



CHAPTER 4

C#

WINDOWS PROGRAMMING

Contents

2

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Contents

3

1. **Introduction**
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Introduction

4

□ Graphical User Interface(GUI)

- Allows the user to **interact visually** with a **program**

□ Building GUI

- GUI's are built from **GUI controls**
 - Also known as **components** or **widgets**
 - They are objects that can:
 - **Display information** on the screen, or
 - **Enable users to interact with an application** via the mouse, keyboard or other form of input
 - Implements **IComponent** interface
 - **Examples** - commonly used types of GUI controls
 - Label, TextBox, Button, CheckBox, ComboBox, ListBox

10/22/2024

□ Components of GUI

- **GUI controls** are components
 - Some have **graphical** representation
 - E.g., Form, Button, Label
 - Others do not
 - E.g., Timer

□ Look-and-Feel Observation

- **Consistent user interfaces** enable a user to learn new applications more quickly because the applications have the same **“look”** and **“feel”**.

Control	Description
Label	Displays images or uneditable text.
TextBox	Enables the user to enter data via the keyboard. It can also be used to display editable or uneditable text.
Button	Triggers an event when clicked with the mouse.
CheckBox	Specifies an option that can be selected (checked) or unselected (not checked).
ComboBox	Provides a drop-down list of items from which the user can make a selection either by clicking an item in the list or by typing in a box.
ListBox	Provides a list of items from which the user can make a selection by clicking an item in the list. Multiple elements can be selected.
Panel	A container in which controls can be placed and organized.
NumericUpDown	Enables the user to select from a range of input values.

□ GUI in MS Visual Studio for C#: Windows Form

- **Windows Forms** (or “**Forms**”) - used to create GUI's for C# programs
 - Create **graphical elements** that appear on the desktop (dialog, window, etc.)
 - A **Form** is a container for controls and components
 - **Active window** is the **front most window**

10/22/2024

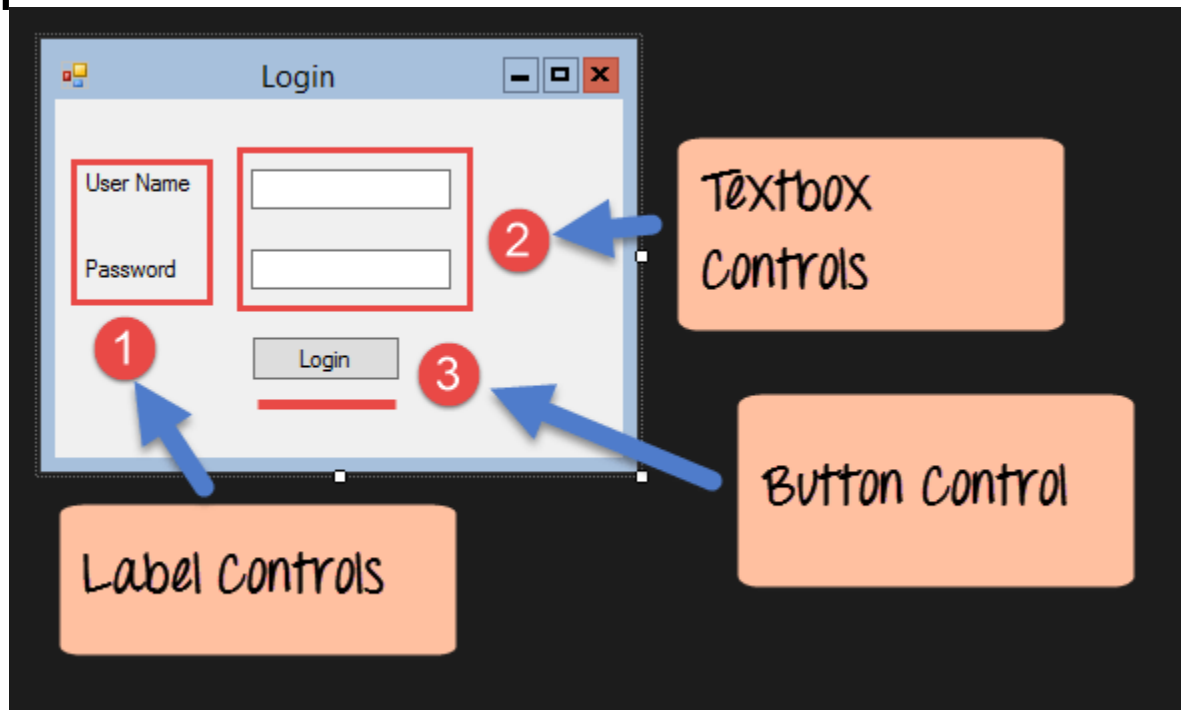
■ In visual programming **Visual Studio**

Example:

A simple Windows form application

8

- A simple Login screen, which is accessible by the user. The user will enter the required credentials and then will click the Login button to proceed



1. This is a collection of label controls which are normally used to describe adjacent controls.
 - So in our case, we have 2 textboxes, and the labels are used to tell the user that one textbox is for entering the user name and the other for the password.
2. The 2 textboxes are used to hold the username and password which will be entered by the user.
3. Finally, we have the button control.
 - The button control will normally have some code attached to perform a certain set of actions. So for example in the above case, we could have the button perform an action of validating the user name and password which is entered by the user.

Contents

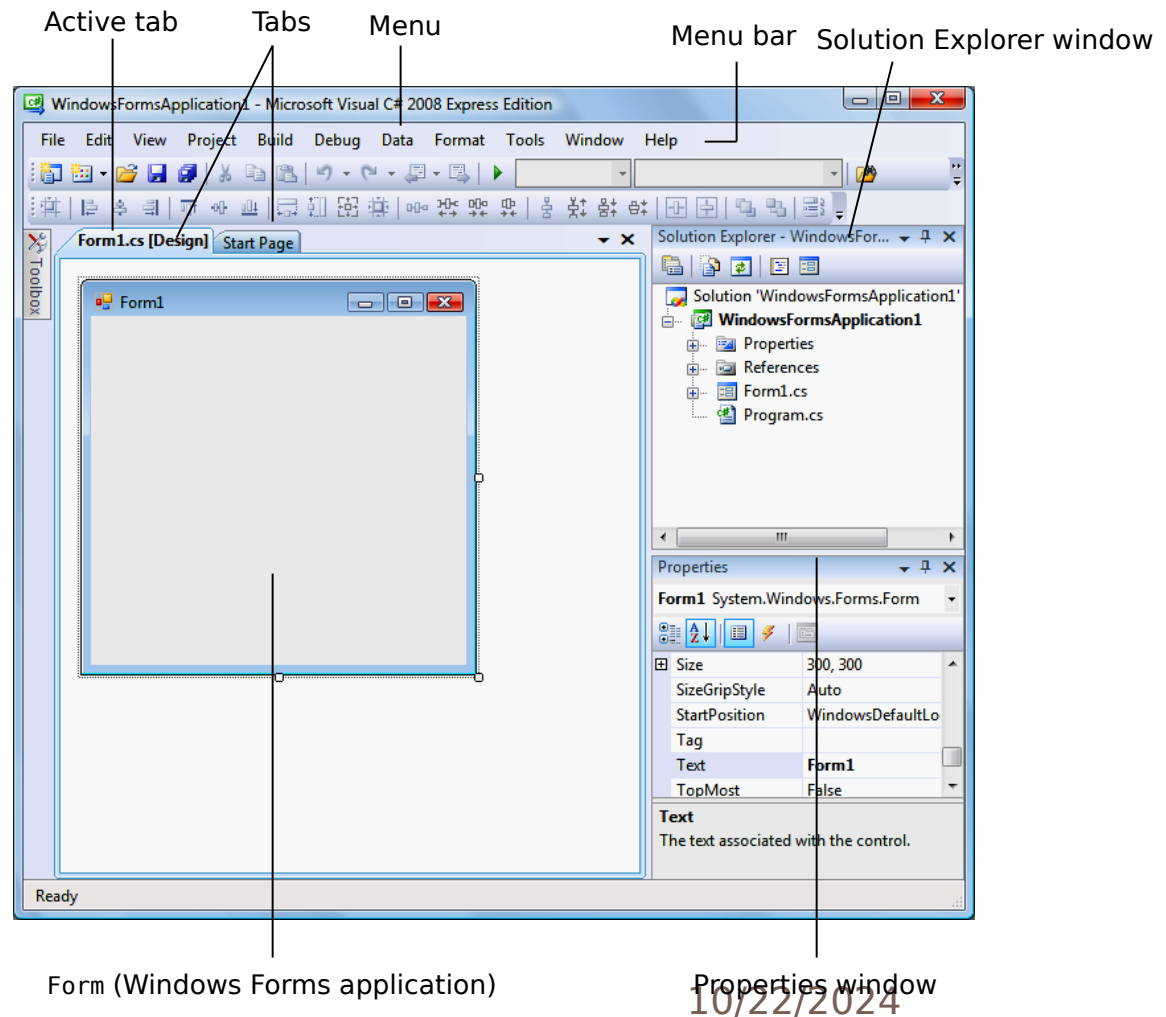
10

1. Introduction
2. **Visual Studio IDE overview**
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Visual Studio IDE Overview

11

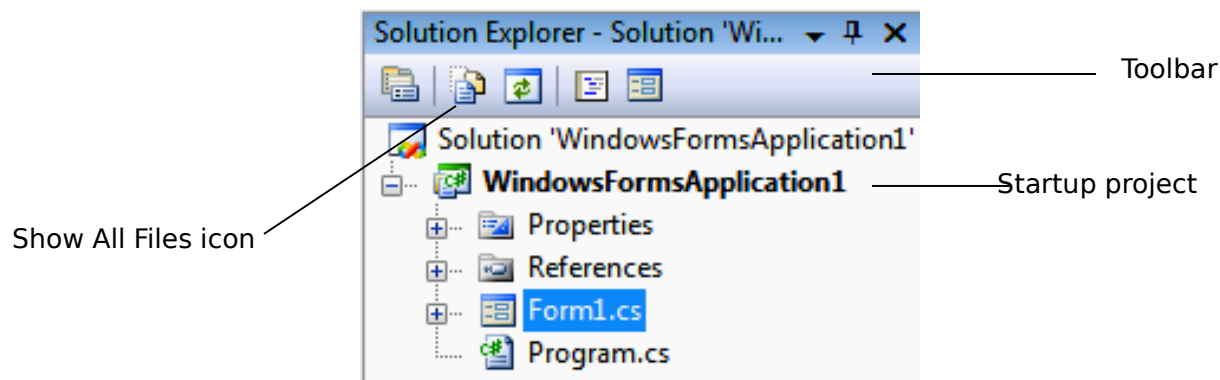
- The gray rectangle (called a **Form**) represents the main window of the application.



10/22/2024

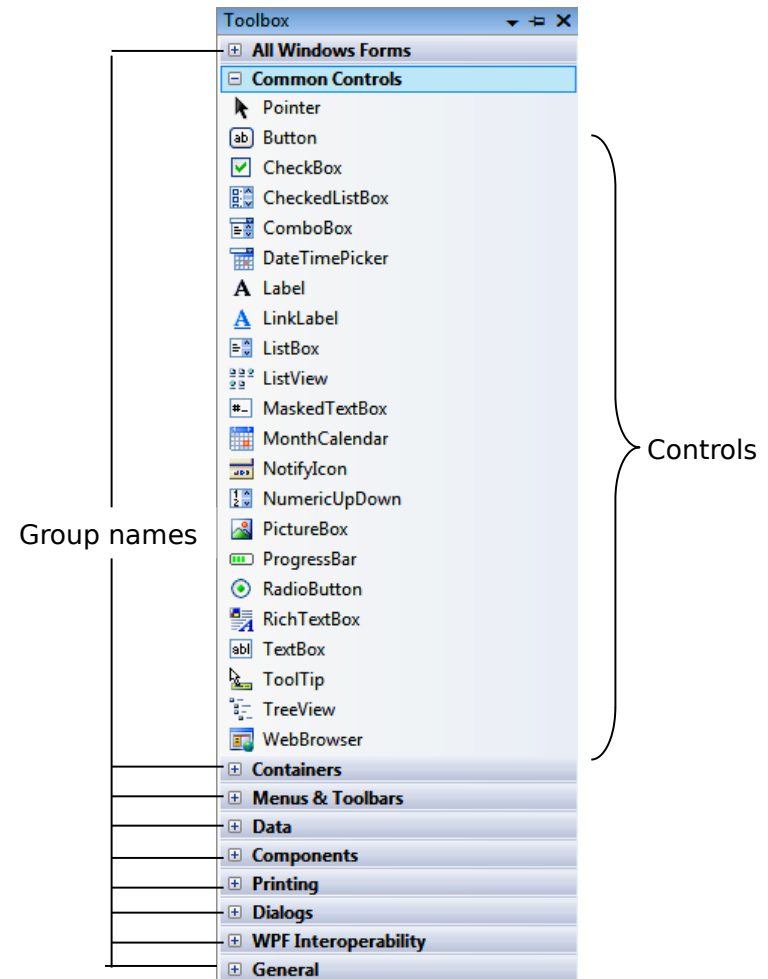
□ Solution Explorer

- The Solution Explorer window provides access to all of a solution's files.
- The solution's startup project runs when you select Debug ➤ Start Debugging.
- The file that corresponds to the Form is named Form1.cs. Visual C# files use the .cs file-name extension.



□ Toolbox

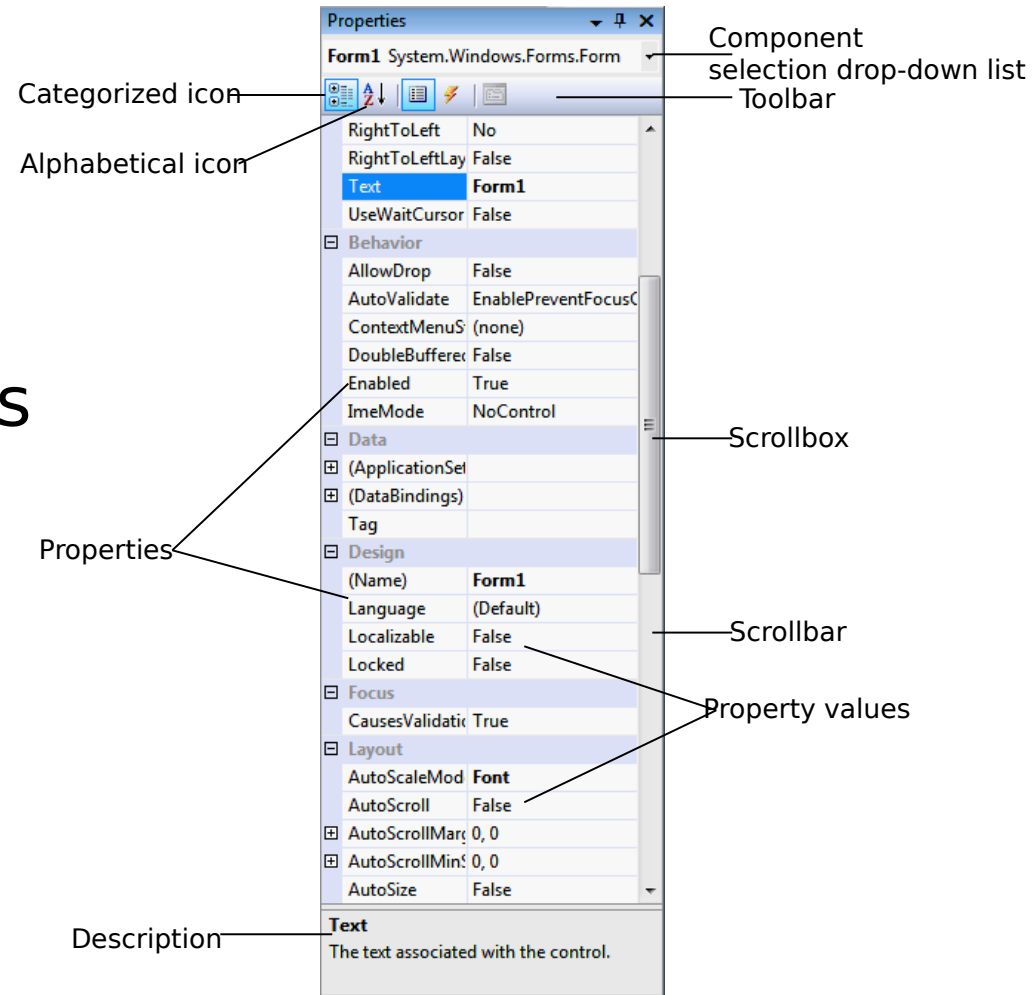
- The Toolbox contains icons representing controls used to customize Forms.
 - Groups the prebuilt controls into categories.
- To open Toolbox:
 - Menu: **View>>Toolbox:**
- To add a component to a Form:
 - Select that component in Toolbox
 - Drag it onto the Form



10/22/2024

□ Properties

- To display the Properties window, select View ▢ Properties Window.
- The Properties window allows you to modify a control's properties visually, without writing code



10/22/2024

Naming Controls

15

- In C#, **default names** for controls/components are:
 - button1, label1, textbox1, etc.
 - Not very descriptive (“generic”)
- Use better, **descriptive names**
 - Names to have meanings that make sense

Conventions for Naming Controls - Start the control name with...

16

□ **Control**

- Button
- TextBox
- ListBox
- Label
- SaveFileDialog

□ **Begin name with**

- btn
- txt
- lbox
- lbl
- sfd

□ **Examples of Naming Controls**

- A button used to calculate a total:
 - btnCalcTotal
- A textbox that allows a user to enter her name:
 - txtEnterName

10/22/2024

Contents

17

1. Introduction
2. Visual Studio IDE overview
3. **Drawing and Windows namespace**
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Namespace Drawing and Windows

18

□ **System.Drawing**

- This namespace provides many graphic data structures which are used throughout the GUI programming model
- It also provides support for low-level drawing operations
- These can be used to draw anything, not just what is offered by the pre-built controls

□ **System.Drawing.Point**

- Structure which represents a 2-D point
- Constructor
 - `Point(int x, int y)`
- Properties
 - X – get/set of X coordinate
 - Y – get/set of Y coordinate

□ **System.Drawing.Size**

- Structure which stores the width and height of something
- Constructor
 - `Size(int width, int height)`
- Properties
 - `Width` – get/set width
 - `Height` – get/set height

□ **System.Windows.Forms**

- This namespace contains all of the controls used on the average Windows interface
- A control is a higher-level object composed of
 - A window in which the control is drawn
 - Visual parts of the control which are drawn in the window
 - A set of delegates which are triggered when various events occur

□ **Form Class**

- This is the top-level window class
- This class contains all other controls
- Normally, your top-level form inherits from the Form class
- Although the class has numerous methods, most of the time you interact with it via properties and delegates

Contents

23

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. **Creating windows applications**
5. Event handling
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Creating Windows Applications

24

- Two ways of building GUI application:
 1. Using a **text editor**
 2. Using **designer in Visual Studio**
- **Example**
 - In creating a GUI application we will use
 - **Application** – a class with static methods to control operation of an application
 - **Label** – a widget that can display static text or an image
 - **Button** – a push button with a textual or image displayed. Able to respond to mouse clicks.

Creating Windows Applications: **Text Editor**

25



10/22/2024

Simple steps to create windows application

26

- Create a **class** and **Inherit** the **Form** class
 - **Fields**
 - Declares the widgets
 - **Constructor**
 - Create the widgets and set their properties
 - Location, text, size, fore color, ..., including event handlers
 - Add the widgets into the form
 - **Methods**
 - Create the event handler methods
 - Create a Main method to your application

10/22/2024

...Text Editor

27

- The first step is to create a class which
 - **Inherits** from **Form**
 - **Declares the widgets** within it

```
public class GreetingForm : Form {  
    Label    greetingLabel;  
    Button   cancelButton;  
  
    ...  
}
```

...Text Editor

28

- Next, create the label and set its properties

```
greetingLabel = new Label();  
greetingLabel.Location = new Point(16, 24);  
greetingLabel.Text = "Hello, World";  
greetingLabel.Size = new Size(216, 24);  
greetingLabel.ForeColor = Color.Black;
```

...Text Editor

29

- Create the cancel button and set its properties

```
cancelButton = new Button();  
cancelButton.Location = new Point(150, 200);  
cancelButton.Size = new Size(112, 32);  
cancelButton.Text = "&Cancel";  
cancelButton.Click += new  
    EventHandler(cancelButton_Click);
```

- Set the properties of the main form

```
this.AutoScaleDimensions = new.SizeF(95.0f, 95.0f);  
this.ClientSize = new Size(300, 300);  
this.Text = "My First Windows App";
```

...Text Editor

30

- Add the controls to the form

```
this.Controls.Add(cancelButton);
this.Controls.Add(greetingLabel);
```
- And provide the event handler

```
protected void cancelButton_Click(
    object sender, EventArgs e) {
    Application.Exit();
}
```
- And Main method

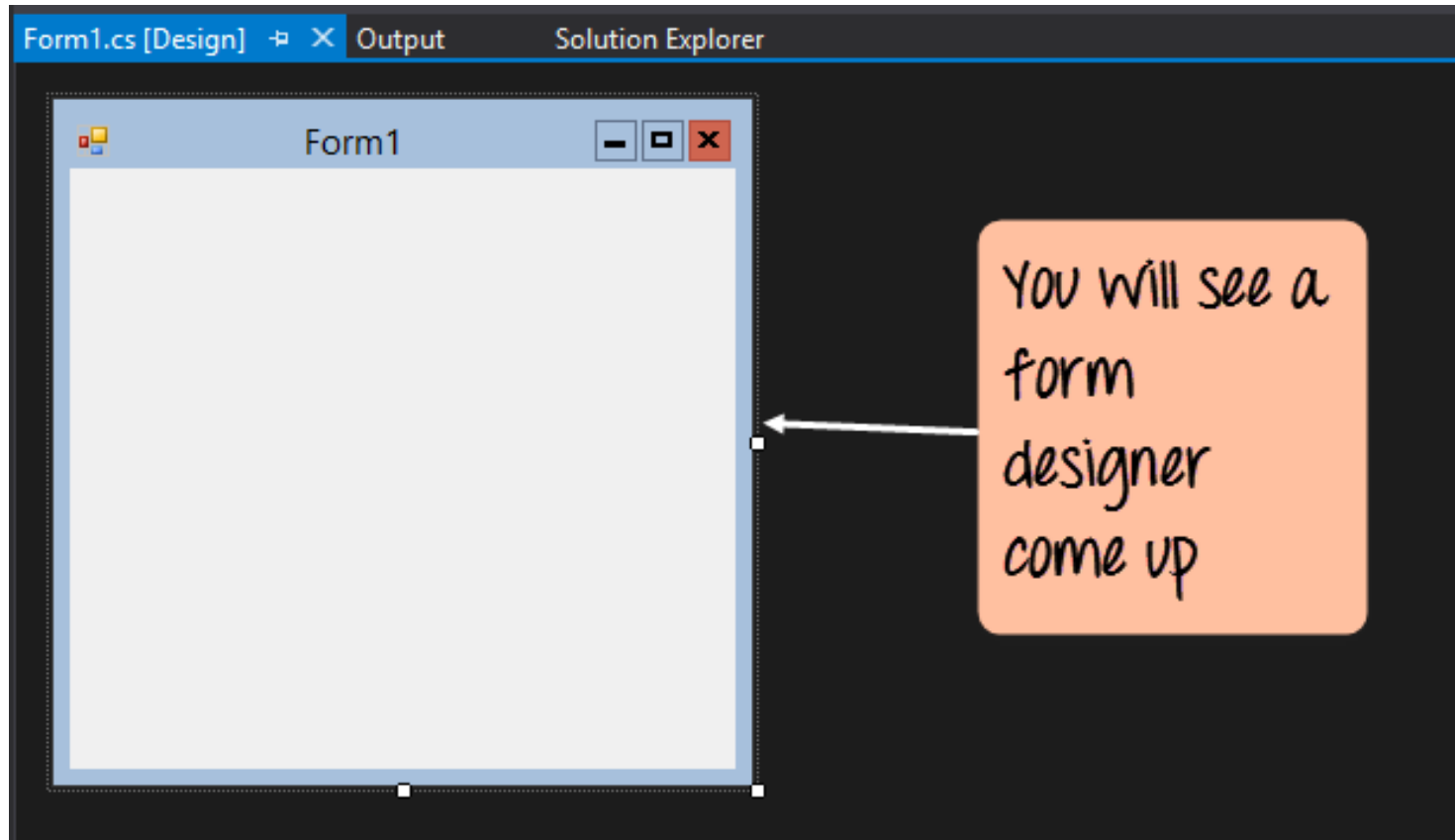
```
Static void Main() {
    Application.Run(new GreetingForm());
}
```

Visual Studio Designer

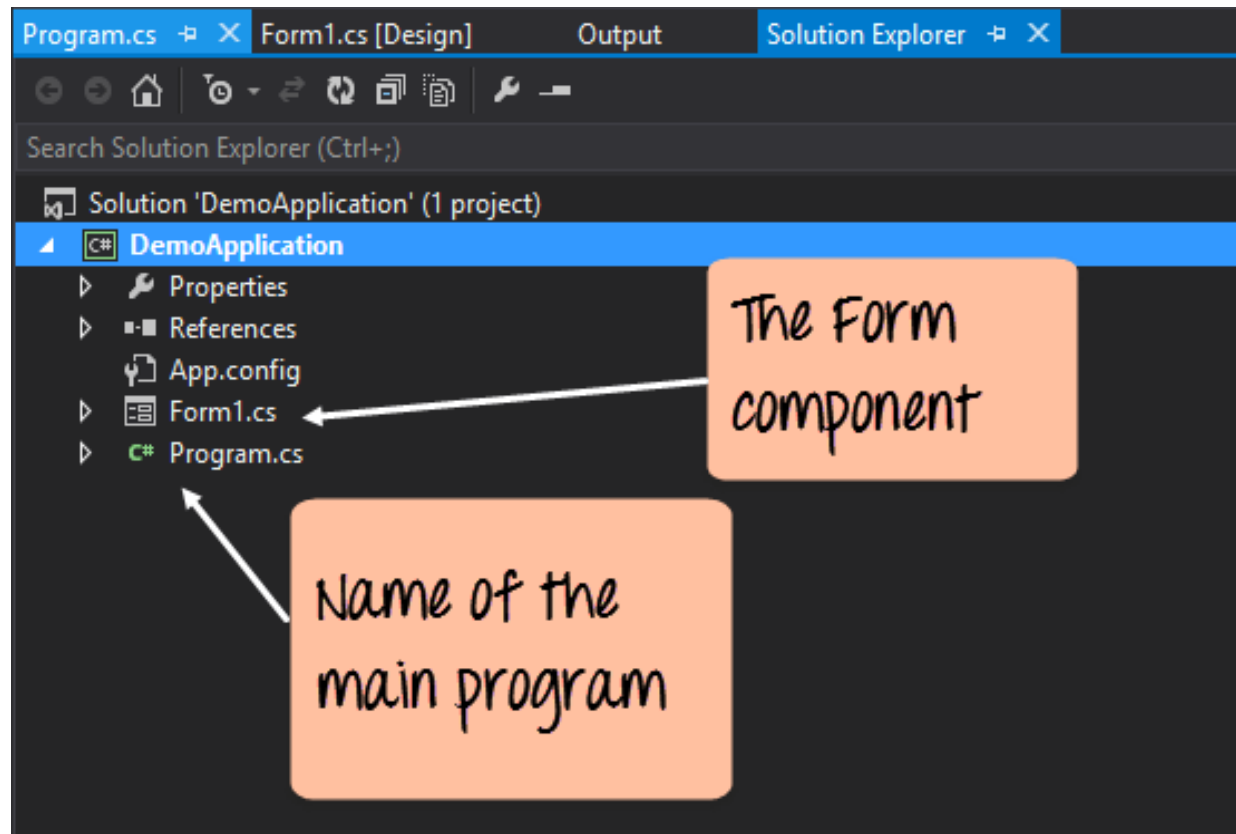
31

- This is a drag and drop interface for drawing a GUI
 - **The code is automatically generated**
- You can hook event handlers onto the events and write the code for them
 - It speeds writing code
 - **You cannot make major modifications to the code it generates**

- **Example:** A simple 'hello world' application in Visual Studio.
 1. After launching Visual Studio, **create a new Project.**
 2. The next step is to choose the project type as a **Windows Forms application.**
 - Here we also need to mention the name and location of our project.
 - Give a project name of **DemoApplication**
 3. Click the '**OK**' button to let Visual Studio create our project.



- You will see a **Form Designer** displayed in Visual Studio.
 - It's in this Form Designer that you will start building your Windows Forms application.



Creating Windows Applications: **Visual Studio Designer**

35

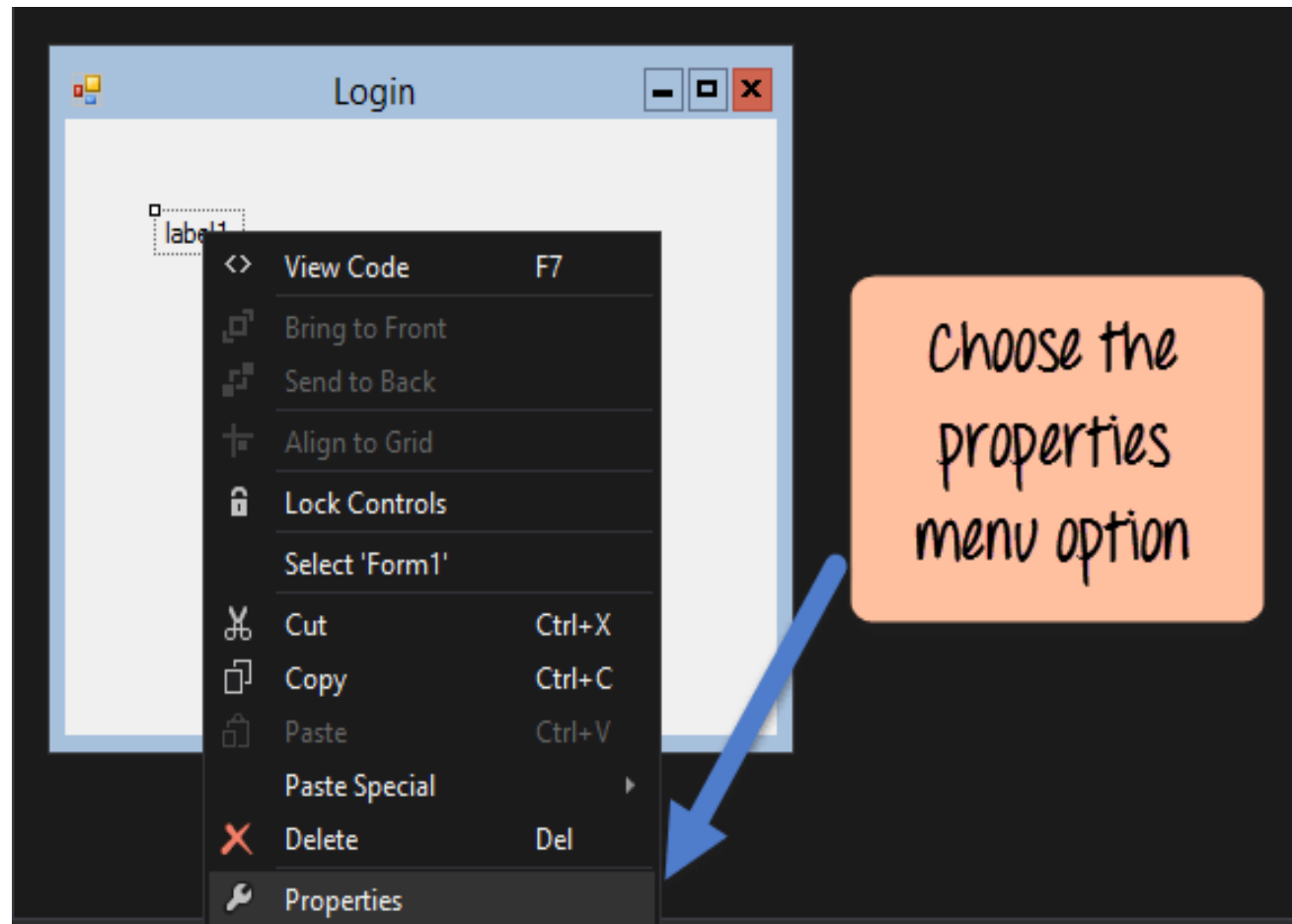


10/22/2024

... Visual Studio Designer

36

- The first step is to **add a label** to the Form which will display "Hello World."
 - From the toolbox, you will need to choose the Label control and **simply drag it onto the Form**.
- The next step is to go to the properties of the control and Change the text to 'Hello Ethiopia'.
 - To go to the properties of a control, you need to right-click the control and choose the Properties menu option and enter **"Hello Ethiopia"**.
 - Each Control has a set of properties which describe the control.



... Visual Studio Designer

38

- Then add a button to the Form
 - From the toolbox, you will need to choose the Button control and **drag it onto the Form.**
 - Go to the properties of the button and enter **“Cancel”.**
- Double click the button and write the following code:
 - **Application.Exit();**

- Class Work : Design the form below and write

Calculator

Operand 1:

Operand 2:

Answer:

Another Look at the Visual Studio Generated Code

40

- Visual Studio Generated Code
 - The auto-generated code is saved in the **Designer.cs** file of the Form
 - **partial** modifier allow the class created **to be split among multiple files**
 - By default, all variable declarations for controls created through C# have a private access modifier
 - The code also includes **Dispose** and **InitializeComponent**

Visual Studio generated code file

41

```
namespace WindowsFormsApplication1
{
    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);
        }

        Windows Form Designer generated code

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button button1;
    }
}
```

```

#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.label1 = new System.Windows.Forms.Label();
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(39, 31);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(72, 13);
    this.label1.TabIndex = 0;
    this.label1.Text = "Hello Ethiopia";
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(86, 173);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 1;
    this.button1.Text = "Cancel";
    this.button1.UseVisualStyleBackColor = true;
    //
    // Form1
    //
    this.AutoScaleMode = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(284, 262);
    this.Controls.Add(this.button1);
    this.Controls.Add(this.label1);
    this.Name = "Form1";
    this.Text = "My First Windows";
    this.ResumeLayout(false);
    this.PerformLayout();
}
#endregion

```

Contents

43

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. **Event handling**
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Event Handling

44

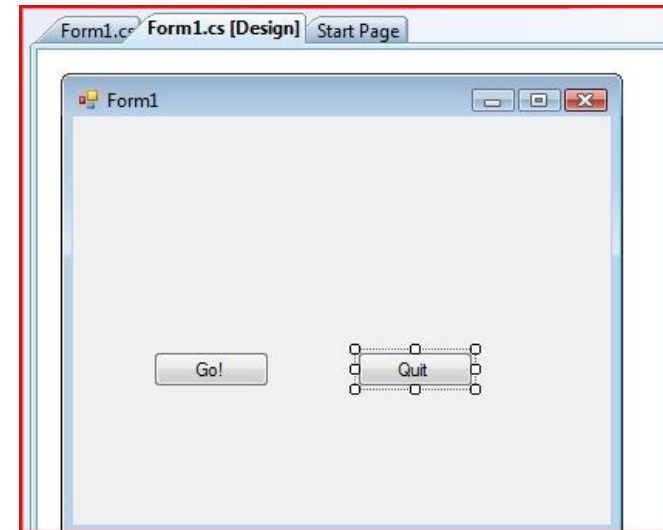
- When a user interacts with a form, this causes an event to occur.
 - E.g., clicking a button, typing in a textbox, etc. are events
- Events signal that certain code should be run to perm some actions
 - **Event Handler**: method that runs after an event occurs
 - **Event Handling**: the overall process of responding to events
- All GUI controls have associated events

Creating an Event Handler

45

- Creating an event handler by double clicking a control
 - By double clicking a control, the Form creates an event handler for that control(default)
- We can also create additional event handlers through the **Properties** window

- Event handler for clicking a Button
- Double-click on the Quit button
 - You will be switched to the **Code View**
 - The following method appears in the code generated for the form:



```
private void btnQuit_Click(object sender, EventArgs e)
{
    code to be written by you can be added here (to be shown later)
}
```

Contents

47

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. **Control properties and layout**
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. Group assignment

Control Properties and Layout

48

□ **Control Properties and Layout**

- **Focus** method
 - Transfers the focus to a control and makes it the active control
- **Enabled** property
 - Indicates whether the user can interact with a control to generate an event
- **Anchoring** property
 - Causes controls to remain at a fixed distance from the sides of the container
- **Docking** property
 - Attaches a control to a container such that the control stretches across an entire side
- **Padding** property
 - Specifies the distance between the docked controls and the Form edges
- **Width** and **Height**
 - Specifies size of Form

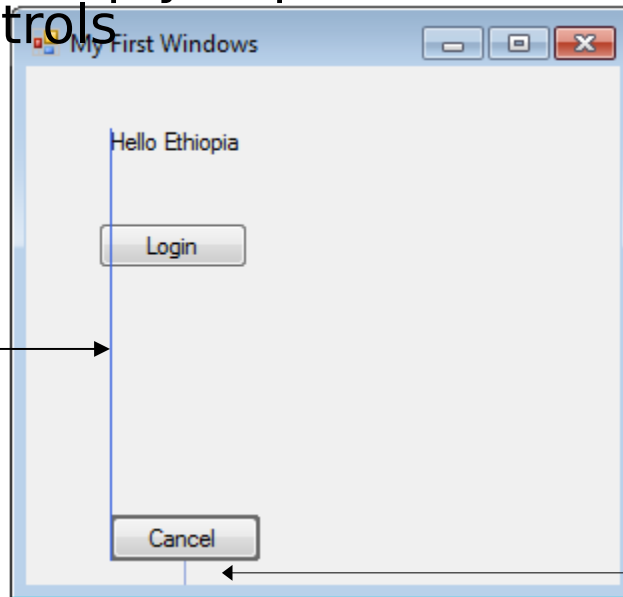
10/22/2024

□ Using Visual Studio To Edit GUI's Layout

- **Snap lines**

- Appear to help you position the control with respect to other controls

Snap line to help align controls on their left sides



Snap line that indicates when a control reaches the minimum recommended distance from the edge of a Form.

Class **Control** properties and methods

50

Class Control properties and methods	Description
<i>Common Properties</i>	
BackColor	The control's background color.
BackgroundImage	The control's background image.
Enabled	Specifies whether the control is enabled (i.e., if the user can interact with it). Typically, portions of a disabled control appear "grayed out" as a visual indication to the user that the control is disabled.
Focused	Indicates whether the control has the focus.
Font	The Font used to display the control's text.
ForeColor	The control's foreground color. This usually determines the color of the text in the Text property.
TabIndex	The tab order of the control. When the <i>Tab</i> key is pressed, the focus transfers between controls based on the tab order. You can set this order.
TabStop	If true , then a user can give focus to this control via the <i>Tab</i> key.

TabStop	If true , then a user can give focus to this control via the <i>Tab</i> key.
Text	The text associated with the control. The location and appearance of the text vary depending on the type of control.
Visible	Indicates whether the control is visible.
<i>Common Methods</i>	
Focus	Acquires the focus.
Hide	Hides the control (sets the Visible property to false).
Show	Shows the control (sets the Visible property to true).

Control layout properties

52

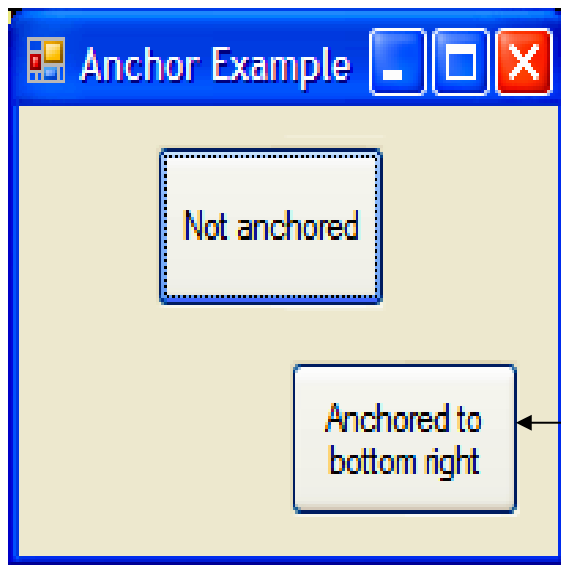
Control layout properties	Description
Anchor	Causes a control to remain at a fixed distance from the side(s) of the container even when the container is resized.
Dock	Allows a control to span one side of its container or to fill the entire container.
Padding	Sets the space between a container's edges and docked controls. The default is 0, causing the control to appear flush with the container's sides.
Location	Specifies the location (as a set of coordinates) of the upper-left corner of the control, in relation to its container.
Size	Specifies the size of the control in pixels as a <code>Size</code> object, which has properties <code>Width</code> and <code>Height</code> .
MinimumSize, MaximumSize	Indicates the minimum and maximum size of a <code>Control</code> , respectively.

10/22/2024

Anchoring demonstration

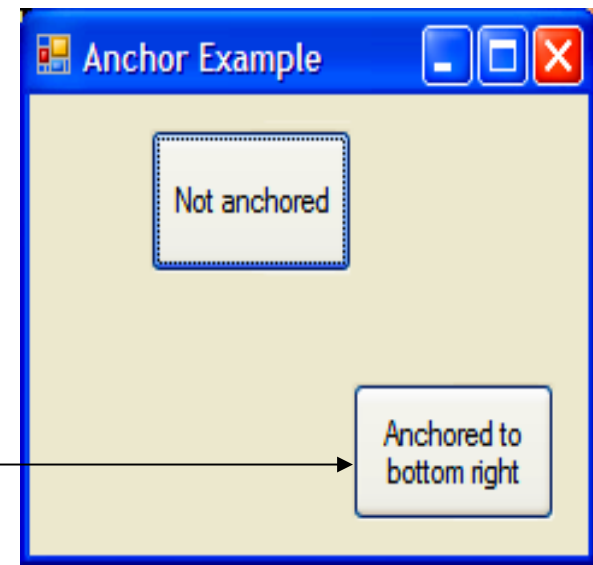
53

Before
resizing



Constant
distance to
right and
bottom sides

After resizing

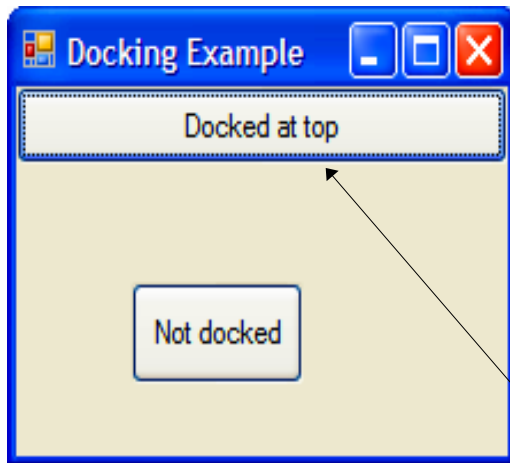


10/22/2024

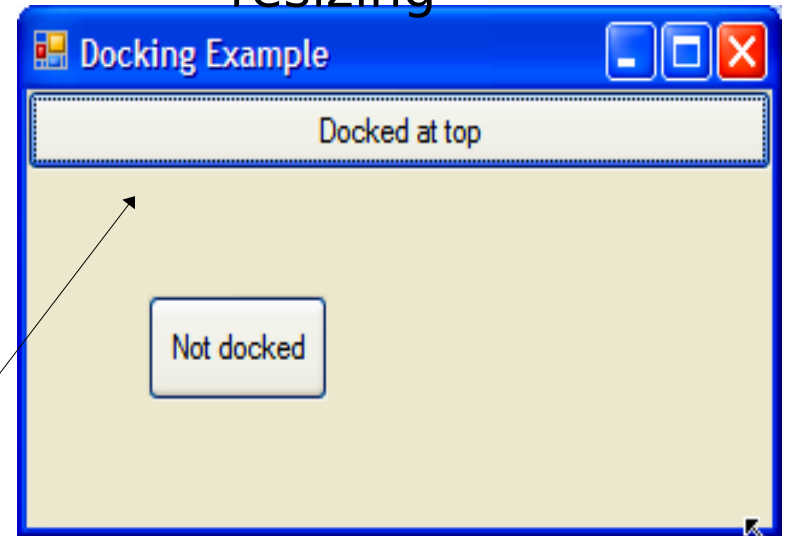
Docking a Button to the top of a form

54

Before resizing



After
resizing



Control
extends along
entire top
portion of form

10/22/2024

Contents

55

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. **Lists of controls**
8. Mouse and Keyboard Event Handling
9. Group assignment

Lists of Controls

56

1. Labels

- Provide text information (as well as images)
- Display text that user cannot directly modify
- Can be changed programmatically

2. TextBoxes

- Area in which either text can be displayed or typed in
- *Password* TextBoxes hides information entered by user

3. Buttons

- Control that user clicks to trigger specific action
- There are several types of buttons, such as *checkboxes* and *radio buttons*
- All buttons derive from class `ButtonBase`

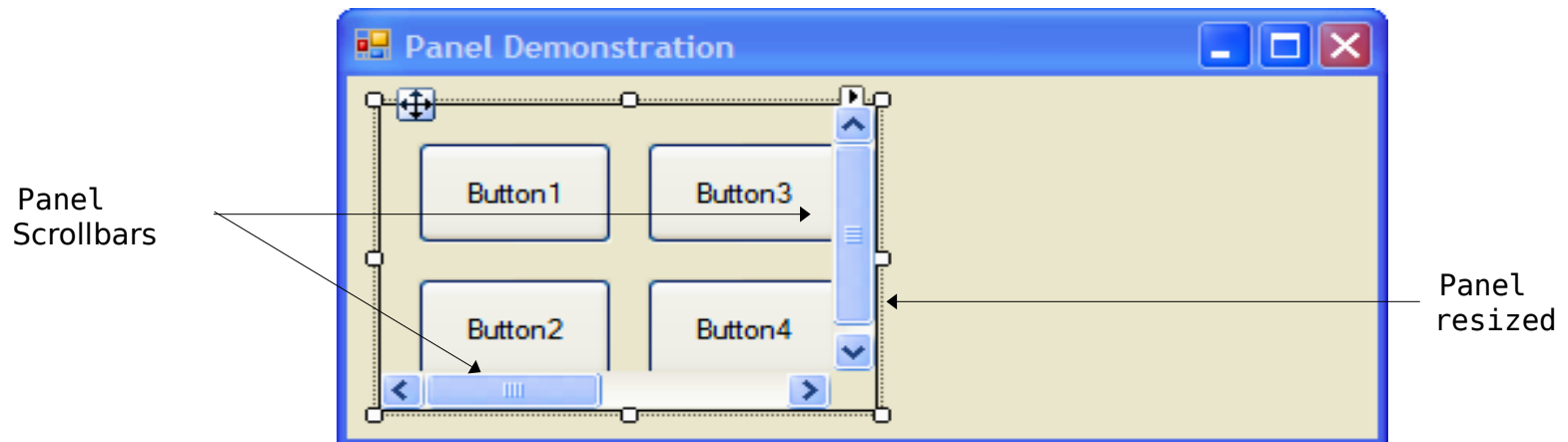
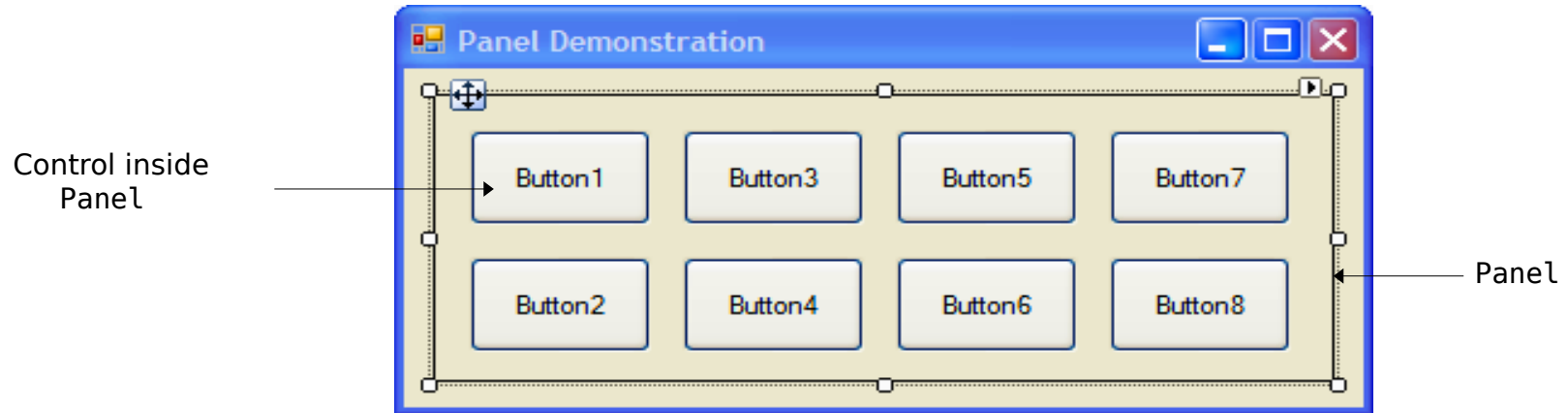
4. GroupBoxes and Panels

- Arrange controls on a GUI
- Used to **group similar functionality** that are related
- Primary difference between these two controls:
 - **GroupBoxes** can **display a caption** (i.e., text) and **do not include scrollbars**
 - **Panels** can **include scrollbars** and **do not include a caption**
- Panels and GroupBoxes can contain other Panels and GroupBoxes for more complex layouts

- We can organize a GUI by anchoring and docking controls inside a GroupBox or Panel.
 - The GroupBox or Panel then can be anchored or docked inside a Form.
 - This divides controls into functional “groups” that can be arranged easily.
- Use Panels with scrollbars to avoid cluttering a GUI and to reduce the GUI’s size.

Creating a Panel with scrollbars

59



10/22/2024

5. C# has two types of state buttons

- **CheckBoxes**

- Small squares that either is blank or contains a check mark
- **Any number of CheckBoxes can be selected at a time**

- **RadioButtons**

- A RadioButton control provides a round interface to select one option from a number of options.
 - Radio buttons are usually placed in a group on a container control, such as a Panel or a GroupBox, and one of them is selected
- Selecting one RadioButton in the group forces all the others to be deselected.
 - RadioButtons represents a set of mutually exclusive options

6. **Picture Boxes**

- Display an image

7. **ToolTips**

- Helpful text that appears when the mouse hovers over an item

8. **NumericUpDown**

- Restrict a user's input choices to a specific range of numeric values.
- Appears as a **TextBox**, with two small Buttons on the right side
- NumericUpDown's ReadOnly property indicates if user can type a number into the control

9. **List Boxes**

- A ListBox control provides a user interface to display a list of items. Users can select one or more items from the list.

10. Combo Boxes

- Combines TextBox features with a drop-down-list

11. Menu Controls

- MainMenu is the container for the Menu structure of the form and menus are made of MenuItem objects that represent individual parts of a menu.

12. DateTimePicker Control

- The DateTimePicker control allows you to display and collect date and time from the user with a specified format.

13. Tree view

- Displays nodes hierarchically in a tree

14. List view

- Similar to a ListBox in that both display lists from which the user can select one or more items. ListView is more versatile and can display items in different formats

Contents

64

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. Lists of controls
8. **Mouse and Keyboard Event Handling**
9. Group assignment

Mouse and Keyboard Event Handling

65

□ **Mouse-Event Handling**

- Mouse events can be handled for any control that derives from class `System.Windows.Forms.Control`
- Class **MouseEventArgs**
 - Contains information related to the mouse event
 - Information about the event is passed to the event-handling method through an object of this class
- The delegate used to create the mouse-event handlers is **EventHandler**

10/22/2014

Mouse events and event arguments

66

Mouse events and event arguments

Mouse Events with Event Argument of Type EventArgs

MouseEnter Occurs when the mouse cursor enters the control's boundaries.

MouseLeave Occurs when the mouse cursor leaves the control's boundaries.

Mouse Events with Event Argument of Type MouseEventArgs

MouseDown Occurs when a mouse button is pressed while the mouse cursor is within a control's boundaries.

MouseHover Occurs when the mouse cursor hovers within the control's boundaries.

MouseMove Occurs when the mouse cursor is moved while in the control's boundaries.

MouseUp Occurs when a mouse button is released when the cursor is over the control's boundaries.

*Class **MouseEventArgs** Properties*

Button	Specifies which mouse button was pressed (Left , Right , Middle or none).
Clicks	The number of times that the mouse button was clicked.
X	The <i>x</i> -coordinate within the control where the event occurred.
Y	The <i>y</i> -coordinate within the control where the event occurred.

□ **Keyboard-Event Handling**

- Key events occur when keyboard keys are pressed and released
- There are three key events:
 - **KeyPress**
 - The event occurs when the user presses a key that represents an ASCII character
 - **KeyUp** and **KeyDown**
 - If information about the modifier keys are important, use the KeyUp or KeyDown events

Keyboard events and events arguments

69

Keyboard events and event arguments

*Key Events with Event Arguments of Type **KeyEventArgs***

KeyDown Generated when a key is initially pressed.

KeyUp Generated when a key is released.

*Key Event with Event Argument of Type **KeyPressEventArgs***

KeyPress Generated when a key is pressed.

*Class **KeyPressEventArgs** Properties*

KeyChar Returns the ASCII character for the key pressed.

Handled Indicates whether the **KeyPress** event was handled.

*Class **KeyEventArgs** Properties*

Alt Indicates whether the *Alt* key was pressed.

Control Indicates whether the *Ctrl* key was pressed.

Shift Indicates whether the *Shift* key was pressed.

Handled Indicates whether the event was handled.

KeyCode	Returns the key code for the key as a value from the Keys enumeration. This does not include modifier-key information. It is used to test for a specific key.
KeyData	Returns the key code for a key combined with modifier information as a Keys value. This property contains all information about the pressed key.
KeyValue	Returns the key code as an int , rather than as a value from the Keys enumeration. This property is used to obtain a numeric representation of the pressed key. The int value is known as a Windows virtual key code.
Modifiers	Returns a Keys value indicating any pressed modifier keys (<i>Alt</i> , <i>Ctrl</i> and <i>Shift</i>). This property is used to determine modifier-key information only.

Contents

71

1. Introduction
2. Visual Studio IDE overview
3. Drawing and Windows namespace
4. Creating windows applications
5. Event handling
6. Control properties and layout
7. Lists of controls
8. Mouse and Keyboard Event Handling
9. **Group assignment**

Group Assignment

72

□ **Team Organization**

- A team of 3/4 students
- Submit team members list including name, Id and email

□ **Purpose**

- The purpose of the assignment is to help you learn the windows programming controls in detail.

□ **Submission**

- Document
- Demo

□ **Due date**

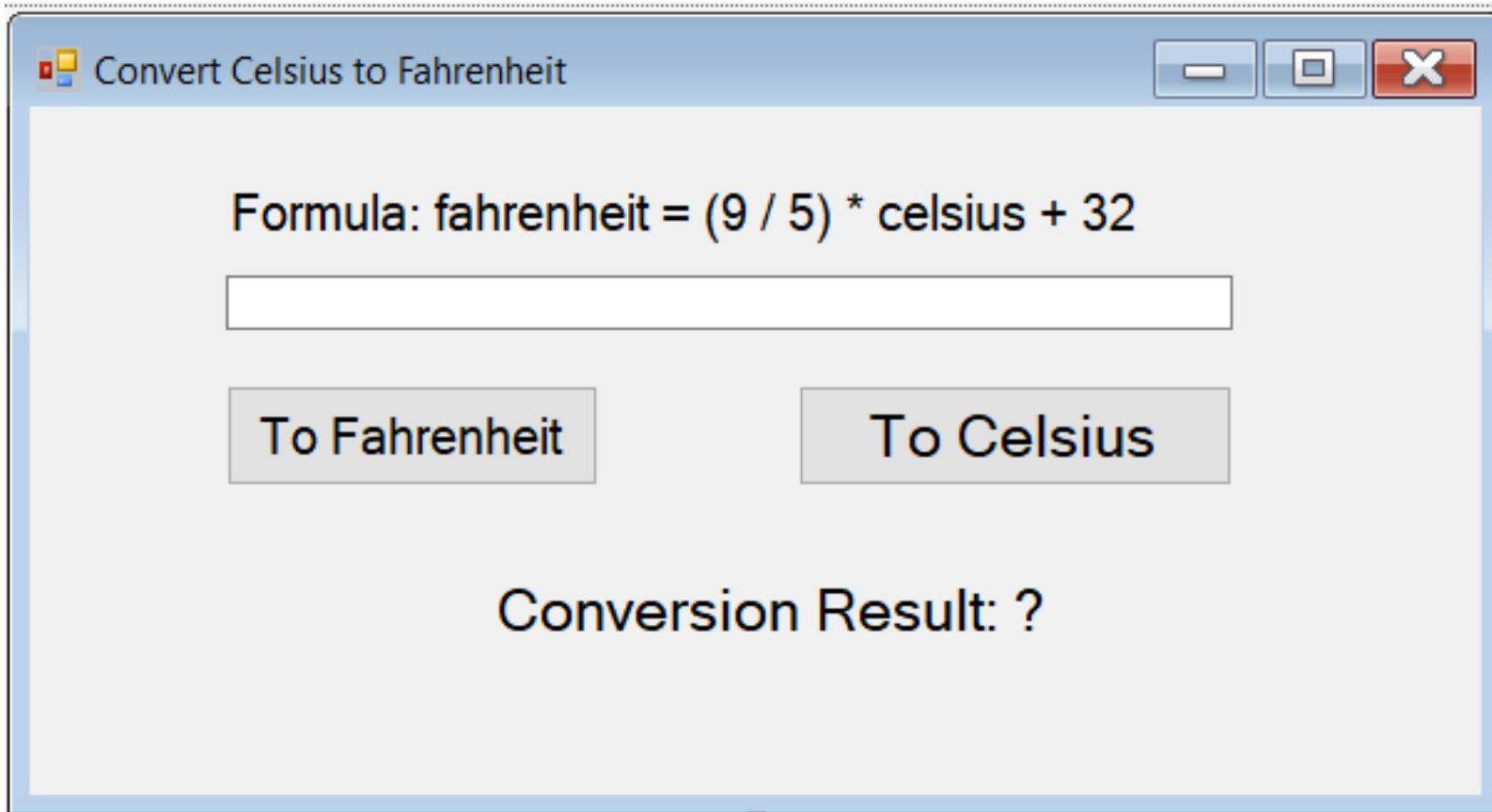
- December 09, 2019

□ **Assignment Title**

- ?

Exercise: 1

73



Convert Celsius to Fahrenheit

Formula: $\text{fahrenheit} = (9 / 5) * \text{celsius} + 32$

To Fahrenheit To Celsius

Conversion Result: ?

Questions?

Thank You