Windows Applications

Introducing Windows Forms

Windows Forms (part of the Microsoft .NET Framework) – the basic element of the user interface (UI) in applications created for the Microsoft Windows operating system.

- Form is a window
  - Contains controls that create a UI for:
    - Display information
    - User interaction with a mouse or a keyboard
- System.Windows.Forms namespace classes

Windows Forms vs. Web Forms

Features	Windows Forms	Web Forms
Deployment	Can be run without altering the registry	No download required
Graphics	Includes GDI+	Interactive or dynamic graphics require round trips to the server for updates
Responsiveness	Provide the quickest response speed for interactive applications	Can take advantage of the browser's dynamic HTML to create rich UI
Platform	Requires .NET Framework running on the client computer	Require only a browser
Programming model	Based on a client-side, Win32-based message-pump model	Applications components are invoked via HTTP
Security	Code-based and role-based security	Role-based security

Component (System.ComponentModel.Component)

- Base class
- Implements the interface IComponent that defines the behaviour of the components

Control (System.Windows.Forms.Control)

- Component with a visual representation
- Visible (components without visual representation are not visible)

Container (System.ComponentModel.Container)

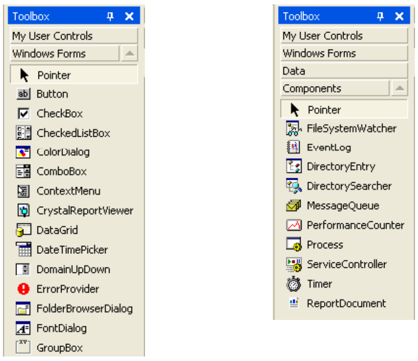
- Encapsulates components
- Dispose method releases resources explicitly (all components within the container)

- System.Object
  - System.MarshalByRefObject
    - System.ComponentModel.Component
      - **System.Windows.Forms.Control**
        - System.Windows.Forms.DataGrid
        - System.Windows.Forms.DateTimePicker
        - System.Windows.Forms.GroupBox
        - System.Windows.Forms.Label
        - System.Windows.Forms.ListControl
        - System.Windows.Forms.ListView
        - System.Windows.Forms.MonthCalendar
        - System.Windows.Forms.PictureBox
        - System.Windows.Forms.PrintPreviewControl
        - System.Windows.Forms.ProgressBar
        - System.Windows.Forms.ScrollBar
        - System.Windows.Forms.Splitter
        - System.Windows.Forms.StatusBar
        - System.Windows.Forms.TabControl
        - System.Windows.Forms.TextBoxBase
        - System.Windows.Forms.ToolBar
        - System.Windows.Forms.TrackBar
        - System.Windows.Forms.TreeView

Form (System.Windows.Forms.Form)

- Control-container for components and controls
- Different form types
  - windows
  - dialog box
  - multiple-document interface (MDI) form
- Properties – define form appearance
- Methods – define form behaviour
- Events – define form interaction with the user

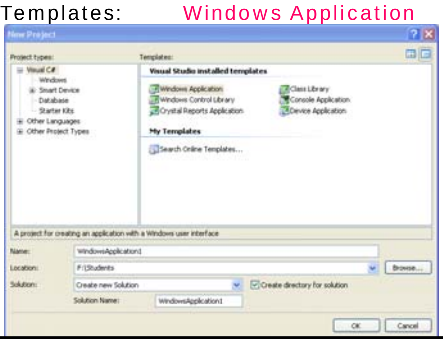
Controls and Components in Toolbox



Creating a Form

Class inherits the Form class

1. Create a new project
   - Project types: Visual C# Projects
   - Templates: Windows Application



Project types:
- Visual C#
  - Windows
  - Smart Device
  - Database
  - Starter Kits
- Other Languages
- Other Project Types

Templates:

Visual Studio installed templates

- Windows Application
- Windows Control Library
- Crystal Reports Application
- Class Library
- Console Application
- Device Application

My Templates

- Search Online Templates...

A project for creating an application with a Windows user interface

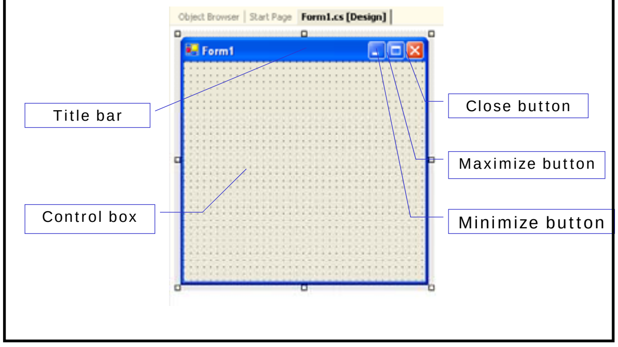
Name: WindowsApplication1

Location: F:\Students

Solution: Create new Solution ☑ Create directory for solution

Solution Name: WindowsApplication1

2. Design view – the Designer creates the default form Form1.



Title bar

Control box

Close button

Maximize button

Minimize button

3. View ⇒ Code

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Partial type definitions allow the definition of a class, struct or interface to be split into multiple files.

Form1.Designer.cs

```
namespace WindowsApplication1
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
        /// disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

Form1.Designer.cs

```
        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }

        #endregion
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace WindowsApplication1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

4. Run

Build ⇒ Build Solution
Debug ⇒ Start Without Debugging

The Main method creates and displays the form.

The System.Windows.Forms.Application.Run method begins running a standard application message loop on the current thread and makes the specified form visible.

The Main method has the attribute [STAThread].

The application closes when the form is closes. The application has to override the Dispose method that is called automatically for the main form of the application. Dispose is called explicitly for any other child form.

The Designer generates a lot of code closed between the directives #region and #endregion – avoid modifying or deleting this code.

Properties, Methods and Layout of Controls

Class Control (System.Windows.Forms)

Properties

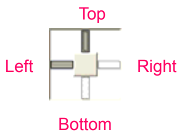
BackColor	Gets/sets the background color for the control.
BackgroundImage	Gets/sets the background image displayed in the control.
Controls	Gets the collection of controls contained within the control.
Enable	Gets/sets true/false indicating whether the control can respond to user interaction.
Focused	Gets true/false indicating whether the control has input focus.
Font	Gets/sets the font of the text displayed by the control.
ForeColor	Gets/sets the foreground color of the control.
TabIndex	Gets/sets the tab order of the control within its container. When the <Tab> is pressed the focus is moved to controls in increasing tab order.
TabStop	Gets/sets true/false indicating whether the user can give the focus to this control using the <TAB>.
Text	Gets/sets the text associated with this control.
TextAlign	Specifies the alignment of the text on the control.
Visible	Gets/sets true/false indicating whether the control is displayed

Methods

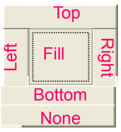
Focus	Transfers the focus to the control.
Hide	Hides the control (sets Visible to false).
Show	Shows the control (sets Visible to true).
SuspendLayout	Temporarily suspends the layout logic for the control.
ResumeLayout	Resumes the usual layout logic.

Layout Properties of Controls Inside a Container

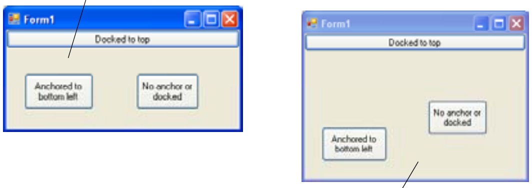
Anchor Side of parent container at which to anchor control – controls stay a fixed distance from the sides of the container, even when the control is resized. Values can be combined (Top, Left).



Dock Side of parent container to dock control. Controls spread itself along an entire side. Values cannot be combined. Fill fills entire parent.



DockPadding (for containers)	Sets the distance from docked controls to the edge of the container. The default value is 0.
Location	Gets/sets the coordinates of the upper left corner of the control, relative to its container.
Size	Gets/sets the size of the control. Takes a Size structure, which has properties Height and Width.
MinimumSize	The minimum size of the form.
MaximumSize (for Windows Forms)	The maximum size of the form.



Before resizing

After resizing

Form Life Cycle

The order of triggering of form events and methods when the Show() method is called:

1. Load
2. Activated
3. GotFocus
4. Closing
5. Closed
6. LostFocus
7. Deactivate
8. Dispose()

1. Creating the Form – new
   - The Initialize event initializes the variables, moves or resizes the controls – the initialization code must be added to the constructor after the call to InitializeComponent method.
2. Displaying the Form – Show method
   - Show includes a implied Load event loads the form into memory and displays the form.
3. Loading the Form – Load event
   - Load assigns default values to the form and its controls.
   - Load loads the form into memory.

4. Activating/Deactivating the Form
   - The Activated event activates the form.
   - Activated and Deactivated events fire each time the user moves among forms.
   - At run time the form is activated using the Activate method.
   - Activated fires when the form receives focus from another form in the same project.
   - Deactivated fires when the form loses focus to another form.
5. Getting the Focus
   - GotFocus event fires.

6. Closing the Form
   - The Closing event fires when the form receives a request to close. If the form checks for data validation and the data are not correct, the Closing event is canceled.
7. Closed the Form
   - The Closed event closes the form.
8. Losing the Focus
   – The LostFocus even fires.
9. Deactivating the Form
   – The Deactivate event fires.

10. Releasing the Recourses
    - The Dispose method is called automatically for the main form in the application; for any other form it must be called explicitly.
11. Hiding the Form
    - The Hide method removes a form from the screen without removing it from the memory.

Set Form Properties

- Using Properties Windows

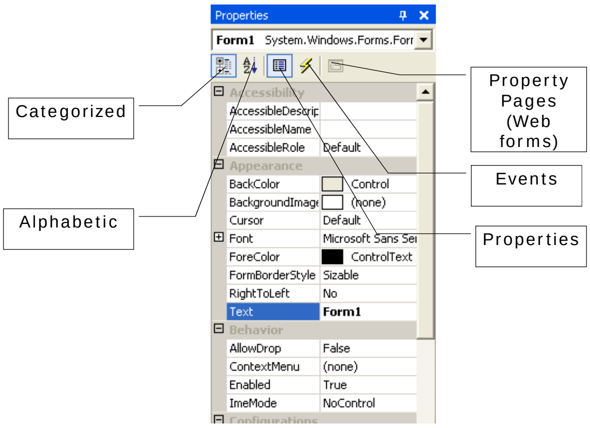
  In Design view:

  View ⇒ Properties Windows

or

- Writing code

  View ⇒ Code



Categorized

Alphabetic

Property Pages (Web forms)

Events

Properties

Properties

Form1 System.Windows.Forms.Forr

Accessibility	
AccessibleDescrip	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
Cursor	Default
Font	Microsoft Sans Sei
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
Text	Form1
Behavior	
AllowDrop	False
ContextMenu	(none)
Enabled	True
ImeMode	NoControl
Configurations	

(Name)	Sets the name (Form1) of the form in the project.
AcceptButton	Sets which button (None) is clicked when the user presses <Enter>.
AutoScaleBaseSize	Gets/sets the base size used for autoscaling of the form.
CancelButton	Sets which button (None) is clicked when the user presses <ESC>.
ClientSize	Gets/sets the size of the client area of the form (excluding the borders and the title bar).

ControlBox	Gets/sets true/false indicating whether a control box (buttons Minimize, Maximize, Help and Close) is displayed in the caption bar of the form.
FormBorderStyle	Gets/sets true/false the border style of the form (None, Sizable, Fix3D).
IsMdiContainer	Gets/sets true/false indicating whether the form is a container for multiple-document interface (MDI) child forms.
MaximizeBox	Gets/sets true/false indicating whether the Maximize button is displayed in the caption bar of the form.

MinimizeBox	Gets/sets true/false indicating whether the Minimize button is displayed in the caption bar of the form.
StartPosition	Gets/sets the starting position of the form at run time.
Size	Gets/sets the size (300; 300) of the form.
Text	Sets the text (Form1) displayed in the caption bar of the control.

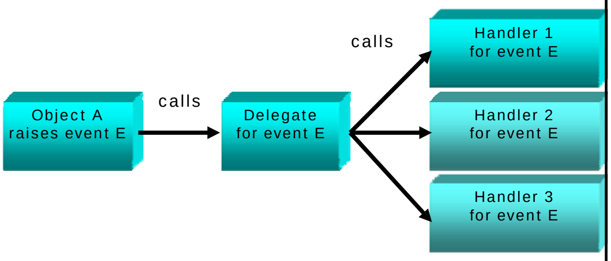
Form Methods

Close	Closes the form.
Dispose	Releases the resources used by the form.
LayoutMdi	Arranges the multiple-document interface (MDI) child forms within the MDI parent form. The MdiLayout parameter defines the layout of MDI child forms – all MDI child icons are: ArrangeIcons – arranged within the client region of the MDI parent form Cascade – cascaded within the client region of the MDI parent form TileHorizontal – tiled horizontally within the client region of the MDI parent form TileVertical – tiled vertically within the client region of the MDI parent form

Handle Form Events

Event – a message that a control sends to notify when the program's user interacts with the control:

- publisher (sender) – generates the event
- subscriber (receiver) – manipulates the event
- multicast delegate – acts as intermediary between the publisher and the subscriber and defines the signature for the control's event handler (a segment of code that is called when a corresponding event occurs)



Form Events

Load	Occurs before a form is displayed for the first time (default).
Click	Occurs when the control is clicked.

Event handler for a control in .NET

1. Adding an event handler

View Designer ⇒ <L> control-sender ⇒ <L^2> ⇒ the event handler <object-sender name>_<default event name> is added

or

View Designer ⇒ <L> control-sender ⇒ Properties windows ⇒ ⚡ ⇒ <L> event ⇒ enter text for the event name <Enter> ⇒ the event handler <event handler name> is added

2. Automatically registration of the event handler in the InitializeComponent() method

```
this.<object-sender name>.<event name> +=
new System.EventHandler (this.<event handler name>);
```

3. Automatically adding a event handler method that implements the handler program logic

```
private void <event handler name> (object sender,
                                    System.EventArgs e)
{
   // user code for event handle
}
```

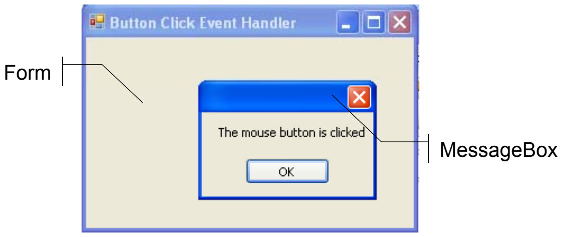
Adding an event handler at run time

```
this.<object-sender name>.<event name> +=
new System.EventHandler (this.<event handler name>);
```

Removing an event handler at run time

```
this.<object-sender name>.<event name> -=
new System.EventHandler (this.<event handler name>);
```

Example: Windows application that implements the button Click event handler that displays the message The mouse button is clicked in the box.



The mouse button is clicked

Class MessageBox (System.Windows.Forms) displays a message box that can contain text, buttons, and symbols that inform and instruct the user.

```
public static DialogResult Show (string text);
```

Method MessageBox.Show displays a message box with specified text.

```
public static DialogResult Show (string text, string caption,
        MessageBoxButtons buttons, MessageBoxIcon icon);
```

Displays a message box with specified:

text
caption
buttons (AbortRetryIgnore, OK, OKCancel, RetryCancel, YesNo, YesNoCancel)
icon (Asterisk, Error, Exclamation, Hand, Information, None, Question, Stop, Warning)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication1
{
   public partial class MyForm : Form
   {
      public MyForm()
      {
         InitializeComponent();
      }
      private void MyForm_Click(object sender, EventArgs e)
      {
         MessageBox.Show("The mouse button is clicked");
      }
   }
}
```

```
namespace WindowsApplication1
{
   partial class MyForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;
      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be
      /// disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
```

```
        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.SuspendLayout();
            // MyForm
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(292, 170);
            this.Name = "MyForm";
            this.Text = "Button Click Event Handler";
            this.Click += new System.EventHandler(this.MyForm_Click);
            this.ResumeLayout(false);
        }
        #endregion
    }
}
```

Controls

Controls – reusable components that encapsulate user interface functionality.

Adding Controls to a Form

Class Control.ContainerCollection represents a collection of Control objects.

1. Method Form.ContainerCollection.Add adds a control to the form.

   `public override void Add (Control value);`

2. Method Form.ContainerCollection.Range adds an array of control objects to the collection.

   `public virtual void AddRange (Control[] controls);`

Using Controls in Forms

1. Defining a control

   `private <Control> <control>;`

2. Creating a control

   `this.<control> = new <Control>();`

3. Setting control properties

```
this.<control>.Location=new System.Drawing.Point(10,10);
this.<control>.Name = "myControl";
this.<control>.Size = new System.Drawing.Size(50, 20);
this.<control>.TabIndex = 0;
this.<control>.Text = "My Control";
...
```

4. Adding controls to a form

   `this.Controls.Add (this.<control>);`

   or

```
this.Controls.AddRange
   (new System.Windows.Forms.Control[] {this.<control>} );
```

5. Adding control event handler

```
this.<control>.<event name> +=
new System.EventHandler (this.<event handler name>);
```

6. Implementing the event handler

```
private void <event handler name> (object sender,
                                   System.EventArgs e)
{
   // user code for event handle
}
```

Control Categorized Based on Their Functionality:

1. Commands category controls
- Button – used to start, stop, or interrupt a process
- ToolBar – contains a collection of button controls

2. Text category controls
- TextBox – displays text entered at design time that can be edited by users at run time, or changed programmatically
- RichTextBox – enables text to be displayed with formatting in plain text or rich-text format (RTF)
- Label – displays text that users cannot directly edit
- StatusBar – displays information about the application's current state by using a framed window – usually located at the bottom of a parent window

3. Options category controls
- CheckedListBox – displays a scrollable list of items, each accompanied by a check box
- ComboBox – displays a drop-down list of items
- ListBox – displays a list of text and graphical items (icons)
- ListView – displays items on one of four different views: text only, text with small icons, text with large icons, and a report view
- TreeView – displays a hierarchical collection of node objects, which can consist of text with optional check boxes or icons

4. Selection category controls
- CheckBox – displays a check box and a label with a text
- RadioButton – represents a radio button control

- DateTimePicker – represents a visual calendar that allows the user to select a date and a time
- MonthCalendar – represents a visual monthly calendar that enables the user to select a date

5. Menu category controls

- MainMenu (MenuStrip) – provides a design-time interface for creating menus
- ContextMenu (ContextMenuStrip) – implements a menu that appears when the user right-clicks an object

6. Dialog boxes category controls

- ColorDialog – represents a common dialog box that displays available colors along with controls that enable the user to define custom colors
- FontDialog – prompts the user to choose a font from among those installed on the local computer

---

- OpenFileDialog – prompts the user to open a file
- PrintDialog – allows users to select a printer and choose which portions of the document to print
- PrintPreviewDialog – represents a the raw preview part of print previewing
- PageSetupDialog – enables users to change page-related print settings, including margins and paper orientation
- SaveFileDialog – prompts the user to select a location for saving a file

7. Containers category controls

- Panel – groups a set of controls on a unlabeled, scrollable frame
- GroupBox – groups a set of controls (such as radio buttons) on a labeled, nonscrollable frame

---

- TabControl – provides a tabbed page for organizing and accessing grouped objects efficiently

8. Graphics category controls

- ImageList – serves as a repository for images
- PictureBox – displays graphical files, such as bitmaps and icons, in a frame

---

Control **Label**

Represents a standard Windows label. The users cannot directly edit the text. Does not get the focus.

Properties

Text	Sets/gets the text for the control
TextAlign	Gets/sets the alignment of text
TabIndex	Gets/sets the tab order of the control within its container
UseMnemonic	Gets/sets a value indicating whether the control interprets an ampersand character (&) in the control's Text property to be an access key prefix character

---

Control **Button**

Represents a Windows button control.

Properties

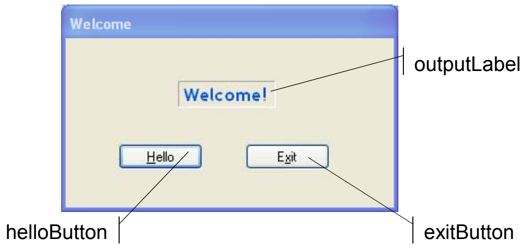
Text	Text displayed on the button face The access key for a button is created using an & before the letter that will be the shortcut (ALT+letter).

Events	Occurs when
Click	the Button control is clicked (default)
MouseEnter	the mouse pointer enters the control
MouseClick	the control is clicked by the mouse
MouseDoubleClick	the user double-clicks the Button control with the mouse
MouseDown	the mouse pointer is over the control and a mouse button is pressed
MouseMove	the mouse pointer is moved over the control
MouseUp	the mouse pointer is over the control and a mouse button is released

Ways to Button Selecting:

1. Use a mouse to click the button.
2. Invoke the button's Click event in code.
3. Move the focus to the button by pressing the <TAB>, and then choose the button by pressing the <SPACEBAR> or <ENTER>.
4. Press the access key (ALT + the underlined letter) for the button.
5. If the button is the "accept" button of the form, pressing <ENTER> chooses the button.
6. If the button is the "cancel" button of the form, pressing <ESC> chooses the button.
7. Call the Button.PerformClick method to select the button programmatically.

Example: Windows application without buttons in the title bar that contains a label and two buttons: Hello (as Acceptance button) and Exit (as Cancel button). When the button Hello (ALT+H) or <Enter> is pressed the label displays the text Welcome!. When the button Exit (ALT+x) or <ESC> is pressed the form closes.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication2
{   public partial class Form1 : Form
    {   public Form1()
        {  InitializeComponent();  }
        private void helloButton_Click(object sender, EventArgs e)
        {
            outpuLabel.Text = "Welcome!";
        }
        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

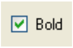
```
namespace WindowsApplication2
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
        /// disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.helloButton = new System.Windows.Forms.Button();
            this.exitButton = new System.Windows.Forms.Button();
            this.outpuLabel = new System.Windows.Forms.Label();
            this.SuspendLayout();

            // helloButton
            this.helloButton.Location = new System.Drawing.Point(47, 94);
            this.helloButton.Name = "helloButton";
            this.helloButton.Size = new System.Drawing.Size(75, 23);
            this.helloButton.TabIndex = 0;
            this.helloButton.Text = "&Hello";
            this.helloButton.UseVisualStyleBackColor = true;
            this.helloButton.Click +=
                          new System.EventHandler(this.helloButton_Click);
```

```
            // exitButton
            this.exitButton.DialogResult =
                          System.Windows.Forms.DialogResult.Cancel;
            this.exitButton.Location = new System.Drawing.Point(160, 94);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(75, 23);
            this.exitButton.TabIndex = 1;
            this.exitButton.Text = "E&xit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click +=
                          new System.EventHandler(this.exitButton_Click);
            // outpuLabel
            this.outpuLabel.AutoSize = true;
            this.outpuLabel.BorderStyle =
                   System.Windows.Forms.BorderStyle.Fixed3D;
            this.outpuLabel.Font = new System.Drawing.Font("Trebuchet MS",
                   12F, System.Drawing.FontStyle.Bold,
                   System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.outpuLabel.ForeColor =
                   System.Drawing.SystemColors.ActiveCaption;
            this.outpuLabel.Location = new System.Drawing.Point(100, 38);
            this.outpuLabel.Name = "outpuLabel";
            this.outpuLabel.Size = new System.Drawing.Size(2, 24);
            this.outpuLabel.TabIndex = 2;
            // Form1
            this.AcceptButton = this.helloButton;
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.CancelButton = this.exitButton;
            this.ClientSize = new System.Drawing.Size(292, 151);
            this.ControlBox = false;
            this.Controls.Add(this.outpuLabel);
            this.Controls.Add(this.exitButton);
            this.Controls.Add(this.helloButton);
            this.Name = "Form1";
            this.Text = "Welcome";
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.Button helloButton;
        private System.Windows.Forms.Button exitButton;
        private System.Windows.Forms.Label outpuLabel;
    }       }
```

Control CheckBox

Displays a check box that allows the user to select a true or false condition.



Properties

Checked — Gets/sets true/false indicating whether the control is in the checked state

CheckState — Gets/sets the state of the control:
Checked – displays a check mark;
Unchecked – empty check box;
Indeterminate – displays a check mark and is shaded

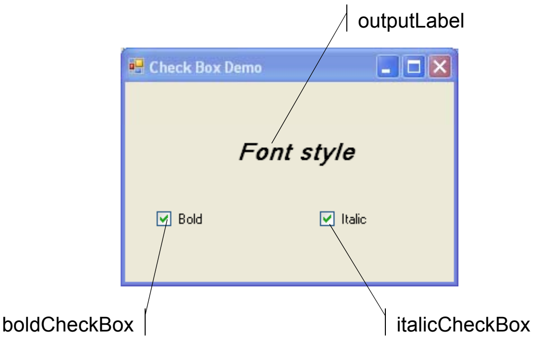
Text — Displays a text on the right of the control

Events

CheckedChanged — Occurs when the value of the Checked property changes (default)

CheckStateChanged — Occurs when the value of the CheckState property changes

Example: Windows application that changes the font style of a label using two CheckBox controls.



Class Font (System.Drawing) – defines a particular format for text, including font face, size, and style attributes.

```
public Font (string familyName, float emSize, FontStyle style );
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication3
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
```

```
      private void boldCheckBox_CheckedChanged (object sender,
                                                            EventArgs e)
      {
         outputLabel.Font = new Font (outputLabel.Font.Name,
               outputLabel.Font.Size, outputLabel.Font.Style ^ FontStyle.Bold);
      }

      private void italicCheckBox_CheckedChanged (object sender,
                                                            EventArgs e)
      {
         outputLabel.Font = new Font (outputLabel.Font.Name,
               outputLabel.Font.Size, outputLabel.Font.Style ^ FontStyle.Italic);
      }
   }
}
```

```
namespace WindowsApplication3
{
   partial class Form1
   {
      private System.ComponentModel.IContainer components = null;

      protected override void Dispose(bool disposing)
      {
         …
      }

      #region Windows Form Designer generated code

      private void InitializeComponent()
      {
         this.outputLabel = new System.Windows.Forms.Label();
         this.boldCheckBox = new System.Windows.Forms.CheckBox();
         this.italicCheckBox = new System.Windows.Forms.CheckBox();
         …
```

```
         …
         this.boldCheckBox.CheckedChanged +=
            new System.EventHandler (this.boldCheckBox_CheckedChanged);
         …
         this.italicCheckBox.CheckedChanged +=
            new System.EventHandler (this.italicCheckBox_CheckedChanged);
         …
         this.Controls.Add (this.italicCheckBox);
         this.Controls.Add (this.boldCheckBox);
         this.Controls.Add (this.outputLabel);
         this.Name = "Form1";
         this.Text = "Check Box Demo";
         …
      }
      #endregion

      private System.Windows.Forms.Label outputLabel;
      private System.Windows.Forms.CheckBox boldCheckBox;
      private System.Windows.Forms.CheckBox italicCheckBox;
   }
}
```

Control TextBox

Represents a Windows text box control. It is used to get input from the user or to display text.

Properties

Text	Sets/returns the current text in the control
PasswordChar	Gets/sets the character used to mask characters of a password in a single-line TextBox control
Lines	Gets/sets the lines of text in a text box control

Methods

Clear	Clears all text from the text box control

Events

TextChanged	Occurs when the Text property value changes (default)

Control ListBox

Represents a Windows control to display a list of items from which the user can select one or more.

Properties

Items
Gets the collection of items in the list control
Add() – adds an item to the list of items
Clear() – removes all items
RemoveAt() – removes the item at the specified index

MultiColumn
Gets/sets true/false indicating whether the ListBox supports multiple columns

SelectedIndex
Gets/sets the zero-based index of the currently selected item (-1 if no item is selected)

SelectedIndices
Gets a collection that contains the zero-based indexes of all currently selected items

SelectedItem
Gets/sets the currently selected item

SelectedItems
Gets a collection containing the currently selected items

Sorted
Gets/sets true/false indicating whether the items are sorted alphabetically

SelectionMode
Gets/sets the method in which items are selected:
MultiExtended – multiple items can be selected using <SHIFT>, <CTRL>, and arrow keys to make selections
MultiSimple – multiple items can be selected
None – no items can be selected
One – only one item can be selected

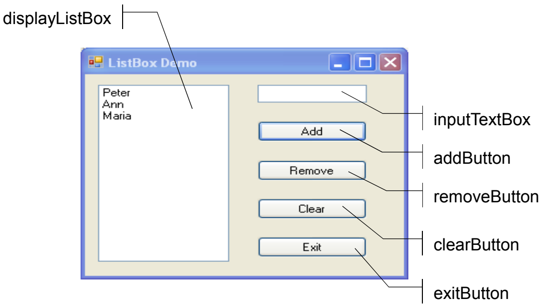
Methods

GetSelected
Returns true/false indicating whether the specified item is selected

Events

SelectedIndexChanged
Occurs when the SelectedIndex property has changed (default)

Example: Windows Form that enables the user to add, remove and clear items from the ListBox.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication4
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void addButton_Click(object sender, EventArgs e)
      {
         displayListBox.Items.Add(inputTextBox.Text);
         inputTextBox.Clear();
      }
```

```
      private void removeButton_Click(object sender, EventArgs e)
      {
         if (displayListBox.SelectedIndex != -1)
            displayListBox.Items.RemoveAt(displayListBox.SelectedIndex);
      }

      private void clearButton_Click(object sender, EventArgs e)
      {
         displayListBox.Items.Clear();
      }

      private void exitButton_Click(object sender, EventArgs e)
      {
         Application.Exit();
      }
   }
}
```

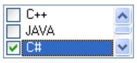
```
namespace WindowsApplication4
{
   partial class Form1
   {
      private System.ComponentModel.IContainer components = null;
      protected override void Dispose(bool disposing) { … }

      #region Windows Form Designer generated code

      private void InitializeComponent()
      {
         this.inputTextBox   = new System.Windows.Forms.TextBox();
         this.addButton       = new System.Windows.Forms.Button();
         this.removeButton  = new System.Windows.Forms.Button();
         this.clearButton     = new System.Windows.Forms.Button();
         this.exitButton       = new System.Windows.Forms.Button();
         this.displayListBox = new System.Windows.Forms.ListBox();
         …
         this.addButton.Click +=
                                 new System.EventHandler(this.addButton_Click);
         this.removeButton.Click +=
                                 new System.EventHandler(this.removeButton_Click);
```

```
         this.clearButton.Click +=
                                 new System.EventHandler(this.clearButton_Click);
          this.exitButton.Click +=
                                 new System.EventHandler(this.exitButton_Click);
         this.Controls.Add(this.displayListBox);
         this.Controls.Add(this.exitButton);
         this.Controls.Add(this.clearButton);
         this.Controls.Add(this.removeButton);
         this.Controls.Add(this.addButton);
         this.Controls.Add(this.inputTextBox);
         this.Name = "Form1";
         this.Text = "ListBox Demo";
      }
      #endregion
      private System.Windows.Forms.TextBox inputTextBox;
      private System.Windows.Forms.Button addButton;
      private System.Windows.Forms.Button removeButton;
      private System.Windows.Forms.Button clearButton;
      private System.Windows.Forms.Button exitButton;
      private System.Windows.Forms.ListBox displayListBox;
   }
}
```

Control CheckedListBox

Displays a ListBox in which a check box is displayed to the left of each item.

Selection – double click.



□ C++
□ JAVA
☑ C#

Properties

CheckedItems	Collection of checked items in the control
CheckedIndices	Collection of checked indexes in the control
SelectionMode	Gets/sets a value specifying the selection mode: None, One

Methods

GetItemChecked	Returns true/false indicating whether the specified item is checked

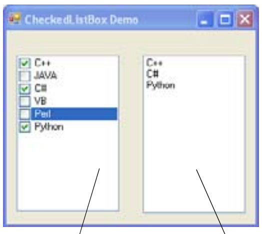
Events

ItemCheck	Occurs when the checked state of an item changes

Properties of ItemCheckEventArgs

CurrentValue	Gets a value indicating the current state of the item's check box: Checked, Unchecked, Indeterminate
Index	Gets the zero-based index of the item to change
NewValue	Gets/sets a value indicating whether to set the check box for the item to be checked, unchecked, or indeterminate

Example: Form that displays selected elements from a list using a CheckedListBox. The user can select multiple items from the CheckedListBox and display the selection in the ListBox.



CheckedListBox Demo

☑ C++
□ JAVA
☑ C#
□ VB
□ Perl
☑ Python

C++
C#
Python

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication5
{  public partial class Form1 : Form
   {  public Form1()
      {  InitializeComponent();  }
      private void inputCheckedListBox_ItemCheck (object sender,
                                                 ItemCheckEventArgs e)
      {
         string item = inputCheckedListBox.SelectedItem.ToString();
         if (e.NewValue == CheckState.Checked)
            displayListBox.Items.Add(item);
         else
            displayListBox.Items.Remove(item);
      }
   }
}
```

```
namespace WindowsApplication5
{
   partial class Form1
   {
      private System.ComponentModel.IContainer components = null;
      protected override void Dispose(bool disposing)  {  …  }

      #region Windows Form Designer generated code
      private void InitializeComponent()
      {
         this.inputCheckedListBox =
                                    new System.Windows.Forms.CheckedListBox();
         this.displayListBox = new System.Windows.Forms.ListBox();

         this.inputCheckedListBox.Items.AddRange(new object[] {
         "C++", "JAVA", "C#", "VB", "Perl", "Python"});

         this.inputCheckedListBox.ItemCheck +=
              new System.Windows.Forms.ItemCheckEventHandler
                   (this.inputCheckedListBox_ItemCheck);
```

```
         this.Controls.Add(this.displayListBox);
         this.Controls.Add(this.inputCheckedListBox);
         this.Name = "Form1";
         this.Text = "CheckedListBox Demo";
         this.ResumeLayout(false);
      }

      #endregion

      private System.Windows.Forms.CheckedListBox inputCheckedListBox;
      private System.Windows.Forms.ListBox displayListBox;
   }
}
```

Control ComboBox

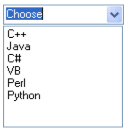
Displays data in a drop-down combo box.

Properties

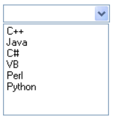
DropDownStyle

Gets/sets a value specifying the style of the combo box:

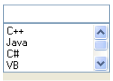
DropDown – the text portion is editable; the user must click the arrow button to display the list portion



DropDownList – the user cannot directly edit the text portion; the user must click the arrow button to display the list portion



Simple – the text portion is editable; the list portion is always visible



Items
Clear(), Add(), Remove()

Gets an object representing the collection of the items

MaxDropDownItems

Gets/sets the maximum number of items (1-100) to be shown in the drop-down portion

SelectedIndex

Gets/sets the index specifying the currently selected item (-1 if no selected item)

SelectedItem

Gets/sets currently selected item

Sorted

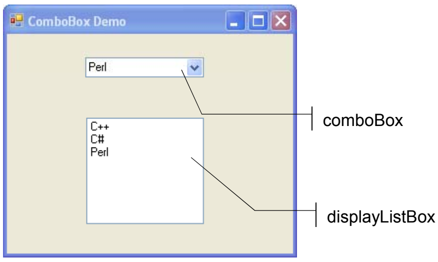
Gets/sets true/false indicating whether the items in the combo box are sorted

Events

SelectedIndexChanged

Occurs when the SelectedIndex property has changed (default)

Example: Form that displays selected elements from a list using a ComboBox. If the selected item from the ComboBox is identical with an item from the ListBox the element is removed from the ListBox otherwise the element is added to the ListBox.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication6
{   public partial class Form1 : Form
    {   public Form1() {    InitializeComponent();    }
        private void comboBox_SelectedIndexChanged(object sender,
                                                   EventArgs e)
        {  string item = comboBox.SelectedItem.ToString();
           for (int i = 0; i < displayListBox.Items.Count; i++)
              if ((string)displayListBox.Items[i] == item)
              {   displayListBox.Items.Remove(item);
                  return;
              }
           displayListBox.Items.Add(item);
        }
    }
}
```

```
namespace WindowsApplication6
{
    partial class Form1
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing) {   …   }

        #region Windows Form Designer generated code

        private void InitializeComponent()
        {
            this.comboBox = new System.Windows.Forms.ComboBox();
            this.displayListBox = new System.Windows.Forms.ListBox();

            this.comboBox.Items.AddRange(new object[] {"C++", "Java", "C#",
                                                "VB", "Perl", "Python"});
            this.comboBox.Text = "Choose";
            this.comboBox.SelectedIndexChanged +=
                new System.EventHandler(this.comboBox_SelectedIndexChanged);
```

```
            this.Controls.Add(this.displayListBox);
            this.Controls.Add(this.comboBox);
            this.Name = "Form1";
            this.Text = "ComboBox Demo";
        }

        #endregion

        private System.Windows.Forms.ComboBox comboBox;
        private System.Windows.Forms.ListBox displayListBox;
    }
}
```

Control RadioButton

Control with two sates (true/false). Radio buttons are grouped by drawing them inside a container such as a Panel control, a GroupBox control, or a form. When a user selects a radio button, the other radio buttons in the same group cannot be selected as well.

Properties

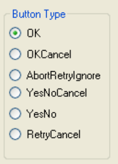
Checked	Gets/sets true/false indicating whether the CheckBox control is checked
Text	Gets/sets the text label associated with the CheckBox

Events

Click	Occurs when the control is clicked
CheckedChanged	Occurs when the value of the Checked property changes (default)

Control GroupBox

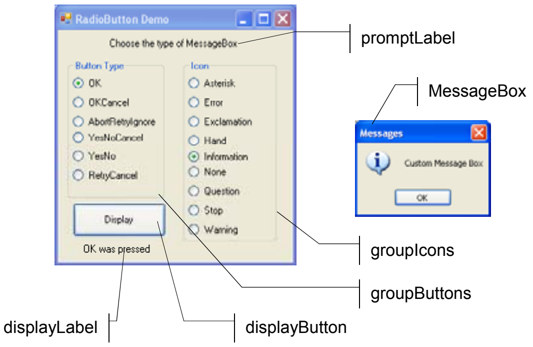
Provides an identifiable grouping for other controls.



Properties

Controls	Gets the collection of controls contained within the control
Text	Gets/sets the text associated with this control

Example: Form uses radio buttons to select options for a MessageBox



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication7
{
   public partial class Form1 : Form
   {
      private MessageBoxButtons buttonType;
      private MessageBoxIcon iconType;
      public Form1()
      {
         InitializeComponent();
         buttonType = MessageBoxButtons.OK;
         iconType=MessageBoxIcon.Information;
         okButton.Checked = true;
         informationButton.Checked = true;
      }
```

```
private void buttonType_CheckedChanged(object sender, EventArgs e)
{
   if (sender == okButton)
      buttonType = MessageBoxButtons.OK;
   else if (sender == okCancelButton)
      buttonType = MessageBoxButtons.OKCancel;
   else if (sender == abortRetryIgnoreButton)
      buttonType = MessageBoxButtons.AbortRetryIgnore;
   else if (sender == yesNoCancelButton)
      buttonType = MessageBoxButtons.YesNoCancel;
   else if (sender == yesNoButton)
      buttonType = MessageBoxButtons.YesNo;
   else
      buttonType = MessageBoxButtons.RetryCancel;
}
```

```
private void iconType_CheckedChanged(object sender, EventArgs e)
{
   if (sender == asteriskButton)
      iconType = MessageBoxIcon.Asterisk;
   else if (sender == errorButton)
      iconType = MessageBoxIcon.Error;
   else if (sender == exclamationButton)
      iconType = MessageBoxIcon.Exclamation;
   else if (sender == handButton)
      iconType = MessageBoxIcon.Hand;
   else if (sender == informationButton)
      iconType = MessageBoxIcon.Information;
   else if (sender == noneButton)
      iconType = MessageBoxIcon.None;
   else if (sender == questionButton)
      iconType = MessageBoxIcon.Question;
   else if (sender == stopButton)
      iconType = MessageBoxIcon.Stop;
   else
      iconType = MessageBoxIcon.Warning;
}
```

```
    private void displayButton_Click(object sender, EventArgs e)
    {  DialogResult result = MessageBox.Show("Custom Message Box",
                                             "Messages", buttonType, iconType);
       switch (result)
       {  case DialogResult.OK:       displayLabel.Text="OK was pressed";
                                      break;
          case DialogResult.Cancel:   displayLabel.Text="Cancel was pressed";
                                      break;
          case DialogResult.Abort:    displayLabel.Text="Abort was pressed";
                                      break;
          case DialogResult.Retry:    displayLabel.Text = "Retry was pressed";
                                      break;
          case DialogResult.Ignore:   displayLabel.Text = "Ignore was pressed";
                                      break;
          case DialogResult.Yes:      displayLabel.Text = "Yes was pressed";
                                      break;
          case DialogResult.No:       displayLabel.Text = "No was pressed";
                                      break;
       }
    }
  }
}
```

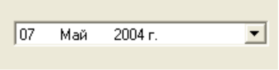
```
namespace WindowsApplication7
{
   partial class Form1
   {
      private System.ComponentModel.IContainer components = null;
      protected override void Dispose(bool disposing) { …   }

      #region Windows Form Designer generated code
      private void InitializeComponent()
      {   …
         this.retryCancelButton.CheckedChanged +=
              new System.EventHandler(this.buttonType_CheckedChanged);
         this.yesNoButton.CheckedChanged +=
               new System.EventHandler(this.buttonType_CheckedChanged);
         this.yesNoCancelButton.CheckedChanged +=
               new System.EventHandler(this.buttonType_CheckedChanged);
         this.abortRetryIgnoreButton.CheckedChanged +=
              new System.EventHandler(this.buttonType_CheckedChanged);
         this.okCancelButton.CheckedChanged +=
              new System.EventHandler(this.buttonType_CheckedChanged);
         this.okButton.CheckedChanged +=
              new System.EventHandler(this.buttonType_CheckedChanged)
```

```
         this.warningButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
         this.stopButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
         this.questionButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
         this.noneButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
         this.informationButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
         this.handButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
         this.errorButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
        this.asteriskButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
        this.exclamationButton.CheckedChanged +=
               new System.EventHandler(this.iconType_CheckedChanged);
        this.displayButton.Click +=
               new System.EventHandler(this.displayButton_Click)
      }
      #endregion
```

```
      private System.Windows.Forms.Label promptLabel;
      private System.Windows.Forms.GroupBox groupButtons;
      private System.Windows.Forms.RadioButton retryCancelButton;
      private System.Windows.Forms.RadioButton yesNoButton;
      private System.Windows.Forms.RadioButton yesNoCancelButton;
      private System.Windows.Forms.RadioButton abortRetryIgnoreButton;
      private System.Windows.Forms.RadioButton okCancelButton;
      private System.Windows.Forms.RadioButton okButton;
      private System.Windows.Forms.GroupBox groupIcons;
      private System.Windows.Forms.RadioButton informationButton;
      private System.Windows.Forms.RadioButton errorButton;
      private System.Windows.Forms.RadioButton asteriskButton;
      private System.Windows.Forms.RadioButton exclamationButton;
      private System.Windows.Forms.Button displayButton;
      private System.Windows.Forms.Label displayLabel;
      private System.Windows.Forms.RadioButton handButton;
      private System.Windows.Forms.RadioButton warningButton;
      private System.Windows.Forms.RadioButton stopButton;
      private System.Windows.Forms.RadioButton questionButton;
      private System.Windows.Forms.RadioButton noneButton;
   }
}
```

Control DateTimePicker

Allows the user to select a single item from a list of dates or times.

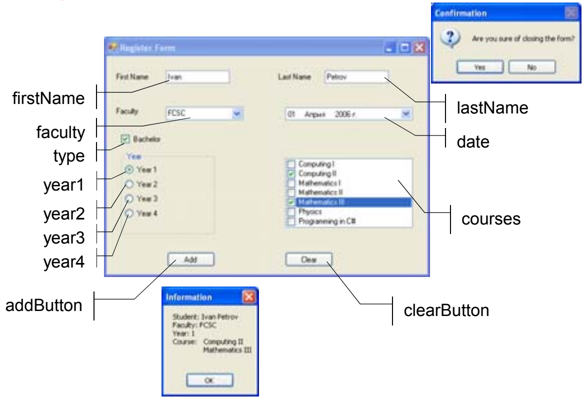


07 Май 2004 г.

Propertis

Value Gets/sets the date/time value assigned to the control

Example: Form enables students to choose courses



Are you sure of closing the form?

First Name: Ivan

Last Name: Petrov

Faculty: FCSC

01 Април 2006 г.

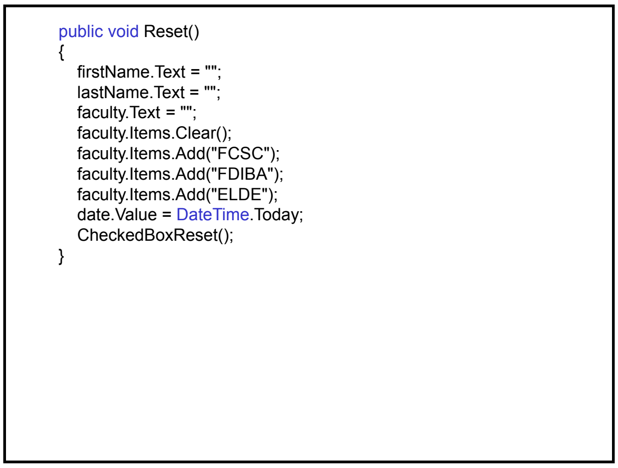
☑ Bachelor

● Year 1 ○ Year 2 ○ Year 3 ○ Year 4

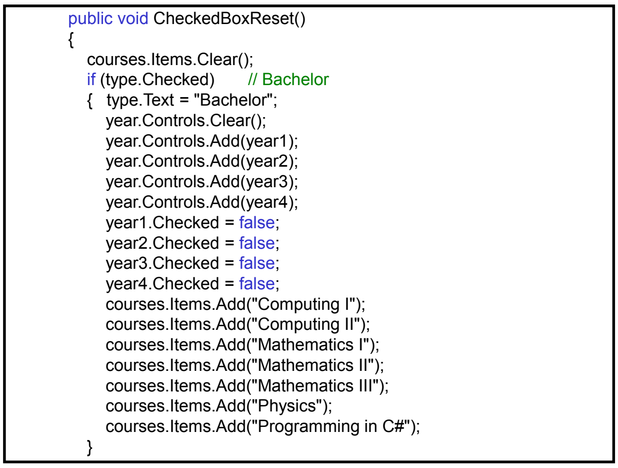
□ Computing I
☑ Computing II
□ Mathematics I
□ Mathematics II
☑ Mathematics III
□ Physics
□ Programming in C#

Student: Ivan Petrov
Faculty: FCSC
Year: I
Course: Computing II
Mathematics III

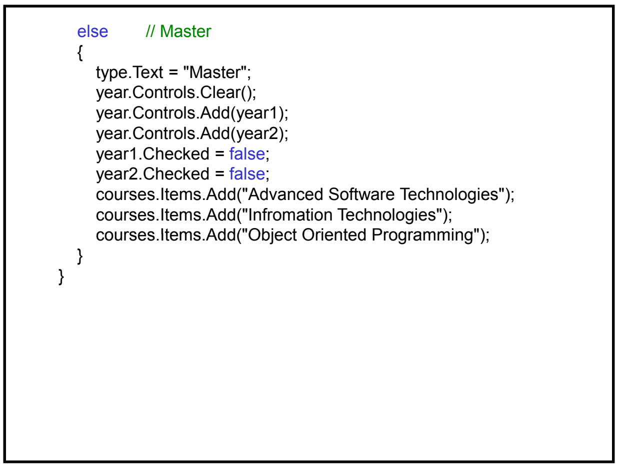
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication8
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         Reset();
      }
```



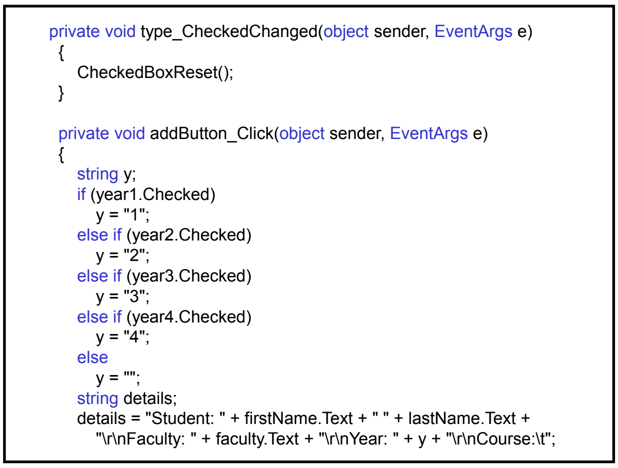
```
public void Reset()
{
    firstName.Text = "";
    lastName.Text = "";
    faculty.Text = "";
    faculty.Items.Clear();
    faculty.Items.Add("FCSC");
    faculty.Items.Add("FDIBA");
    faculty.Items.Add("ELDE");
    date.Value = DateTime.Today;
    CheckedBoxReset();
}
```



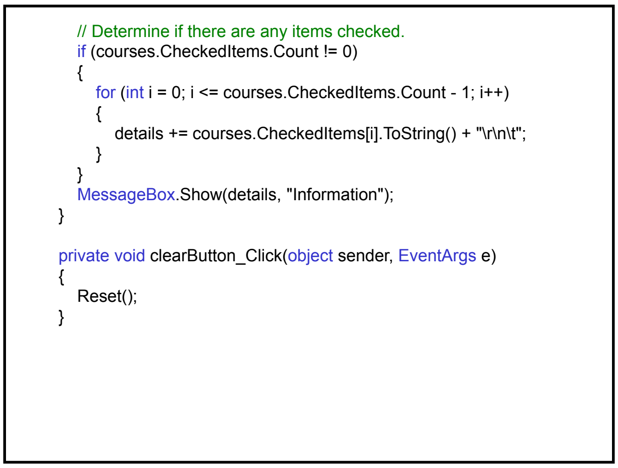
```
public void CheckedBoxReset()
{
    courses.Items.Clear();
    if (type.Checked)       // Bachelor
    {   type.Text = "Bachelor";
        year.Controls.Clear();
        year.Controls.Add(year1);
        year.Controls.Add(year2);
        year.Controls.Add(year3);
        year.Controls.Add(year4);
        year1.Checked = false;
        year2.Checked = false;
        year3.Checked = false;
        year4.Checked = false;
        courses.Items.Add("Computing I");
        courses.Items.Add("Computing II");
        courses.Items.Add("Mathematics I");
        courses.Items.Add("Mathematics II");
        courses.Items.Add("Mathematics III");
        courses.Items.Add("Physics");
        courses.Items.Add("Programming in C#");
    }
```



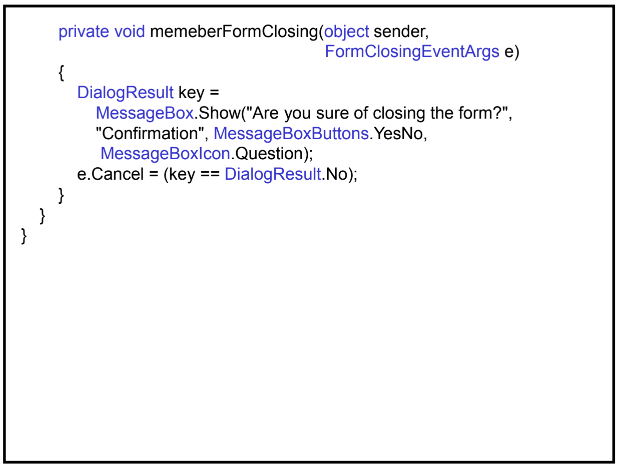
```
    else        // Master
    {
        type.Text = "Master";
        year.Controls.Clear();
        year.Controls.Add(year1);
        year.Controls.Add(year2);
        year1.Checked = false;
        year2.Checked = false;
        courses.Items.Add("Advanced Software Technologies");
        courses.Items.Add("Infromation Technologies");
        courses.Items.Add("Object Oriented Programming");
    }
}
```



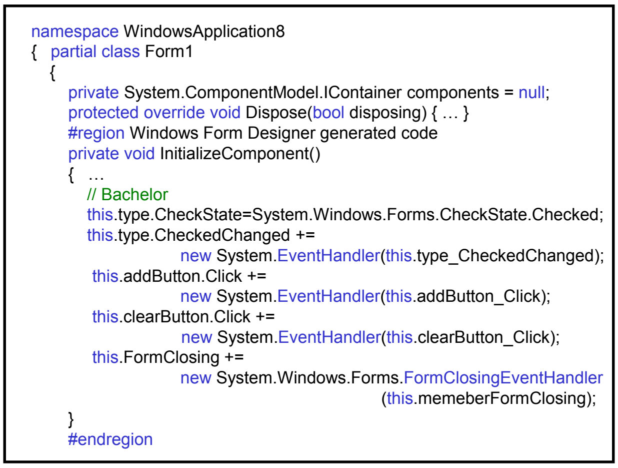
```
private void type_CheckedChanged(object sender, EventArgs e)
 {
    CheckedBoxReset();
 }

 private void addButton_Click(object sender, EventArgs e)
 {
    string y;
    if (year1.Checked)
        y = "1";
    else if (year2.Checked)
        y = "2";
    else if (year3.Checked)
        y = "3";
    else if (year4.Checked)
        y = "4";
    else
        y = "";
    string details;
    details = "Student: " + firstName.Text + " " + lastName.Text +
        "\r\nFaculty: " + faculty.Text + "\r\nYear: " + y + "\r\nCourse:\t";
```



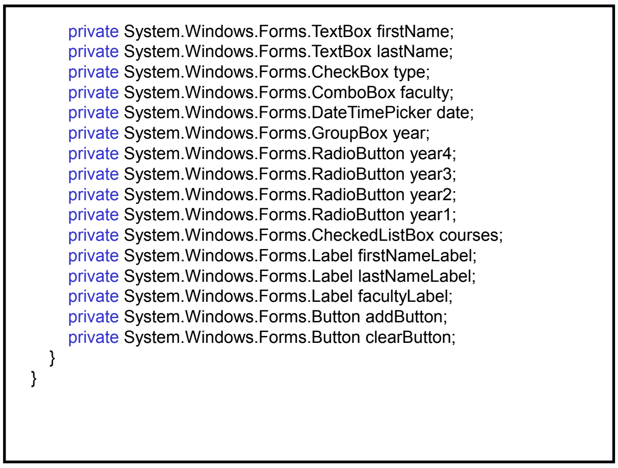
```
    // Determine if there are any items checked.
    if (courses.CheckedItems.Count != 0)
    {
        for (int i = 0; i <= courses.CheckedItems.Count - 1; i++)
        {
            details += courses.CheckedItems[i].ToString() + "\r\n\t";
        }
    }
    MessageBox.Show(details, "Information");
}

private void clearButton_Click(object sender, EventArgs e)
{
    Reset();
}
```



```
    private void memeberFormClosing(object sender,
                                            FormClosingEventArgs e)
    {
        DialogResult key =
            MessageBox.Show("Are you sure of closing the form?",
            "Confirmation", MessageBoxButtons.YesNo,
             MessageBoxIcon.Question);
        e.Cancel = (key == DialogResult.No);
    }
  }
}
```



```
namespace WindowsApplication8
{   partial class Form1
    {
       private System.ComponentModel.IContainer components = null;
       protected override void Dispose(bool disposing) { … }
       #region Windows Form Designer generated code
       private void InitializeComponent()
       {   …
          // Bachelor
          this.type.CheckState=System.Windows.Forms.CheckState.Checked;
          this.type.CheckedChanged +=
                         new System.EventHandler(this.type_CheckedChanged);
           this.addButton.Click +=
                         new System.EventHandler(this.addButton_Click);
           this.clearButton.Click +=
                         new System.EventHandler(this.clearButton_Click);
           this.FormClosing +=
                         new System.Windows.Forms.FormClosingEventHandler
                                                   (this.memeberFormClosing);
       }
       #endregion
```



```
            private System.Windows.Forms.TextBox firstName;
            private System.Windows.Forms.TextBox lastName;
            private System.Windows.Forms.CheckBox type;
            private System.Windows.Forms.ComboBox faculty;
            private System.Windows.Forms.DateTimePicker date;
            private System.Windows.Forms.GroupBox year;
            private System.Windows.Forms.RadioButton year4;
            private System.Windows.Forms.RadioButton year3;
            private System.Windows.Forms.RadioButton year2;
            private System.Windows.Forms.RadioButton year1;
            private System.Windows.Forms.CheckedListBox courses;
            private System.Windows.Forms.Label firstNameLabel;
            private System.Windows.Forms.Label lastNameLabel;
            private System.Windows.Forms.Label facultyLabel;
            private System.Windows.Forms.Button addButton;
            private System.Windows.Forms.Button clearButton;
        }
    }
```