



# Chapter 13

## **File and Multimedia Design**

# 13-1 File and Stream

- **Data process in previous chapters**
  - ⇒ data and source code in the same place, get input data from keyboard, output data is not saved
- **Disadvantages of putting data and code together**
  - ⇒ have to modify data in the source code
  - ⇒ key in data every time the program runs
  - ⇒ execution is required to see the result
- **Advantages of separating data and code**
  - ⇒ use program to modify data files
  - ⇒ one program can process many data files

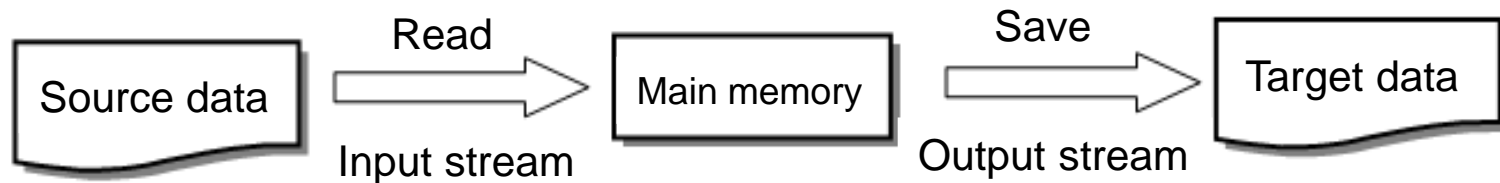


## 13-1 File and Stream

- C# uses stream to process data input and output
- Stream is like a tap that water flows from higher place to lower place
  - ⇒ single direction, the processed character cannot be processed again
- Stream is assembled by characters or bytes
- Data process can be divided into “character stream” and “data stream”

# 13-1 File and Stream

- **Input stream**  
data from keyboard or file reading
- **Output stream**  
result of data process is saved into file, printed or shown on the screen





# 13-1 File and Stream

Types of data file:

## 1. Text file

- Save character data, every byte is character
- Character stream
- Process character by character in single direction

## 2. Binary file

- Also called non-character file, Bytes data not processed
- Binary stream
- As scrambled code in text editor

## 13-2 System.IO Namespace

- When C# program uses stream, there is a code required:  
**using System.IO;**
- System.IO classes for file and directory process

Class	Description
Directory	Providing directory of creating, deleting, moving and showing. All methods are static methods, object realization is not required
DirectoryInfo	Providing directory of creating, deleting and so on. Methods are identical to methods of Directory, but object realization is required
File	Providing file of creating, opening, duplication, deleting and moving. All methods are static, object realization is not required

Class	Description
FileInfo	Providing file creating, opening and so on. Methods are identical to methods of File, but object realization is required
FileStream	Providing file I/O process. Synchronous and asynchronous file reading and writing are supported
StreamWriter	Character stream file writer
StreamReader	Character stream file reader
BinaryWriter	Binary stream file writer
BinaryReader	Binary stream file reader

## 13-3 Directory Operation

- Create, delete, move and get working directory in C#:
  1. Directory class
  2. DirectoryInfo class



# Directory Class

- All methods are static methods
- Ex: create “my” folder and its sub folder “test” in C: root directory  
`Directory.CreateDirectory("c:\\my\\test\\")`
- Add @ symbol before path string for ignoring \ as an escape character  
`Directory.CreateDirectory(@"c:\my\test\")`

# DirectoryInfo Class

- All methods are NOT static methods
- The way to create “D:\my\test1” with DirectoryInfo class is rewritten in DirectoryInfo class:

```
DirectoryInfo dirInfo = new DirectoryInfo(@"D:\my\test1\");  
dirInfo.Create();
```

# Directory Static Methods

## 1. CreateDirectory(path)

create the folder of designated path. The parent folders are also created if they do not exist

**Ex: create directory “test” under “D:\my”, there are 2 usages**

```
Directory.CreateDirectory(@"D:\my\test\");
```

```
Directory.CreateDirectory("D:\\my\\test\\"); ⇐ 省略@
```

# Directory Static Methods

## 2. Exists(path)

examine whether the directory of path is in existence or not. Return true for existence and return false for not

**Ex: examine “D:\my” is in existence or not. Create D:\my if the directory does not exist**

```
string path = @"D:\my\";  
if (Directory.Exists(path) == false)    // if path does not exist  
    Directory.CreateDirectory(path);    // create path
```

if (Directory.Exists(path) == false) can be rewritten in  
if (!Directory.Exists(path))

# Directory Static Methods

## 3. Delete (path, recursive)

- Delete the folder designated path
- If recursive = true, the sub folders and files are also deleted
- If recursive = false and there are folders or files in the directory, deleting is cancelled

**Ex: delete all sub folders and files in D:\my directory**

**Directory.Delete(@"D:\my\", true);**



# Directory Static Methods

## 4. Move(sourceDir, destDir)

- Move all files and folders in sourceDir to destDir
- Ex: move folders and files in D:\my to D:\you  
`Directory.Move(@"D:\my", @"D:\you");`



# Directory Static Methods

## 5. **GetDirectories(path)**

get list of child directories of the path's designated folder, return string array

## 6. **GetFiles(path)**

get list of child files of the path's designated folder, return string array

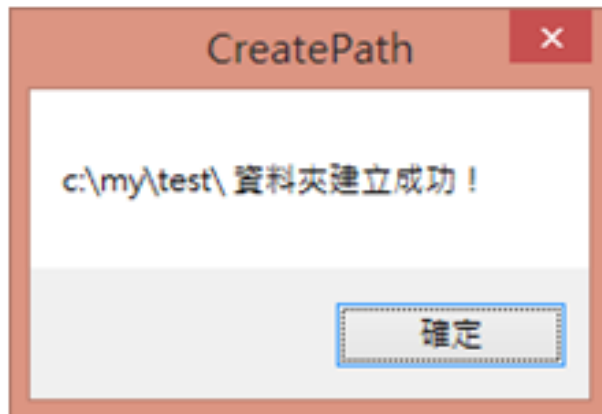
## Example(CreatePath):

Design a program to create folders. Requirements:

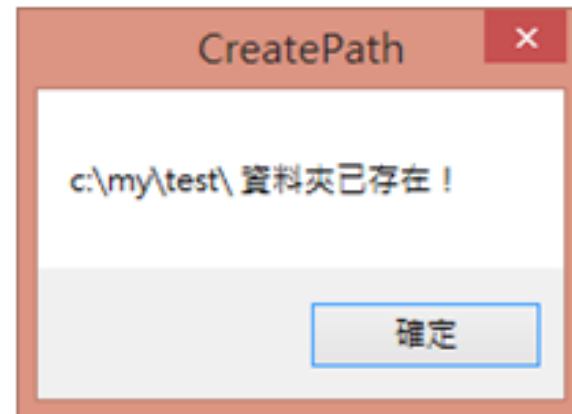
Create C:\my\test folder in C: disc

1. If the folder exists, show message “C:\my\test資料夾已存在”
2. If the folder does not exist, create the folder and show message “C:\my\test資料夾建立成功”

## Result:



或

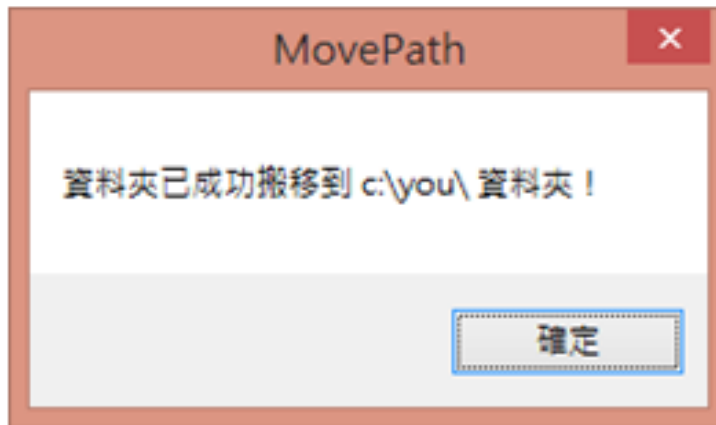




## Example(MovePath):

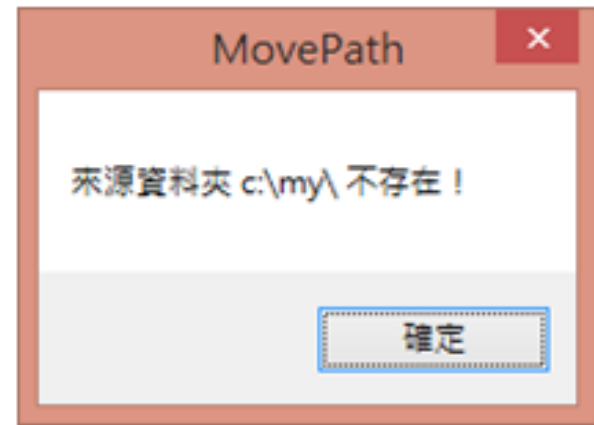
Design a program to move folders. Requirements:  
Move folder “C:\my”, its sub folders and files to “C:\you”

## Result:



↑ Directory moving successfully

或



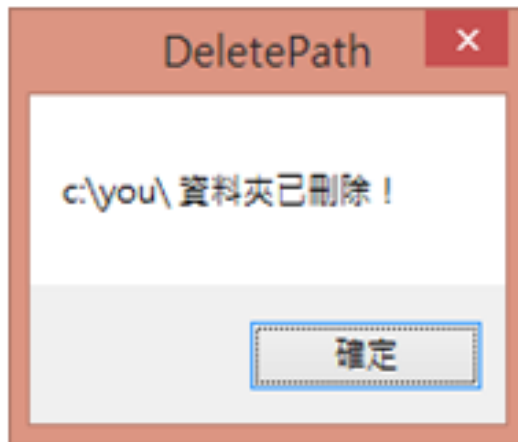
↑ Directory don't exist

## Example(DeletePath):

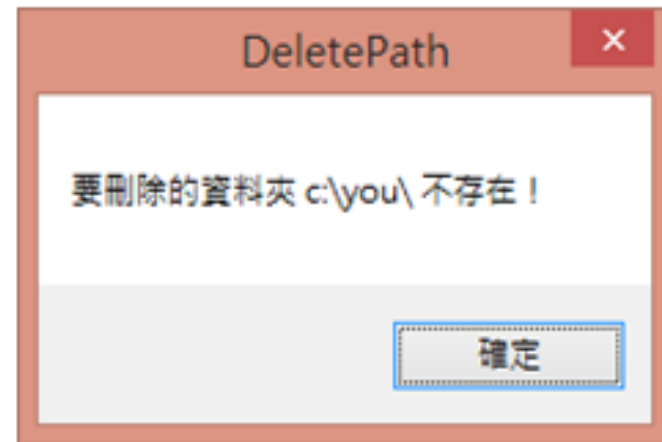
From previous example, design a program to delete folders. Requirements:  
Delete C:\you folder, its sub folders and files

1. If the folder is deleted successfully, show the left figure
2. If the folder you are going to delete does not exist, show the right figure

## Result:



或





## 13-4 File Operation

- **C# uses File class and FileInfo class to access file data**
- **File class**  
all methods are static methods  
⇒ use **File.methodName** to call the method
- **FileInfo class**  
realization is required

# FileInfo Constructor

## 1. FileInfo(String path)

- For building solid file object
- Parameter path is path of directory or file
- Ex: create a FileInfo object called finfo and open “D:\my\htc.txt”

**FileInfo finfo = new FileInfo(@"d:\my\htc.txt");**

**or**

**FileInfo finfo = new FileInfo("d:\\my\\htc.txt");**

# FileInfo Properties

Property	Description
Exists	Acquire whether the designated file is in existence or not true: exist, false: do not exist. Ex: <code>finfo.Exists</code>
FullName	Get full path of the file, the path includes directories ex: <code>finfo.FullName</code>
Name	Get the file name with the extension Ex: <code>finfo.Name</code>
Extension	Get the extension of the file Ex: <code>finfo.Extension</code>
DirectoryName	Get the path of the directory which the file belongs to. Ex: <code>finfo.DirectoryName</code>
Length	Get the size of file Ex: <code>finfo.Length</code>

# FileInfo Methods

## 1. Create()

- To create and open the file assigned by FileInfo object
- The path has to exist , otherwise an error will occur.
- Ex: finfo is a FileInfo object and finfo stands for “D:\my\htc.txt”

```
FileInfo finfo = new FileInfo("d:\\my\\htc.txt");  
FileStream fs = finfo.Create();           // create htc.txt
```

# FileInfo Methods

## 2. Close()

- To close the file referred by the opened FileInfo object
- File stream in the same function has to be closed if the stream is no longer to be used  
⇒ otherwise repeated realization error will occurs
- Ex: close the file referred by FileInfo object fs  
`fs.Close();`



# FileInfo Methods

## 3. CopyTo (String path, Boolean)

- Copy the current file to the file assigned by path, the source file still exists
- Copy to bin\Debug of project directory if the path is not defined
- The path of the target directory has to be created in advance, otherwise the error will occur when copying
- If the second parameter is true, overwrite the file if it is already in existence, and false for not overwriting



# FileInfo Methods

**Ex: copy the file “D:\my\htc.txt” assigned by finfo to  
“D:\you\newhtc.txt”**

```
FileInfo finfo = new FileInfo(“d:\\my\\htc.txt”);  
finfo.CopyTo(“d:\\you\\newhtc.txt”, true);
```

# FileInfo Methods

## 4. MoveTo(String path)

- Move the current file to the file assigned by path, the source file is removed
- Moving does not have overwriting function, the target directory has to be created and the target file cannot exist before moving

**Ex:** move the file “D:\my\htc.txt” assigned by finfo to “D:\you\newhtc.txt”

```
FileInfo finfo = new FileInfo(“d:\\my\\htc.txt”);  
finfo.MoveTo(“d:\\you\\newhtc.txt”);
```

# FileInfo Methods

## 5. Delete()

- Delete the designated file referred by FileInfo object
- Ex: delete the file “D:\you\newhtc.txt” assigned by finfo object

```
FileInfo finfo = new FileInfo("d:\\you\\newhtc.txt");  
finfo.Delete();
```



# FileInfo Methods

## 6. CreateText()

- Create and open a new text file assigned by FileInfo object
- The new file has no data, StreamWriter is required for writing data
- If the file exists, clear the content. If the file does not exist, create a new file
- The designated path of the directory has to be created in advance otherwise the errors would occur

# FileInfo Methods

- **Ex: create a new file called “htc.txt” in D:\my directory and assign StreamWriter “sw” as output data stream to write output data into the file**  
**FileInfo finfo = new FileInfo(“d:\\my\\htc.txt”);**  
**StreamWriter sw = finfo.CreateText();**

# FileInfo Methods

## 7. AppendText()

- Use StreamWriter to append the data to the end of the text file
- If the file exists ⇒ open the file
- If the file does not exist ⇒ create a new file
- The directory has to be created in advance, otherwise the error will occur

Ex: open the text file “D:\my\htc.txt” and assign StreamWriter sw as output data stream. Write output data at the end of file with output data stream

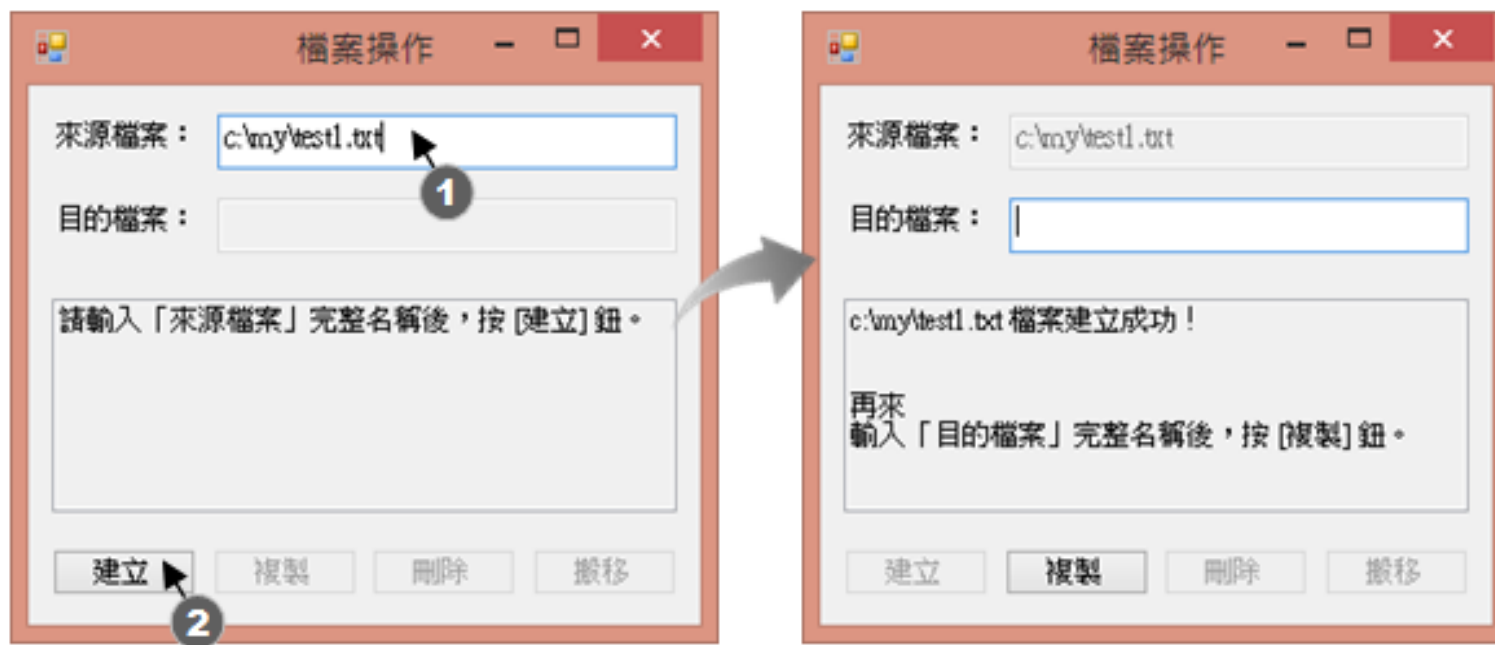
```
FileInfo finfo = new FileInfo("d:\\my\\htc.txt");  
StreamWriter sw = finfo.AppendText();
```

## Example(FileOperate):

Design a program which can create, copy, delete and move data file.

Requirements:

1. The program starts as shown in figure 1. “建立” button is available but others are not
2. Input “C:\my\test1.txt” in source file, and press “建立” button to show message “C:\my\test1.txt檔案建立成功!” as shown in figure 2. In the meantime, “複製” button is available and others are not



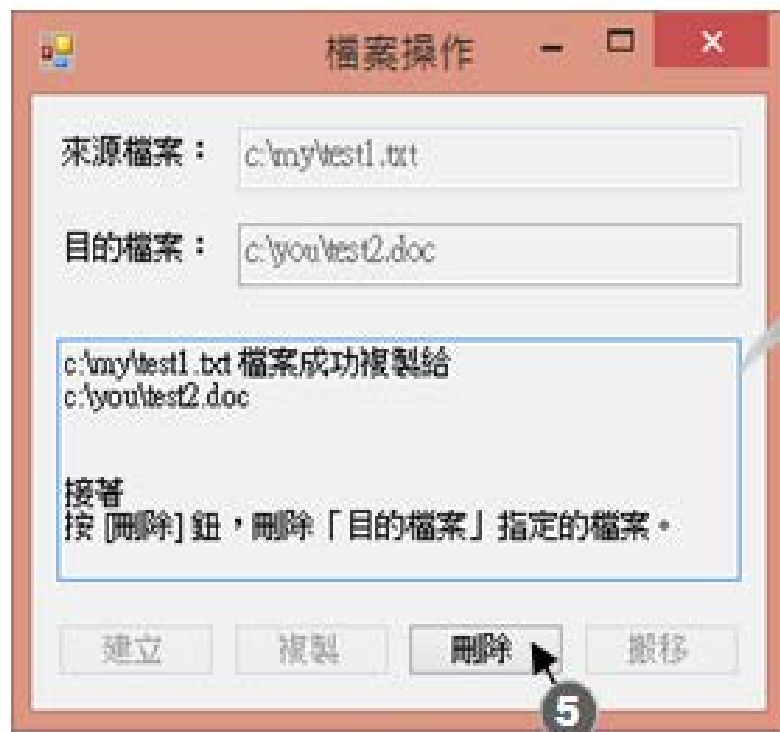


↑圖 3



↑圖 4





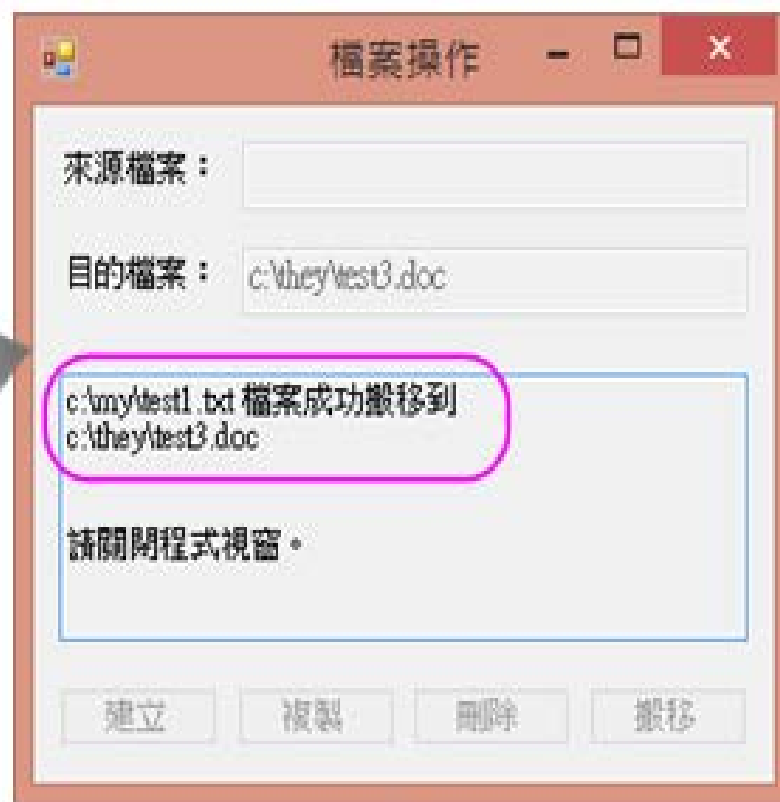
↑圖 5



↑圖 6



↑圖 7



↑圖 8

# Design User Interface



# 13-5 Text Data Read & Write

- **StreamWriter Class**  
⇒ process text data writing
- **StreamReader Class**  
⇒ process text data reading
- **StreamWriter Methods:**

Method	Description
<code>Write(String)</code>	Write string data into the file and attach to the end of file
<code>WriteLine(String)</code>	Write string data and new line character into the file
<code>Flush()</code>	Clear buffer data
<code>Close()</code>	Close data stream object



**Ex: use the methods provided by FileInfo and  
StreamWriter to write data. The data is written into  
D:\my\apple.txt through StreamWriter sw**

# Steps to Write Text File

## **Step 1** create FileInfo object

```
FileInfo finfo = new FileInfo(@"d:\my\apple.txt");
```

```
FileInfo finfo = new FileInfo("d:\\my\\apple.txt");
```

## **Step 2** choose the way to open data file

### 1. use CreateText()

```
StreamWriter sw = finfo.CreateText();
```

### 2. use AppendText()

```
StreamWriter sw = finfo.AppendText();
```



### **Step 3** write data

1. `sw.Write(string)`

write the string and attach to the end of file

2. `sw.WriteLine(string)`

write the string with new line character

### **Step 4** put data of output stream into the file and erase the buffer

`sw.Flush();`

### **Step 5** Close the file

`sw.Close();`

**Ex1: put input data of textBox1 into the string array called product line by line**

```
int k = 0;
foreach (string item in textBox1.Lines )
{
    product[k] = item ;
    k++ ;
}
```

**Ex2: show every element of product string array on label1. The AutoSize property of label1 should set to false before execution, so can enlarge manually for multi-line display**

```
foreach (string item in product )
    lblShow.Text += item + "\n";
```



**Ex3: create FileInfo object called finfo, the object opens D:\my\apple.txt text file. The file is created if the file does not exist**

```
string filename = @"D:\my\apple.txt";  
FileInfo finfo = new FileInfo(filename);  
if (!Directory.Exists(finfo.DirectoryName))  
    Directory.CreateDirectory(finfo.DirectoryName);
```

**Ex4: create and open the text file referred by finfo object. The data output stream sw is also created. To output data, use the data output stream to write data into the file referred by finfo object**

```
StreamWriter sw = finfo.CreateText();
```

**Ex5: write every element of product string array into the file assigned by sw stream object**

```
foreach (string item in product)
    sw.WriteLine(item);
```

**Ex6: write all data of textBox1 into the file assigned by sw stream object**

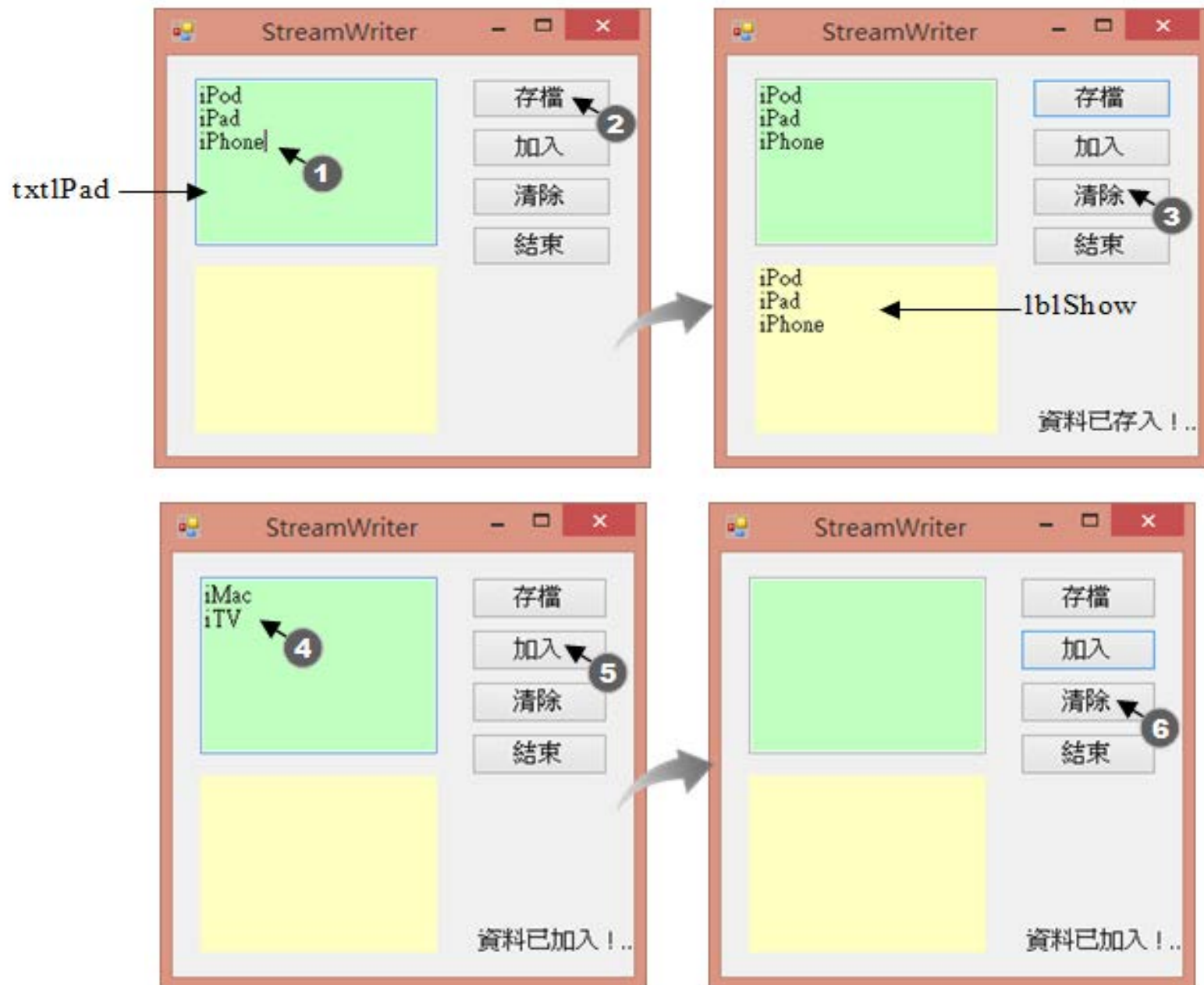
```
sw.WriteLine(textBox1.Text);
```

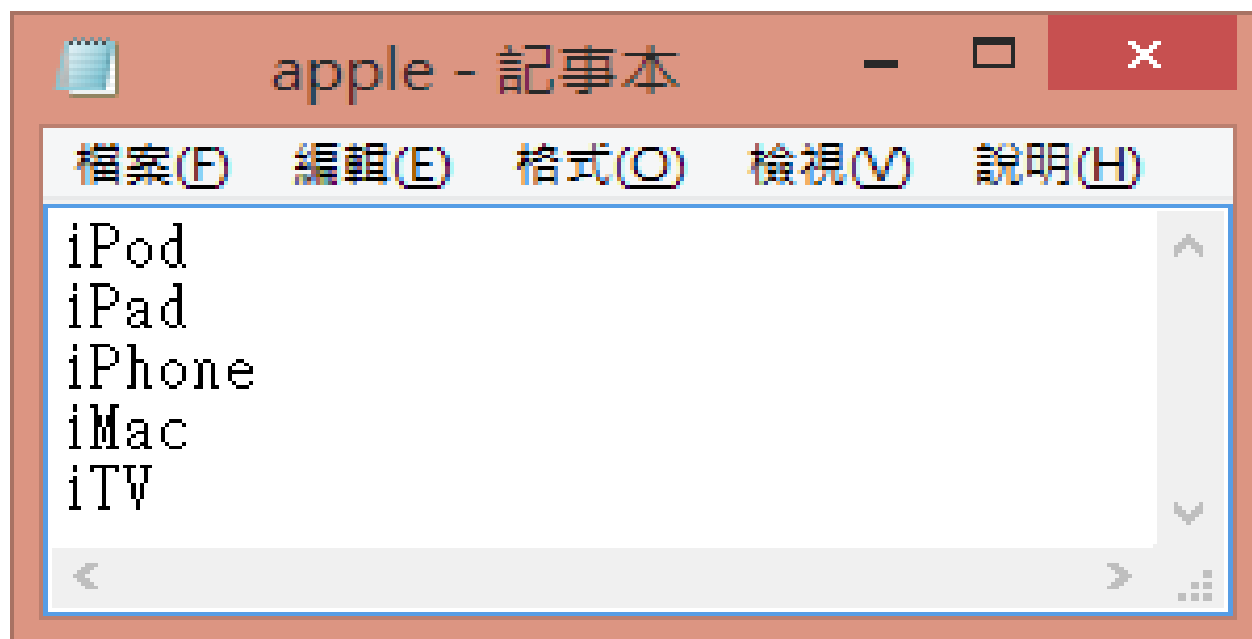
## Example(StreamWriter1):

Design a program to write text inputted by keyboard into the text file.

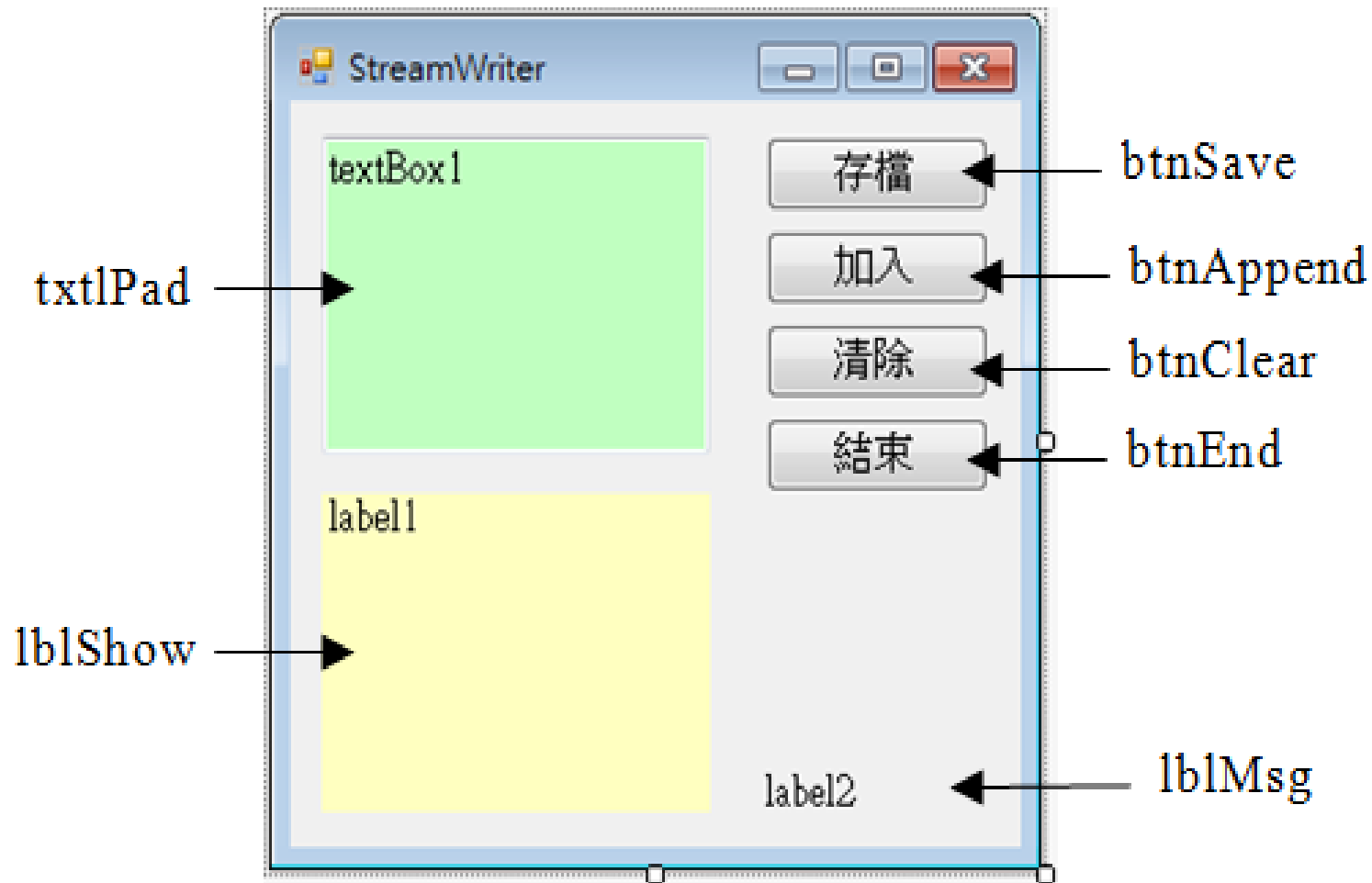
Requirements:

1. Input 3-line text into txtIPad text box with keyboard. Use enter to change to new line.
2. Press “存檔” button to verify and save 3-line text into d:\my\apple.txt, then show the message in lblShow label





# Design User Interface





## Practice13.1 (write)

- Let textbox can save as txt.
- Build the file at the first, if success, show message box.
- After successful create file, let user type and save.
- Can also let user move file to other place.



Form1

Warning

創建成功

確定

目的路徑

建立檔案

存檔

搬移

myFile.txt





# Read Character Stream Data from Text File

- Use **StreamReader** class to read character stream data from the text file
- When reading:
  - ① read a character at a time
  - ② read a line of string at a time
  - ③ read the whole text

# StreamReader Methods

Method	Description
Read()	Read a character or a Chinese word from text file
ReadLine()	Read a line of string without new line character from text file. Return null if read to the end of data stream
ReadToEnd()	Read from the beginning of file to the end of file
Peek()	Examine the next character for reading, return -1 if it is the end of the file

# Steps to Read Text File

## Step1 create stream reader

`StreamReader sr = new StreamReader(path);`

## Step2 read data

1. read a character every time

Ex: in the loop, use `Read()` to get a character every time, and use `Peek()` to check the document is fully read or not. `Peek()` returns -1 when reading is finished and exit the loop

```
do
{
    ch = (char)sr.Read();
    if (sr.Peek() == -1)
        break;
    txtShow.Text += ch;
} while(true);
```

Use MessageBox to check the read character

```
DialogResult result;  
result = MessageBox.Show("按<是>鈕繼續輸出 按<否>鈕離開, "是否繼續",  
    MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);  
if (result == DialogResult.No) // 按 <否>鈕  
{  
    txtShow.Text = "";  
    return;  
}
```

## 2. read a line of string every time

Ex: in the loop, use `ReadLine()` to get a string which has no new line character every time, add “`\r\n`” manually each time when displayed. If already read to the end of data stream, return null to check whether reading has ended.

```
do
{
    data = sr.ReadLine();
    if (data == null) break;
    txtShow.Text += data + "\r\n";
} while(true);
```

Escape char	Description
\'	Insert a single quote
\"	Insert a double quote
\\	Insert a back slash
\a	Trigger a system alarm sound
\b (Backspace)	Backspace
\f (Form Feed)	Insert a form feed
\n (New line)	Insert a new line
\r (Return)	Move cursor to the beginning of line
\t (Tab)	Insert a tab
\udddd	Insert an Unicode character
\v	Insert a vertical tab
\0 (Null space)	A null space

**3. read one chapter at a time**  
**use ReadToEnd() to read text from the current**  
**position of the cursor to the end of file**

```
txtShow.Text = sr.ReadToEnd();
```

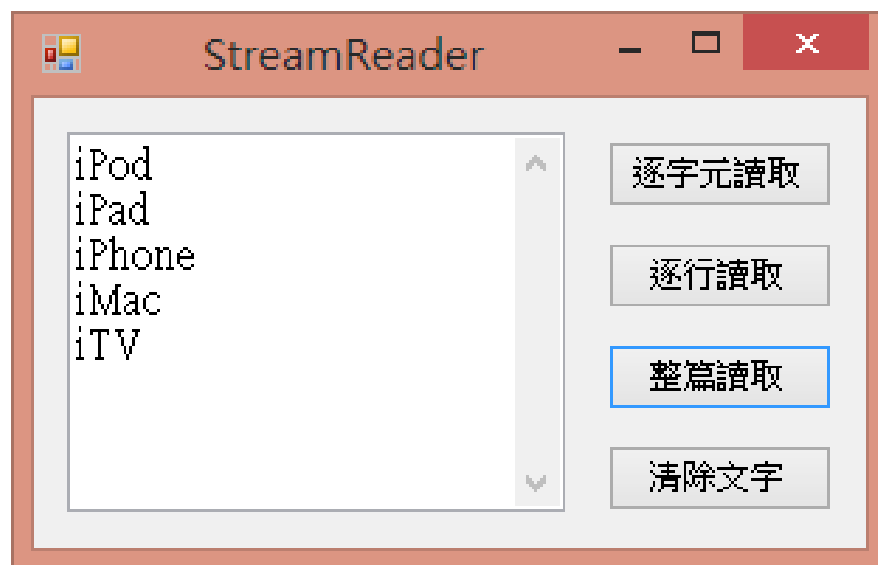
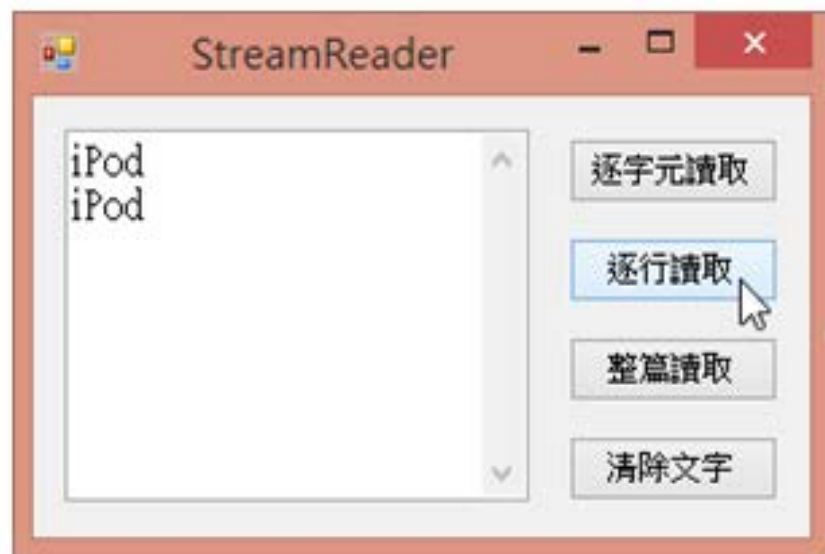
**Step3 close data stream**  
**sr.Close();**

```
sr.Close();
```

## Example(StreamReader1):

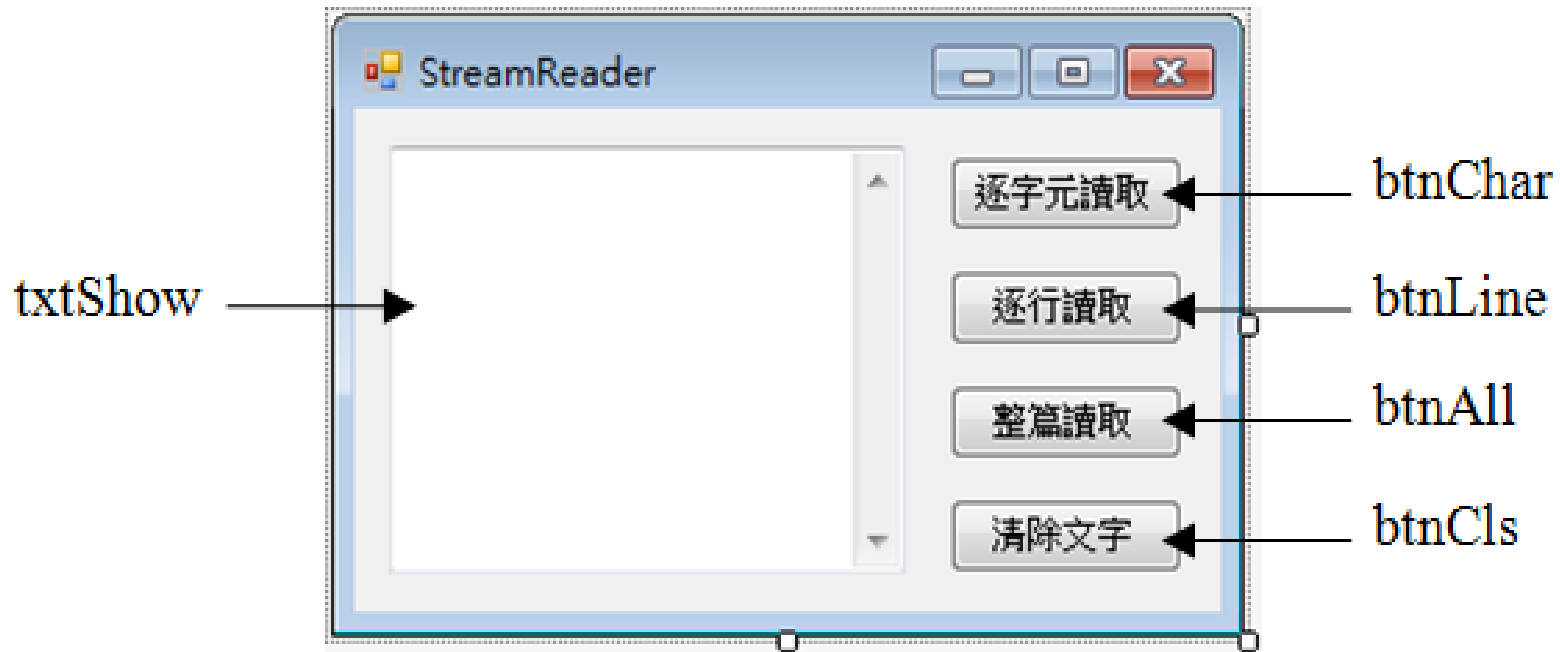
Design a program to load text file. The functions include “逐字元讀取”, “逐行讀取” and “整篇文章讀取”. The example file is D:\my\apple.txt. The program reads the characters by the chosen function.







# Design User Interface



# 13-6 Binary File Read and Write

- C# provides `FileStream`, `BinaryWriter` and `BinaryReader` for processing binary data writing and reading
- Write or read binary file in **bit**
- ① Use `FileStream` object to link file and set the file open mode
- ② Assign `BinaryReader` and `BinaryWriter` to process binary data stream



# FileStream() Constructor

- **C# provides several types of FileStream(), this chapter introduces the FileStream() with 2 parameters to open binary file**  
**FileStream(string path, FileMode.mode);**
- **First parameter path: assign binary file name for creating or opening**
- **Second parameter FileMode.mode: whether file access mode is set**

## FileMode Enumeration

FileMode.mode const	Description
FileMode.Create	Create new file, overwrite the file if it exists, the file is in written mode
FileMode.Open	Open current file, error occurs if the file does not exist, the file is in read mode
FileMode.OpenOrCreate	If the file exists, open it; if the file does not exist, create it. The file is in read and write mode
FileMode.Append	If the file exists, open it and append data to the end of file; if the file does not exist, create it. The file is in written mode

**Ex: create a FileStream object called fsOut, and fsOut links to D:\my\data.bin. If the file does not exist, create it; if the file exists, open it. File mode is OpenOrCreate**

```
FileStream fsOut = new FileStream("d:\\my\\data.bin", FileMode.OpenOrCreate);
```

**Ex: create a FileStream object called fsIn, and fsIn links to D:\my\data.bin. If the file does not exist, show error message; if the file exists, open it. File mode is Open**

```
FileStream fsIn = new FileStream("d:\\my\\data.bin", FileMode.Open);
```

# BinaryWriter Class

- **How to**

- ① create **FileStream** object called **fsOut**

- ② create **BinaryWriter** object called **bw**

- ③ write binary output stream into the designated binary file

```
FileStream fsOut = new FileStream("d:\\my\\data.bin",  
    FileMode.OpenOrCreate);
```

```
BinaryWriter bw = new BinaryWriter(fsOut);
```

- **Can be merged into 1 line:**

```
BinaryWriter bw = new BinaryWriter(new  
FileStream("d:\\mydata.bin",FileMode. OpenOrCreate);
```



# BinaryWriter Methods

Method	Description
<code>Write(DataType)</code>	Write assigned data type into file stream
<code>Flush()</code>	Clear buffer of writer and write data in buffer into file
<code>Close()</code>	Close the current BinaryWriter data stream

# BinaryReader Class

- How to
  - ① create `FileStream` object called `fsIn`
  - ② create `BinaryReader` object called `br`
  - ③ read designated binary file “D:\mydata.bin” to stream

```
FileStream fsIn = new FileStream (“d:\mydata.bin”,  
    FileMode.Open);  
BinaryReader bw = new BinaryReader(fsIn);
```
- Can be merged into 1 line:

```
BinaryReader bw = new BinaryReader(new  
FileStream("d:\mydata.bin",FileMode.Open);
```



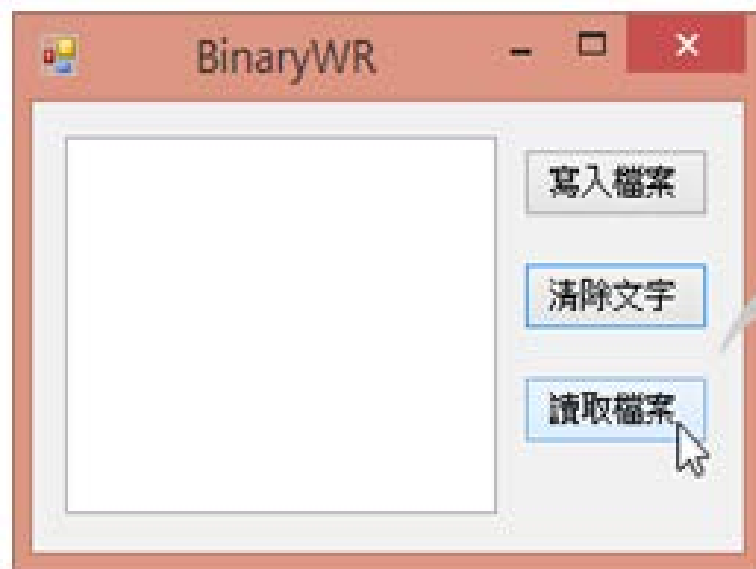
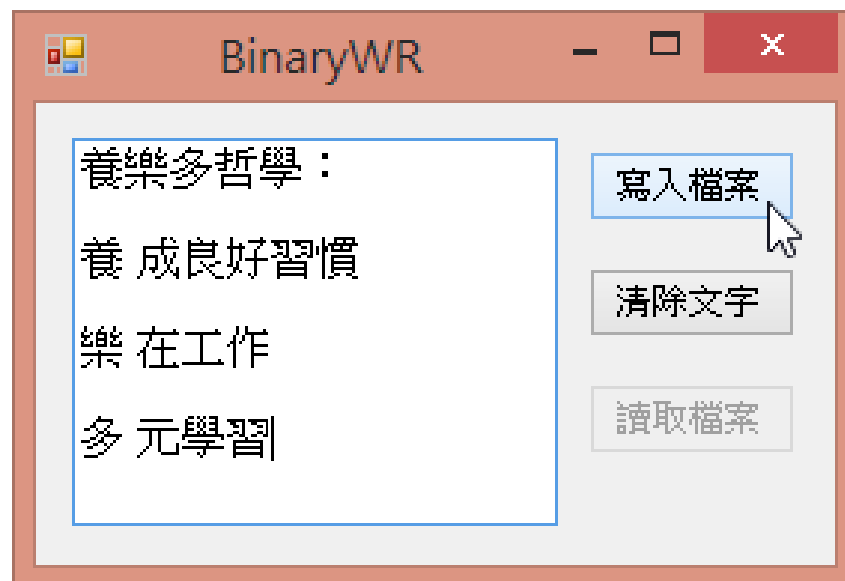
## BinaryReader Methods

Method	Description
Close()	Close current reader and data stream
ReadString()	Read string data from opened data stream
ReadBytes()	Read designated bytes from current data stream, and put data into byte array

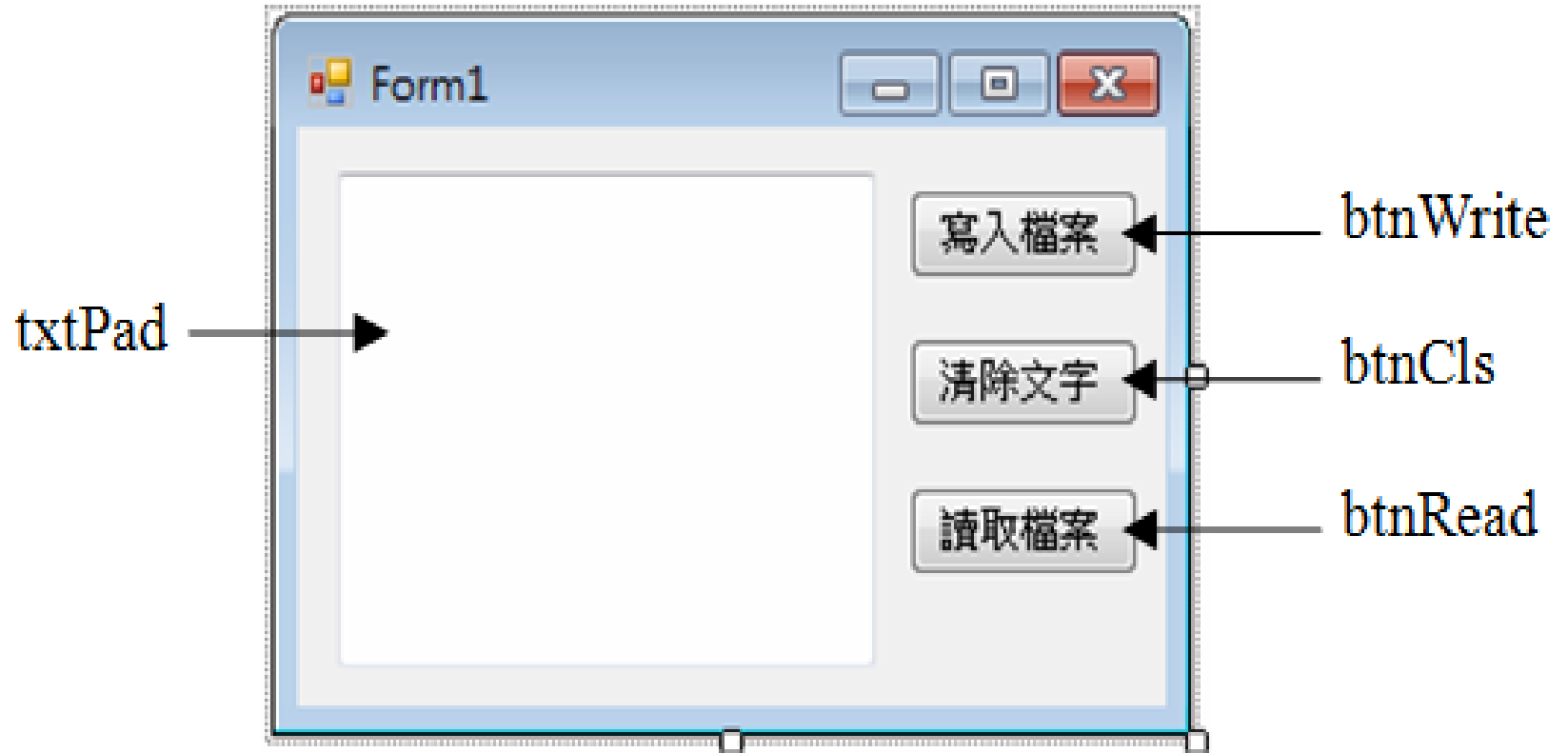
## Example(BinaryWR):

Apply the method of BinaryWriter and BinaryReader mentioned before, now write a program for reading and writing binary file. Requirements:

1. When the program starts, “寫入檔案” and “清除文字” buttons are available but “讀取檔案” button is disabled
  - ① Input text in the text box
  - ② Press “寫入檔案” button to write text into D:\test\data.bin in binary mode
  - ③ All buttons are available
2. Press “清除文字” button to clear text in the text box
3. Press “讀取檔案” button to load content of D:\test\data.bin and put into the text box

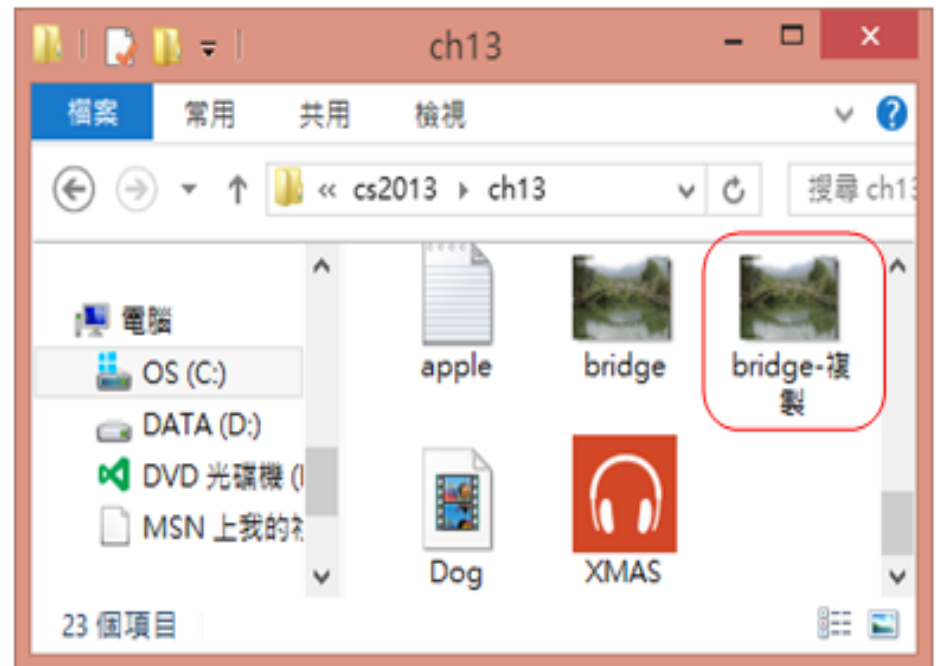


# Design User Interface

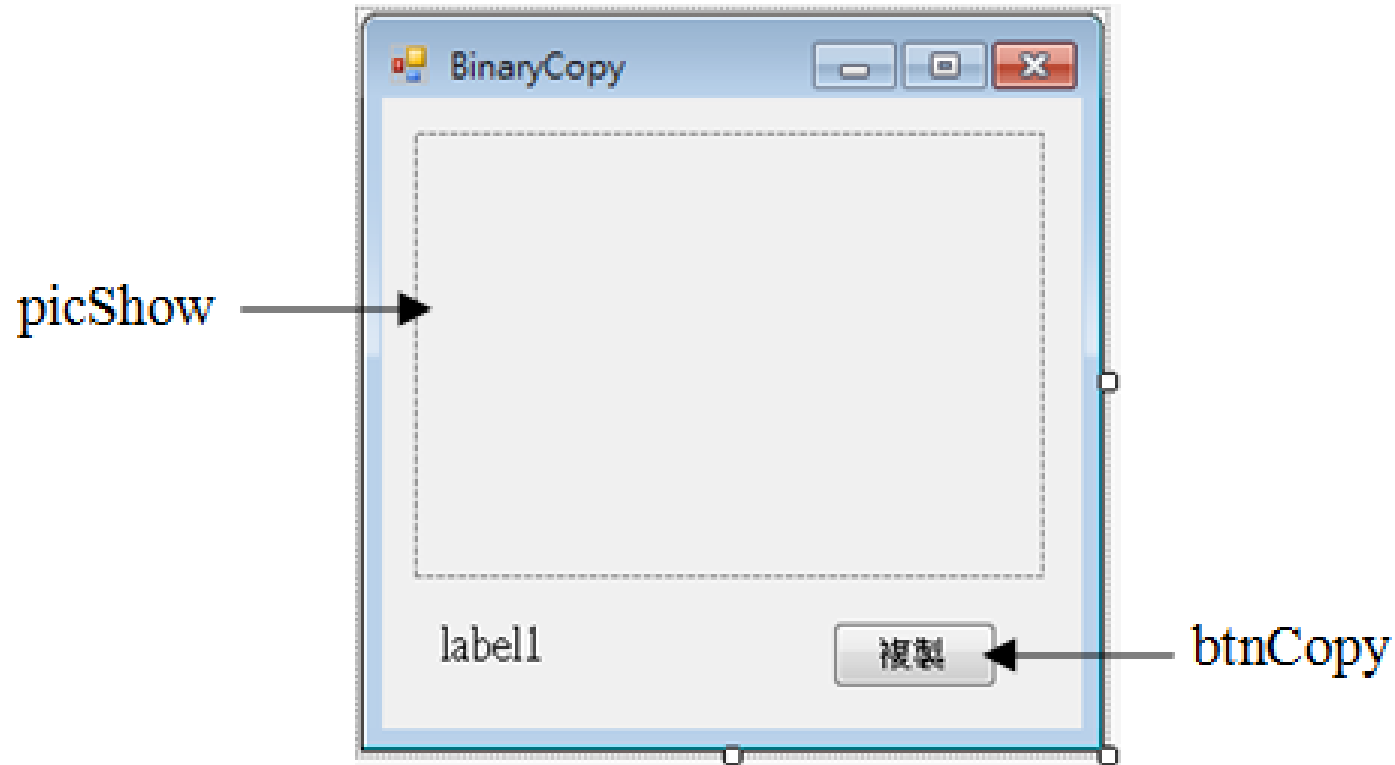


## Example(BinaryCopy):

Use binary stream to load the designated image file and show the file in the picture box control item. Press “複製” button to copy the file in the same directory and rename it.



# Design User Interface



# Practice13.2 (binary)

Like Practice(write), Add BinaryWriter, BinaryReader function to read, save and output the words.

Use Practice(write) to read and write and use “寫入二進位檔，從二進位檔讀出”， create a dat file to read and output.

Form1

哈囉

Text you write

Text you read

目的路徑

建立檔案 存檔 搬移

寫入二進位檔 從二進位檔讀出

Form1

哈囉

Text you write

Text you read

目的路徑

建立檔案 存檔 搬移

寫入二進位檔 從二進位檔讀出

DAT myFile

## 13-7 Play Sound

- Common file format: .wav
- C# program can play .wav file, System.Media namespace is required:  
    using System.Media;
- System.Media Classes

Class	Description
SystemSounds	Get effect sound configuration of Windows OS
SoundPlayer	Control play of .wav file



# SystemSounds Class

- Get event-related effect sounds of Windows operating system. This class cannot be inherited

## 1. SystemSounds Properties

Property	Description
Asterisk	Get current sound related to Asterisk program event in Windows sound configuration
Beep	Get current sound related Beep program event in Windows sound configuration
Exclamation	Get current sound related to Exclamation program event in Windows sound configuration
Hand	Get current sound related to Hand program event in Windows sound configuration
Question	Get current sound related to Question program event in Windows sound configuration

# Play() Method

- Play Windows-defined sounds to hint users
- Ex: play questionn sound

**SystemSounds.Question.Play();**

# SoundPlayer Class

- Create a SoundPlayer object to load sound, play sound and stop playing sound

## 1. SoundPlayer Constructor

### ① SoundPlayer()

create SoundPlayer object, then use SoundLocation property to link the full path of sound file

```
String filename = @"D:\media\XMAS.wav";  
SoundPlayer player = new SoundPlayer();  
player.SoundLocation = filename;
```

## ② **SoundPlayer(String)**

- **When creating SoundPlayer object, link to the full path of target sound file**

- **Ex:**

**String filename = @"D:\media\XMAS.wav";**

**SoundPlayer player = new SoundPlayer(filename);**

### ③ **SoundPlayer (Stream)**

- **Establish player category SoundPlayer object; meanwhile, attach .waz file to the designated file stream.**

- **Ex:**

```
String filename = @"D:\media\XMAS.wav";  
FileStream fin = new FileStream(filename,  
    FileMode.Open);  
SoundPlayer player = new SoundPlayer(fin);
```



# SoundPlayer Methods

## ① Load()

to load the designated sound file to main memory for playing

## ② Play()

to play the loaded sound file, play only once. When play the sound file, the system automatically loads the sound file if the file is not be loaded

```
player.Load();      player.Play();
```

## ③ Stop()

to stop playing sound

## ④ PlayLooping()

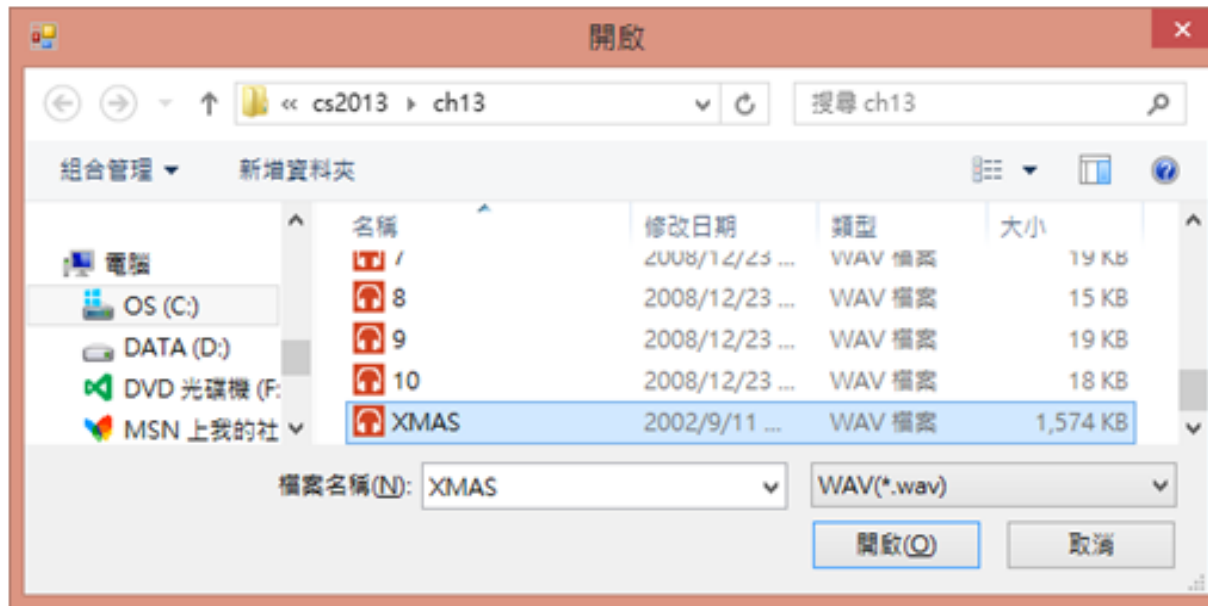
play sound repeatedly, until Stop() is executed

## Example(WavPlayer):

Design a sound player which can browse files, play the file once, play the file repeatedly, stop playing and exit. Requirements:



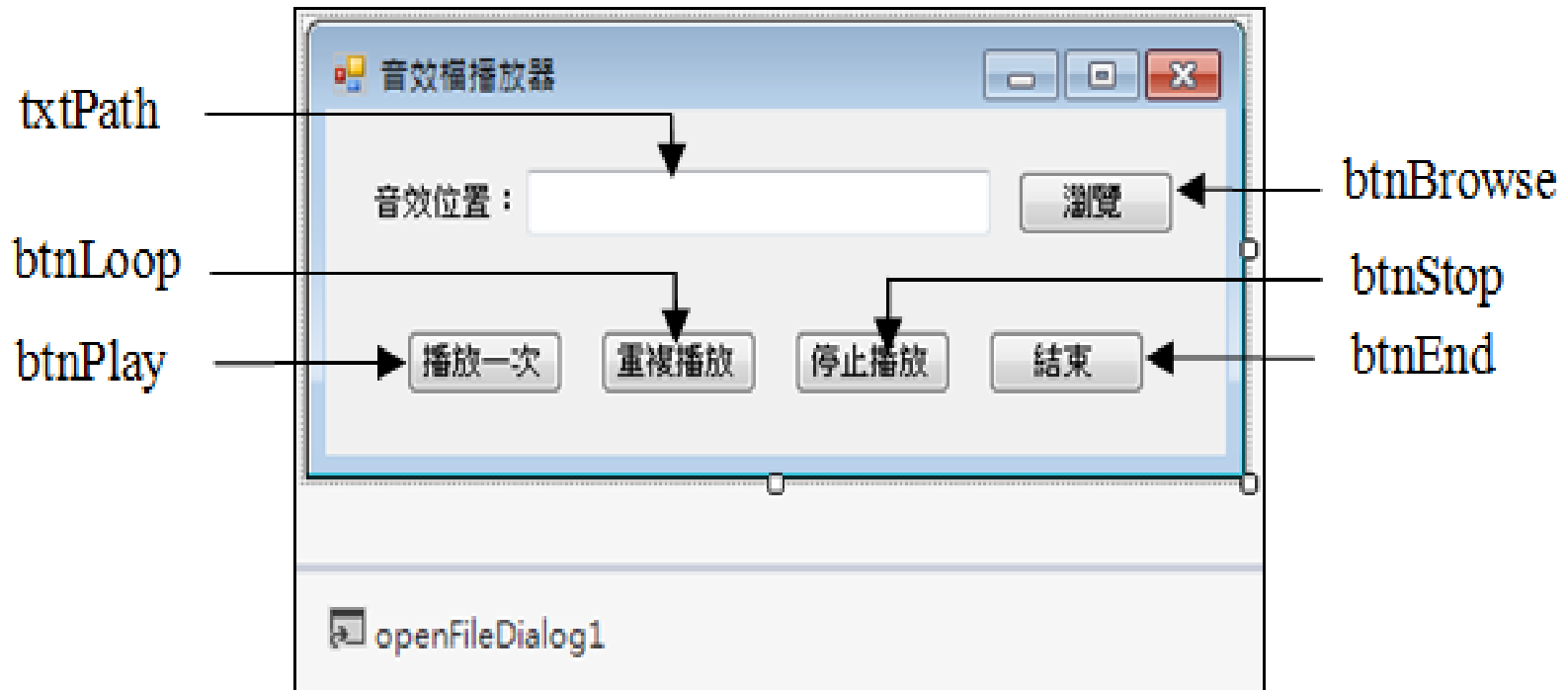
1. When the program starts, the text box used for sound file path is blank. Press “瀏覽” button to open dialog to choose the path of sound file.



2. Press “播放一次” button to play sound file once
3. Press “重複播放” button to play sound file repeatedly
4. Press “停止播放” button to stop playing
5. Press “結束” button to exit the program

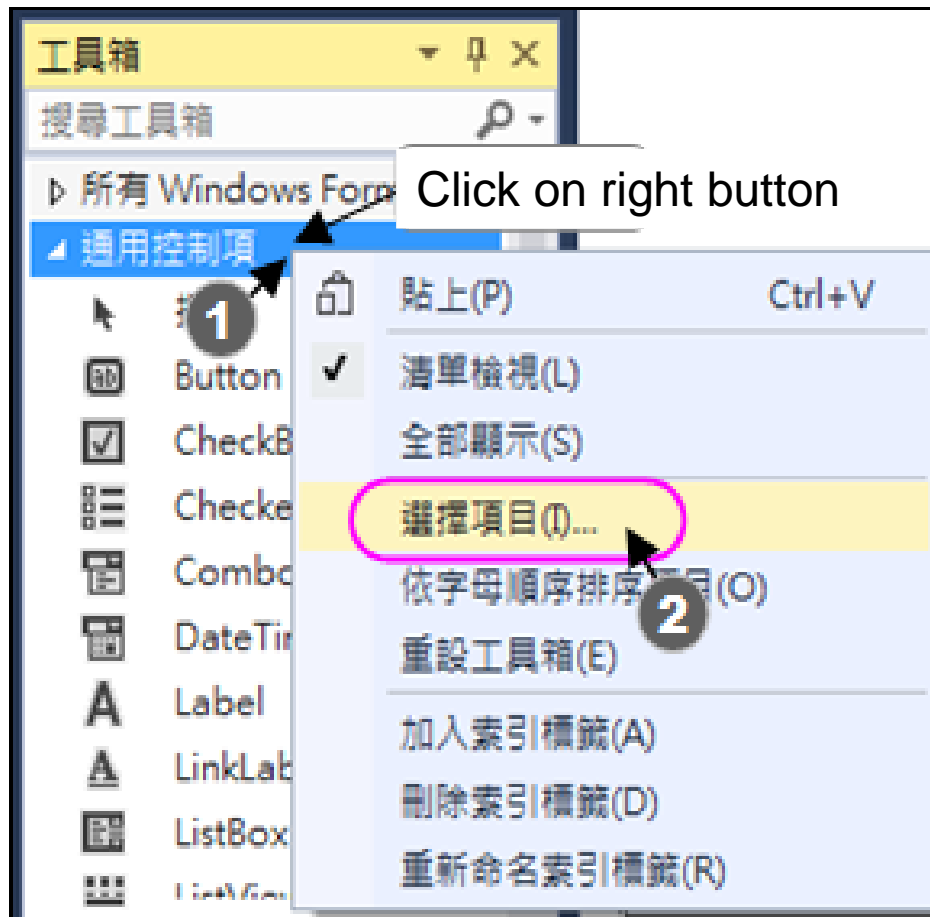


# Design User Interface

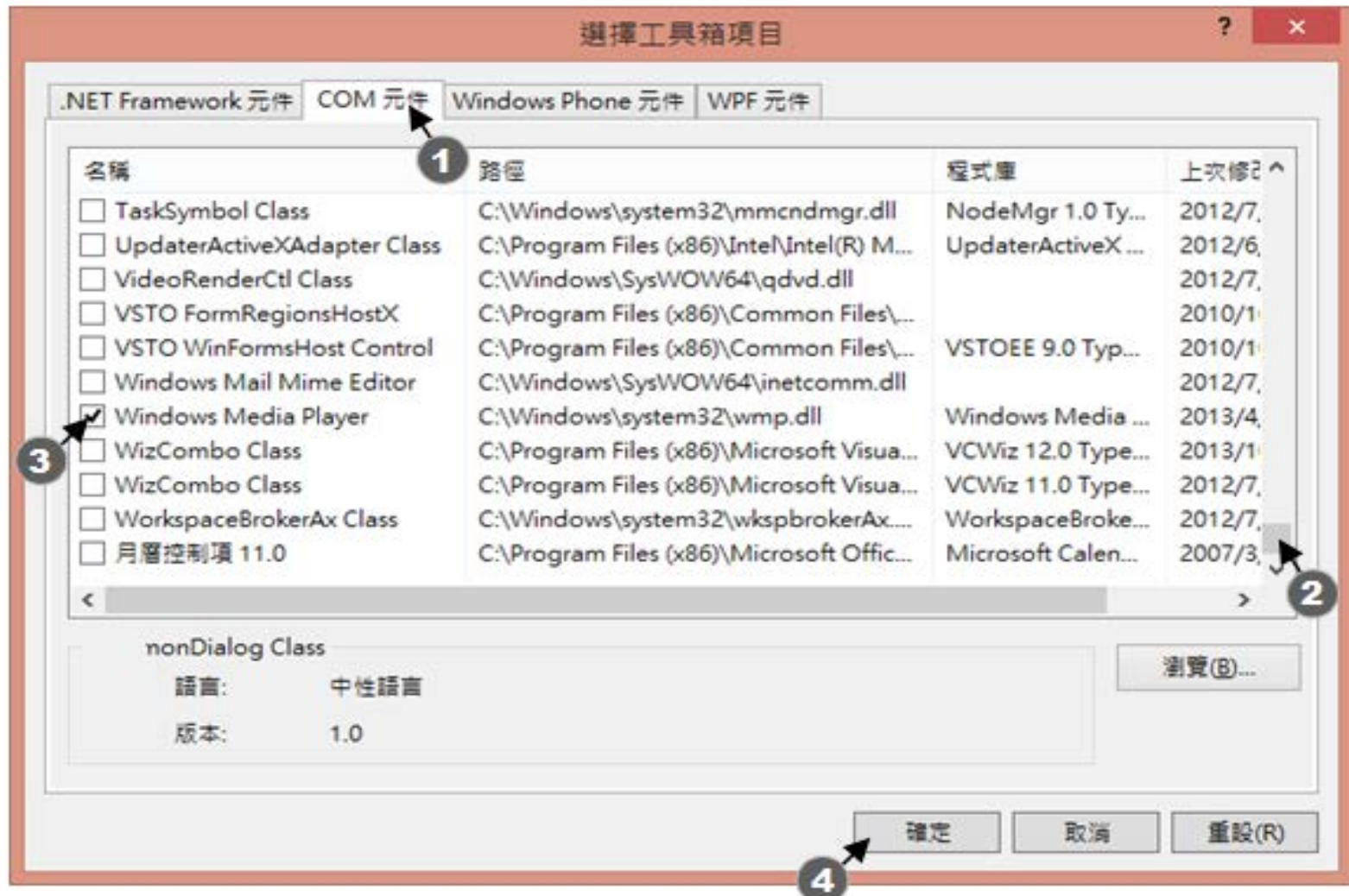


# 13-8 Multimedia Play

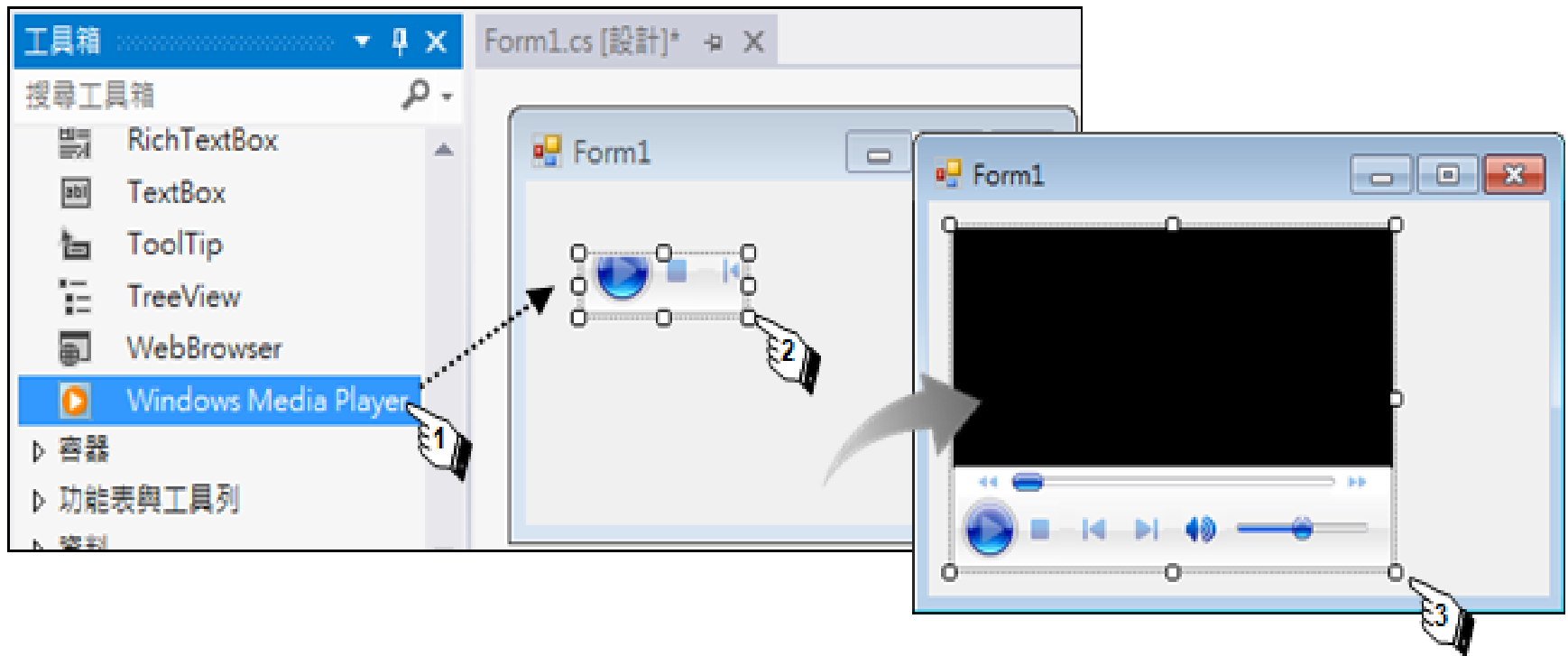
## Import Windows Media Player components



# Check on Windows Media Player component



# Create “Windows Media Player” Control Item



# Windows Media Player Properties

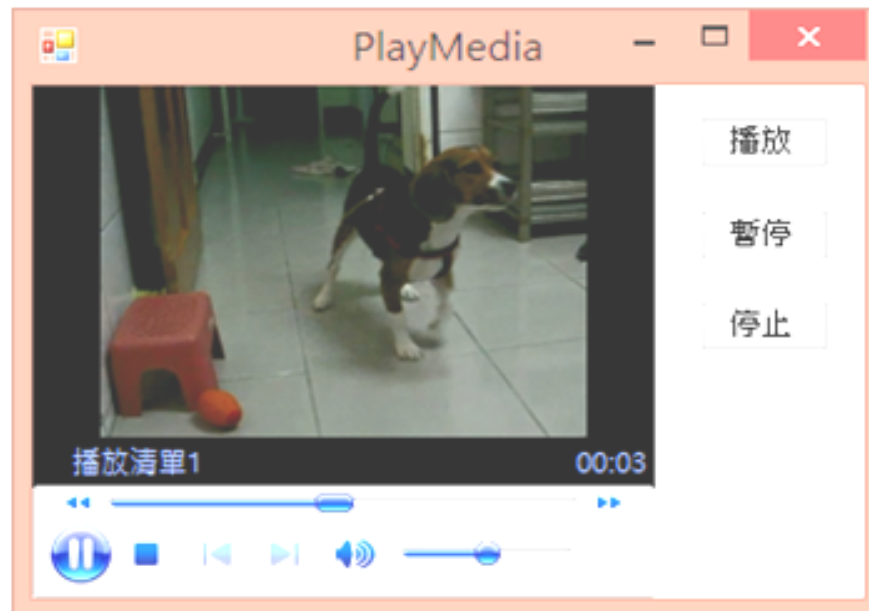
Property	Description
Name	Object name of Windows Media Player control item, default: axWindowsMediaPlayer1
URL	Get or set the full path of multimedia file to play
uiMode	User interface mode: <ol style="list-style-type: none"><li>1. full(default): full control bar, ex: status window, play, stop, mute, volume, fast forward, reverse, next, previous and so on</li><li>2. none: only the frame of video, no control bar</li><li>3. mini: only status window, play, stop, mute, volume</li><li>4. invisible: hide the frame</li></ol>
fullScreen	Display in full screen or not, default: False. The URL has to be assigned if to set fullScreen True

Property	Description
Dock	Set the relation between border of control item and border of container. Top, Bottom, Left, Right, Fill and None(default)
stretchToFit	Auto resize window, default: False
Ctlcontrols	<p>To operate control item when playing:</p> <ol style="list-style-type: none"> <li>1. Play axWindowsMediaPlayer1.Ctlcontrols.play()</li> <li>2. Pause axWindowsMediaPlayer1.Ctlcontrols.pause()</li> <li>3. Stop axWindowsMediaPlayer1.Ctlcontrols.stop()</li> <li>4. Go to previous axWindowsMediaPlayer1.Ctlcontrols.previous()</li> <li>5. Go to next axWindowsMediaPlayer1.Ctlcontrols.next()</li> </ol>

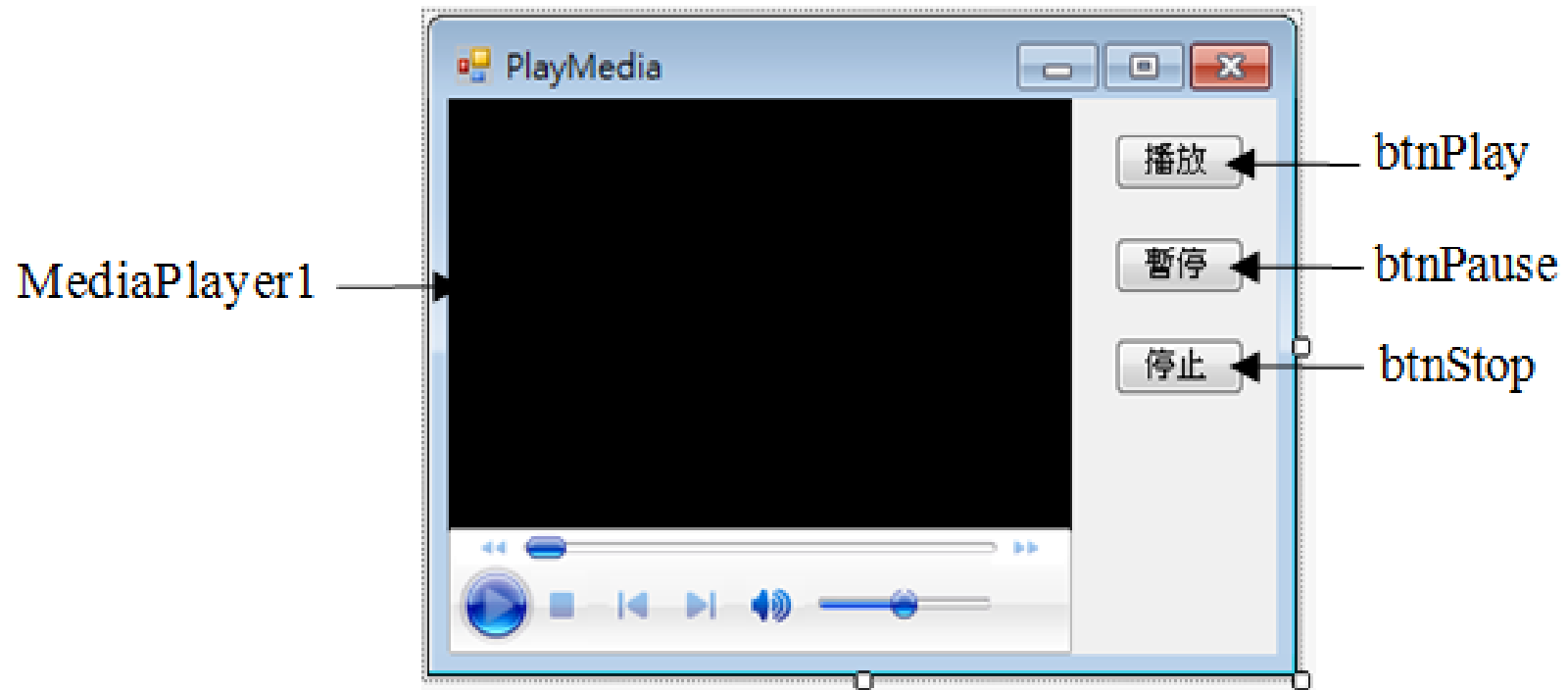
## Example(PlayMedia):

Design a multimedia player program, which includes 3 buttons called “Play”, “Pause” and “Stop”. The program loads video “C:\ch13\Dog.wmv” when starting.

## Result:



# Design User Interface







## **Practice(video)**

- **Play the attach video by media player you made.**



# The End

**Take a Break ...**