



Chapter 6

Introduction to Windows Form Application

6-1 First Discovered Windows Application

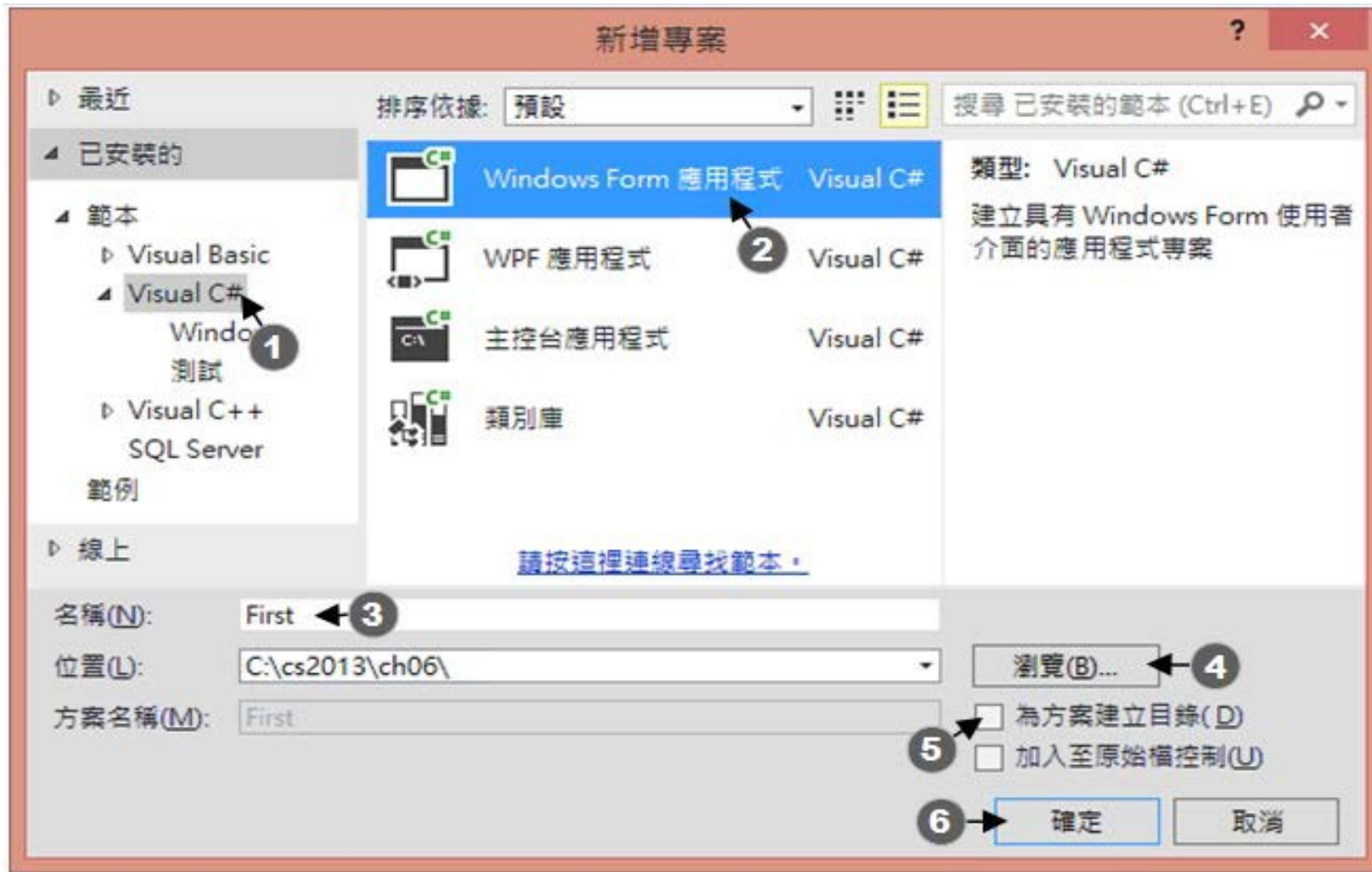
- Former chapters: run under **console mode**
- Windows application runs under **window mode**
- Main difference
 - ① input and output at I/O interface
 - ② WYSIWYG(What You See Is What You Get) user interface maintenance, no need to wait until execution



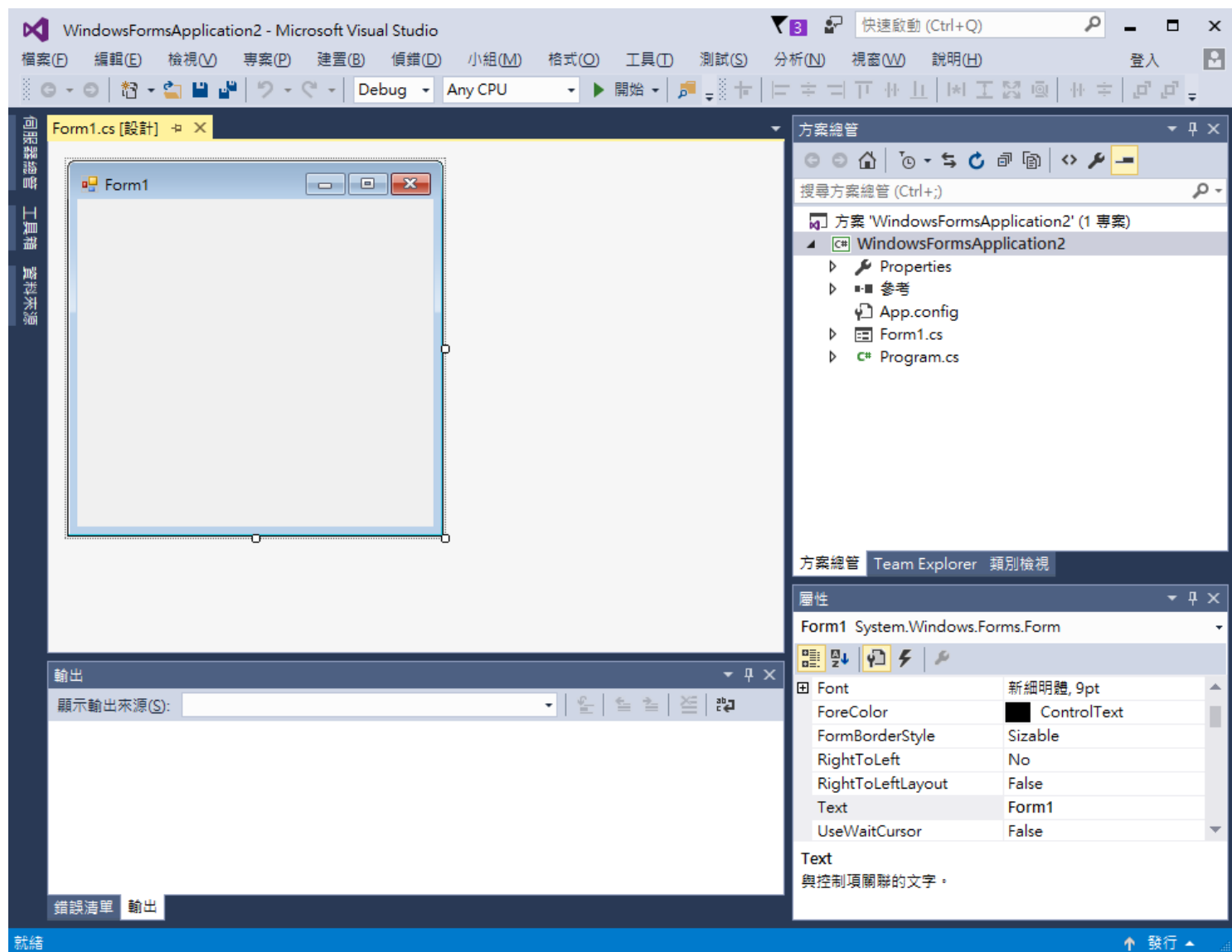
Project

- Use “Form” as a container
- Includes menu, list, text box, command, button, check box, option, etc.
- Use **event-driven method** to connect source code, “form” and “control items”
- **Form** is going to be a **Windows application** as the project is running

1. Open Visual C# IDE



Entering IDE



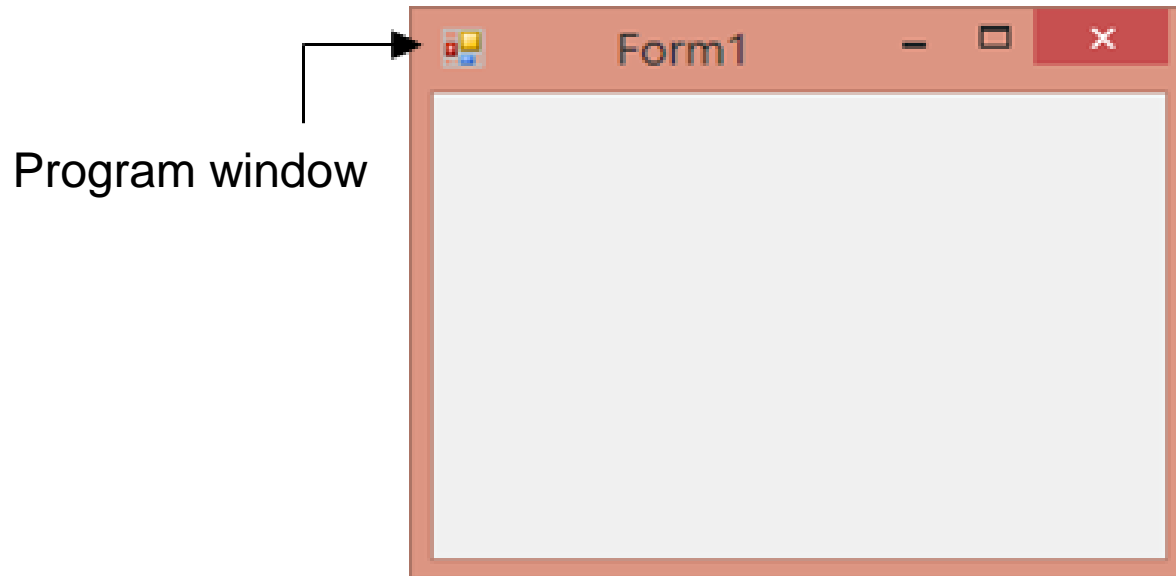
2. Windows Application – Run & Close

1. Run

① menu Debug(D)/Start Debug(S)


② shortcut key F5

Open the program window named Form1

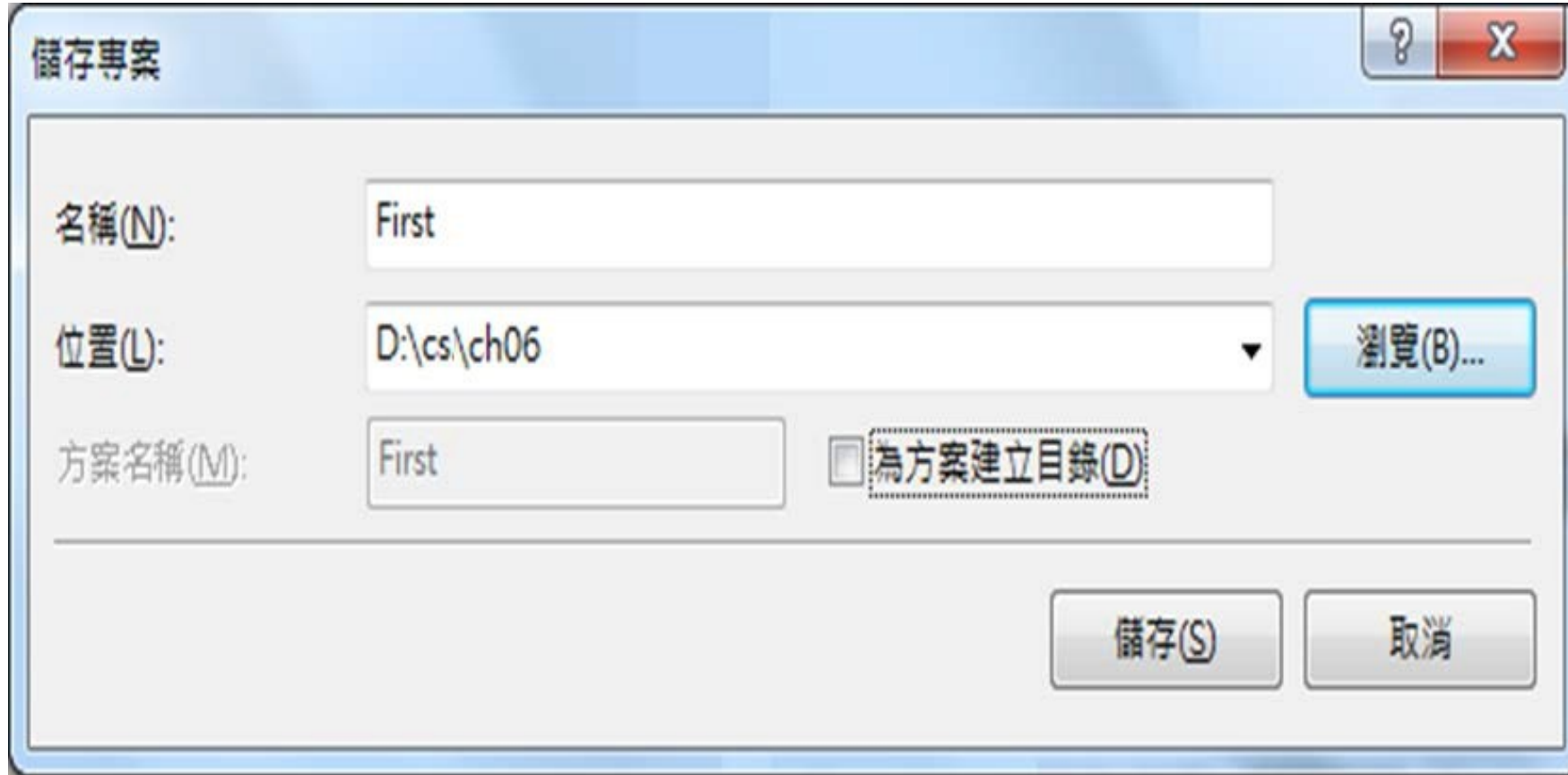


2. Windows Application – Run & Close

2. Close

- ① press the close button at the right-top side of window 
- ② menu Debug(D)/Stop debug(E)
terminate running Windows application and back to the IDE

3. Windows Application – Save and Open



The image shows a '儲存專案' (Save Project) dialog box. It has a title bar with a question mark and a close button. The main area contains three input fields: '名稱(N):' (Name) with the value 'First', '位置(L):' (Location) with the value 'D:\cs\ch06', and '方案名稱(M):' (Project Name) with the value 'First'. To the right of the '方案名稱(M):' field is a checkbox labeled '為方案建立目錄(D)' (Create directory for project). To the right of the '位置(L):' field is a button labeled '瀏覽(B)...' (Browse...). At the bottom right are two buttons: '儲存(S)' (Save) and '取消' (Cancel).

儲存專案

名稱(N): First

位置(L): D:\cs\ch06 瀏覽(B)...

方案名稱(M): First ☐ 為方案建立目錄(D)

儲存(S) 取消

3. Windows Application – Save and Open

1. Save program

① press the “Save all” icon at tool bar 

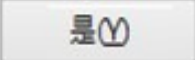


② menu File(F)/Save all(L)

The related file of First project is stored at assigned directory

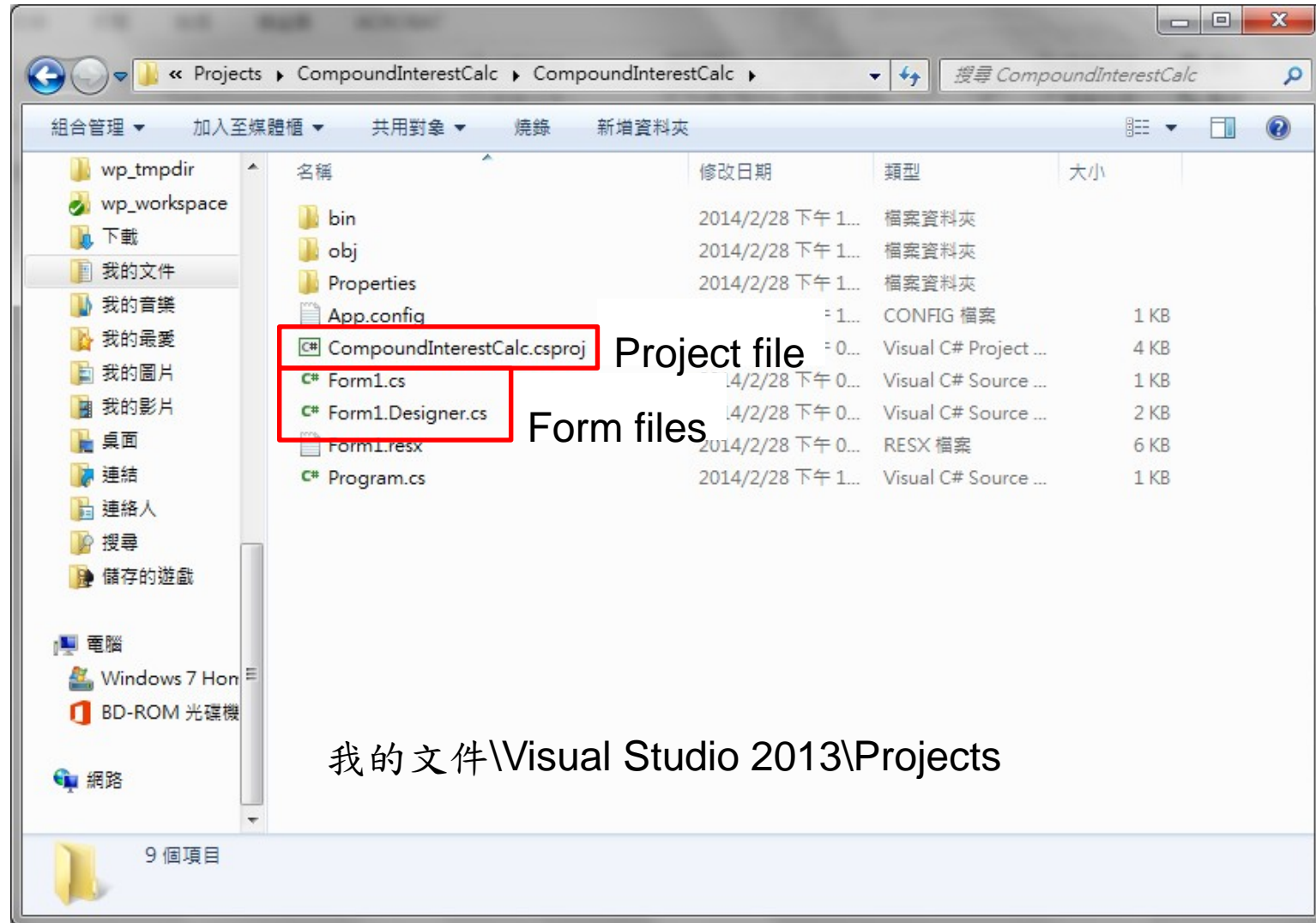
2. Exiting IDE

When execute File(F)/Exit(X) to leave IDE, the IDE will question about saving if the project is modified.

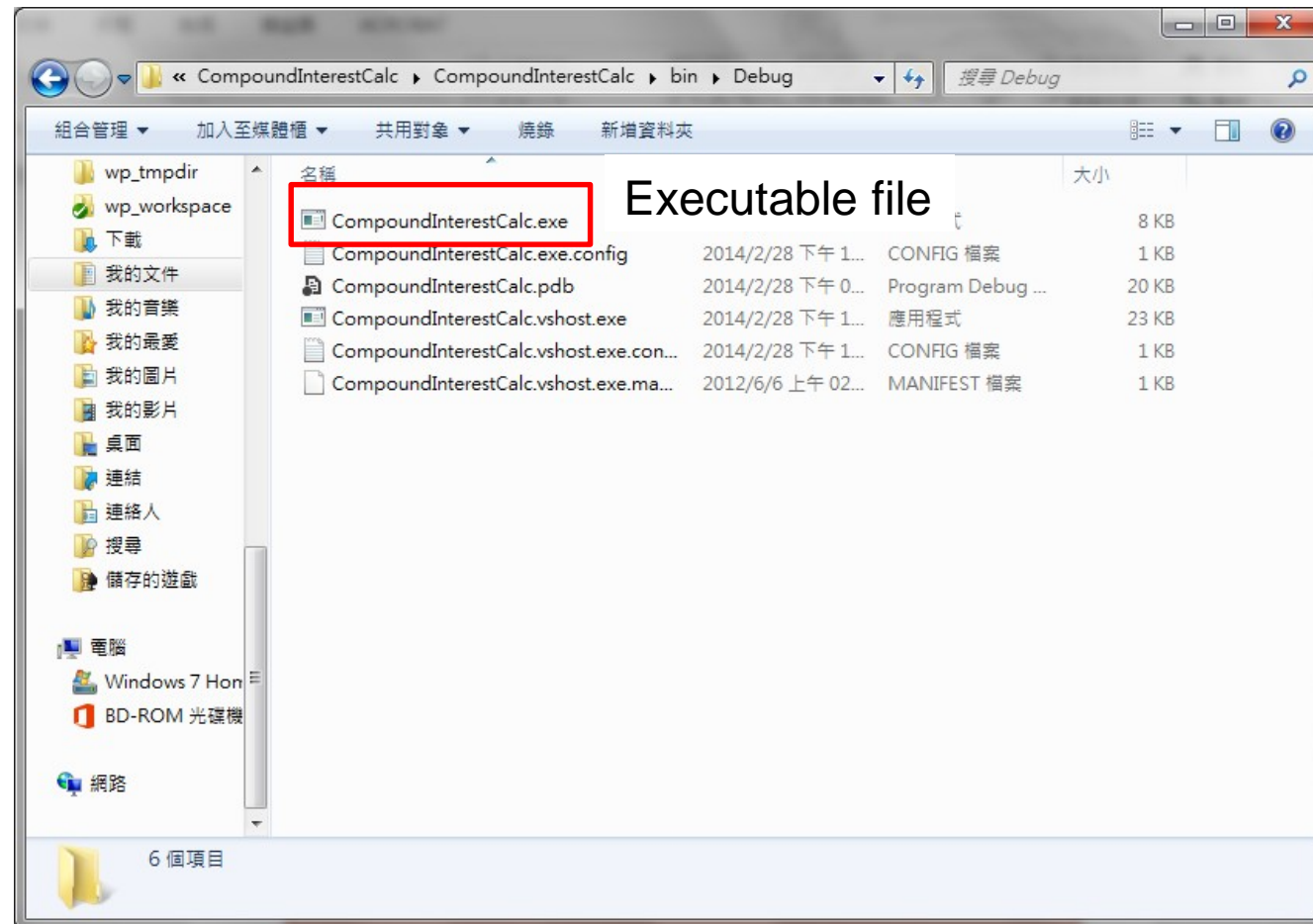


- | | | |
|---|---|------------------------------------|
| ① |  | Replace the original content |
| ② |  | Preserve the original content |
| ③ |  | Return to IDE and continue editing |

3. Open Project Directory

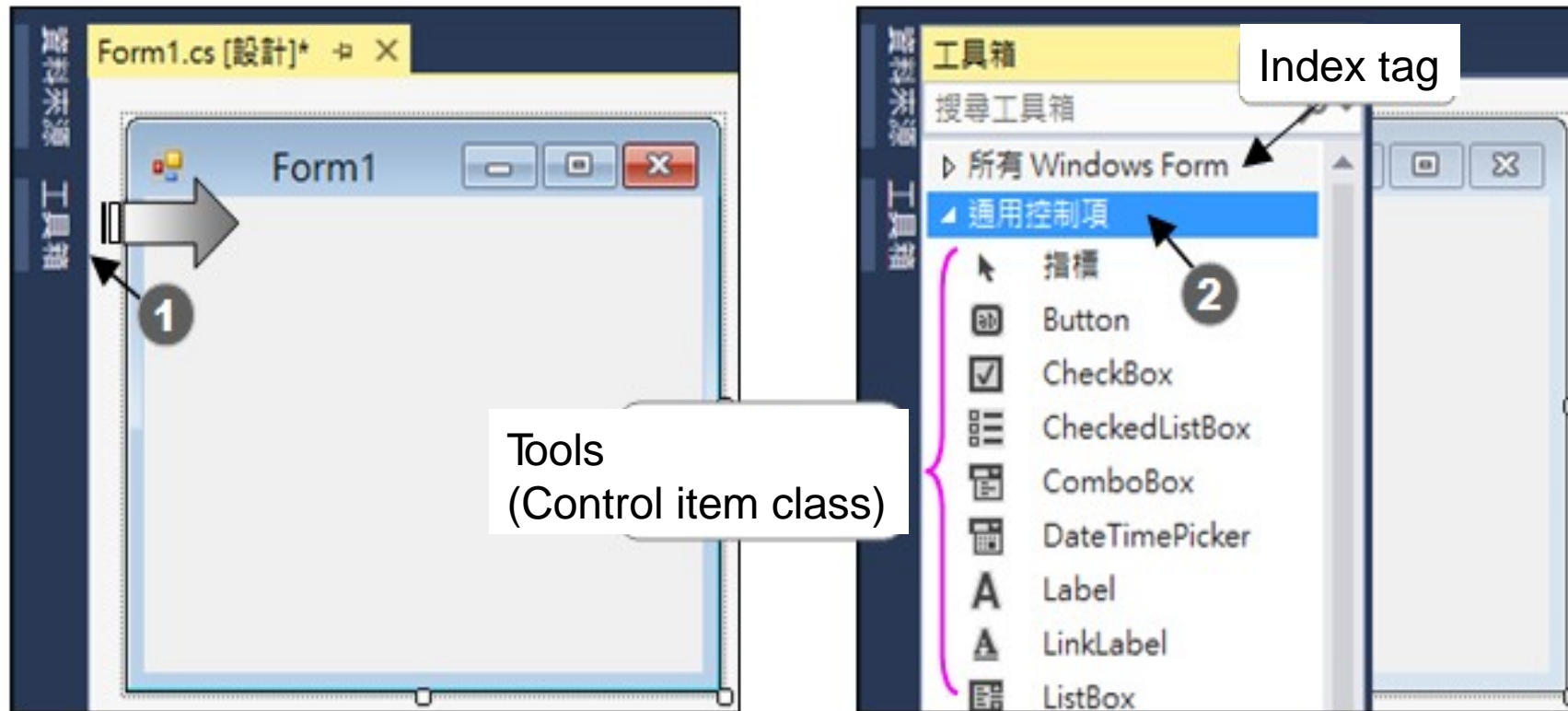


- In bin\Debug directory:
the project is compiled and placed at the executable
Windows application

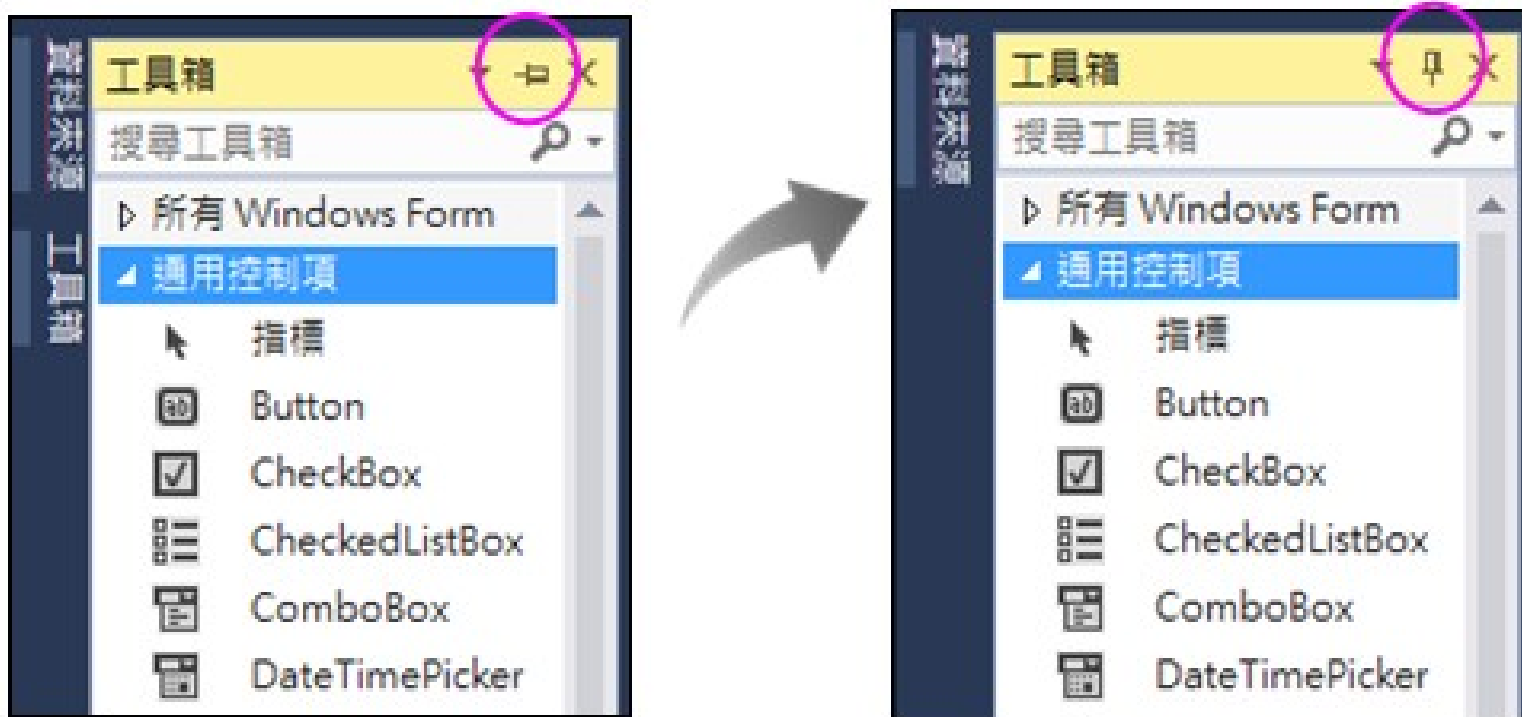


6-2 Development Environment of Windows Application

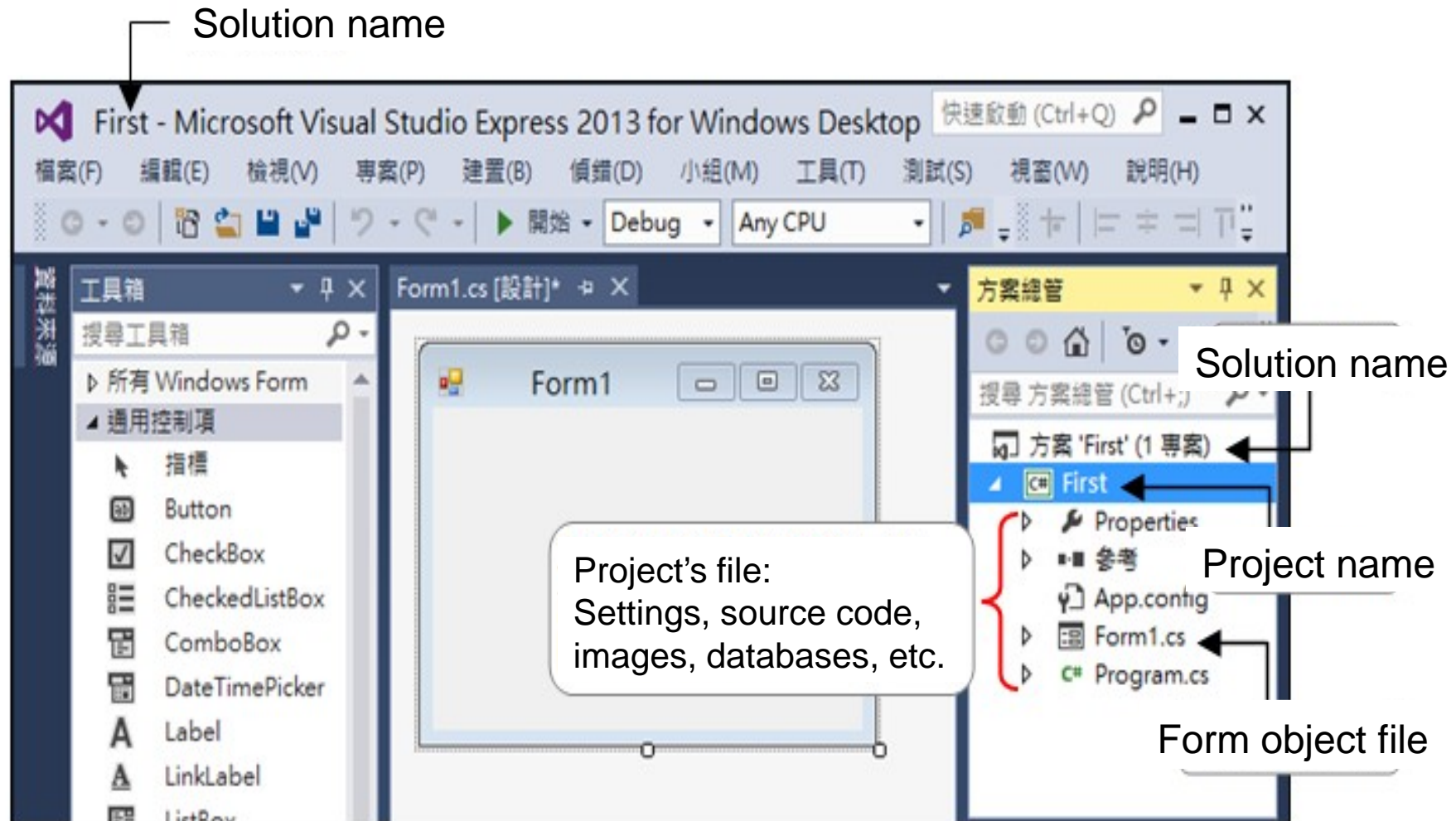
1. pop-up toolbox



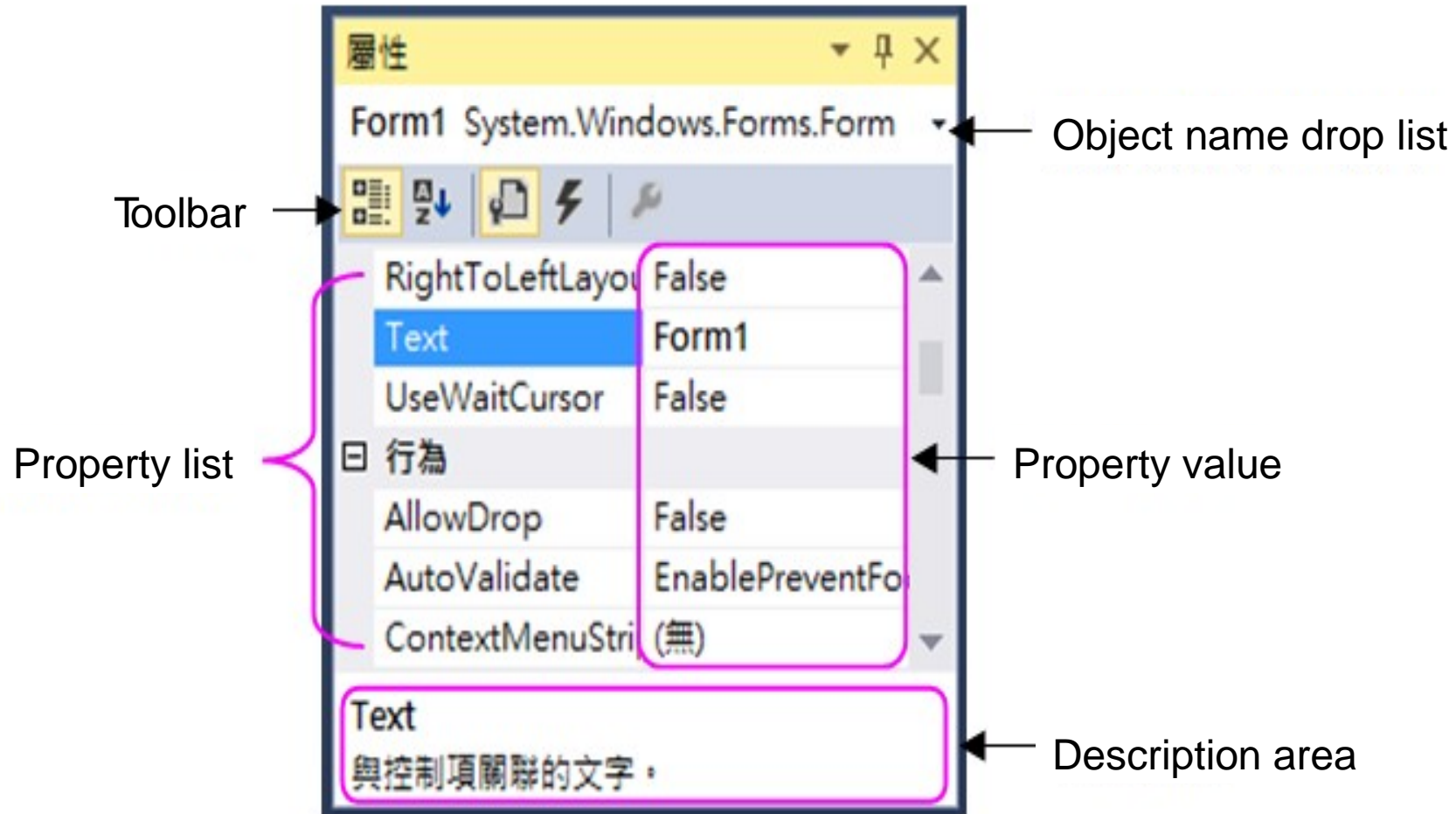
2. Fixed toolbox



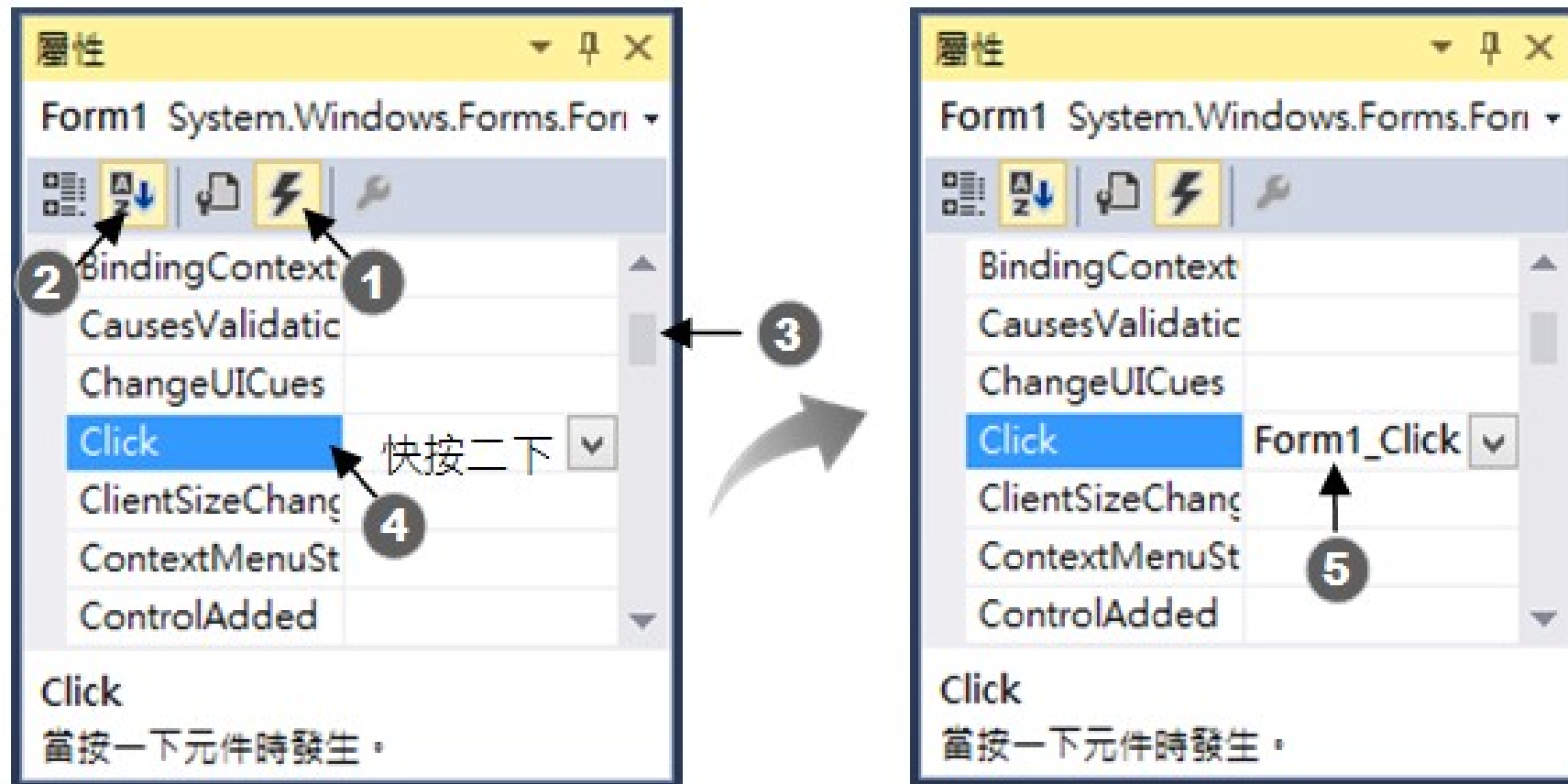
2. Project Manager



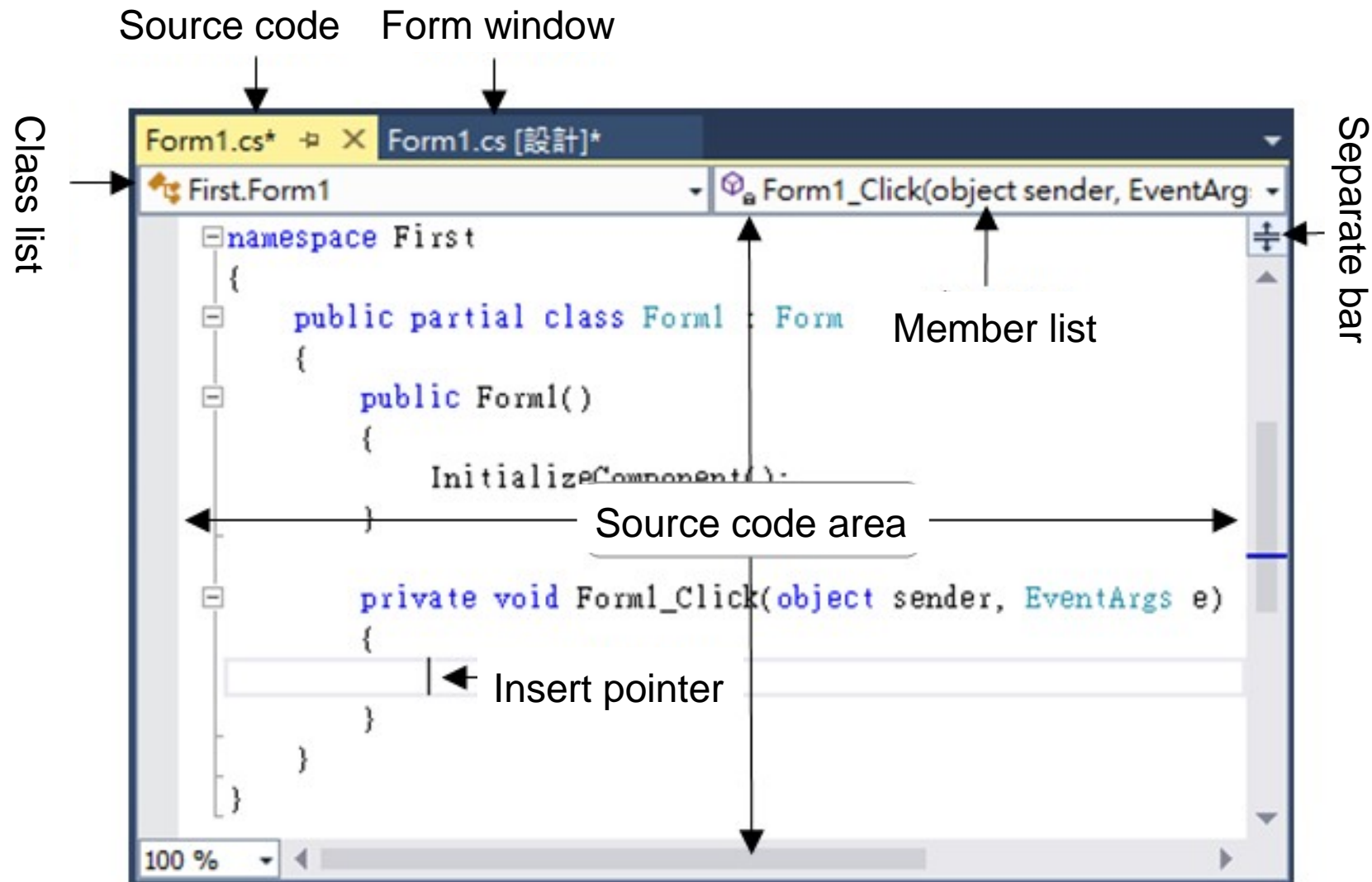
3. Property Window



2. Event Window



4. Source Code Window



6-3 Phases of Windows Application Design

4 Phases of Windows Application Design

- | | | |
|---|---|---------------------------------------|
| 1. Input and output
interface creation | } | Form design phase |
| 2. Property assignment | | |
| 3. Source code editing | } | Program design and
execution phase |
| 4. Project building and
debugging | | |

3 ways to set properties

1. Input type: directly set the value of property
2. Menu type: select a value from all available property values 
3. Dialog type:  show a dialog window to programmer for advanced settings

Example(Compound):

Follow the 4 phases mentioned just now to design a compound-interest-calculation Windows Application. The requirements are:

1. Provide 3 pairs of label and textbox control items for showing messages and data inputation
2. Provide a “Calculate” button control item
3. Provide a label control item for showing the sum of principal and interest, the sum of all interest in 2 lines.
4. When the principal, interest rate, saving years, press “Calculate” button to calculate the sum of principal and interest with the formula. The formula is:

sum of principal and interest = principal x (1 + interest rate)^{years}

function: Math.Pow(2, 3) = 2³

interest = sum of principal and interest – principal

Result:



The diagram illustrates the process of using a compound interest calculator. It shows two states of a window titled "複利計算程式" (Compound Interest Calculator). An arrow points from the initial state on the left to the final state on the right.

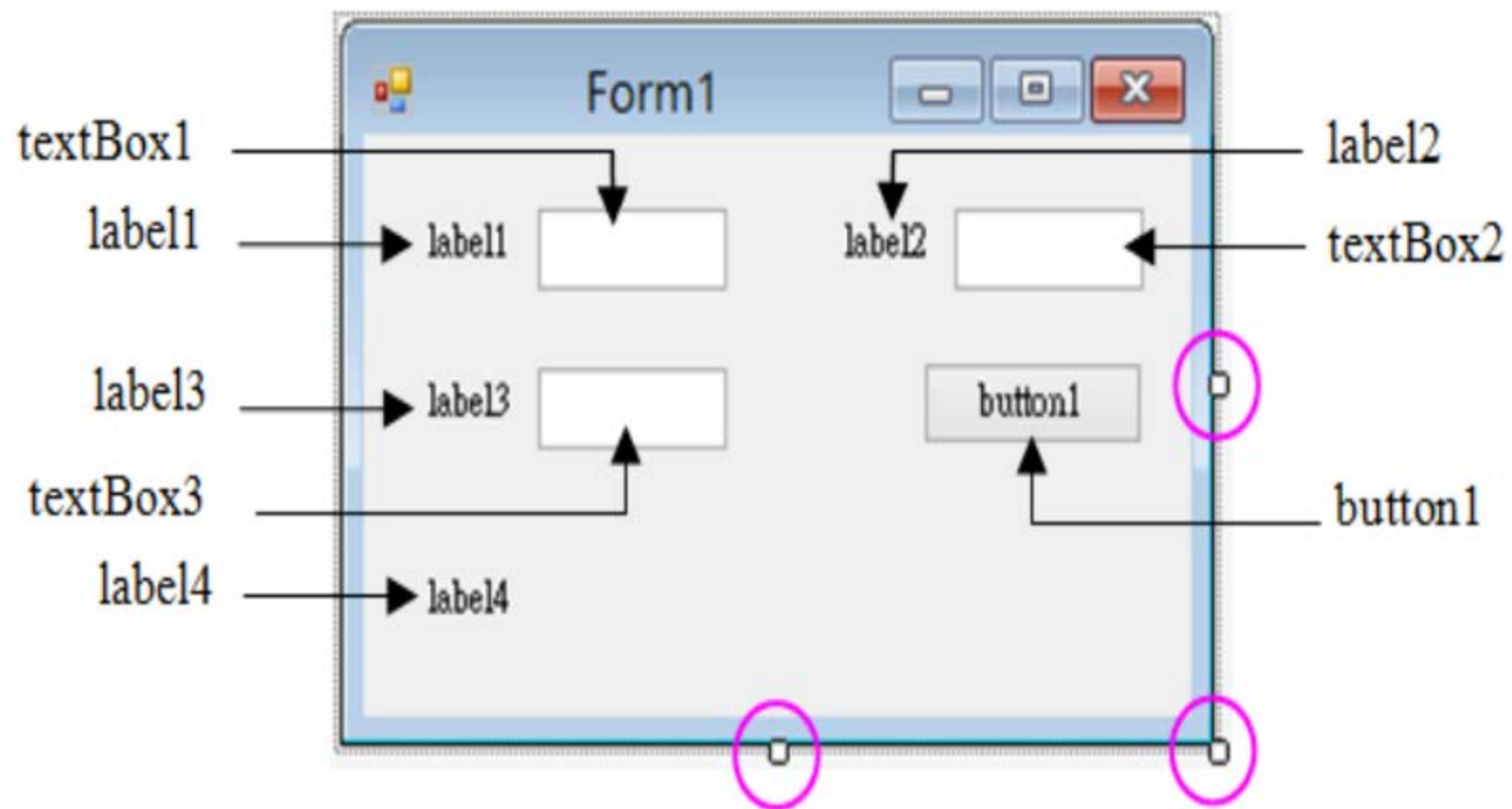
Initial State (Left Window):

- 本金(元):
- 利率(%):
- 存款年數(年):
- 計算:
- Result area:

Final State (Right Window):

- 本金(元):
- 利率(%):
- 存款年數(年):
- 計算:
- Result area:

本利和 = NT\$ 2,675,874.4 元
總利息 = NT\$ 275,874.4 元





Set the Name and Text properties

1. label1: Name reserved, Text = “Principal:”
2. label2: Name reserved, Text = “Years:”
3. label3: Name reserved, Text = “Rate(%):”
4. textBox1: Name = txtCapi, Text = “” (empty string)
5. textBox2: Name = txtYear, Text = “”
6. textBox3: Name = txtRate, Text = “”
7. button1: Name = btnCal, Text = “Calculate”
8. label4: Name reserved, Text reserved



3. Source Code

// FileName : Compound.sln

```
01 using System;
02 using System.Collections.Generic;
03 using System.ComponentModel;
04 using System.Data;
05 using System.Drawing;
06 using System.Linq;
07 using System.Text;
08 using System.Threading.Tasks;
09 using System.Windows.Forms;
10
11 namespace Compound
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19     }
20 }
```

Automatically generated by IDE

```
private void Form1_Load(object sender, EventArgs e)
```

```
{  
    label4.Text = "";  
}
```

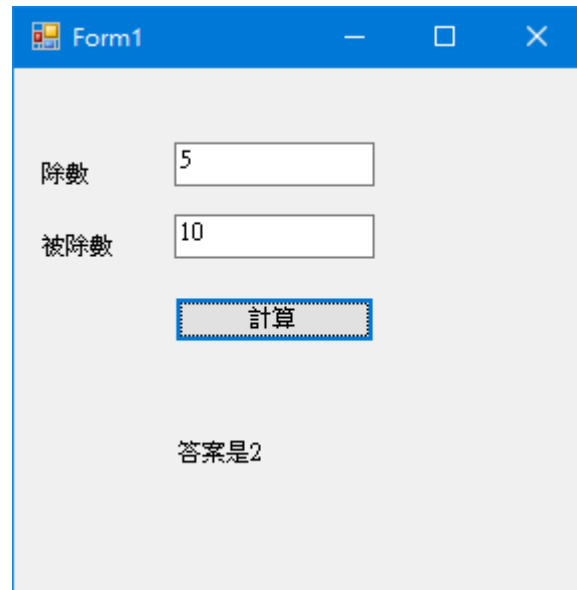
```
private void btnCal_Click(object sender, EventArgs e)
```

```
{  
    double money = double.Parse(txtCapi.Text);    //本金  
    double years = double.Parse(txtYear.Text);    //年期  
    double yrate = double.Parse(txtRate.Text);    //年利率  
    double total;    //本利和  
    total = money * Math.Pow((1 + yrate / 100), years);  
    label4.Text = "本利和 = NT$ " + total.ToString("#,#.0") + " 元";  
    label4.Text += "\n總利息 = NT$ " + (total - money).ToString("#,#.0") + " 元";  
}
```

Practice 6.1: Calculator

- Take a simple practice.

Use the Button event to calculate and give the answer.



Form1

除數 5


被除數 10

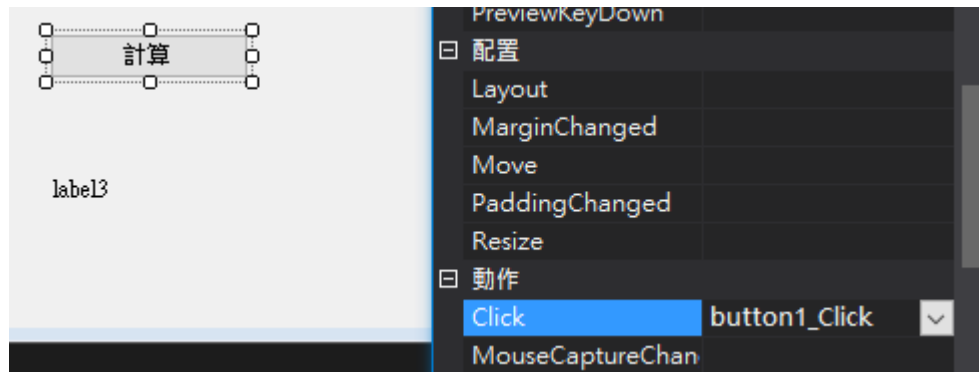
計算

答案是2

Practice 6.1: Calculator

- Tips:

- If you want to give an element an event, click the element and find the  icon in the attribute window.
- Use the click button event as an example, set the name as button1_Click in the click event. Then you can see the function in the code script.



```
private void button1_Click(object sender, EventArgs e)
{
    |
}
```

6-4 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:

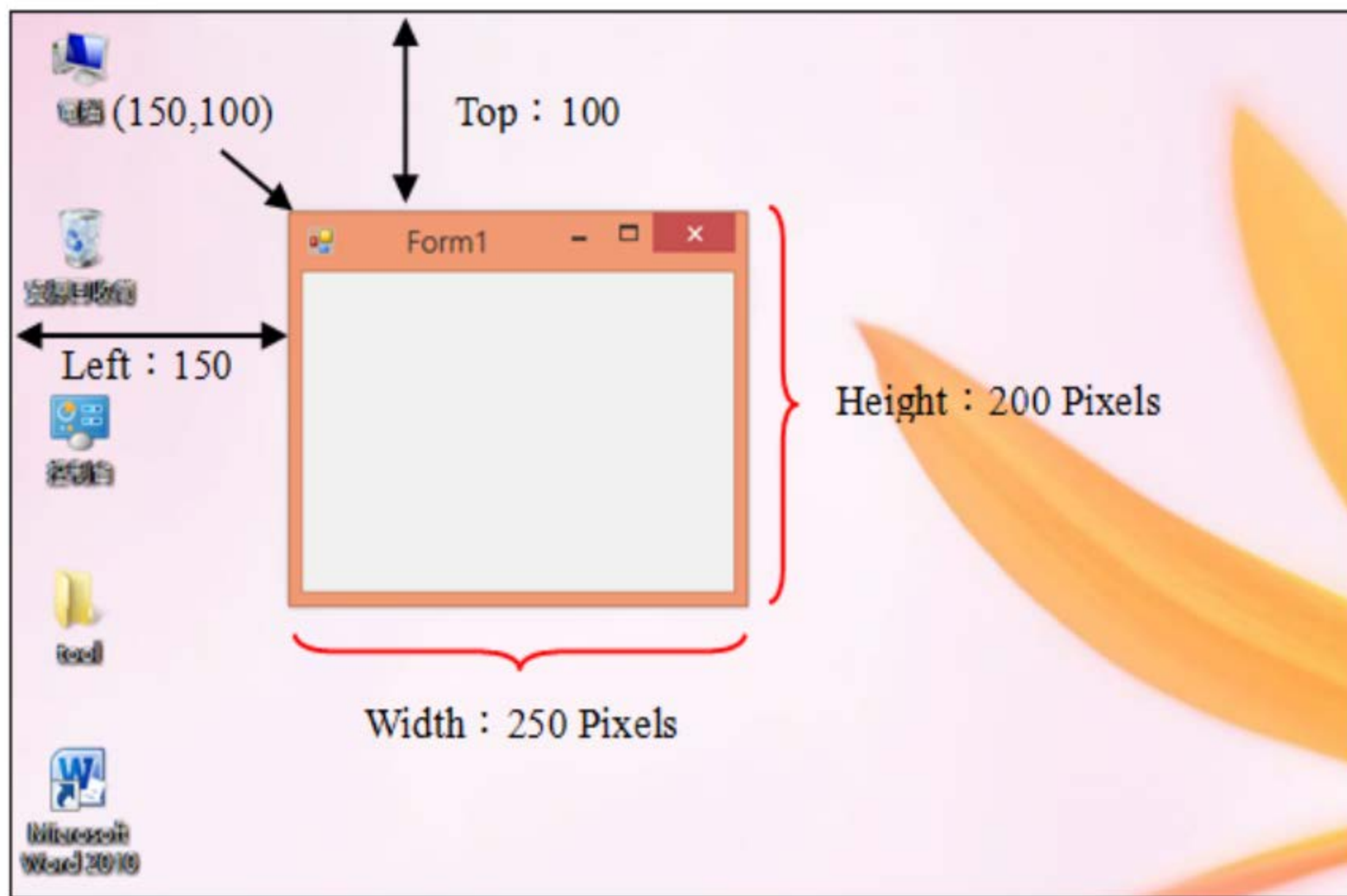


```
button1.TextAlign = ContentAlignment.BottomRight;
```

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 :

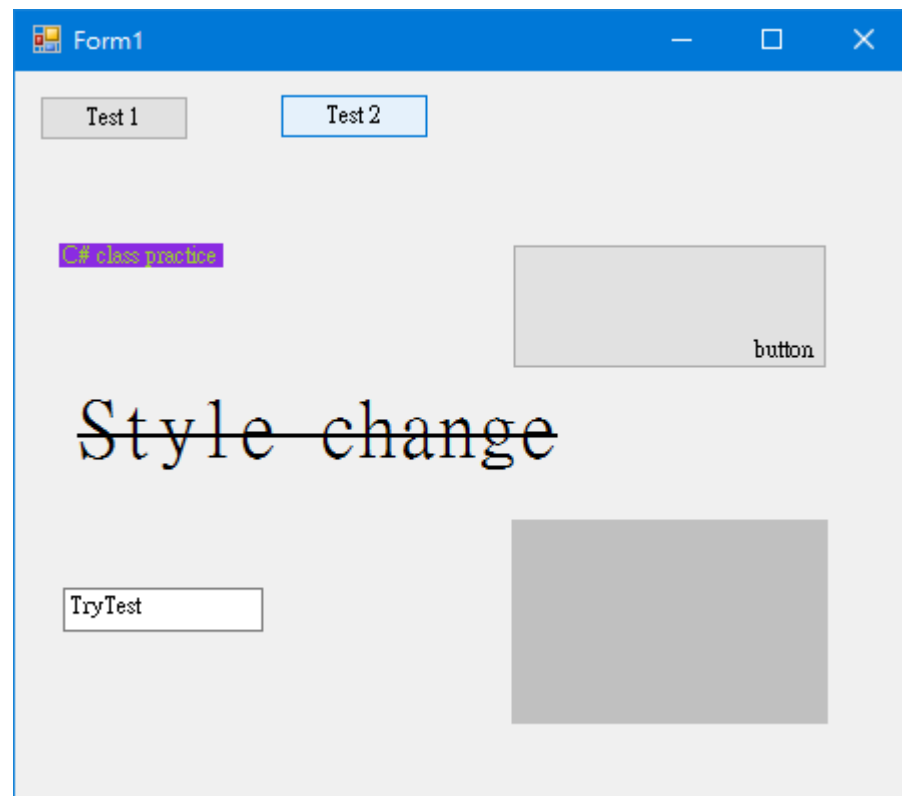
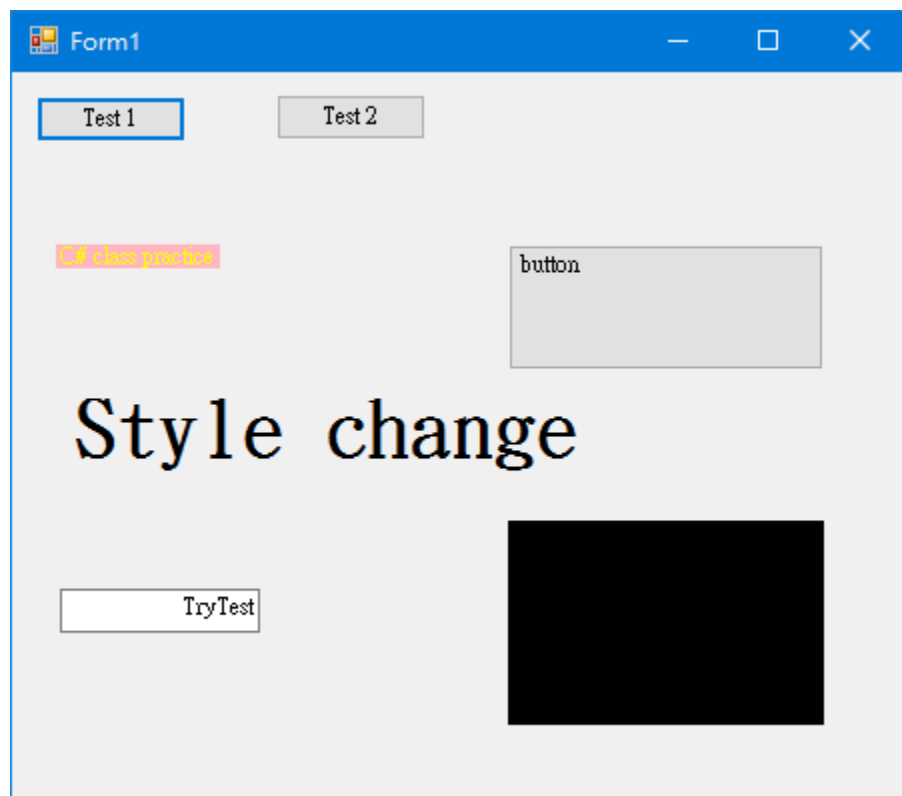
label1 – BackColor: BlueViolet ; ForeColor: YellowGreen

label2 – Font: 細明體, 30, Strikeout

button3 – TextAlign: BottomRight

textBox1 – TextAlign : Left

pictureBox1 – BackColor: Silver



Practice 6.2: WindowStyleChange

- Design a style previewer, 2 styles
- Style1:
 - label1 – BackColor: Yellow ; Font: 微軟正黑體, 18, Bold
 - label2 – Font: 微軟正黑體, 14, Bold
 - textBox1 – BackColor: White
 - button3 – TextAlign: BottomRight
 - pictureBox1 – BackColor: White

Practice 6.2: WindowStyleChange

- **Style2:**

label1 – BackColor: Transparent ; Font: 新細明體, 18, Underline

label2 – Font: 新細明體, 14, Italic

textBox1 – BackColor: Brown

button3 – TextAlign: MiddleRight

pictureBox1 – BackColor: Blue

Result

Form1

style1 style2

title text

subtitle text

button text

textbox text

A screenshot of a Windows form titled 'Form1'. The form has a light gray background. At the top, there are two buttons labeled 'style1' and 'style2'. Below them, the text 'title text' is displayed in a bold, black font, and 'subtitle text' is displayed in a regular, black font. To the right of the text, there is a gray rectangular area labeled 'button text'. At the bottom left, there is a white rectangular area labeled 'textbox text'. A large white rectangle is positioned at the bottom right of the form.

Form1

style1 style2

title text

subtitle text

button text

textbox text

A screenshot of a Windows form titled 'Form1'. The form has a light gray background. At the top, there are two buttons labeled 'style1' and 'style2'. Below them, the text 'title text' is displayed in a regular, black font, and 'subtitle text' is displayed in an italicized, black font. To the right of the text, there is a gray rectangular area labeled 'button text'. At the bottom left, there is a red rectangular area labeled 'textbox text'. A large blue rectangle is positioned at the bottom right of the form.

6-5 Form

Appearance Styles

Property	Description
<code>BackColor</code>	Set the background color of the form
<code>BackgroundImage</code>	Set the background image's path of the form
<code>BackgroundImageLayout</code>	None: the image has left-top alignment Tile: the image arranged side by side (default) Center: the image is put at center Stretch: the image is expanded and fit to the border Zoom: zoom the image
<code>Cursor</code>	The shown cursor style when the mouse pointer is moved to the control item, 28 styles

ForeColor	The foreground color is set
Text	The text of the title bar
FormBorderStyle	<p>None: no border and no title bar</p> <p>FixedSingle: single line border</p> <p>Fixed3D: 3-D border line</p> <p>FixedDialog: dialog, thick border and the title bar has only close button</p> <p>Resizable: normal border line and the size can be changed (default)</p> <p>FixedToolWindow: unresizable tool window border, with only close button</p> <p>ResizableToolWindow: adjustable tool window border</p>

Layout Styles

Properties	Description
AutoScroll	True: automatically show the scroll bars when the control item is bigger than working area. False: default
AutoSize	True: automatically resize according to the size of control item False: no auto resize
AutoSizeMode	GrowAndShrink: the control item resizes to fit the content, but cannot be adjust manually GrowOnly: enlarge if needed to fit the content, but does not shrink to the size which is smaller then the Size property
Location	The coordinate of the control item's left-top corner relates to the form's left-top corner

Size	The size of control item (pixel)
StartPosition	Manual: defined by Location property CenterScreen: the center of the screen WindowsDefaultLocation: positioned at the Windows default location and has the bounds determined by Windows default (default) WindowsDefaultBounds: centered within the bounds of its parent form CenterParent: positioned in the bound of parent form
WindowState	Normal: default size (default) Minimized: minimized window Maximized: maximized window

Window Styles

Properties	Description
ControlBox	True: show control box (default) False: hidden control box
HelpButton	True: show the help button False: hidden the help button (default)
Icon	Set the icon when the window is minimized
MaximizeBox	True: show the maximize button (default) False: hidden the maximize button
MinimizeBox	True: show the minimize button (default) False: hidden the minimize button
ShowIcon	True: show the icon on the title bar (default) False: hidden the icon on the title bar
ShowInTaskbar	True: show the window on the Windows Taskbar (default) False: hidden the window on the Windows Taskbar
TopMost	True: show the form on the most top layer False: show the form normally (default)

Form Methods

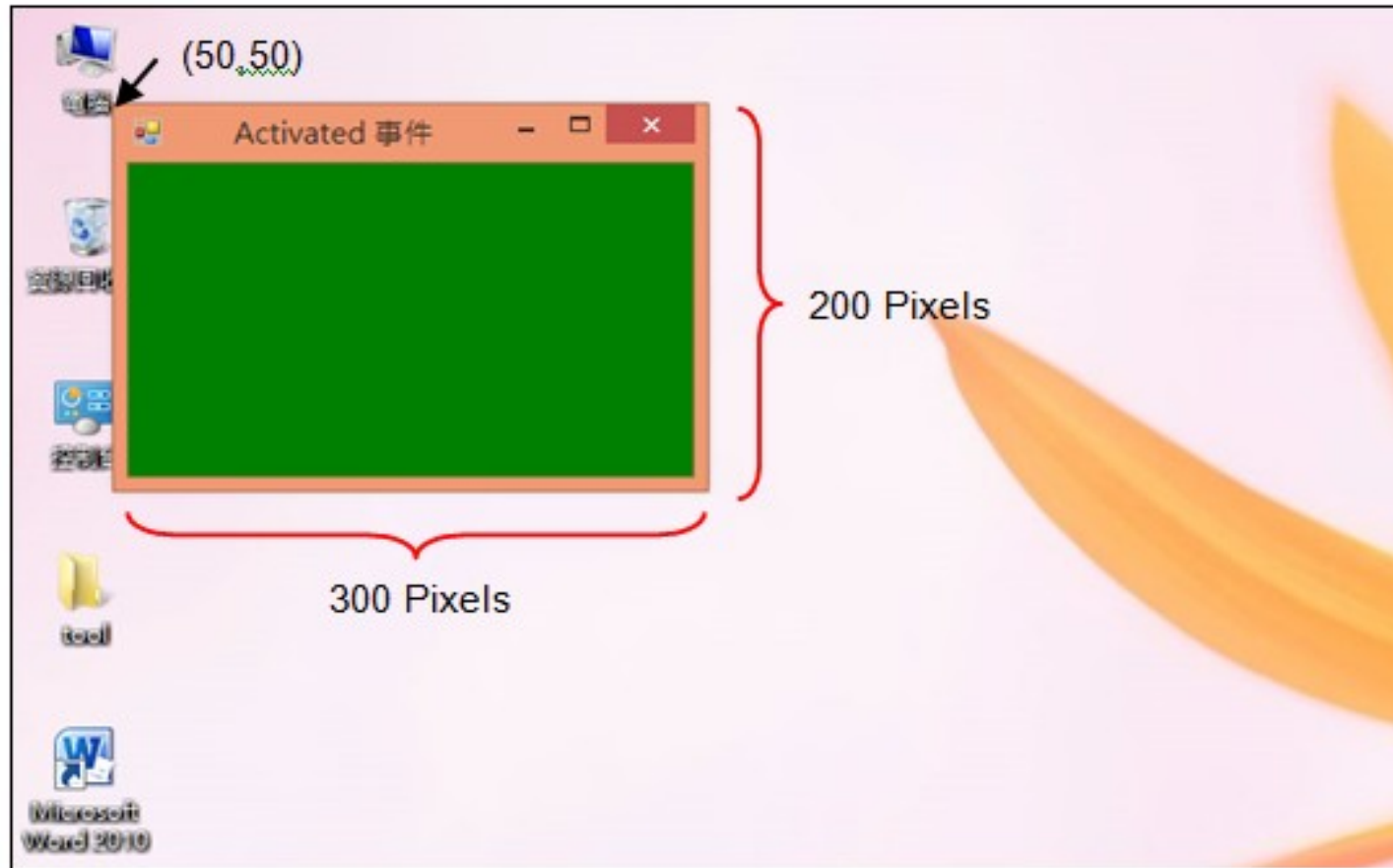
Event	Description
Load	Triggered when the form is first time loaded
Activated	Triggered when the form is focused. The form also triggers this event when the form is loaded first time
Paint	The Canvas triggers this event when filling color or images
Click	Triggered when the mouse clicks on the blank area of the form
DoubleClick	Triggered when the mouse clicks twice
Resize	Triggered when the form or control item has been resized

Example(MyForm):

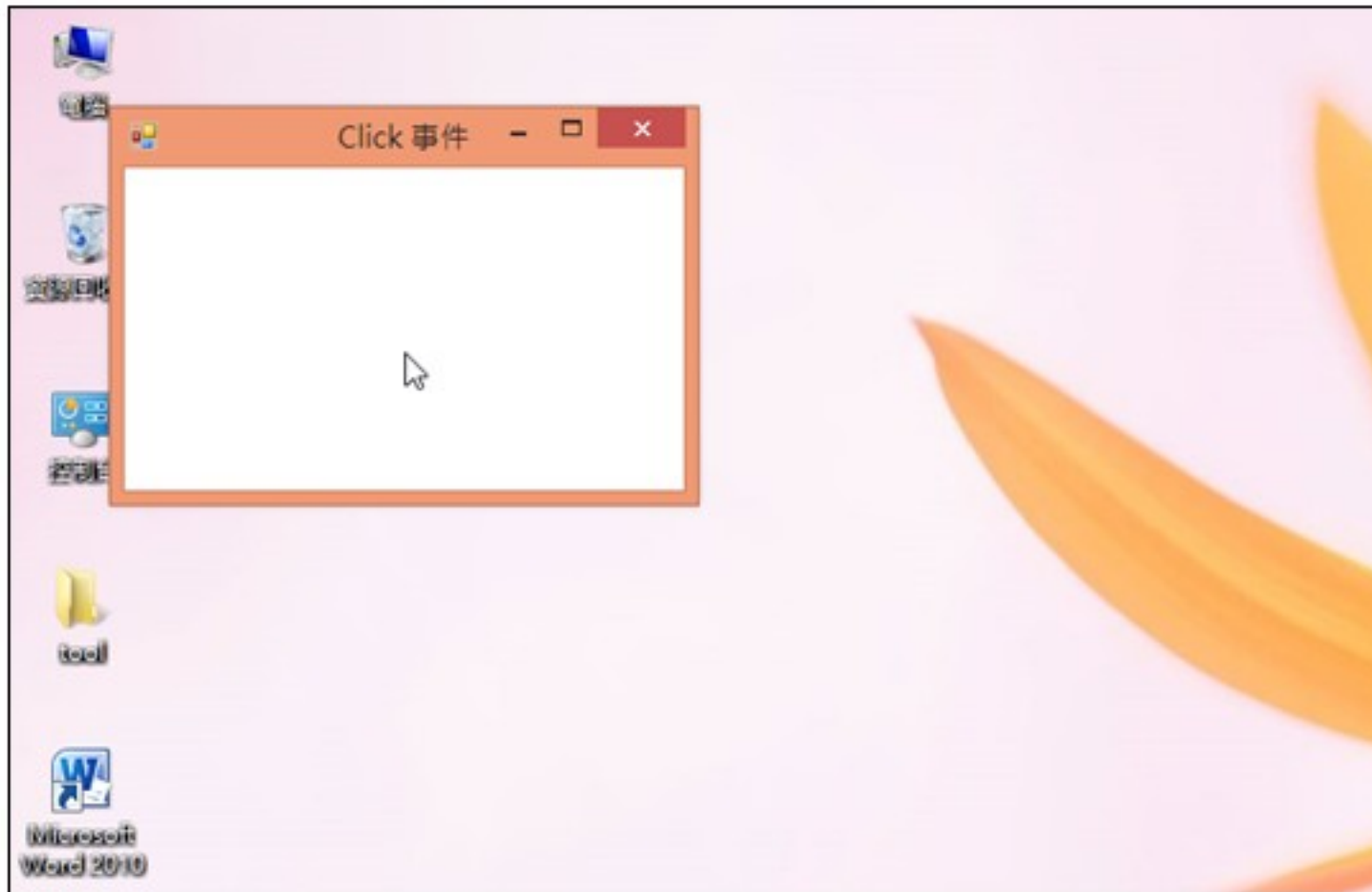
Use the event to adjust properties of the form. Requirements:

1. When the program starts, the form occurs at (50, 50) from the left-top corner, the size of the form is 300*200 pixels. The background color is green, and the title bar shows “Activated 事件”
2. The title bar shows “Click 事件” when click on the working area of the form. Also the background color changes to white
3. Click on the desktop or other windows to make the current form be an inactive window
4. Click on the title bar of the inactive window to trigger the Activate event
5. Click on the working area of the form to trigger the Click event

Result:



Now the title of the form is “Activated 事件”, and the background color of work area is “green”



Click on the work area of the form, the title text is changed to “Click 事件” and the background color is also changed to white

Source Code

```
private void Form1_Activated(object sender, EventArgs e)
{
    this.Text = "Activated Event";
    this.Location = new Point(50, 50);
    this.Size = new Size(300, 200);
    this.BackColor = Color.Green;
}
private void Form1_Click(object sender, EventArgs e)
{
    this.Text = "Click Event";
    this.BackColor = Color.White;
}
```



Exercise:

When the form resized (trigger Resize event), its background color needs to set to be yellow

6-6 Label, Button, TextBox

Property	Description
AutoSize (automatically resize)	True: the control item resizes automatically (default) False: manually assign the size
BorderStyle (border's style)	None: no border (default) FixedSingle: single line border Fixed3D: 3-D border line
Dock (attach the border)	Top: attach to the top border of the container(form) Bottom: attach to the bottom border of the container(form) Left: attach to the left border of the container(form) Right: attach to the right border of the container(form) Fill: fill the all space of the container(form) None: no attachment (default)

ImageAlign (image alignment)	<p>Used when the Image property of the control item is set and the image size is smaller than the size of the control item. 9 property values:</p> <p>TopLeft, TopCenter, TopRight, MiddleLeft, MiddleCenter, MiddleRight, BottomLeft, BottomCenter, BottomRight</p>
Locked (lock)	<p>During form design phase: True: the position of the control item is locked False: the control item can be moved (default)</p>
TextAlign (text alignment)	<p>TopLeft (default), TopCenter, TopRight, MiddleLeft, MiddleCenter, MiddleRight, BottomLeft, BottomCenter, BottomRight</p>



Ex1: how to show the image at the top left side of the control item. Usage:

```
label1.ImageAlign = ContentAlignment.TopLeft;
```

Ex2: how to show the text at the bottom right side of the control item. Usage:

```
label1.TextAlign = ContentAlignment.BottomRight;
```

Ex3: how to use Dock property to fill all space of the form with label 1 control item. Usage:

```
label1.AutoSize = false;  
label1.Dock = DockStyle.Fill;
```

Button Control Item

Property	Description
Enabled (available)	True: can be pressed (default) False: disabled
Visible (revelation)	True: show the button (default) False: hide button
TabIndex (priority of tab)	Set the priority when the user press tab key to switch the focused control item
TabStop (tab parking)	True: can be focused by tab key (default) False: cannot be focused by tab key

TextBox Control Item

Property	Description
Text	The most important property of TextBox. This property stores string data.
MaxLength	Set the maximum string length of TextBox. 0 is unlimited, default value is 32767
PasswordChar	Use other character to replace the original ones
ReadOnly	True: the string can only be selected or scrolled False: modification allowed (default)
Multiline	True: multi-line display allowed False: multi-line display not allowed (default)

WordWrap	True: auto new line allowed (default) False: auto new line not allowed
ScrollBars	None: no scroll bars (default) Horizontal: show horizontal scroll bar Vertical: show vertical scroll bar Both: show horizontal and vertical scroll bars
CharacterCasing	Normal: show alphabets normally (default) Upper: show all alphabets in upper case Lower: show all alphabets in lower case
Lines	Use string array to store the TextBox's content when the Multiline property is set True. Separate the strings by new line

1. Make use of Multiline 、 WordWrap and ScrollBars properties

① Multiline = False

the characters out of the TextBox's length are hidden

② Multiline = True

the TextBox's height can be adjusted

③ Multiline = True and WordWrap = True

the characters out of the TextBox's will be moved to new line

④ Multiline = True and WordWrap = False

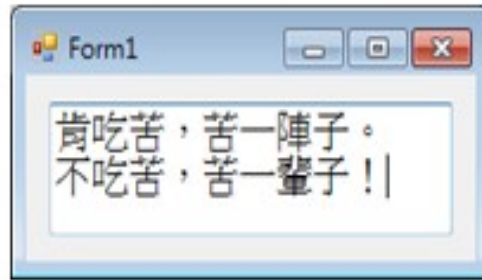
ScrollBars is also been set so that the horizontal and vertical bars can be used

2. Make textBox1 show text in upper case, usage: textBox1.CharacterCasing = CharacterCasing.Upper;

Example(Lines) 1:

Assign strAry1 string array to textBox control item in multi lines. Remember to set the Multiline property of textBox1 to True.

```
string[] strAry1 = new string[] {"肯吃苦，苦一陣子。", "不吃苦，苦一輩子!"};  
textBox1.Lines = strAry1;
```



Example(Lines) 2:

From the previous example, show the text of textBox1 in lable1 and label2 separately.

```
label1.Text = textBox1.Lines[0];  
label2.Text = textBox1.Lines[1];
```

Result:

label1 shows “肯吃苦，苦一陣子。”

label2 shows “不吃苦，苦一輩子！”

Example(Lines) 3:

From the previous example, put the text of textBox1 into the string array strAry2

```
string[] strAry2 = new string[2] ;  
for (int n=0; n <= strAry2.Length-1; n++)  
    strAry2[n] = textBox1.Lines[n];
```

Example(Lines) 4:

From the previous example, use for loop to show all elements of strAry2 on the label3

```
label3.Text = "";  
for (int k = 0; k <= strAry2.Length - 1; k++)  
    label3.Text += (strAry2[k] + '\n');
```

TextBox Methods

1. Clear()

set the showing content of the component to blank. Ex: clear the showing text of textBox1, usage:

```
textBox1.Clear( );
```

or

```
textBox1.Text = ""
```

2. Focus()

programs can be switched by Tab key, use Focus() method to switch to the target control item directly

```
textBox1.Focus( );
```



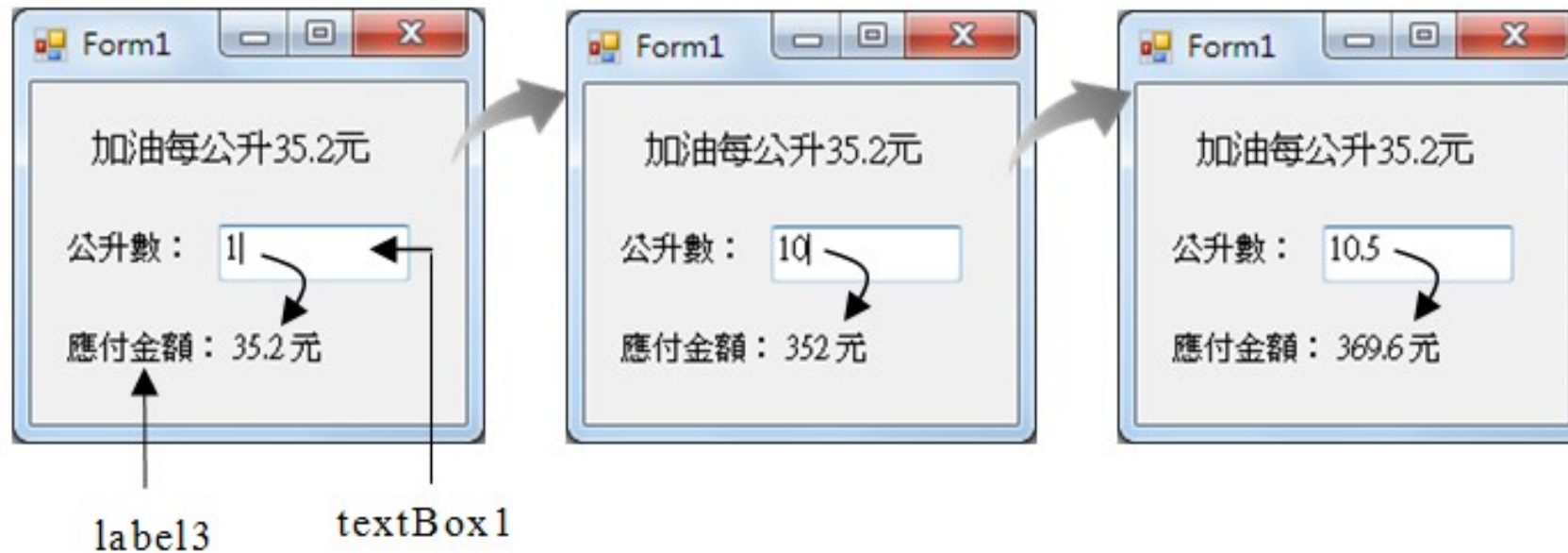
TextBox TextChanged Event

- **Default event of TextBox**
- **The event is triggered when the value of Text property is changed**

Example(gasoline):

Write the code of the textBox1's TextChanged event handler. The label3 shows the payment when the textBox1 gets the liter number of gasoline.

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label3.Text = "應付金額： " + float.Parse(textBox1.Text) * 35.2 + " 元";
}
```



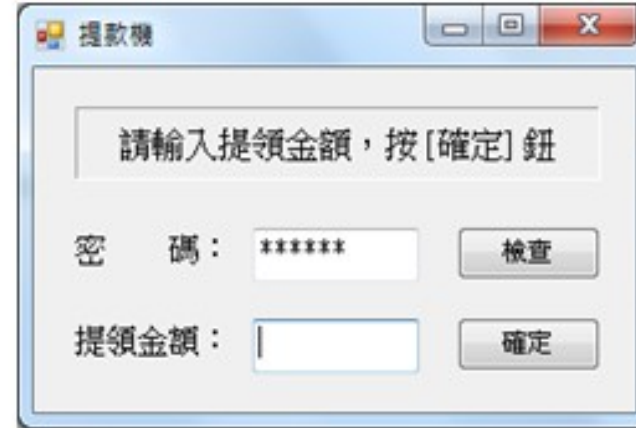
Example(Counter):

Design an ATM, the user can withdraw money when the password is right. Requirements:

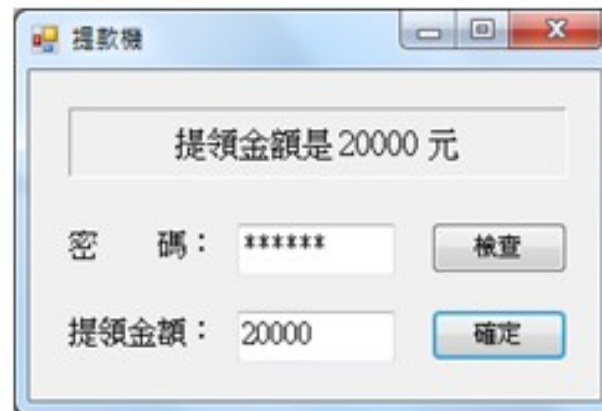
1. The hint text is shown as the figure. The TextBox txtPW gets the password and shows password in '*'. The maximum number of characters is 8
2. The right password is "123456". Check the password after pressing "檢查" button. The TextBox txtMoney and "確定" button is disabled before the "檢查" button has been pressed



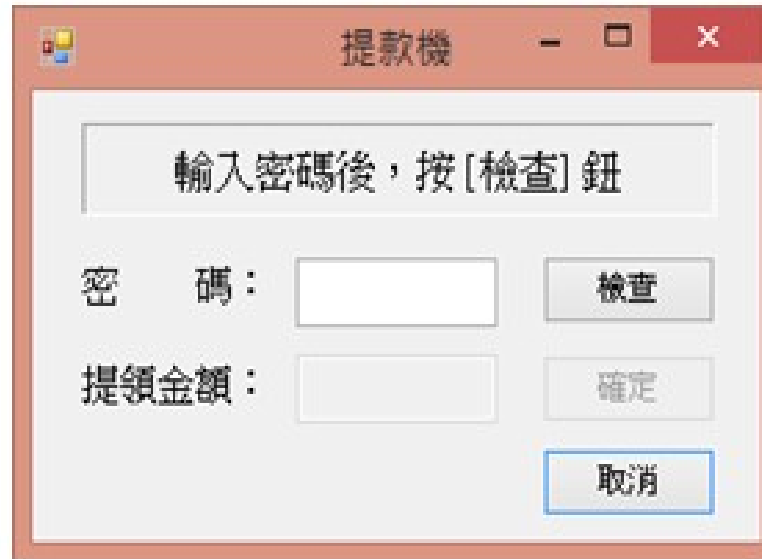
3. If the password is wrong, clear txtPW and change the hint as the left figure. If the password is right, show the hint as the right figure and enable txtMoney TextBox and “確定” button



4. The hint text changes as the figure after input the number of money and press “確定” button



5. The ATM returns to the initial condition after pressing “取消” button



提款機

輸入密碼後，按[檢查] 鈕

密 碼：

提領金額：

檢查

確定

取消

Property values:
Name = lblShow
AutoSize = False
BorderStyle = Fixed3D

The image shows a Windows form titled "提款機" (ATM). The form contains several controls and labels with arrows pointing to them from external text:

- lblShow**: A label at the top left of the form, indicated by an arrow from the text "Property values: Name = lblShow, AutoSize = False, BorderStyle = Fixed3D".
- txtPW**: A text box for password input, indicated by an arrow from the label "txtPW" on the right.
- btnCheck**: A button labeled "檢查" (Check), indicated by an arrow from the label "btnCheck" on the right.
- txtMoney**: A text box for withdrawal amount input, indicated by an arrow from the label "txtMoney" on the left.
- btnOk**: A button labeled "確定" (OK), indicated by an arrow from the label "btnOk" on the right.
- btnCancel**: A button labeled "取消" (Cancel), indicated by an arrow from the label "btnCancel" on the right.

Other visible controls include a label "密碼:" (Password:) next to the password text box, and a label "提領金額:" (Withdrawal Amount:) next to the amount text box.


```
private void btnCheck_Click(object sender, EventArgs e)
{
    string passwd = txtPW.Text;
    if (passwd == "123456")
    {
        lblShow.Text = "請輸入提領金額，按[確定]鈕";
        txtMoney.Enabled = true;
        btnOK.Enabled = true;
    }
    else
    {
        lblShow.Text = "密碼錯誤! 請重新輸入";
        txtPW.Clear();
    }
}

private void btnCancel_Click(object sender, EventArgs e)
{
    lblShow.Text = "請輸入密碼後，按[檢查]鈕";
    txtMoney.Clear();
    txtMoney.Enabled = false;
    btnOK.Enabled = false;
    txtPW.Clear();
}
```



Practice6.3 : System Input Interface

- **Hint:**

- **Set password to other sign like “ * ”**
- **Set a account to login**
- **Use textbox, label's Visible to show and hide.**
- **Use array to let user add new data, delete and search.**

The image displays two side-by-side screenshots of a Windows-style application window titled "Form1".

Left Screenshot: The form contains two labels in blue text: "帳號:" (Username) and "密碼:" (Password). Each label is followed by an empty text input field. Below these fields is a single button labeled "登入" (Login).

Right Screenshot: The form is identical to the left one, but the input fields are populated. The "帳號:" field contains the text "Supervi". The "密碼:" field contains seven asterisks "*****". The "登入" button is now disabled, indicated by its grayed-out appearance.

Log in menu

Form1

新增

查詢

刪除

登出

Form1

身分證字號 A123456789 新增

性別 查詢

電話 刪除

地址 確認 登出

各欄位不能為空, 請重新輸入

Add menu

Form1

身分證字號 新增

性別 查詢

電話 刪除

地址 確認 登出

Form1

身分證字號 A123456789

性別 男

電話 0910084789

地址 台北 確認

Form1

身分證字號 新增

性別 查詢

電話 刪除

地址 確認 登出

資料已存入
目前已有1筆資料!!

Form1

身分證字號

欄位不能為空！

搜尋

新增

查詢

刪除

登出

Form1

身分證字號

A124596789

無此筆資料

搜尋

新增

查詢

刪除

登出

Form1

身分證字號

A123456789

搜尋

新增

性別

男

查詢

電話

0910084789

刪除

地址

台北

登出

Search menu

Form1

身分證字號

A123456789

確認刪除?

新增

查詢

刪除

登出

Form1

身分證字號

刪除成功 !!

A123456789

確認刪除?

新增

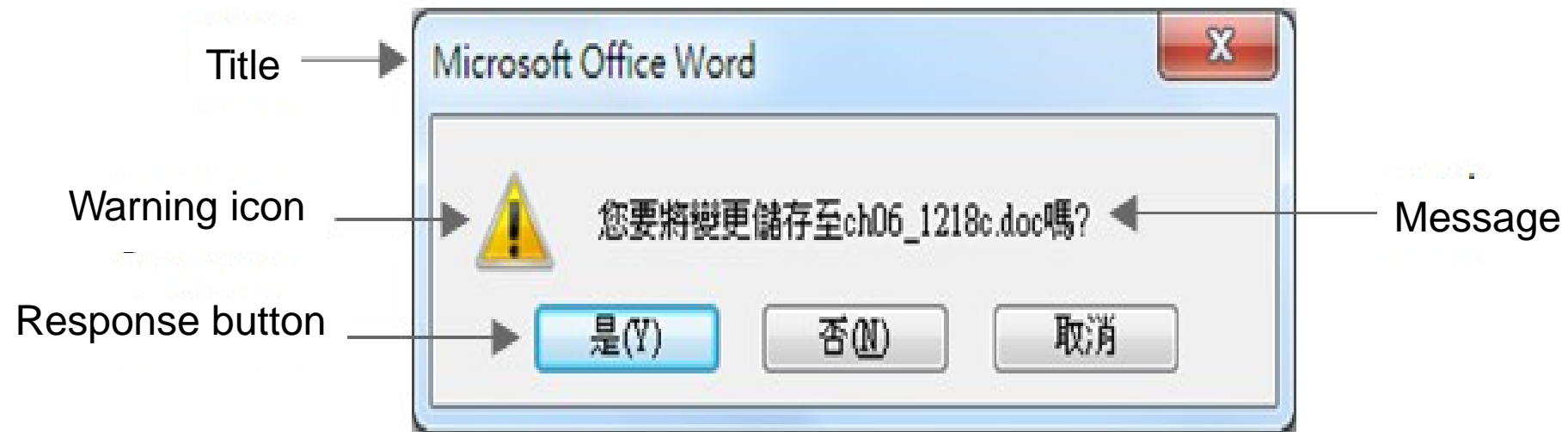
查詢

刪除

登出

Delete menu

6-8 MessageBox Class









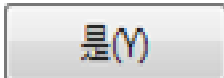
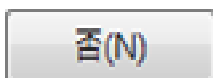
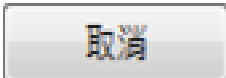
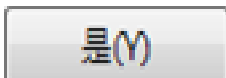



MessageBox.Show Method

Grammar

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

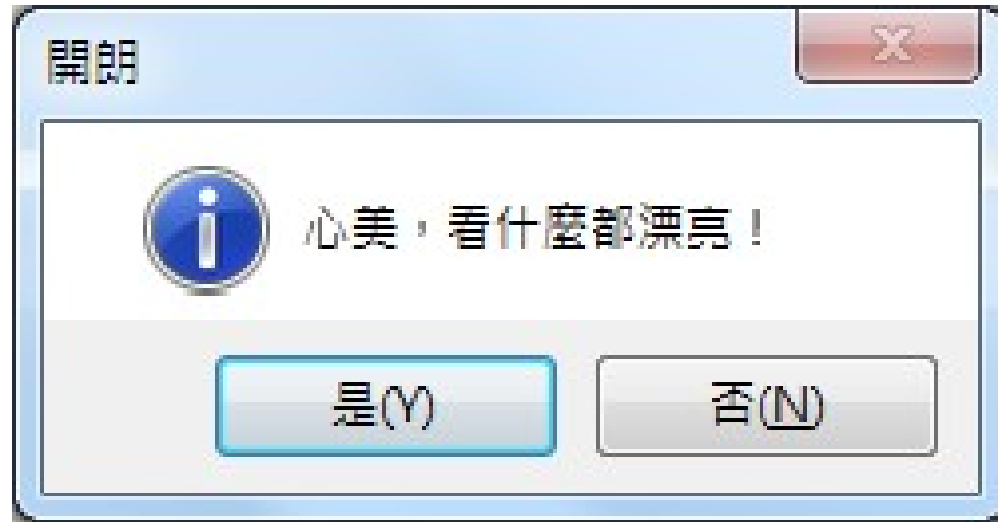
```
retValue = MessageBox.Show(msg[, title[, buttonConst[, iconConst]]]);
```


MessageBox.Show – Button Constant

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

MessageBox.Show – Icon Constant




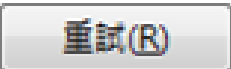


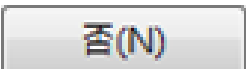
Icon Constant	Representation	Description
MessageBoxIcon.None	No icon	
MessageBoxIcon.Hand		Error message
MessageBoxIcon.Question		Question message
MessageBoxIcon.Exclamation		Warning message
MessageBoxIcon.Asterisk		Information message



Usage:

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

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();
}
```

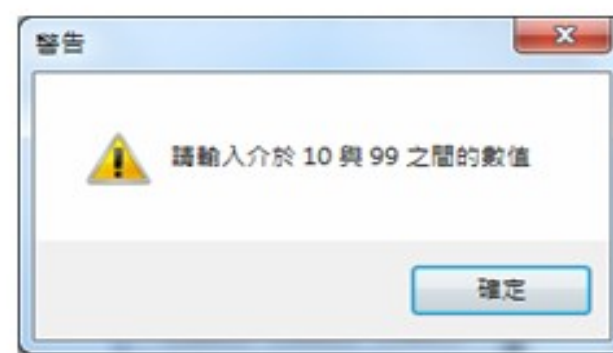
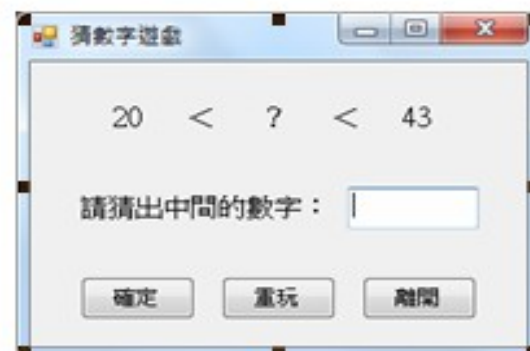
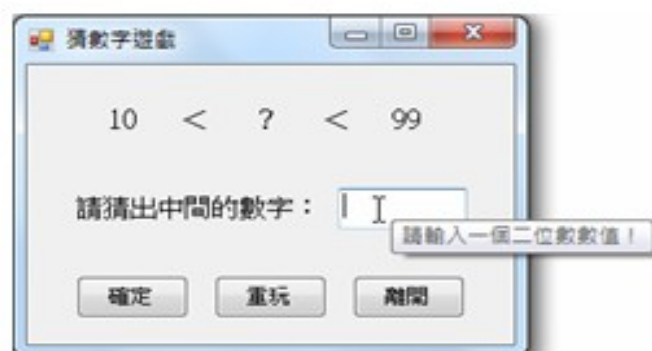
Example(Guess):

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

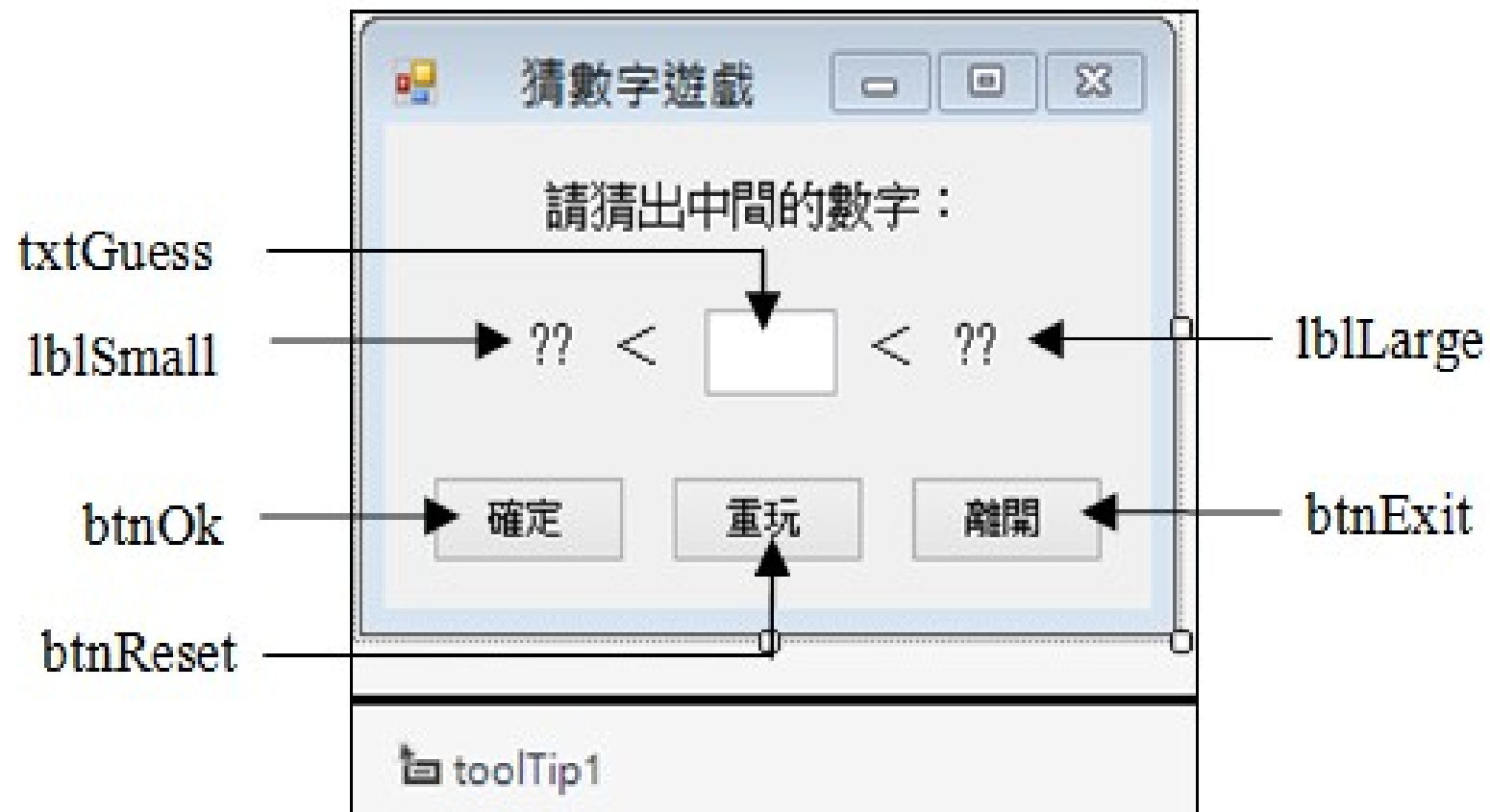
1. There is a TextBox, which shows a tool tip "Please input a 2-digit number!" when the mouse is on this TextBox
2. The generated random number has between 10 and 99. If the player's answer is correct, use MessageBox to show "Congratulations!"
3. If the data of TextBox is not a number, show an error 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



4. Debug

1. Start debug

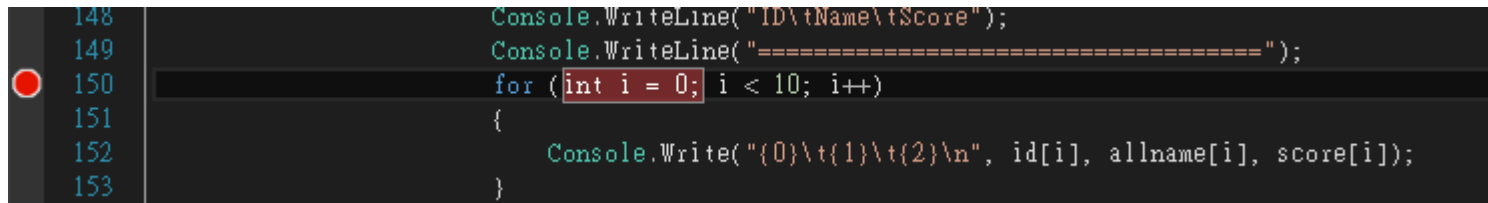
press F5 or Debug(D)/Start debug(S) to compile and run the program, examine whether every function fits the requirements or not

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.

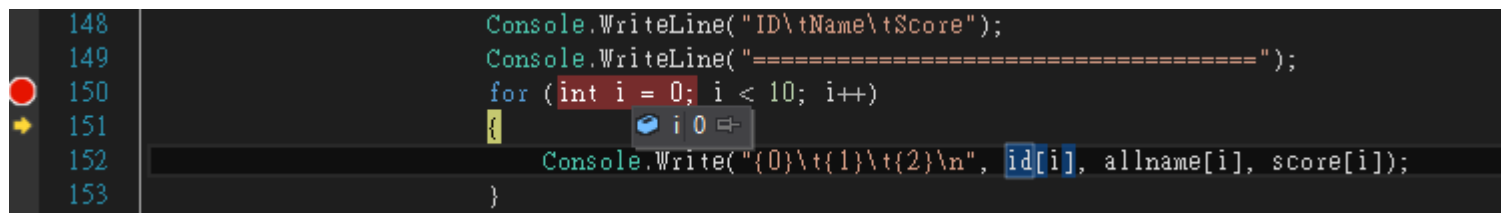
Debug

- 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.WriteLine("{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.WriteLine("{0}\\t{1}\\t{2}\\n", id[i], allname[i], score[i]);
153 }
```

Practice6.4 . Change US dollar

Write the code of the textBox1's Changed event handler. The label3 shows the NT when the textBox1 gets the number of US dollars.

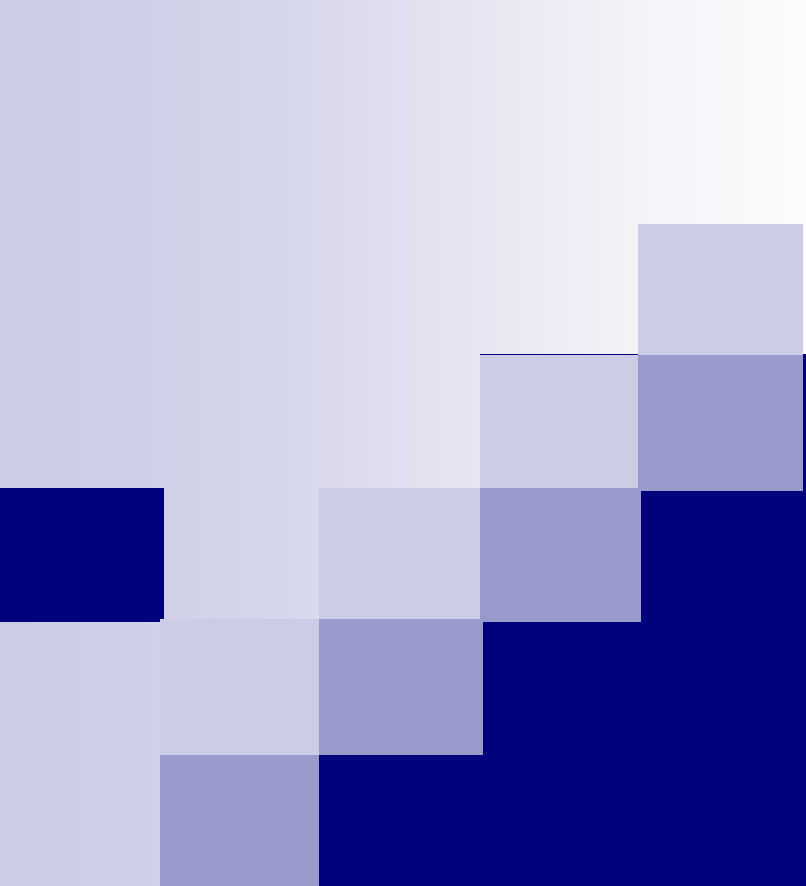
The image displays three sequential screenshots of a Windows application window titled "Form1". Each screenshot shows a currency conversion interface with the following elements:

- A title bar with standard Windows window controls (minimize, maximize, close).
- A label at the top: "一塊美金可兌換33塊台幣" (1 US dollar can be converted to 33 NT dollars).
- A label and a text box: "美金 : " followed by a text box containing a numerical value.
- A label at the bottom: "可兌換: " followed by a numerical value and "台幣" (NT dollars).

The three screenshots illustrate the state of the application for different input values in the "美金" (US dollar) text box:

美金 (US Dollar)	可兌換 (NT Dollar)
0	0
35.9	1184
220	7260

In the middle screenshot, an additional button labeled "瀏覽器連結" (Browser Link) is visible in the title bar area.



End

Take a Break