



# **LẬP TRÌNH C# 3**

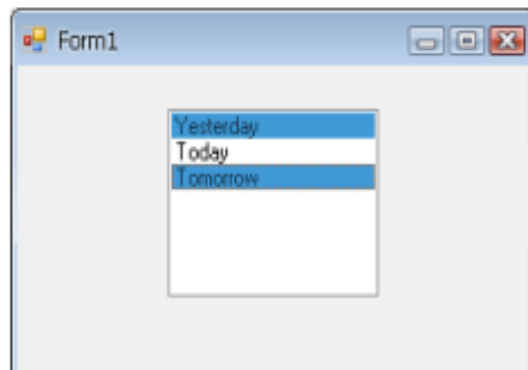
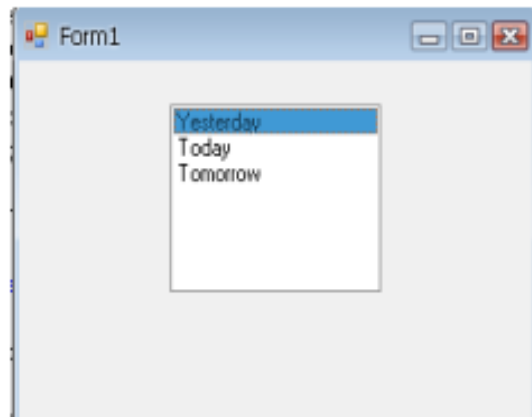
## **BÀI 2: COLLECTION CONTROLS**

- ⊙ ComboBox, ListBox
- ⊙ GroupBox & Panel, TabControl
- ⊙ Menu Strip, Tool Strip
- ⊙ Listview
- ⊙ Truyền dữ liệu giữa các form



- ❑ Liệt kê danh sách phần tử cho phép người dùng chọn một hay nhiều
- ❑ Có các thuộc tính chung quan trọng sau:
  - ❖ DataSource: chọn tập dữ liệu điền vào điều khiển (VD string [] , ArrayList là tập dữ liệu đưa vào)
  - ❖ SelectedText, SelectValue, SelectedItem, SelectedIndex để lấy giá trị hay đối tượng chọn
- ❑ Thuộc tính riêng
  - ❖ Combobox: DropDownStyle ( DropDown: cho phép nhập thêm chuỗi ; DropDownList: chỉ cho phép chọn chuỗi/item đã có)
  - ❖ ListBox: SelectionMode (True: cho phép chọn nhiều)

- Listbox cho phép ta tạo ra một danh sách các phần tử để lựa chọn



DisplayMember	(none)
Items	(Collection)
Tag	

DropDownStyle	DropDown
FlatStyle	Simple
Font	DropDown
ForeColor	DropDownList
RightToLeft	No
Text	
UseWaitCursor	False

SelectionMode	MultiSimple
Sorted	None
TabIndex	One
TabStop	MultiSimple
UseTabStops	MultiExtended
Visible	TRUE

## SelectionMode

Indicates if the list box is to be single-select, multi-select, or not selectable.

```
private void Form1_Load
(object sender, EventArgs e)
{
    //Code Here
    string[] data = ( "Yesterday", "Today", "Tomorrow" );
    listBox1.DataSource = data;
}
```

## Properties

- DisplayMember
- Items
- SelectionMode
- SelectedIndex

## Method

- ClearSelected
- GetItemText
- GetSelected
- SetSelected

## Event

- SelectedIndexChanged
- SelectedValueChanged
- ValueMemberChanged

```

ListBox lstCountry = new ListBox();
Boolean flag;
lstCountry.Name = "lstCountry";
lstCountry.Items.Add("U.S");
lstCountry.Items.Add("U.A.E.");
lstCountry.Items.Add("Japan");
lstCountry.Items.Add("China");
lstCountry.Items.Add("Russia");
lstCountry.SelectionMode = SelectionMode.MultiExtended;
flag=lstCountry.GetSelected(2);

private void lstCountry_SelectedIndexChanged(object sender, EventArgs e)
{
    MessageBox.Show(lstCountry.SelectedItem.
        ToString());
}

```

- ❑ Thuộc tính Items cho phép thêm item vào ListBox

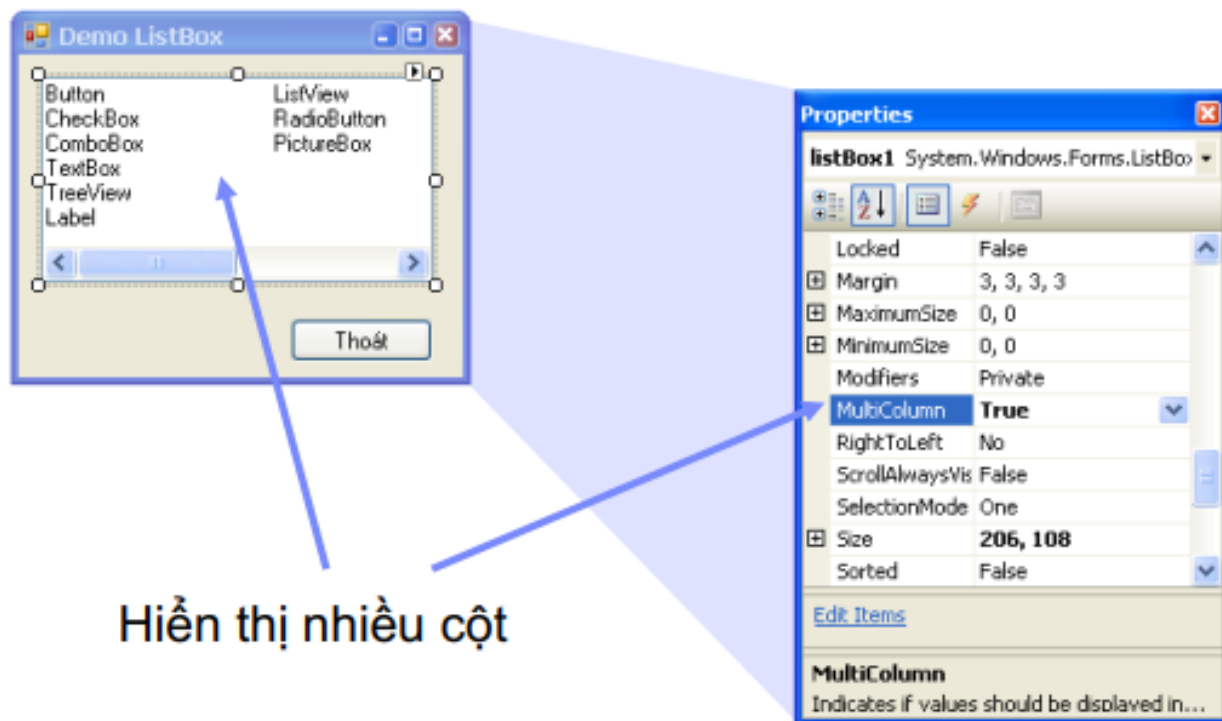
The diagram illustrates the process of adding items to a ListBox. It shows three windows:

- Demo ListBox**: A form with a ListBox containing the items: Button, CheckBox, ComboBox, and TextBox. A button labeled "Thoát" is at the bottom.
- Properties**: The Properties window for **listBox1** (System.Windows.Forms.ListBox). The **Items** property is highlighted, showing it is a **(Collection)**. Below the list, there is an **Edit Items** link.
- String Collection Editor**: A dialog box titled "String Collection Editor" with the text "Enter the strings in the collection (one per line):". It contains a list of items: Button, CheckBox, ComboBox, and TextBox. There are OK and Cancel buttons at the bottom.

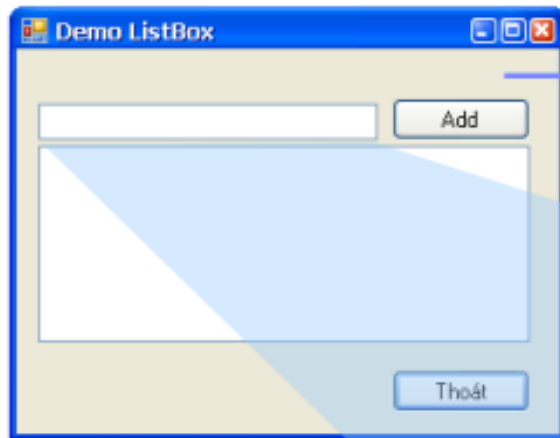
Annotations:

- Danh sách item**: Points to the **Items** property in the Properties window.
- Cho phép thêm item trong màn hình thiết kế form**: Points to the String Collection Editor dialog.

## ❑ ListBox hiển thị dạng Multi Column



## ❑ Sự kiện SelectedIndexChanged



Mỗi khi kích chọn vào item trong listbox  $\Rightarrow$  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);
}
```



❑ Là điều khiển chứa danh sách các Item cho người dùng lựa chọn

### Properties

- DropDownStyle
- Items
- MaxDropDownItem
- SelectedItem
- SelectedIndex
- Text
- ValueMember

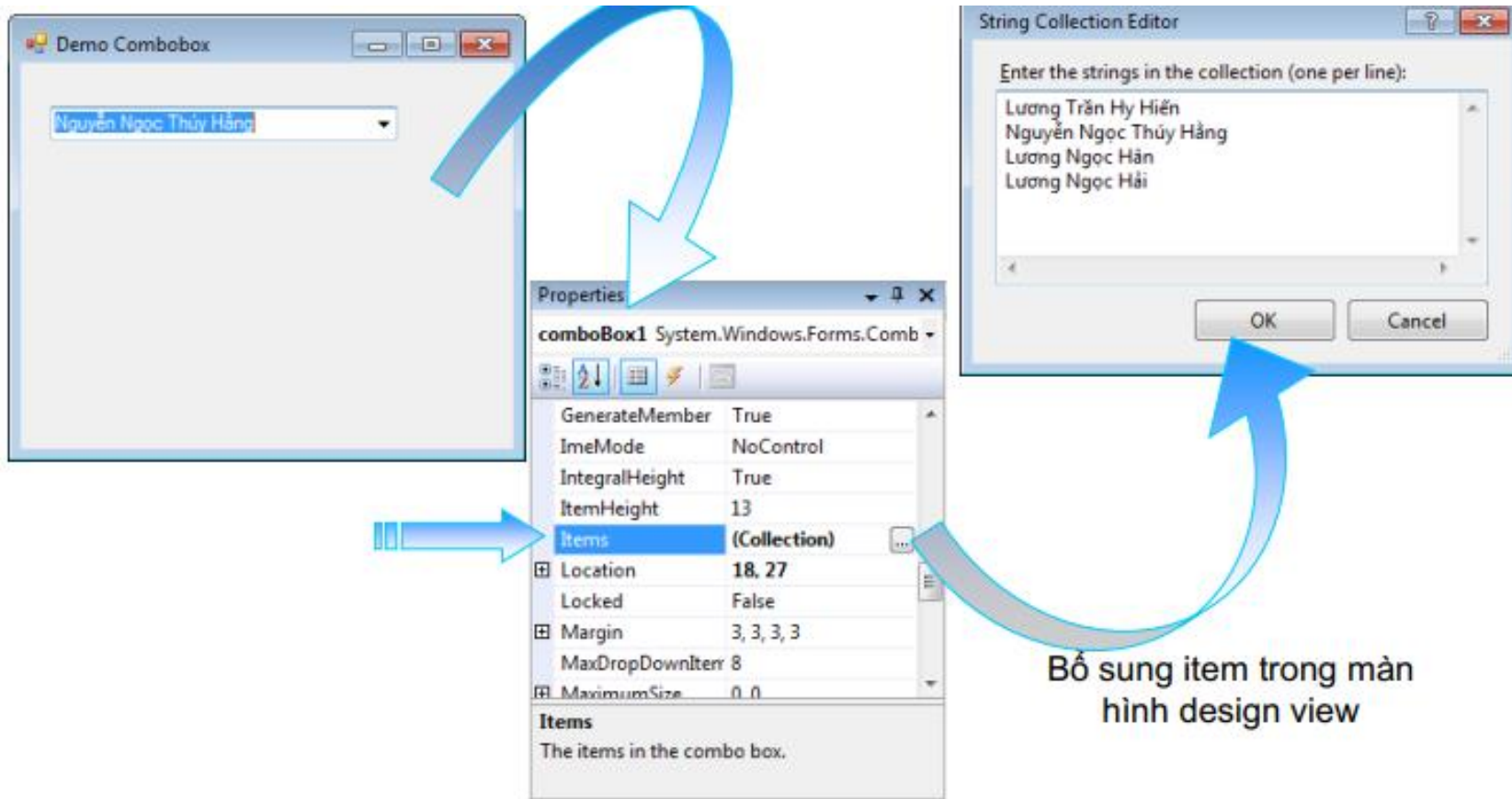
### Method

- GetItemText
- SelectAll
- Select ( int start , int length)

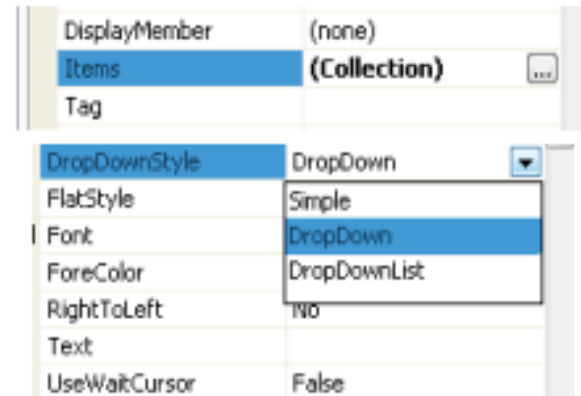
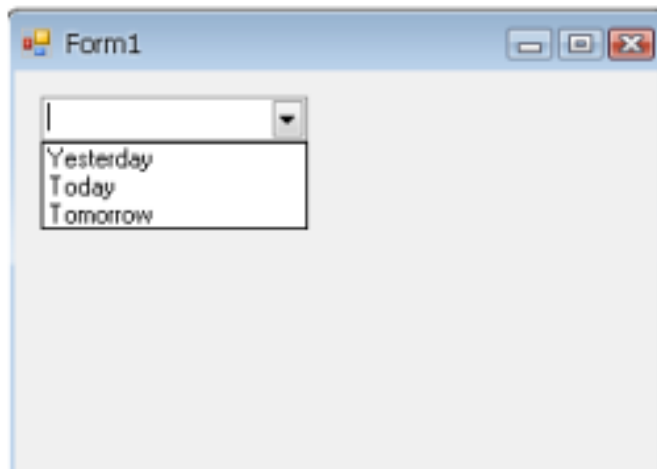
### Event

- DropDown
- SelectedIndex Changed
- SelectedValue Changed
- ValueMember Changed

## ❑ Thêm item từ design view

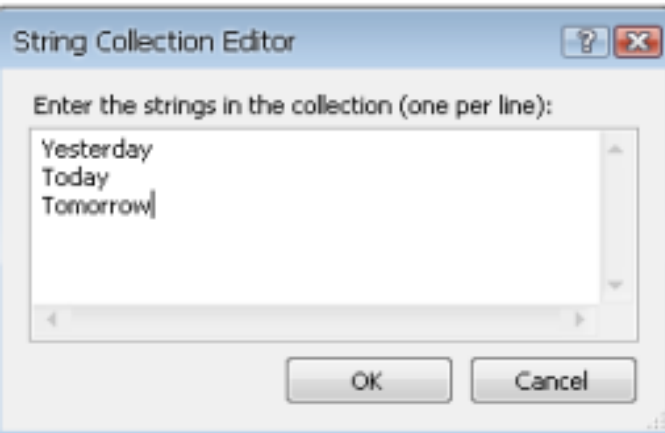


- ❑ Combobox: dùng datasource để thêm và selectedindexchanged xác định item được chọn

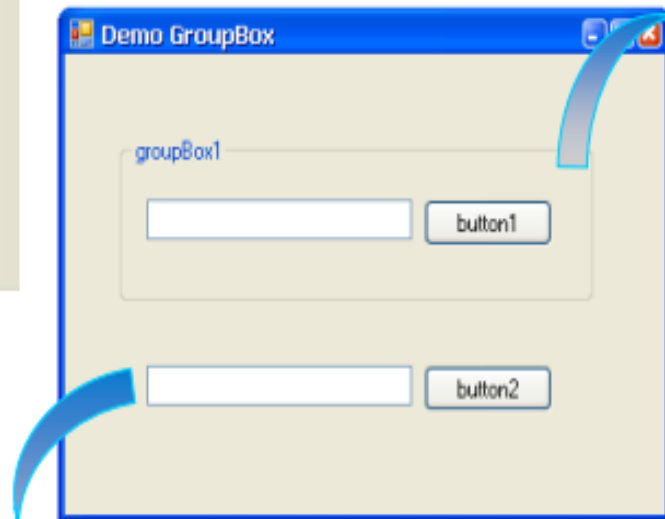
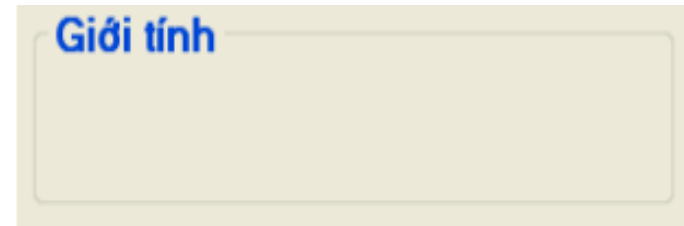
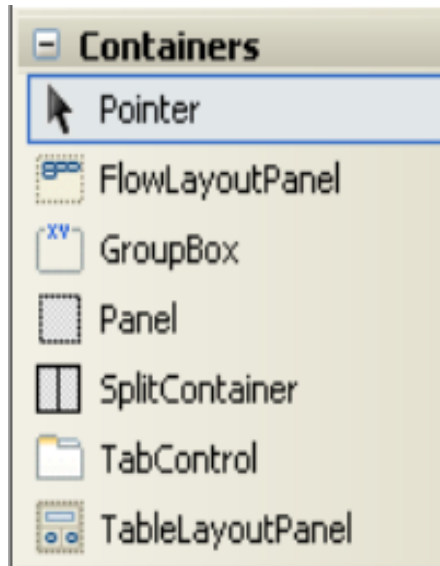


```
private void Form1_Load
(object sender, EventArgs e)
{
    //Code Here
    string[] data = { "Yesterday", "Today", "Tomorrow" };
    comboBox1.DataSource = data;
}
```

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    MessageBox.Show(comboBox1.SelectedItem.ToString());
}
```



## Minh họa GroupBox



groupBox1 chứa 2 control  
textBox1 và button1

textBox2 và button2 chứa trong  
Controls của Form

## ❑ Bố trí controls trên GUI

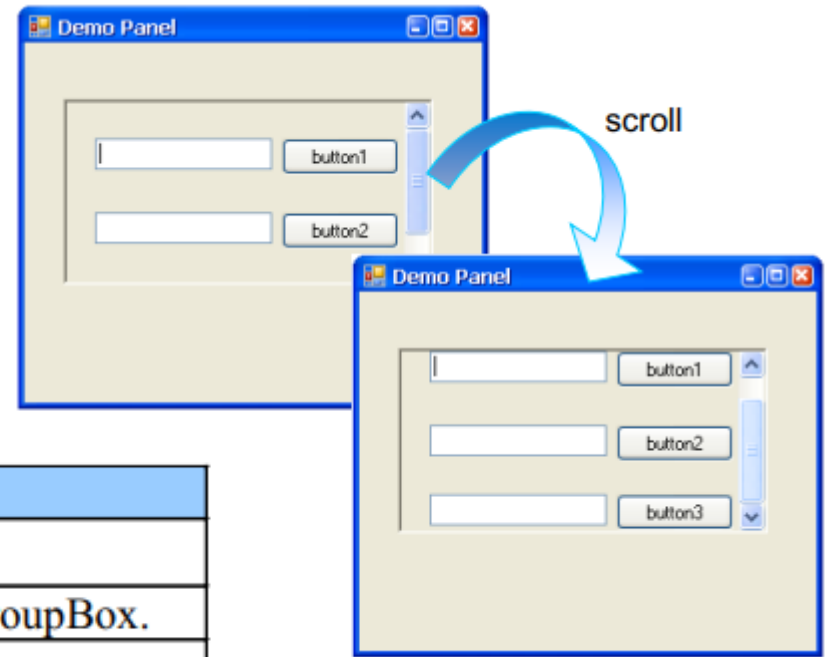
### ❑ GroupBox

- ❖ 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
- ❖ Thiết lập giá trị của GroupBox sẽ ảnh hưởng đến các control mà nó chứa

### ❑ Panel

- ❖ Chứa nhóm các control
- ❖ Không có caption
- ❖ Có thanh cuộn (scrollbar)

- ❑ Minh họa Panel
- ❑ Thuộc tính thông dụng



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

## ❑ Ví dụ panel

```
Panel pnlCustomerDetails = new Panel();  
pnlCustomerDetails.Name = "pnlCustomerDetails";  
pnlCustomerDetails.BorderStyle = BorderStyle.FixedSingle;  
  
private void pnlCustomerDetails_StyleChanged(object  
sender, EventArgs e)  
{  
    MessageBox.Show("Style Changed");  
}
```

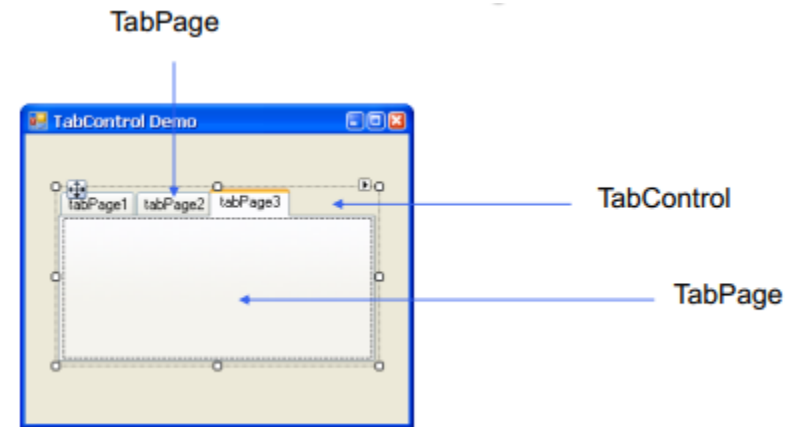
- ❑ Dạng container chứa các control khác
- ❑ Cho phép thể hiện nhiều page trên một form duy nhất
- ❑ Mỗi page chứa các control tương tự như group control khác
  - ❖ Mỗi page có tag chứa tên của page
  - ❖ Kích vào các tag để chuyển qua lại giữa các page
- ❑ Ý nghĩa
  - ❖ Cho phép thể hiện nhiều control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



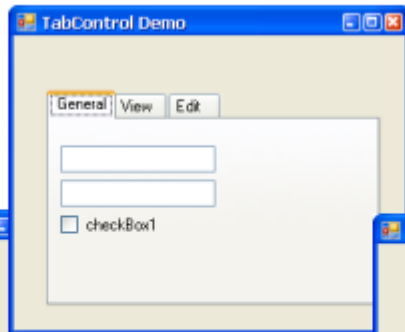
❑ TabControl có thuộc tính TabPages

❖ Chứa các đối tượng tabPage

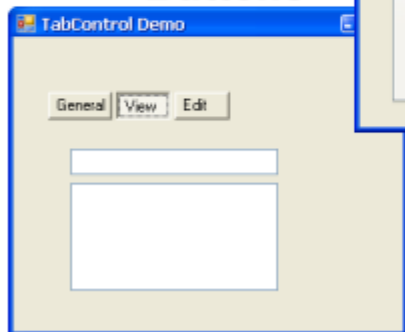
❑ Thuộc tính Appearance



**Normal**



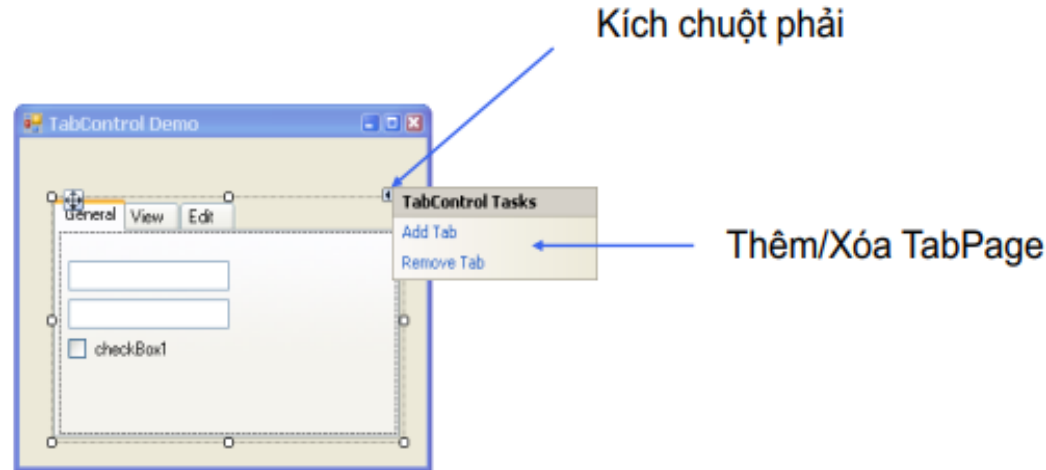
**Buttons**



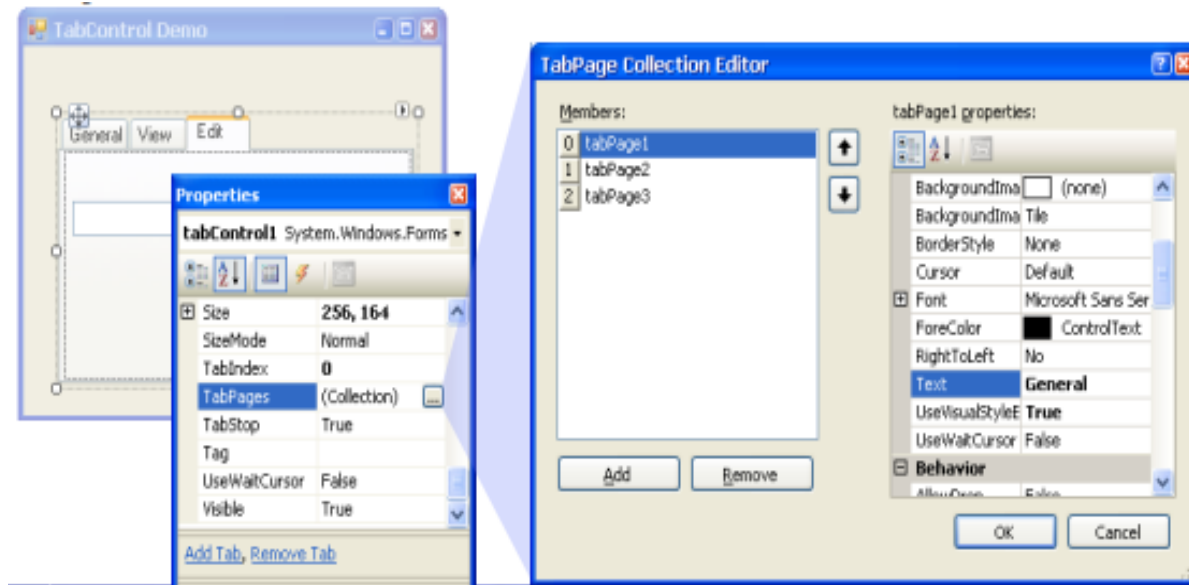
**FlatButton**



## Thêm/Xóa TabPage

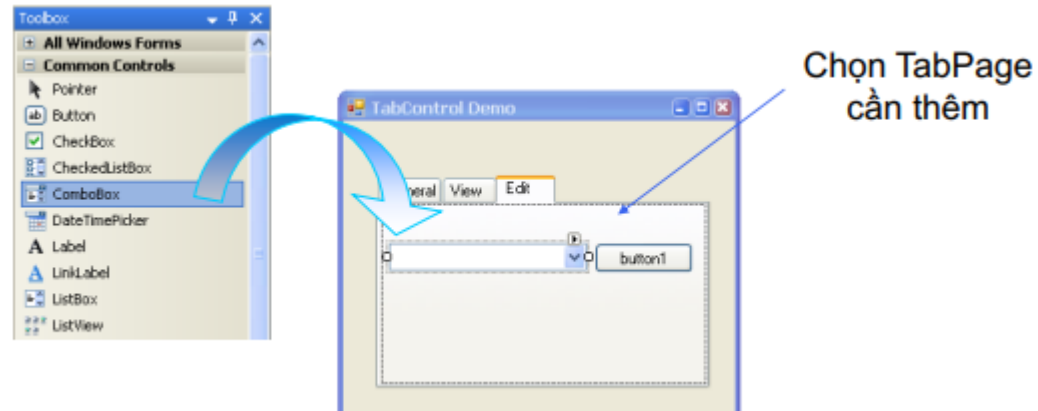


## Chỉnh sửa các TabPage



## ❑ Bổ sung Control vào TabControl

- ❖ Chọn TabPage cần thêm control Chọn TabPage cần thêm control



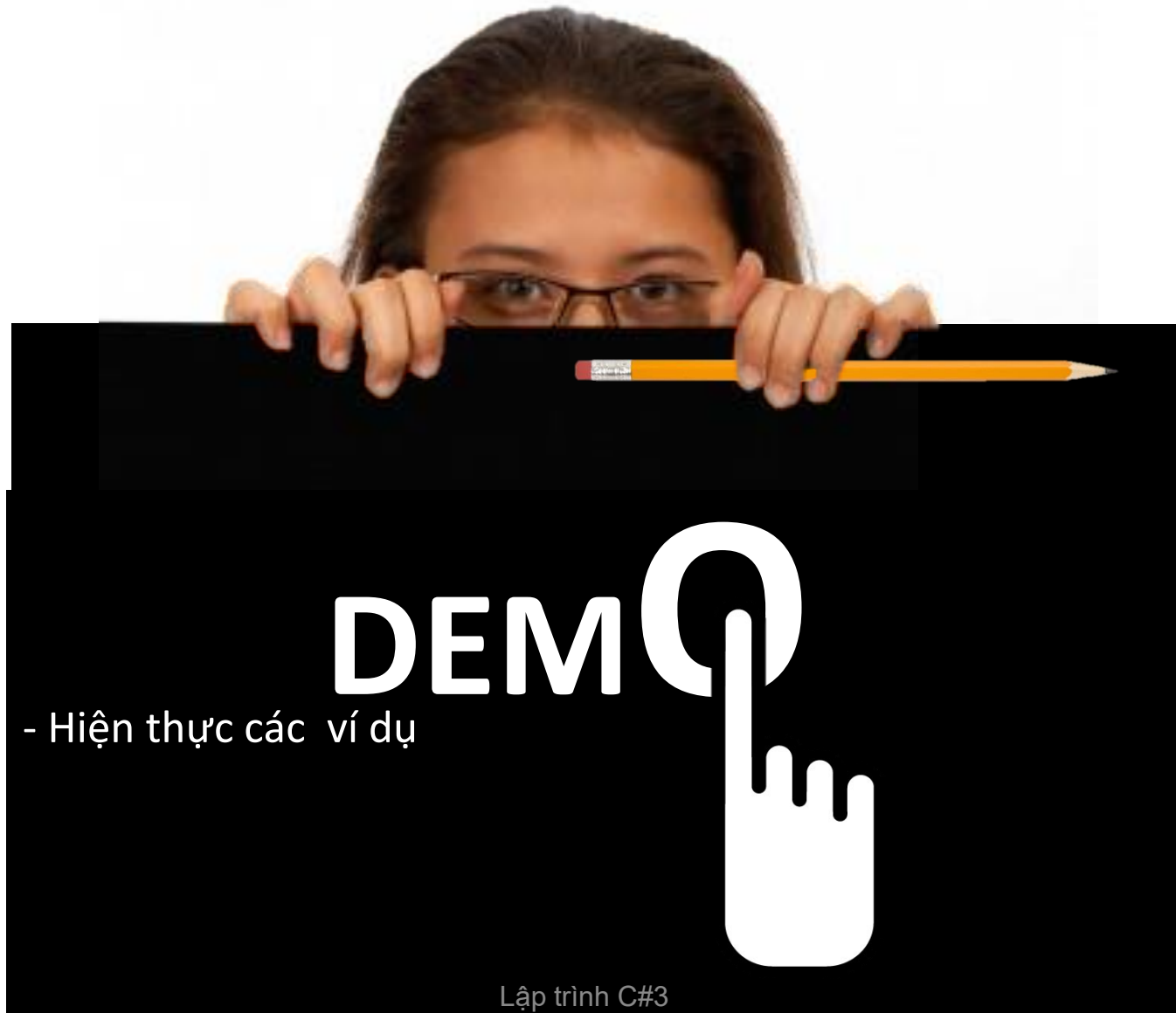
- ❖ Sử dụng code để thêm các TabPage vào TabControl

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

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

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

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



# DEMO

- Hiện thực các ví dụ



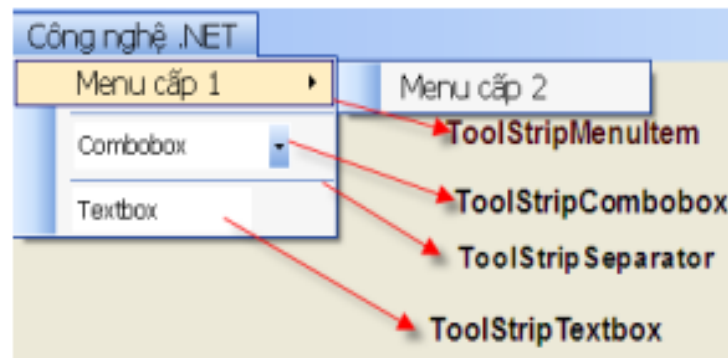
Lập trình C#3



# **LẬP TRÌNH C# 3**

## **BÀI 2: ADVANCED COMPONENTS WINDOW FORM (P2)**

- ❑ Điều khiển MenuStrip cho phép thiết kế hệ thống menu trên Form (menu một cấp hay nhiều cấp). Ví dụ hệ thống menu của chương trình Word, Visual Studio 2017
- ❑ MenuStrip cho phép thiết kế menu với các điều khiển: ToolStripSeparator (Gạch phân cách), ToolStripMenuItem (Menu con), ToolStripComboBox (Combobox), ToolStripTextbox (Textbox)



## ❑ Một số thuộc tính thường dùng của MenuStrip

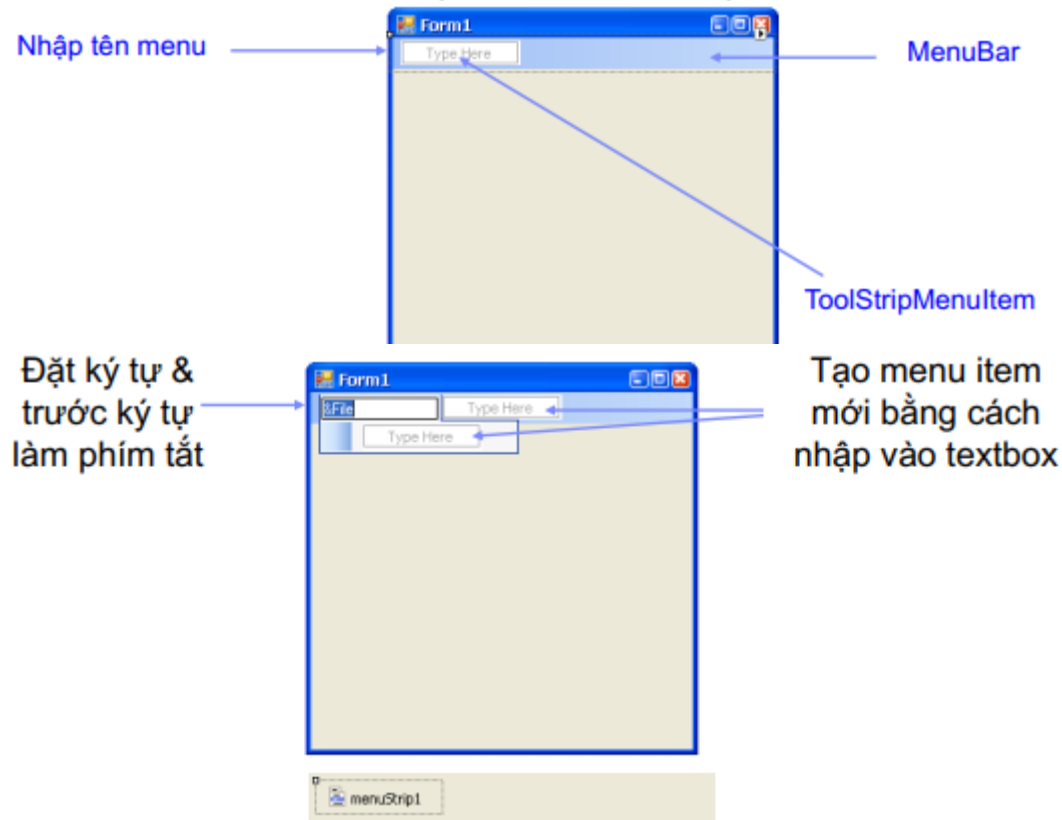
Tên	Ý nghĩa
TextDirection	Chọn hình thức trình bày Menu (quay ngược, quay 90° ...)
Items	Thêm các menu con, kiểu của menu (Menu con, Textbox, Combobox, gạch phân cách). Thông qua giao diện đồ họa bạn có thể thêm các menu vào (tập các Items này là một Colleciton).
RightToLeft	Nhận một trong hai giá trị Yes hay No <ul style="list-style-type: none"> <li>Yes: trình bày menu từ phải qua trái</li> <li>No: trình bày menu từ trái qua phải</li> </ul>

## ❑ Một số thuộc tính ToolStripMenuItem

Checked	Xác định trạng thái check của menu item
Index	Chỉ mục menu item trong menu cha
DropDownItems	Chứa những menu item con
ShortcutKeys	Phím tắt
Text	Tiêu đề menu item
ShowShortcutKeys	Xác định trạng thái hiện thị phím tắt bên cạnh menu item

## □ Cách tạo menu

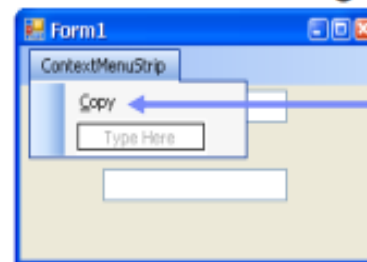
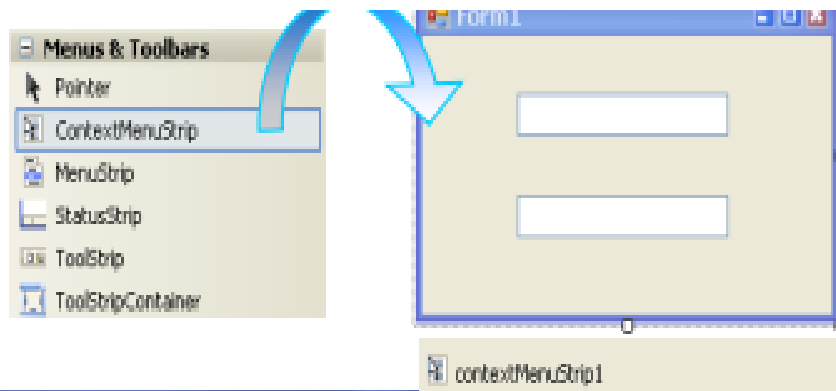
- ❖ Trong ToolBox kéo control MenuStrip thả vào form
- ❖ Thanh menuBar xuất hiện trên cùng của form
- ❖ Trong màn hình design dễ dàng tạo lập các menu item





## Context Menu

- ❖ Xuất hiện khi user kích chuột phải
- ❖ Thông thường menu này xuất hiện tùy thuộc vào đối tượng trong vùng kích chuột phải
- ❖ Mỗi control đều có property là ContextMenuStrip
- ❖ Kích đúp vào menu item của Context Menu để tạo
- ❖ Trong ToolBox kéo ContextMenuStrip thả vào form



Soạn thảo Context Menu  
tương tự như Menu bình  
thường

## ❑ Khai báo trình xử lý sự kiện Click cho menu item

The image shows a Visual Studio IDE with a Windows Form named 'Form1'. A 'ContextMenuStrip' is attached to the form, containing a menu item '&Remove'. The 'Properties' window for the 'mnuRemove' menu item is open, and the 'Click' event is selected, with the handler 'mnuRemove\_Click' assigned. A blue arrow points from the 'Click' event in the Properties window to a code block containing the implementation of the 'mnuRemove\_Click' method.

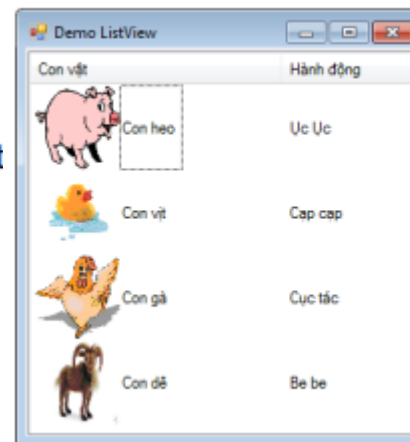
```
private void mnuRemove_Click(object sender, EventArgs e)
{
    // xác định item được chọn trong listbox
    if (listBox1.SelectedIndex >= 0)
    {
        listBox1.Items.Remove(listBox1.SelectedItem);
    }
}
```

- ❑ Dạng control phổ biến hiện thị một danh sách item, các item có thể có các item con gọi là subitem.
- ❑ Có nhiều cách hiển thị: Xem dạng chi tiết thông tin, Xem dạng icon nhỏ, Xem dạng icon lớn, Xem dạng tóm tắt...



### Large Icons

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

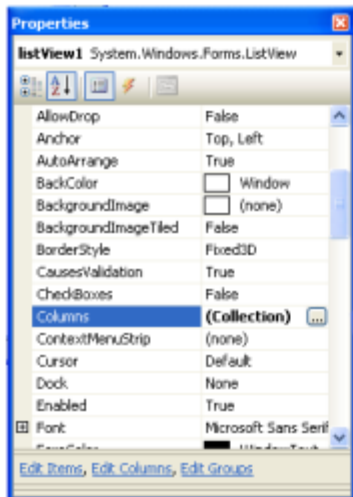


### Detail

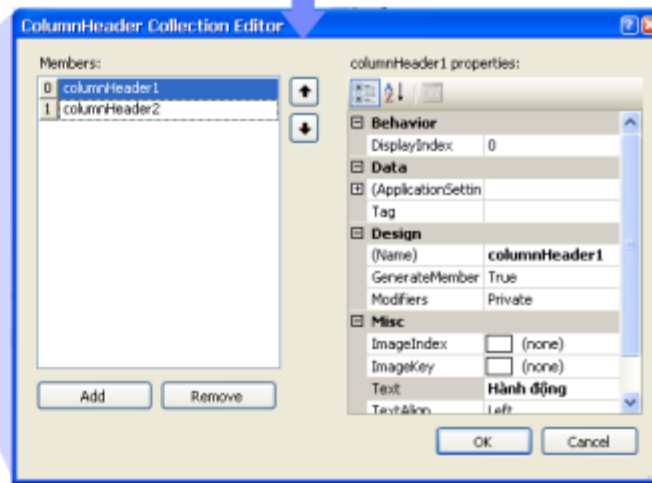
Mỗi item xuất hiện trên một dòng, mỗi dòng có các cột chứa thông tin chi tiết

## ❑ Tạo các cột cho ListView – Details theo cách:

- ❖ Cửa sổ properties ▪ Columns để tạo
- ❖ Sử dụng code trong chương trình



Dialog soạn thảo cột



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

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

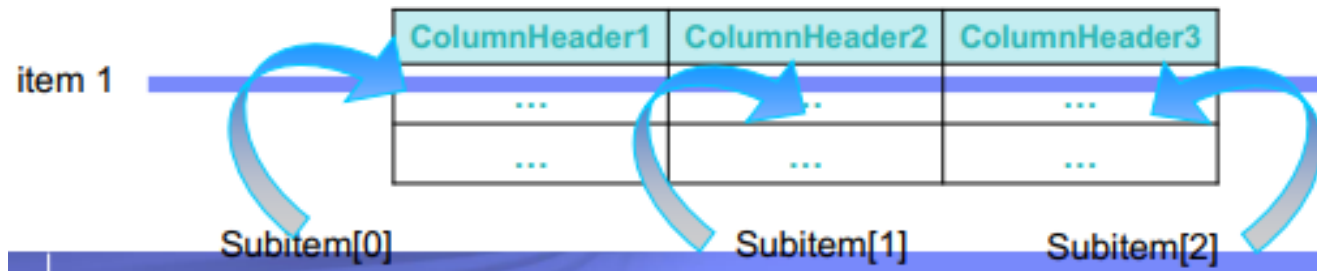
```
ListView1.Columns.Add(columnHeader1);  
ListView1.Columns.Add(columnHeader2);  
ListView1.Columns.Add(columnHeader3);
```

## ❑ Thêm các item vào ListView theo cách:

- ❖ 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



## ❑ Minh họa thêm item qua code

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

```
item1.Text = "HCMUP";
subitem1.Text = "280 An Duong Vuong, HCMC";
```

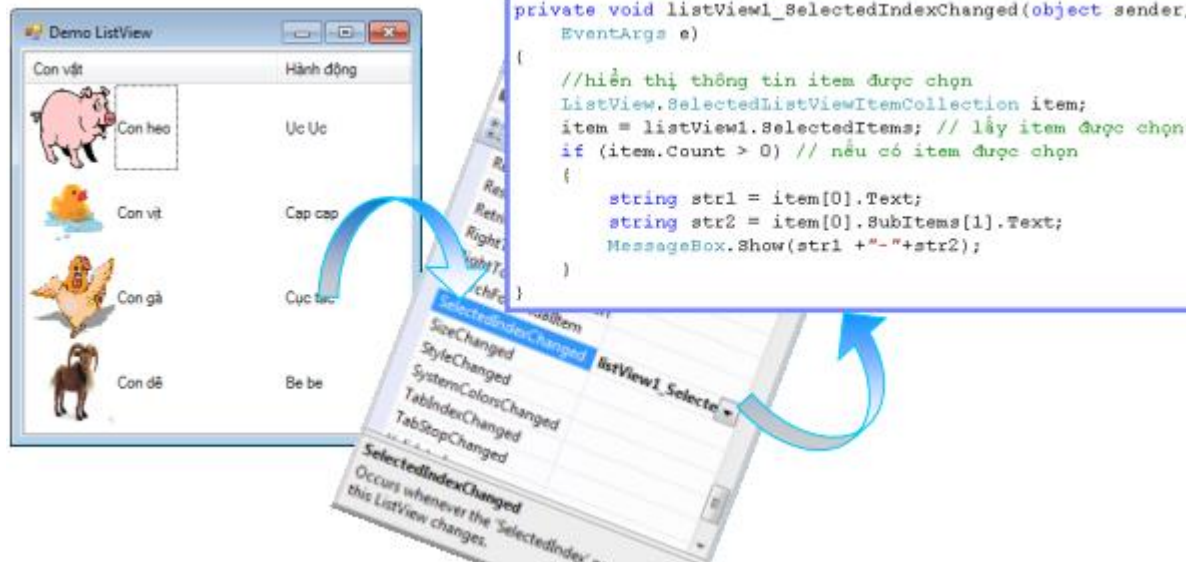
```
item1.SubItems.Add(subitem1);
```

Thêm subitem vào item

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

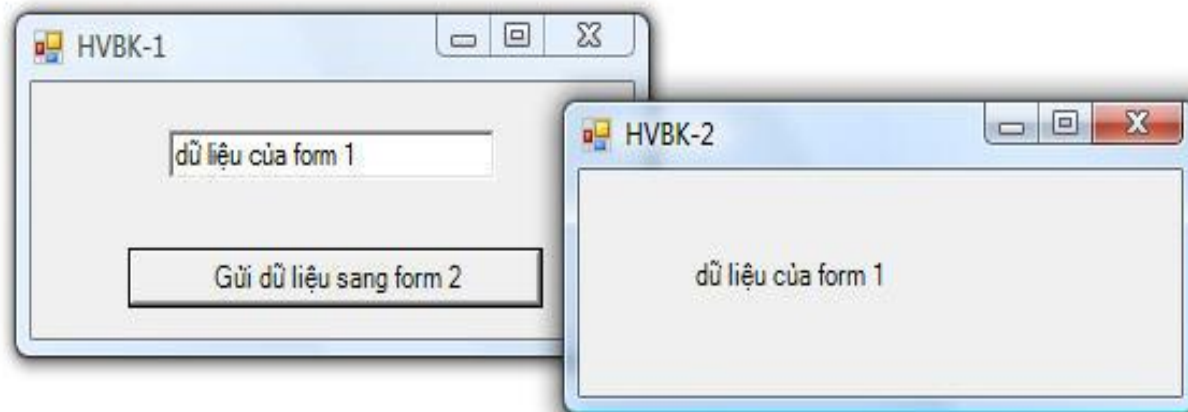
Thêm item vào danh sách items của ListView

## ❑ Sự kiện SelectedIndexChanged



```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    //hiển thị thông tin item được chọn
    ListViewItemCollection 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);
    }
}
```

- ❑ Với ứng dụng WinForms, trong một số trường hợp chúng ta cần phải trao đổi dữ liệu từ form này sang form khác.
- ❑ Có bốn phương pháp cơ bản để passing data giữa các form: Sử dụng constructor, objects, properties, delegates (delegates được đánh giá cao nhất vì nó không phụ thuộc vào quá trình khởi tạo đối tượng, sự tồn tại của đối tượng)



- ❑ Phương Pháp Constructor: Với phương pháp này ta chỉ cần thêm một tham biến vào hàm dựng của form2.

```
public Form2(string strTextBox)
{
    InitializeComponent();
    label1.Text=strTextBox;//Gán dữ liệu nhận được vào Label để thể hiện
}
```

- ❑ Tại button "Gửi dữ liệu sang form 2" ta tiến hành khởi tạo Form2 và Form2 sẽ hiện ra mỗi khi ta click vào nút này

```
private void button1_Click(object sender, System.EventArgs e)
{
    Form2 frm=new Form2(textBox1.Text);
    frm.Show();
}
```



Bước 1:

☐ Phương Pháp Object  
Tại Form1 ta tiến hành thay đổi access modifier cho textbox từ private thành public.

```
public class Form1 : System.Windows.Forms.Form
{
    public System.Windows.Forms.TextBox textBox1;
    .....
}
```

Tại Form2 ta tiến hành khai báo một biến frm1 có kiểu Form1

```
public class Form2 : System.Windows.Forms.Form
{
    private System.Windows.Forms.Label label1;
    public Form1 frm1;
    .....
}
```

Bước 2: Tại Form1 ta tiến hành code hàm sự kiện Clicked của nút button "Gửi dữ liệu sang form 2"

```
private void btnSend_Click(object sender, System.EventArgs e)
{
    Form2 frm= new Form2();
    frm.frm1=this; //Gán form1 cho biến frm1 có kiểu Form1 của Form2
    frm.Show(); //Hiện Form2
}
```

Bước 3: Trong hàm Load của Form2 ta tiến hành gán giá trị của ô textbox1 của form1 cho Label của form2

```
private void Form2_Load(object sender, System.EventArgs e)
{
    label1.Text=((Form1)frm1).textBox1.Text;
}
```

- ❑ Sử dụng Properties: trong phương pháp này ta add thêm thuộc tính (Properties) cho mỗi form
- ❑ Ở Form1 ta add thêm một thuộc tính, để nhận lại giá trị từ ô textbox
- ❑ Ở Form2 ta add thêm một thuộc tính, để gán giá trị cho Label
- ❑ - Ở Form1, trong sự kiện button click ta tiến hành khởi tạo và hiển thị Form2. Giá trị của ô textbox trong Form1 sẽ được chuyển sang cho Form2 thông qua thuộc tính, và gán vào Label của Form2

```
public string _textBox1
{
    get{return textBox1.Text;}
}
```

```
private void button1_Click(object sender, System.EventArgs e)
{
    Form2 frm=new Form2();
    frm.textBox=_textBox1; //gán giá trị của thuộc tính _textBox1 của
    Form1 cho thuộc tính _textbox của Form2
    frm.Show(); // Hiển thị Form2
}
```

```
public string _textBox
{
    set{label1.Text=value;}
}
```

## □ Sử dụng Delegates

Bước 1: Ở Form1 ta tiến hành add một delegates như sau

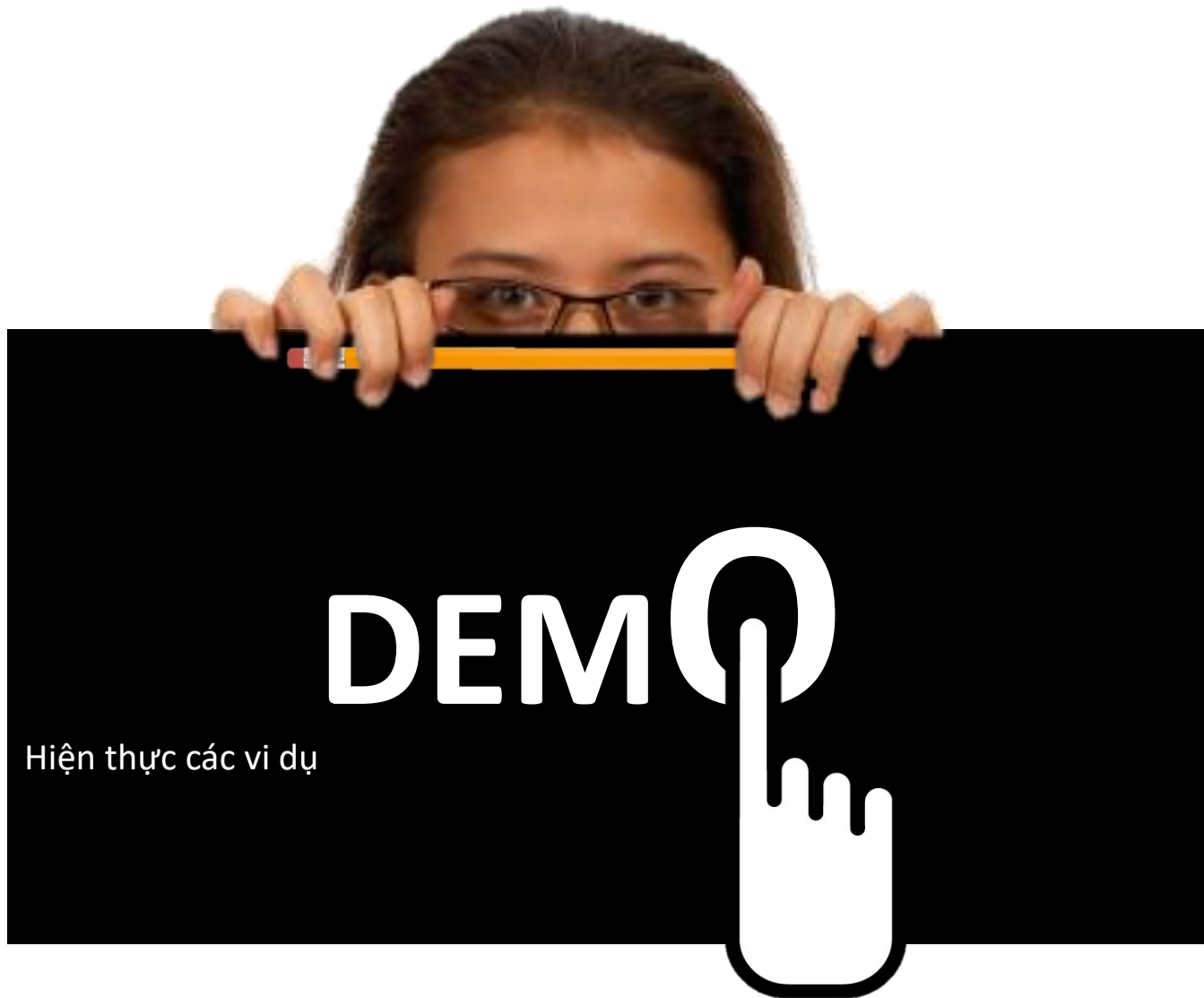
```
public delegate void delPassData(TextBox text);
```

Bước 2: Trong Form2 ta tiến hành tạo một phương thức để delegate sẽ trở tới. Trong phương thức này ta tiến hành gán giá trị của ô textbox trong Form1 vào Label của Form2:

```
public void funData(TextBox txtForm1)  
{  
    label1.Text = txtForm1.Text;  
}
```

Bước 3: Trong sự kiện button click của form1 ta tiến hành khởi tạo Form2 và delegate. Chỉ ra một phương thức của Form2 và tiến hành gọi delegate như sau:

```
private void btnSend_Click(object sender, System.EventArgs e)  
{  
    Form2 frm= new Form2();  
    delPassData del=new delPassData(frm.funData);  
    del(this.textBox1);  
    frm.Show();  
}
```



## Tổng kết bài học

- ◎ Combobox, ListBox
- ◎ GroupBox & Panel, TabControl
- ◎ Menu Strip, Tool Strip
- ◎ Listview
- ◎ Truyền dữ liệu giữa các form





**KẾT THÚC**