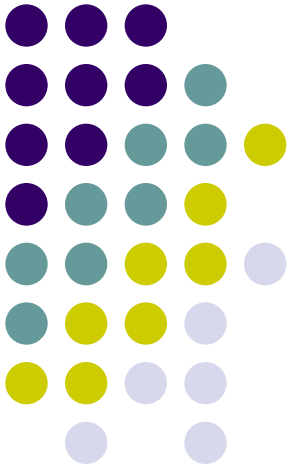
第十四章
檔案處理

認識串流

學習檔案的開啟與關閉

學習如何處理文字檔

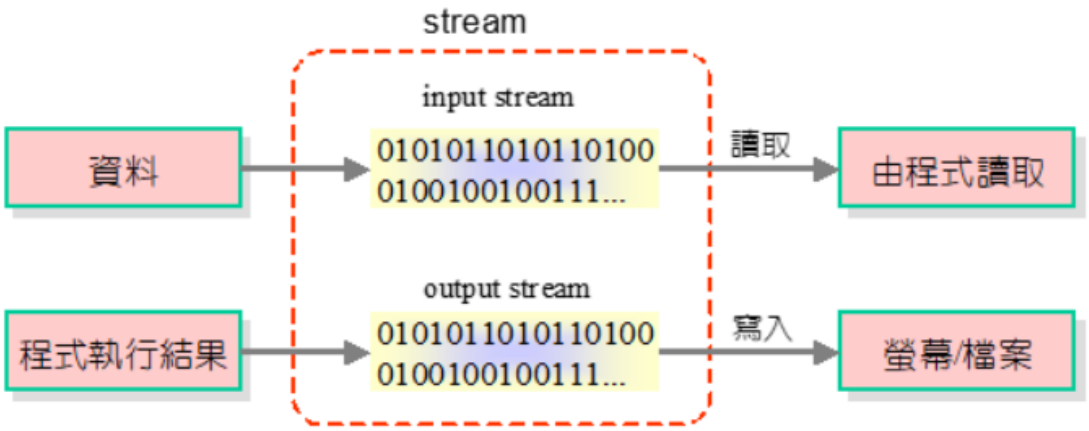
學習如何處理二進位檔

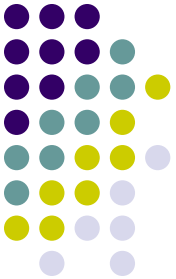


串流的認識

- 串流裡資料的組成
  - 字元（characters）
  - 位元（bits）
- 串流分為兩種
  - 「輸入串流」（input stream）
  - 「輸出串流」（output stream）

- 下圖說明串流如何做為檔案處理的橋樑





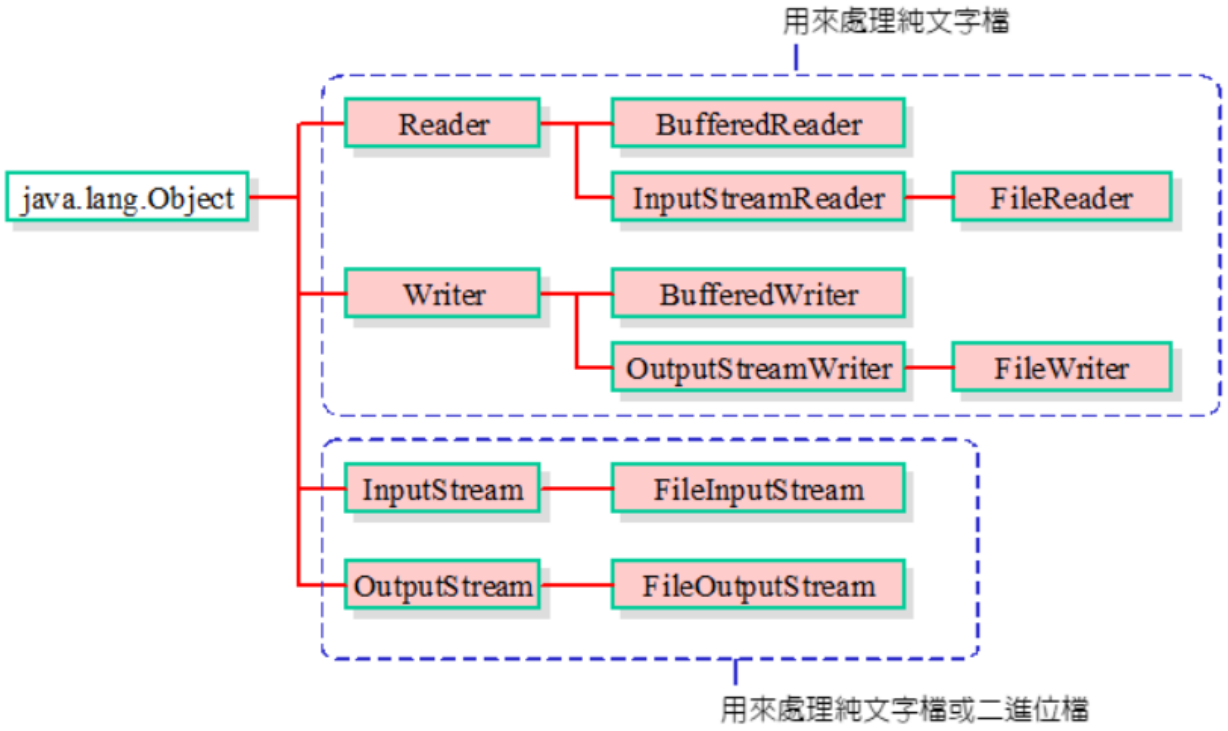
檔案的處理步驟

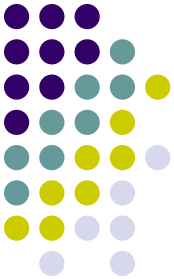
- InputStream與OutputStream類別用來處理「位元串流」（bit stream），也就是二進位檔（binary file）
- Reader與Writer類別是用來處理「字元串流」（character stream），也就是純文字檔（text file）
- 檔案處理的步驟：

  1. 透過檔案相關類別的建構元建立物件
  2. 利用物件的read() 或write() 函數讀取或寫入資料
  3. 資料處理完後用close() 函數關閉串流

檔案類別的繼承圖

- 下圖列出與檔案相關類別的繼承圖：



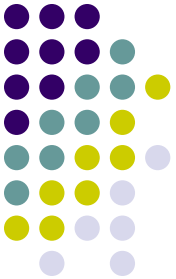


檔案處理的函數(1/2)

- 下面列出Reader類別所提供的函數

Reader 類別的函數

函數	主要功能
void close()	關閉串流
int read()	讀取串流中的一個字元
int read(char[] cbuf)	從串流讀取資料，放到字元陣列 cbuf 中，並傳回所讀取字元的總數
int read(char[] cbuf, int off, int len)	從串流讀取資料，並放到陣列 cbuf 的某個範圍（off 表示陣列索引值，len 表示讀取字元數）
long skip(long n)	跳過 n 個字元不讀取

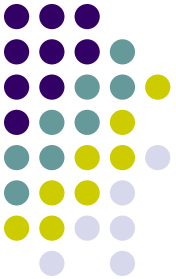


檔案處理的函數 (2/2)

- 下面列出Writer類別所提供的函數

Writer 類別的函數

函數	主要功能
void close()	關閉串流
abstract void flush()	將緩衝區的資料寫到檔案裡。注意這是抽象函數，其明確的定義是撰寫在 Writer 的子類別裡
void write(char[] cbuf)	將字元陣列輸出到串流
void write(char[] cbuf, int off, int len)	將字元陣列依指定的格式輸出到串流中（off 表示陣列索引值，len 表示寫入字元數）
void write(int c)	將單一字元 c 輸出到串流中
void write(String str)	將字串 str 輸出到串流中
void write(String str, int off, int len)	將字串 str 輸出到串流（off 表示陣列索引值，len 表示寫入字元數）



使用FileReader類別

- FileReader類別可用來讀取文字檔
- 讀取檔案步驟：

  (1) 呼叫FileReader() 建構元建立FileReader類別的物件

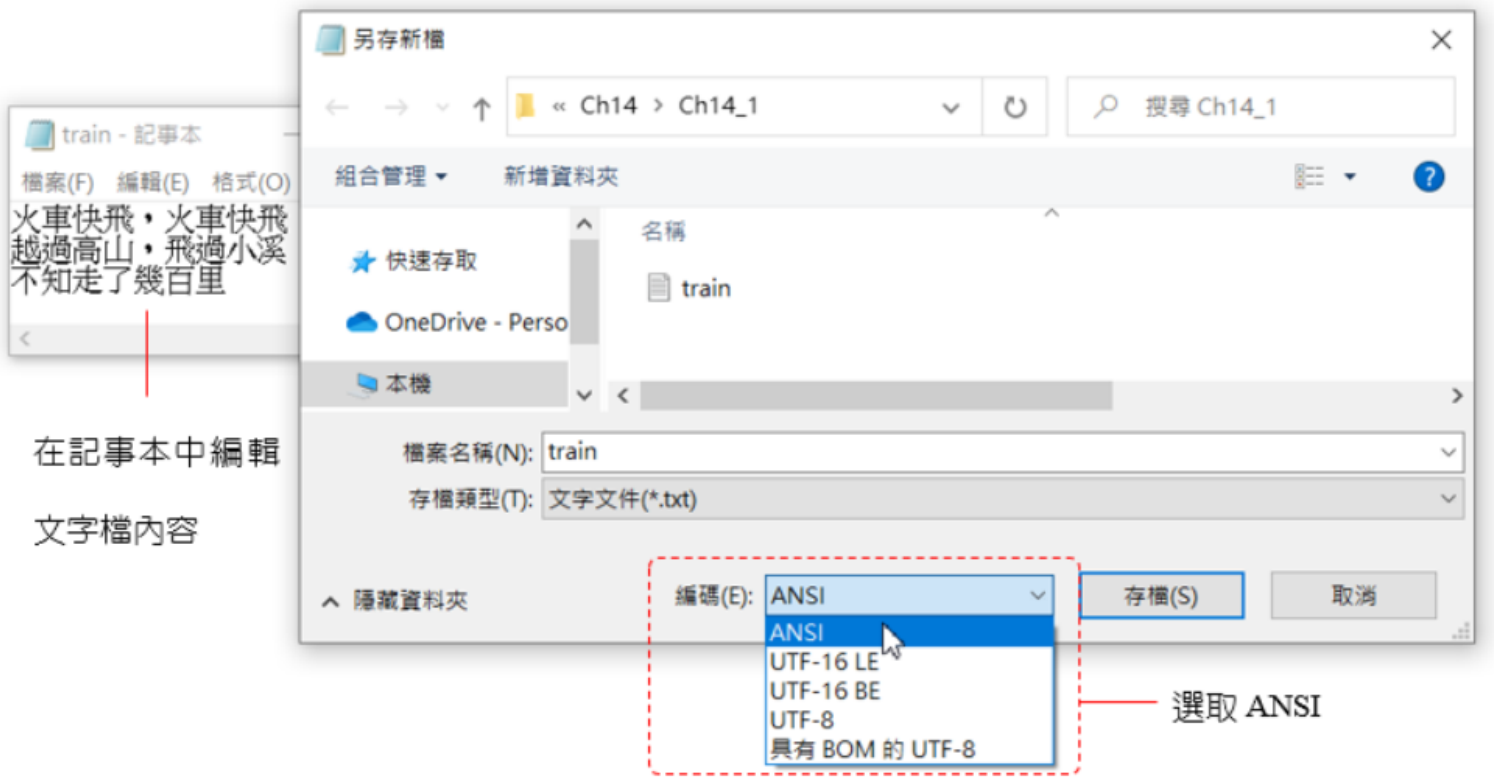
  (2) 利用此物件呼叫read() 函數來讀取檔案
- FileReader() 建構元的格式可參考下表：

FileReader 建構元

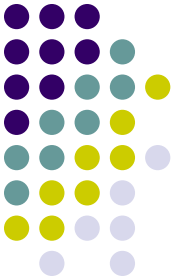
建構元	主要功能
FileReader(String name)	依檔案名稱建立一個可供讀取字元的輸入串流物件

讀取文字檔(1/2)

- 文字檔train.txt儲存時要用ANSI格式儲存：



讀取文字檔(2/2)



- 下面的範例說明如何讀取文字檔train.txt：

```
01  // Ch14_1, 使用 FileReader 類別讀取檔案
02  import java.io.*;                          // 載入 java.io 類別庫裡的所有類別
03  public class Ch14_1{
04     public static void main(String args[]) throws IOException{
05        char data[]=new char[128]; // 建立可容納 128 個字元的陣列
06        // 建立物件 fr
07        FileReader fr=new FileReader("C:\\MyJava\\Ch14\\Ch14_1\\train.txt");
08
09        int num=fr.read(data);      // 將資料讀入字元陣列 data 內
10        String str=new String(data,0,num);  // 將字元陣列轉換成字串
11        System.out.println("Characters read= "+num);
12        System.out.println(str);
13
14        fr.close();
15     }
16  }
```

必須用兩個反斜線來分隔子目錄

read() 會拋出IOException例外

執行結果：

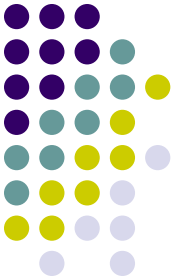
```
Characters read= 29
火車快飛，火車快飛
越過高山，飛過小溪
不知走了幾百里
```

火	車	快	飛	，	火	車	快	飛	\r	\n
1	2	3	4	5	6	7	8	9	10	11

越	過	高	山	，	飛	過	小	溪	\r	\n
12	13	14	15	16	17	18	19	20	21	22

不	知	走	了	幾	百	里
23	24	25	26	27	28	29

Java把一個中文字看成是一個字元，在Windows裡，換行字元「\r\n」是兩個字元



使用FileWriter類別

- FileWriter類別可將字元型態的資料寫入檔案
- 寫入檔案步驟：

  (1) 呼叫FileWriter() 建構元建立FileWriter類別的物件

  (2) 用此物件呼叫write() 寫入資料
- FileWriter() 建構元的格式 ：

FileWriter 建構元

建構元	主要功能
FileWriter(String filename)	依檔案名稱建立一個可供寫入字元資料的串流物件，原先的檔案會被覆蓋
FileWriter(String filename, Boolean a)	同上，但如果 a 設為 true，則會將資料附加在原先的資料後面

將資料寫入檔案

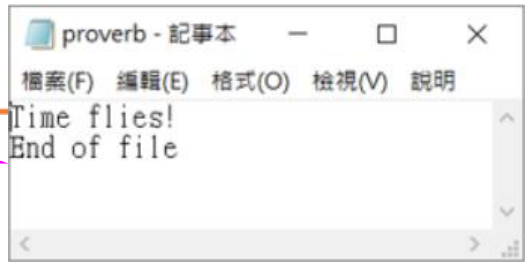
- 下面的範例以FileWriter類別將資料寫到檔案裡：

```
01  // Ch14_2, 使用 FileWriter 類別將資料寫入檔案內
02  import java.io.*;
03  public class Ch14_2{
04     public static void main(String args[]) throws IOException   {
05        FileWriter fw=new FileWriter("C:\\MyJava\\Ch14\\Ch14_2\\proverb.txt");
06        char data[]={'T','i','m','e',' ','f','l','i','e','s','!','\r','\n'};
07        String str="End of file";
08        fw.write(data);                  // 將字元陣列寫到檔案裡
09        fw.write(str);                   // 將字串寫到檔案裡
10        fw.close();
11     }
12  }
```

write() 會拋出IOException例外

執行結果：

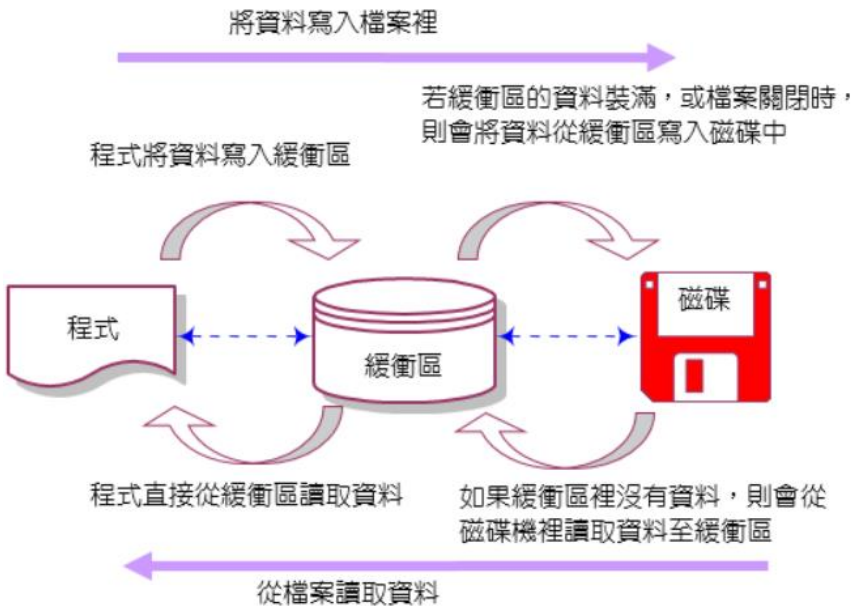
文字檔proverb.txt的內容

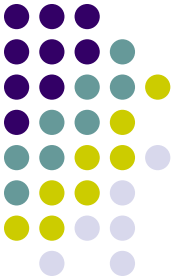


緩衝區的認識

- 有緩衝區的檔案處理方式
  - 存取時會先將資料放到緩衝區
  - 不需要一直做磁碟讀取
- 優點：
  - 增加程式執行的效率
- 缺點：
  - 會佔用一塊記憶體空間
  - 可能會因沒有關閉檔案或是系統當機而造成資料的流失

有緩衝區的檔案處理流程





使用BufferedReader類別

- 下表列出BufferedReader類別常用的建構元與函數：

BufferedReader 的建構元

建構元	主要功能
BufferedReader(Reader in)	建立緩衝區字元讀取串流
BufferedReader(Reader in, int size)	建立緩衝區字元讀取串流，並設定緩衝區大小

BufferedReader 的函數

函數	主要功能
void close()	關閉串流
int read()	讀取單一字元
int read(char [] cbuf, int off, int len)	讀取字元陣列（off 表示陣列索引值，len 表示讀取位元數）
long skip(long n)	跳過 n 個字元不讀取
String readLine()	讀取一行字串

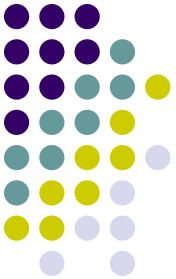
從緩衝區裡讀入資料

- 下面的範例說明如何從緩衝區讀入文字檔裡的資料：

```
01  // Ch14_3, 從緩衝區裡讀入資料
02  import java.io.*;
03  public class Ch14_3{
04     public static void main(String args[]) throws IOException   {
05        String str;
06        int count=0;
07        FileReader fr=new FileReader("C:\\MyJava\\Ch14\\Ch14_3\\number.txt");
08        BufferedReader bfr=new BufferedReader(fr);
09
10        while((str=bfr.readLine())!=null){    // 每次讀取一行，直到檔案結束
11           count++;                           // 計算讀取的行數
12           System.out.println(str);
13        }
14        System.out.println(count+" lines read");
15        fr.close();                           // 關閉檔案
16     }
17  }
```

執行結果：

```
12
34
63
14
16
56
6 lines read
```



使用BufferedWriter類別

- 下表列出BufferedWriter類別常用的建構元與函數：

BufferedWriter 的建構元

建構元	主要功能
BufferedWriter(Writer out)	建立緩衝區字元寫入串流
BufferedWriter(Writer out, int size)	建立緩衝區字元寫入串流，並設定緩衝區的大小

BufferedWriter 的函數

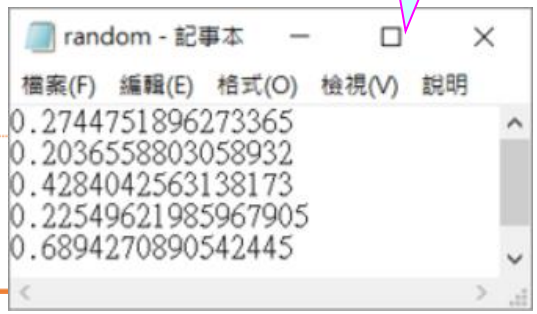
函數	主要功能
void close()	關閉串流
void flush()	寫入緩衝區內的字元到檔案裡
void newLine()	寫入換行字元
void write(int c)	寫入單一字元
void write(char[] cbuf, int off, int len)	寫入字元陣列（off 表示陣列索引值，len 表示讀取位元數）
void write(String s, int off, int len)	寫入字串（off 與 len 代表的意義同上）

將資料寫到緩衝區

- 下面的範例說明如何使用BufferedWriter類別：

```
01  // Ch14_4, 將資料寫到緩衝區內
02  import java.io.*;
03  public class Ch14_4{
04     public static void main(String[] args) throws IOException   {
05        FileWriter fw=new FileWriter("C:\\MyJava\\Ch14\\Ch14_4\\random.txt");
06        BufferedWriter bfw=new BufferedWriter(fw);
07
08        for(int i=1;i<=5;i++){
09           bfw.write(Double.toString(Math.random())); // 寫入亂數到緩衝區
10           bfw.newLine(); // 寫入換行符號
11        }
12        bfw.flush();       // 將緩衝區內的資料寫到檔案裡
13        fw.close();        // 關閉檔案
14     }
15  }
```

執行結果：



由於亂數的關係，讀者執行時裡面的數字應和本例不同

FileInputStream類別

- InputStream與OutputSteram類別可處理的資料
  - 純文字檔
  - 二進位檔（binary file）
- FileInputStream類別可處理
  - 以「位元組」為主的輸入工作
- 下表列出FileInputStream類別的建構元：

FileInputStream 的建構元

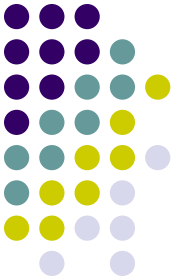
建構元	主要功能
FileInputStream (String name)	根據所給予的檔案名稱建立 FileInputStream 類別的物件

FileInputStream類別的函數

- 下表列出FileInputStream類別的函數：

FileInputStream 類別的函數

函數	主要功能
int available()	取得所讀取資料所佔的位元組數（bytes）
void close()	關閉位元組串流
long skip(long n)	在位元串流裡略過 n 個位元組的資料
int read()	從輸入串流讀取一個位元組
int read(byte[] b)	從輸入串流讀取位元組資料，並它存放到陣列 b 中
int read(byte[] b, int off, int len)	從輸入串流讀取位元組資料，並存放到指定的陣列中（off 表示陣列索引值，len 表示讀取位元組數）



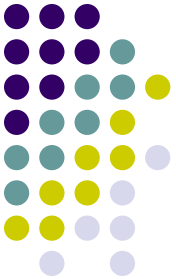
讀取檔案

- 下面的範例示範如何使用FileInputStream類別：

```
01  // Ch14_5, 利用 FileInputStream 讀取檔案
02  import java.io.*;
03  public class Ch14_5{
04     public static void main(String args[]) throws IOException   {
05        FileInputStream fi=new FileInputStream("C:\\MyJava\\train.txt");
06        System.out.println("file size="+fi.available());
07        byte ba[]=new byte[fi.available()]; // 建立 byte 陣列
08
09        fi.read(ba);  // 將讀取的內容寫到陣列 ba 裡
10        System.out.println(new String(ba)); // 印出陣列 ba 的內容
11        fi.close();
12     }
13  }
```

執行結果：

```
file size=54
火車快飛，火車快飛
越過高山，飛過小溪
不知走了幾百里
```



使用FileOutputStraem類別

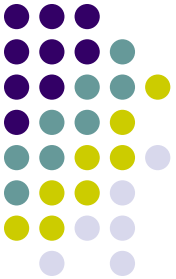
- 下表列出FileOutputStream類別的建構元與常用函數：

FileOutputStream 建構元

建構元	主要功能
FileOutputStream(String filename)	依檔案名稱建立一個可供寫入資料的輸出串流物件，原先的檔案會被覆蓋
FileOutputStream(String name, Boolean a)	同上，但如果 a 設為 true，則會將資料附加在原先的資料後面

FileOutputStream 類別的函數

函數	主要功能
void close()	關閉位元組串流
void write(byte[] b)	寫入位元組陣列 b 到串流裡
void write(byte[] b, int off, int len)	寫入位元組陣列 b 到串流裡（off 表示陣列索引值，len 表示寫入位元組數）



處理二進位檔案 (1/2)

- Ch14_6示範如何讀入一個圖檔，並將它另存新檔：

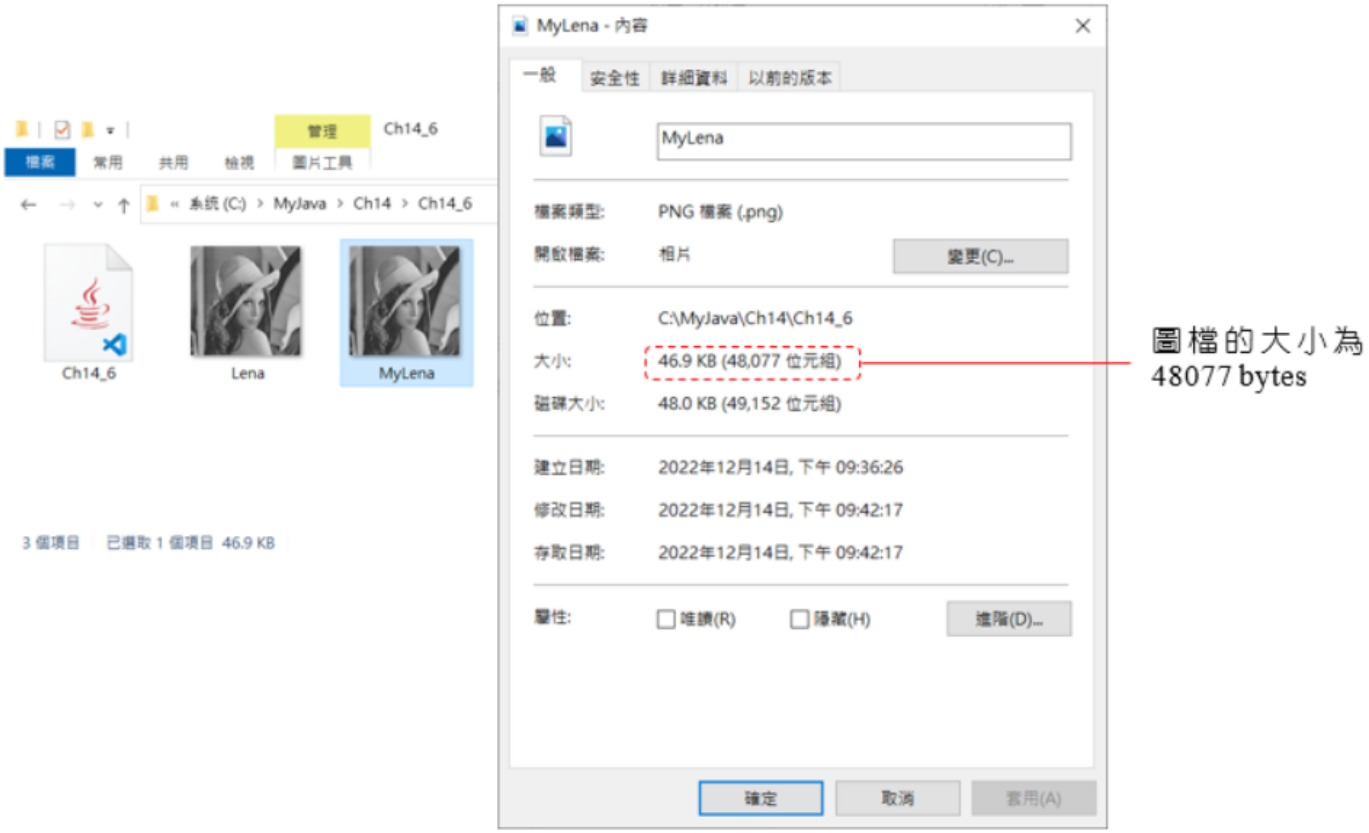
```
01 // Ch14_6, 讀入與寫入二進位檔案
02 import java.io.*;
03 public class Ch14_6{
04   public static void main(String[] args) throws IOException{
05     FileInputStream fi=new FileInputStream("C:\\MyJava\\Ch14\\Ch14_6\\Lena.png");
06     FileOutputStream fo=new FileOutputStream("C:\\MyJava\\Ch14\\Ch14_6\\MyLena.png");
07
08       System.out.println("file size="+fi.available()); // 印出檔案大小
09       byte data[]=new byte[fi.available()];  // 建立 byte 型態的陣列 data
10
11       fi.read(data);          // 將圖檔讀入 data 陣列
12       fo.write(data);         // 將 data 陣列裡的資料寫入新檔 my_lena.gif
13       System.out.println("file copied and renamed");
14       fi.close();
15       fo.close();
16    }
17 }
```

執行結果：

```
file size=48077
file copied and renamed
```

處理二進位檔案 (2/2)

- 查詢my_lena圖檔的大小：




-The End-