Windows Forms kullanarak Graphical User Interface - II

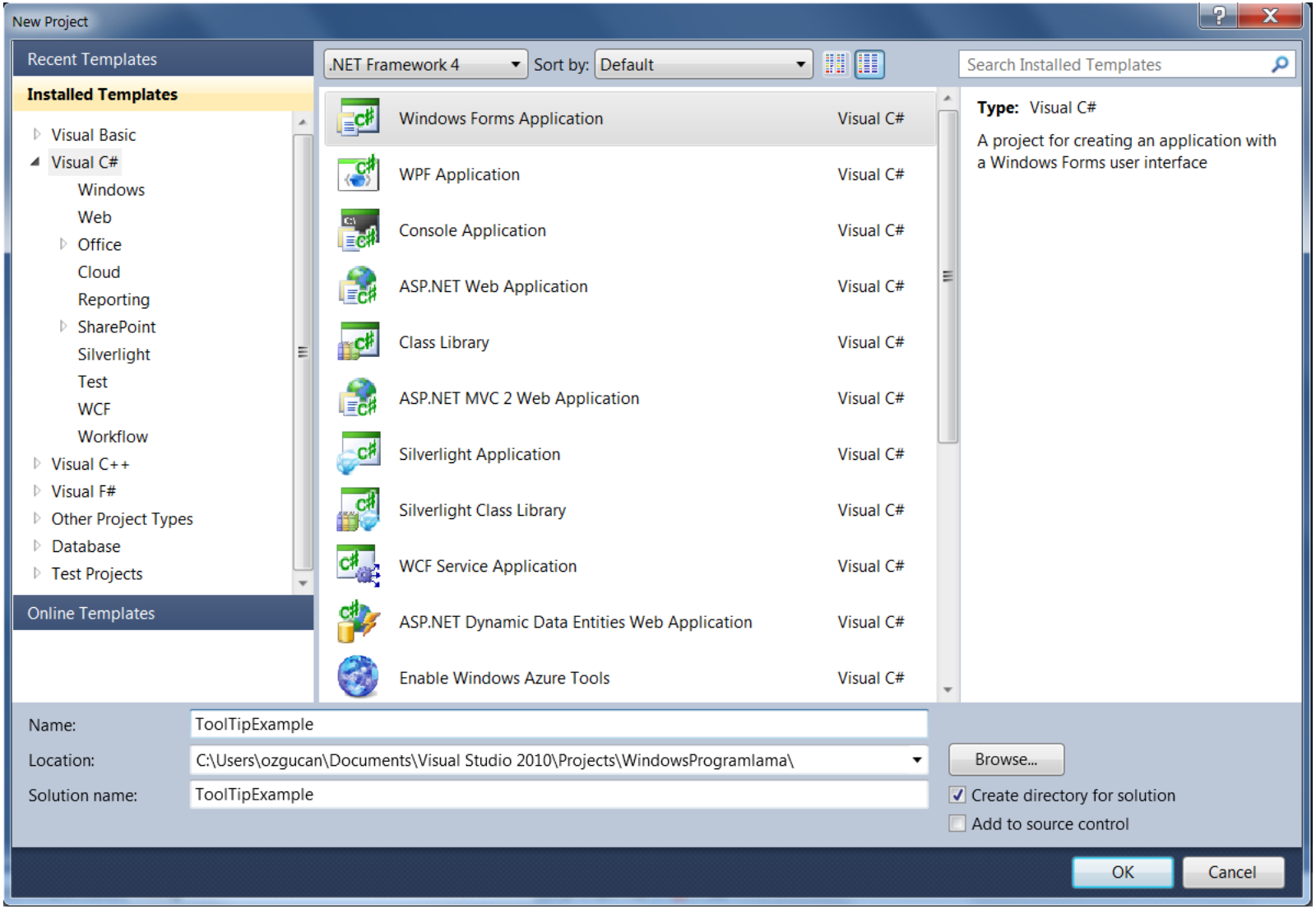
Yrd. Doç. Dr. Özgü Can

ToolTip

- Mouse'un GUI elemanı üzerinde gezinirken gösterdiği yardımcı metin

ToolTip properties and an event	Description
Common Properties	
AutoPopDelay	The amount of time (in milliseconds) that the tool tip appears while the mouse is over a control.
InitialDelay	The amount of time (in milliseconds) that a mouse must hover over a control before a tool tip appears.
ReshowDelay	The amount of time (in milliseconds) between which two different tool tips appear (when the mouse is moved from one control to another).
Common Event	
Draw	Raised when the tool tip is displayed. This event allows programmers to modify the appearance of the tool tip.

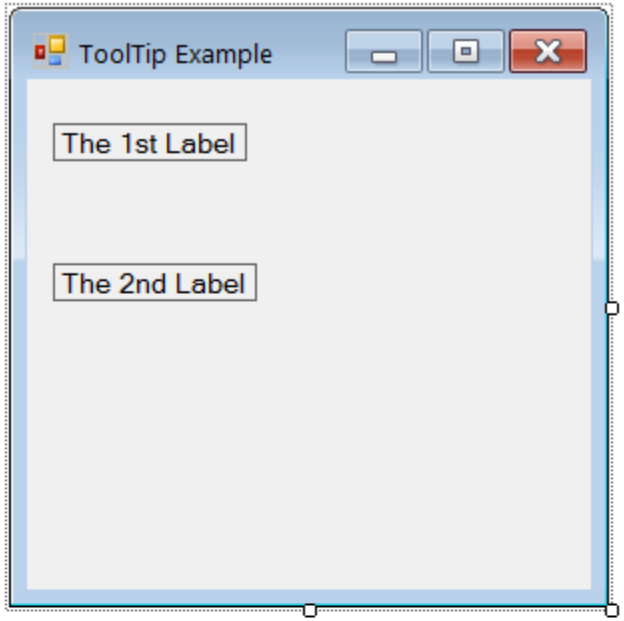
Örnek Uygulama (ToolTip)

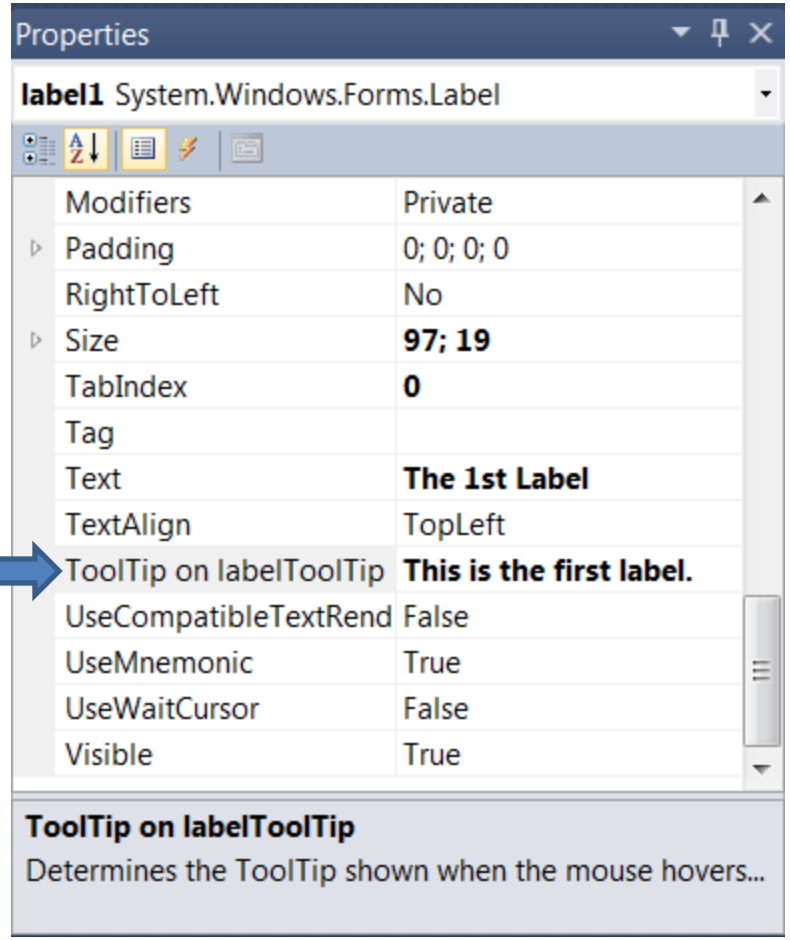


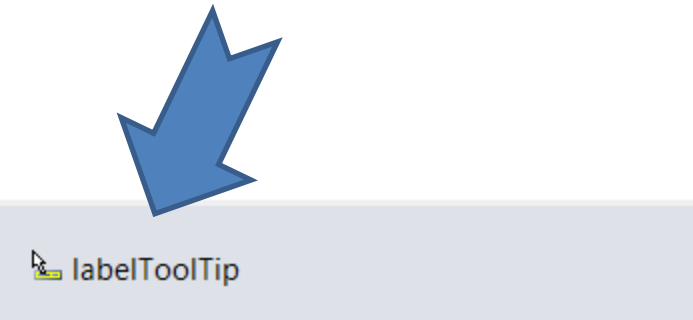
Örnek Uygulama (ToolTip)

- Form
  - Text = **ToolTip Example**
- 2 Label
  - Text = **1st Label**
  - Text = **2nd Label**
  - BorderStyle = **FixedSingle**
- ToolTip
  - Name = **labelToolTip**

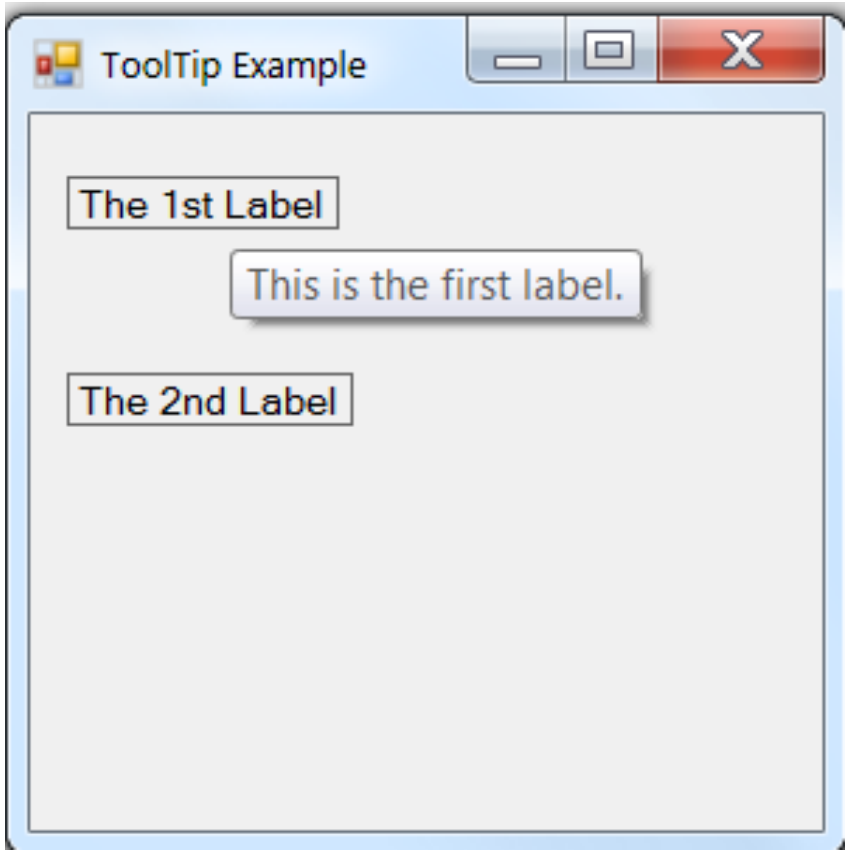
Form1.cs | Form1.cs [Design]

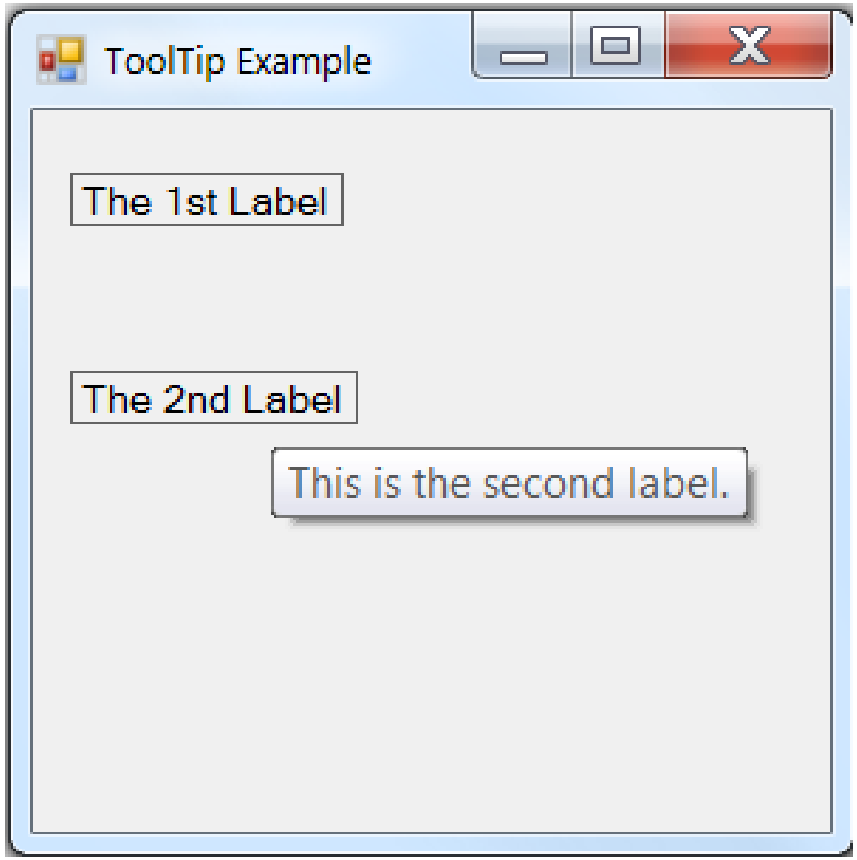






Örnek Uygulama (ToolTip)





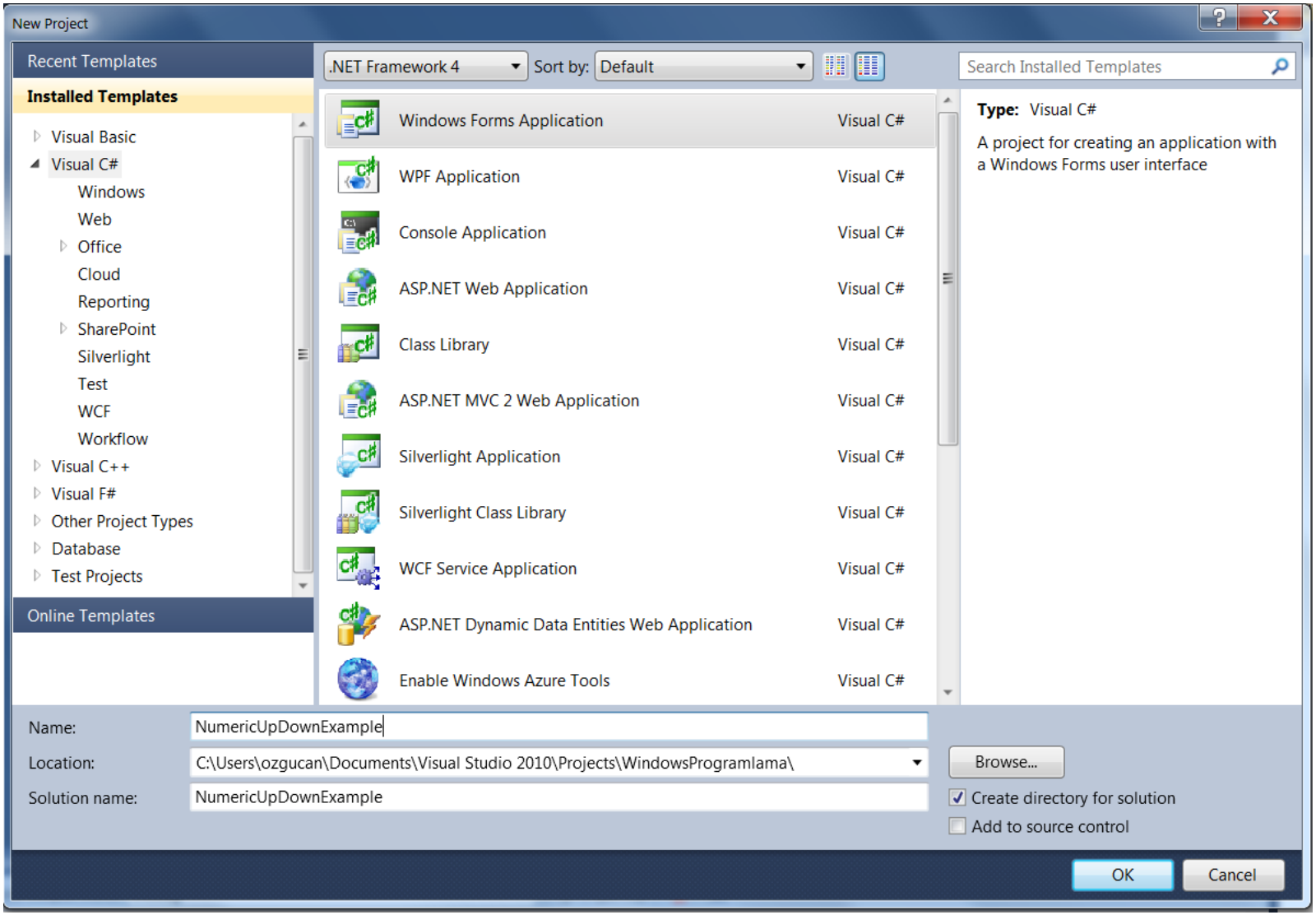
NumericUpDown

- Kullanıcı girdilerini belirli bir sayı değeri aralığı ile sınırlar.

NumericUpDown

NumericUpDown properties and an event	Description
Common Properties	
`DecimalPlaces`	Specifies how many decimal places to display in the control.
`Increment`	Specifies by how much the current number in the control changes when the user clicks the control's up and down arrows.
`Maximum`	Largest value in the control's range.
`Minimum`	Smallest value in the control's range.
`UpDownAlign`	Modifies the alignment of the up and down `Buttons` on the `NumericUpDown` control. This property can be used to display these `Buttons` either to the left or to the right of the control.
`Value`	The numeric value currently displayed in the control.
Common Event	
`ValueChanged`	This event is raised when the value in the control is changed. This is the default event for the `NumericUpDown` control.

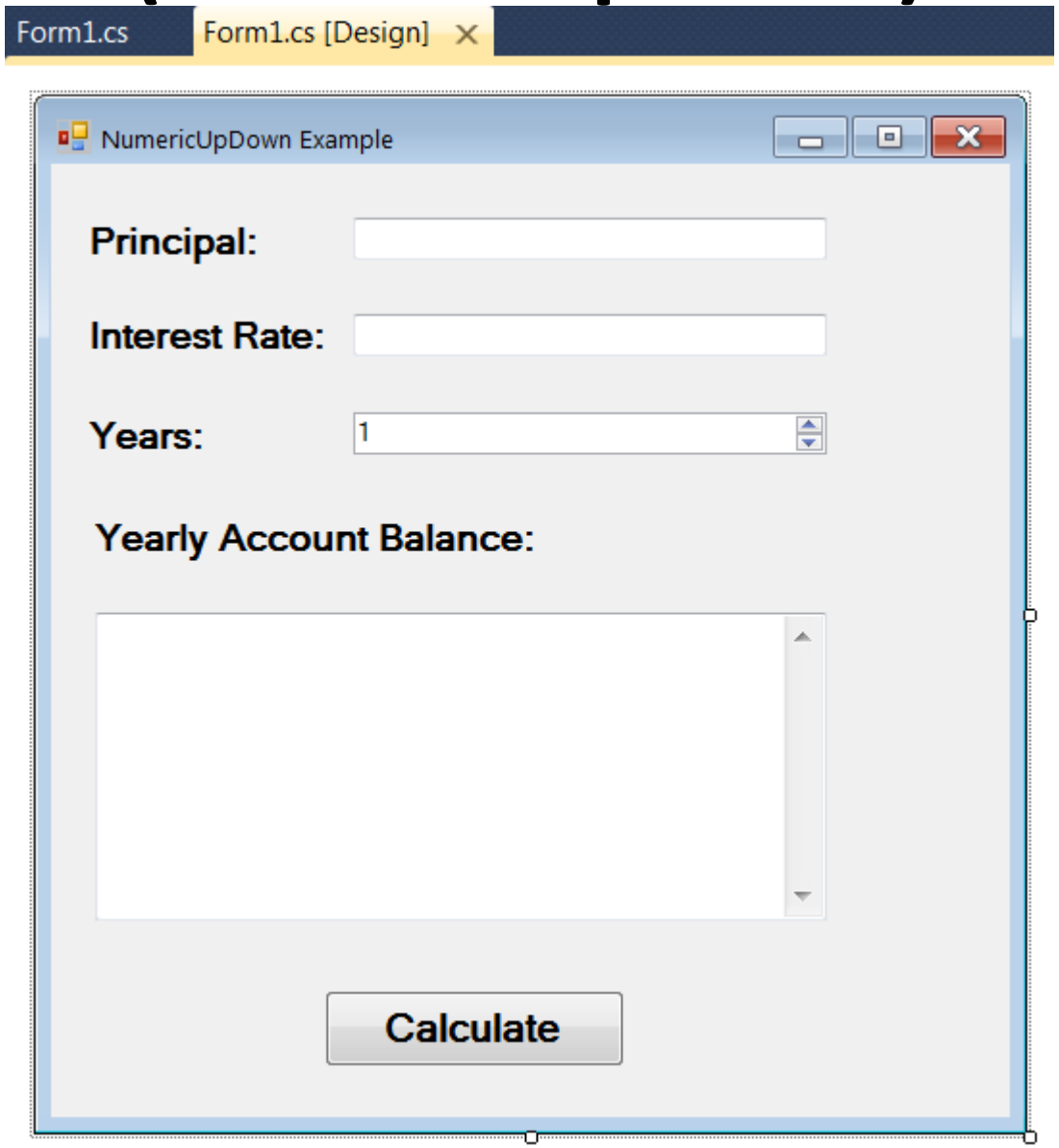
Örnek Uygulama (NumericUpDown)



Örnek Uygulama (NumericUpDown)

- Form
  - Text = **NumericUpDown Example**
- 4 Label
  - Text = **Principal**
  - Text = **Interest Rate**
  - Text = **Years**
  - Text = **Yearly Account Balance**
- NumericUpDown
  - Name = **yearsNumericUpDown**
  - Value = **1**
  - Minimum = **1**
  - Maximum = **10**
- 3 TextBox
  - MultiLine = **True**
  - Scrollbar = **Vertical**
- Button
  - Text = **Calculate**

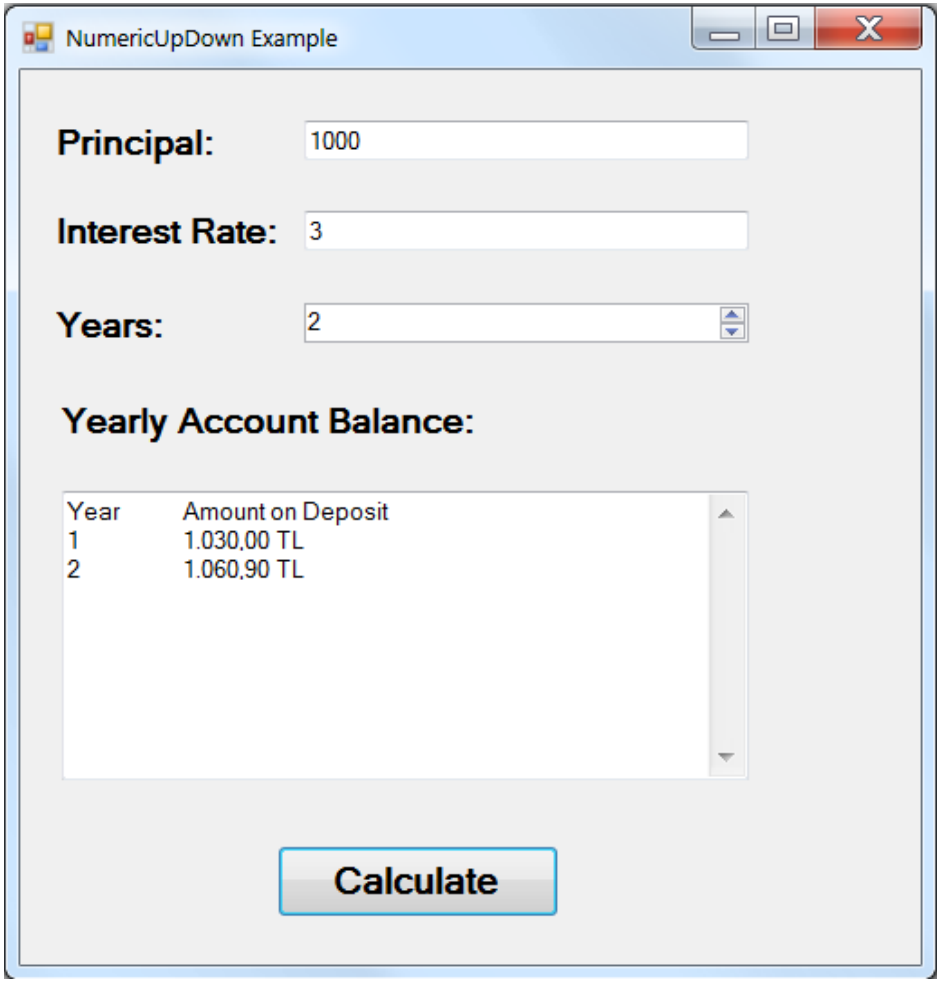
Örnek Uygulama (NumericUpDown)

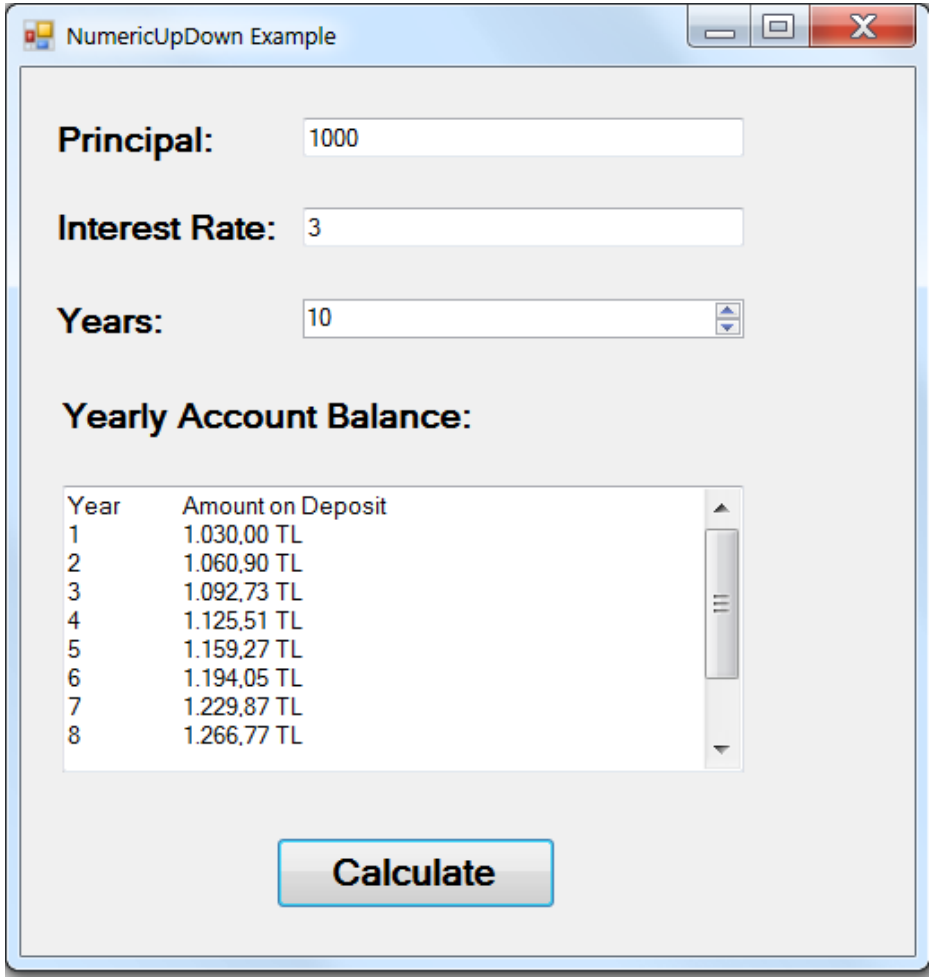


Örnek Uygulama (NumericUpDown)

```
private void calculateButton_Click(object sender, EventArgs e)
{
    decimal principal; // store principal
    double rate; // store interest rate
    int year; // store number of years
    decimal amount; // store amount
    string output; // store output

    // retrieve user input
    principal = Convert.ToDecimal(principalTextBox.Text);
    rate = Convert.ToDouble(interestRateTextBox.Text);
    year = Convert.ToInt32(yearsNumericUpDown.Value);

    // set output header
    output = "Year\tAmount on Deposit\r\n";

    // calculate amount after each year and append to output
    for (int yearCounter = 1; yearCounter <= year; yearCounter++)
    {
        amount = principal *
           ((decimal)Math.Pow((1 + rate / 100), yearCounter));
        output += (yearCounter + "\t" + String.Format("{0:C}", amount) + "\r\n");
    }

    displayTextBox.Text = output;
}
```

Örnek Uygulama (NumericUpDown)





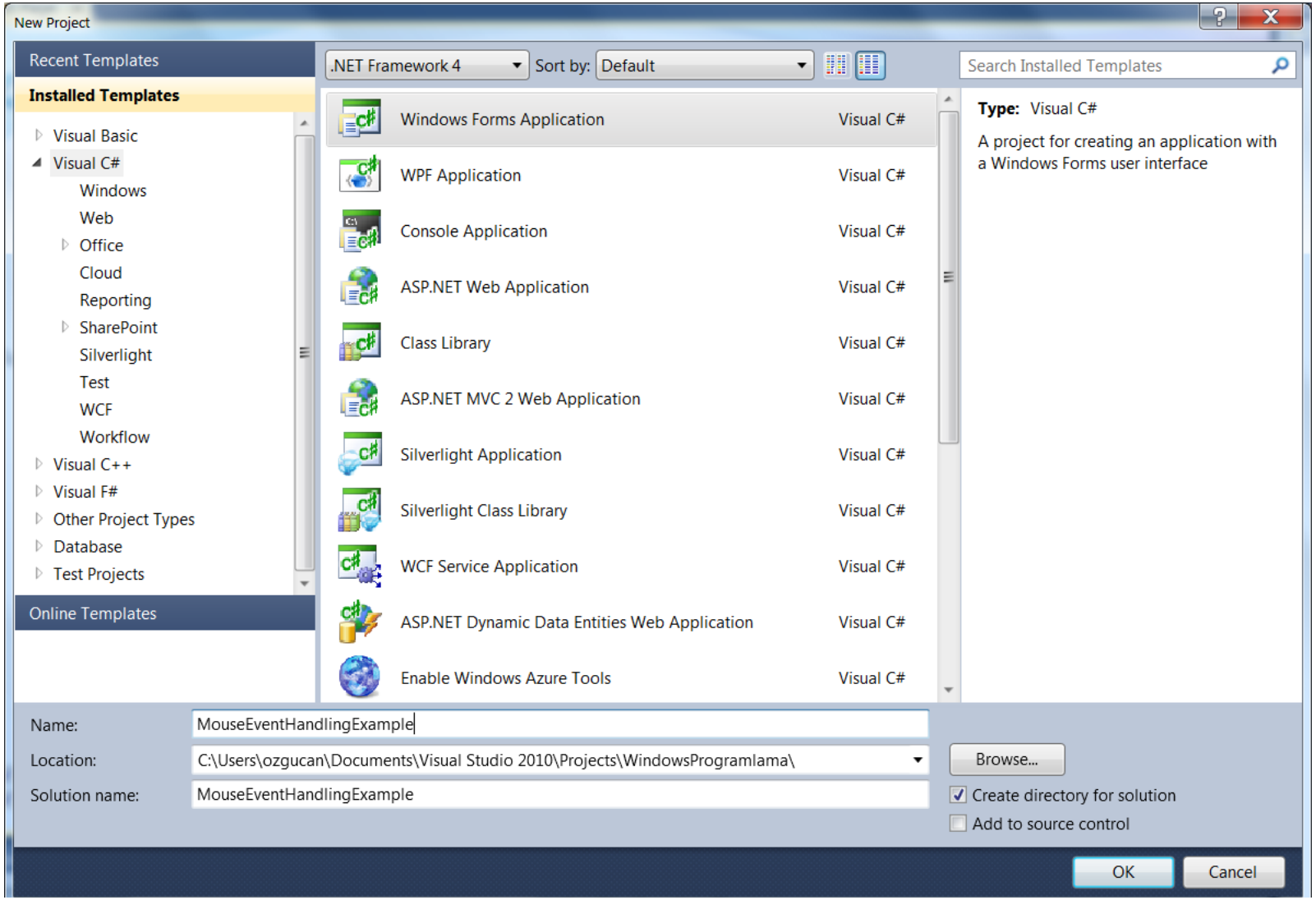
Mouse Olay-İşleme (Mouse Event-Handling)

- Kullanıcı, mouse aracılığı ile kontrol ile etkileşim kurar.
- Olay-işleme metoduna bilgi **`MouseEventArgs`** sınıfının nesnesi ile iletilir.
- **`MouseEventHandler`** → Mouse olay-işlemenin yaratılması

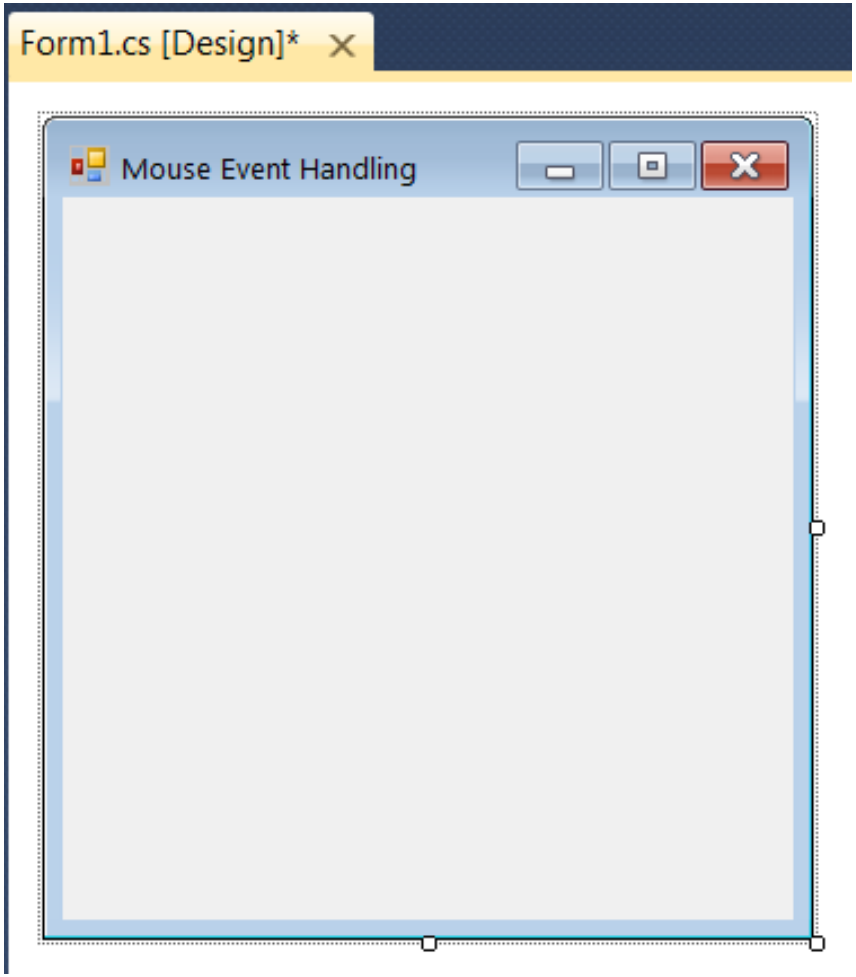
Mouse Olay-İşleme (Mouse Event-Handling)

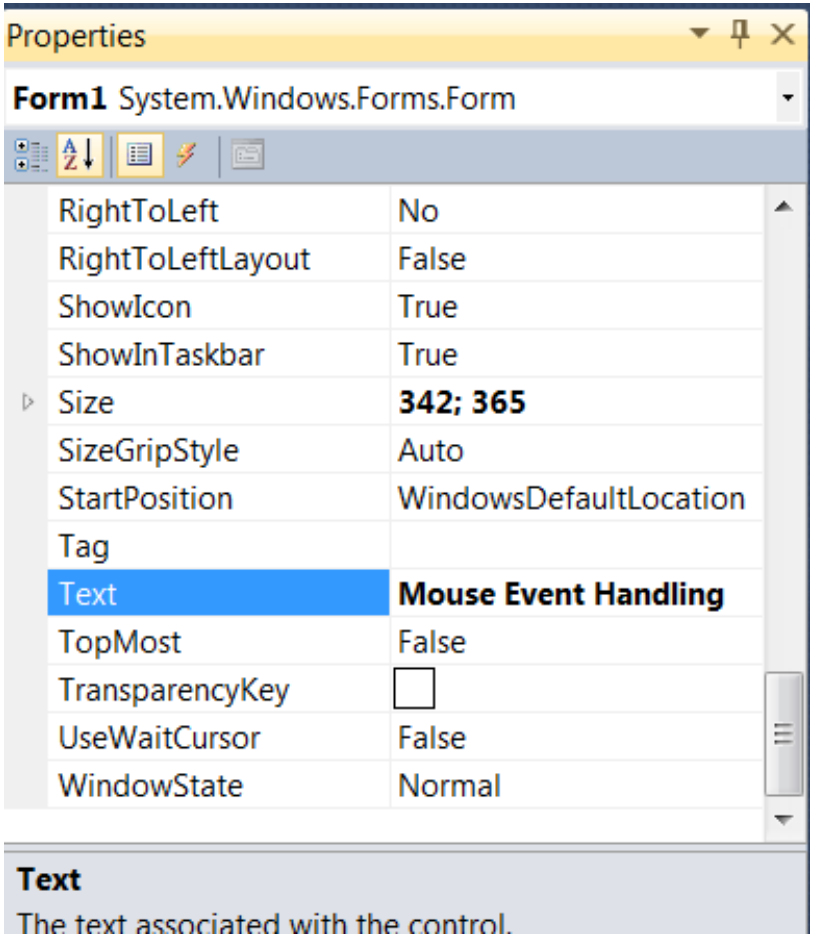
Mouse events and event arguments	
Mouse Events with Event Argument of Type `EventArgs`	
`MouseEnter`	Occurs when the mouse cursor enters the control's boundaries.
`MouseHover`	Occurs when the mouse cursor hovers within the control's boundaries.
`MouseLeave`	Occurs when the mouse cursor leaves the control's 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.
`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 *x*-coordinate within the control where the event occurred.
`Y`	The *y*-coordinate within the control where the event occurred.

Örnek Uygulama (Mouse Olay-İşleme)

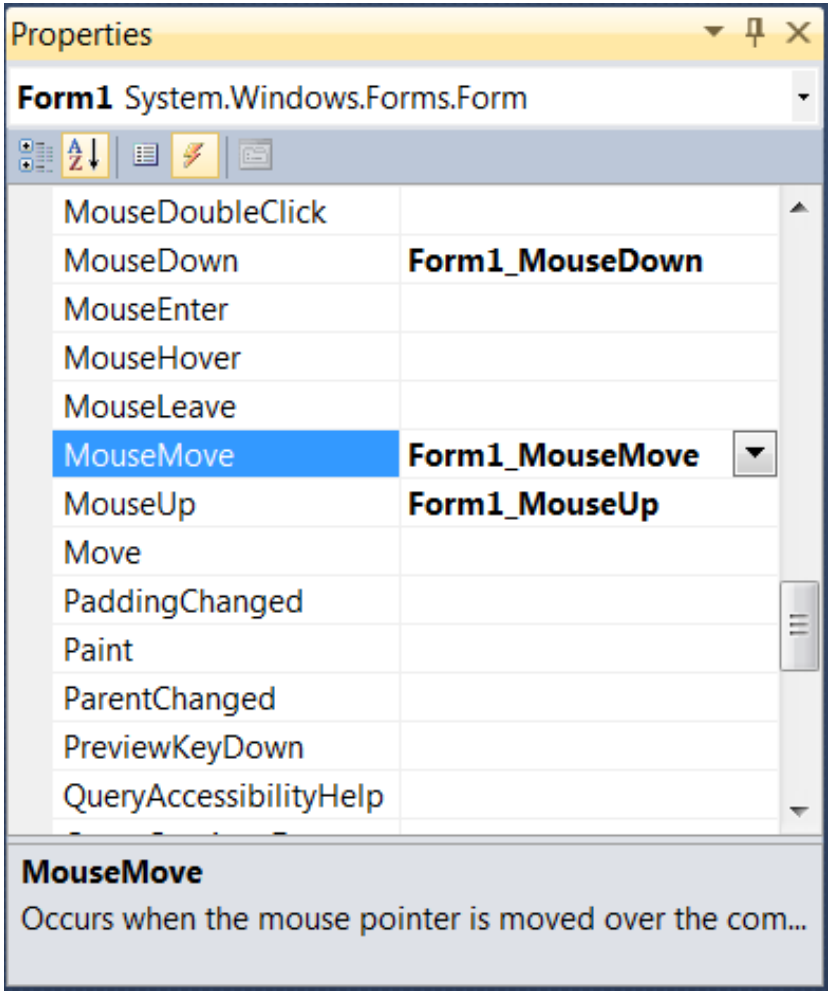


Örnek Uygulama
(Mouse Olay-İşleme)



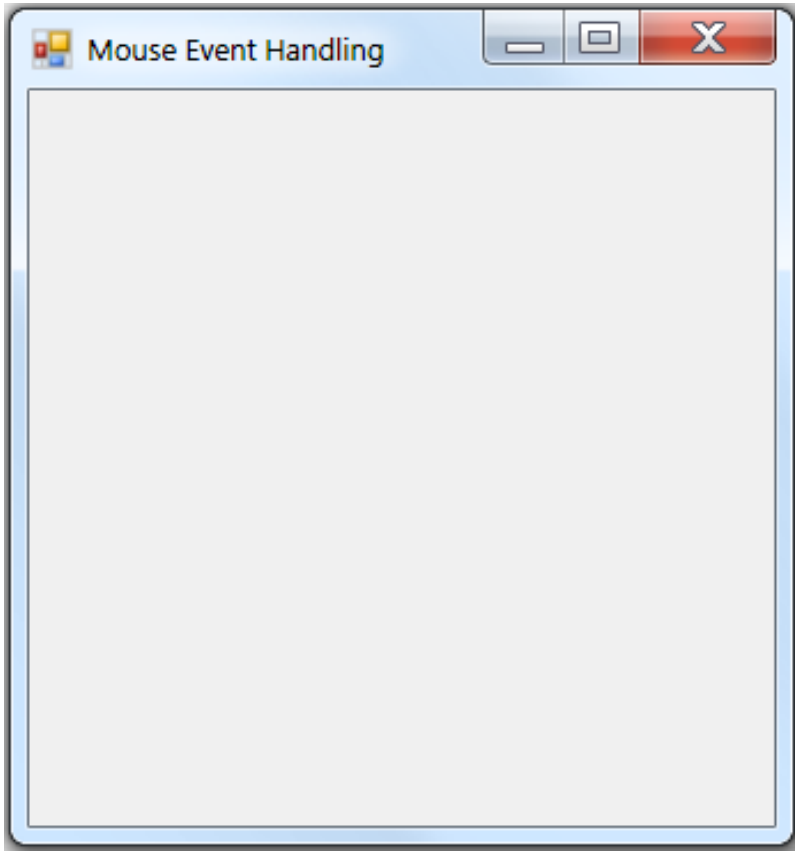


Örnek Uygulama (Mouse Olay-İşleme)



```
namespace MouseEventHandlingExample
{
    public partial class Form1 : Form
    {
        bool shouldPaint = false; // determines whether to paint

        public Form1()
        {
            InitializeComponent();
        }

        // paint when mouse button is pressed down
        private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            // indicate that user is dragging the mouse
            shouldPaint = true;
        }

        // stop painting when mouse button is released
        private void Form1_MouseUp(object sender, MouseEventArgs e)
        {
            // indicate that user released the mouse button
            shouldPaint = false;
        }
       // draw circle whenever mouse moves with its button held down
       private void Form1_MouseMove(object sender, MouseEventArgs e)
       {
           if (shouldPaint) // check if mouse button is being pressed
           {
               // draw a circle where the mouse pointer is present
               using (Graphics graphics = CreateGraphics())
               {
                   graphics.FillEllipse(new SolidBrush(Color.BlueViolet), e.X, e.Y, 4, 4);
               } // end using; calls graphics.Dispose()
           }
       }

       private void Form1_Load(object sender, EventArgs e)
       {

       }
    }
```

Örnek Uygulama
(Mouse Olay-İşleme)



Klavye Olay-İşleme (Keyboard Event-Handling)

- Klavye tuşlarına basıldığında ortaya çıkar.
- 3 klavye olayı:
  - `KeyPress`
  - `KeyUp`
  - `KeyDown`

Klavye Olay-İşleme (Keyboard Event-Handling)

- **`KeyPress`**
  - Kullanıcı ASCII karakterlerini temsil eden bir tuşa bastığında ortaya çıkmaktadır.
  - `TextBox`'da `Enter` tuşuna basıldığında diğer `TextBox`'a geçme.
  - Basılan tuş, olay işleyicinin **`KeyPressEventArgs`** argumanının **`KeyChar`** özelliği ile belirtilir.
  - Modifier tuşlar (*Shift*, *Alt*, *Ctrl* vs.) ile çalışmaz.

Klavye Olay-İşleme (Keyboard Event-Handling)

- Modifier tuşlar ile çalışılacaksa;
  - **`KeyUp`**
  - **`KeyDown`**

`KeyEventArgs` argumanı modifier tuşlar ile ilgili bilgiyi içerir.

Klavye Olay-İşleme (Keyboard Event-Handling)

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. Raised after `KeyDown` and before `KeyUp`.
Class `KeyPressEventArgs` *Properties*	
`KeyChar`	Returns the ASCII character for the key pressed.
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.
`KeyCode`	Returns the key code for the key as a value from the `Keys` enumeration. This does not include modifier-key information. It's 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 (*Alt*, *Ctrl* and *Shift*). This property is used to determine modifier-key information only.

Klavye Olay-İşleme (Keyboard Event-Handling)

- Geçerli (default) olan → Klavye olayları mevcut kontrol tarafından işlenir.
- Bunu değiştirmek için;
  - `Form`'un **`KeyPreview`** özelliği → **`true`**
- Böylece, Form klavye olaylarını kontrolden önce işleyecektir.
- `Enter`'a basıldığında `Button`'a basılmış olması için;
  - `Form`'un **`AcceptButton`** özelliği kullanılır.

Örnek Uygulama-1
(Klavye Olay-İşleme)

New Project

Recent Templates

Installed Templates

- Visual Basic
- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - Reporting
  - SharePoint
  - Silverlight
  - Test
  - WCF
  - Workflow
- Visual C++
- Visual F#
- Other Project Types
- Database
- Test Projects

Online Templates

.NET Framework 4 | Sort by: Default | Search Installed Templates

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
ASP.NET Web Application	Visual C#
Class Library	Visual C#
ASP.NET MVC 2 Web Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
WCF Service Application	Visual C#
ASP.NET Dynamic Data Entities Web Application	Visual C#
Enable Windows Azure Tools	Visual C#

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: EnterButtonExample

Location: C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\

Solution name: EnterButtonExample

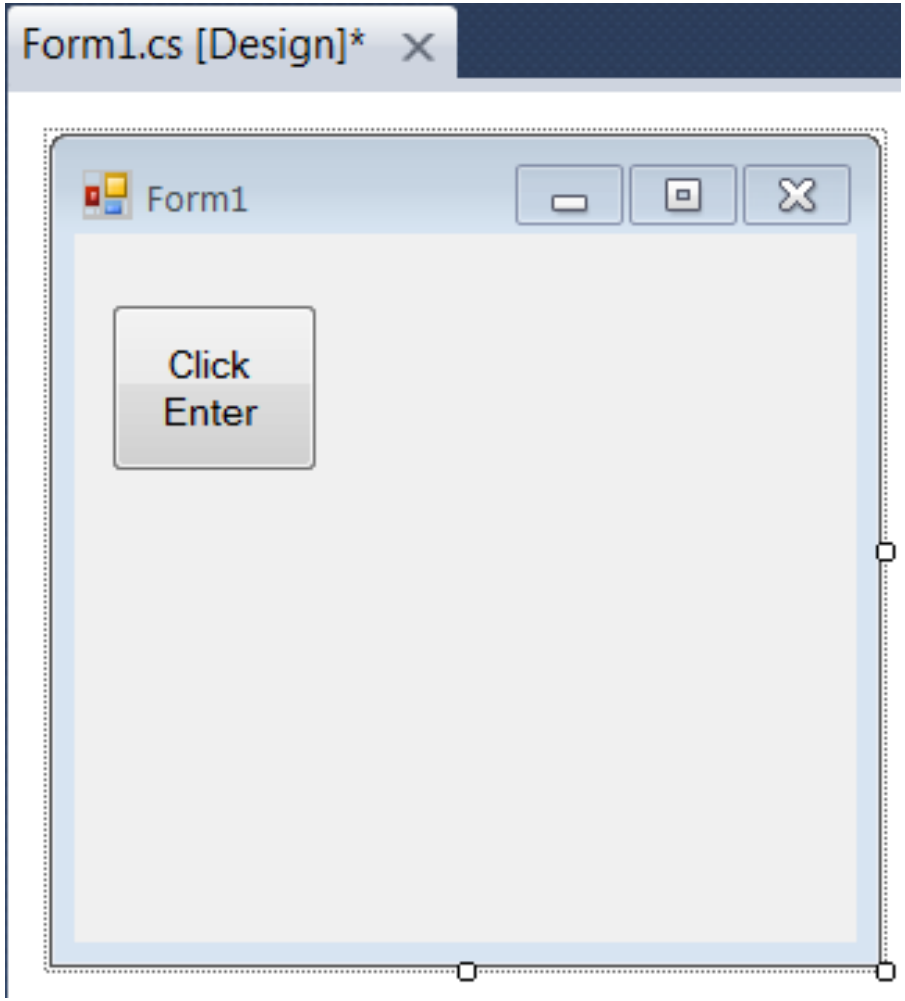
Browse...

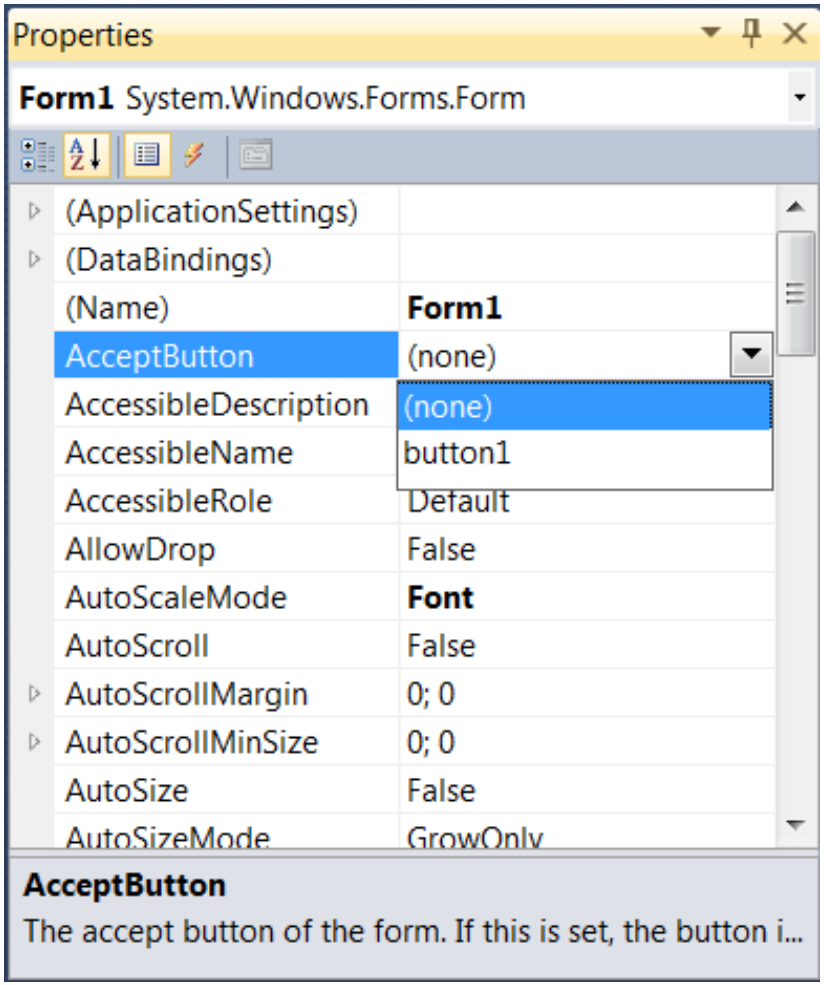
☑ Create directory for solution

☐ Add to source control

OK | Cancel

Örnek Uygulama-1
(Klavye Olay-İşleme)





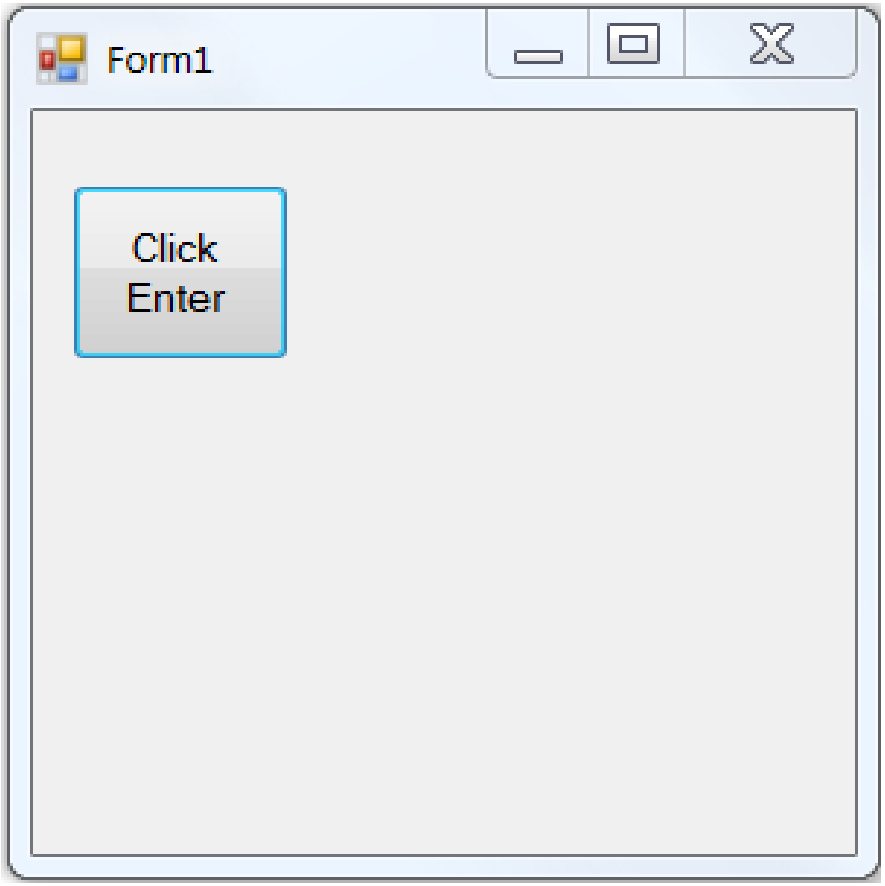
Örnek Uygulama-1
(Klavye Olay-İşleme)

Form1.cs | Form1.cs [Design]

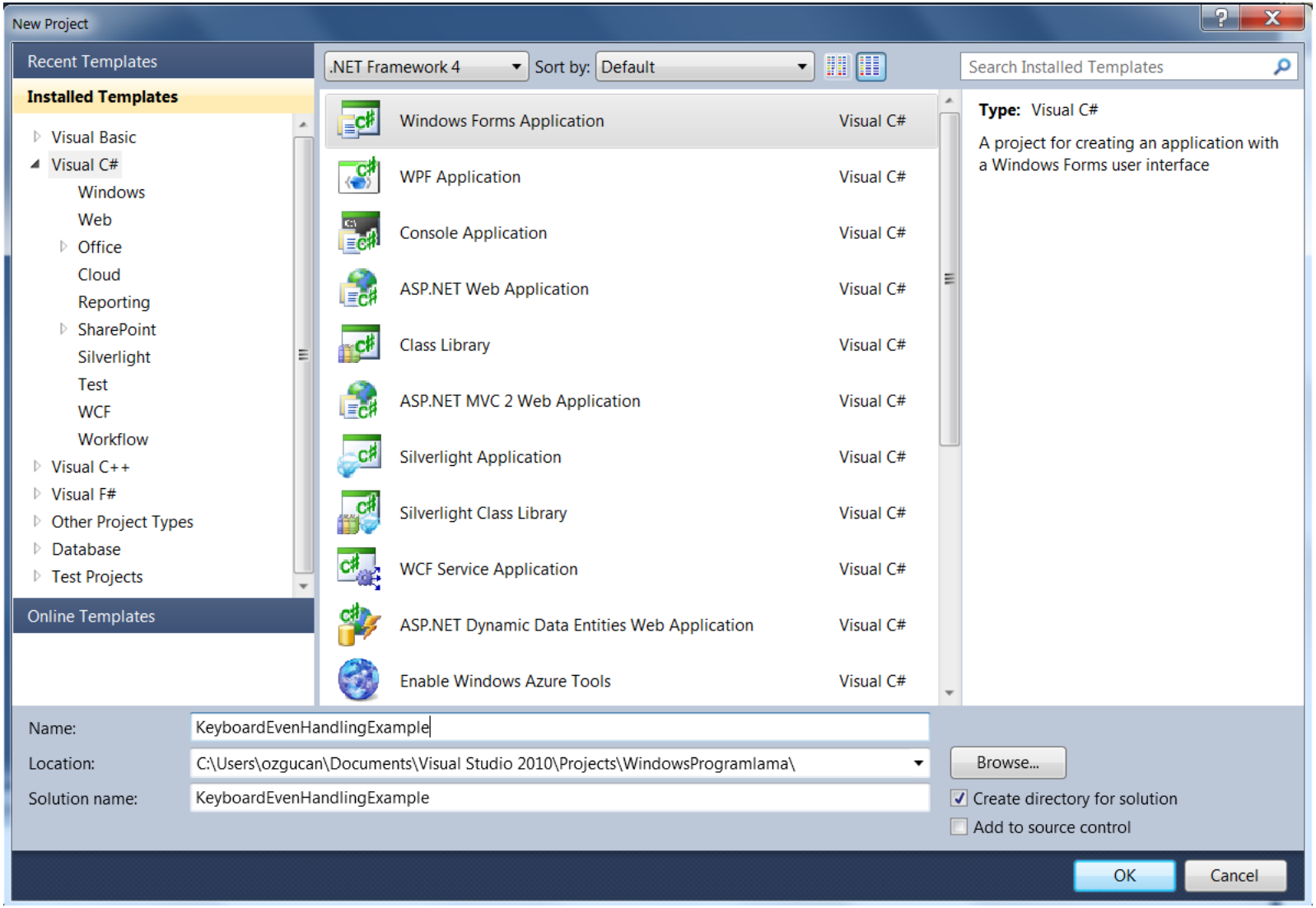
EnterButtonExample.Form1 | button1_Cli

```
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9
10  namespace EnterButtonExample
11  {
12      public partial class Form1 : Form
13      {
14          public Form1()
15          {
16              InitializeComponent();
17          }
18
19          private void button1_Click(object sender, EventArgs e)
20          {
21              MessageBox.Show("You Clicked Enter!");
22          }
23      }
24  }
25
```

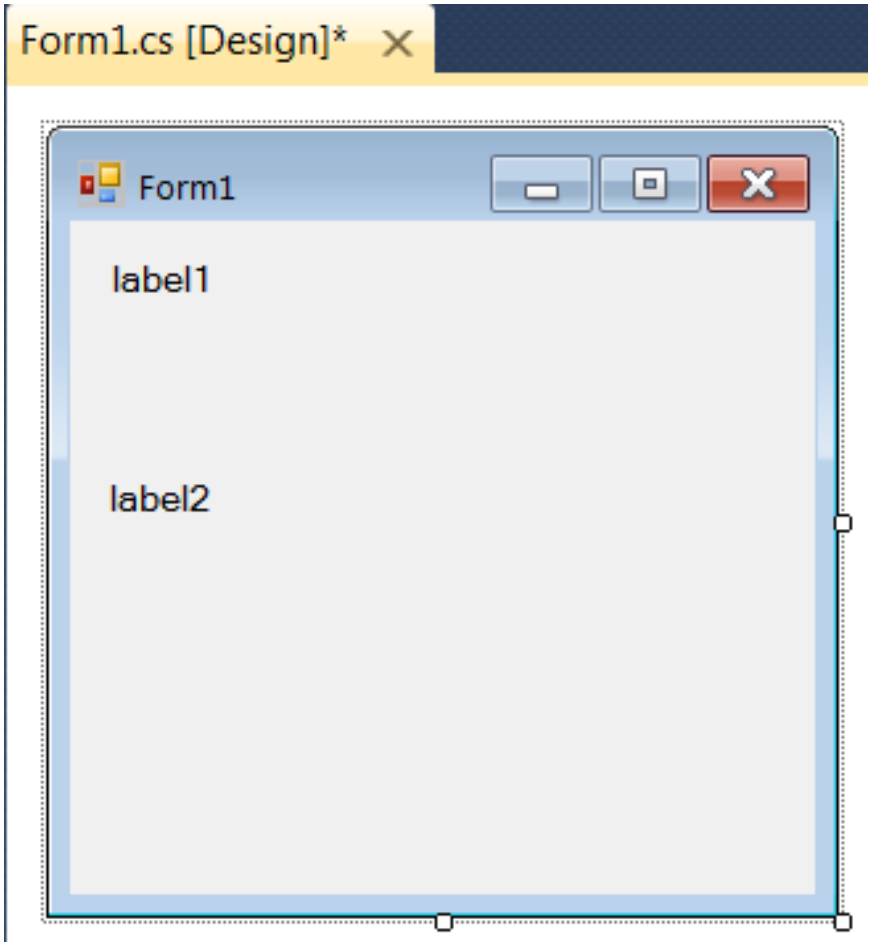
Örnek Uygulama-1
(Klavye Olay-İşleme)



Örnek Uygulama-2
(Klavye Olay-İşleme)

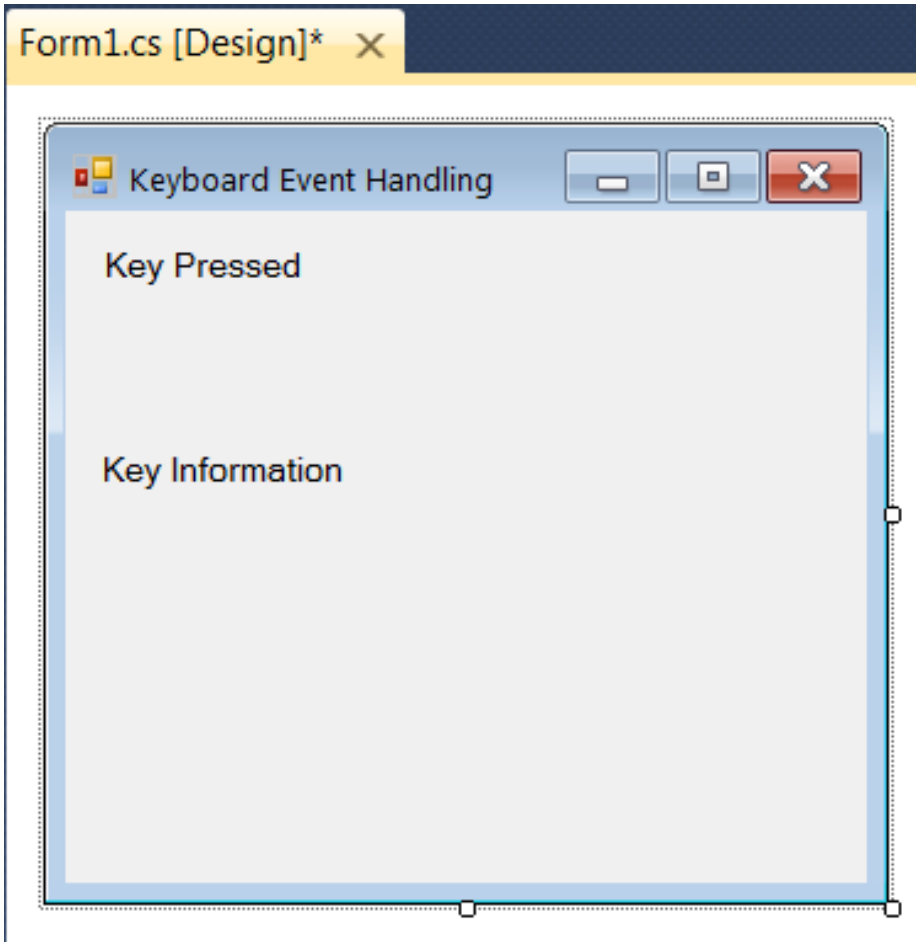


Örnek Uygulama-2
(Klavye Olay-İşleme)

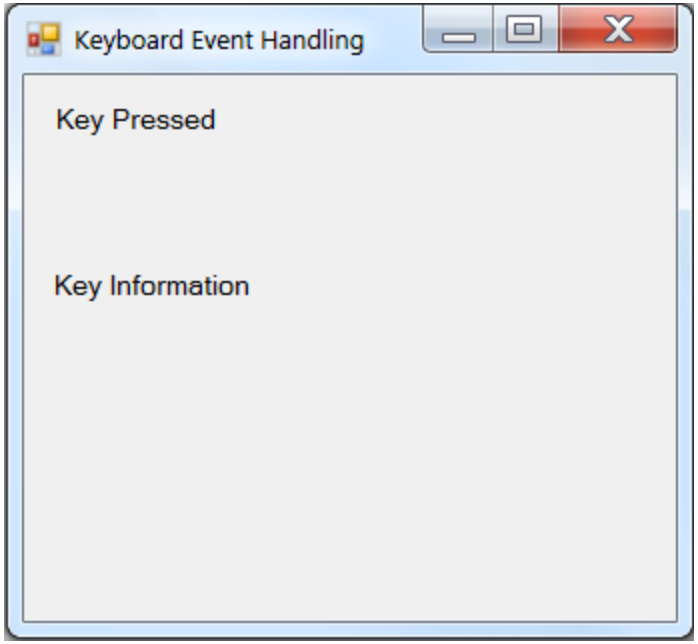


- Label1
  - **Text = Key Pressed**
  - **Name = charLabel**
- Label2
  - **Text = Key Information**
  - **Name = keyInfoLabel**
- Form1
  - **Text = Keyboard Event Handling**

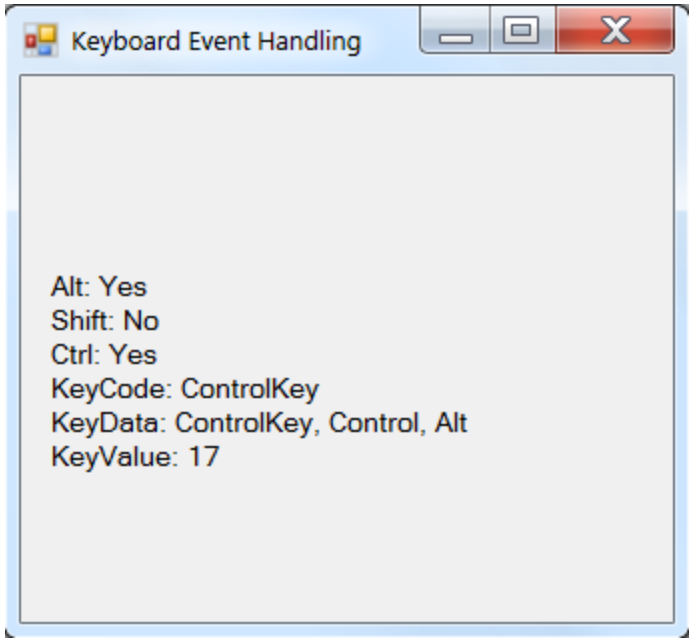
Örnek Uygulama-2
(Klavye Olay-İşleme)

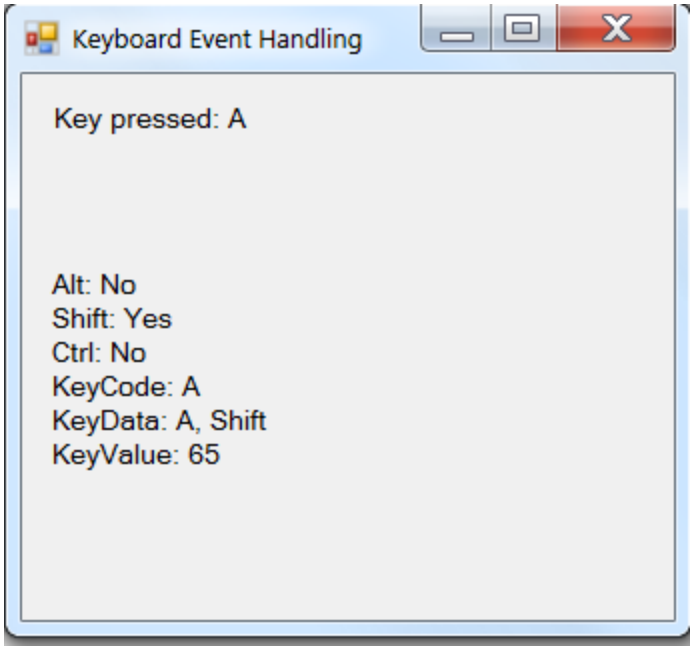


```
namespace KeyboardEvenHandlingExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // display the character pressed using KeyChar
        private void Form1_KeyPress(object sender, KeyPressEventArgs e)
        {
            charLabel.Text = "Key pressed: " + e.KeyChar;
        }

        // display modifier keys, key code, key data and key value
        private void Form1_KeyDown(object sender, KeyEventArgs e)
        {
            keyInfoLabel.Text =
            "Alt: " + (e.Alt ? "Yes" : "No") + '\n' +
            "Shift: " + (e.Shift ? "Yes" : "No") + '\n' +
            "Ctrl: " + (e.Control ? "Yes" : "No") + '\n' +
            "KeyCode: " + e.KeyCode + '\n' +
            "KeyData: " + e.KeyData + '\n' +
            "KeyValue: " + e.KeyValue;
        }

       // clear Labels when key released
       private void Form1_KeyUp(object sender, KeyEventArgs e)
       {
           charLabel.Text = "";
           keyInfoLabel.Text = "";
       }


    }
```


Keyboard Event Handling
Key Pressed
Key Information


Keyboard Event Handling
Key pressed: c
Alt: No
Shift: No
Ctrl: No
KeyCode: C
KeyData: C
KeyValue: 67


Keyboard Event Handling
Alt: Yes
Shift: No
Ctrl: Yes
KeyCode: ControlKey
KeyData: ControlKey, Control, Alt
KeyValue: 17


Keyboard Event Handling
Key pressed: A
Alt: No
Shift: Yes
Ctrl: No
KeyCode: A
KeyData: A, Shift
KeyValue: 65