MCTE 4327

Software Engineering

Week 07 GUI Engineering

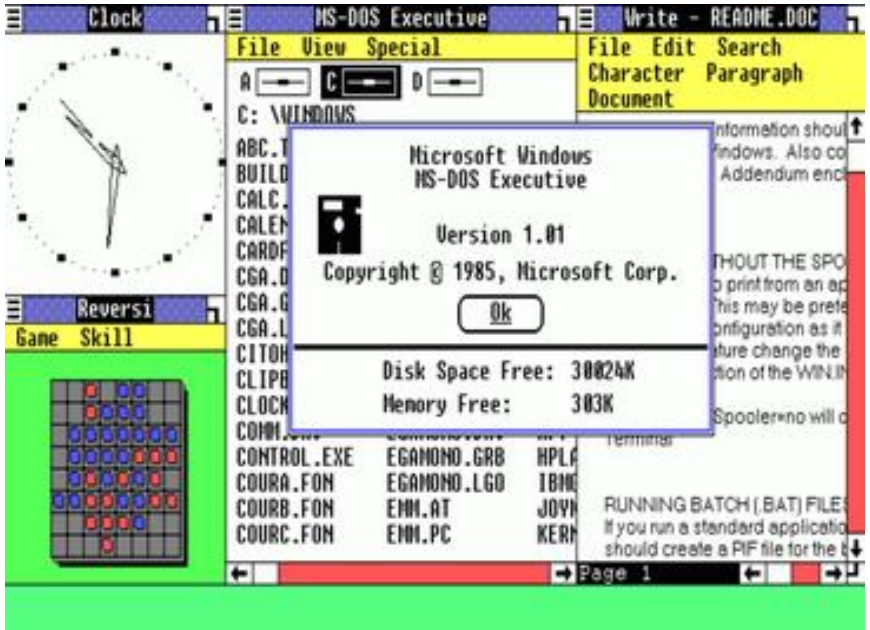
Outline

- GUI frameworks
- Common GUI elements
- Basic image processing

GUI

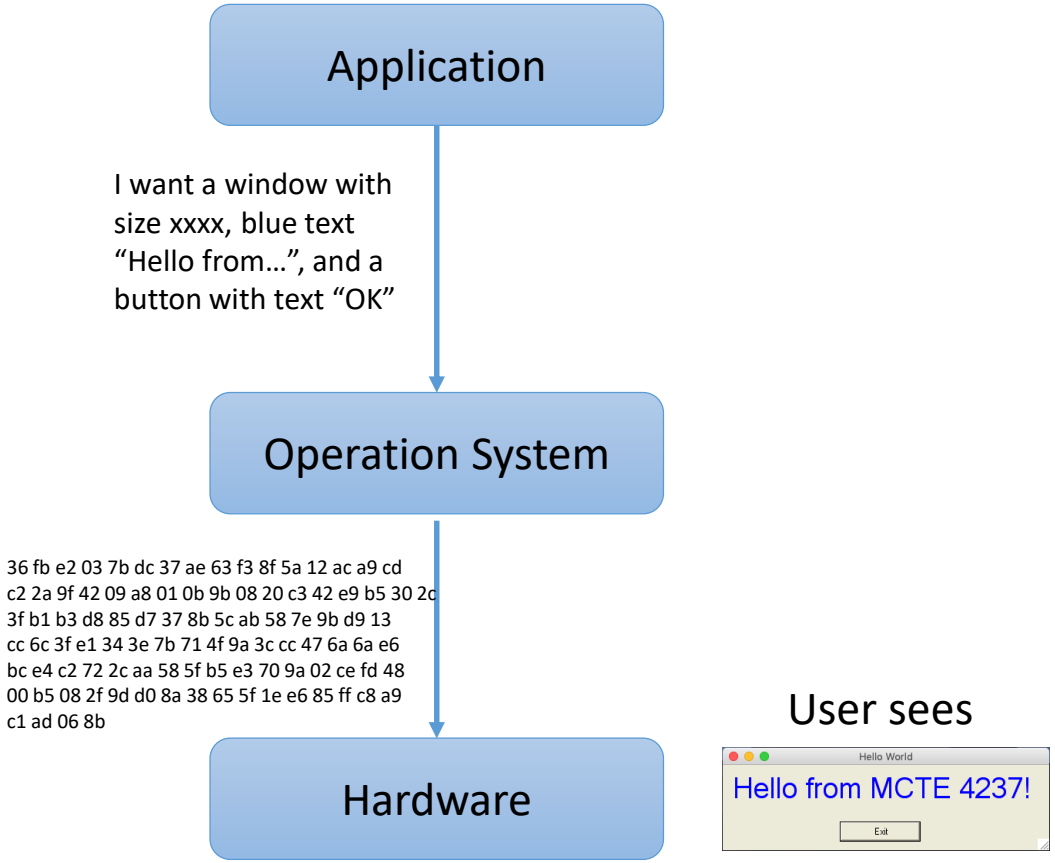
- The graphical user interface (GUI: pronounced as gu ee) is a form of user interface that allows users to interact through graphical icons and visual indicators instead of text-based user interfaces such as command prompts and terminals.

GUI



GUI

- GUIs are handled by the OS. An app that wants GUI just need to inform the OS. The OS handles all the low-level details.
- But different OS'es require different manners of informing.



Cross-platform GUI frameworks

- GTK for desktop platforms (https://www.gtk.org/)
- QT for desktop platforms (https://www.qt.io/)

- Xamarin for mobile platforms (https://visualstudio.microsoft.com/xamarin/)

WinForms

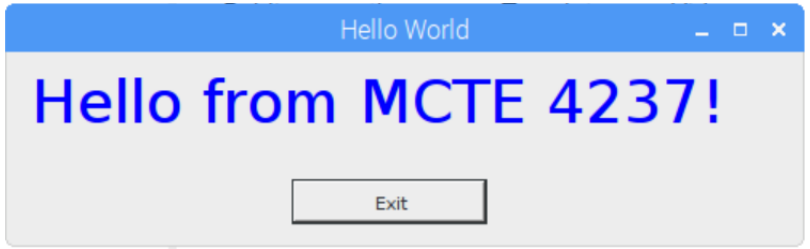
- Part of .NET framework under *System.Windows.Forms* namespace.
- It is not 100% cross-platform because some features rare wrapped around the Windows32 API.

Example of a program that uses WinForms and runs across multiple platforms.

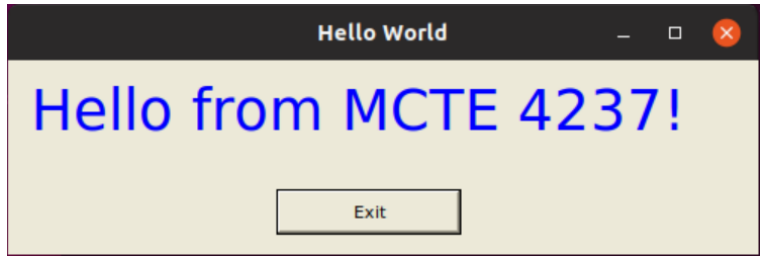
Windows 10

Mac OS 10.12 Sierra

Raspbian Jessie

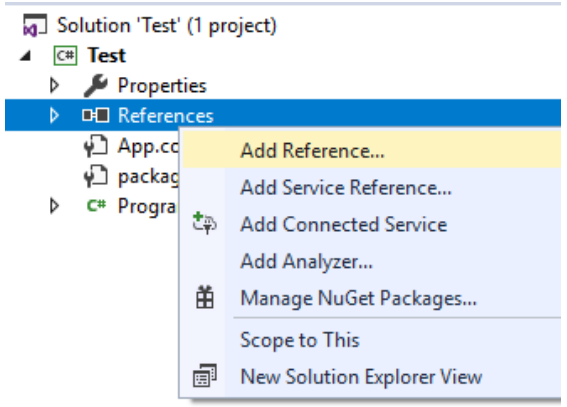


Ubuntu 18



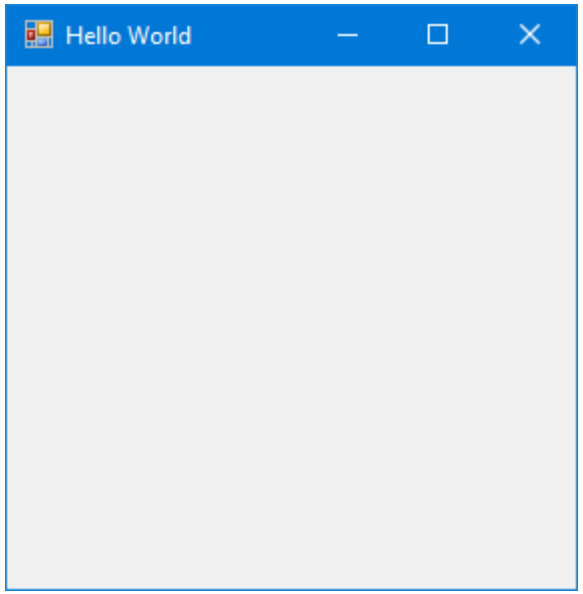
Simple Windows

```
using System;
using System.Windows.Forms; //Need to add System.Windows.Forms as a reference as well

namespace Test
{
    class Program
    {

        static void Main()
        {
            Form form1 = new Form(); //Windows are called "forms"
            form1.Text = "Hello World";  //Set the title of the window
            form1.Show(); //Show the Form
        }

    }

}
```

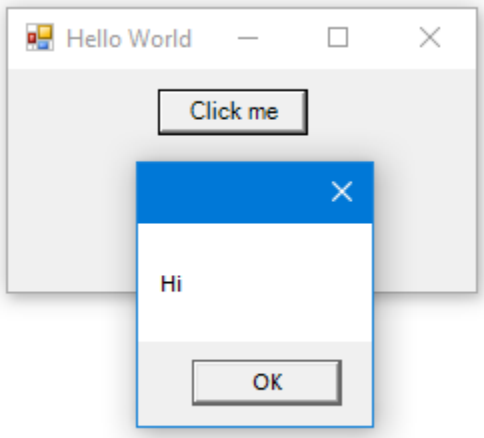


The code produces no visible output because as soon as the form has been shown, the program ends.

```
using System;
using System.Windows.Forms;

namespace Test
{
    class Program
    {
        static void Main()
        {
            Form form1 = new Form();
            form1.Text = "Hello World";

            Application.Run(form1); //Show form1 and pause execution until form1 gets closed
        }

    }

}
```



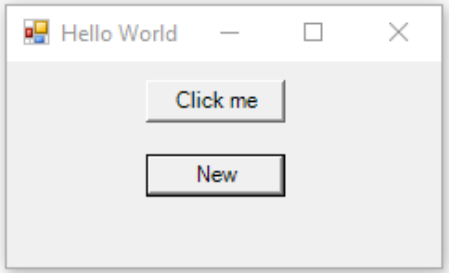
```
using System.Windows.Forms;

namespace Test
{
    class Program
    {
        static void Main()
        {
            Form form1 = new Form();
            form1.Text = "Hello World";
            form1.Width = 250; //The form is 250 units wide
            form1.Height = 150; //The form is 150 units long

            Button button1 = new Button(); //New button
            button1.Text = "Click me"; //Set the button's text
            button1.Top = 10;  //The button's position is 10 units from the top of the window
            button1.Left = 75; //The button's position is 75 units from the left of the window

            button1.Click += (a, e) => MessageBox.Show("Hi"); //Handle click event

            form1.Controls.Add(button1); //Add button1 to form1

            Application.Run(form1);
        }

    }

}
```



```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace Test
{
    class Program
    {
        static void Main()
        {
            Form form1 = new Form();
            form1.Text = "Hello World";
            form1.Width = 250;
            form1.Height = 150;

            Button button1 = new Button();
            button1.Text = "Click me";
            button1.Top = 10;
            button1.Left = 75;
            button1.Click += (a, e) => MessageBox.Show("Hi");

            Button button2 = new Button(); //Another button
            button2.Text = "New"; //Set the button's text
            button2.Top = 50;  //The button's position is 50 units from the top of the window
            button2.Left = 75; //The button's position is 75 units from the left of the window
            button2.Click += (a, e) => //When the user clicks the button
            {
                    Form form2 = new Form();  //Another form
                    form2.BackColor = Color.Aqua;  //Sets background color
                    form2.Text = "New Window opened"; //Set title text
                    form2.Width = 450;  //Set the form's width
                    form2.Height = 200; //Set the form's height
                    Button button3 = new Button(); //New button
                    button3.Text = "Close";  //Set the button's text
                    button3.Click += (x, y) => form2.Close(); //Handle the click event
                    form2.Controls.Add(button3); //Add the button to form2
                    form2.Show(); //Display form2
            };

            form1.Controls.Add(button1);
            form1.Controls.Add(button2); //Add button2 to form1
            Application.Run(form1);
        }
    }
}
```

- Now we have 2 windows.
- The first window has 2 buttons and the second window has 1 button.
- The code becomes unmanageable if we keep adding more UI elements.

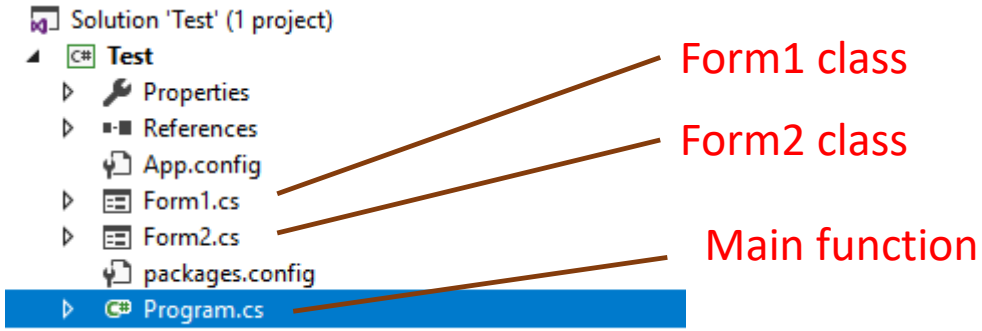
Solution: create a separate class for each window. Separate presentation from logic.



Separating Forms as different files

Program.cs

```
using System.Windows.Forms;

namespace Test
{
    class Program
    {
        static void Main()
        {
            Application.Run(new Form1());
        }

    }

}
```



Form1.cs

```
using System.Windows.Forms;

namespace Test
{
    class Form1 : Form
    {
        public Form1() //Constructor
        {
            InitializeComponents();
        }

        private void InitializeComponents()
        {
            Text = "Hello World";
            Width = 250;
            Height = 150;

            Button button1 = new Button();
            button1.Text = "Click me";
            button1.Top = 10;
            button1.Left = 75;

            button1.Click += (a, e) => MessageBox.Show("Hi");

            Button button2 = new Button();
            button2.Text = "New";
            button2.Top = 50;
            button2.Left = 75;
            button2.Click += (a, e) => new Form2().Show();

            Controls.Add(button1);
            Controls.Add(button2);
        }
    }
}
```

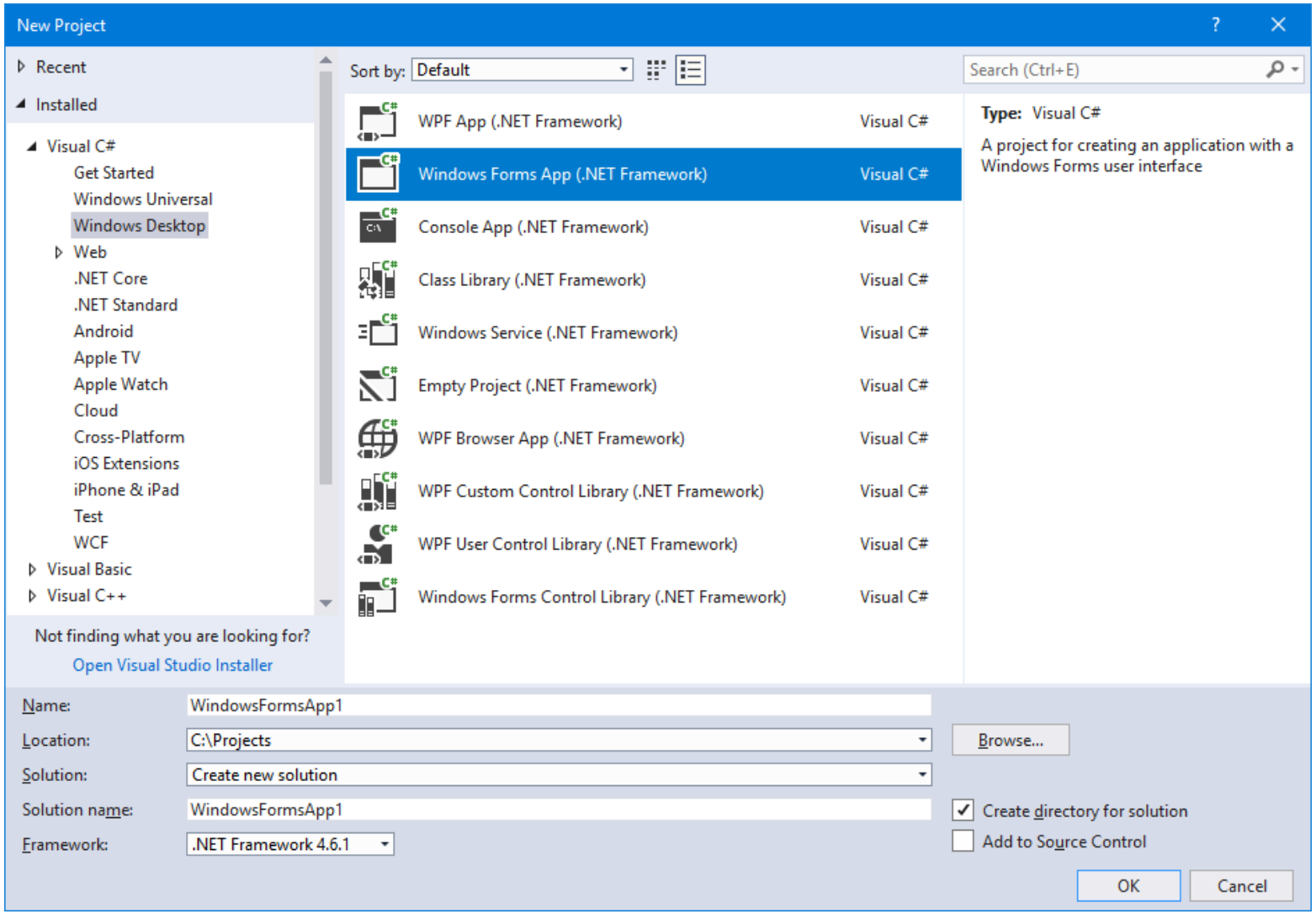
Form2.cs

```
using System.Windows.Forms;
using System.Drawing;

namespace Test
{
    class Form2 : Form
    {
        public Form2() //Constructor
        {
            InitializeComponents();
        }

        private void InitializeComponents()
        {
            BackColor = Color.Aqua;
            Text = "New Window opened";
            Width = 450;
            Height = 200;
            Button button3 = new Button();
            button3.Text = "Close";
            button3.Click += (x, y) => Close();
            Controls.Add(button3);
        }
    }
}
```

GUI Project

- Instead of console project, a GUI-based project type called “Windows Form App” can be selected under “New Project”.

- This will automatically create a Form called Form1 and create the main() function that launches Form1.



Program.cs | Form1.cs [Design]

WindowsFormsApp1 | WindowsFormsApp1.Program | Main()

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5   using System.Windows.Forms;
6
7   namespace WindowsFormsApp1
8   {
        0 references
9       static class Program
10      {
11          /// <summary>
12          /// The main entry point for the application.
13          /// </summary>
14          [STAThread]
            0 references
15          static void Main()
16          {
17              Application.EnableVisualStyles();
18              Application.SetCompatibleTextRenderingDefault(false);
19              Application.Run(new Form1());
20          }
21      }
22  }
23
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'WindowsFormsApp1' (1 project)
  - WindowsFormsApp1
    - Properties
    - References
    - App.config
    - Form1.cs
    - Program.cs

Solution Explorer | Team Explorer

Properties

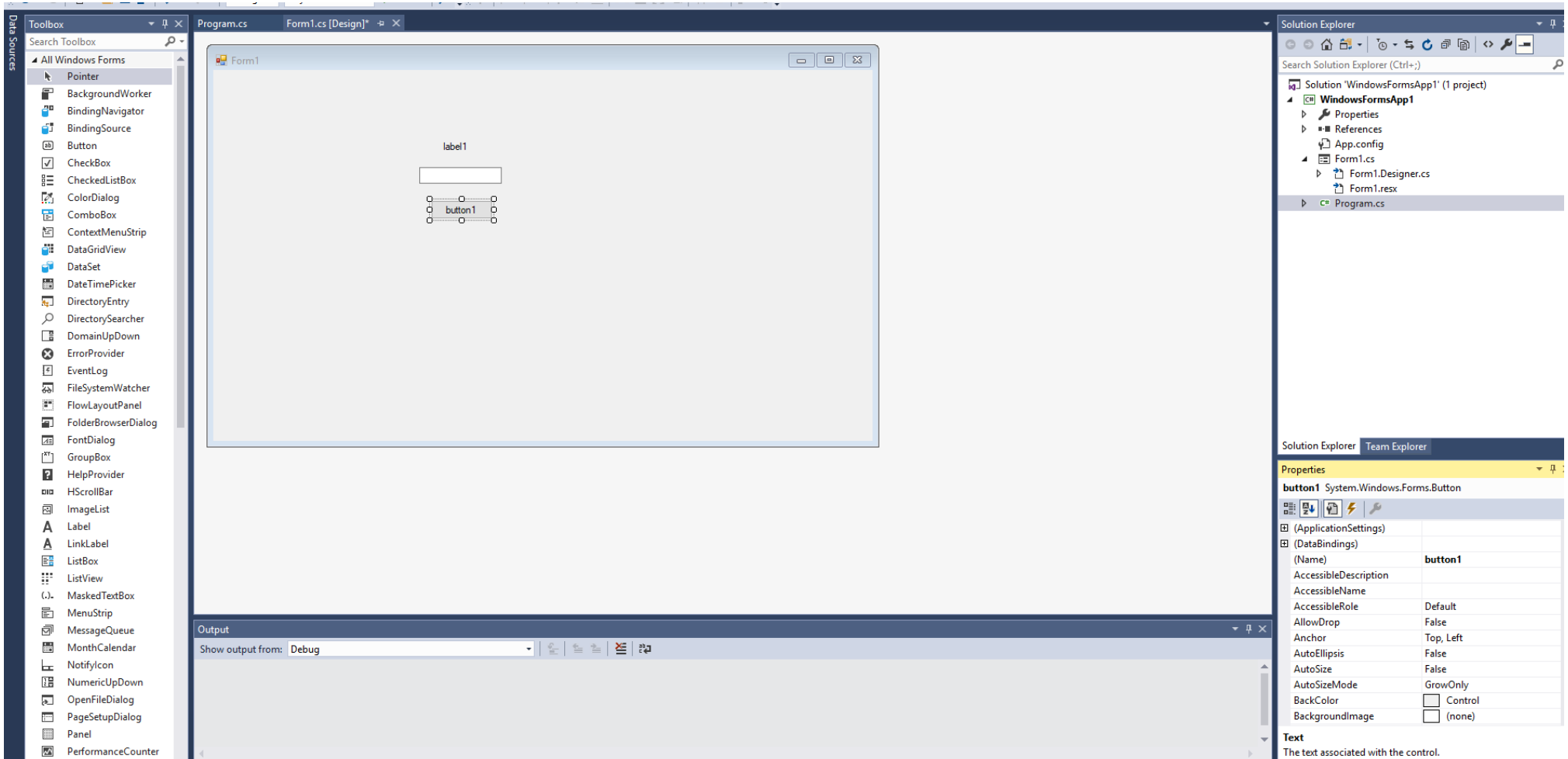
Program.cs File Properties

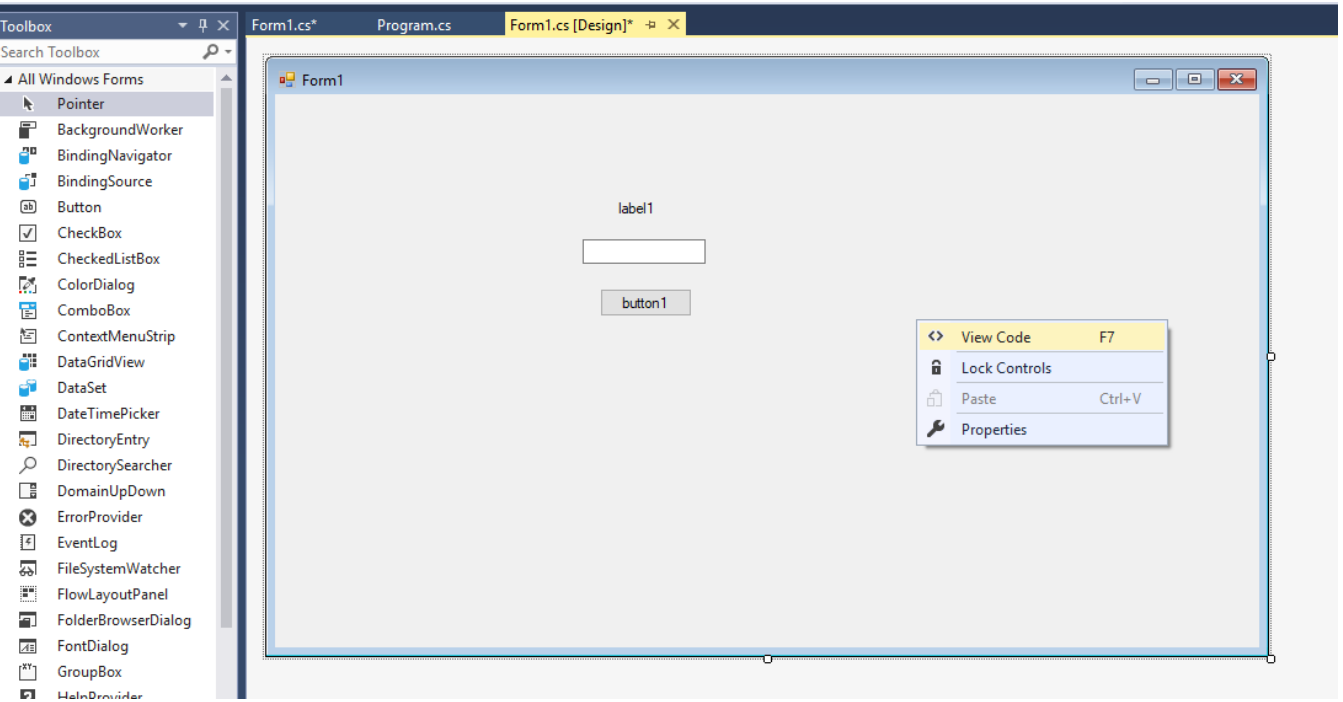
Build Action	Compile

GUI designer

A programmer can just drag and drop UI elements (called controls) onto the designer window.

The GUI designer will automatically create the code in Form1.Designer.cs.





To view the code, right-click on the form and select “View Code”.

```
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10
11  namespace WindowsFormsApp1
12  {
        3 references
13      public partial class Form1 : Form
14      {
            1 reference
15          public Form1()
16          {
17              InitializeComponent();
18          }
19      }
20  }
21
```

Form1.cs* | Program.cs | Form1.cs [Design]*

WindowsFormsApp1 | WindowsFormsApp1.Form1 | Form1()

View Designer	Shift+F7
Quick Actions and Refactorings...	Ctrl+.
Rename...	Ctrl+R, Ctrl+R
Remove and Sort Usings	Ctrl+R, Ctrl+G
Peek Definition	Alt+F12
Go To Definition	F12
Go To Implementation	Ctrl+F12
Find All References	Shift+F12
View Call Hierarchy	Ctrl+K, Ctrl+T
Create Unit Tests	
IntelliTest	▸
Code Map	▸
Breakpoint	▸
Run To Cursor	Ctrl+F10
Execute in Interactive	Ctrl+E, Ctrl+E
Snippet	▸
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Annotation	▸
Outlining	▸
Live Run	Shift+Alt+F5

The code automatically generated by the GUI designer is placed under InitializeComponents() function. To view it right click on it and choose “Go to Definition”.

Basic controls

- BackgroundWorker
- BindingNavigator
- BindingSource
- Button — Normal button
- CheckBox — Boolean type of input
- CheckedListBox
- ColorDialog — Color selection box
- ComboBox — User input with limited options
- ContextMenuStrip
- DataGridView
- DataSet
- DateTimePicker — Date time selection window
- DirectoryEntry
- DirectorySearcher
- DomainUpDown
- ErrorProvider
- EventLog
- FileSystemWatcher
- FlowLayoutPanel
- FolderBrowserDialog — Folder selection window
- FontDialog — Font selection window
- GroupBox
- HelpProvider
- HScrollBar — Horizontal scroll bar
- ImageList
- Label — For displaying text on windows

Basic controls

- LinkLabel — Hyperlink type of label
- ListBox
- ListView
- MaskedTextBox — Menu bar
- MenuStrip
- MessageQueue
- MonthCalendar
- NotifyIcon
- NumericUpDown
- OpenFileDialog — Window for opening files
- PageSetupDialog
- Panel
- PerformanceCounter
- PictureBox — For displaying images
- PrintDialog — For printing
- PrintDocument
- PrintPreviewControl
- PrintPreviewDialog
- Process
- ProgressBar — Progress bar
- PropertyGrid
- RadioButton
- RichTextBox — Textbox with formatting

Basic controls

- SaveFileDialog — Window for saving
- SerialPort — Serial port for communication with Arduino
- ServiceController
- SplitContainer
- Splitter
- StatusStrip
- TabControl
- TableLayoutPanel
- TextBox — Textbox for getting user input
- Timer — Timer
- ToolStrip
- ToolStripContainer
- ToolTip
- TrackBar — For getting numeric user input. User drags the bar instead of typing.
- TreeView
- VScrollBar
- WebBrowser

Common properties

Most controls have the following properties.

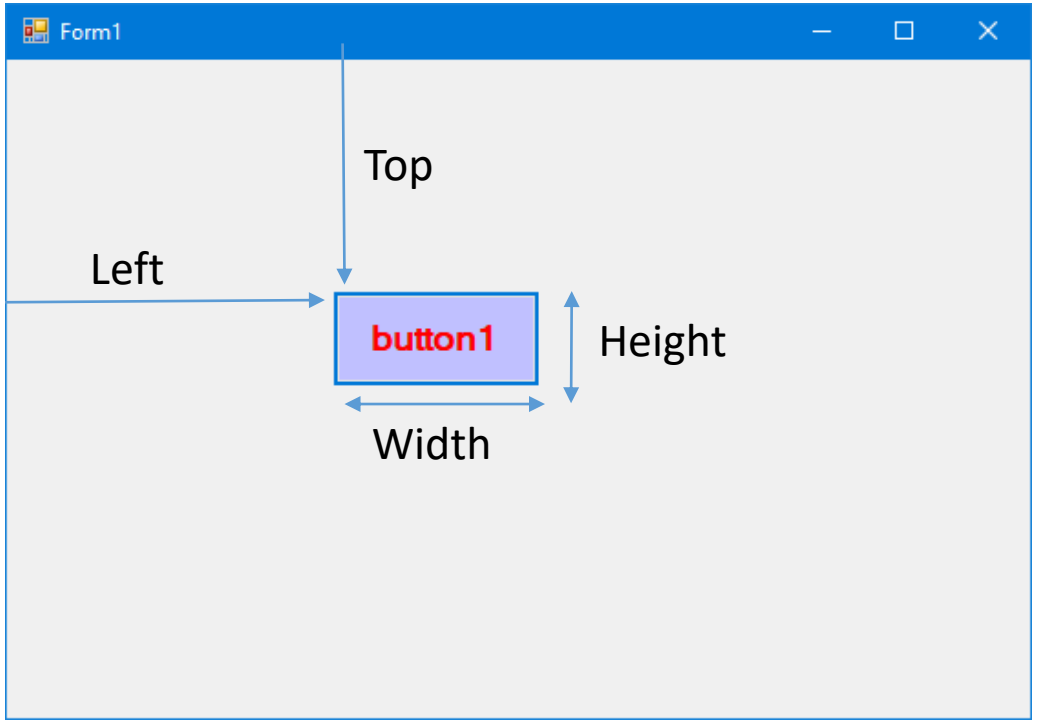
- Top
- Left

Combined as a property called Location.

- Length
- Width

Combined as a property called Size.

- Text
- Font
- ForeColor
- BackColor




```
button1.BackColor = Color.FromArgb(192, 192, 255)
button1.Font = new Font("Microsoft Sans Serif", 14.25F, FontStyle.Bold, GraphicsUnit.Point, ((byte)(0)));
button1.ForeColor = Color.Red;
button1.Location = new Point(178, 126);
button1.Size = new Size(114, 53);
button1.Text = "button1";
```

Color class

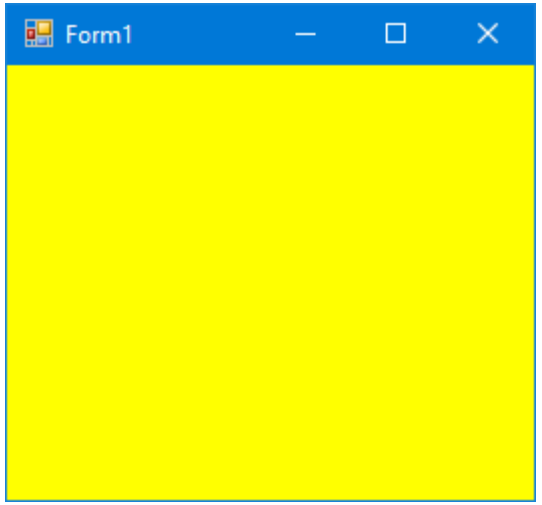
- Many properties of UI elements use the Color class, which is under the System.Drawing namespace.

- There are many built-in colors which can be accessed statically for example
  - Color.Red
  - Color.Blue
  - Color.Pink
- For full list of built-in colors, go to https://docs.microsoft.com/en-us/dotnet/api/system.drawing.color?view=netframework-4.7.2

Color class

- Any color can be made by combining 3 primary colors (red, green and blue) using the function Color.FromArgb(R, G, B).

```
public Form1()
{
    InitializeComponent();

    BackColor = Color.FromArgb(255, 255, 0);
}
```

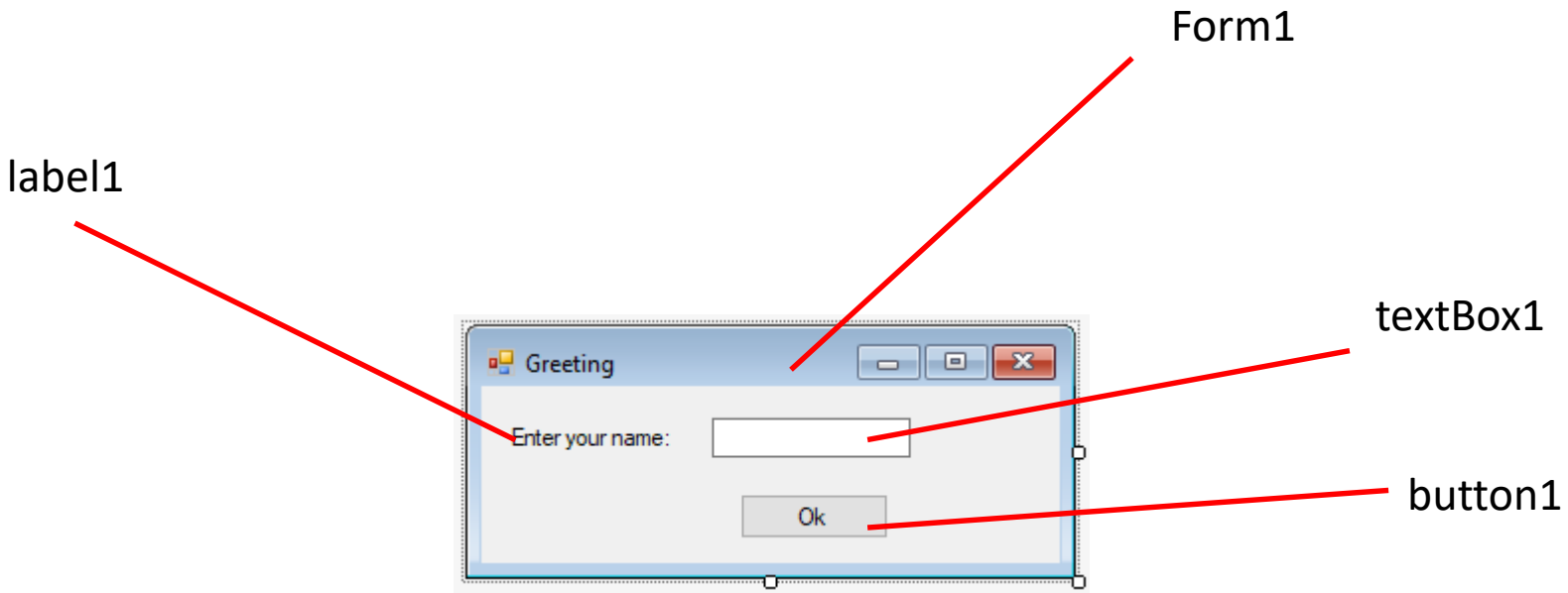


Common events

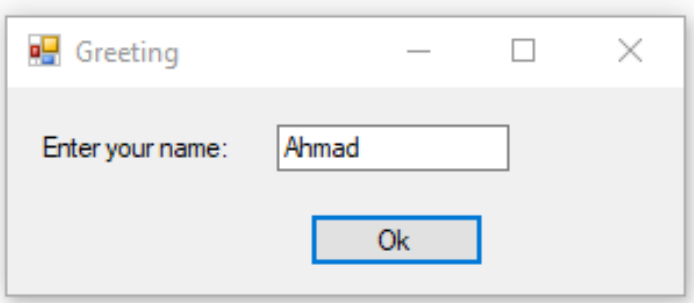
- Click (Triggered when the user click on the control)
- DragDrop (Triggered when the user drop something onto the control)
- MouseHover (Triggered when the user hovers their mouse onto the control)
- MouseLeave (Triggered when the user’s mouse leaves the control)
- TextChanged (Triggered when the control’s text got changed)

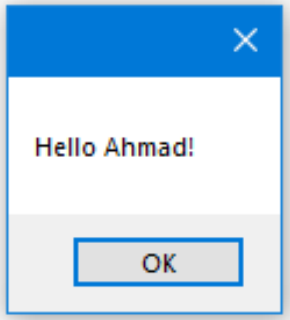
Exercise 1

Develop a GUI-based program that asks the user for their name and upon clicking the “OK” button it displays “Hello ****!” where **** is the user’s name.



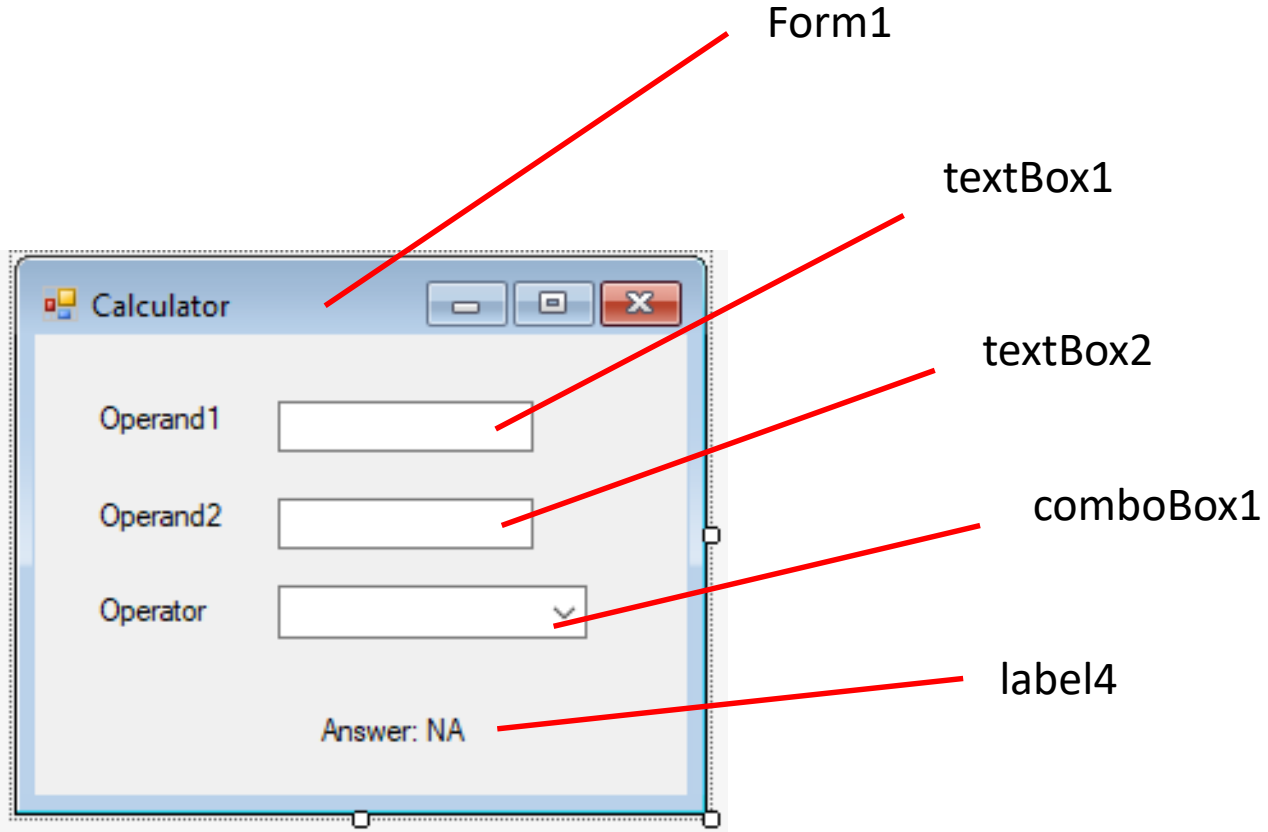
```
using System;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            button1.Click += Button1_Click;
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello " + textBox1.Text + "!");
        }
    }
}
```





Exercise 2

Develop a simple calculator that works on two operands and supports 4 types of operations (addition, subtraction, multiplication and division) as shown below. The program has no calculate button. As the user is typing, it should perform the selected operation on the fly and display the result. If the input is invalid, display “Answer: NA”

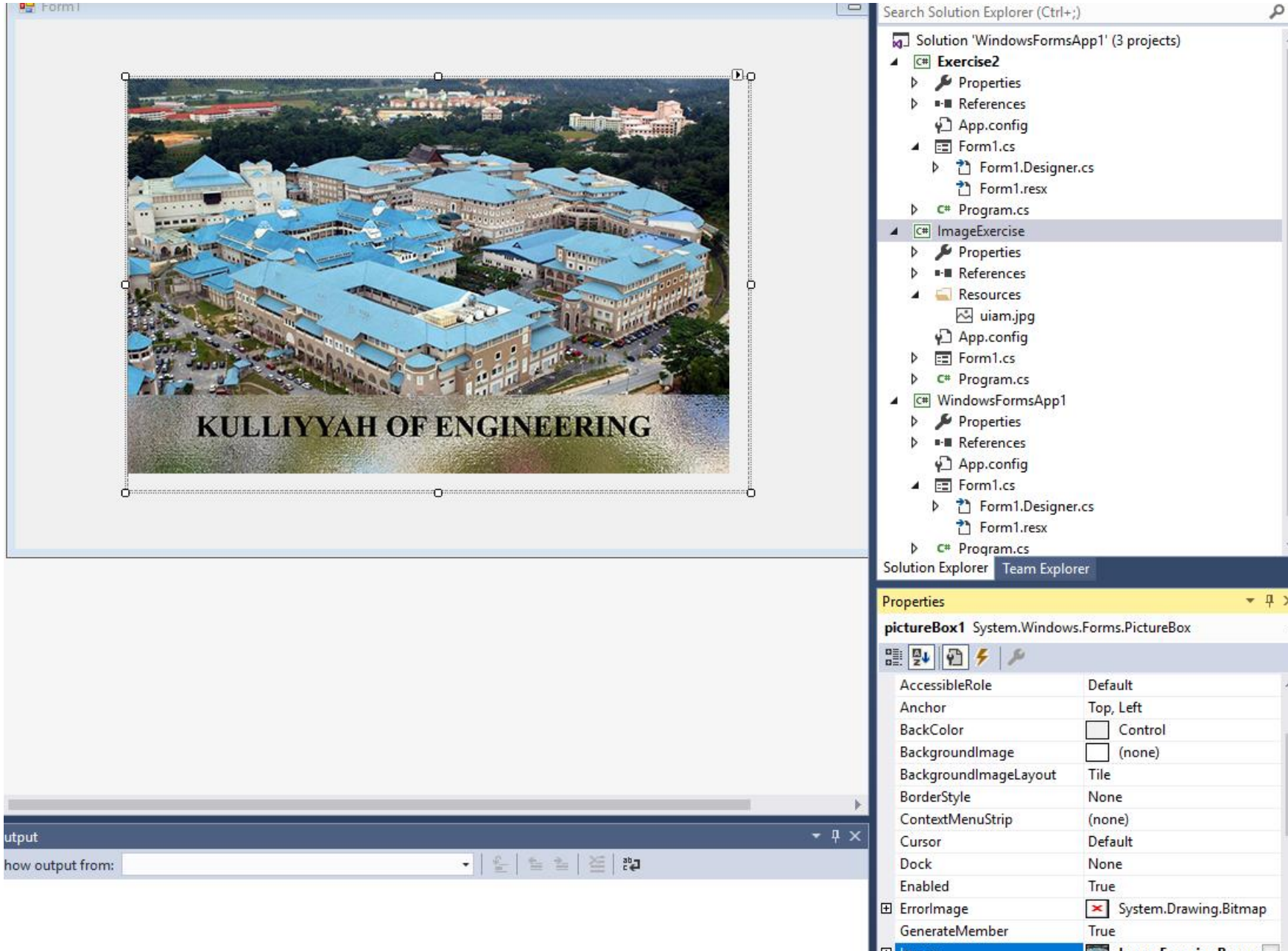


```
public Form1()
{
    InitializeComponent();


    comboBox1.Items.Add("Addition");
    comboBox1.Items.Add("Subtraction");
    comboBox1.Items.Add("Multiplication");
    comboBox1.Items.Add("Division");
    comboBox1.SelectedIndex = 0;

    textBox1.TextChanged += (a, e) => Calculate();
    textBox2.TextChanged += (a, e) => Calculate();
    comboBox1.SelectedIndexChanged += (a, e) => Calculate();
}
```

```
private void Calculate()
{
    try
    {
        double number1 = double.Parse(textBox1.Text);
        double number2 = double.Parse(textBox2.Text);

        double output = 0;

        switch (comboBox1.SelectedIndex)
        {
            case 0:
                output = number1 + number2;
                break;
            case 1:
                output = number1 - number2;
                break;
            case 2:
                output = number1 * number2;
                break;
            case 3:
                output = number1 / number2;
                break;
        }

        label4.Text = output.ToString();
    }
    catch
    {
        label4.Text = "Answer: NA";
    }
}
```

PictureBox

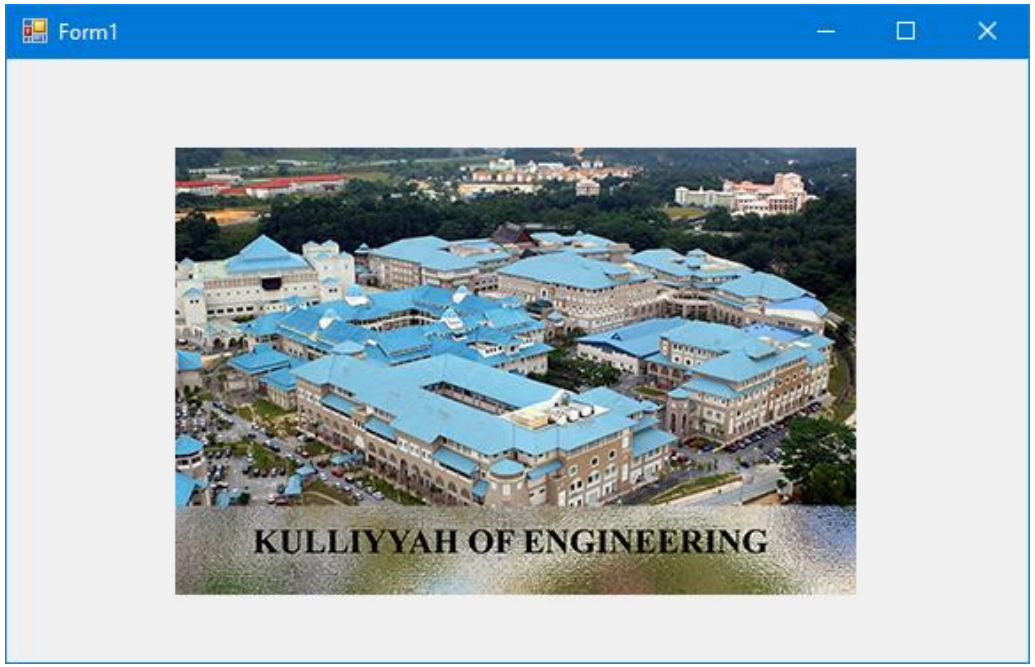
- The PictureBox control display images.
- The **Image** property of the PictureBox class gets or sets the image of the PictureBox.
- Images can be either embedded into the program at compiled time or can be read at run-time from a specific location.



Example

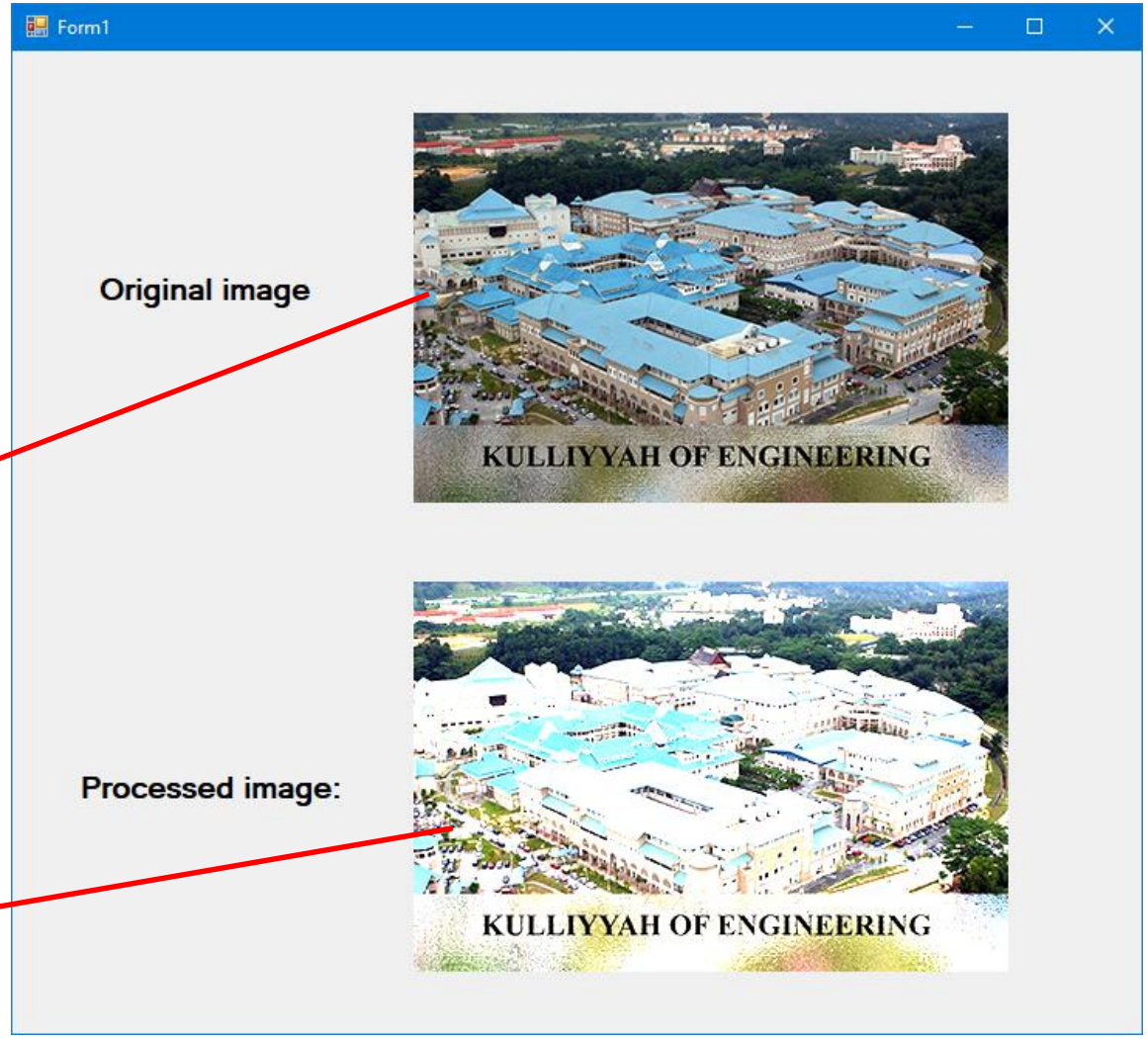
The following code sets the image of a PictureBox at run time.

```
using System.Drawing;
using System.Windows.Forms;

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

            Bitmap image = Bitmap(@"C:\uiam.jpg");
            pictureBox1.Image = image;

        }
    }
}
```



Exercise

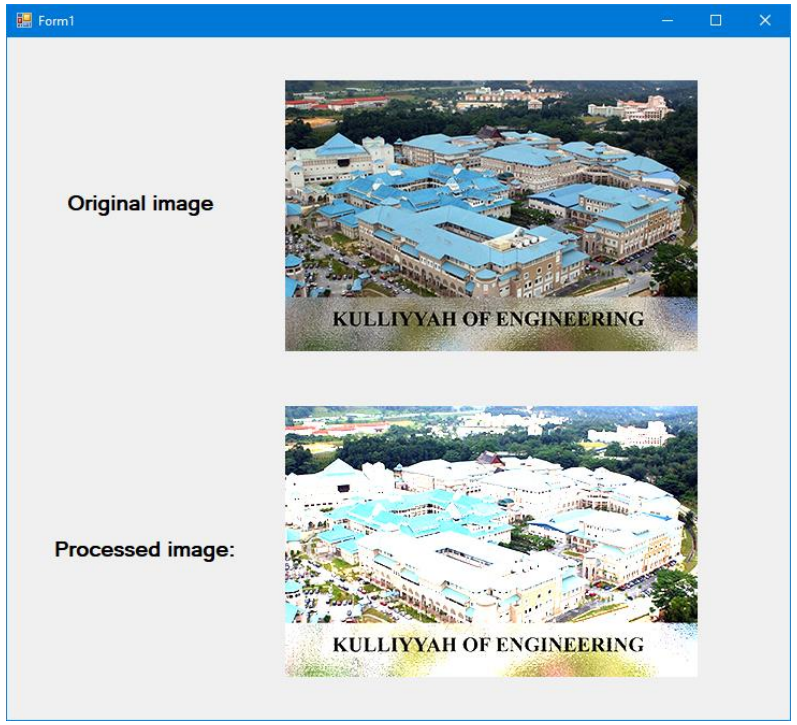
Develop a program that increases the brightness of an image.



PictuerBox1

PictuerBox2

```
public Form1()
{
    InitializeComponent();

    Bitmap image = new Bitmap(@"C:\uiam.jpg");
    pictureBox1.Image = image;
    pictureBox2.Image = Process(image);
}

private Bitmap Process(Bitmap image)
{
    Bitmap processed_image = new Bitmap(image.Width, image.Height);
    for (int row = 0; row < processed_image.Width; row++)
    {
        for (int col = 0; col < processed_image.Height; col++)
        {
            Color color = image.GetPixel(row, col);

            int new_red = color.R * 2;
            int new_green = color.G * 2;
            int new_blue = color.B * 2;

            new_red = (new_red > 255) ? 255 : new_red;
            new_green = (new_green > 255) ? 255 : new_green;
            new_blue = (new_blue > 255) ? 255 : new_blue;
            processed_image.SetPixel(row, col, Color.FromArgb(new_red, new_green, new_blue));
        }
    }
    return processed_image;
}
```



- The image’s dimension is 284x252 only.
- The previous code uses the GetPixel method, which is very slow. The processing time is around 150ms on Core i7 8700 CPU.
- For video, the expected frame rate is only 6 frames per second. It cannot be used for a real-time video stream.
- The performance can be improved by using pointer operations like C++ by turning on unsafe mode.

```
private unsafe Bitmap UnsafeProcess(Bitmap image)
{
    Bitmap processed_image = new Bitmap(image);
    BitmapData imageData = processed_image.LockBits(new Rectangle(0, 0, processed_image.Width, processed_image.Height), ImageLockMode.ReadWrite,
                PixelFormat.Format24bppRgb);
    int bytesPerPixel = 3;

    byte* scan0 = (byte*)imageData.Scan0.ToPointer(); //Pointer that points to the base of the image
    int stride = imageData.Stride;

    for (int col = 0; col < imageData.Height; col++)
    {
        byte* rowdata = scan0 + (col * stride); //Pointer that points to the base row

        for (int row = 0; row < imageData.Width; row++)
        {
            int red = rowdata[row * bytesPerPixel];
            int green = rowdata[row * bytesPerPixel + 1];
            int blue = rowdata[row * bytesPerPixel + 2];

            red *= 2;
            green *= 2;
            blue *= 2;

            rowdata[row * bytesPerPixel] = (red > 255) ? (byte)255 : (byte)red;
            rowdata[row * bytesPerPixel + 1] = (green > 255) ? (byte)255 : (byte)green;
            rowdata[row * bytesPerPixel + 2] = (blue > 255) ? (byte)255 : (byte)blue;
        }
    }

    processed_image.UnlockBits(imageData);

    return processed_image;
}
```

- Processing time is now around 2.1ms per frame or 476 frames per second.

Timer

- Timers trigger certain events regularly set a set interval.
- C# has many different timers.
- The most common is System.Windows.Forms.Timer which is tied to the man UI thread.
- Other timers are System.Timers.Timer and System.Threading.Timer.

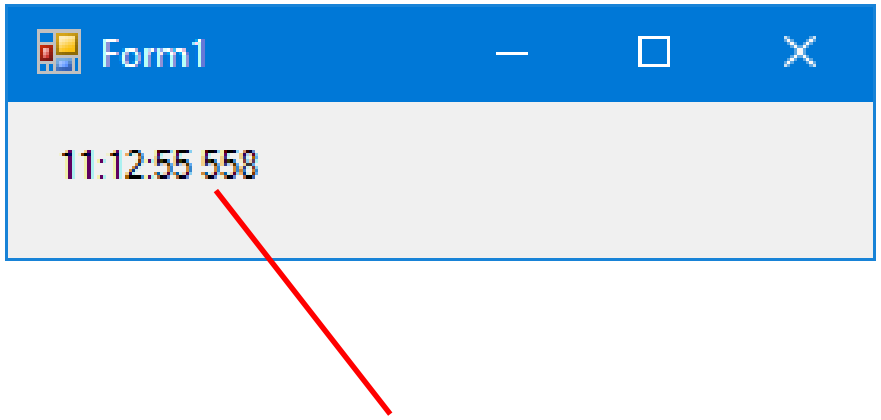
- System.Windows.Forms.Timer has two important properties
  - Enable (Boolean: setting it as ‘true’ will enable the timer)
  - Interval (Integer: in millisecond)

- System.Windows.Forms.Timer has Elapsed event that gets triggered at the set interval.

Example

The program display the system's current time up to ms.

```
using System;
using System.Windows.Forms;

namespace TimerExercise
{
    public partial class Form1 : Form
    {

        private Timer timer1 = new Timer();

        public Form1()
        {
            InitializeComponent();
            timer1.Interval = 1;
            timer1.Tick += (a, e) => label1.Text = DateTime.Now.ToString("HH:mm:ss fff");
            timer1.Enabled = true;
        }
    }
}
```



label1