chức năng theo giao diện được thiết kế như sau**

- Mặc định giới tính là nam, dân tộc kinh.
- Tại checklistbox chứa danh sách kỹ năng lập trình. Người dùng sẽ chọn những kỹ năng của mình. Nếu có kỹ năng khác, người dùng sẽ nhập textbox kỹ năng mới đó và bấm thêm để bổ sung vào danh sách(kỹ năng mới thêm mặc định được checked)

Lý lịch sinh viên

Nhập thông tin cá nhân Thông tin lý lịch

Họ và tên: \_\_\_\_\_

Ngày sinh: \_\_\_\_\_

Giới tính:  Nam  Nữ

Dân tộc: \_\_\_\_\_

Tình trạng hôn nhân:  
 Độc thân  Kết hôn  Ly hôn

Ngoại ngữ:  Anh  Pháp  Hoa

Kỹ năng lập trình:

\_\_\_\_\_

C/C++  
 C#



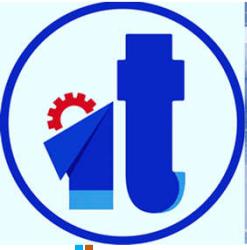
# Bài tập

- ☐ Khi người dùng bấm nút xem thông tin, tabpage [Thông tin lý lịch] hiển thị những thông tin vừa nhập

Lý lịch sinh viên

Nhập thông tin cá nhân  Thông tin lý lịch

Họ và tên:	Nguyễn Minh Anh
Ngày sinh:	16/08/1996
Giới tính:	Nam
Dân tộc:	Kinh
Tình trạng hôn nhân:	Độc thân
Ngoại ngữ:	Anh
Kỹ năng	<input type="text"/>



Q&A



LVHN