

PHẦN 4: WINFORM

- Graphical User Interface (GUI);
- Event Driven Programming;
- Ứng dụng Windows Form dùng C#;
- Khuôn mẫu của ứng dụng Windows Form chuẩn.

GUI

```
--- rr.cktpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 112.076/112.076/112.076/0.000 ms
bash-2.05b$ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sdal          /mnt/usbkey
/dev/sda2          /mnt/ipod
bash-2.05b$ date
Wed May 25 11:36:56 PDT 2005
bash-2.05b$ lsmod
Module            Size  Used by
joydev           8256   0
ipw2200        175112   0
ieee80211       44228   1 ipw2200
ieee80211_crypt  4872   2 ipw2200,ieee80211
e1000          84468   0
bash-2.05b$
```

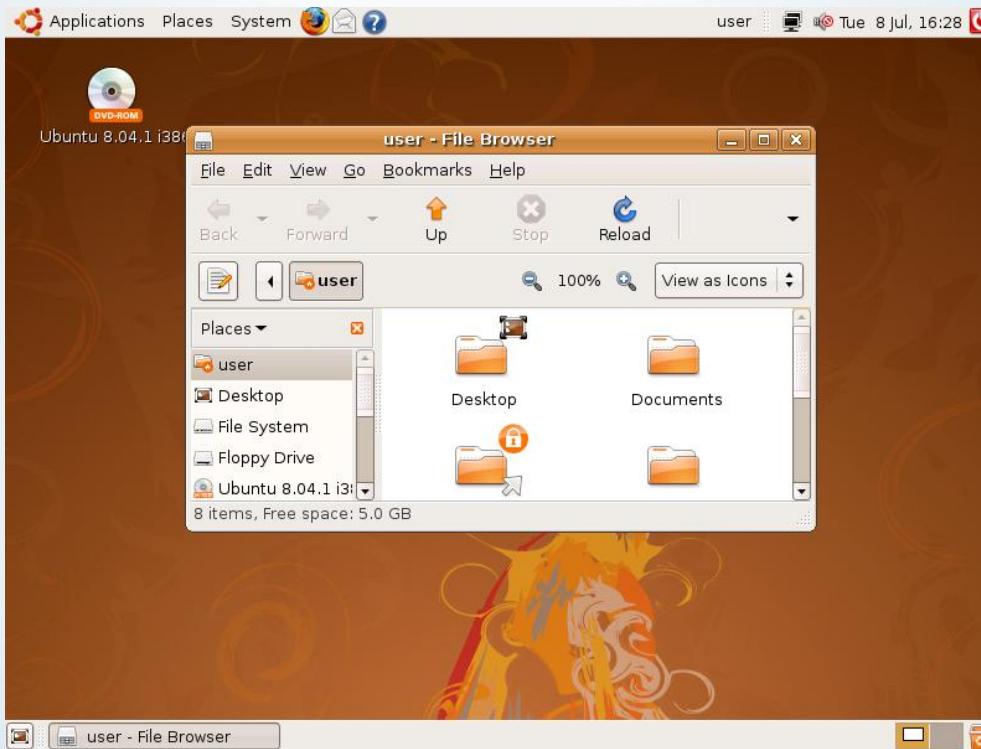


Tương tác qua keyboard
Thực thi tuần tự

GUI dựa trên text
Mức độ tương tác cao hơn

GUI

- Tương tác qua giao diện đồ họa độ phân giải cao;
- Đa số các hệ OS hiện đại đều dùng GUI;
- Cho phép user dễ dàng thao tác.



GUIs

- Chương trình hiện đại đều dùng GUI;
- Graphical: text, window, menu, button...;
- User: người sử dụng chương trình;
- Interface: cách tương tác chương trình;
- Thành phần đồ họa điển hình:
 - Window: một vùng bên trong màn hình chính;
 - Menu: liệt kê những chức năng;
 - Button: nút lệnh cho phép click vào;
 - TextBox: cho phép user nhập dữ liệu text.

GUI APPLICATION

- Windows Form là nền tảng GUI cho ứng dụng desktop
 - (Ngược với Web Form ứng dụng cho Web);
 - Single Document Interface (SDI);
 - Multiple Document Interface (MDI);
- Các namespace chứa các lớp hỗ trợ GUI trong .NET:
 - System.Windows.Forms:
 - ✓ Chứa GUI components/controls và form.
 - System.Drawing:
 - ✓ Chức năng liên quan đến tô vẽ cho thành phần GUI;
 - ✓ Cung cấp chức năng truy cập đến GDI+ cơ bản.

EVENT-DRIVEN PROGRAMMING

Cách truyền thống

Danh sách các lệnh thực thi
tuần tự

Việc kế tiếp xảy ra chính là
lệnh tiếp theo trong danh sách

Chương trình được thực thi
bởi máy tính

Event-Driven Programming

Các đối tượng có thể kích hoạt
sự kiện và các đối tượng khác
phản ứng với những sự kiện
đó

Việc kế tiếp xảy ra phụ thuộc
vào sự kiện kế tiếp

Luồng chương trình được điều
kiển bởi sự tương tác User-
Computer

EVENT-DRIVEN PROGRAMMING

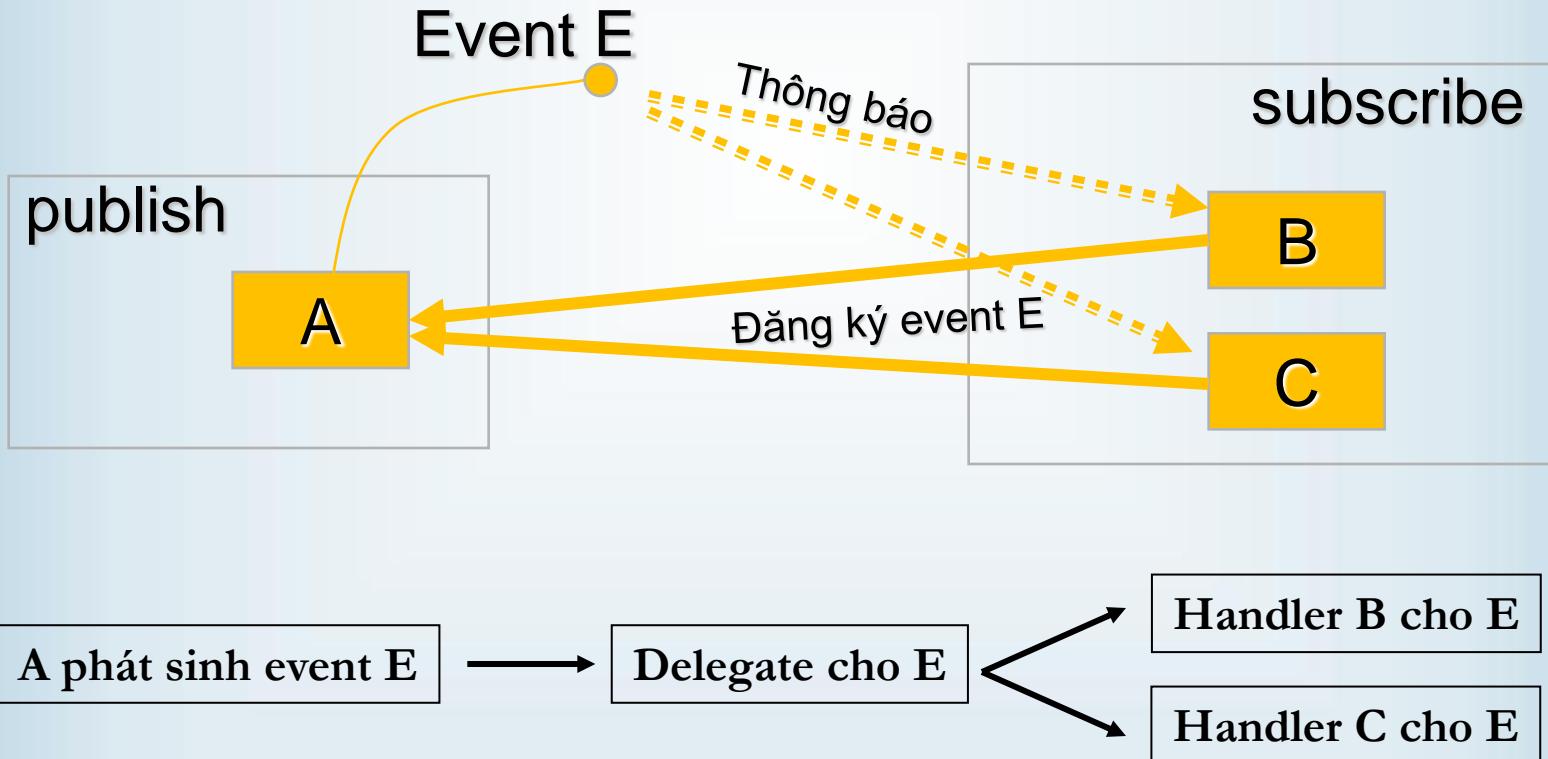
- Chương trình GUI thường dùng Event-Driven Programming;
- Chương trình chờ cho event xuất hiện và xử lý;
- Ví dụ sự kiện:



- Firing an event: khi đối tượng khởi tạo sự kiện;
- Listener: đối tượng chờ cho sự kiện xuất hiện;
- Event handler: phương thức phản ứng lại sự kiện.

EVENT-DRIVEN PROGRAMMING

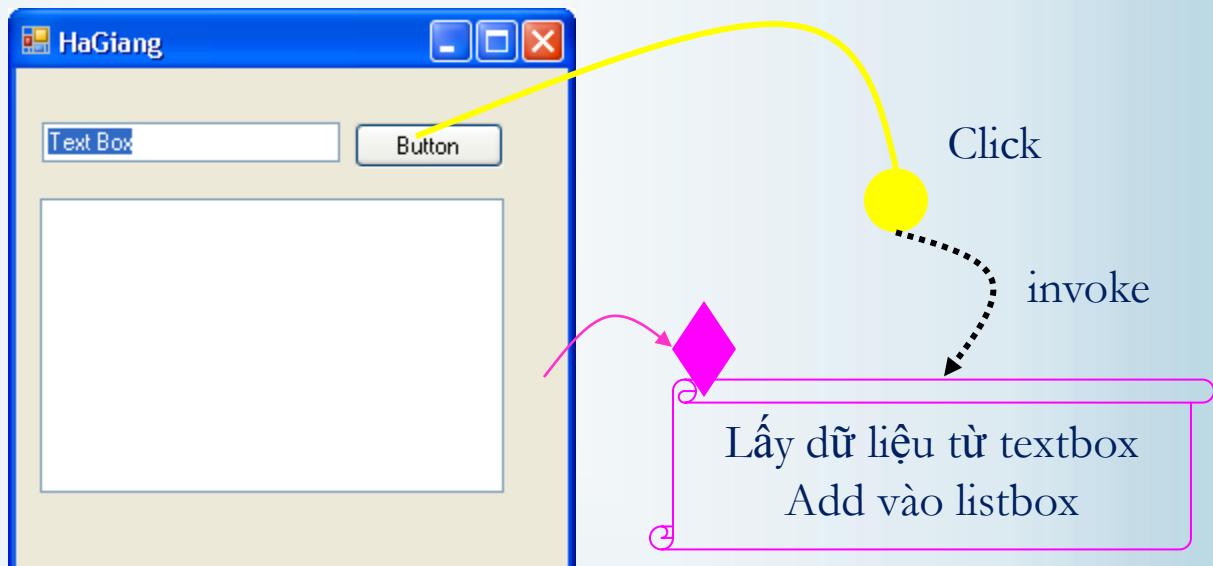
- Trong C#, Event-Driven Programming được thực thi bởi event (xem slide Delegate & Event):



EVENT-DRIVEN PROGRAMMING

- Minh họa xử lý trong form:

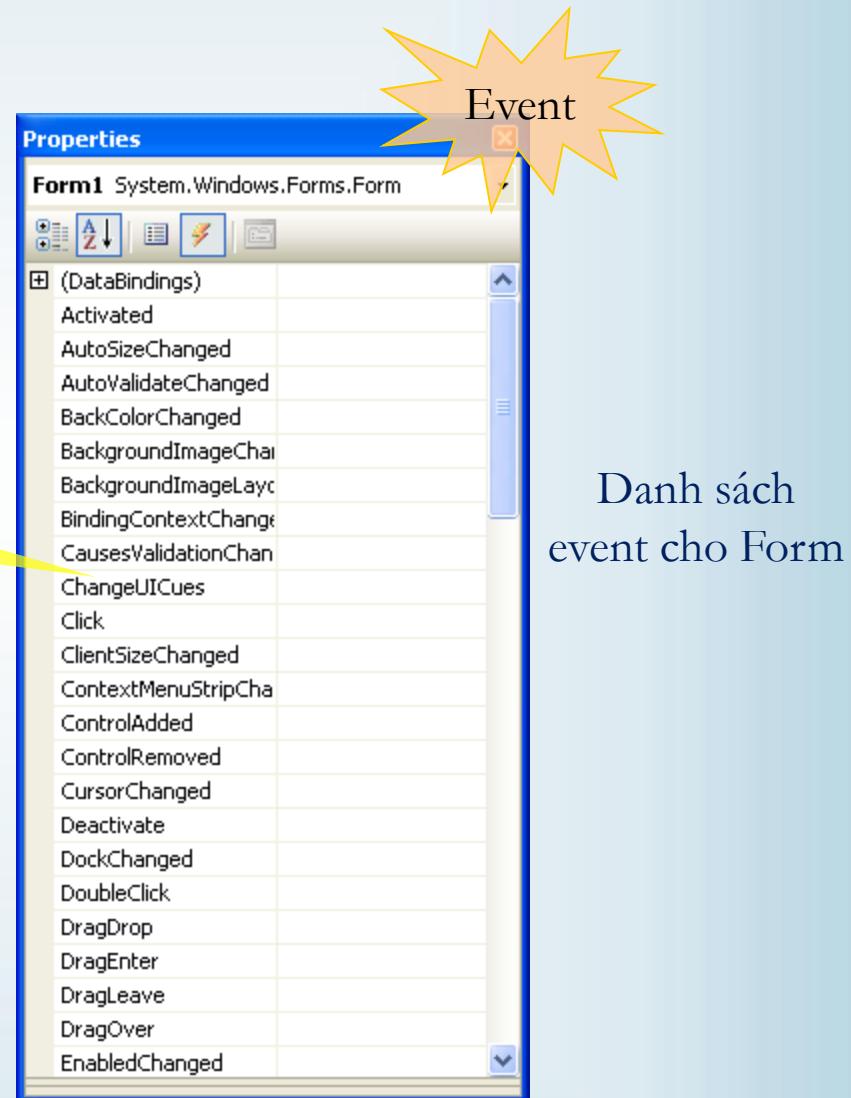
User nhập text vào textbox → click Button để add chuỗi nhập vào listbox



Button đưa ra sự kiện click
Form có event handler cho click của button

EVENT-DRIVEN PROGRAMMING

- GUI-based events:
 - Mouse move
 - Mouse click
 - Mouse double-click
 - Key press
 - Button click
 - Menu selection
 - Change in focus
 - Window activation
 - ...



WINDOWS FORMS APPLICATION

- Sử dụng GUI làm nền tảng;
- Event-driven programming cho các đối tượng trên form;
- Ứng dụng dựa trên một “form” chứa các thành phần:
 - Menu;
 - Toolbar;
 - StatusBar;
 - TextBox, Label, Button...
- Lớp cơ sở cho các form của ứng dụng là Form.

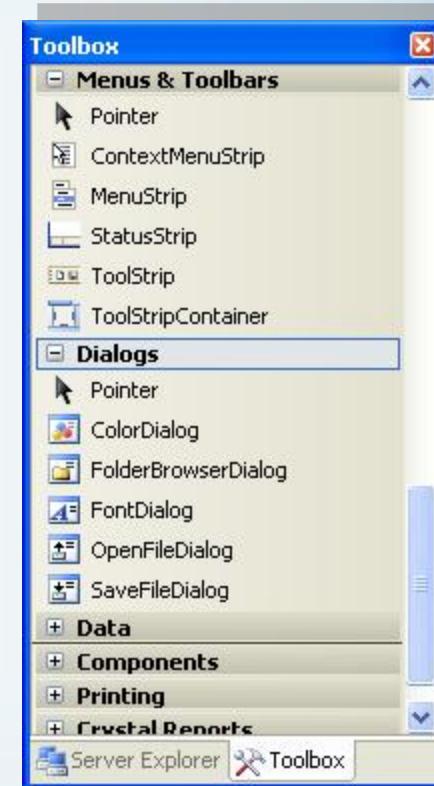
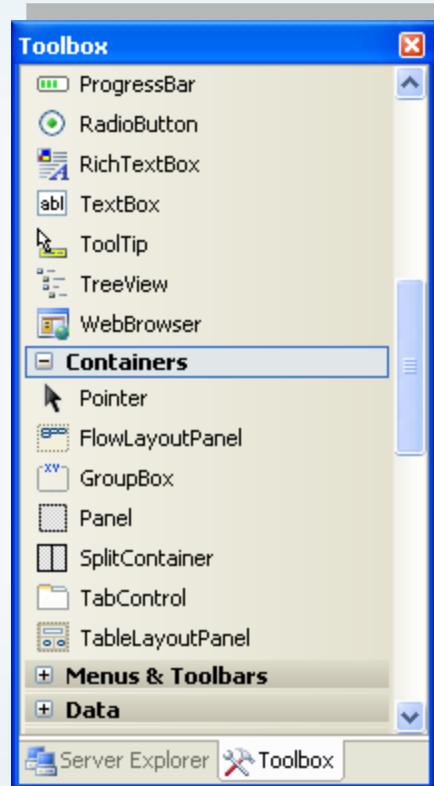
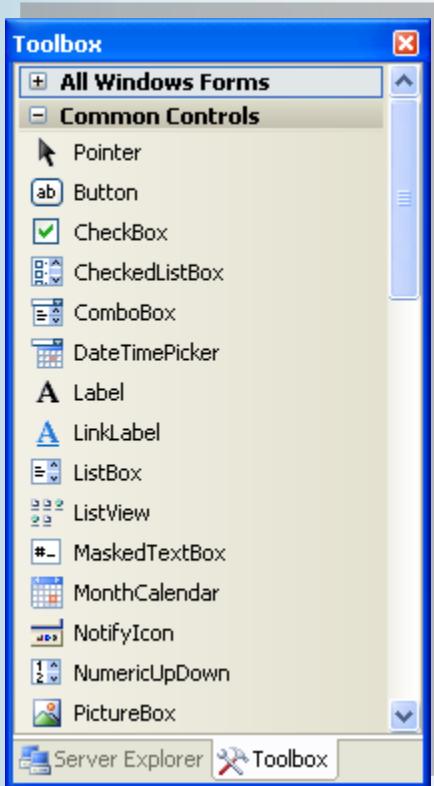
System.Windows.Forms.

Namespace

Form

Class

COMPONENTS & CONTROLS CHO WINDOWS FORM



ỨNG DỤNG WINFORM ĐƠN GIẢN

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace HaGiang
{
    public class Form1 : Form
    {
        Label title;
        public Form1()
        {
            this.Text = "Hello World!";
            this.Size = new Size(400, 200);
            title = new Label();
            title.Text = "Hello World";
            title.Location = new Point(50, 50);
            this.Controls.Add(title);
        }

        public static void Main(string[] argv)
        {
            Application.Run(new Form1());
        }
    }
}
```

Form1.cs

Lớp Form cơ sở

Control kiểu Label

Thiết kế form & control

Add control vào form

Chạy ứng dụng với Form1
làm form chính

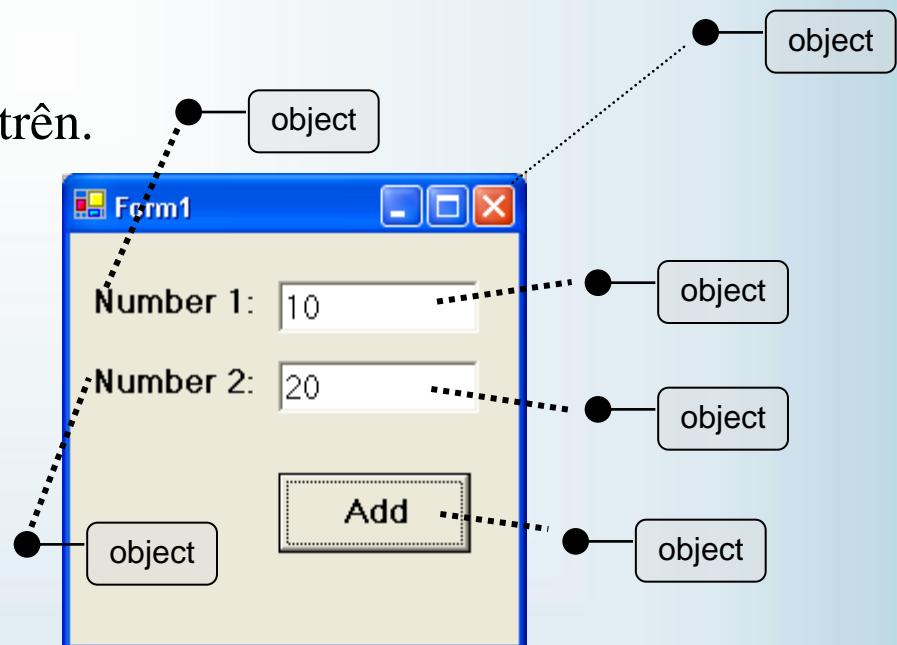
CÁC BƯỚC TẠO UD WINFORM CƠ BẢN

- Tạo lớp kế thừa từ lớp Form cơ sở;
- Bổ sung các control vào form:
 - Thêm các label, menu, button, textbox...
- Thiết kế layout cho form (bố trí control):
 - Hiệu chỉnh kích thước, trình bày, giao diện cho:
 - ✓ Form;
 - ✓ Control chứa trong form.
- Viết các xử lý cho các control trên form và các xử lý khác;
- Hiển thị Form:
 - Thông qua lớp Application gọi phương thức Run.

Nên sử dụng IDE hỗ trợ thiết kế GUI!

FORM VÀ CONTROL

- Tất cả các thành phần trên form đều là đối tượng;
- Các control là những lớp của FCL:
 - System.Windows.Forms.Label;
 - System.Windows.Forms.TextBox;
 - System.Windows.Forms.Button;
 - ...
- Các control là instance của các lớp trên.



CÁC THUỘC TÍNH CỦA FORM

Property	Description	Default
Name	Tên của form sử dụng trong project	Form1, Form2...
AcceptButton	Thiết lập button là click khi user nhấn Enter	
CancelButton	Thiết lập button là click khi user nhấn Esc	
ControlBox	Hiển thị control box trong caption bar	True
FormBorderStyle	Biên của form: none, single, 3D, sizable	Sizable
StartPosition	Xác định vị trí xuất hiện của form trên màn hình	WindowsDefaultLocation
Text	Nội dung hiển thị trên title bar	Form1, Form2, Form3
Font	Font cho form và mặc định cho các control	
Method	Description	
Close	Đóng form và free resource	
Hide	Ẩn form	
Show	Hiển thị form đang ẩn	
Event	Description	
Load	Xuất hiện trước khi form show	

TỔNG QUAN CONTROL

- Control là một thành phần cơ bản trên form;
- Có các thành phần:
 - Thuộc tính;
 - Phương thức;
 - Sự kiện.
- Tất cả các control chứa trong namespace:
System.Windows.Forms

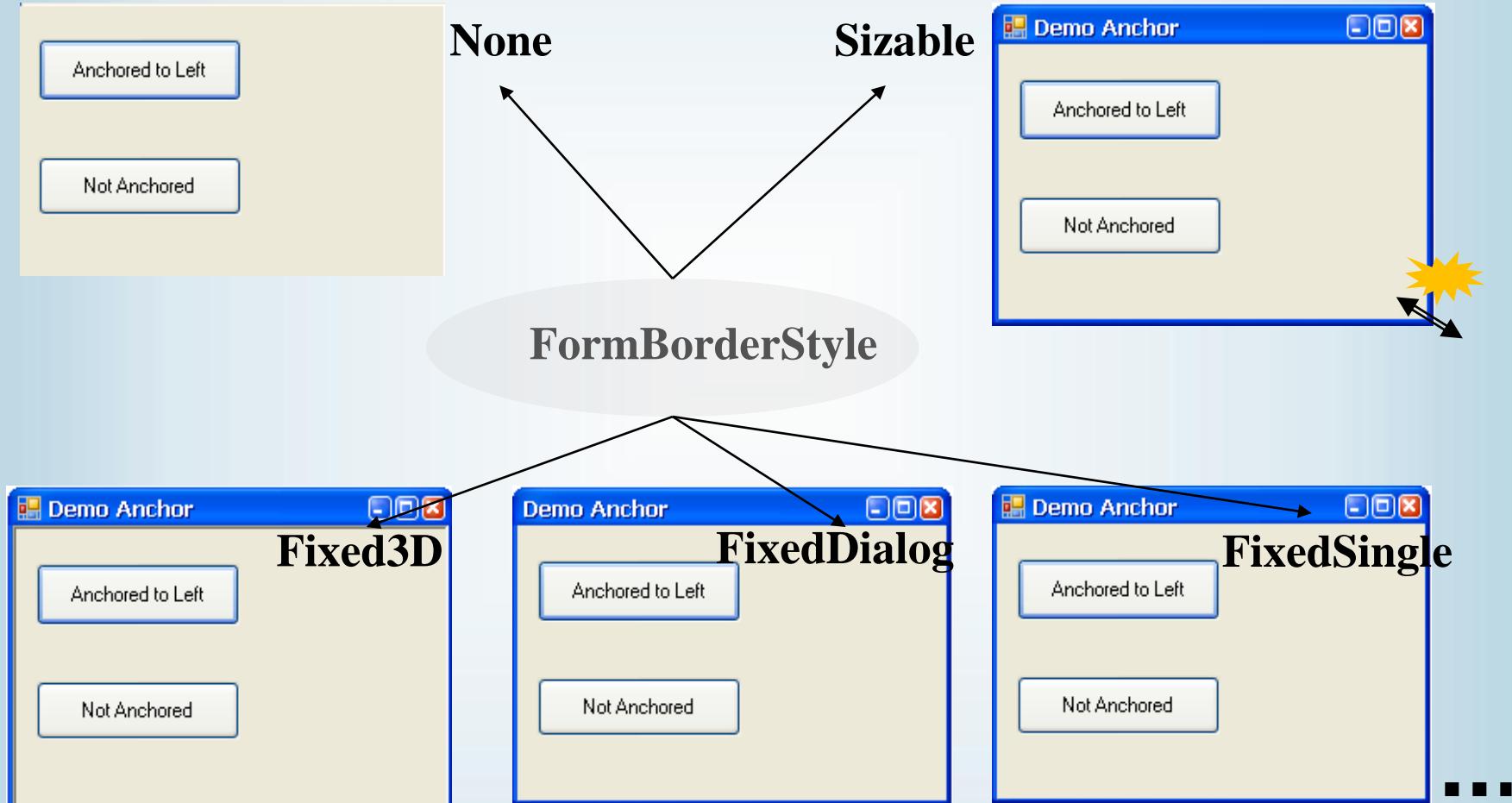
TỔNG QUAN CONTROL

- Một số thuộc tính của control:
 - Text: mô tả text xuất hiện trên control;
 - Focus: phương thức chuyển focus vào control;
 - TabIndex: thứ tự của control nhận focus:
 - ✓ Mặc định được VS.NET thiết lập.
 - Enable: thiết lập trạng thái truy cập của control;
 - Visible: ẩn control trên form, có thể dùng phương thức Hide;
 - Anchor:
 - ✓ Neo giữ control ở vị trí xác định;
 - ✓ Cho phép control di chuyển theo vị trí.
 - Size: xác nhận kích thước của control.

THUỘC TÍNH CONTROL

Common Properties	Description
BackColor	Màu nền của control
BackgroundImage	Ảnh nền của control
ForeColor	Màu hiển thị text trên form
Enabled	Xác định khi control trạng thái enable
Focused	Xác định khi control nhận focus
Font	Font hiển thị text trên control
TabIndex	Thứ tự tab của control
TabStop	Nếu true, user có thể sử dụng tab để select control
Text	Text hiển thị trên form
TextAlign	Canh lè text trên control
Visible	Xác định hiển thị control

CONTROL LAYOUT - ANCHOR

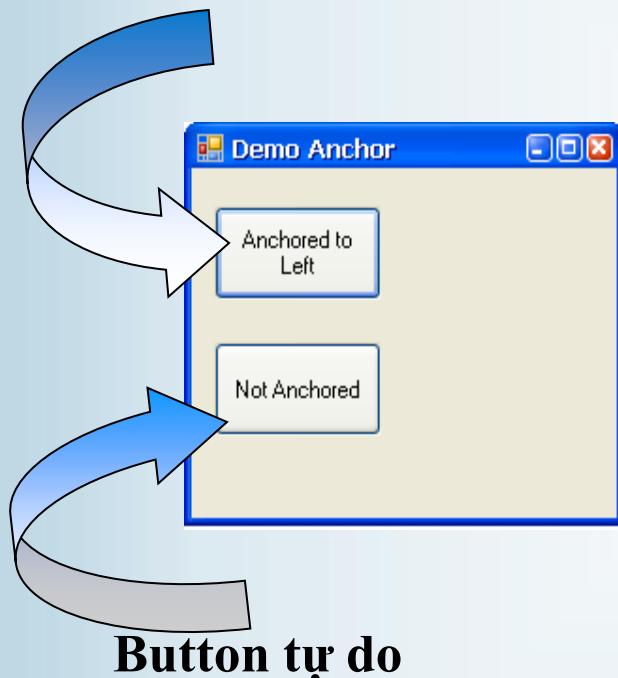


CONTROL LAYOUT - ANCHOR

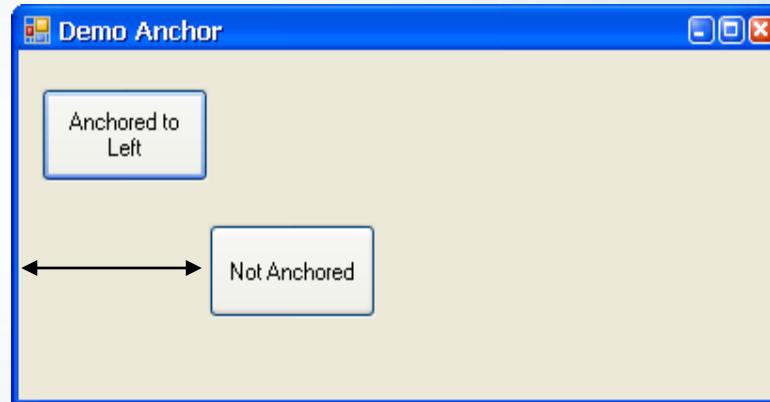
- Khi FormBorderStyle = Sizable, form cho phép thay đổi kích thước khi Runtime;
 - Sự bố trí của control cũng thay đổi!
- Sử dụng thuộc tính Anchor:
 - Cho phép control phản ứng lại với thao tác resize của form:
 - ✓ Control có thể thay đổi vị trí tương ứng với việc resize của form;
 - ✓ Control cố định không thay đổi theo việc resize của form.
 - Các trạng thái neo:
 - ✓ Left: cố định theo biên trái;
 - ✓ Right: cố định theo biên phải;
 - ✓ Top: cố định theo biên trên;
 - ✓ Bottom: cố định theo biên dưới.

CONTROL LAYOUT - ANCHOR

Button được neo biên trái



Vị trí tương đối với biên trái không đổi

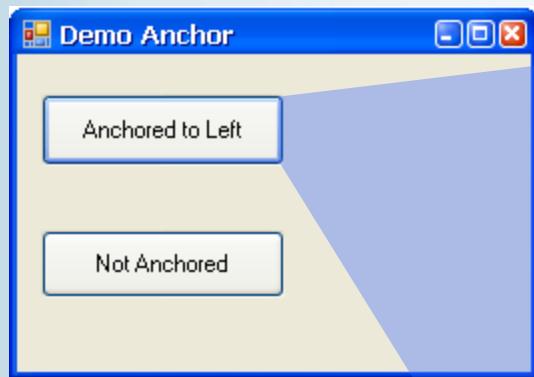


Button tự do

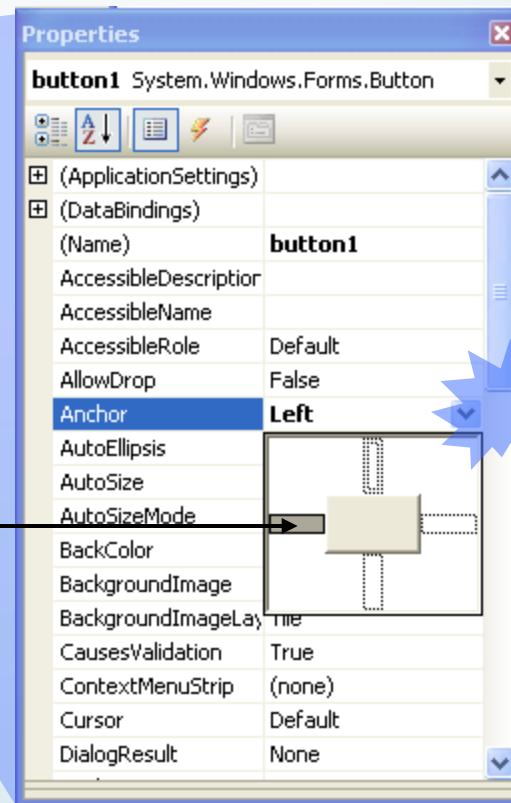
Di chuyển tương ứng theo kích thước mới

CONTROL LAYOUT - ANCHOR

- Thiết lập Anchor cho control:



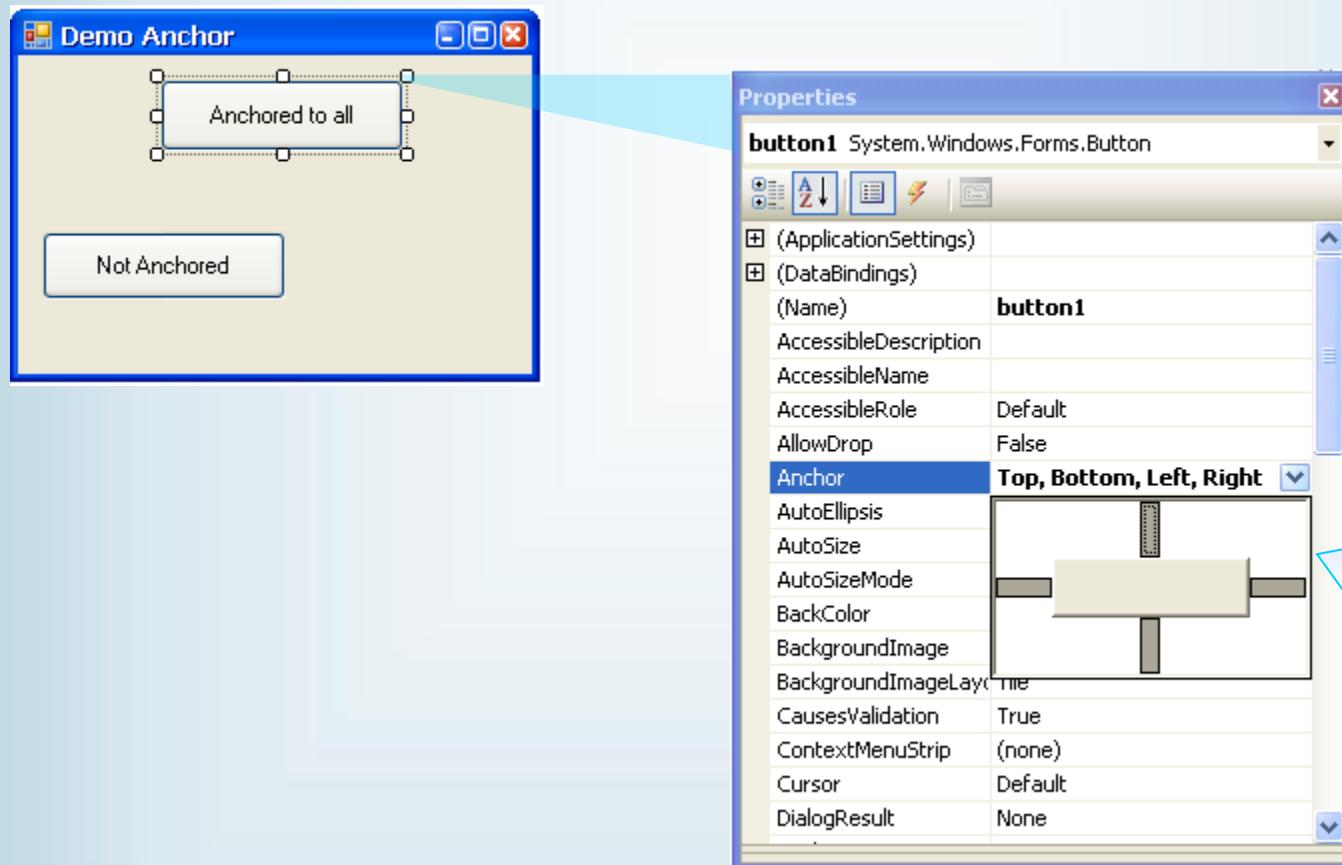
Biên được
chọn neo,
màu đậm



Chọn các
biên để neo

CONTROL LAYOUT - ANCHOR

- Thiết lập Anchor cho control:

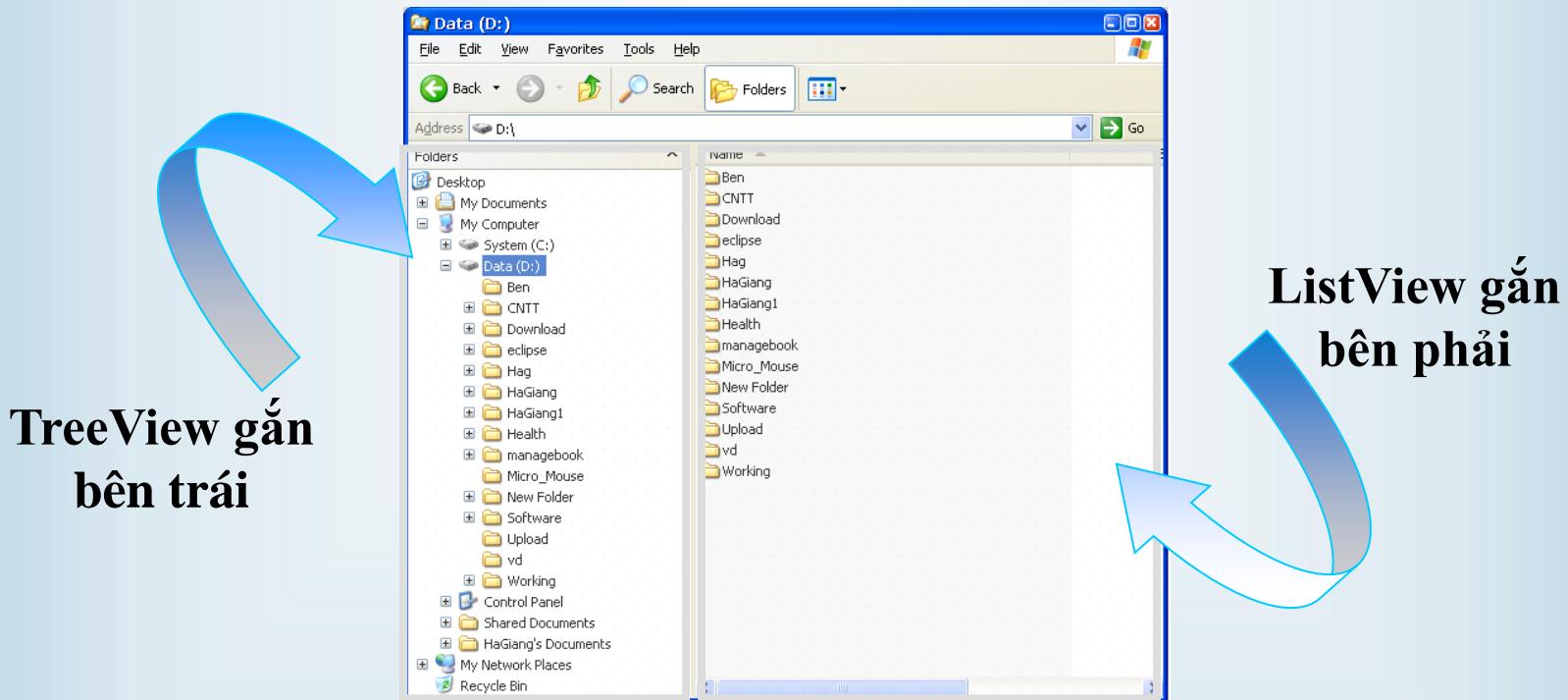


Neo theo
bốn phía

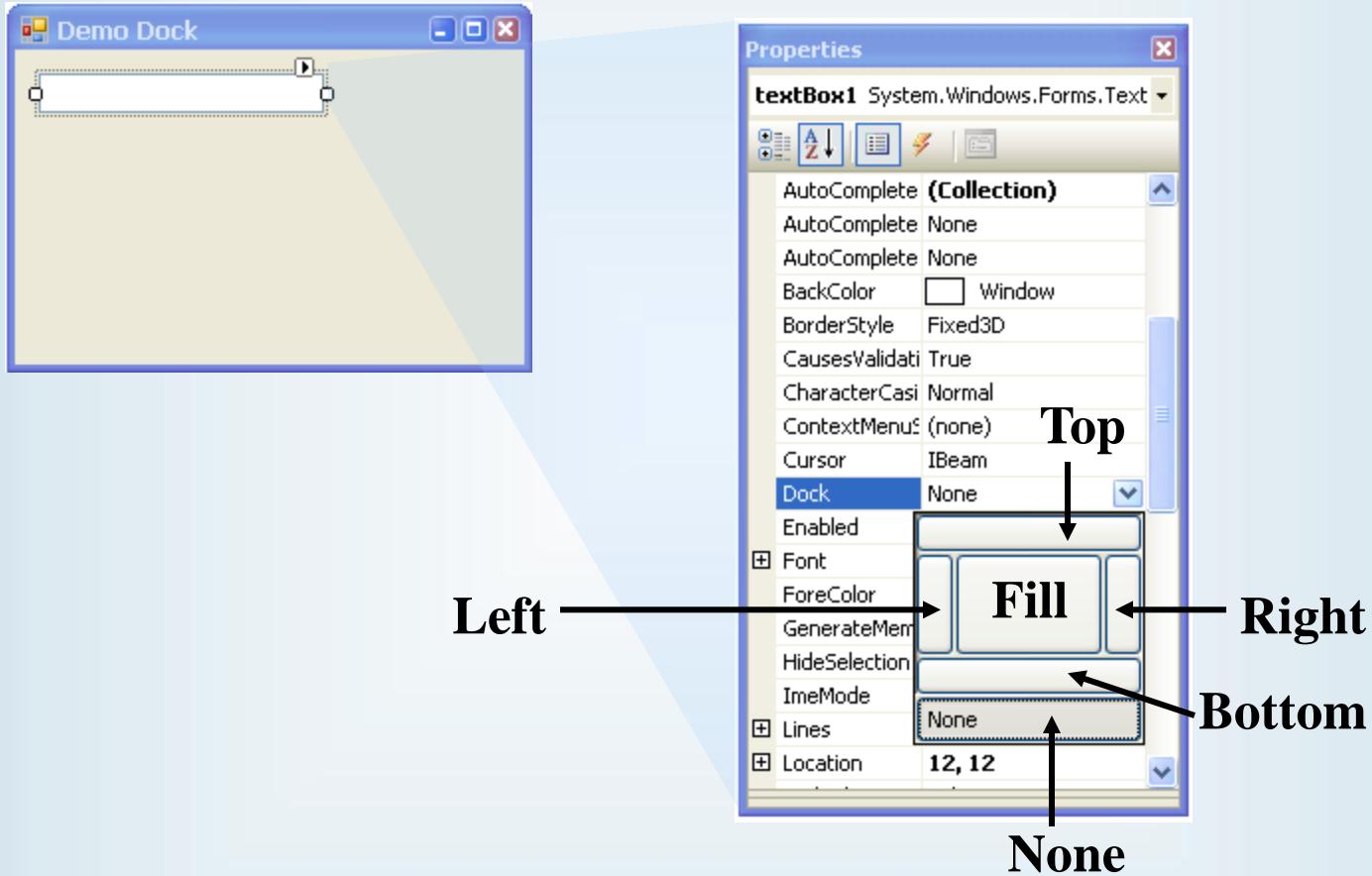
CONTROL LAYOUT - DOCKING

- Các control có thể gắn (dock) với một cạnh nào đó của form, hoặc container của control:

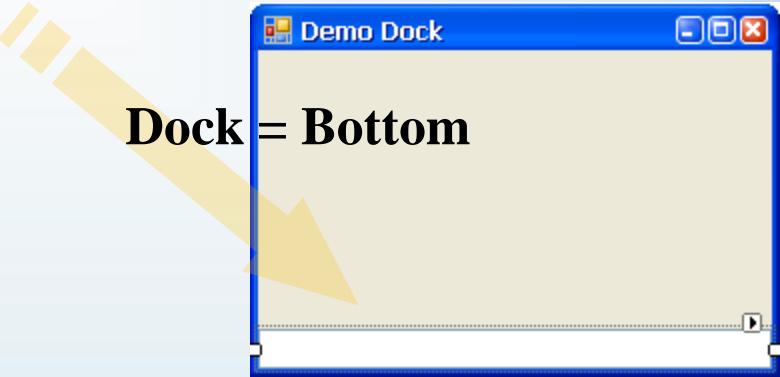
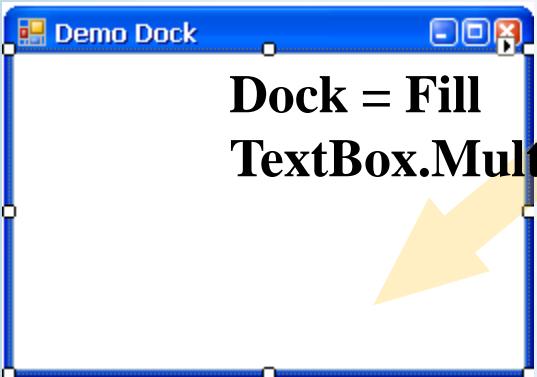
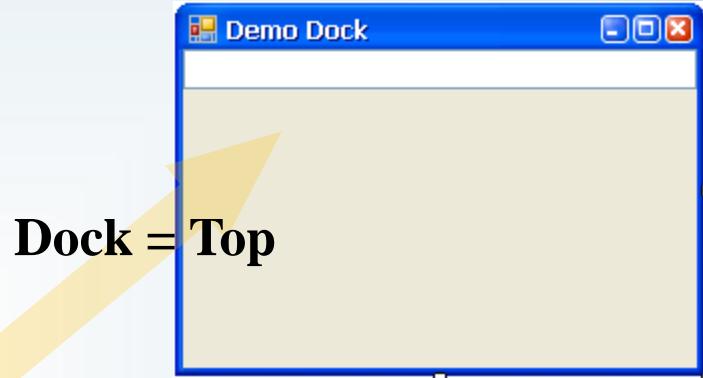
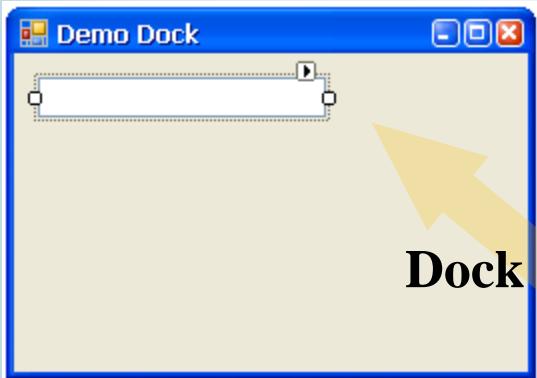
Windows Explorer



CONTROL LAYOUT - DOCKING



CONTROL LAYOUT - DOCKING



LABEL

- Cung cấp chuỗi thông tin chỉ dẫn:
 - Chỉ đọc;
 - Được định nghĩa bởi lớp Label:
 - ✓ Dẫn xuất từ control.

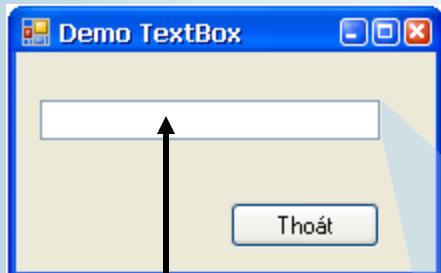
<i>Thuộc tính thường dùng</i>	
Font	Font hiển thị của text
Text	Nội dung text hiển thị
TextAlign	Canh lè text
ForeColor	Màu text
Visible	Trạng thái hiển thị

TEXTBOX

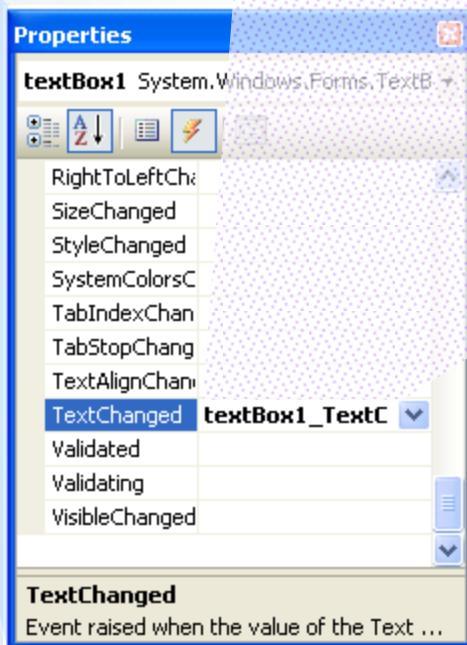
- Thuộc lớp TextBox;
- Vung cho phép user nhập dữ liệu: cho phép nhận dạng Password.

<i>Thuộc tính thường dùng</i>	
AcceptsReturn	Nếu true: nhấn enter tạo thành dòng mới trong chế độ multiline
Multiline	Nếu true: textbox ở chế độ nhiều dòng, mặc định là false
PasswordChar	Chỉ hiển thị ký tự đại diện cho text
ReadOnly	Nếu true: textbox hiển thị nền xám, và ko cho phép nhập liệu, mặc định là false
ScrollBars	Thanh cuộn cho chế độ multiline
<i>Event thường dùng</i>	
TextChanged	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

DEMO TEXTBOX



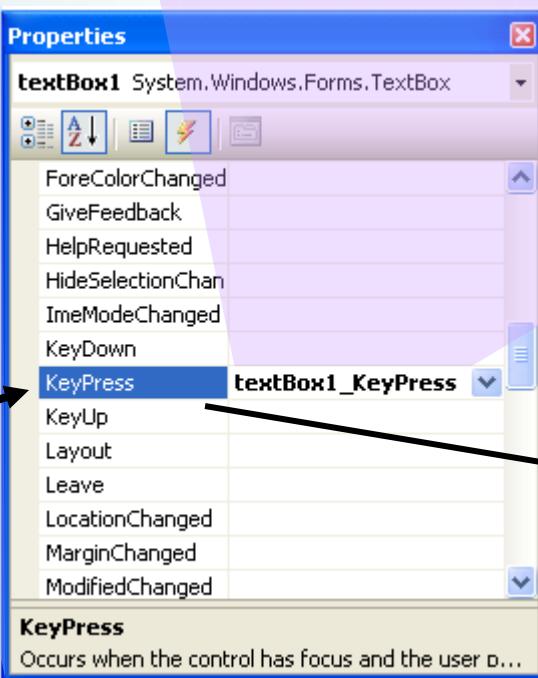
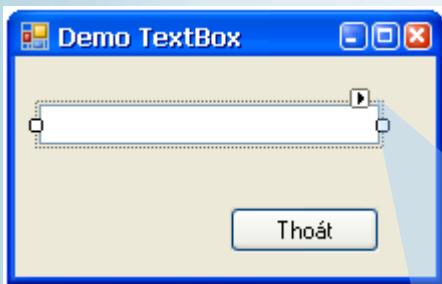
Double click vào
textbox để tạo event
handler cho event
TextChanged



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

Chuyển thành chữ hoa

DEMO TEXTBOX



```
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 KeyPress

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

BUTTON

- Cho phép cài đặt một hành động:
 - CheckBox và radioButton.
- Dẫn xuất từ ButtonBase.

Thuộc tính thường dùng

Text	Chuỗi hiển thị trên bề mặt button
-------------	-----------------------------------

Event thường dùng

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.
--------------	--

System.Object
System.Windows.Threading.DispatcherObject
System.Windows.DependencyObject
System.Windows.Media.Visual
System.Windows.UIElement
System.Windows.FrameworkElement
System.Windows.Controls.Control
System.Windows.Controls.ContentControl
System.Windows.Controls.Primitives.ButtonBase
System.Windows.Controls.Button

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

SelectedItems

Sorted

Text



LISTBOX (METHOD & EVENT)

Method

ClearSelected

GetSelected

SetSelected

FindString

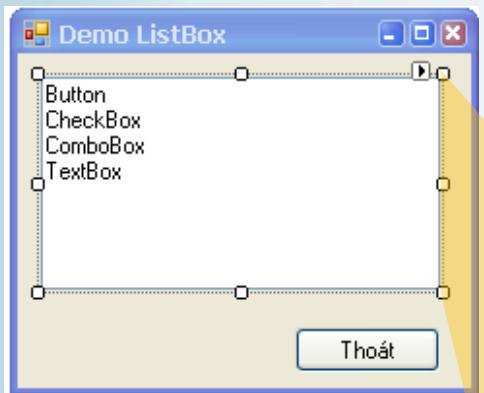


SelectedIndexChanged

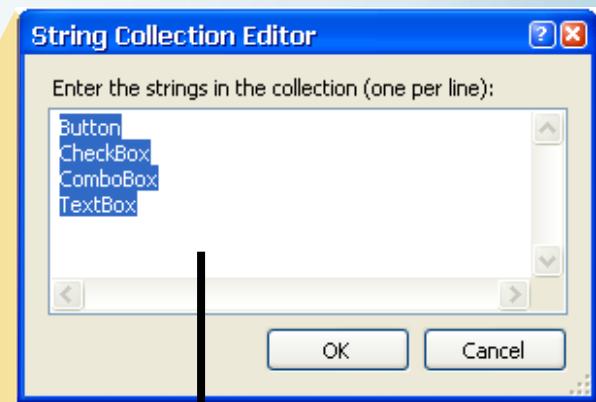
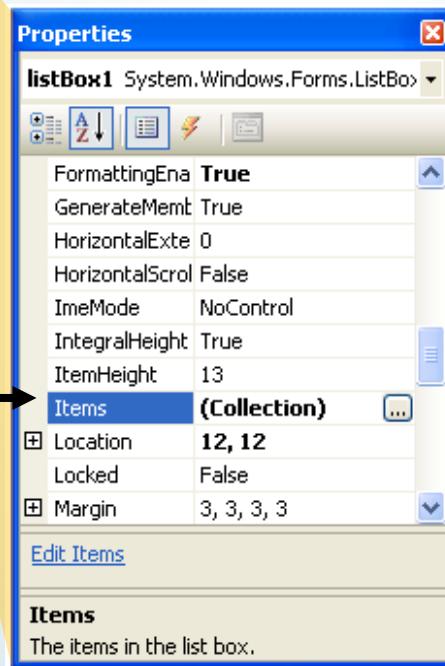
SelectedValueChanged

LISTBOX

- Thuộc tính **Items** cho phép thêm item vào ListBox.



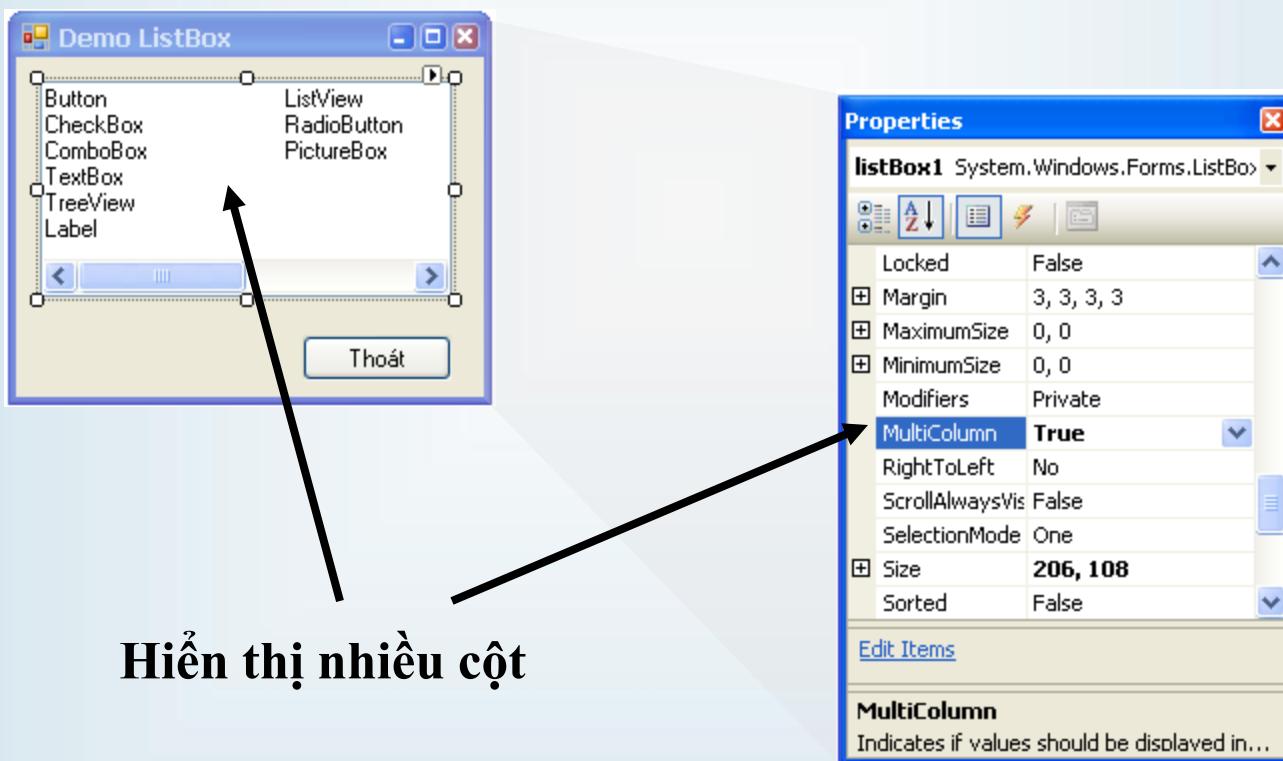
Danh sách item



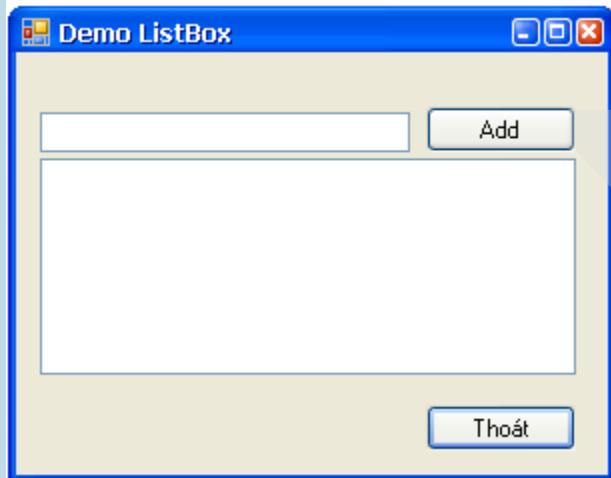
Cho phép thêm item
trong màn hình thiết
kế form

LISTBOX

- ListBox hiển thị dạng Multi Column:



DEMO LISTBOX



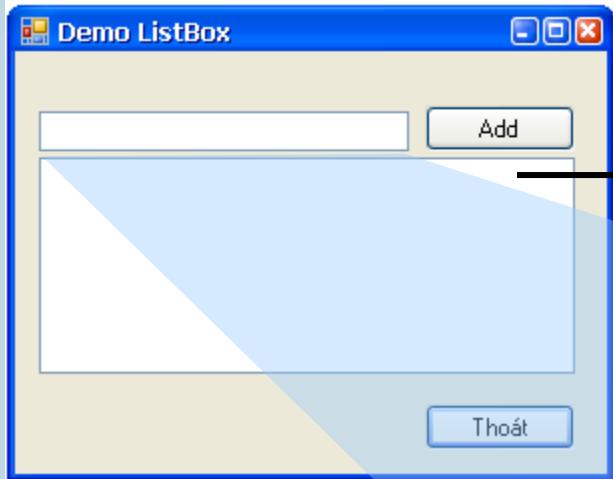
```
private void button2_Click(object sender, EventArgs e)
{
    if (listBox1.Items.IndexOf(textBox1.Text) >= 0)
        listBox1.SelectedItem = textBox1.Text;
    else if (textBox1.Text.Length > 0)
        listBox1.Items.Add(textBox1.Text);
}
```

Kiểm tra xem chuỗi nhập có trong list box?

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

LISTBOX

- Sự kiện *SelectedIndexChanged*:



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



SelectedIndexChanged

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

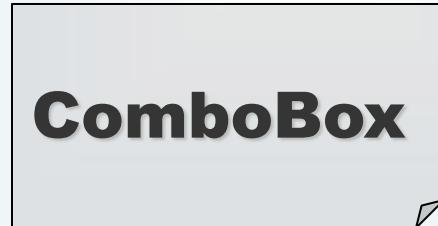
COMBOBOX

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

Items

DropDownStyle

Text



Sorted

MaxDropDownItems

AutoCompleteMode

DropDownHeight

COMBOBOX (DROP DOWN STYLE)

Simple



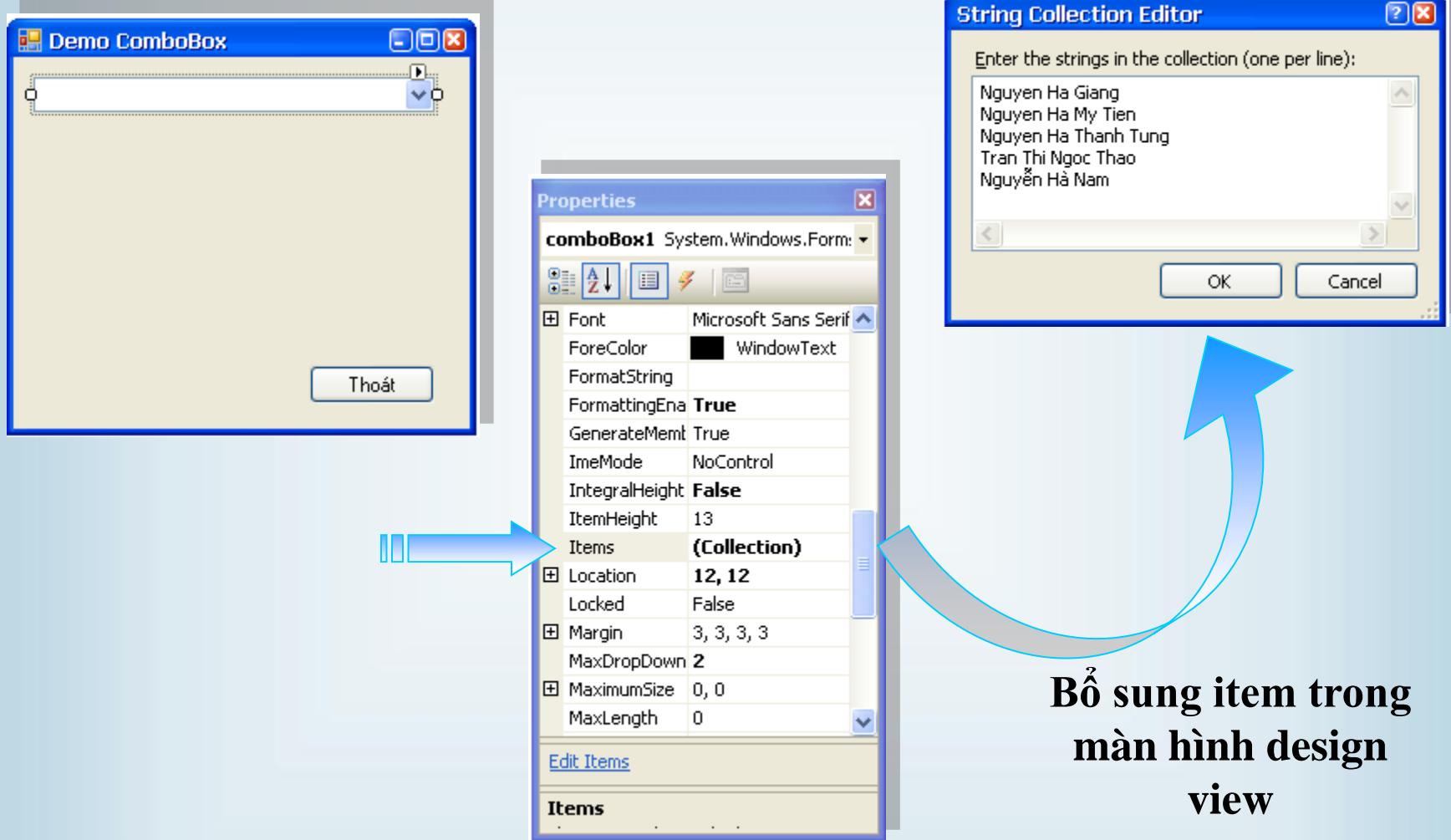
DropDown



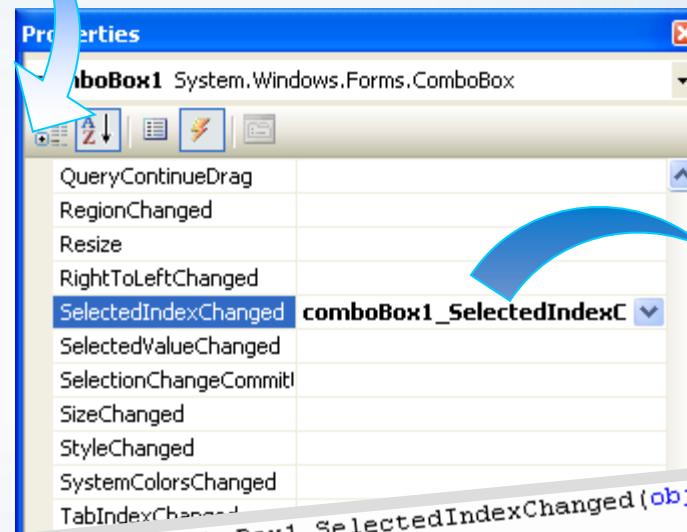
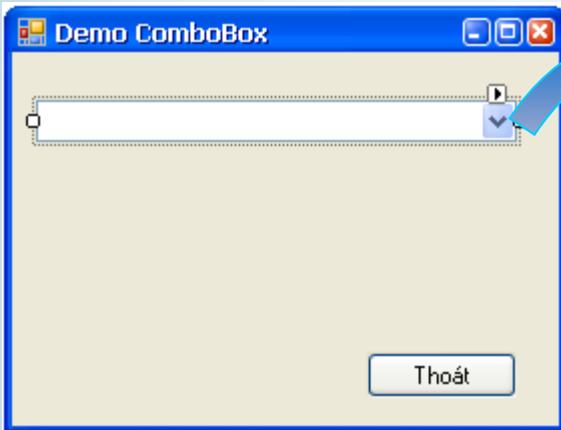
DropDownList



COMBOBOX



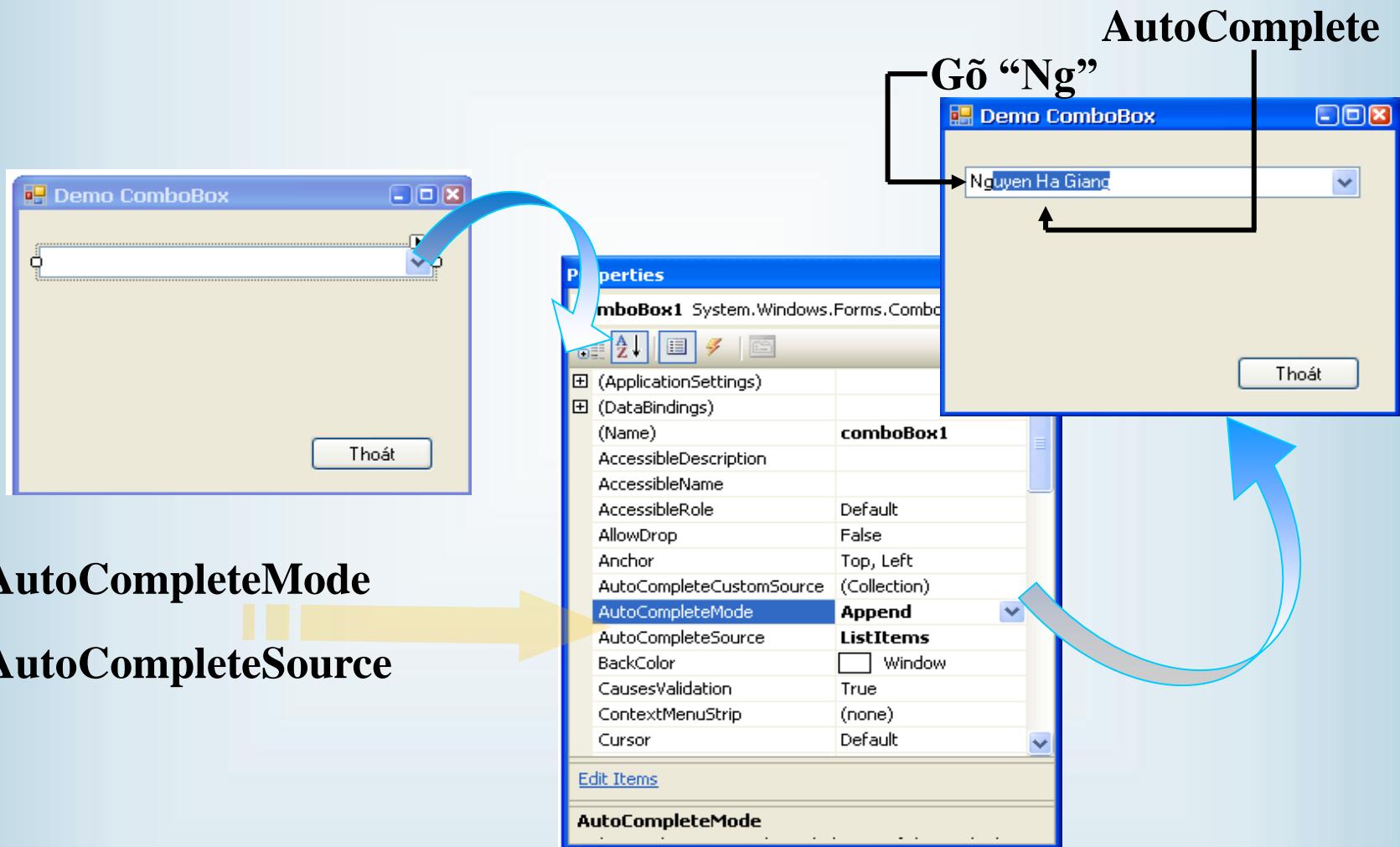
COMBOBOX



```
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 (AUTO COMPLETE)



LISTVIEW

- Dạng control phổ biến hiện thị một danh sách item:
 - Các item có thể có các item con gọi là subitem.
- Windows Explorer hiển thị thông tin thư mục, tập tin...
 - 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 (PROPERTIES)

GridLines	View
Sorting	SmallImageList
Columns	LargeImageList
Items	MultiSelect
	FullRowSelect

LISTVIEW

- Các dạng thẻ hiện của ListView:

Details

Small Icons

ListView

List

Large Icons

Tile

LISTVIEW

Large Icons

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



Small Icons

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



LISTVIEW

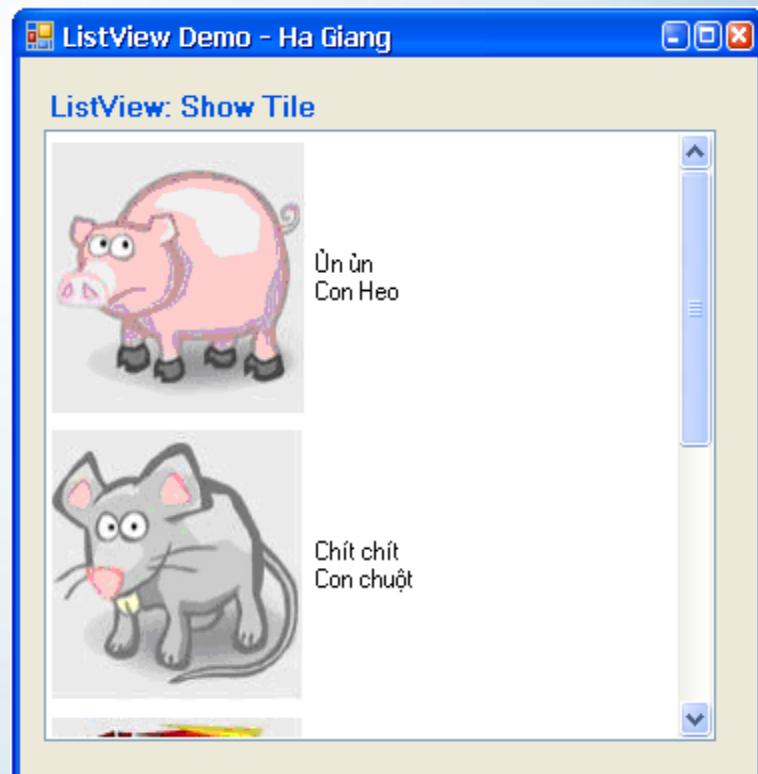
List

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



Title

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



LISTVIEW

Detail

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



Hành động	Động vật
 Ồn Ồn	Con Heo
 Chít chít	Con chuột
 Ô ô	Con Gà
 Cập cập	Con vịt

LISTVIEW

- Tạo các cột cho ListView – Details qua:
 - Cửa sổ properties → Columns để tạo;
 - Sử dụng code trong chương trình.

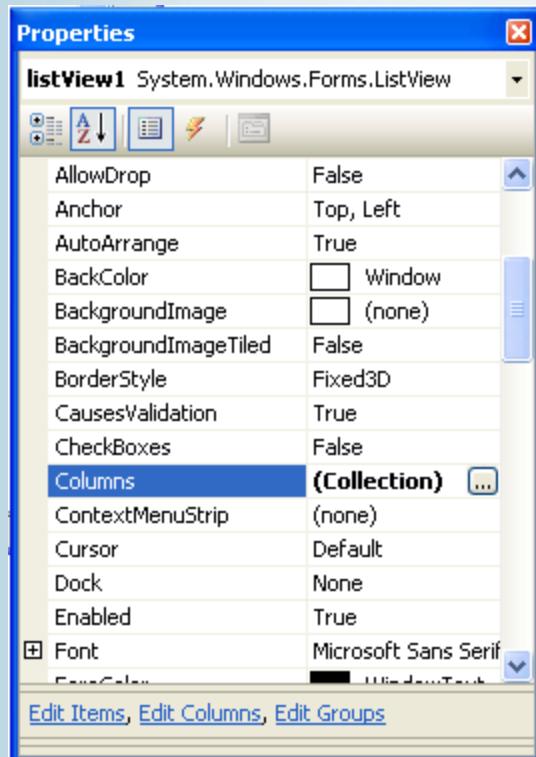
```
ColumnHeader columnHeader1 = new ColumnHeader();  
ColumnHeader columnHeader2 = new ColumnHeader();  
ColumnHeader columnHeader3 = new ColumnHeader();
```

```
columnHeader1.Text = "Name";  
columnHeader2.Text = "Address";  
columnHeader3.Text = "Telephone Number";
```

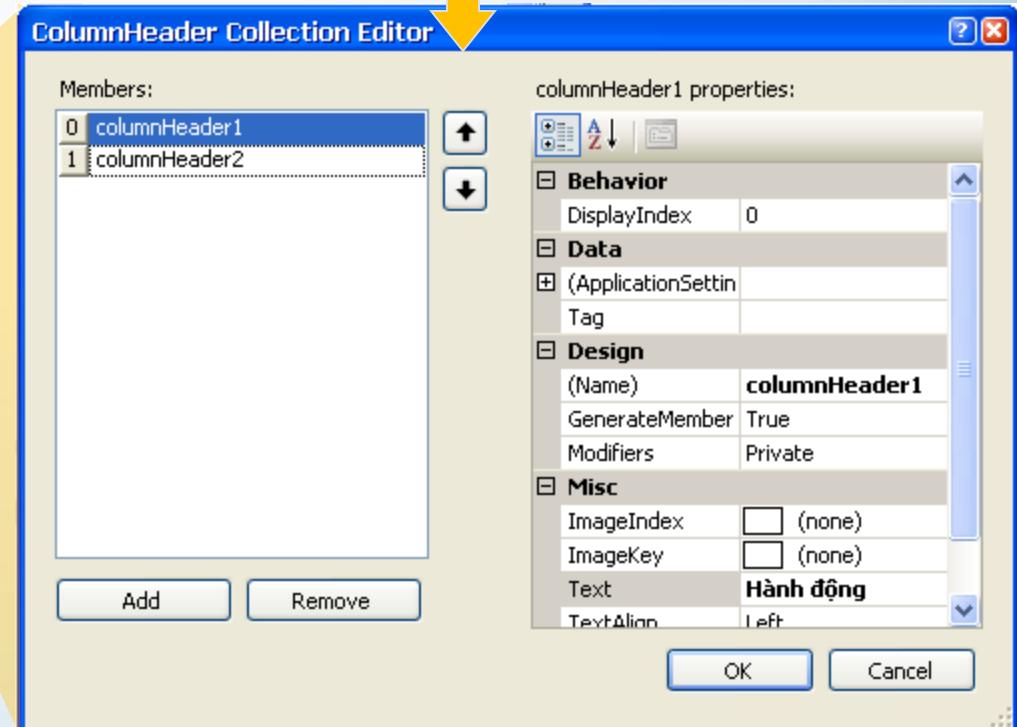
```
listView1.Columns.Add(columnHeader1);  
listView1.Columns.Add(columnHeader2);  
listView1.Columns.Add(columnHeader3);
```

Code

LISTVIEW

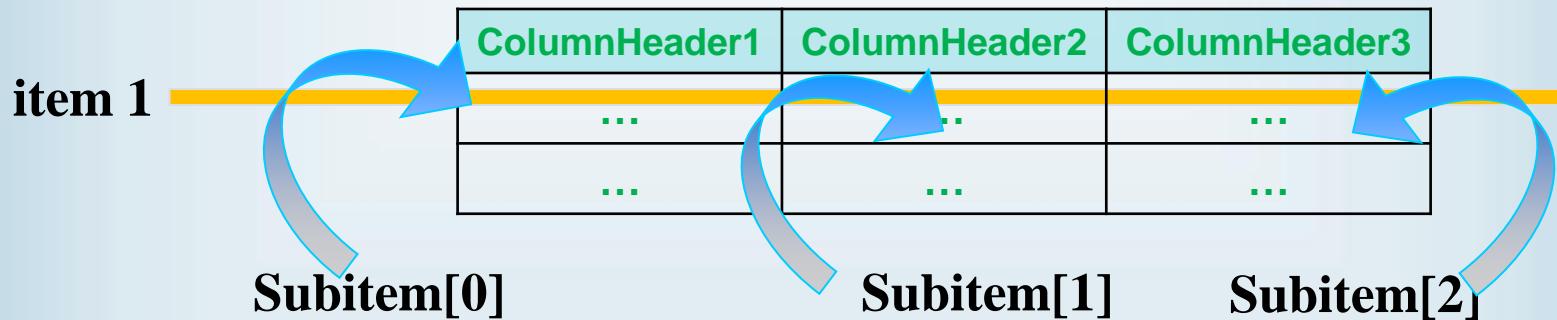


Dialog soạn thảo cột



LISTVIEW

- Thêm các item vào ListView:
 - Thêm item trong màn hình thiết kế form;
 - Thêm item thông qua code.
- 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.



LISTVIEW

```
ListViewItem item1 = new ListViewItem();  
ListViewItem.ListViewSubItem subitem1;  
subitem1 = new ListViewItem.ListViewSubItem();
```

```
item1.Text = "Hutech";  
subitem1.Text = "144/24 DBP - F.25 - Q.BT";
```

```
item1.SubItems.Add(subitem1);
```

```
listView1.Items.Add(item1);
```

Thêm subitem vào item

Thêm item vào danh sách items của ListView

LISTVIEW (SỰ KIỆN SELECTEDINDEXCHANGED)

```
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);
    }
}
```

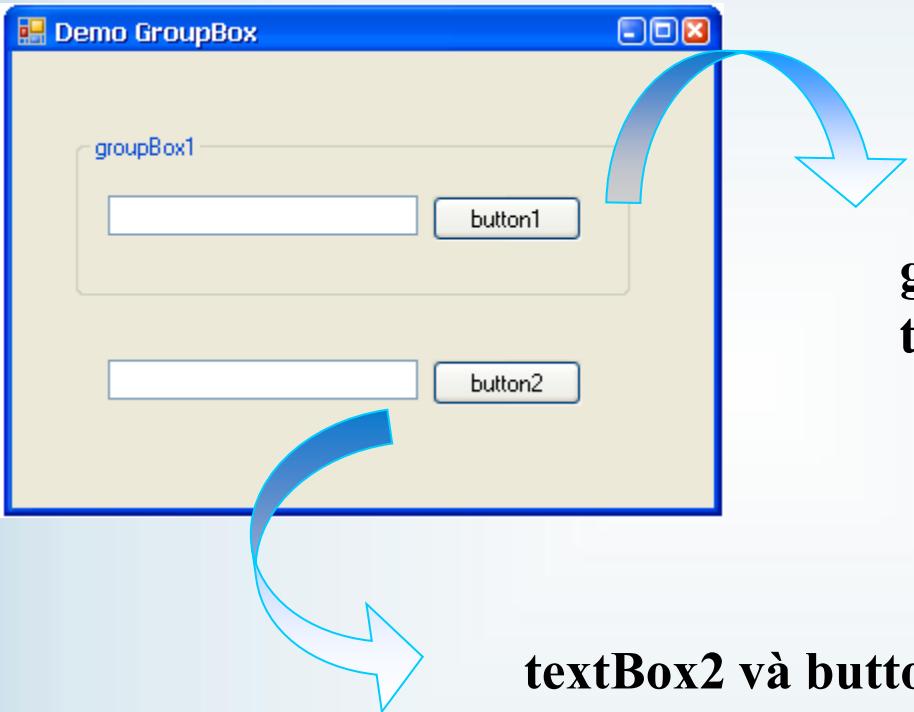
GROUPBOX & PANEL

- GroupBox:
 - Hiển thị một khung bao quanh một nhóm control;
 - Có thể hiển thị một tiêu đề:
 - ✓ Thuộc tính **Text**.
 - Khi xóa một GroupBox thì các control chứa trong nó bị xóa theo;
 - Lớp GroupBox kế thừa từ **System.Windows.Forms.Control**.
- Panel:
 - Chứa nhóm các control;
 - Không có Caption;
 - Có thanh cuộn (scrollbar):
 - ✓ Xem nhiều control khi kích thước panel giới hạn.

GROUPBOX & PANEL

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

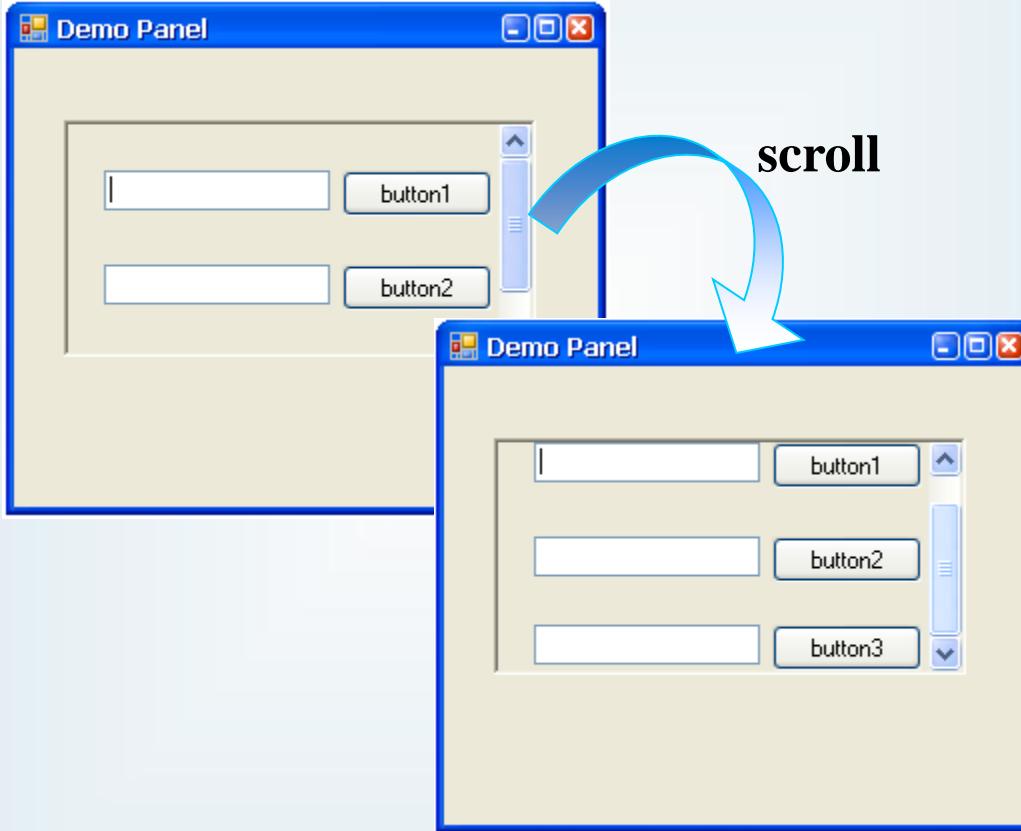
DEMO GROUPBOX



groupBox1 chứa 2 control
textBox1 và **button1**

textBox2 và **button2** chứa
trong **Controls** của Form

DEMO PANEL



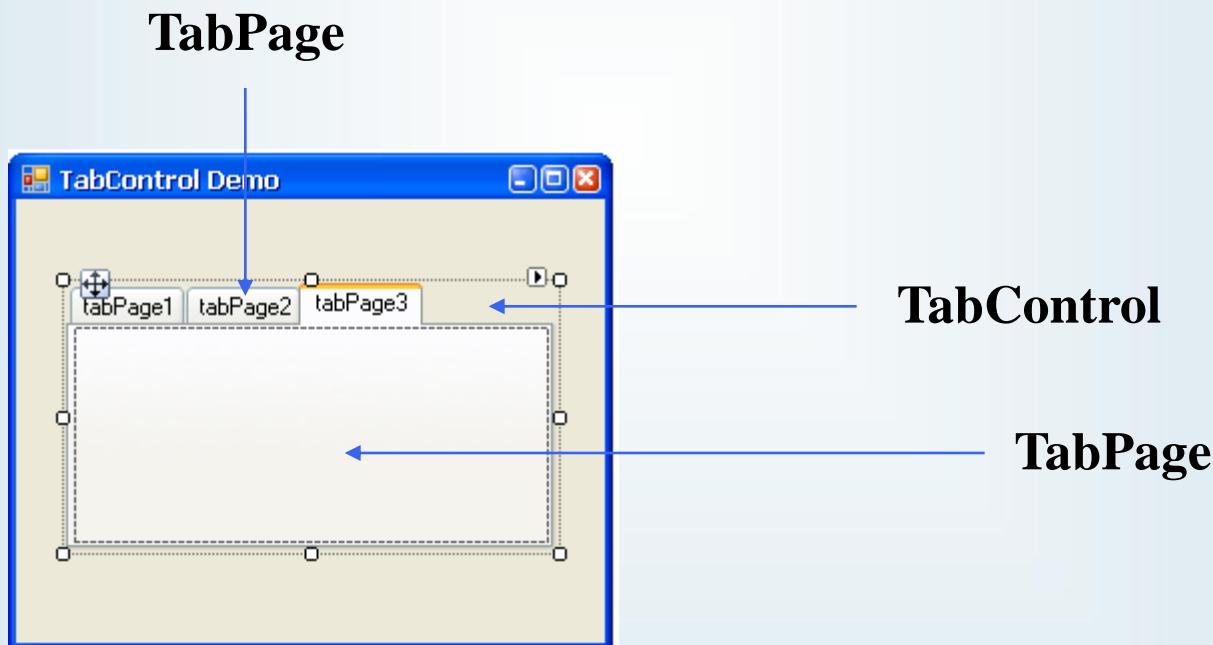
TABCONTROL

- Dạng container chứa các control khác;
- Cho phép thể hiện nhiều page trên một form duy nhất;
- 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 qua lại giữa các 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).

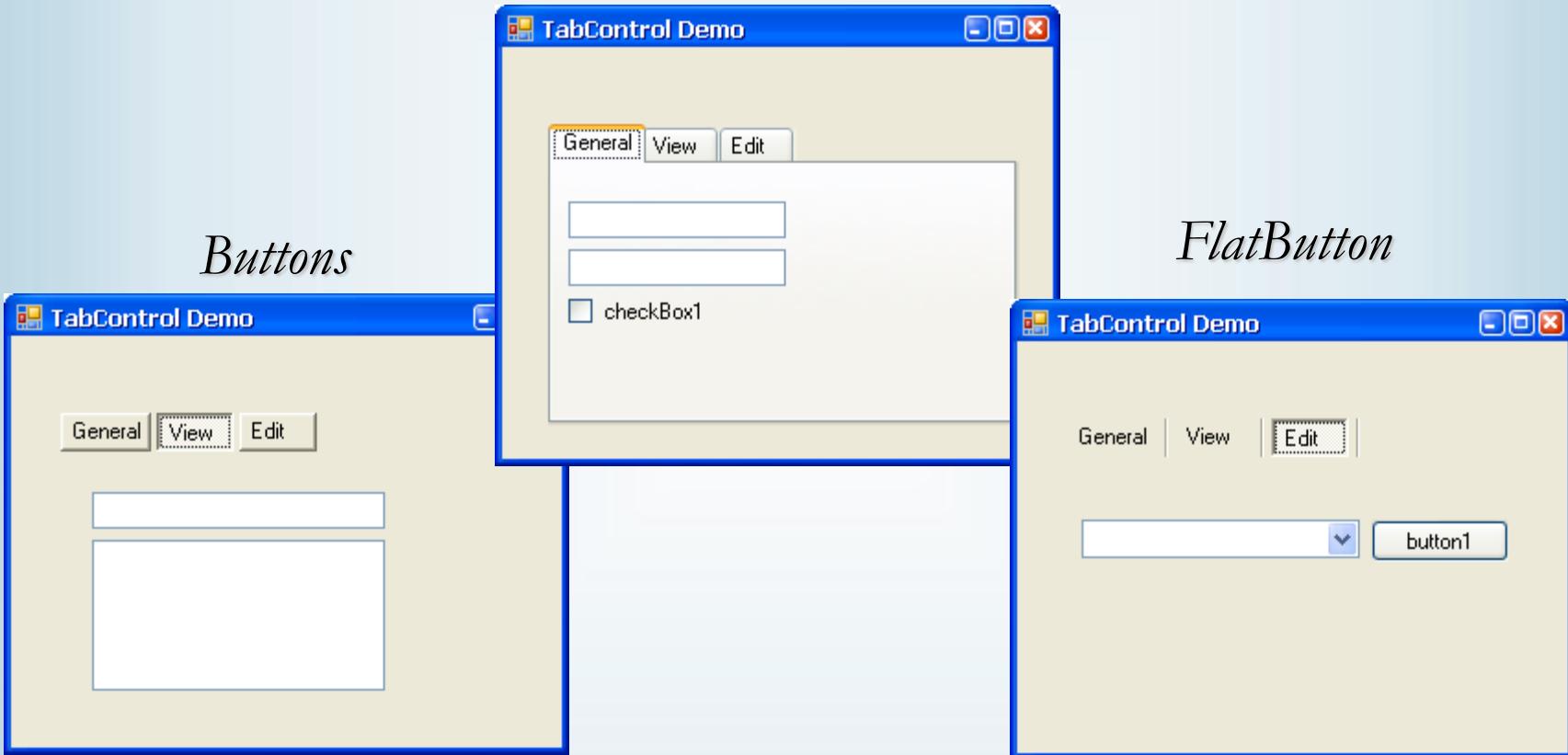
```
System.Object
System.MarshalByRefObject
System.ComponentModel.Component
System.Windows.Forms.Control
System.Windows.Forms.TabControl
```

TABCONTROL – THUỘC TÍNH TABPAGES

- Chứa các đối tượng TabPage:



TABCONTROL – THUỘC TÍNH APPEARANCE



TABCONTROL

- Thuộc tính, phương thức và sự kiện thường dùng:

Properties

TabPage

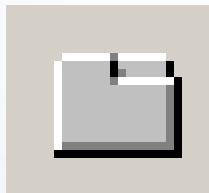
TabCount

SelectedTab

Multiline

SelectedIndex

TabControl



Method

SelectTab

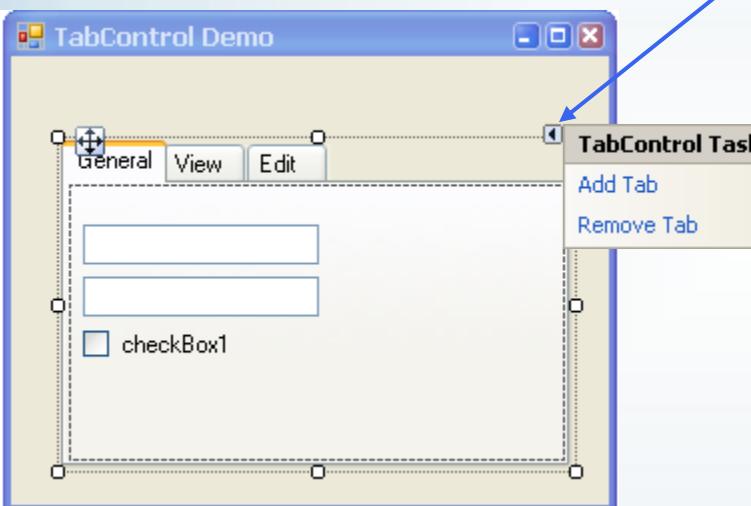
DeselectTab

Event

SelectedIndexChanged

TABCONTROL

THÊM, XÓA TABPAGE



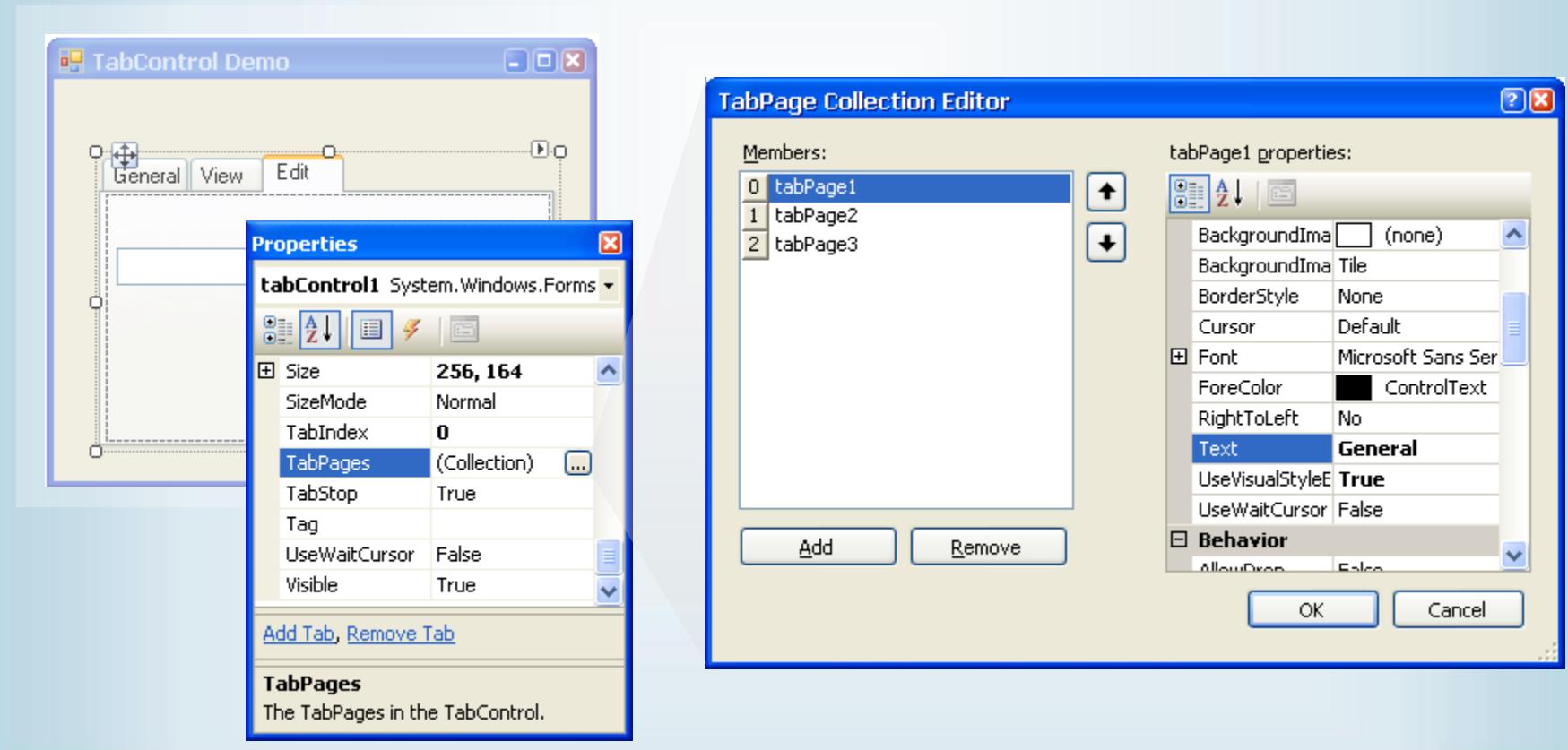
Kích chuột phải

Thêm/XóaTabPage

TABCONTROL

CHỈNH SỬA TABPAGE

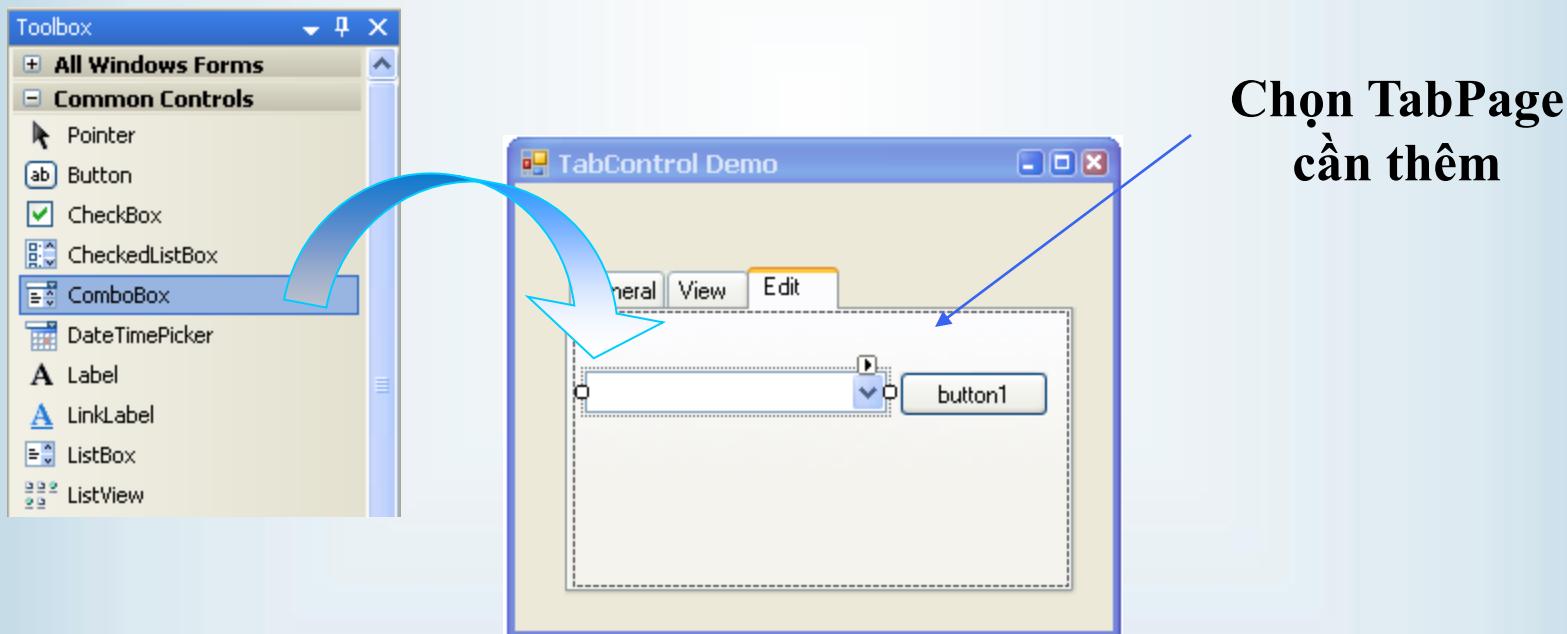
- Chọn thuộc tính TabPages của TabControl;
- Sử dụng màn hình TabPage Collection Editor để chỉnh sửa.



TABCONTROL

BỎ SUNG CONTROL VÀO TABCONTROL

- Chọn TabPage cần thêm control;
- Kéo control từ ToolBox thả vào TabPage đã chọn.



TABCONTROL

BỎ SUNG CONTROL VÀO TABCONTROL (CODE)

```
private void AddTabControl()
{
    TabControl tabControl1 = new TabControl();
   TabPage tabPageGeneral = newTabPage("General");
   TabPage tabPageView = newTabPage("View");

    tabControl1.TabPages.Add(tabPageGeneral);
    tabControl1.TabPages.Add(tabPageView);

    tabControl1.Location = new Point(20, 20);

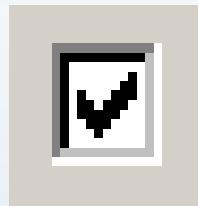
    this.Controls.Add(tabControl1);
}
```

CHECKBOX

- Control đưa ra một giá trị cho trước và user có thể:
 - Chọn giá trị khi Checked = true;
 - Không chọn giá trị: Checked = false.
- Lớp đại diện CheckBox.

Properties

Appearance



CheckedChanged

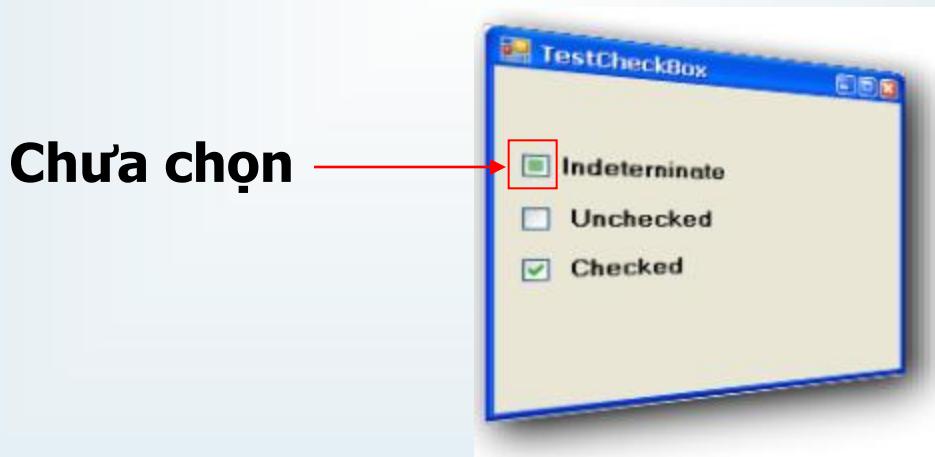
Checked

Text

ThreeState

CHECKBOX

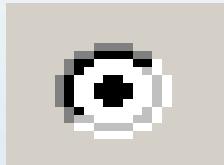
- ThreeState = true : cho phép thiết lập 3 trạng thái:
 - Checkstate = Indeterminate: không xác định;
 - CheckState= Checked: chọn;
 - CheckState= Unchecked: không chọn.



RADIOBUTTON

- 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 nhóm CheckBox cho phép chọn nhiều option, còn RadioButton chỉ cho chọn một trong số các option.

Appearance



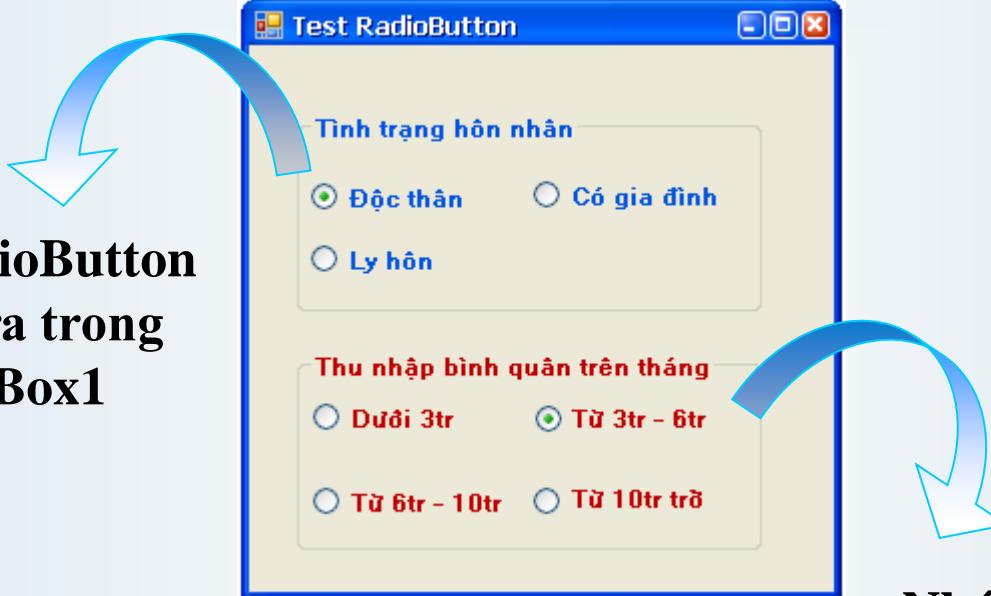
Checked

Text



CheckedChanged

RADIOBUTTON



Nhóm RadioButton
thứ 1 chứa trong
GroupBox1

Nhóm RadioButton
thứ 2 chứa trong
GroupBox2

CHECKLISTBOX

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

Properties

CheckedItems

CheckedIndices

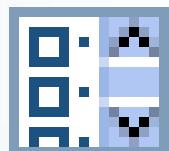
SelectedIndices

SelectedIndices

MultiColumn

SelectionMode

Items



SelectedIndexChanged

SelectedValueChanged

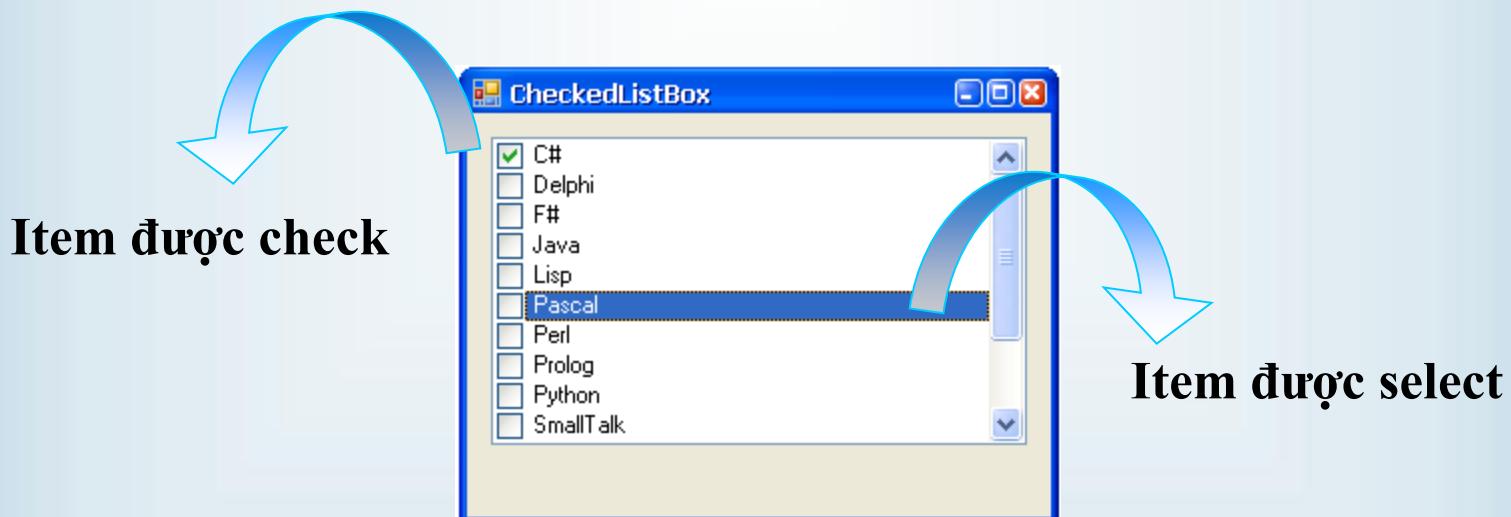
Method

ClearSelected

SetSelected

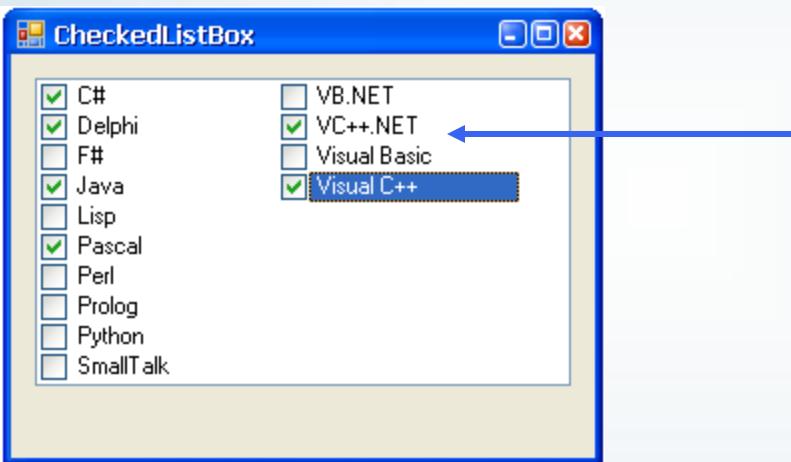
CHECKLISTBOX

- Thuộc tính **Items** lưu trữ danh sách item;
- Có thể bổ sung vào thời điểm:
 - Design time;
 - Run time.



CHECKLISTBOX

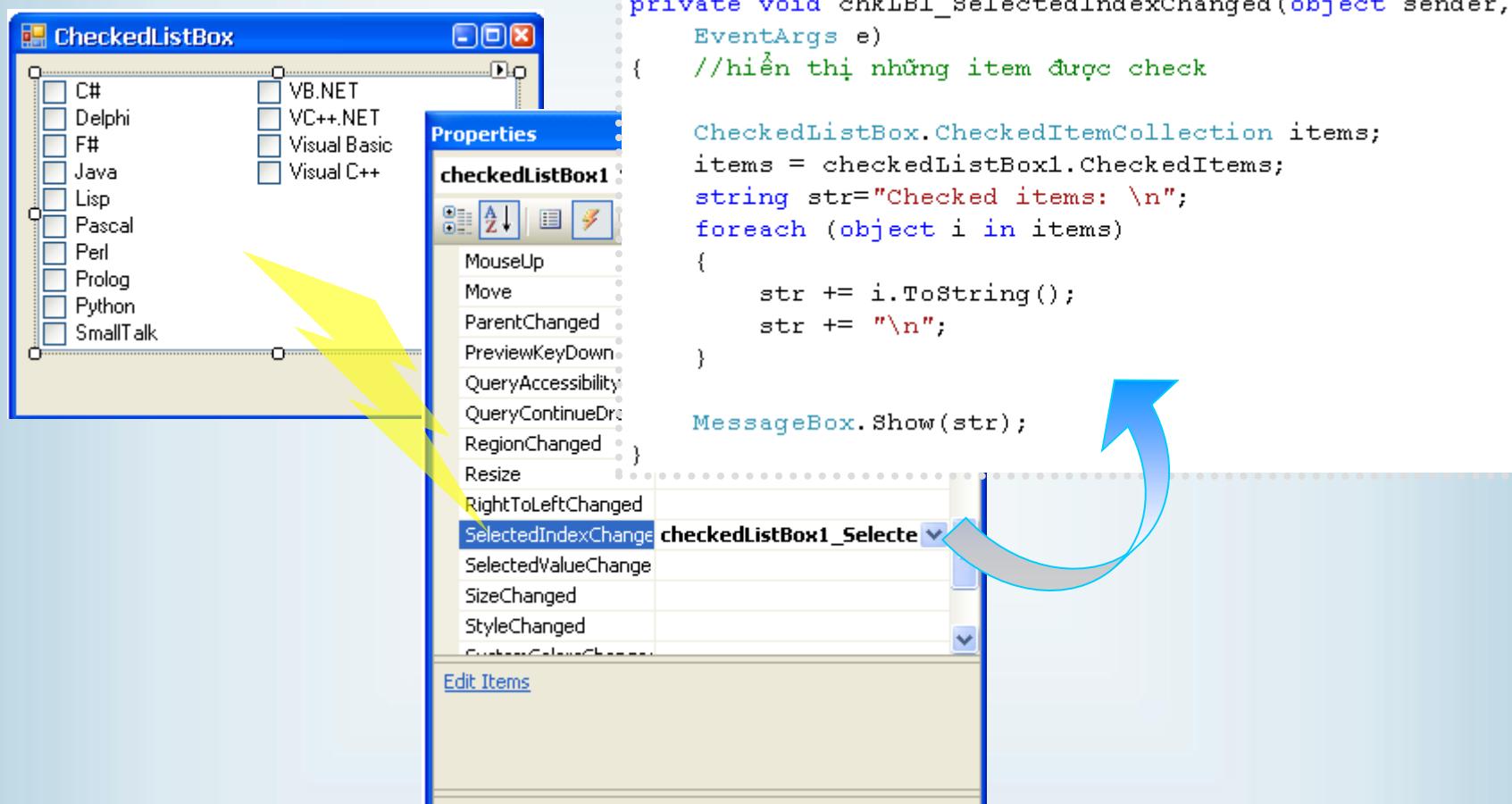
- **MultiColumn = true.**



Các item được tổ
chức theo nhiều cột

CHECKLISTBOX

SỰ KIỆN SELECTEDINDEXCHANGED



MOUSE EVENT

- Mouse là thiết bị tương tác thông dụng trên GUI;
- Một số các thao tác phát sinh từ mouse:
 - Di chuyển;
 - Kích chuột.
- Ứng dụng cần xử lý sự kiện chuột nào sẽ khai báo trình xử lý tương ứng;
- Lớp **MouseEventArgs** được sử dụng để chứa thông tin truyền vào cho trình xử lý sự kiện mouse;
- Mỗi trình xử lý sự kiện sẽ có tham số là đối tượng object và đối tượng MouseEventArgs (hoặc EventArgs).

MOUSE EVENT

- Tham số cho sự kiện liên quan đến mouse

MouseEventArgs

Số lần kích chuột



Button được nhấn

Tọa độ (x,y) của con
trỏ chuột

MOUSE EVENT

Sự kiện chuột với tham số kiểu EventArgs

MouseEnter	Xuất hiện khi con trỏ chuột đi vào vùng biên của control
-------------------	--

MouseLeave	Xuất hiện khi con trỏ chuột rời khỏi biên của control
-------------------	---

Sự kiện chuột với tham số kiểu MouseEventArgs

MouseDown/ MouseUp	Xuất hiện khi button được nhấn/thả và con trỏ chuột đang ở trong vùng biên của control
-------------------------------	--

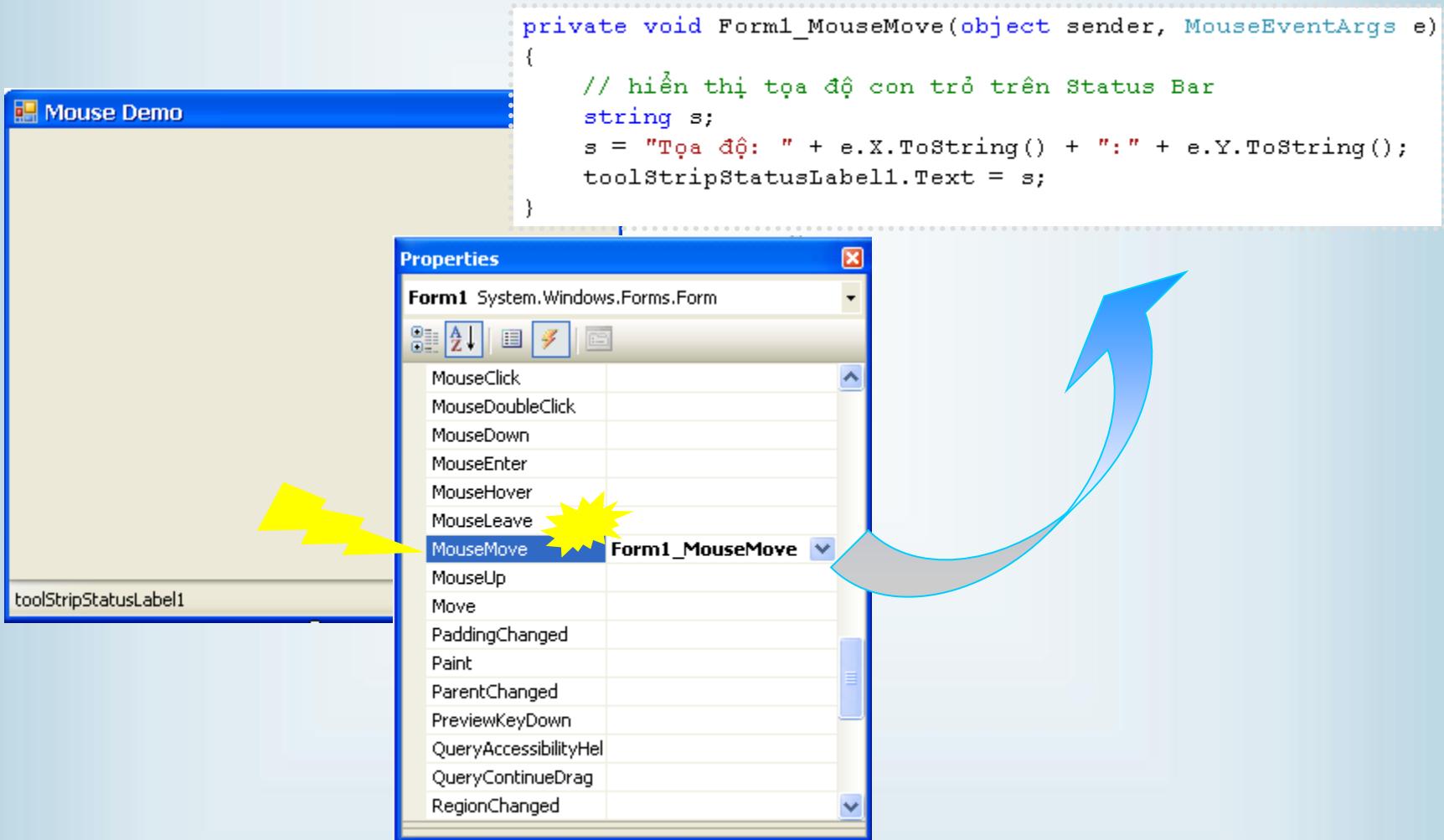
MouseMove	Xuất hiện khi chuột di chuyển và con trỏ chuột ở trong vùng biên của control
------------------	--

MOUSE EVENT

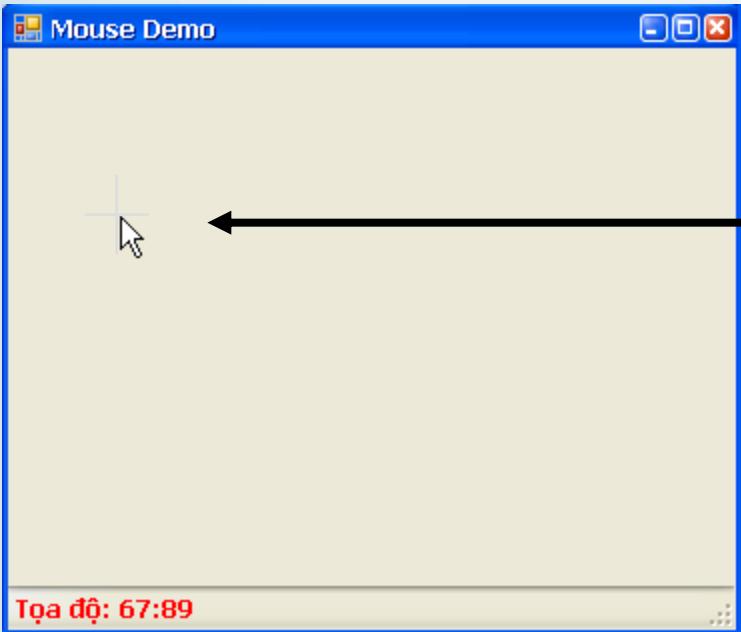
Thuộc tính của lớp MouseEventArgs

Button	Button được nhấn {Left, Right, Middle, none} có kiểu là MouseButtons
Clicks	Số lần button được nhấn
X	Tọa độ x của con trỏ chuột trong control
Y	Tọa độ y của con trỏ chuột trong control

MOUSE EVENT – MOuseMove



DEMO MOUSE EVENT



Vị trí hiện tại của
con trỏ chuột

Hiển thị tọa độ hiện
tại của con trỏ chuột

MOUSE EVENT

- Demo thao tác: kích chuột trái tại một điểm A, giữ chuột trái và di chuyển chuột, chương trình sẽ vẽ đường thẳng từ điểm A đến vị trí hiện tại chuột.
- Các sự kiện cần xử lý
 - MouseDown:
 - Xác định điểm A ban đầu
 - MouseMove
 - Kiểm tra nếu Left button của chuột đang giữ
 - Sử dụng Graphics để vẽ đường thẳng từ A đến vị trí hiện tại

MOUSE EVENT

- Bước 1:

- Tạo biến lưu trữ điểm A khi user kích chuột trái
- Biến pA có kiểu Point là biến thành viên của Form1

Lớp Form1

```
public partial class Form1 : Form
{
    private Point pA; // biến lưu giữ tọa độ A

    public Form1()
    {
        InitializeComponent();
    }
}
```

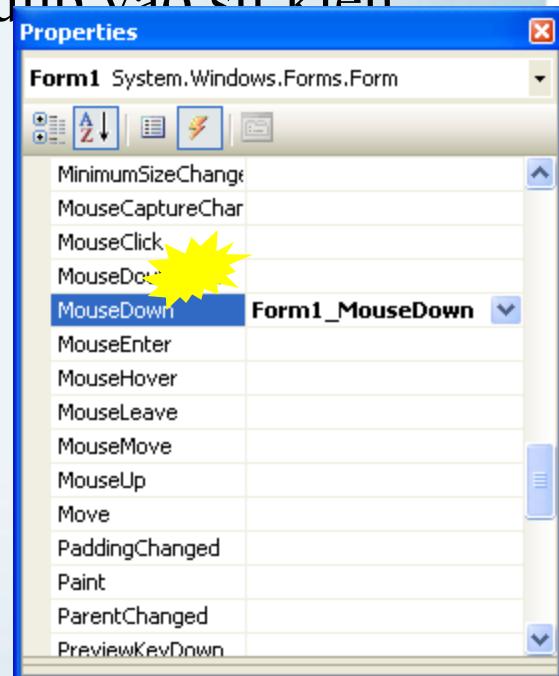
Biến pA lưu giữ tọa độ khi
chuột trái được click

MOUSE EVENT

- Bước 2
 - Khai báo xử lý sự kiện MouseDown trong Form1
 - Trong cửa sổ event của Form1, kích đúp vào sự kiện MouseDown

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    // lưu lại điểm thứ 1
    pA = e.Location;
}
```

Lưu lại điểm được nhấn chuột

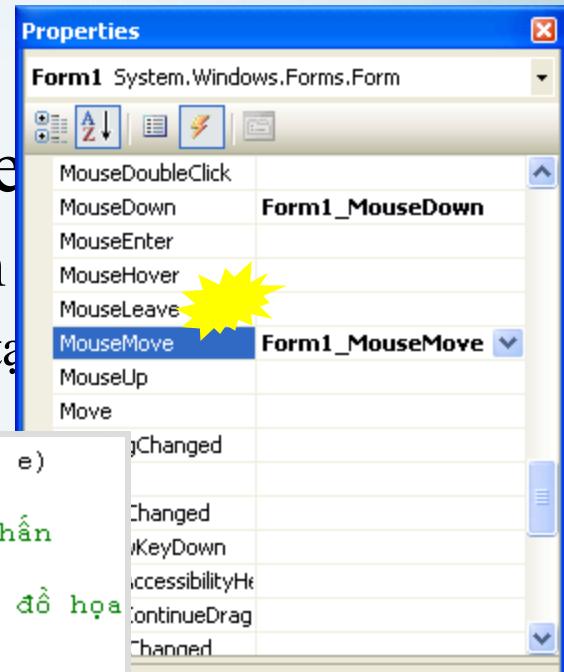


MOUSE EVENT

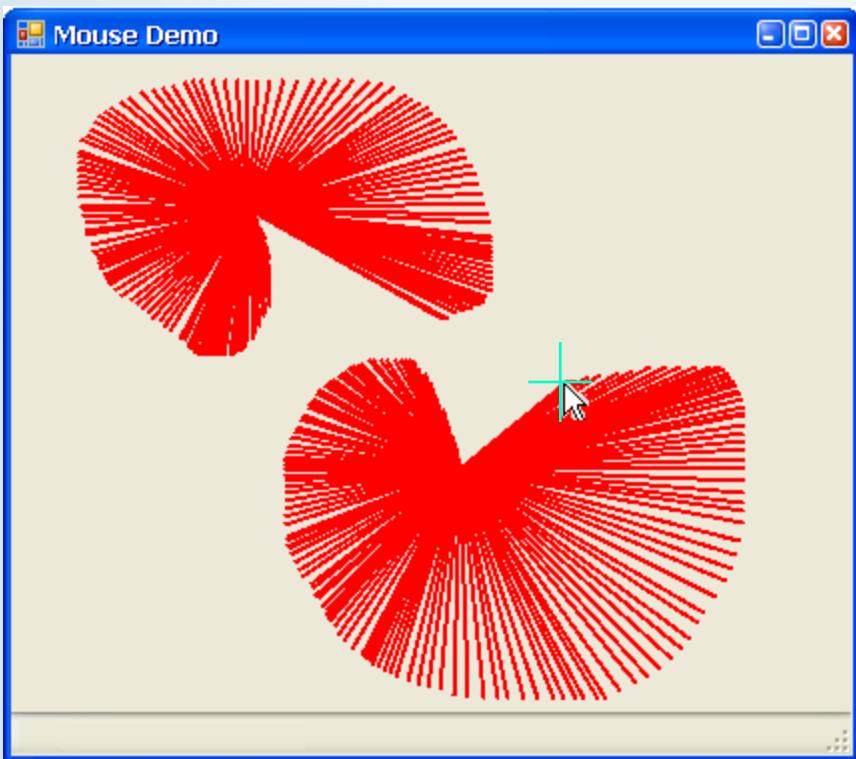
- Bước 3
 - Cài đặt xử lý sự kiện MouseMove
 - Kiểm tra nếu LeftButton được nhấn
 - Vẽ đường thẳng từ pA đến vị trí hiện tại

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left) // LeftButton được nhấn
    {
        Graphics g = this.CreateGraphics(); // lấy thiết bị đồ họa
        Pen pen = new Pen(Color.Red, 2f); // tạo bút vẽ

        g.DrawLine(pen, pA, e.Location); //vẽ đường thẳng
    }
}
```



MOUSE EVENT



KEYBOARD EVENT

- Phát sinh khi một phím được nhấn hoặc thả
- Có 3 sự kiện
 - KeyPress
 - KeyUp
 - KeyDown
- KeyPress phát sinh kèm theo với mã ASCII của phím được nhấn
- KeyPress không cho biết trạng thái các phím bổ sung {*Shift*, *Alt*, *Ctrl*...}
- Sử dụng KeyUp & KeyDown để xác định trạng thái các phím bổ sung.

KEYBOARD EVENT

Sự kiện với tham số kiểu KeyEventArgs

KeyDown Phát sinh khi phím được nhấn

KeyUp Phát sinh khi phím được thả

Sự kiện với tham số kiểu KeyPressEventArgs

KeyPress Khởi tạo khi phím được nhấn

Thuộc tính của lớp KeyPressEventArgs

KeyChar Chứa ký tự ASCII của phím được nhấn

Handled Cho biết sự kiện KeyPress có được xử lý chưa

Thuộc tính của lớp KeyEventArgs

Alt, Control, Shift Trạng thái các phím bổ sung

Handled Cho biết sự kiện đã xử lý

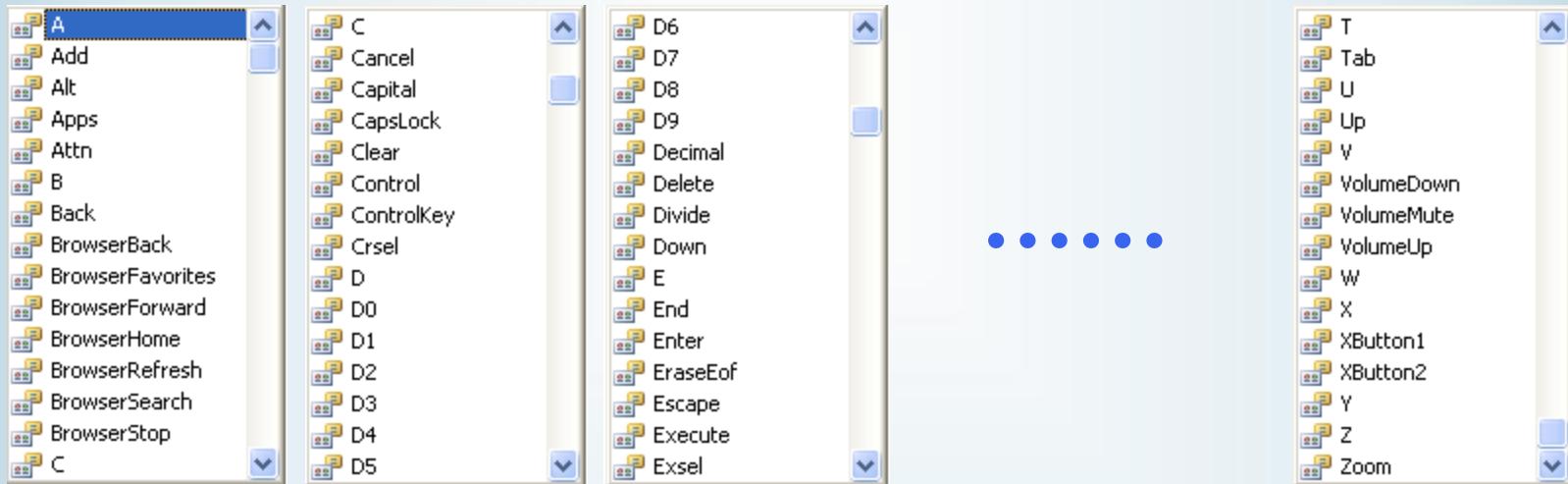
KEYBOARD EVENT

Thuộc tính của lớp KeyEventArgs (tt)

<i>Thuộc tính của lớp KeyEventArgs (tt)</i>	
KeyCode	Trả về mã ký tự được định nghĩa trong Keys enumeration
KeyData	Chứa mã ký tự với thông tin phím bổ sung
KeyValue	Trả về số int, đây chính là mã Windows Virtual Key Code
Modifier	Trả về giá trị của phím bổ sung

KEYBOARD EVENT

- Keys Enumeration



KEYBOARD EVENT

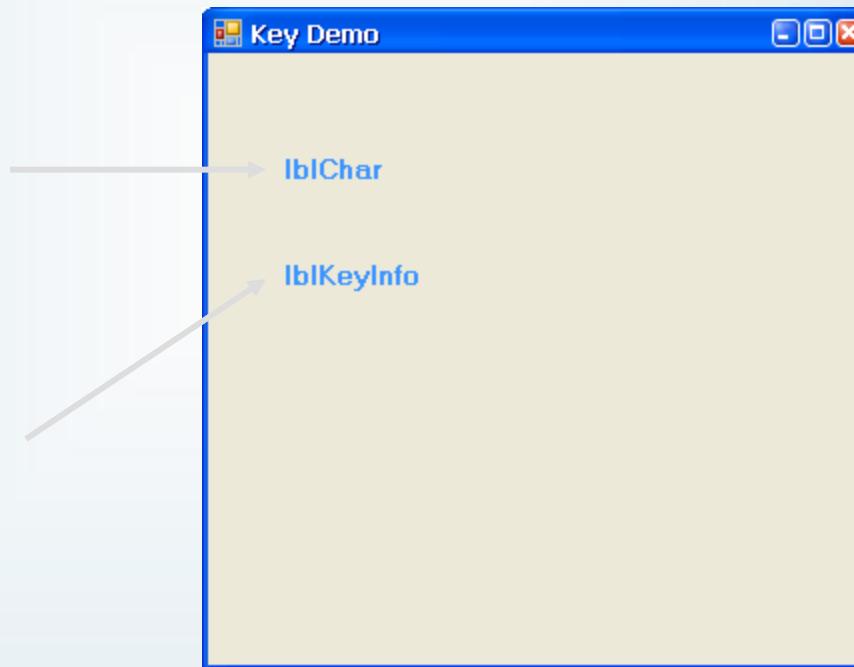
- Minh họa các sự kiện: KeyPress, KeyDown, KeyUp
 - Khi user nhấn một phím
 - ✓ Bắt sự kiện KeyPress: xuất ra phím được nhấn
 - ✓ Bắt sự kiện KeyDown: xuất ra các tham số trong KeyEventArgs
 - Khi user thả phím
 - ✓ Xóa các thông tin mô tả phím được nhấn trong các label
- Cách thực hiện
 - Tạo một form minh họa
 - Thiết kế trên form có 2 Label:
 - ✓ lblChar: hiển thị ký tự được nhấn trong KeyPress
 - ✓ lblKeyInfo: hiển thị các thông tin của KeyEventArgs khi KeyDown

KEYBOARD EVENT

- Bước 1: tạo Windows Form như hình mô tả

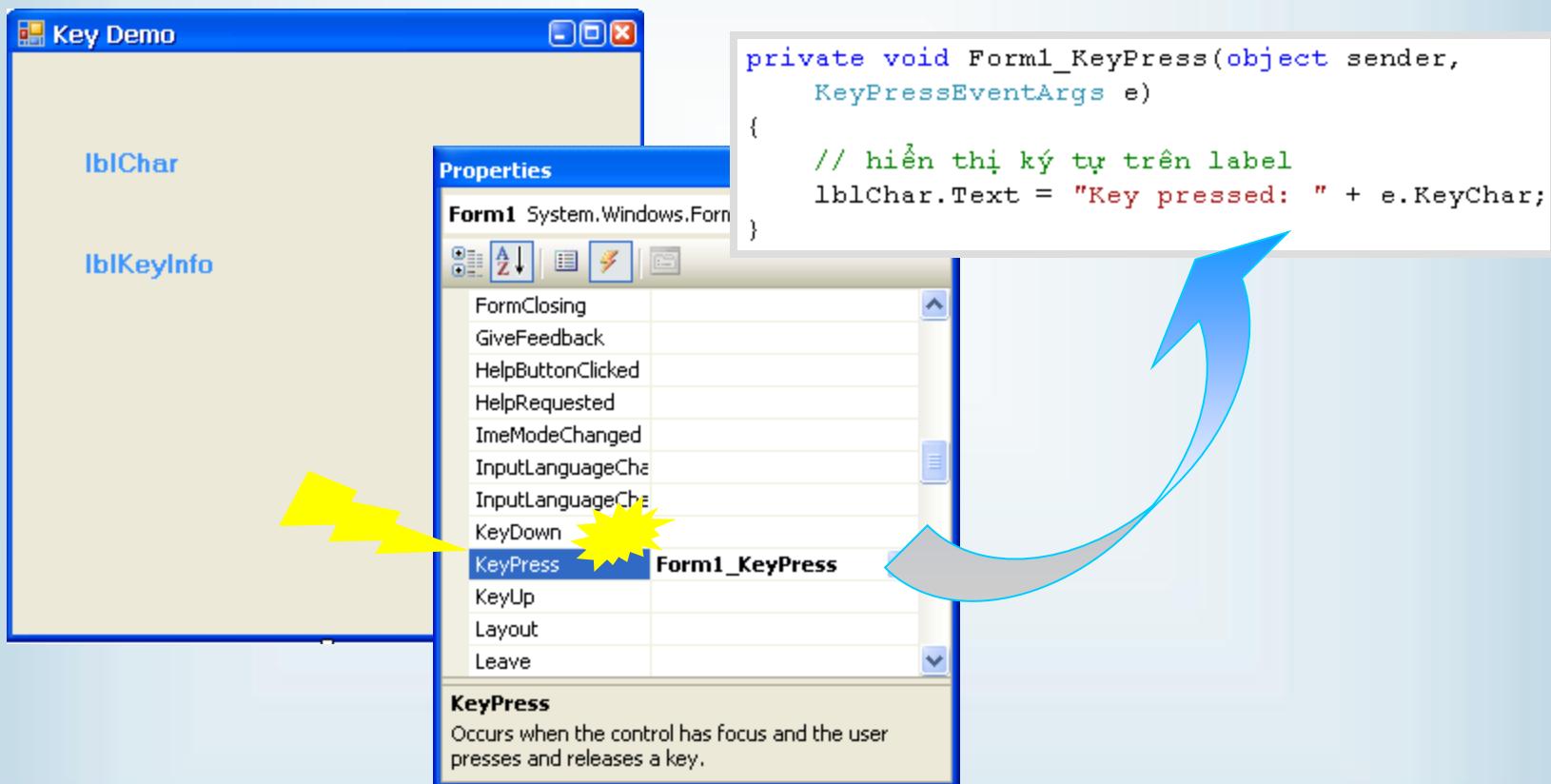
**Label chứa ký tự
được nhấn trong
sự kiện KeyPress**

**Label chứa thông
tin mã ký tự được
nhấn trong sự
kiện KeyDown**



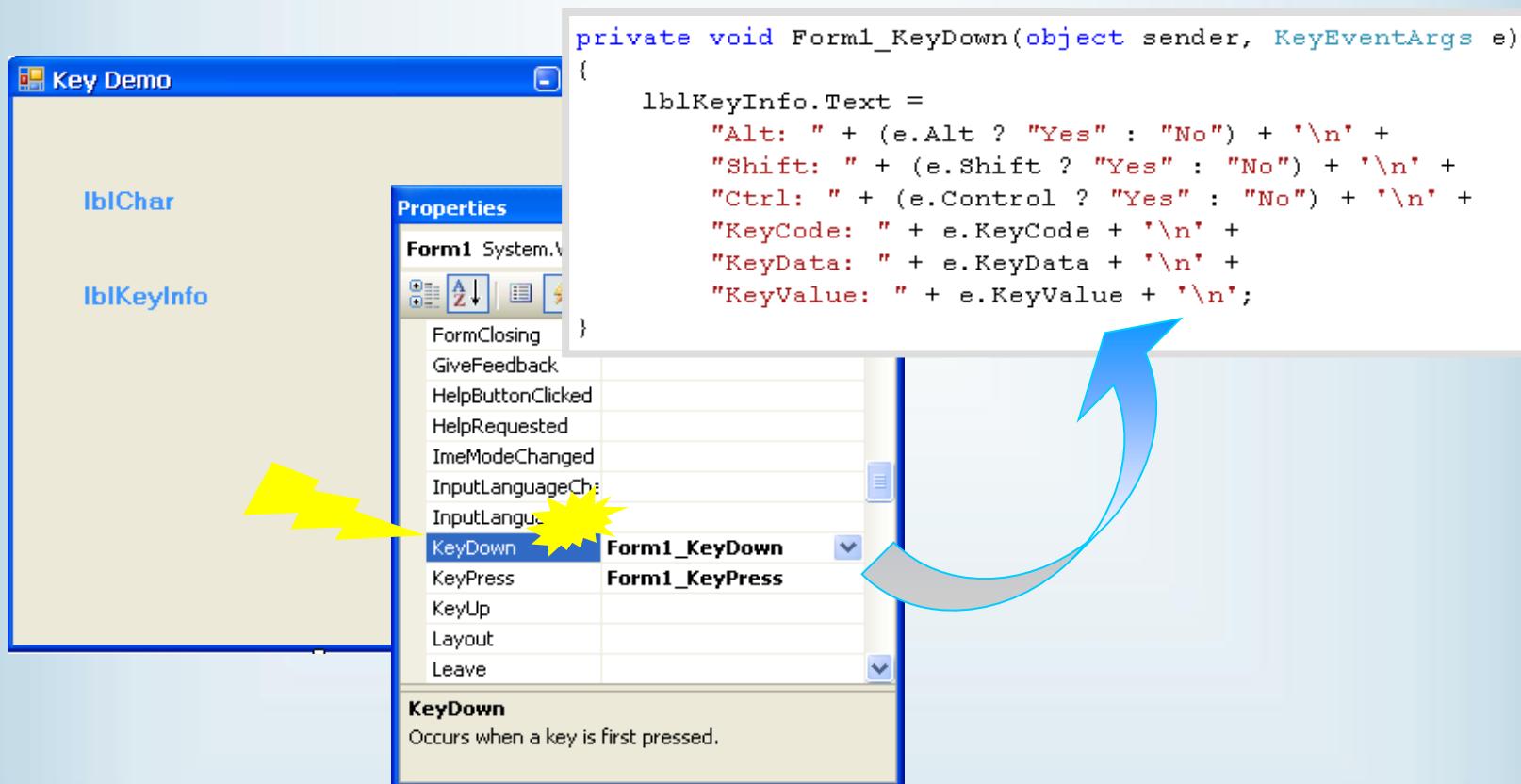
KEYBOARD EVENT

- Bước 2:
 - Tạo KeyPress Event Handling cho form

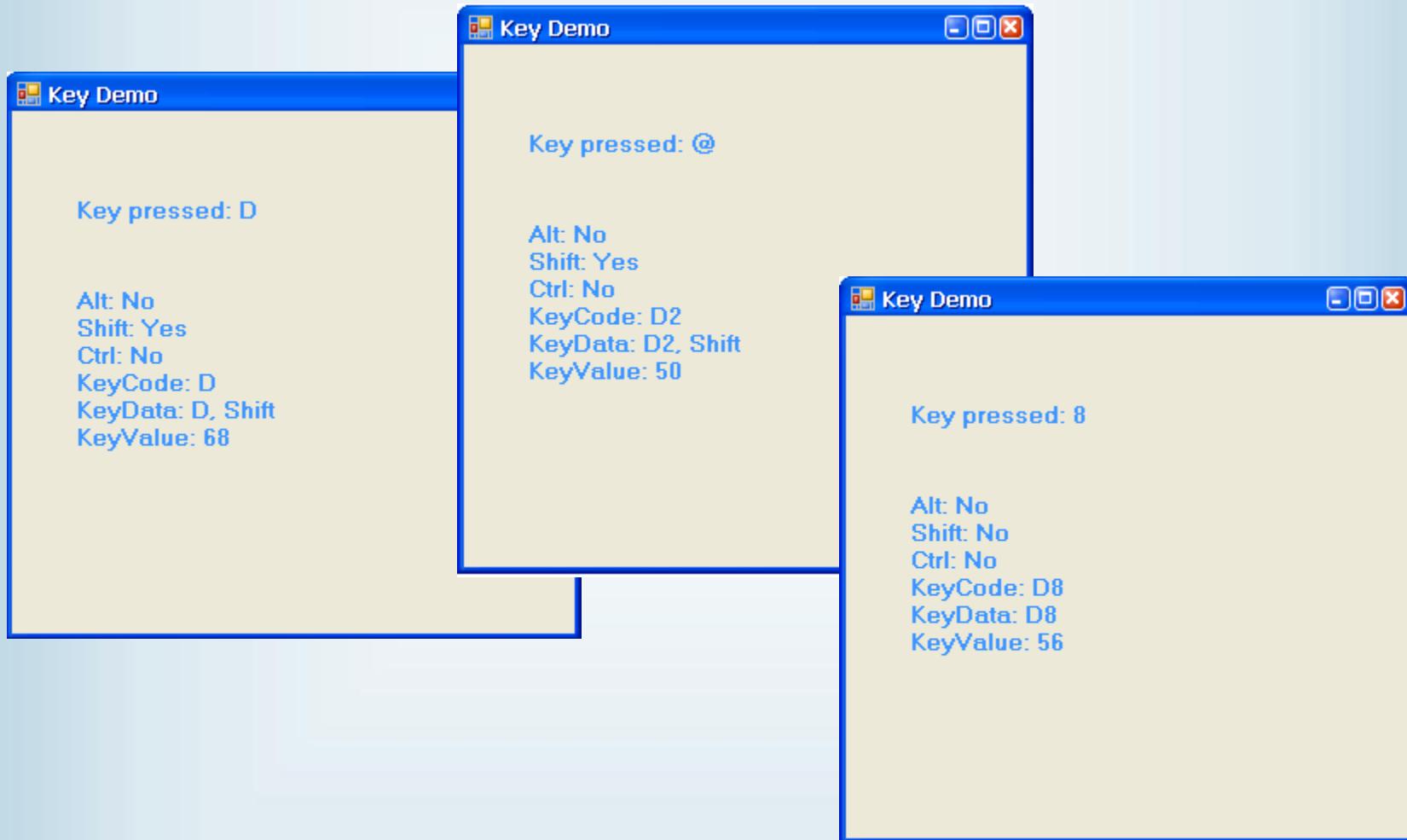


KEYBOARD EVENT

- Bước 3:
 - Tạo KeyDown Event Handling cho form

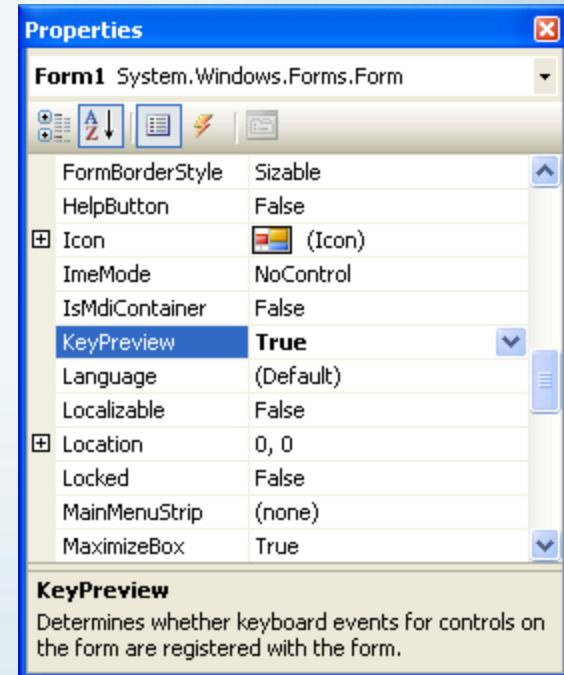


DEMO KEYBOARD EVENT



KEYBOARD EVENT

- CT Calculator (BT3) mở rộng cho phép xử lý các phím
 - Form nhận xử lý thông điệp KeyDown
 - Xác định các phím tương ứng rồi gọi sự kiện click của button
 - VD: user gõ phím 1, tương tự như button “1” được nhấn
- Cách thực hiện
 - Khai báo trình xử lý sự kiện KeyDown cho Form chính
 - Thiết lập thuộc tính *KeyPreview* cho Form để nhận sự kiện bàn phím.



KEYBOARD EVENT

- Viết phần xử lý cho sự kiện KeyDown
 - Xác định các phím tương ứng để gọi sự kiện click của các button.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.D1:
            btnNum1.PerformClick();
            break;
        case Keys.D2:
            btnNum2.PerformClick();
            break;
        //.....
        case Keys.Oemplus:
            if (!e.Shift)// phím '='
                btnEqual.PerformClick();
            else // phím '+'
                btnPlus.PerformClick();
            break;
        // còn tiếp cho các phím khác.....
    }
}
```

Gọi event Click
của button “1”

Phím '=' được nhấn

Phím '+' được nhấn

COMMON DIALOG

- Các dialog thường được sử dụng
 - Được warp thành các lớp trong FCL/BCL

OpenFileDialog

Common
Dialog

ColorDialog

SaveFileDialog

FontDialog

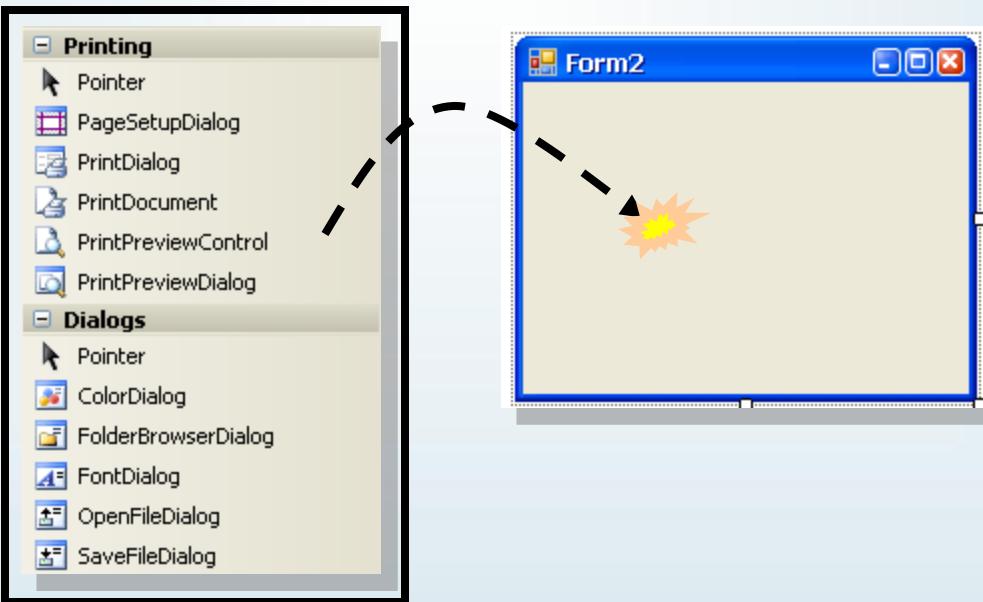
PrintDialog

....

COMMON DIALOG

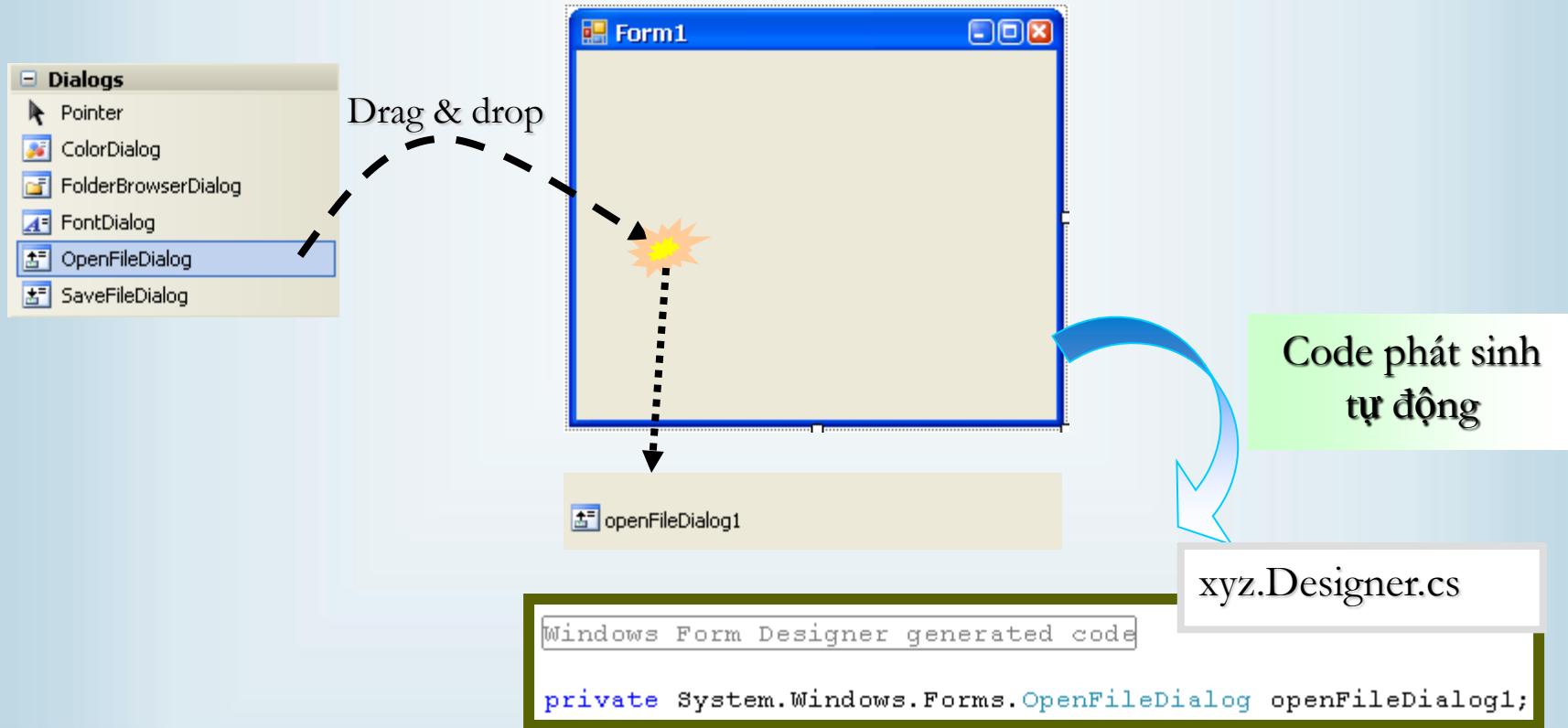
- Cách sử dụng
 - 1. Kéo thả dialog vào form
 - ✓ VS sẽ tạo thẻ hiện chứa trong lớp form
 - 2. Khai báo đối tượng và tạo thẻ hiện của lớp CD
 - ✓ VD: OpenFileDialog oFile = new OpenFileDialog();

Common Dialog



OPENFILEDIALOG

- Sử dụng để chọn file lưu trên đĩa
- Cách sử dụng từ ToolBox



OPENFILEDIALOG

- Code phát sinh của VS

```
private void InitializeComponent()
{
    this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
    this.SuspendLayout();
    //
    // openFileDialog1
    //
    this.openFileDialog1.FileName = "openFileDialog1";
    //
    // Form1
    //
    this.AutoScaleMode = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleModeMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(262, 202);
    this.Name = "Form1";
    this.Text = "Form1";
    this.Load += new System.EventHandler(this.Form1_Load);
    this.ResumeLayout(false);
}

#endregion

private System.Windows.Forms.OpenFileDialog openFileDialog1;
```

xyz.Designer.cs

Tạo thẻ hiện

Hiển thị Form

openFileDialog1.ShowDialog();

OPENFILEDIALOG

- Code tự viết – không dùng Toolbox

```
// tạo thẻ hiện OpenFileDialog
 OpenFileDialog OFileDlg = new OpenFileDialog();
// thiết lập các option
// cho phép chọn nhiều file
OFileDlg.Multiselect = true;
// lọc chỉ mở những file C# source code
OFileDlg.Filter = "C# source code|*.cs";

DialogResult result;
// hiển thị và lấy giá trị trả về
result = OFileDlg.ShowDialog();
// xử lý tiếp theo...
```

OPENFILEDIALOG

<i>Thuộc tính, phương thức</i>	
<i>Thuộc tính</i>	
FileName	Lấy tên file được chọn
Filenames	Lấy tên tất cả các file được chọn
Filter	Xác định kiểu file cần mở
InitialDirectory	Thư mục khởi tạo
Multiselect	Cho phép chọn nhiều file
Title	Tiêu đề của dialog
<i>Phương thức</i>	
ShowDialog	Hiển thị dialog
Sự kiện	
FileOk	Xuất hiện khi user kích vào OK

SAVEFILEDIALOG

- Sử dụng để tạo file trên đĩa.
- Cách sử dụng
 - Sử dụng SaveFileDialog component trên Toolbox
 - ✓ Tương tự như OpenFileDialog!
 - Tạo thể hiện của lớp SaveFileDialog

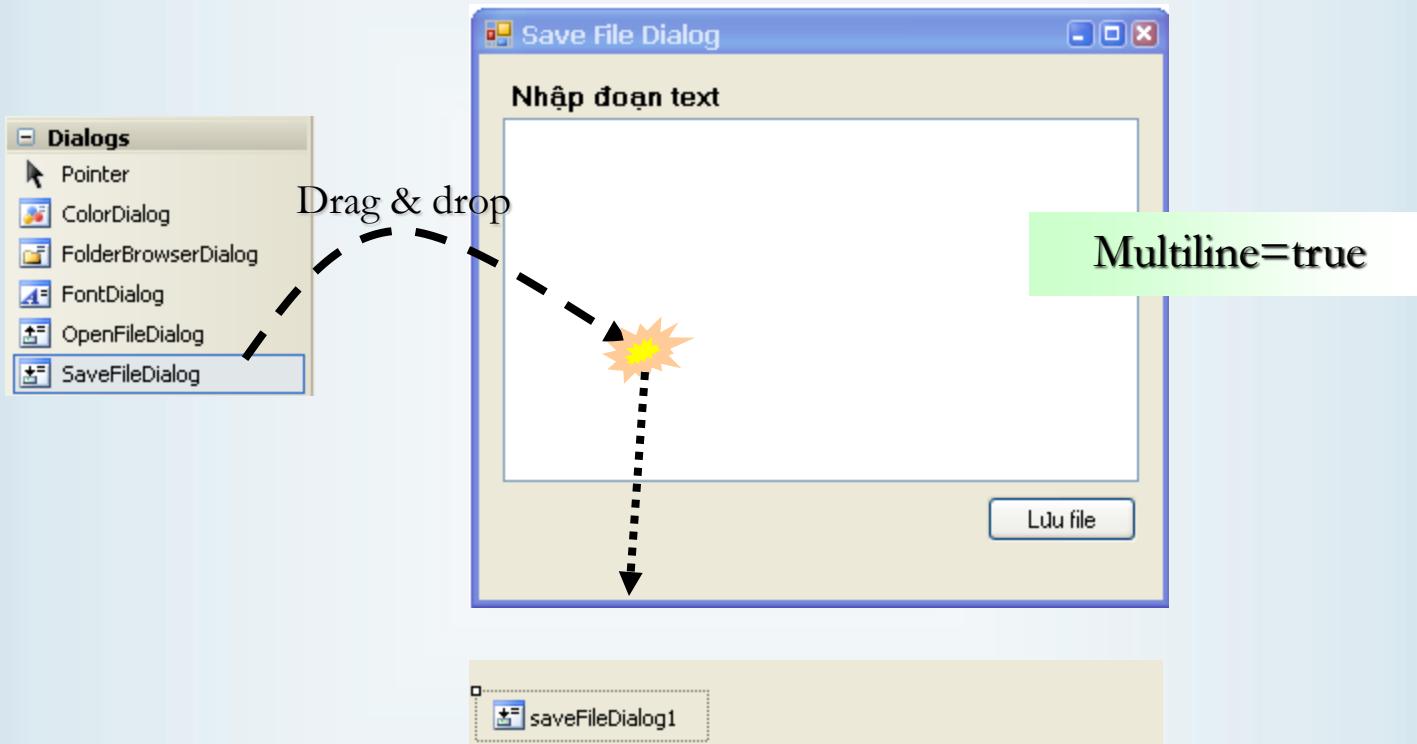
```
// tạo thể hiện Save File Dialog
SaveFileDialog saveDlg = new SaveFileDialog();
// save dưới dạng txt
saveDlg.Filter = "Text File|*.txt";
// hiển thị dialog và lấy giá trị trả về của dialog
DialogResult result = saveDlg.ShowDialog();
// nếu user chọn button "OK"
if (result == DialogResult.OK)
{
    // TO DO...
}
```

SAVEFILEDIALOG

- Demo: nhập văn bản vào textbox, sau đó lưu xuống file *.txt.
 - Tạo ứng dụng Windows Form có các control
 - ✓ 1 label: caption của textbox
 - ✓ 1 textbox: chứa nội dung text do user nhập
 - ✓ 1 button: gọi SaveFileDialog và lưu file
 - ✓ 1 SaveFileDialog: khai báo dialog SaveFile.

SAVEFILEDIALOG

THIẾT KẾ FORM



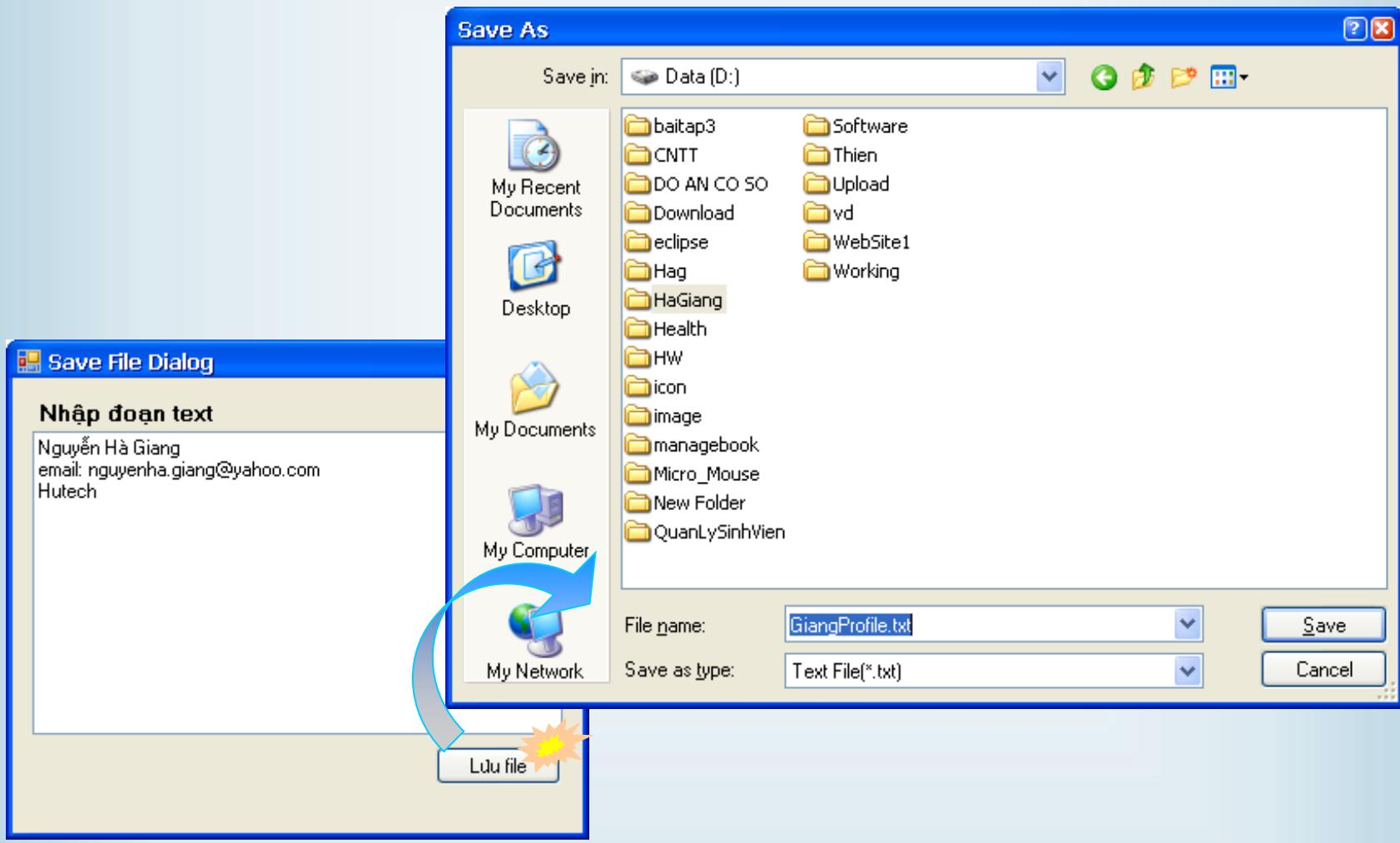
SAVEFILEDIALOG

- Viết phần xử lý cho button “Lưu file”

System.IO

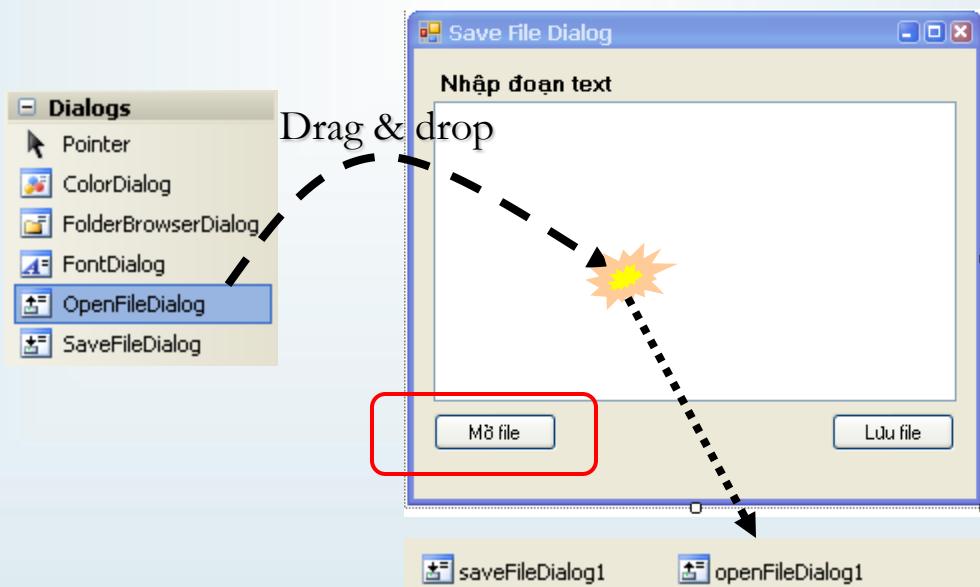
```
21 // phần xử lý cho button lưu
22 // lưu dạng text
23 saveFileDialog1.Filter = "Text File (*.txt) | *.txt";
24
25 if (saveFileDialog1.ShowDialog() == DialogResult.OK)
26 {
27     // sau khi user nhập tên file cần lưu và nhấn "OK"
28
29     // tạo stream để đọc ghi file
30     Stream myStream = saveFileDialog1.OpenFile();
31     // tạo đối tượng ghi vào file
32     StreamWriter writer = new StreamWriter(myStream);
33     // lưu nội dung text của textBox1 vào file
34     writer.WriteLine(textBox1.Text);
35
36     writer.Close();
37     myStream.Close();
38 }
```

SAVEFILEDIALOG



SAVEFILEDIALOG

- Mở rộng bổ sung phần đọc file text từ demo trên
 - Open file *.txt và hiển thị nội dung của file trên TextBox.
- Bổ sung thêm button “Mở file” và kéo thành phần OpenFileDialog thả vào form

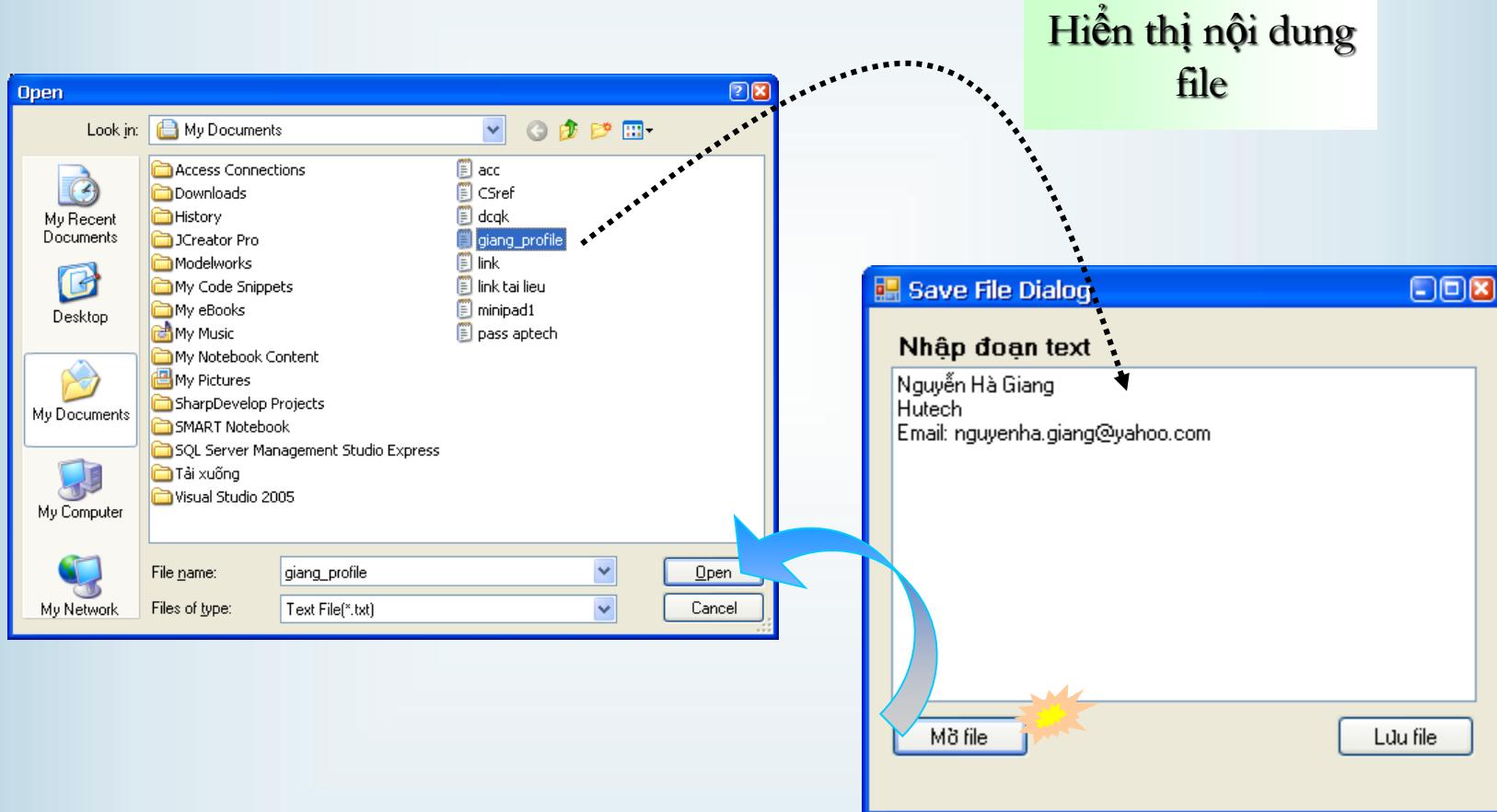


SAVEFILEDIALOG

- Viết phần xử lý button “Mở file”

```
43 // chức năng mở file txt và hiển thị nội dung
44 openFileDialog1.Filter = "Text File (*.txt) | *.txt";
45
46 if (openFileDialog1.ShowDialog() == DialogResult.OK)
47 {
48     // tạo stream đọc/ghi file
49     Stream myStream = openFileDialog1.OpenFile();
50     // tạo đối tượng đọc file
51     StreamReader reader = new StreamReader(myStream);
52     // đọc toàn bộ nội dung vào textbox1
53     textBox1.Text = reader.ReadToEnd();
54
55     reader.Close();
56     myStream.Close();
57 }
```

DEMO SAVEFILEDIALOG



FONTDIALOG

- Chức năng hiển thị hộp thoại chọn font chữ được install trong máy
 - Trong ứng dụng làm việc với document, đồ họa...
- Sử dụng FontDialog
 - Từ ToolBox kéo thả FontDialog vào Form
 - ✓ Sử dụng trong lúc thiết kế
 - Khai báo thể hiện FontDialog và hiển thị
 - ✓ Viết code

FONTDIALOG

Thuộc tính, phương thức thường dùng

Thuộc tính

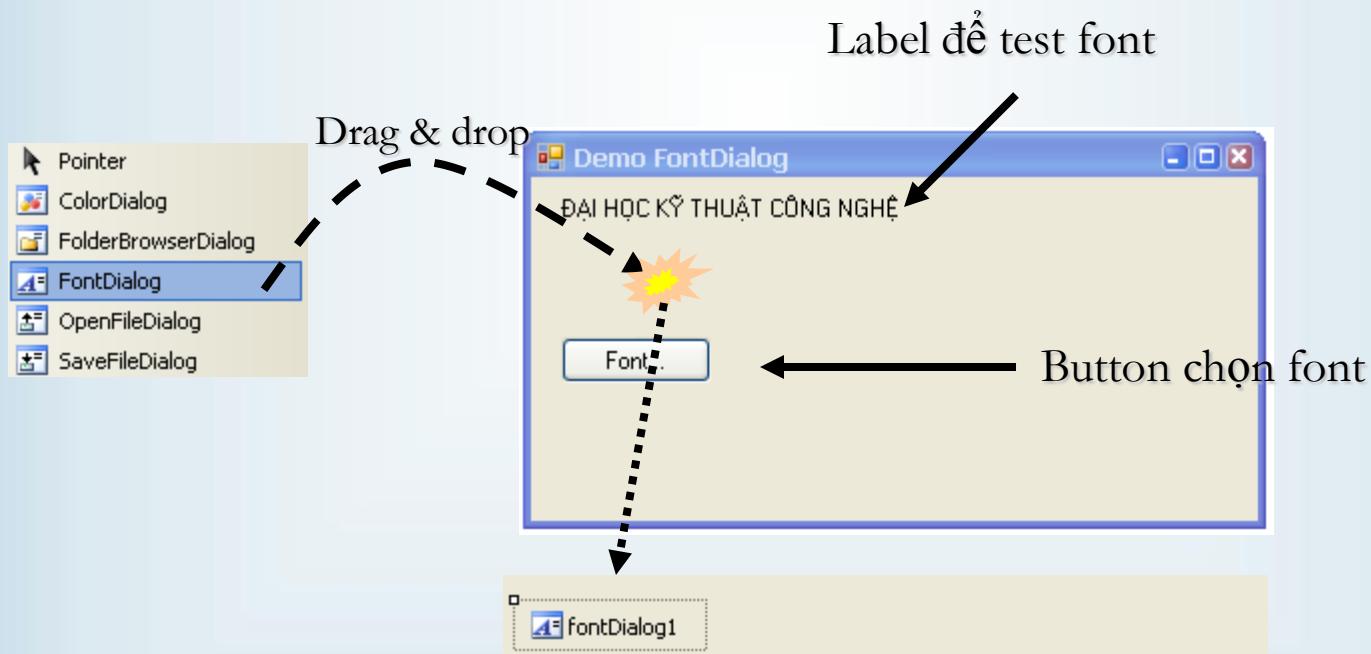
Font	Get/set font được chọn
Color	Get/set màu của font
ShowColor	Hiển thị hay không hiển thị chọn màu
ShowApply	Hiển thị/ không button Apply

Phương thức

ShowDialog	Hiển thị dialog ra màn hình
Sự kiện	
Apply	Kích hoạt khi user chọn apply

FONTDIALOG

- Demo: gọi FontDialog thiết lập font cho control
- Tạo Form có dạng sau

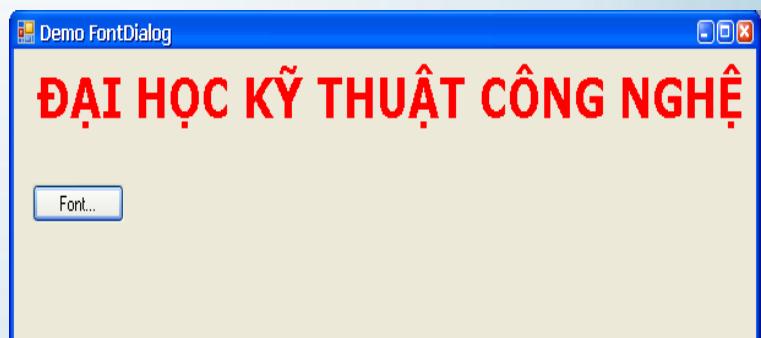
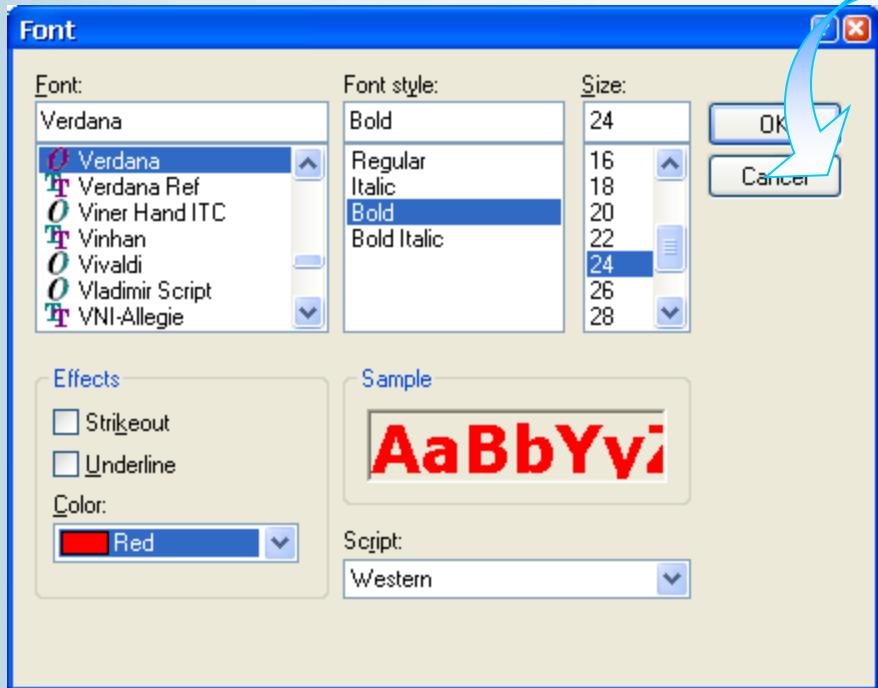


FONTDIALOG

- Phần xử lý cho button “Font”

```
21 // hiển thị chức năng chọn màu
22 fontDialog1.ShowDialog = true;
23 // chọn màu mặc định là màu label
24 fontDialog1.Color = label1.ForeColor;
25 // thiết lập font mặc định là font của label
26 fontDialog1.Font = label1.Font;
27
28 if (fontDialog1.ShowDialog() == DialogResult.OK)
29 {
30     // thiết lập font mới cho label
31     label1.Font = fontDialog1.Font;
32     // thiết lập màu mới cho label
33     label1.ForeColor = fontDialog1.Color;
34 }
```

FONTDIALOG



FONTDIALOG

- Bổ sung chức năng “Apply” của FontDialog
 - Khi FontDialog đang hiển thị cho phép user kích chọn nút Apply ⇒ label thay đổi font theo.
- Cách thực hiện
 - Khai báo có hiện thị button Apply cho FontDialog
 - ✓ `fontDialog1.ShowApply = true`
 - ✓ Đăng ký trình xử lý sự kiện cho button “Apply”
 - Tạo trình xử lý cho sự kiện “Apply” của FontDialog
 - Đăng ký trình xử lý cho sự kiện Apply của FontDialog

FONTDIALOG

- Code minh họa

```
28 // hiển thị button Apply  
29 fontDialog1.ShowApply = true;  
30  
31 // đăng ký sự kiện khi button Apply được nhấn  
32 fontDialog1.Apply += new EventHandler(ChangeFont);
```



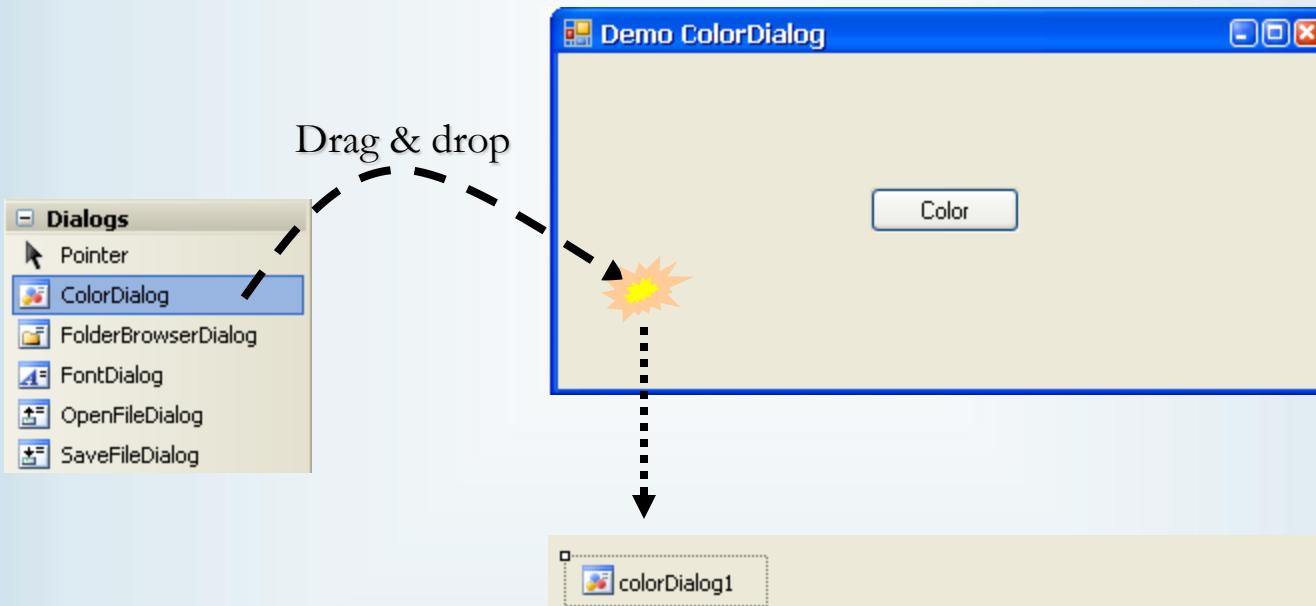
```
46 void ChangeFont(object sender, EventArgs e)  
47 {  
48     // thiết lập font mới cho label  
49     label1.Font = fontDialog1.Font;  
50     // thiết lập màu mới cho label  
51     label1.ForeColor = fontDialog1.Color;  
52 }
```

COLORDIALOG

- Hiển thị chức năng chọn màu
- Sử dụng theo 2 cách
 - Kéo ColorDialog từ Toolbox thả vào form...
 - Viết code tạo đối tượng...
- ColorDialog sử dụng giá trị màu có kiểu Color
 - Màu ARGB
 - ✓ Alpha
 - ✓ Red
 - ✓ Green
 - ✓ Blue

COLORDIALOG

- Demo chọn màu: chọn màu nền cho form



COLORDIALOG

- Phân xử lý cho button “Color”

```
if (colorDialog1.ShowDialog() == DialogResult.OK)
{
    // lấy bề mặt vẽ của form
    Graphics g = this.CreateGraphics();
    // hình chữ nhật của form
    Rectangle rect = new Rectangle(0,0,Width,Height);
    // tạo brush gradient: có màu được chọn -> white
    LinearGradientBrush brush = new LinearGradientBrush(rect,
        colorDialog1.Color, Color.White,
        LinearGradientMode.Horizontal);
    // tô nền của form
    g.FillRectangle(brush, rect);
}
```



MESSAGEBOX

- Hiển thị hộp thoại chứa thông tin chỉ dẫn đến user
- Bao gồm các phần
 - Text
 - Button
 - Symbol
- Lớp MessageBox có phương thức tĩnh Show để hiển thị dialog.
 - Bắt giá trị trả về để biết phản ứng của user

MESSAGEBOX

- Có khoảng 21 phiên bản của Show
- Một số phiên bản thường sử dụng
 - DialogResult Show(string text);
 - DialogResult Show(string text, string caption);
 - DialogResult Show(string text, string caption, MessageBoxButtons button);
 - DialogResult Show(string text, string caption, MessageBoxButtons button, MessageBoxIcon icon);
 - ...

MESSAGEBOX

- Các button hiển thị theo message
 - Cho phép user chọn lựa các phản ứng với message
 - Được định nghĩa trong MessageBoxButtons

OK	MessageBoxButtons
OKCancel	
AbortRetryIgnore	
YesNoCancel	
YesNo	
RetryCancel	

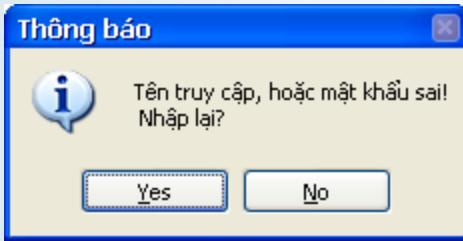
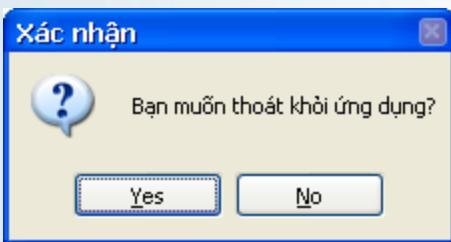
MESSAGEBOX

- MessageBoxIcon: định nghĩa các icon kèm theo message

	MessageBoxIcon.Asterisk MessageBoxIcon.Information
	MessageBoxIcon.Error MessageBoxIcon.Stop MessageBoxIcon.Hand
	MessageBoxIcon.Exclamation MessageBoxIcon.Warning
	MessageBoxIcon.Question
	MessageBoxIcon.None

MESSAGEBOX

- Một số các MessageBox minh họa

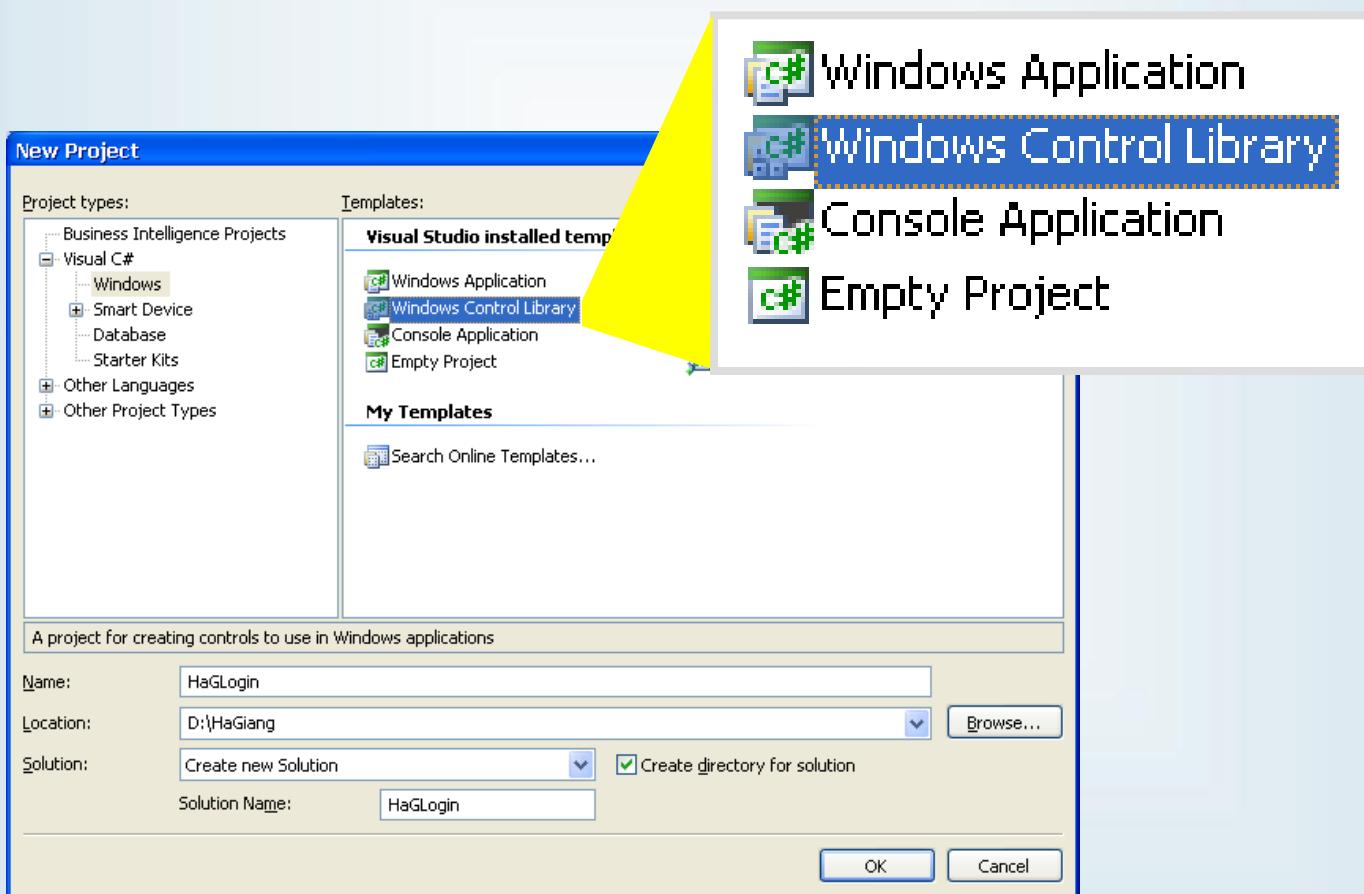


USER CONTROL

- Công việc kéo thả nhóm control thường sử dụng ⇒ nhảm chán
- User control phục vụ mục đích tái sử dụng
- Vấn đề:
 - Nhóm control phục vụ cho **việc login** thường xuất hiện trong ứng dụng
- Yêu cầu:
 - Tạo nhóm control phục vụ cho việc login
 - Nhóm control này bao gồm:
 - ✓ 2 label
 - ✓ 2 textbox

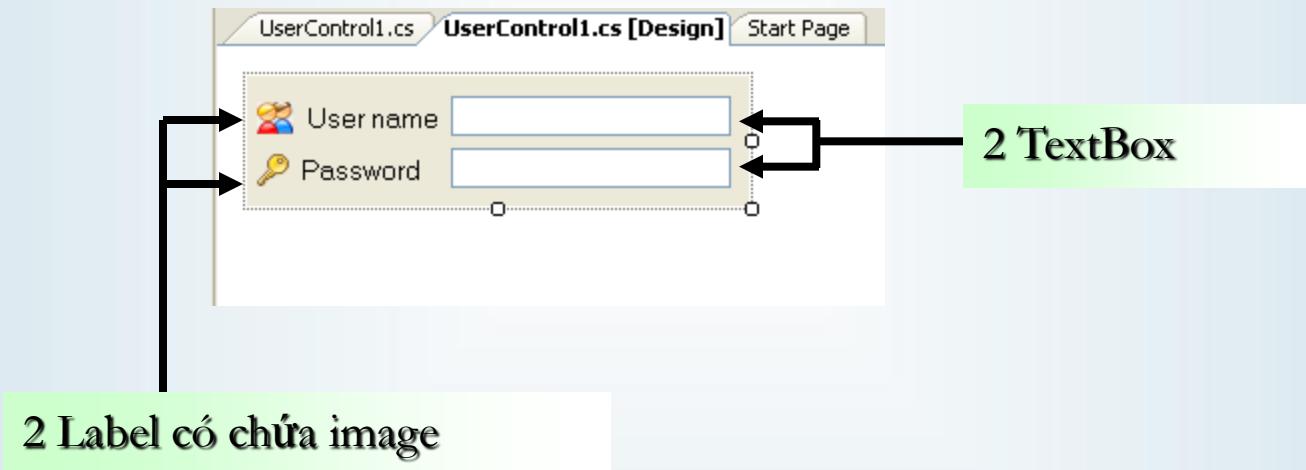
USER CONTROL

- Bước 1: Tạo ứng dụng Windows Control Library



USER CONTROL

- Bước 2: thiết kế user control như sau



USER CONTROL

- Khai báo các property cho user control:
 - Mỗi property gắn với textBox:
 - ✓ UserName gắn với txtUID;
 - ✓ Password gắn với txtPwd.

```
[Category("Data"), Description("User Name")]
public string UserName
{
    get { return txtUID.Text; }
    set { txtUID.Text = value; }
}
[Category("Data"), Description("Password")]
public string Password
{
    get { return txtPwd.Text; }
    set { txtPwd.Text = value; }
}
```

USER CONTROL

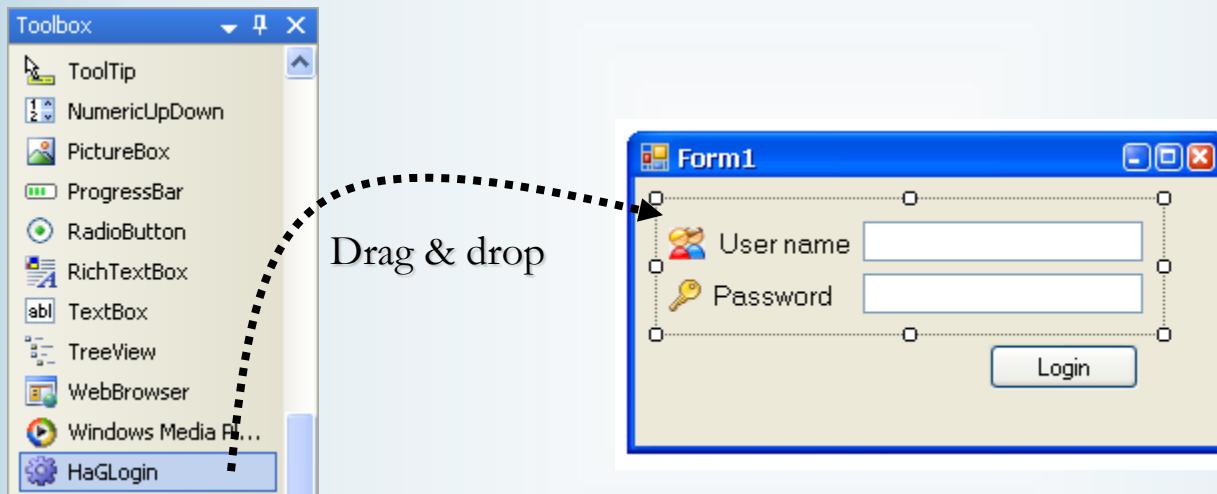
- Biên dịch user control
- Kết quả được file assembly có phần mở rộng là DLL
- Sử dụng user control trong client app
 - Add vào phần References của project
 - ✓ Tạo đối tượng user control, add vào form...
 - Add vào ToolBox | Choose Items...
 - ✓ Kéo user control thả vào form, code phát sinh tự động

USER CONTROL

- Tạo ứng dụng Test User control
 - Tạo ứng dụng Windows Form
 - Thêm User Control vào ToolBox
 - ✓ Kích chuột phải vào ToolBox
 - ✓ Chọn chức năng Choose Items...
 - ✓ Chọn file DLL của User control vừa tạo
 - Kéo user control thả vào form

USER CONTROL

- Kéo thả user control vào form



Phát sinh code trong
designer.cs

```
private HAGLogin.HaGLogin haGLogin1;  
  
private void InitializeComponent()  
{  
    this.haGLogin1 = new HAGLogin.HaGLogin();  
    this.SuspendLayout();
```

USER CONTROL

- Sử dụng User Control như control bình thường trên form.
- Truy cập user control HAGLogin thông qua 2 property đã định nghĩa khi xây dựng control này
 - UserName: là textBox User Name
 - Password: là nội dung của textBox Password.

