Java I/O

本章重點

I/O基礎概論

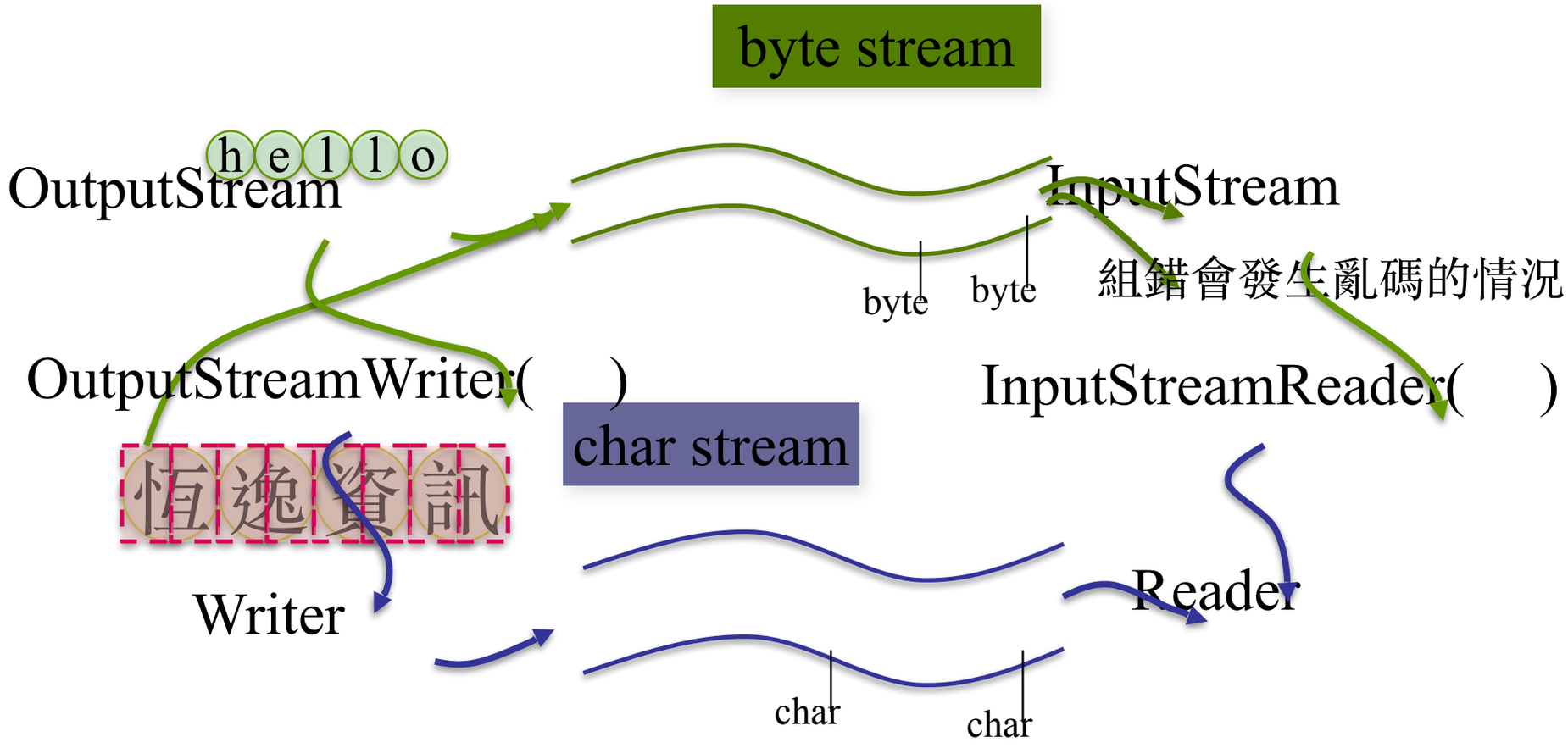
主控台(Console) I/O

BufferedReader 與 Scanner

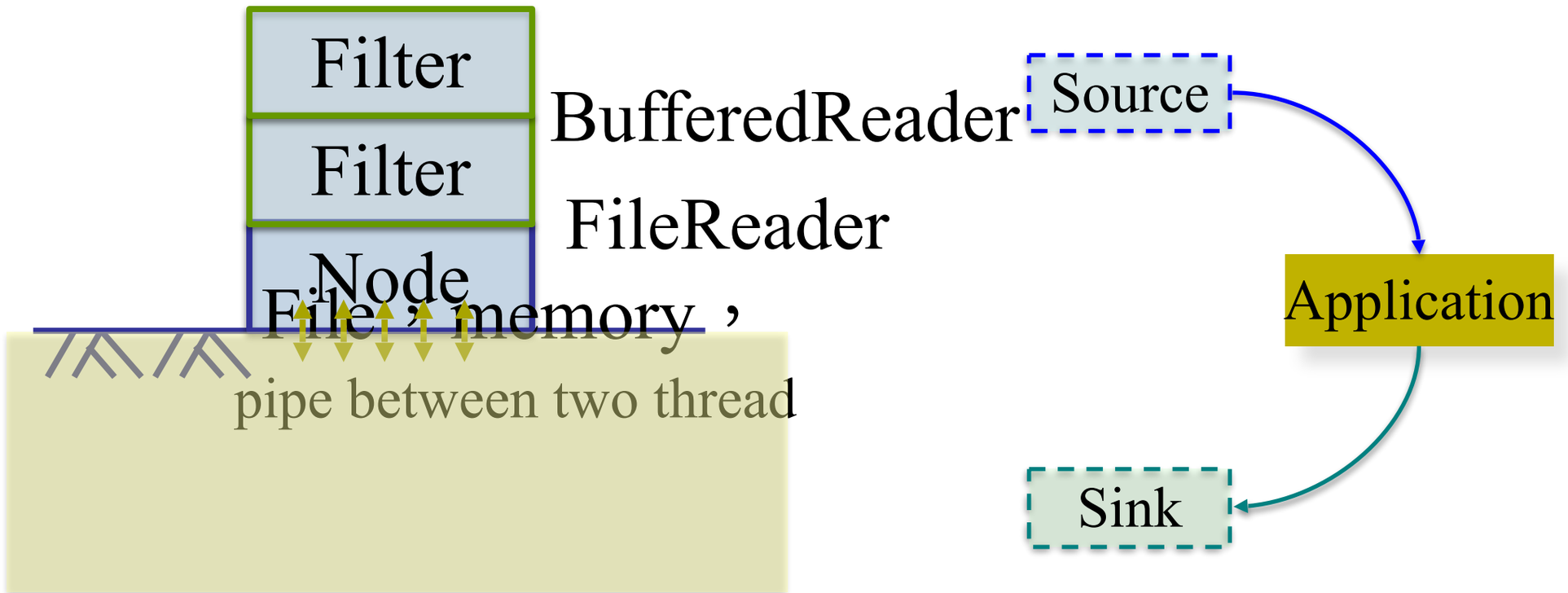
Formatter

檔案(File) I/O

I/O基礎概論：輸出入串流(I/O stream)



輸出入串流(I/O stream)



I/O基礎概論

- 一個串流(stream)可以想成是一道來自源頭(source)或流向盡頭(sink)的資料流
- 一個來源串流(source stream)開啟輸入資料流，所以又稱為輸入串流(input stream)
- 一個盡頭串流(sink stream)開啟輸出資料流，所以又稱為輸出串流(output stream)

Stream	Byte Streams	Character Streams
Sourse streams	InputStream	Reader
Sink streams	OutputStream	Writer

串流內的資料

- Java技術支援兩種類型的串流：字元(character)和位元組(byte)
  - 字元資料的輸入和輸出是由reader和writer來處理的
  - 位元組資料的輸入和輸出是由input stream和output stream來處理的
  - 一般來說，stream這個術語指的是byte stream
  - Reader和Writer的術語指的是character stream

I/O類別的基本方法

- InputStream類別
- OutputStream類別
- Reader類別
- Writer類別

加工串流(Processing Stream)

Type	Character Streams	Byte Streams
Buffering	BufferedReader BufferedWriter	BufferedInputStream BufferedOutputStream
Filtering	FilterReader FilterWriter	FilterInputStream FilterOutputStream
Converting beween bytes and chartacter	InputStreamReader OutputStreamWriter	
Performing object serialization		ObjectInputStream ObjectOutputStream

類別需實作Serializable介面

加工串流(Processing Stream)

Type	Character Streams	Byte Streams
Performing data conversion		DataInputStream DataOutputStream
Counting	LineNumberReader	LineNumberInputStream
Peeking ahead	PushbackReader	PushbackInputStream
Printing	PrintWriter	PrintStream

Console I/O

- System.out變數讓我們可以將資料輸出到系統的標準輸出端(standard output，它是一個PrintStream型別的物件
- System.in變數讓我們可以從系統的標準輸入端(standard input)讀入資料，是一個InputStream型別的物件
- System.err變數讓我們可以將錯誤資訊輸出到系統的標準錯誤輸出端(standard error)，它是一個PrintStream型別的物件

從標準輸入端讀取資料

```
import java.io.*;
public class ReadInput {
  public static void main(String args[]) {
     String s;
     BufferedReader in = null;
     try {
        InputStreamReader ir = new
  InputStreamReader(System.in);
           in = new BufferedReader(ir);
```

從標準輸入端讀取資料(續)

```
        s = in.readLine();
        System.out.println("The param1 : " + s);
        s = in.readLine();
        System.out.println("The param2 : " +
Integer.parseInt(s));

        } catch (IOException e) {
                e.printStackTrace();
        } finally{
            if(in!=null){
              try {
                in.close();
              } catch (IOException e) {
                e.printStackTrace();
              }
        }
}
```

格式化輸入

Scanner API 提供格式化輸入的功能，亦可以套用在console的輸入串流，也可以用在檔案或網路的串流

我們可以用下列方式讀取console的輸入：

```
import java.io.*;
import java.util.Scanner;
public class ScannerDemo {
  public static void main(String [] args) {
        Scanner s = new Scanner(System.in);
        String param = s.next();
        System.out.println(“The first param :" + param);
        int value = s.nextInt();
        System.out.println(“The second param :" + value);
         s.close();
  }
}
```

格式化輸出

我們可以使用格式化(formatting)的功能如下：

System.out.printf(“%s %d%n”, “Ken”, 100);

常用的格式控制碼列表如下：

Code	Description
%s	Formats the argument as a string, usually by calling the toString method on the object.
%d %o %x	Formats an integer, as a decimal, octal, or hexadecimal value
%f %g	Formats a floating point number. The %g code uses scientific notation
%n	Inserts a newline character to the string or stream.
%%	Inserts the % character to the string or stream.
%b	Formats a boolean value

檔案和檔案I/O

- java.io的套件讓我們可以做到：
  - 建立File物件
  - 操作File物件
  - 讀取或寫入檔案的串流(File streams)

File的檢測和公用程式

檔案資訊

String getName()

String getPath()

String getAbsolutePath()

String getParent()

long lastModified()

long length()

檔案修改

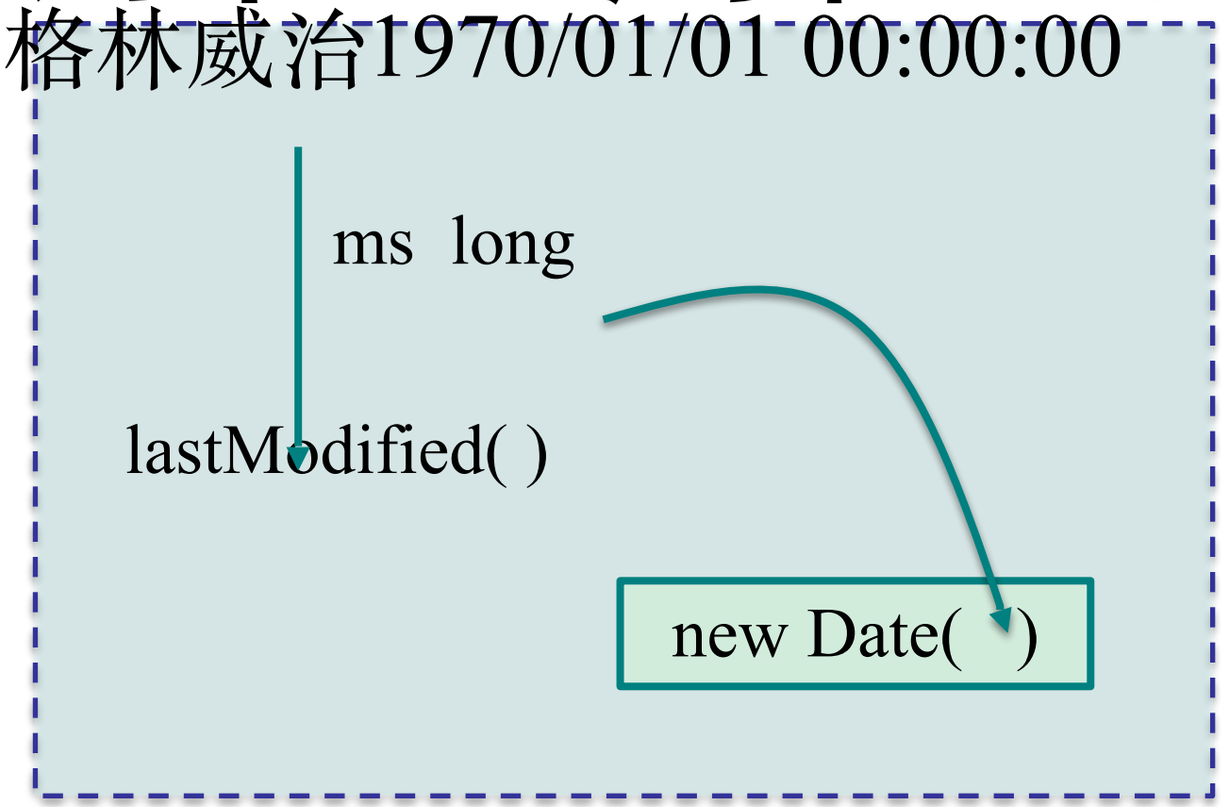
boolean renameTo(File newName)

boolean delete()

目錄公用程式

boolean mkdir()

String[] list()



File的檢測和公用程式(續)

檔案檢測：

boolean exists()

boolean canWrite()

boolean canRead()

boolean isFile()

boolean isDirectory()

boolean isAbsolute();

檔案串流I/O

- 文字檔案讀取：
  - 利用FileReader類別來讀取字元
  - 透過BufferedReader類別來使用readLine方法
- 文字檔案寫入：
  - 利用FileWriter類別來寫入字元
  - 透過PrintWriter類別來使用print和println方法

檔案串流I/O範例

```
public class ReadFile {
public static void main(String[] args) {
   String fileName = "my.txt";
   if (args.length != 0) {
       fileName = args[0];
   }
File file = new File(fileName);
BufferedReader in = null;
try {
    in = new BufferedReader(new FileReader(file));
    String s = in.readLine();
    while (s != null) {
        System.out.println("Read: " + s);
        s = in.readLine();
    }
}//續下頁
```

檔案串流I/O範例(續)

```
} catch (IOException e2) {
   e2.printStackTrace();
} finally {
   if (in != null) {
      try {
        in.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }
}
```

Lab：文字檔的複製

在這個練習中，你將學會使用I/O類別來處理文字檔的複製。

Lab: 圖檔的複製

在這個練習中，你將學會使用I/O類別來處理圖檔的複製。