

Đại học Khoa học Tự nhiên, ĐHQG-HCM
Khoa Công Nghệ Thông Tin

Bài 2: Lập trình giao diện trên .NET Compact Framework

TS. Trần Minh Triết



Các control không hỗ trợ trong .NetCF

- Các control không được hỗ trợ trong .NETCF

- **CheckedListBox**
- **ColorDialog**
- **ErrorProvider**
- **FontDialog**
- **GroupBox**
- **HelpProvider**



- **LinkLabel**
- **NotificationBubble**
- **NotifyIcon**
- **All Print controls**
- **RichTextBox**
- **Splitter**



Các tính năng không hỗ trợ trong .NetCF

○ Các tính năng không được hỗ trợ trong .Net CF

- **AcceptButton**
- **CancelButton**
- **AutoScroll**
- **Anchor**
- **Multiple Document Interface (MDI)**
- **KeyPreview**
- **TabIndex**
- **TabStop**
- **Drag and drop**
- **Printing**
- **Hosting ActiveX controls**



Tăng tốc độ khởi tạo Form

Mã nguồn được tự động phát sinh khi thiết kế giao diện → Bottom Up

```
// This code is generated by the VS.NET Form Designer.  
// It builds forms from the bottom-up.  
// Button added to Panel...Panel added to Form  
this.panel1.Controls.Add(this.button1);  
this.panel1.Location = new System.Drawing.Point(16, 16);  
this.panel1.Size = new System.Drawing.Size(208, 168);  
//  
// button1  
//  
this.button1.Location = new System.Drawing.Point(8, 16);  
this.button1.Size = new System.Drawing.Size(72, 24);  
this.button1.Text = "button1";  
//  
// Form1  
//  
this.Controls.Add(this.panel1);  
this.Menu = this.mainMenu1;  
this.Text = "Form1";
```



Tăng tốc độ khởi tạo Form

Mã nguồn được “điều chỉnh” thủ công → Top Down

```
// This code is hand written and modifies the code generated  
// by the VS.NET designer.  
// It builds forms from the top-down.  
// Panel added to Form...Button added to Panel...
```

```
this.panel1.Location = new System.Drawing.Point(16, 16);
```

```
this.panel1.Size = new System.Drawing.Size(208, 168);
```

```
this.Controls.Add(this.panel1);
```

```
//
```

```
// button1
```

```
//
```

```
this.button1.Location = new System.Drawing.Point(8, 16);
```

```
this.button1.Size = new System.Drawing.Size(72, 24);
```

```
this.button1.Text = "button1";
```

```
//
```

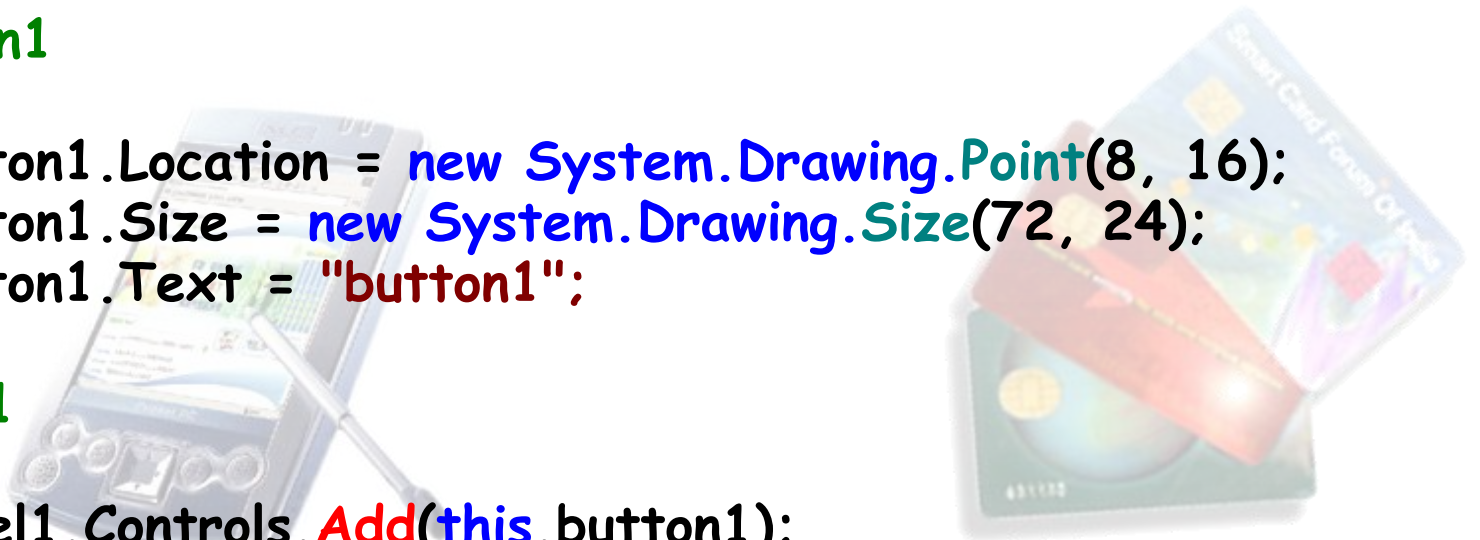
```
// Form1
```

```
//
```

```
this.panel1.Controls.Add(this.button1);
```

```
this.Menu = this.mainMenu1;
```

```
this.Text = "Form1";
```



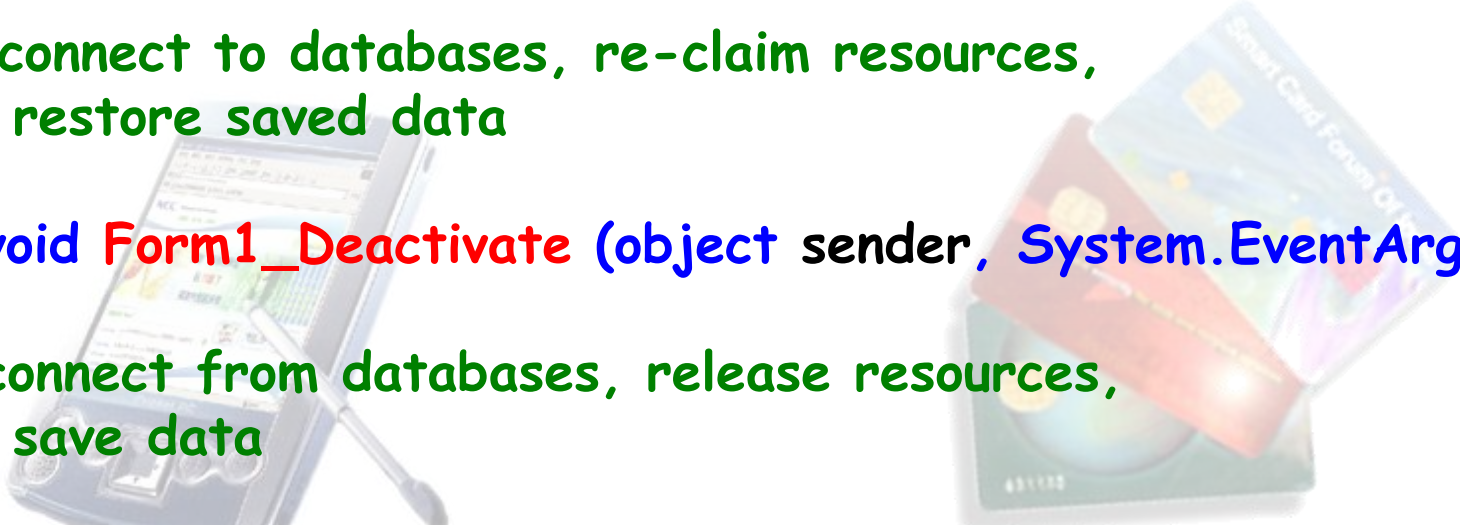
Xử lý sự kiện Activate và Deactivate

- Sự kiện **Activate** và **Deactivate** không được liệt kê trong phần **Properties** khi thiết kế **Form**!
- Cần tự bổ sung hàm xử lý sự kiện này (khi cần thiết)
- HĐH có thể chấm dứt 1 tiến trình (đang không hoạt động) nếu hệ thống bị thiếu tài nguyên!



Xử lý sự kiện Activate và Deactivate

```
private void InitializeComponent()  
{  
    this.Activated += new EventHandler(Form1_Activated);  
    this.Deactivate += new EventHandler(Form1_Deactivate);  
    // ...  
}  
private void Form1_Activate (object sender, System.EventArgs e)  
{  
    // Re-connect to databases, re-claim resources,  
    // and restore saved data  
}  
private void Form1_Deactivate (object sender, System.EventArgs e)  
{  
    // Disconnect from databases, release resources,  
    // and save data  
}
```

A PDA device is shown on the left, displaying a screen with various data fields and a keypad. To its right is a stack of three smart cards. The top card is blue and labeled 'Smart Card Forum Of...'. The middle card is red, and the bottom card is green. The cards are slightly overlapping and have a glossy finish.

Cách gán Icon cho 1 chương trình



Đối tượng Form

- Thuộc tính **FormBorderStyle**
 - Giá trị mặc định: **FormBorderStyle.FixedSingle**
 - Trên PocketPC:
 - **FormBorderStyle.None**: không có border, không có title. Có thể resize hay di chuyển bằng mã lệnh
 - **FormBorderStyle.FixedSingle** hoặc các giá trị khác: Form chiếm trọn vẹn desktop, không thể resize hay di chuyển
 - Trên Windows CE:
 - **FormBorderStyle.FixedDialog** hoặc **FormBorderStyle.None**: không có border, không có title. Có thể resize và di chuyển bằng code
 - **FormBorderStyle.FixedSingle** hoặc các giá trị khác: form có kích thước là Size, có border và title. Có thể resize và di chuyển bằng code. Có thể di chuyển bằng tay.

Đối tượng Form

- Thuộc tính **MinimizeBox** and **MaximizeBox**
 - **MinimizeBox**:
 - **TRUE**: nút X(minimize) trên title
 - **FALSE**: nút OK (close) trên title
 - **MaximizeBox**: không có ý nghĩa



Đối tượng Form

- Thuộc tính **WindowState**
 - **FormWindowState.Normal**: ứng dụng chiếm toàn màn hình, trừ vùng Start Menu và menu bar chính
 - **FormWindowState.Maximized**: ứng dụng chiếm toàn màn hình (chiếm luôn cả vùng Start Menu), trừ menu bar chính
- Thuộc tính **Size** và **Location**
 - Trên PocketPC: chỉ có ý nghĩa khi sử dụng **FormBorderStyle.None**.

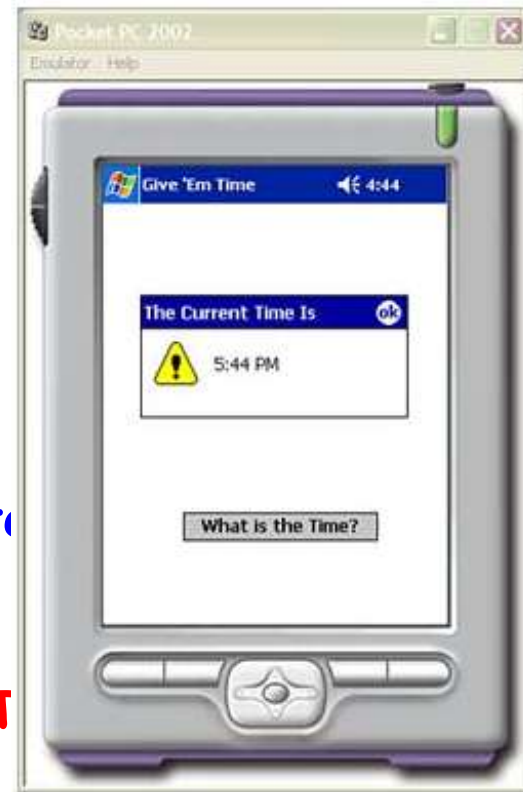


Đối tượng Button

- System.Windows.Forms.Button

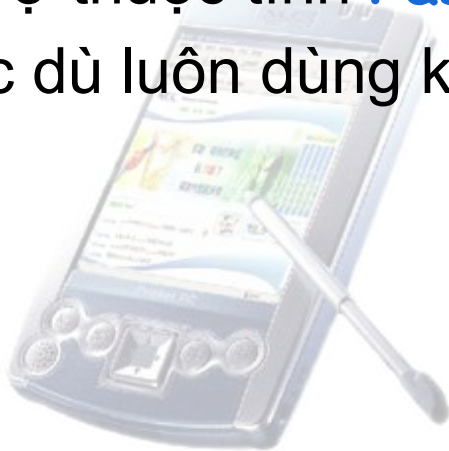
- Event: Click

```
private void button_Click(object sender, System.EventArgs e)
{
    MessageBox.Show(DateTime.Now.ToShortTimeString(),
        "The Current Time Is",
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation,
        MessageBoxDefaultButton.Button1);
}
```



Đối tượng TextBox

- Hỗ trợ thuộc tính **BackColor** và **ForeColor**
- Các sự kiện:
 - **KeyPress**
 - **KeyUp**
 - **KeyDown**
 - Không hỗ trợ sự kiện **Click**!
- Hỗ trợ thuộc tính **PasswordChar**
(mặc dù luôn dùng ký tự *)



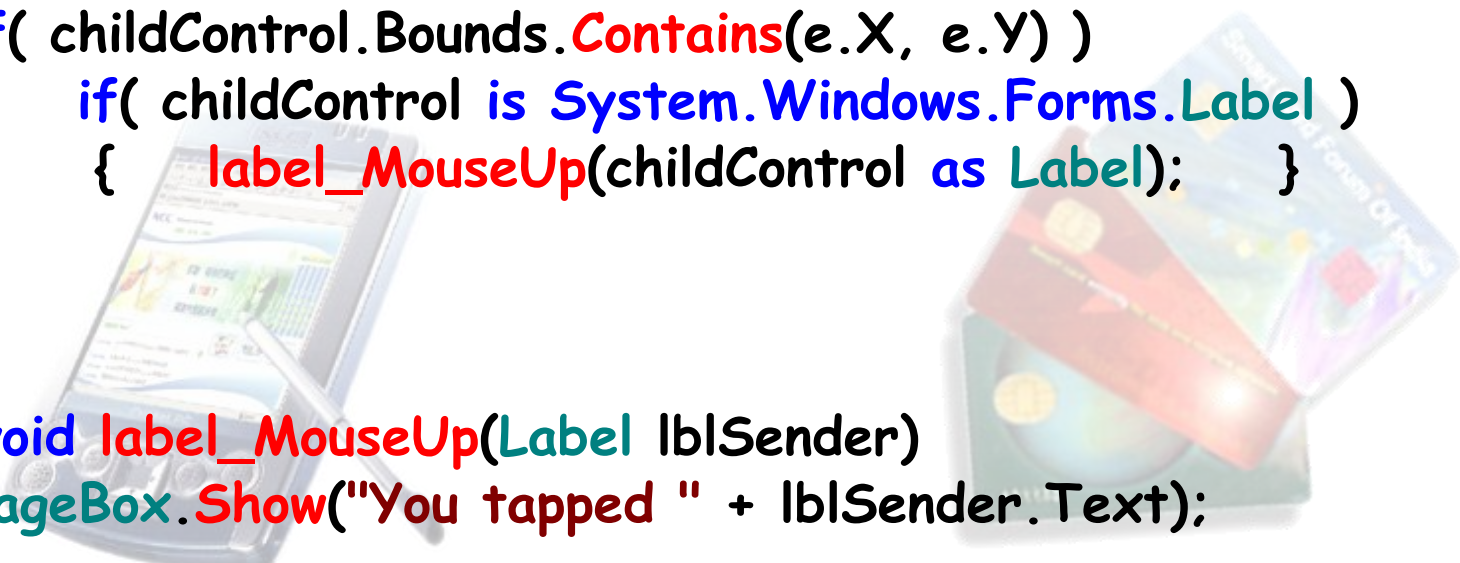
Đối tượng Label

- Thuộc tính
 - **TextAlign**:
 - **TopLeft**, **TopCenter**, **TopRight**
- Sự kiện:
 - **TextChanged** :khi thay đổi nội dung text



Sử dụng Panel và Label để tạo Linked Label “giả”

```
private void InitializeComponent()  
{  
    ...  
    this.panelABC.MouseUp +=  
        new MouseEventHandler(panelABC_MouseUp);  
}  
private void panelABC_MouseUp( object sender, MouseEventArgs e )  
{  
    foreach( Control childControl in panelABC.Controls )  
    {  
        if( childControl.Bounds.Contains(e.X, e.Y) )  
        {  
            if( childControl is System.Windows.Forms.Label )  
            {  
                label_MouseUp(childControl as Label);  
            }  
        }  
    }  
}  
private void label_MouseUp(Label lblSender)  
{  
    MessageBox.Show("You tapped " + lblSender.Text);  
}
```





Xử lý các nút trên Pocket PC

- Sự kiện **KeyDown** trên **Form**
- Trong đối tượng kiểu **KeyEventArgs**, xét thuộc tính **KeyCode**
 - **Keys.Up**
 - **Keys.Down**
 - **Keys.Left**
 - **Keys.Right**
 - **Keys.Return**



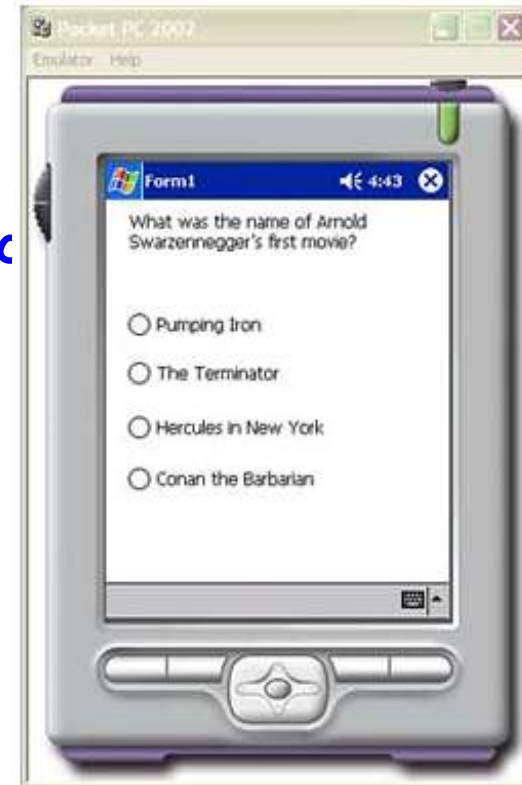
RadioButton

- Nên đặt các **RadioButton** cùng nhóm trong 1 đối tượng **Panel**
- Sự kiện:
 - Khi thay đổi trạng thái **checked**: phát sinh 2 event:
 - **Click**: khi NSD nhấn vào **RadioButton**. Không phát sinh khi NSD dùng mã lệnh để thay đổi trạng thái của **RadioButton**
 - **CheckedChanged**: khi thay đổi trạng thái của **RadioButton** (bằng tay hay dùng mã lệnh)



RadioButton

```
private void radioButton2_CheckedChanged(object  
    System.EventArgs e)  
{  
    if (this.radioButton2.Checked)  
        MessageBox.Show ("Wrong", "Wrong!");  
}
```



CheckBox

- Thuộc tính:
- **CheckState**
 - **Unchecked**
 - **Checked**
 - **Indeterminate** (khi thuộc tính **ThreeState** là **true**)
- **AutoCheck**
 - Nếu **FALSE**: không cho phép thay đổi trạng thái của đối tượng



ComboBox và ListBox

```
comboBox1.Items.Add("Hi");  
comboBox1.Items.Add("Howdy");  
comboBox1.Items.Add("Wuz Up");
```

```
listBox1.Items.Add("Hi");  
listBox1.Items.Add("Howdy");  
listBox1.Items.Add("Wuz Up");
```

SelectedIndex
SelectedItem



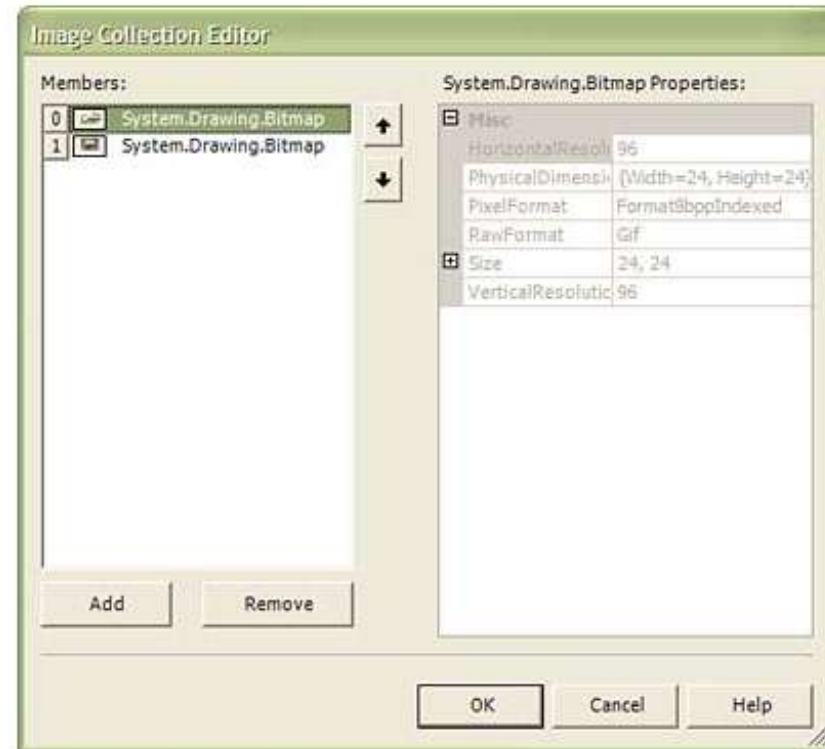
Toolbar



Toolbar

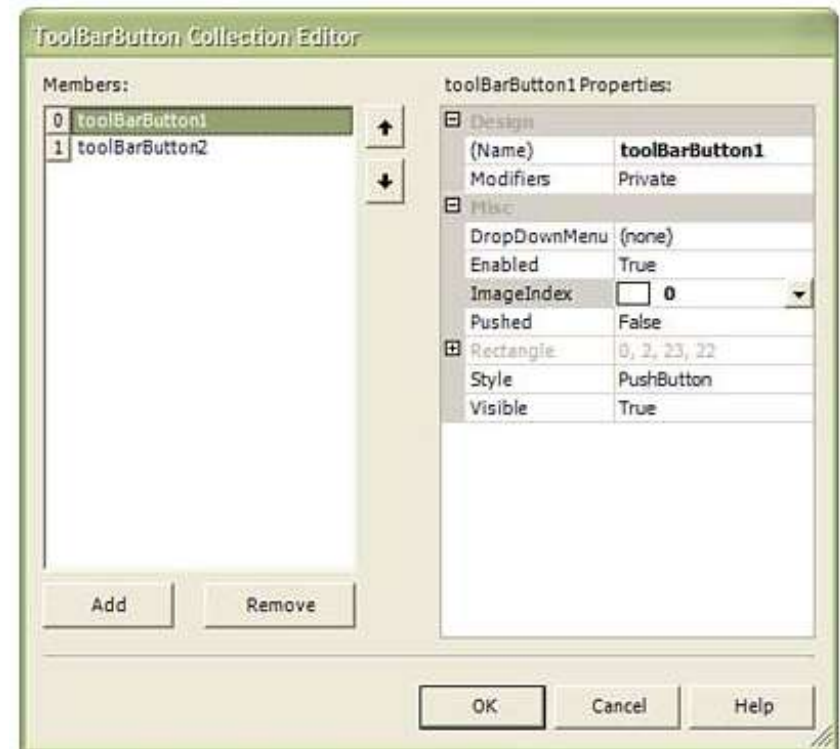
- Kéo control **ImageList** vào form → Xuất hiện **ImageList** ở phía dưới vùng thiết kế form
- Chọn thuộc tính **Images** của **ImageList** → Xuất hiện màn hình **Image Collection Editor**
- Bổ sung hình vào **ImageList** (kích thước 16x16)

Lưu ý: không cần chép file ảnh kèm theo chương trình khi deploy



Toolbar

- Kéo control **Toolbar** vào form
- Đặt thuộc tính **ImageList** của **Toolbar** là tên của **ImageList** vừa tạo
- Chọn thuộc tính **Button** của **Toolbar**



Toolbar

- Bổ sung các **button** trên **toolbar** (chọn chỉ số của **icon** tương ứng, tính từ 0)
- Style của **button**

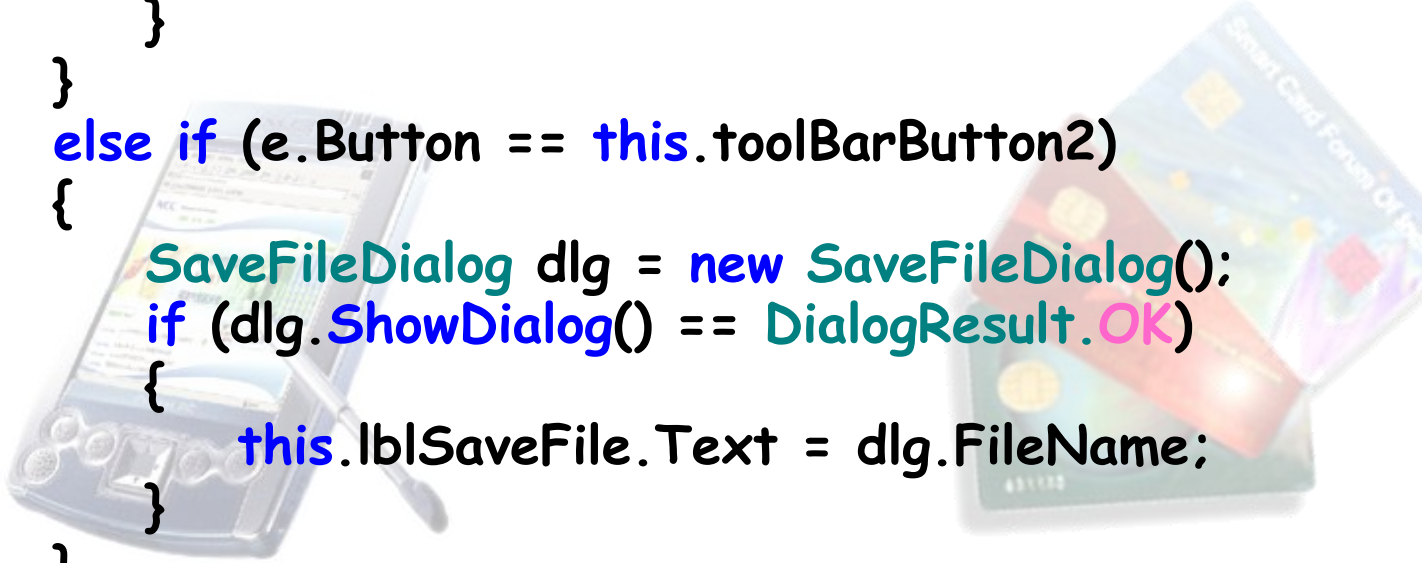
MEMBER NAME	DESCRIPTION
-------------	-------------

<code>DropDownButton</code>	When the button is clicked, a menu or other window is displayed.
<code>PushButton</code>	The regular, three-dimensional button (default).
<code>Separator</code>	A space or graphic between toolbar buttons.
<code>ToggleButton</code>	A button that appears pressed when clicked and remains pressed until it is clicked again.



Toolbar

```
private void toolBar1_ButtonClick(object sender,  
    System.Windows.Forms.ToolBarButtonClickEventArgs e)  
{  
    if (e.Button == this.toolBarButton1)  
    {  
        OpenFileDialog dlg = new OpenFileDialog();  
        if (dlg.ShowDialog() == DialogResult.OK)  
        {  
            this.lblOpenFile.Text = dlg.FileName;  
        }  
    }  
    else if (e.Button == this.toolBarButton2)  
    {  
        SaveFileDialog dlg = new SaveFileDialog();  
        if (dlg.ShowDialog() == DialogResult.OK)  
        {  
            this.lblSaveFile.Text = dlg.FileName;  
        }  
    }  
}
```



Menu

```
MenuItem fileMenu = new MenuItem();  
MenuItem newItem = new MenuItem();  
MenuItem sepItem = new MenuItem();  
MenuItem exitItem = new MenuItem();
```

```
fileMenu.Text = "File";  
newItem.Text = "New";  
sepItem.Text = "-";  
exitItem.Text = "Exit";
```

```
fileMenu.MenuItems.Add(newItem);  
fileMenu.MenuItems.Add(sepItem);  
fileMenu.MenuItems.Add(exitItem);  
mainMenu1.MenuItems.Add(fileMenu);
```



Context Menu

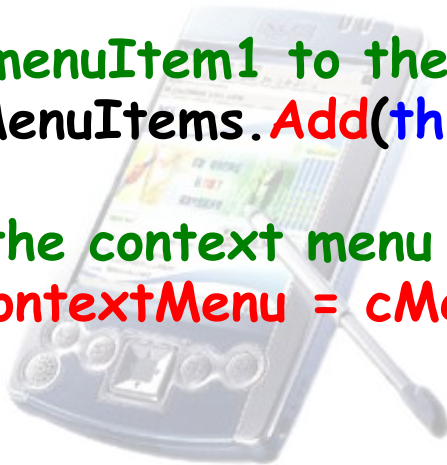
```
ContextMenu cMenu = new ContextMenu();  
MenuItem menuItem1 = new MenuItem();  
MenuItem menuItem2 = new MenuItem();
```

```
menuItem1.Text = "Default Item 1";  
menuItem2.Text = "Default Item 2";
```

```
// Add menuItem2 as a child of menuItem1  
menuItem1.MenuItems.Add(this.menuItem2);
```

```
// Add menuItem1 to the context menu  
cMenu.MenuItems.Add(this.menuItem1);
```

```
// Add the context menu to a label control  
label1.ContextMenu = cMenu;
```



Context Menu

```
private void contextMenu1_Popup(object sender,  
    System.EventArgs e)  
{  
    this.contextMenu1.MenuItems.Clear();  
    if (this.checkBox1.Checked)  
  
        this.contextMenu1.MenuItems.Add(this.menuItem1);  
    if (this.checkBox2.Checked)  
        this.contextMenu1.MenuItems.Add(this.menuItem2);  
    if (this.checkBox3.Checked)  
        this.contextMenu1.MenuItems.Add(this.menuItem3);  
    // Always add the default menu  
    this.contextMenu1.MenuItems.Add(this.menuItem4);  
}
```

Event: Popup cho phép xử lý trước khi hiển thị

Context Menu

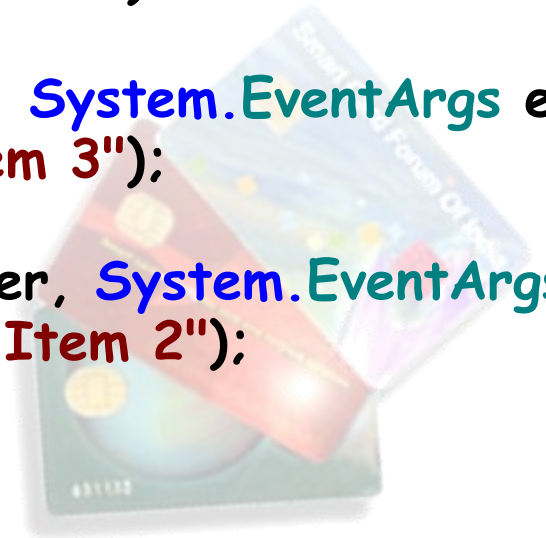
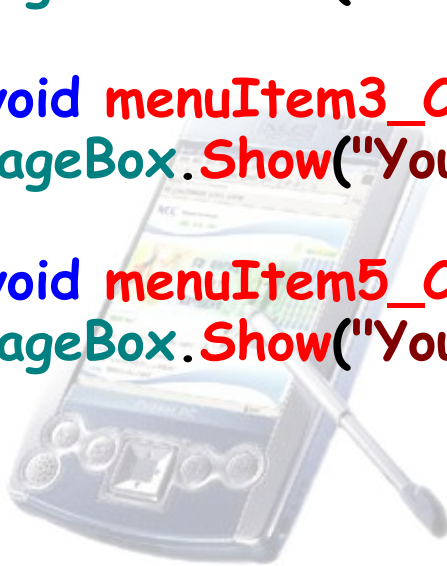
```
private void menuItem1_Click(object sender, System.EventArgs e)
{
    MessageBox.Show("You selected MenuItem 1");
}

private void menuItem2_Click(object sender, System.EventArgs e)
{
    MessageBox.Show("You selected MenuItem 2");
}

private void menuItem5_Click(object sender, System.EventArgs e)
{
    MessageBox.Show("You selected MenuItem 3");
}

private void menuItem3_Click(object sender, System.EventArgs e)
{
    MessageBox.Show("You selected MenuItem 3");
}

private void menuItem5_Click_1(object sender, System.EventArgs e)
{
    MessageBox.Show("You selected Default Item 2");
}
```



Timer

- Event: **Tick** (lưu ý: chỉ phát sinh event khi **Enabled** là **true**)
- Stop/Pause Timer bằng cách đặt **Enabled** là **False**
- Thuộc tính **Interval** tính bằng miligiây

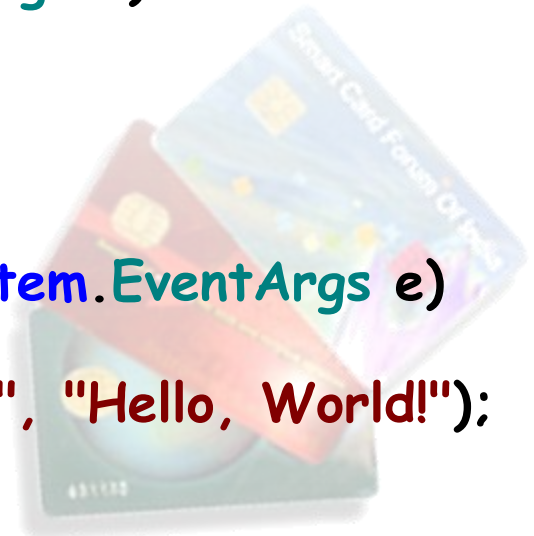
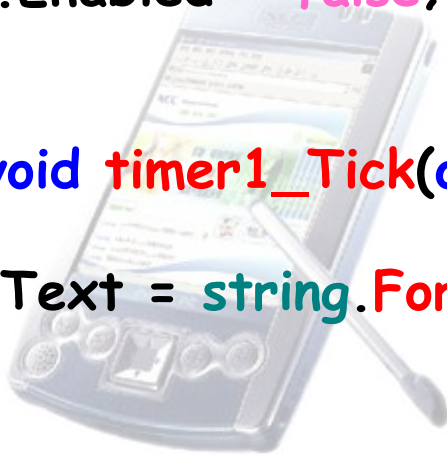


Xử lý hành động Tap&Hold

```
private void Form1_MouseDown(object sender,  
    System.Windows.Forms.MouseEventArgs e)  
{  
    timer1.Enabled = true;  
}
```

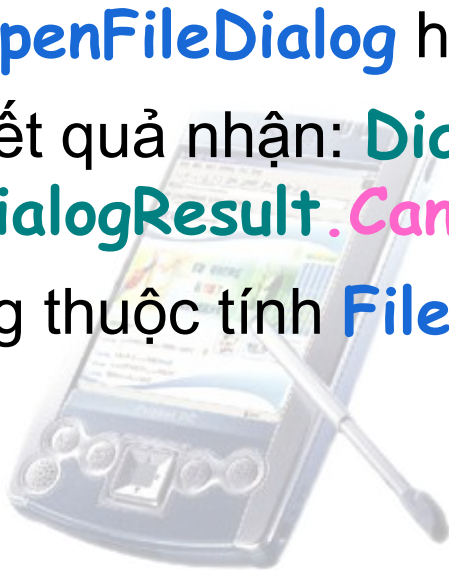
```
private void Form1_MouseUp(object sender,  
    System.Windows.Forms.MouseEventArgs e)  
{  
    timer1.Enabled = false;  
}
```

```
private void timer1_Tick(object sender, EventArgs e)  
{  
    label2.Text = string.Format("Message: {0}", "Hello, World!");  
}
```



OpenFileDialog và SaveFileDialog

- **"Dynamically Linked Library|*.dll|Executable|*.exe"**
- **InitialDirectory** mặc định, hoặc thư mục được chọn không tồn tại thì sẽ chọn là My Documents
- Hiện thị **Open/SaveFileDialog**:
 - Dùng phương thức **ShowDialog** của control **OpenFileDialog** hay **SaveFileDialog**
 - Kết quả nhận: **DialogResult.OK** hay **DialogResult.Cancel**
- Dùng thuộc tính **Filename** để nhận kết quả



OpenFileDialog và SaveFileDialog

```
OpenFileDialog ofDlg = new OpenFileDialog();  
ofDlg.Filter = "DLL|*.dll|Executable|*.exe";  
ofDlg.InitialDirectory = "\\My Documents";
```

```
if(DialogResult.OK == ofDlg.ShowDialog())  
{  
    MessageBox.Show("You Selected " + ofDlg.FileName);  
}  
else  
{  
    MessageBox.Show("Go ahead, select a file!");  
}
```



Panel

- Không hỗ trợ:
 - **BorderStyle** property
 - **BackGroundImage** property
 - **AutoScroll** property



ImageList và PictureBox

```
Bitmap image =  
    new Bitmap(Assembly.GetExecutingAssembly().  
        GetManifestResourceStream("image1.jpg");  
ImageList imgList = new ImageList();  
imgList.Images.Add(image);
```

```
pictureBox1.Image =  
    new Bitmap(@"\Program  
        Files\PictureBoxControl\tinyemulator_content.jpg");
```



ListView

• Các dạng View:

Details Items appear on separate lines, with information arranged in columns.

LargeIcon Items appear as large icons with labels underneath.

List Items appear as small icons with labels to their right, arranged in columns, but no column headers are displayed.

SmallIcon Items appear as small icons with labels to their right, arranged in columns, but no column headers are displayed.



ListView

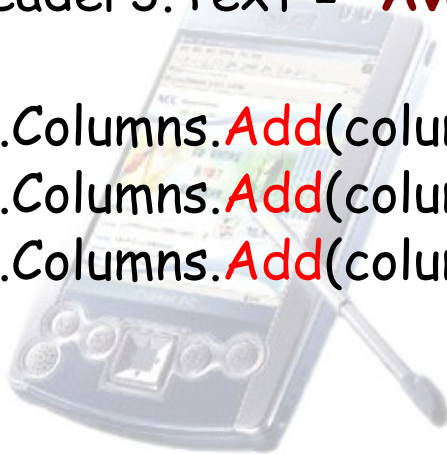
```
System.Windows.Forms.ColumnHeader columnHeader1 = new  
    ColumnHeader();
```

```
System.Windows.Forms.ColumnHeader columnHeader2 = new  
    ColumnHeader();
```

```
System.Windows.Forms.ColumnHeader columnHeader3 = new  
    ColumnHeader();
```

```
columnHeader1.Text = "Name";  
columnHeader2.Text = "Purpose";  
columnHeader3.Text = "Availability";
```

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

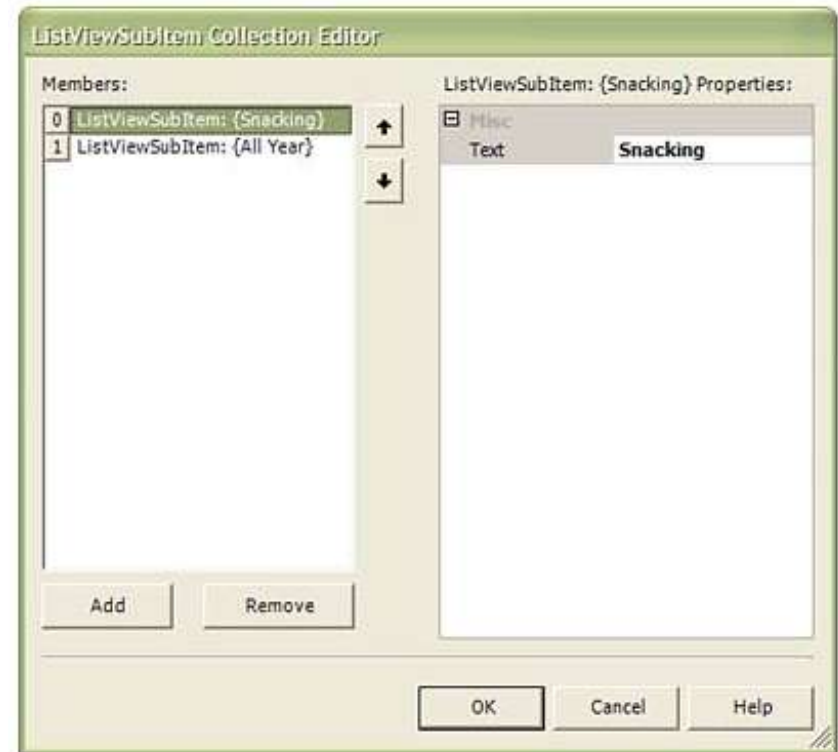


Viết code!!!



ListView

- Thiết kế bằng công cụ
 - Tạo **ListView** control trên form
 - Vào thuộc tính **Columns**



ListView

- Bổ sung item vào **ListView**
 - Item (class **ListViewItem**) và SubItem (class **ListViewSubItem**)
 - Item gồm text và image (nếu có)

```
ListViewItem listViewItem1 = new ListViewItem();
```

```
ListViewSubItem listViewSubItem1 = new ListViewSubItem();
```

```
ListViewSubItem listViewSubItem2 = new ListViewSubItem();
```

```
listViewItem1.Text = "Red Delicious";
```

```
listViewSubItem1.Text = "Snacking";
```

```
listViewSubItem2.Text = "All Year";
```

```
listViewItem1.SubItems.Add(listViewSubItem1);
```

```
listViewItem1.SubItems.Add(listViewSubItem2);
```

```
listView1.Items.Add(listViewItem1);
```



ListView

- Bổ sung **Item** lúc thiết kế: chọn thuộc tính **Items** của control. Với mỗi **Item**, có thể thiết lập **SubItems**
- Có thể bổ sung **ImageList** lúc thiết kế



ListView

- **ListView** có thể kèm theo 2 Image List
 - **LargeImageList**: dùng ở dạng **View.LargeIcon**
 - **SmallImageList**: Dùng ở các dạng View khác
- Dùng thuộc tính **ImageIndex** của mỗi Item để gán chỉ số của Image tương ứng (tính từ 0)



ListView

```
private void listView1_SelectedIndexChanged(object sender,  
    System.EventArgs e)  
{  
    if(this.listView1.SelectedIndices.Count <= 0)  
        return;  
  
    int selNdx = this.listView1.SelectedIndices[0];  
    label3.Text = listView1.Items[selNdx].Text;  
}
```

Có thể duyệt qua Items và kiểm tra thuộc tính Selected
Không cho phép MultiSelect



ListView

● Một số thuộc tính khác

PROPERTY	DESCRIPTION
<code>CheckBoxes</code>	This determines whether a check box appears next to each item in the control. When an item's checked state is about to change, an <code>ItemCheck</code> event is raised. The <code>View</code> property must be set to <code>View.Details</code> . Using check boxes is a great work-around for overcoming the lack of <code>MultiSelect</code> support.
<code>FullRowSelect</code>	This property determines whether clicking an item selects all its sub-items. The <code>View</code> property must be set to <code>View.Details</code> .
<code>HeaderStyle</code>	This determines the type of column headers to display. This property is a <code>ColumnHeaderStyle</code> enumeration. See Table 3.7 for a description of the <code>ColumnHeaderStyle</code> members.



ListView

- Một số giá trị của **ColumnHeaderStyle**

MEMBER	DESCRIPTION
<code>Clickable</code>	The column headers act as buttons. Clicking a header generates a <code>ColumnClick</code> event.
<code>Nonclickable</code>	Clicking the column headers do not generate <code>ColumnClick</code> events.
<code>None</code>	The column headers are not displayed.



TreeView

● Thuộc tính của đối tượng **TreeView**

PROPERTY	DESCRIPTION
<code>ImageList</code>	This is the <code>ImageList</code> that contains the images that will be displayed next to the tree nodes.
<code>ImageIndex</code>	This is the index of the image that is displayed next to the tree nodes. Because only one index can be specified, every tree node will display the same image. You can override the image on a specific node by setting the <code>TreeNode.ImageIndex</code> property.
<code>SelectedImageIndex</code>	Here's the index of the image that that is displayed next to a tree node when it is selected. Because only one index can be specified, every tree node will display the same image. You can override the image on a specific node by setting the <code>TreeNode.SelectedImageIndex</code> property.
<code>ShowLines</code>	This determines whether lines are drawn between tree nodes.

TreeView

● Thuộc tính của đối tượng **TreeView**

ShowPlusMinus

This property determines whether plus-sign and minus-sign buttons are displayed next to the tree nodes that contain child tree nodes.

ShowRootLines

This determines whether lines are drawn between the tree nodes that are at the root of the tree.

CheckBoxes

This property determines whether check boxes are displayed next to the tree nodes. The **Checked** property of the **TreeNode** objects is set to true if the node is checked; false, otherwise. When the user checks a node, the **AfterCheck** event is raised.



TreeView



TreeView

```
TreeNode treeNode1 = new TreeNode();
```

```
TreeNode treeNode2 = new TreeNode();
```

```
treeNode1.Text = "Red Apples";
```

```
treeNode2.Text = "Red Delicious";
```

```
treeNode1.Nodes.Add(treeNode2);
```

```
treeView1.Nodes.Add(treeNode1);
```



TreeView

TreeViewAction

MEMBER	DESCRIPTION
<code>ByKeyboard</code>	The event was raised by a keystroke.
<code>ByMouse</code>	The event was raised by a mouse click.
<code>Collapse</code>	The event was raised by the node's collapsing.
<code>Expand</code>	The event was raised by the node's expanding.
<code>Unknown</code>	The action that raised the event was unknown.



TreeView

```
private void  
treeView1_AfterSelect(object sender,  
    System.Windows.Forms.TreeViewEventArgs e)  
{  
    TreeNode selNode = e.Node;  
    label2.Text = selNode.Text;  
}
```



TreeView

● Thuộc tính của đối tượng **TreeNode**

PROPERTY	DESCRIPTION
<code>Text</code>	This is the text displayed in the label of the node.
<code>ImageIndex</code>	This is the index of the image that that is displayed next to the tree nodes. By default this value is equal to <code>TreeView.ImageIndex</code> . To specify another image index, you must set this property explicitly.
<code>SelectedImageIndex</code>	Here is the index of the image that that is displayed next to the node when the node is selected. By default this value is equal to <code>TreeView.SelectedImageIndex</code> . To specify another image index, you must set this property explicitly.

