



Chapter 6-2

Introduction to Windows Form Application

6-6 Set Properties in Source Code

Grammar

```
Object.propertyName = value;
```

Ex1: set the Text property of button1 to “計算”, the property value is a string, usage:

```
button1.Text = "計算";
```

Ex2: set the Width property of button1 to 75, the property value is an integer, usage:

```
button1.Width = 75;
```

Ex3: set the Enabled property of button1 to false, the property value is a Boolean, usage:

```
button1.Enabled = false;
```

2. Integrated Enumeration Class

Grammar

```
Object.propertyName = Enumeration.member;
```

Example

```
Object.ForeColor = Color.member;
```

```
Object.BackColor = Color.member;
```

The list of Color enumeration in common use:

| Member | Color | Member | Color | Member | Color |
|--------|-------|-----------|------------|---------|----------|
| Black | black | Navy | navy blue | Red | red |
| Blue | blue | Olive | olive | Silver | silver |
| Brown | brown | Orange | orange | SkyBlue | sky blue |
| Gold | gold | OrangeRed | orange red | Tomato | tomato |
| Green | green | Pink | pink | White | white |
| Gray | gray | Purple | purple | Yellow | yellow |

Ex: set the BackColor property of label4 to yellow

```
label4.BackColor = Color.Yellow;
```

Use the principal of 3 primary colors – Red, Green, Blue, use method FromArgb(R, G, B) to mixing color, the number scope of color is 0~255

Grammar

```
Object.ForeColor = Color.FromArgb(R, G, B);
```

```
Object.BackColor = Color.FromArgb(R, G, B);
```

Ex1: set the background color of the form to “blue”

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

“this” stands for the form, this.BackColor stands for the background color of the form

Ex2: set the background color of button1 to “purple” (red + blue)

```
button1.BackColor = Color.FromArgb(255, 0, 255);
```

Ex3: set the background color of the form to white(red + green + blue)

```
this.BackColor = Color.FromArgb(255, 255, 255);
```




Ex4: set the background color of textBox1 to black

```
textBox1.BackColor = Color.FromArgb(0, 0, 0);
```

Ex5: set the background color of the form to gray

```
this.BackColor = Color.FromArgb(125, 125, 125);
```

2. BorderStyle Enumeration: BorderStyle

| Member | None | FixedSingle | Fixed3D |
|-------------|---|---|---|
| Description | No border | Single line border | 3-D border line |
| Example |  |  |  |
| Code | <code>BorderStyle.None</code> | <code>BorderStyle.FixedSingle</code> | <code>BorderStyle.Fixed3D</code> |

Ex: set the border of label1 to 3-D border line, usage:

`label1.BorderStyle = BorderStyle.Fixed3D;`

3. TextAlign Enumeration: ContentAlignment

| | | |
|------------|--------------|-------------|
| TopLeft | TopCenter | TopRight |
| MiddleLeft | MiddleCenter | MiddleRight |
| BottomLeft | BottomCenter | BottomRight |

Ex: set the align of the text “計算” of button1 to the right-bottom of the control item, usage:

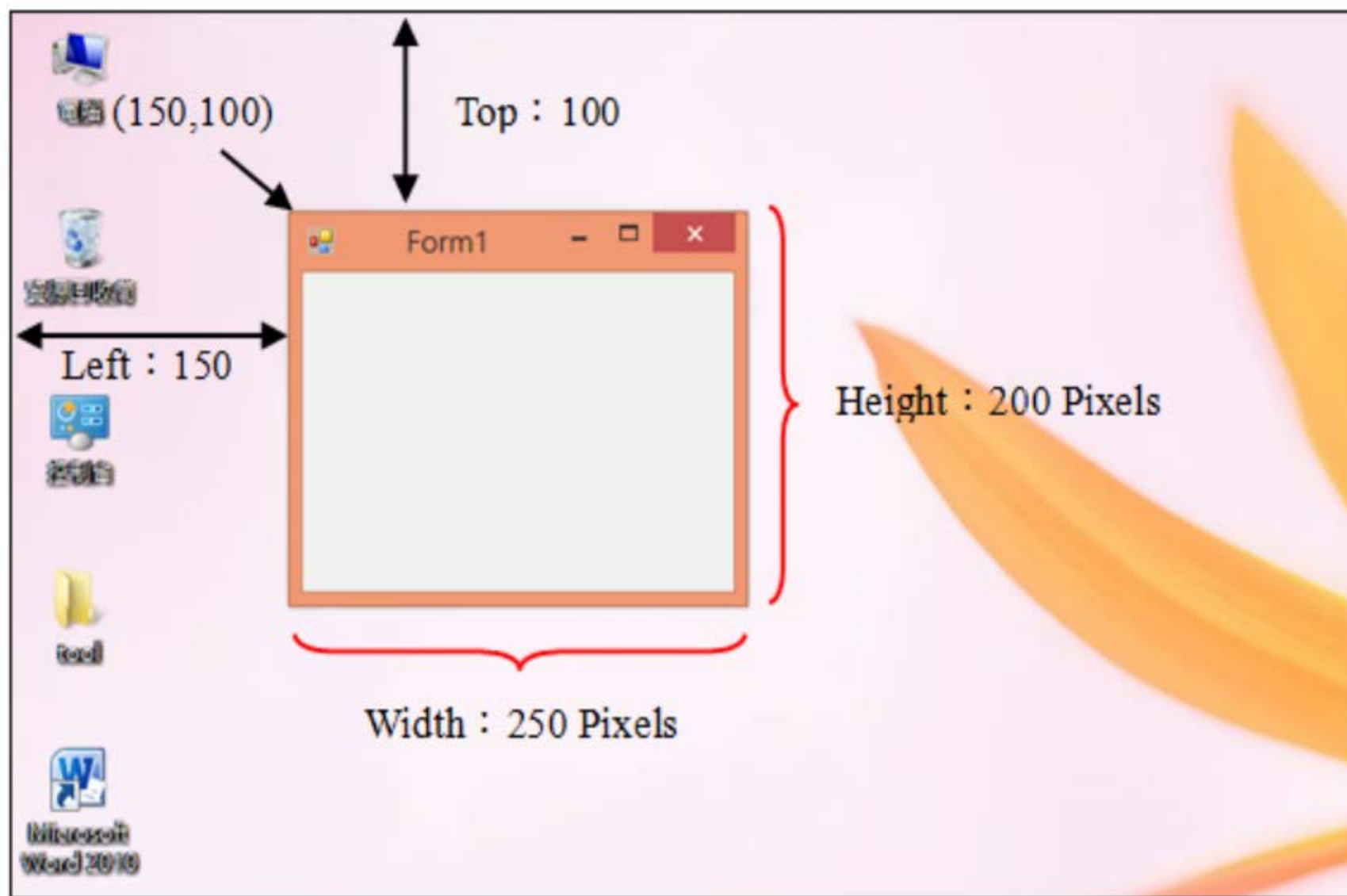


```
button1.TextAlign = ContentAlignment.BottomRight;
```

3. new

Grammar

```
Object.propertyName = new className(arg1, arg2, ...);
```





**Ex: use Top and Left properties to set the coordinate,
usage:**

```
this.Top = 150;  
this.Left = 100;
```

**Ex: set the coordinate of form to (150, 100) from left-top
of the screen, usage:**

```
this.Location = new Point(150, 100);
```



Ex: use Width and Height properties to set the size of the form, usage:

```
this.Width = 250;  
this.Height = 200;
```

Ex: Size property can include Width and Height properties, usage:

```
this.Size = new Size(250, 200);
```

Image and BackgroundImage Properties

| Grammar |
|--|
| <pre>Object.Image = Image.FromFile("imagePath"); Object.BackgroundImage = Image.FromFile("imagePath");</pre> |

| Grammar |
|--|
| <pre>Object.Image = Image.Bitmap("imagePath"); Object.BackgroundImage = Image.Bitmap("imagePath");</pre> |

Ex1: load "C:\cs\ch06\duck.jpg" as the background image of the form:

```
this.BackgroundImage = Image.FromFile ("c:\\cs\\ch06\\duck.jpg");  
this.BackgroundImage = new Bitmap("c:\\cs\\ch06\\duck.jpg");
```

Ex2: remove the background image of the form

```
this.BackgroundImage = null;
```

3. Use Font property to set the style of the Text property content, like font, size, style, etc. usage:

Grammar

```
Object.Font = New Font(FontName, FontSize, FontStyle) ;
```

Five font styles:

1. FontStyle.Bold
2. FontStyle.Italic
3. FontStyle.Regular
4. FontStyle.Strikeout
5. FontStyle.Underline

Use HorizontalAlignment to specifies how an object or text in a control is horizontally aligned relative to an element of control

| Member | Description |
|--------|--|
| Center | The control item of center alignment of object or text ◦ |
| Left | The control item of left alignment of object or text ◦ |
| Right | The control item of right alignment of object or text ◦ |

Grammar

```
TextBox.TextAlign = HorizontalAlignment.Right
```



Use System.Drawing namespace provides access to basic graphics functionality, take color for example

Grammar

```
Picturebox.backcolor = System.Drawing.color.Black
```

Example (change)

- Test 1 :

label1 – BackColor: LightPink ; ForeColor: Yellow

label2 – Font: 標楷體, 30, Bold

button3 – TextAlign: topLeft

textBox1 – TextAlign : Right

pictureBox1 – BackColor: Black

Example (change)

- Test 2 :

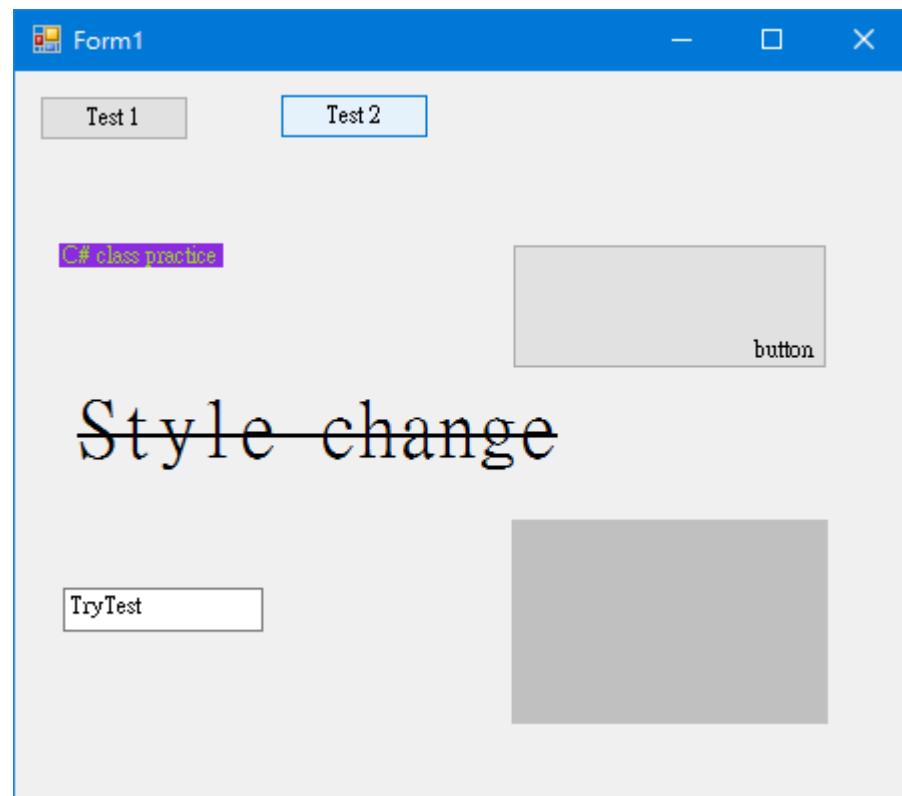
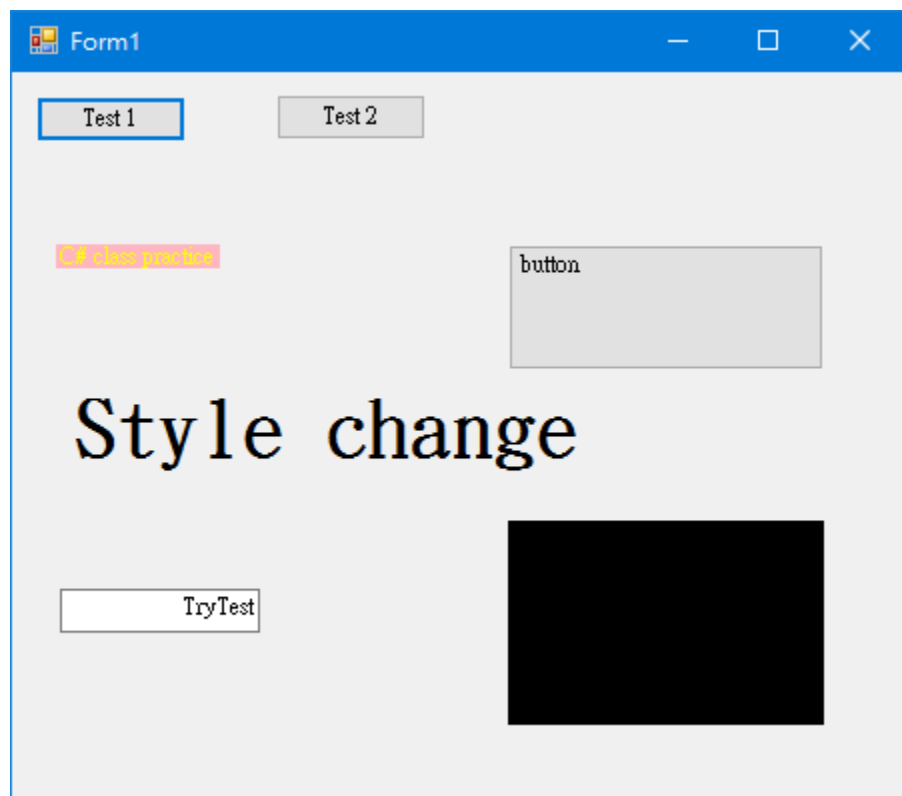
label1 – BackColor: BlueViolet ; ForeColor: YellowGreen

label2 – Font: 細明體, 30, Strikeout

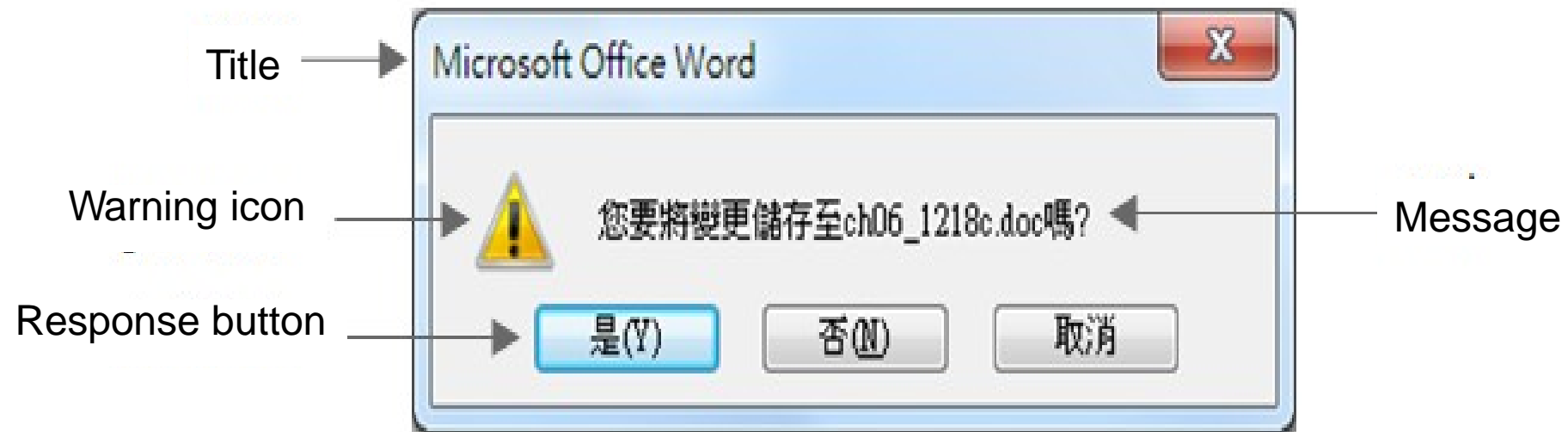
button3 – TextAlign: BottomRight

textBox1 – TextAlign : Left

pictureBox1 – BackColor: Silver



6-7 MessageBox Class









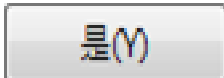
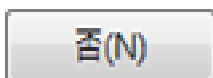
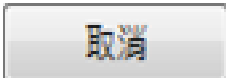
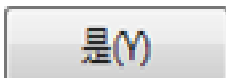



MessageBox.Show Method

Grammar

```
MessageBox.Show(msg[, title[, buttonConst[, iconConst]]]);
```

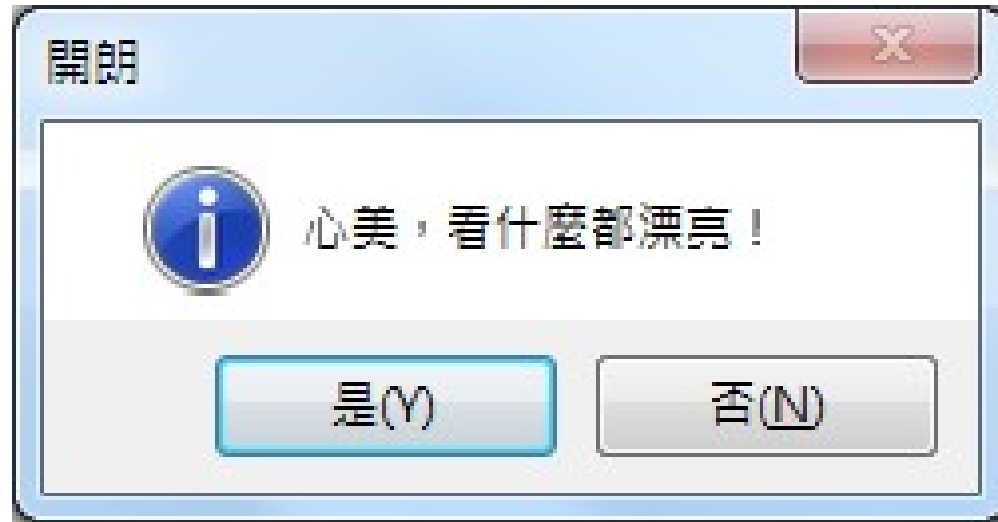
```
retValue = MessageBox.Show(msg[, title[, buttonConst[, iconConst]]]);
```

MessageBox.Show – Button Constant

| Button constant | Representation |
|------------------------------------|---|
| MessageBoxButtons.OK |  |
| MessageBoxButtons.OKCancel |   |
| MessageBoxButtons.AbortRetryIgnore |    |
| MessageBoxButtons.YesNoCancel |    |
| MessageBoxButtons.YesNo |   |
| MessageBoxButtons.RetryCancel |   |

MessageBox.Show – Icon Constant



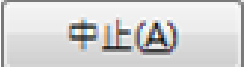
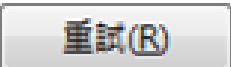
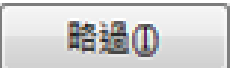


| Icon Constant | Representation | Description |
|----------------------------|---|---------------------|
| MessageBoxIcon.None | No icon | |
| MessageBoxIcon.Hand |  | Error message |
| MessageBoxIcon.Question |  | Question message |
| MessageBoxIcon.Exclamation |  | Warning message |
| MessageBoxIcon.Asterisk |  | Information message |



Usage:

```
MessageBox.Show("心美，看什麼都漂亮！", "開朗",  
    MessageBoxButtons.YesNo,  
    MessageBoxIcon.Question);
```

MessageBox.Show Method – Return Value

| Button | Returned constant |
|---|---------------------|
|  | DialogResult.OK |
|  | DialogResult.Cancel |
|  | DialogResult.Abort |
|  | DialogResult.Retry |
|  | DialogResult.Ignore |
|  | DialogResult.Yes |
|  | DialogResult.No |


Example(WindowsFormMessageBox): if the user presses Yes button, then close the program window. Usage:

```
DialogResult result;      //宣告一個 result 變數來存放傳回值
result = MessageBox.Show ("關閉視窗?", "結束", MessageBoxButtons.YesNo);
if(result == DialogResult.Yes)
{
    Application.Exit();
}
```

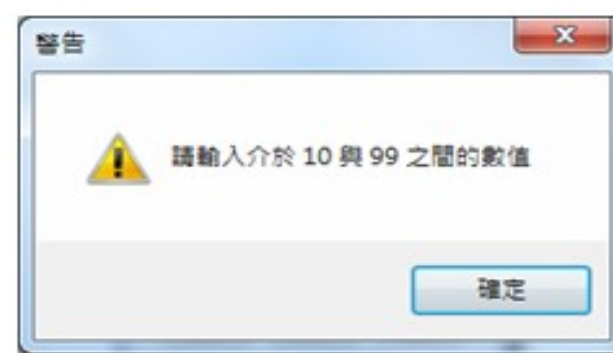
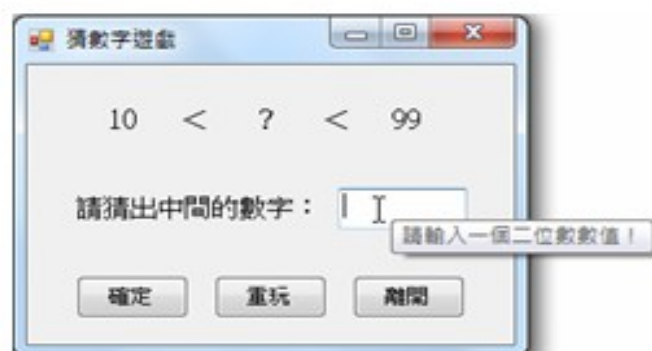
Example(Guess):

Design a number-guess game which generates a 2-digit random number, requirements:

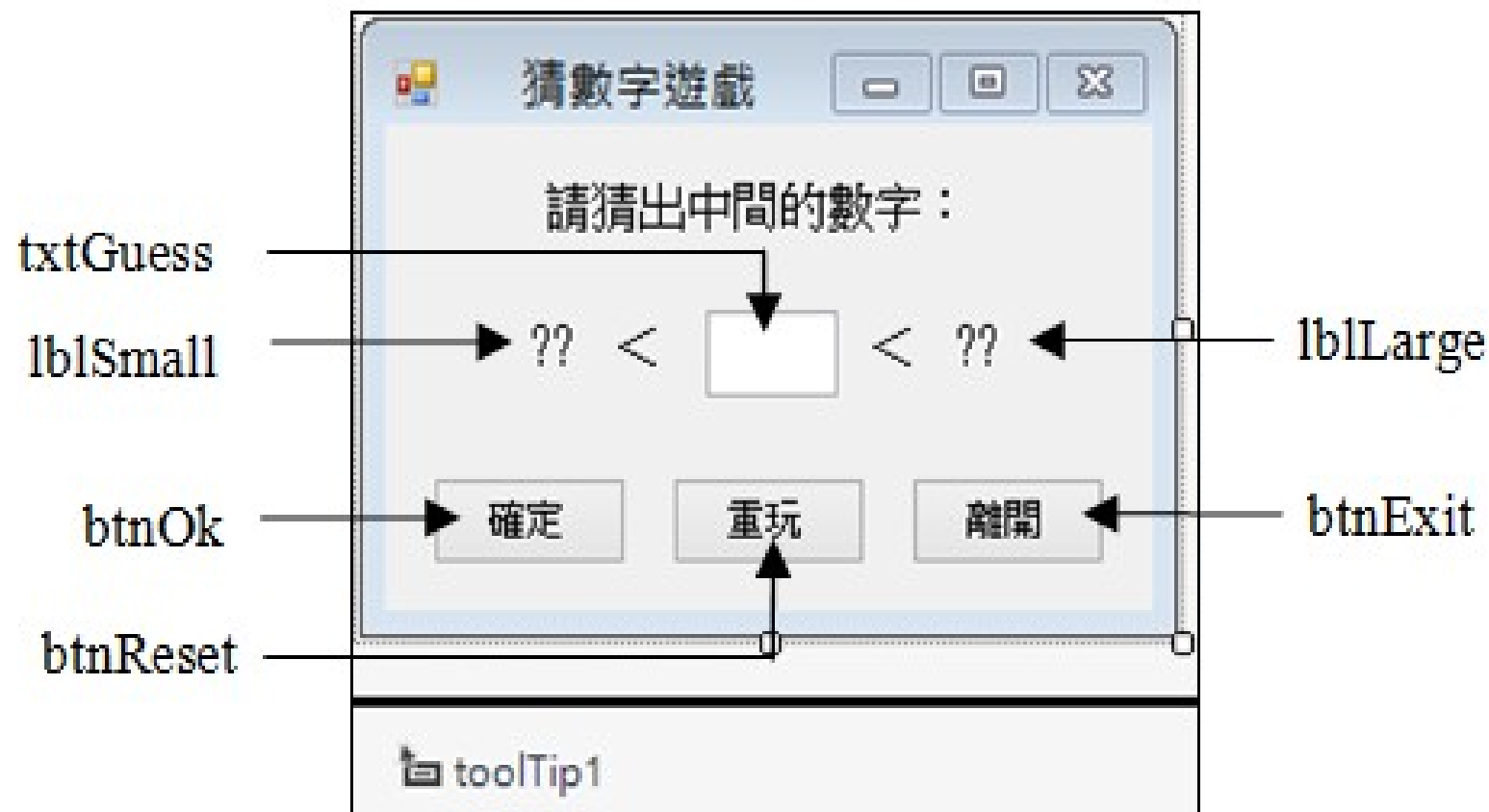
1. There is a TextBox, which shows a tool tip "Please input a 2-digit number!" when the mouse is on this TextBox
2. The generated random number has between 10 and 99. If the player's answer is correct, use MessageBox to show "Congratulations!"
3. If the data of TextBox is not a number, show an error message "Please input a 2-digit number" and a warning message "Please input a number between the minimum number and the maximum number"
4. If the player's input number is greater than the answer, show an info message "Please input a smaller number" and use the input number to replace the content of lblLarge Label
5. If the player's input number is smaller than the answer, show an info message "Please input a greater number" and use the input number to replace the content of lblSmall Label
6. If input number is not between the minimum number and the maximum number, show an info message "Please input a number between the minimum number and the maximum number"
7. The TextBox is cleared and focused when the message box is closed except for the info message "Congratulations!"

- 
8. Press “重玩” button, back to the initial condition and regenerate the answer randomly
 9. Press “離開” button, close the window

Result:



Result



6-8. Debug

1. Start debug

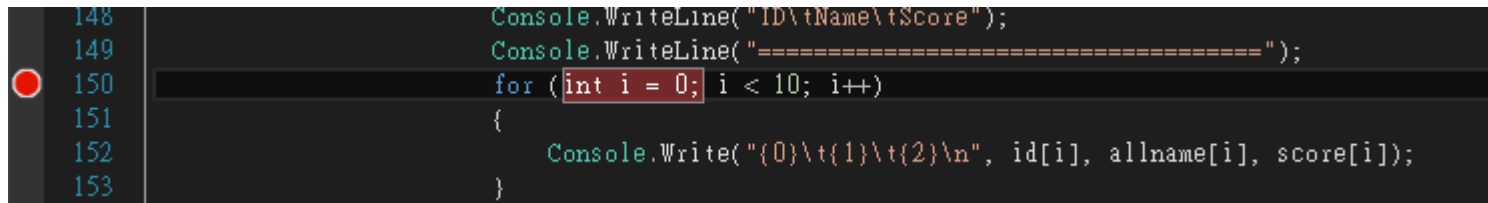
press F5 or Debug(D)/Start debug(S) to compile and run the program,
examine whether every function fits the requirements or not

2. Stop debug

press the close button at the right-top side of window  or
Debug(D)/Stop debug(E) to terminate the program and return to IDE.

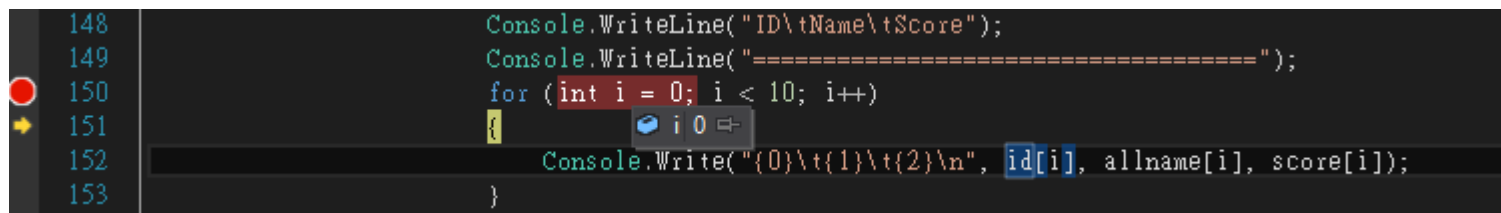
Debug

- 1. press the mouse button and mark the line you feel has problem

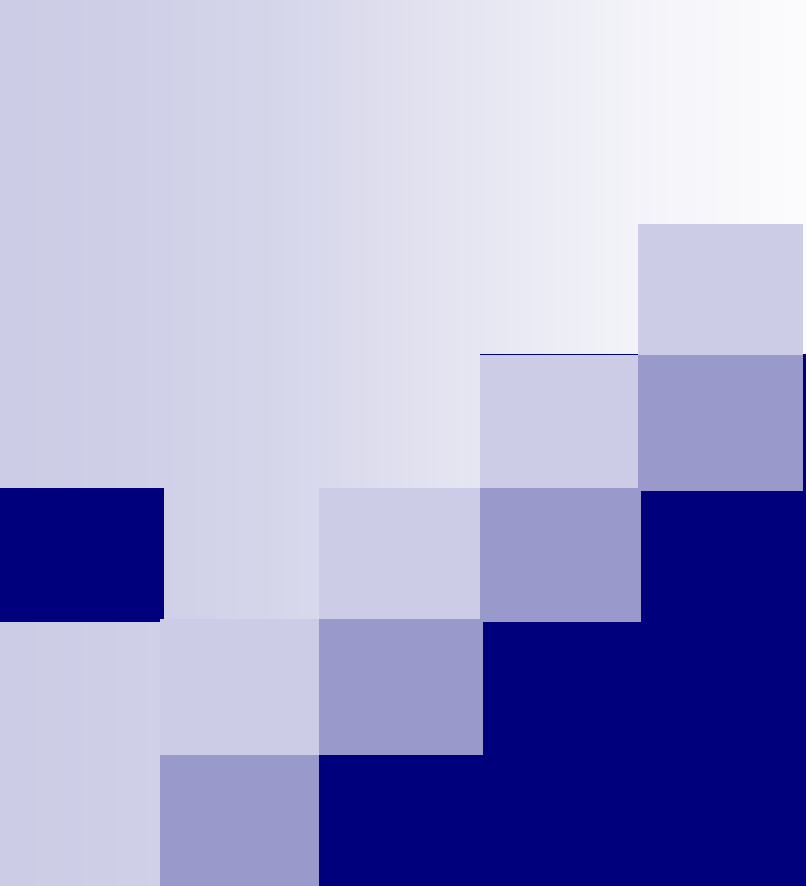


```
148 Console.WriteLine("ID\\tName\\tScore");
149 Console.WriteLine("=====");
150 for (int i = 0; i < 10; i++)
151 {
152     Console.Write("{0}\\t{1}\\t{2}\\n", id[i], allname[i], score[i]);
153 }
```

- 2. press f5 to start the program, it will stop at that line you mark and press f11 you can run the program step by step and see the detail of all variable.



```
148 Console.WriteLine("ID\\tName\\tScore");
149 Console.WriteLine("=====");
150 for (int i = 0; i < 10; i++)
151 {
152     Console.Write("{0}\\t{1}\\t{2}\\n", id[i], allname[i], score[i]);
153 }
```



End

Take a Break