

Course: EGDF20
Module: EGE202 Application Programming

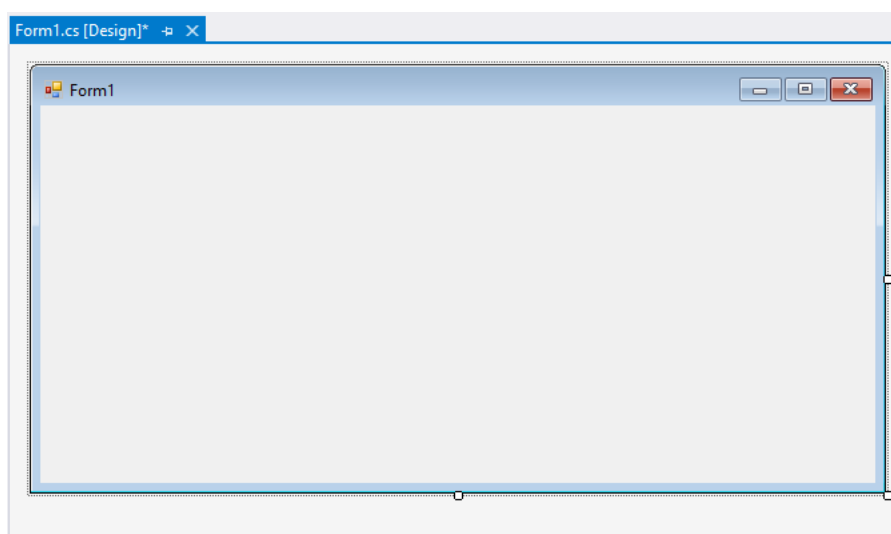
Practical 2: Building an Interactive Application

Objectives: At the end of this lab, the student should be able to develop simple interactive application and understand how event driven model of GUI software application can provide such interactivity.

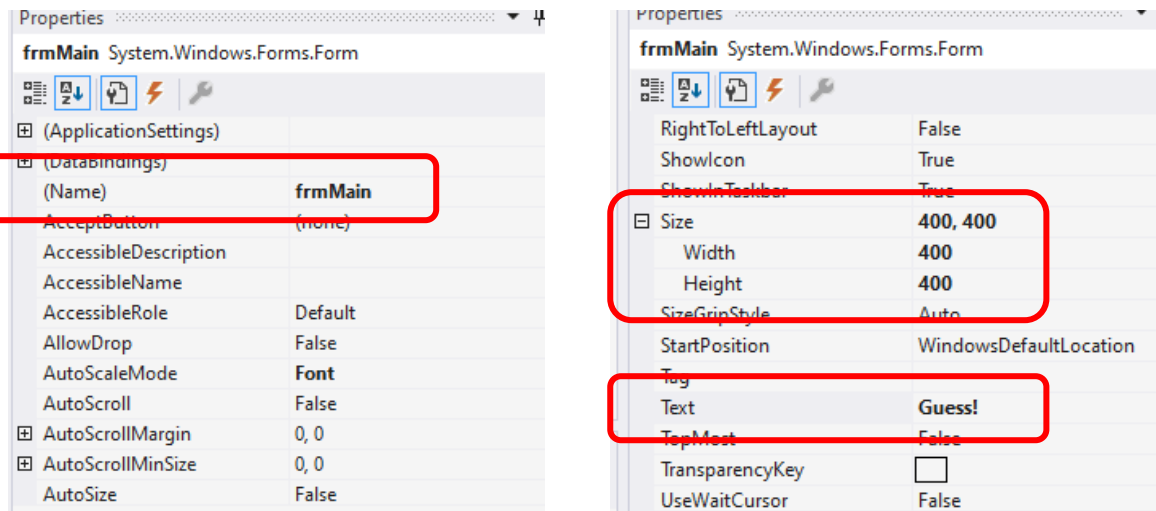
Exercise 1 – Building an Interactive GUI Application

Part 1: GUI Design

1. Under the **File** menu, click **New Project** or use the **New Project** button to create a new project. Alternatively, use the **Create New Project** link in the **Get Started** popup dialog.
2. From the pop-up dialog, select “C#” for the **Language filter**, “Windows” for the **Platform filter** and “Desktop” for the **Project type filter**.
3. Then choose **Windows Forms App (.Net Framework)** and click the **Next** button.
4. Type the name of your new project as **Guess!** and keep the Solution name the same as Project name.
5. Set the Location to put the project in your own created folder.
6. **Do not** tick on the check-box of [☐ **Place solution and project in the same directory**].
7. Click the **Create** button to start your project.
8. In the **Properties** window of the **Form** control, change the **TopMost** property of the **Form1** to ‘True’.
9. Form is the main application container control that presents UI (User Interface) to users. It comes with Application Title Bar. The default **Form** file name and class name is always **Form1**.

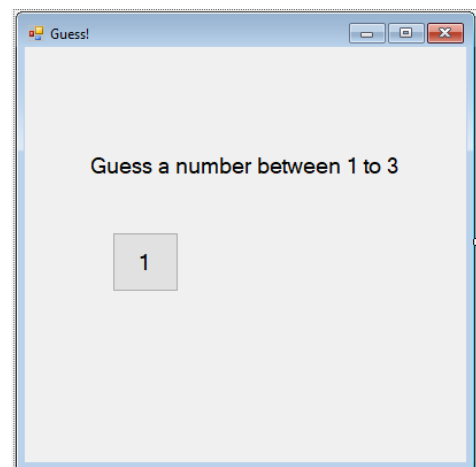


10. Double click on “Form1.cs” **Solution Explorer** window to launch the **Form Designer** tab.
11. Change the **TopMost** property of the **Form1** to ‘True’.
12. Change the **Text** property of the **Form** from ‘Form1’ to ‘Guess!’ and the **Name** property from ‘Form1’ to ‘frmMain’
13. Finally set the **Size (Width & Height)** to 400 by 400.



14. From the **Toolbar**, drag in 1 **Button** and 1 **Label** control into the **frmMain** window area. Modify the properties based on the table below and resize **frmMain** accordingly

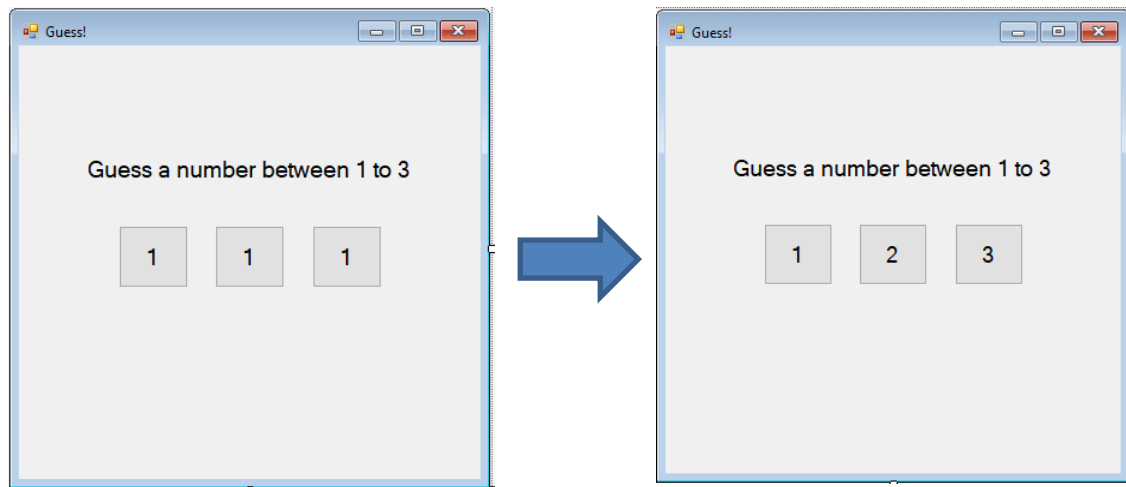
{Name} From	{Name} To	{Text}	{Font->Size}
Form1	frmMain	Guess!	
label1	lblDisplay	Guess a number between 1 to 3	14
button1	btn1	1	14



15. Resize and rearrange/re-position the Button and Label control accordingly.

No	Actions	Observation / Explanation/Action
1	Try resizing the Button control in the Form Designer using your mouse. Which properties would it change?	
2	Try re-position the Button control in the Form Designer using your mouse. Which properties would it change?	

16. Next in the Form Designer, **select the Button '1'** then **copy the button using <Ctrl + C>** key. Subsequently **paste <Ctrl + V> 2 times** and rearrange them according to the figure below.



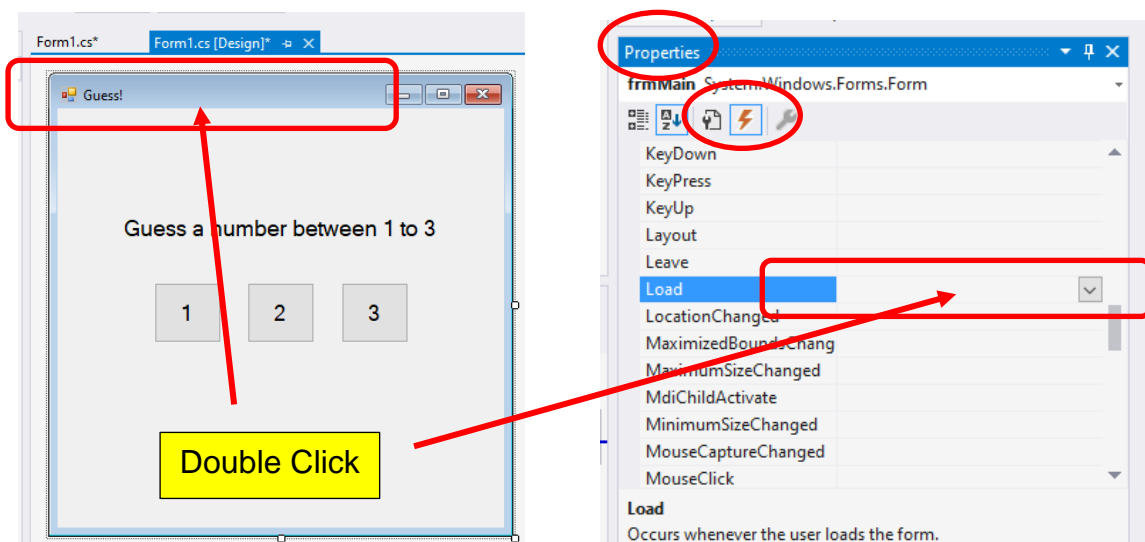
17. Finally rename the **Text** and **Name** properties of the buttons such that **'1'** correspond to **'btn1'**, **'2'** correspond to **'btn2'** and **'3'** correspond to **'btn3'**.
18. Build and run your application by hitting <F5> or <Ctrl + F5> key. You should see an application with the UI shown in the diagram above.

Part 2: Game Logic Design

1. Now let's create a random secret number between 1 to 3. When shall we create this random number?
At the point when the application starts?

That corresponds to the application LOAD event

2. At the **Form Designer** tab, double click on the **Form Title Bar**.
(Alternatively, select the Form at the **Form Designer** tab and then proceed to the **Properties Panel** event's tab to double click on the dropdown list next to Load)



3. The action in the previous step will automatically create a **LOAD** event handler. Add the following codes into the **LOAD** event handler

```
private void frmMain_Load(object sender, EventArgs e)
{
    secretNum = r.Next(1, 4);
}
```

4. Noticed that the added line is being highlighted with a red color wavy underscore.

This is a Syntax Error. Any attempt to compile or build the codes or application will flag an error

```
1 reference
private void frmMain_Load(object sender, EventArgs e)
{
    secretNum = r.Next(1, 4);
}
```

5. To fix the syntax error, we need to declare what is **secretNum** and **r**. Add the following codes for declaration. Note that we are declaring global variables so that it can be assessable to any functions.

```
public partial class frmMain : Form
{
    Random r = new Random();
    int secretNum;

    public frmMain()
    {
        InitializeComponent();
    }

    private void frmMain_Load(object sender, EventArgs e)
    {
        secretNum = r.Next(1, 4);
    }
}
```

Int is on the supported datatypes. It simply means **secretNum** can only integer values and not others such as float, String etc.

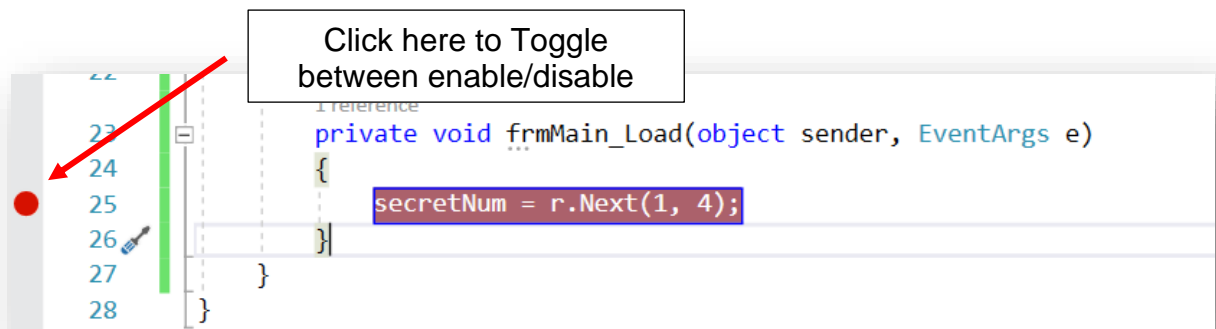
Random is class datatype, defined in .NET libraries. (In programming, a library is just a collection of functions and classes). The variable **r** is instantiated (create an instance of object based on a specific Class) from **Random** Class. Being an object it has properties and methods (such as the **Next()** method which generate a random number).

6. Build and run your application. Ensure that there are no errors.


- Let's observe the value of secretNum to see if it is indeed randomized. To do this we will need to add breakpoints.

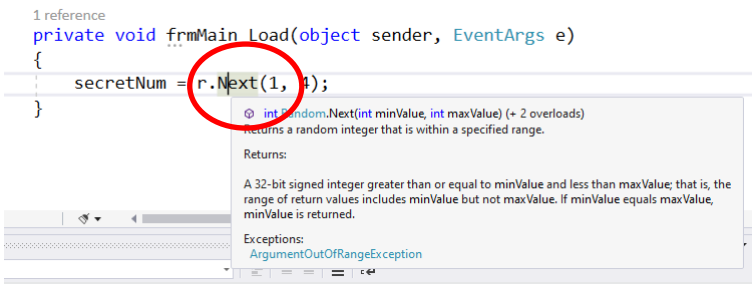
Breakpoints is used to temporarily pause the program execution so that you can observe the state of your application such as examining the value of the various variables.

- Setup a breakpoint at the following location by clicking on the shaded area, to the right of the numbering, next to the code.



- With the breakpoint set, perform the following activity.

No	Actions	Observation
1	Click Start to run the application. Did the codes stop at the breakpoint? Mouseover the cursor to the variable secretNum. What is the observed current value?	
2	Click the StepOver button once. What does it perform?	
3	Mouseover the cursor to the variable secretNum. What is the observed current value?	
4	Click Stop and repeat Step 1 to 3. Observe the value of secretNum	

5	<p>Mouse over the cursor to the method <i>Next(...)</i>.</p> <p>From the pop up description of the method, what is the range of values to be produced by the Random object <i>r.Next(...)</i></p>	 <pre> 1 reference private void frmMain_Load(object sender, EventArgs e) { secretNum = r.Next(1, 4); } </pre> <p>int Random.Next(int minValue, int maxValue) (+ 2 overloads) Returns a random integer that is within a specified range.</p> <p>Returns: A 32-bit signed integer greater than or equal to <i>minValue</i> and less than <i>maxValue</i>; that is, the range of return values includes <i>minValue</i> but not <i>maxValue</i>. If <i>minValue</i> equals <i>maxValue</i>, <i>minValue</i> is returned.</p> <p>Exceptions: ArgumentOutOfRangeException</p>
---	---	--

10. Now let's complete the application. On the Form Designer, double click on each of the buttons labeled '1', '2', and '3' to add **Click** event handlers.
11. Then add the following codes for ***btn1_Click()***

```

private void btn1_Click(object sender, EventArgs e)
{
    if (secretNum == 1)
    {
        MessageBox.Show("Correct");
    }
    else
    {
        MessageBox.Show("Wrong");
    }
    Button btn = (Button)sender;
    btn.Enabled = false;
}

```

12. **Repeat with proper modification** from previous step for ***btn2_Click()*** and ***btn3_Click()***
13. Build and test the application.

Part 3: Adding Interactivity

Allow User to Restart the Guessing Game

1. In the **Code Editor**, add the following function to the codes in **Form1.cs**

```
private void frmMain_Load(object sender, EventArgs e)
{
    secretNum = r.Next(1, 4);
}

private void startOver()
{
    var result = MessageBox.Show("Start Over?", "End of Game",
    MessageBoxButtons.YesNo);

    secretNum = r.Next(1, 4); // regenerate new random number

    if (result == DialogResult.Yes)
    {
        btn1.Enabled = true;
        btn2.Enabled = true;
        btn3.Enabled = true;
    }
    else
    {
        Application.Exit();
    }
}
```

2. Then add the following codes for **btn1_Click()**

```
private void btn1_Click(object sender, EventArgs e)
{
    if (secretNum == 1)
    {
        MessageBox.Show("Correct");
    }
    else
    {
        MessageBox.Show("Wrong");
    }
    Button btn = (Button)sender;
    btn.Enabled = false;
    startOver();
}
```

3. **Repeat with proper modification** from previous step for **btn2_Click()** and **btn3_Click()**
4. Build and test the application.