

# Java程式設計進階

## Java資料輸出入處理

鄭安翔

ansel\_cheng@hotmail.com

# 課程大綱

## 1) **Java 資料輸出入處理架構**

- 資料流
- **InputStream**
- **OutputStream**
- **Reader**
- **Writer**

## 2) 節點資料流 Node Stream

## 3) 資料流串接

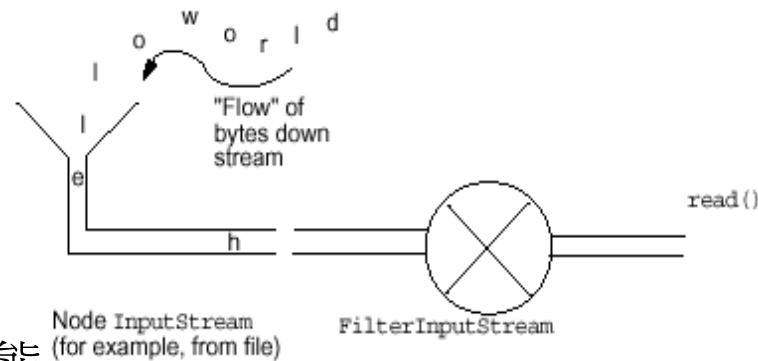
## 4) **Java檔案結構**

# 輸出入資料

## ■ 資料流 Stream :

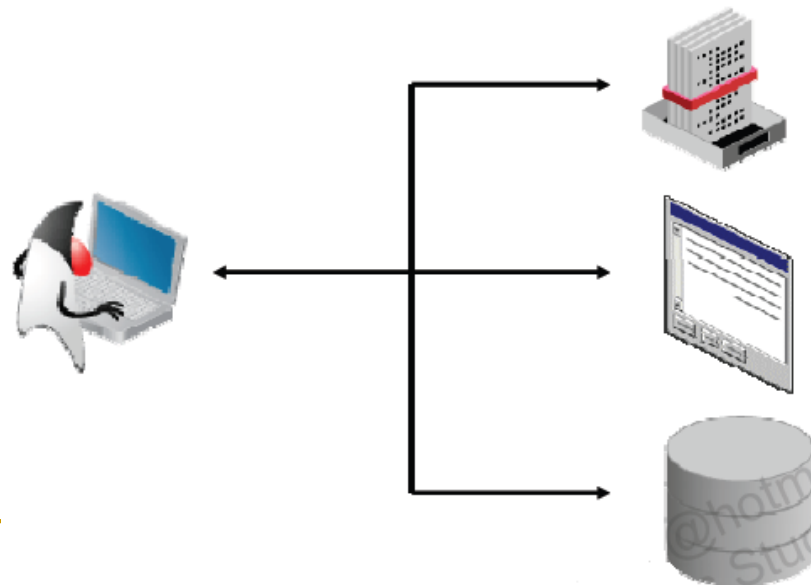
### □ 輸出入資料流動像溪流的概念

- 可分為資料源頭及目的地
- 有著固定方向。
- 不同**Stream**子類別代表不同資料型態(檔案/裝置/記憶體/其他程式)的來源及目的地



### □ 常見的輸出入資料型態

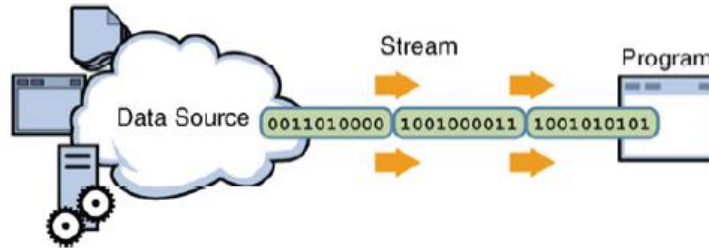
- 檔案及目錄
- 終端機 Console
- 插口端點 Socket based



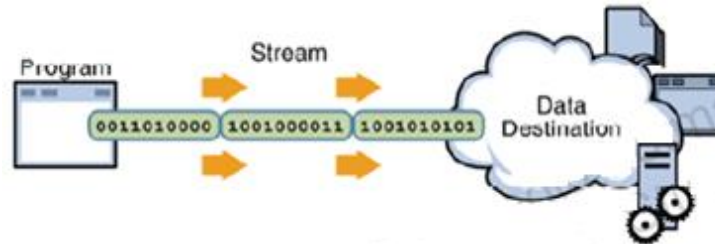
# Java 資料輸出入處理

## ■ java.io 套件 來處理資料輸出入

### □ 來源端 Source



### □ 目的端 Sink



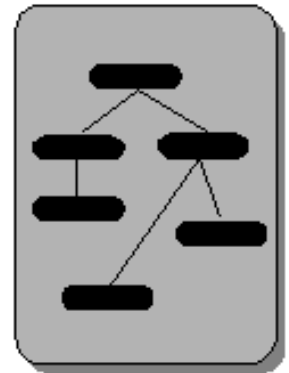
### □ 位元組資料流

- 以byte (8bits)為單位

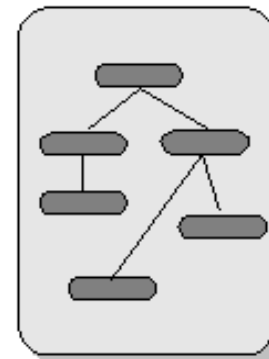
### □ Unicode 字元資料流

- 以character(16bits Unicode)為單位

Byte Streams

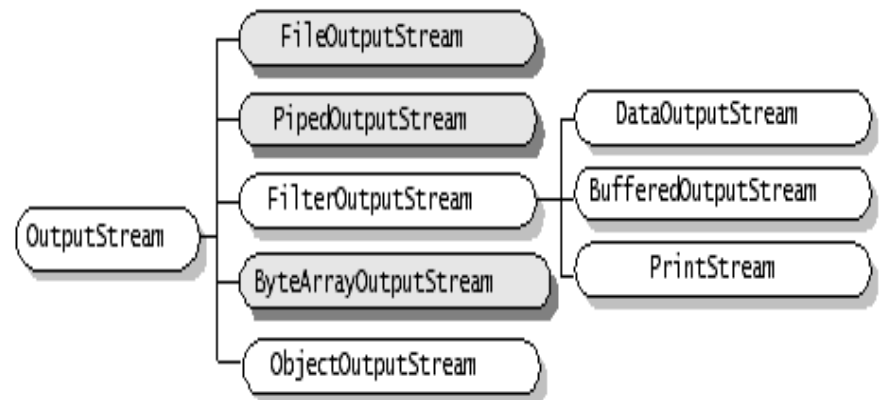
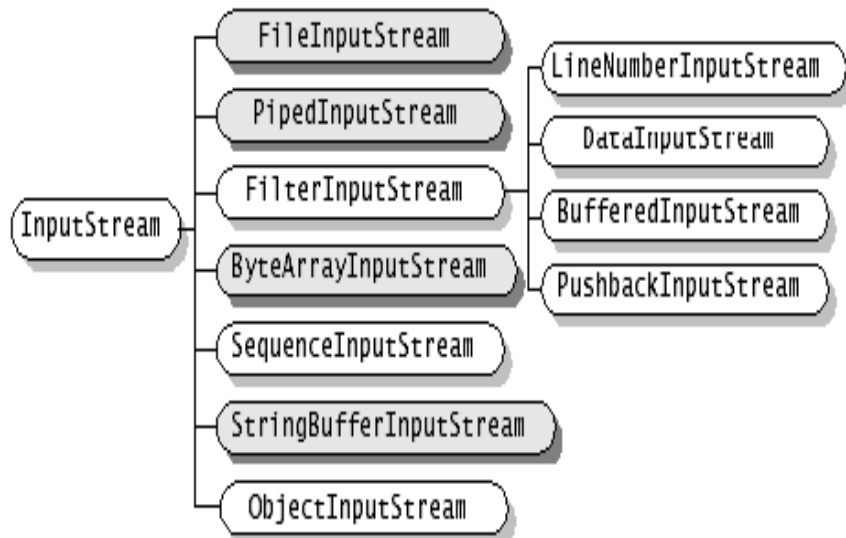


Character Streams



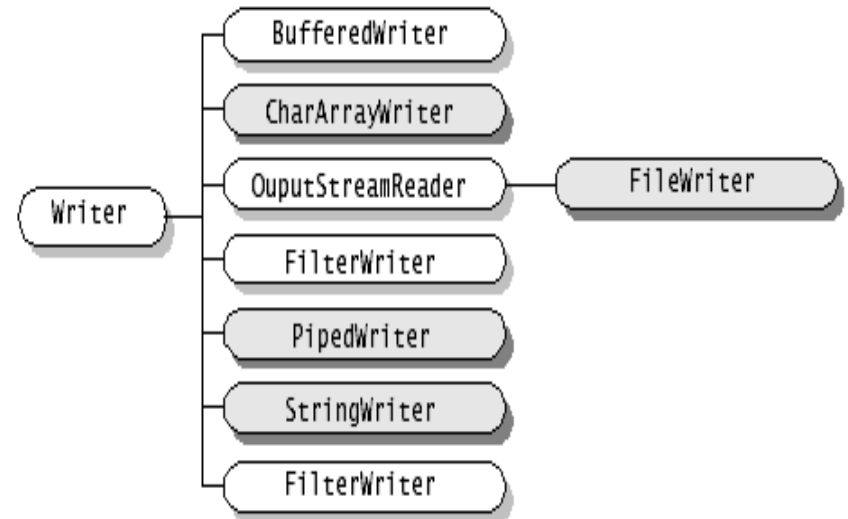
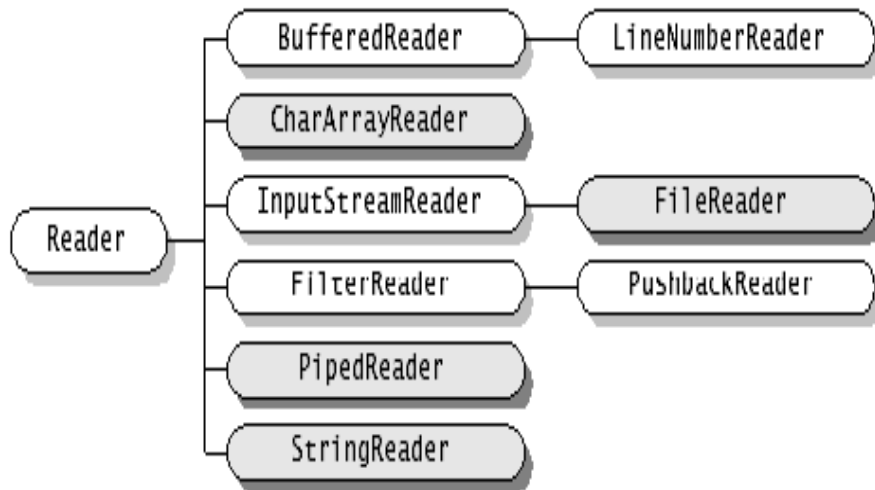
# Java IO 套件資料流類別

資料流	位元組資料流 Byte Stream	Unicode 字元資料流 Character Stream
來源資料流 Source	<b>InputStream</b>	<b>Reader</b>
目標資料流 Sink	<b>OutputStream</b>	<b>Writer</b>



# Java IO 套件資料流類別

資料流	位元組資料流 Byte Stream	Unicode 字元資料流 Character Stream
來源資料流 Source	<b>InputStream</b>	<b>Reader</b>
目標資料流 Sink	<b>OutputStream</b>	<b>Writer</b>



# InputStream 類別

建構子		說明
InputStream()		抽象類別
常用方法	傳回值	說明
available() throws IOException	int	傳回inputStream中預計可被讀取的byte數量
read() throws IOException	int	讀取inputStream物件中下一個位元組資料,並傳回其值
read(byte[] b) throws IOException	int	從inputStream物件中讀取位元組資料放入緩衝陣列中,並傳回已讀取位元組數目
read(byte[] b, int off, int len) throws IOException	int	從inputStream物件中讀取len個位元組資料,由緩衝陣列off位置開始存放,並傳回已讀取位元組數目
close() throws IOException	void	關閉資料流並釋放資料流所占用的系統資源

# OutputStream 類別

建構子		說明
OutputStream()		抽象類別
常用方法	傳回值	說明
write(int b) throws IOException	void	寫入一個指定的byte到outputStream
write(byte[] b) throws IOException	void	將緩衝陣列b中的資料寫入outputStream
write(byte[] b, int off, int len) throws IOException	void	將緩衝陣列b中off位置開始,len個位元組資料, 寫入outputStream
flush() throws IOException	void	強制緩衝區中的資料寫入outputStream
close() throws IOException	void	關閉資料流並釋放資料流所占用的系統資源



# Reader 類別

建構子		說明
Reader()		抽象類別
常用方法	傳回值	說明
ready() throws IOException	boolean	判斷這個串流是否已可被讀取
read() throws IOException	int	讀取Reader物件下一個字元資料,並傳回其值
read(char[] c) throws IOException	int	從reader物件中讀取字元資料放入緩衝陣列中,並傳回已讀取字元數目
read(char[] c, int off, int len) throws IOException	int	從reader物件中讀取len個字元資料,由緩衝陣列off位置開始存放,並傳回已讀取字元數目
close() throws IOException	void	關閉資料流並釋放資料流所占用的系統資源

# Writer 類別

建構子		說明
Writer()		抽象類別
常用方法	傳回值	說明
write(int c) throws IOException	void	寫入一個指定的字元到writer
write(char[] c) throws IOException	void	將緩衝陣列c中的資料寫入writer
write(char[] c, int off, int len) throws IOException	void	將緩衝陣列c中off位置開始,len個字元資料,寫入writer
write(String str) throws IOException	void	寫入一個指定的字串到writer
write(String str, int off, int len) throws IOException	void	將字串str中off位置開始,共len個字元資料,寫入writer
flush() throws IOException	void	強制緩衝區中的資料寫入writer
close() throws IOException	void	關閉資料流並釋放資料流所占用的系統資源

# 課程大綱

- 1) Java 資料輸出入處理架構
  - 2) 節點資料流 **Node Stream**
    - **FileInputStream**
    - **FileReader**
    - **FileOutputStream**
    - **FileWriter**
  - 3) 資料流串接
  - 4) Java檔案結構
  - 5) 主控台(Console) 輸出入
-

# 節點資料流 Node Stream

- ❑ 可直接連結到實體資源的資料流Stream物件
  - ❑ 檔案 Files
  - ❑ 記憶體 Memory ：陣列、字串物件
  - ❑ 管線 Pipe ：程序或執行緒間的資料傳遞

種類	位元組資料流	字元資料流
檔案(File)	<b>FileInputStream</b> <b>FileOutputStream</b>	<b>FileReader</b> <b>FileWriter</b>
記憶體(陣列)	<b>ByteArrayInputStream</b> <b>ByteArrayOutputStream</b>	<b>CharArrayReader</b> <b>CharArrayWriter</b>
記憶體(字串)		<b>StringReader</b> <b>StringWriter</b>
管線(Pipe)	<b>PipedInputStream</b> <b>PipedOutputStream</b>	<b>PipedReader</b> <b>PipedWriter</b>

# FileInputStream 類別

建構子	說明
<code>FileInputStream(String name)</code> throws <code>FileNotFoundException</code>	以檔名字串建立一個FileInputStream物件
<code>FileInputStream(File file)</code> throws <code>FileNotFoundException</code>	以File物件建立一個FileInputStream物件

常用方法	傳回值	說明
<code>available()</code> throws <code>IOException</code>	<code>int</code>	傳回FileInputStream中預計可被讀取的byte數量
<code>read()</code> throws <code>IOException</code>	<code>int</code>	讀取FileInputStream物件中下一個位元組資料,並傳回其值
<code>read(byte[] b)</code> throws <code>IOException</code>	<code>int</code>	從FileInputStream物件中讀取位元組資料放入緩衝陣列中,並傳回已讀取位元組數目
<code>read(byte[] b, int off, int len)</code> throws <code>IOException</code>	<code>int</code>	從FileInputStream物件中讀取len個位元組資料,由緩衝陣列off位置開始存放,並傳回已讀取位元組數目
<code>close()</code> throws <code>IOException</code>	<code>void</code>	關閉資料流並釋放資料流所占用的系統資源
<code>skip(long n)</code> throws <code>IOException</code>	<code>long</code>	跳過n個位元組的資料

# FileReader 類別

建構子	說明
<code>FileReader(String name)</code> throws <code>FileNotFoundException</code>	以檔名字串建立一個FileReader物件
<code>FileReader(File file)</code> throws <code>FileNotFoundException</code>	以File物件建立一個FileReader物件

常用方法	傳回值	說明
<code>ready()</code> throws <code>IOException</code>	<code>boolean</code>	判斷這個FileReader串流是否已可被讀取
<code>read()</code> throws <code>IOException</code>	<code>int</code>	讀取FileReader物件下一個字元資料,並傳回其值
<code>read(char[] c)</code> throws <code>IOException</code>	<code>int</code>	從FileReader物件中讀取字元資料放入緩衝陣列中,並傳回已讀取字元數目
<code>read(char[] c, int off, int len)</code> throws <code>IOException</code>	<code>int</code>	從FileReader物件中讀取len個字元資料,由緩衝陣列off位置開始存放,並傳回已讀取字元數目
<code>close()</code> throws <code>IOException</code>	<code>void</code>	關閉資料流並釋放資料流所占用的系統資源

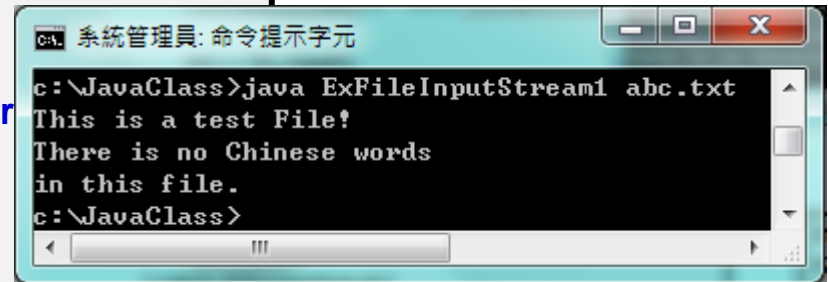
# 資料流讀取及寫入資料演算法

讀取	寫出
建立資料流 <b>while</b> (還有資料) { 讀取資料 }	建立資料流 <b>while</b> (還有資料){ 寫出資料 }
關閉資料流	強制寫入緩衝區 關閉資料流

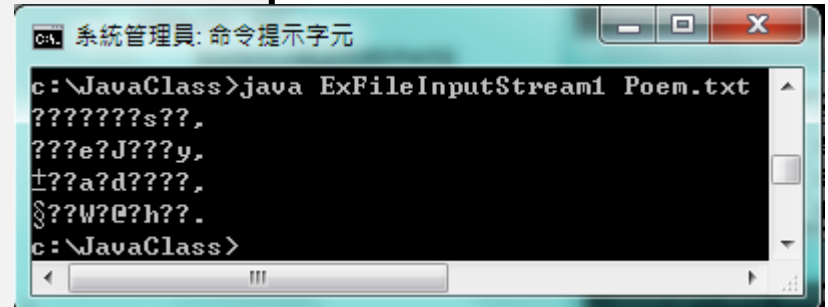
# FileInputStream 範例

```
01 import java.io.*;
02
03 public class ExFileInputStream1 {
04     public static void main(String argv[]) {
05         try {
06             FileInputStream fin = new FileInputStream(
07
08                 int i = fin.read();
09                 while (i != -1) {
10                     System.out.print((char)i);
11                     i = fin.read();
12                 }
13
14                 fin.close();
15             } catch (IOException e) {
16                 System.err.println(e);
17             }
18         }
19     }
```

01	This is a test File!
02	There is no Chinese words
03	in this file.



01	白日依山盡,
02	黃河入海流,
03	欲窮千里目,
04	更上一層樓.

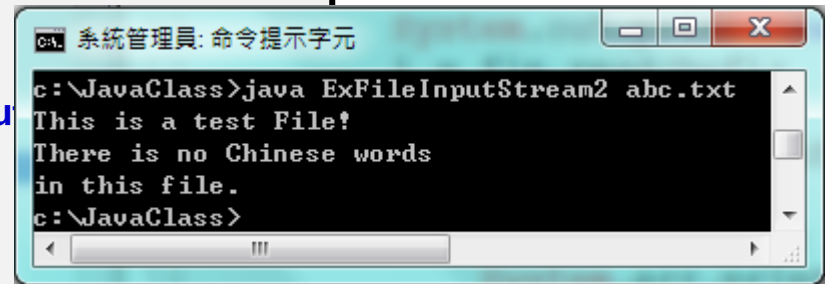




# FileInputStream 範例

```
01 import java.io.*;
02
03 public class ExFileInputStream2 {
04     public static void main(String argv[]) {
05         try {
06             FileInputStream fin = new FileInputStream(
07
08                 byte buf[ ] = new byte[16];
09
10                 int i = fin.read(buf);
11                 while (i != -1) {
12                     System.out.print(new String(buf));
13                     i = fin.read(buf);
14                 }
15                 System.out.print(new String(buf, 0, i));
16                 fin.close();
17             } catch (IOException e) {
18                 System.err.println(e);
19             }
20         }
21     }
```

```
01 This is a test File!
02 There is no Chinese words
03 in this file.
```



```
01 白日依山盡，
02 黃河入海流，
03 欲窮千里目，
04 更上一層樓。
```



# FileReader 範例

```
01 import java.io.*;
02
03 public class ExFileReader {
04     public static void main(String argv[]) {
05         try {
06             FileReader fin = new FileReader(
07
08                 int i = fin.read();
09                 while (i != -1) {
10                     System.out.print((char)i);
11                     i = fin.read();
12                 }
13                 fin.close();
14             } catch (IOException e) {
15                 System.err.println(e);
16             }
17         }
18     }
```

```
01 This is a test File!
02 There is no Chinese words
03 in this file.
```



```
系統管理員: 命令提示字元
c:\JavaClass>java ExFileReader abc.txt
This is a test File!
There is no Chinese words
in this file.
c:\JavaClass>
```

```
01 白日依山盡,
02 黃河入海流,
03 欲窮千里目,
04 更上一層樓.
```



```
系統管理員: 命令提示字元
c:\JavaClass>java ExFileReader Poem.txt
白日依山盡,
黃河入海流,
欲窮千里目,
更上一層樓.
c:\JavaClass>
```

# FileOutputStream 類別

建構子	說明
<code>FileOutputStream(String name)</code>	以檔名字串建立一個FileOutputStream物件, 預設為取代模式
<code>FileOutputStream(String name, boolean append)</code>	以檔名字串建立一個FileOutputStream物件, <code>boolean</code> 值指定是否為附加模式

常用方法	傳回值	說明
<code>write(int b) throws IOException</code>	<code>void</code>	寫入一個指定的byte到outputStream
<code>write(byte[] b) throws IOException</code>	<code>void</code>	將緩衝陣列b中的資料寫入outputStream
<code>write(byte[] b, int off, int len) throws IOException</code>	<code>void</code>	將緩衝陣列b中off位置開始,len個位元組資料, 寫入outputStream
<code>flush() throws IOException</code>	<code>void</code>	強制緩衝區中的資料寫入outputStream
<code>close() throws IOException</code>	<code>void</code>	關閉資料流並釋放資料流所占用的系統資源

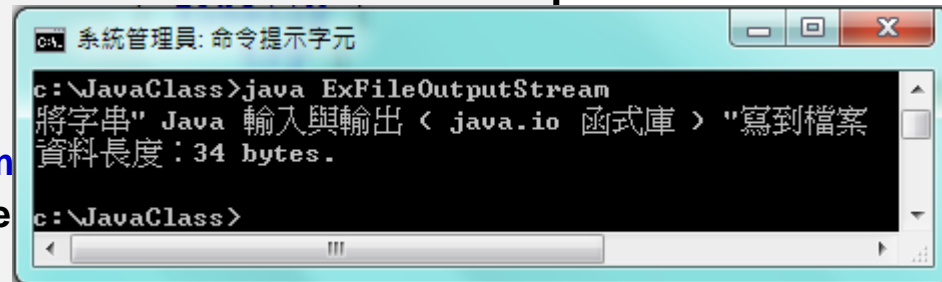
# FileWriter 類別

建構子	說明
FileWriter(String fileName) throws IOException	以檔名字串建立一個FileWriter物件, 預設為取代模式
FileWriter(String fileName, boolean append) throws IOException	以檔名字串建立一個FileWriter物件, boolean值指定是否為附加模式

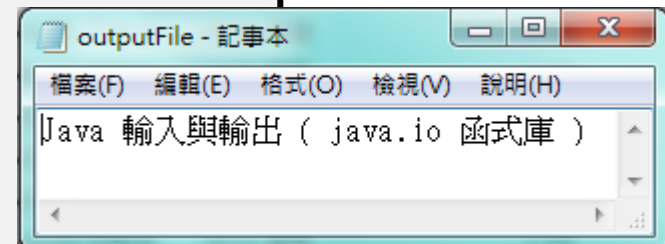
常用方法	傳回值	說明
write(int c) throws IOException	void	寫入一個指定的字元到writer
write(char[] c) throws IOException	void	將緩衝陣列c中的資料寫入writer
write(char[] c, int off, int len) throws IOException	void	將緩衝陣列c中off位置開始,len個字元資料,寫入writer
write(String str) throws IOException	void	寫入一個指定的字串到writer
write(String str, int off, int len) throws IOException	void	將字串str中off位置開始,共len個字元資料,寫入writer
flush() throws IOException	void	強制緩衝區中的資料寫入writer
close() throws IOException	void	關閉資料流並釋放資料流所占用的系統資源

# FileOutputStream 範例

```
01 import java.io.*;
02 public class ExFileOutputStream {
03     public static void main(String[] args) {
04         String s = "Java 輸入與輸出 ( java.io 函式庫 )";
05         byte[ ] data = s.getBytes();
06         System.out.println("將字串\" " + s + "\"寫到檔案");
07         System.out.println("資料長度 : " + data.length + " bytes.");
08
09         FileOutputStream fos = null;
10         try {
11             fos = new FileOutputStream("outputFile.txt");
12             fos.write(data); //直接將 byte
13         } catch(IOException e){
14         } finally {
15             try {
16                 fos.close(); // 關閉檔案
17             } catch(IOException e){
18             }
19         }
20     }
21 }
```



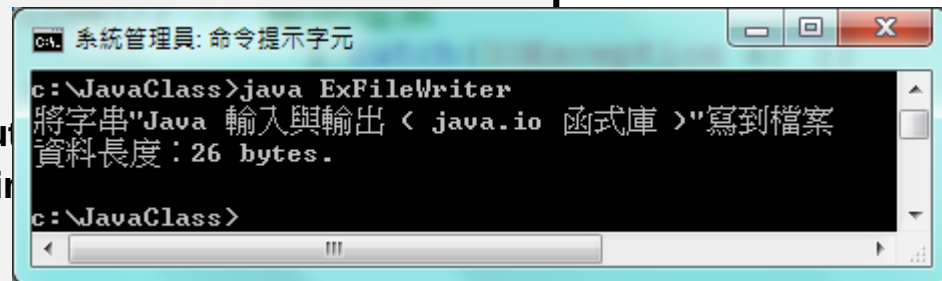
```
C:\JavaClass>java ExFileOutputStream
將字串" Java 輸入與輸出 < java.io 函式庫 > "寫到檔案
資料長度 : 34 bytes.
C:\JavaClass>
```



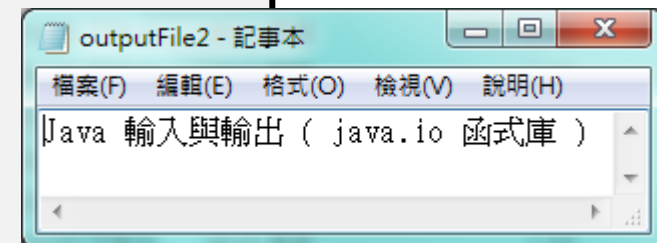
```
outputFile - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
Java 輸入與輸出 ( java.io 函式庫 )
```

# FileWriter 範例

```
01 import java.io.*;
02 public class ExFileWriter {
03     public static void main(String[] args) throws Exception {
04         String data = "Java 輸入與輸出 ( java.io 函式庫 )";
05         System.out.println("將字串\" + data + "\"寫到檔案");
06         System.out.println("資料長度 : " + data.length() + " bytes.");
07         FileWriter fw = null;
08         try {
09             fw = new FileWriter("outputFile2.txt");
10             fw.write(data); //直接將 String 寫到檔案
11         } catch(IOException e) {
12         } finally {
13             try {
14                 fw.close(); // 關閉檔案
15             } catch(IOException e) {}
16         }
17     }
18 }
```



```
系統管理員: 命令提示字元
c:\JavaClass>java ExFileWriter
將字串"Java 輸入與輸出 ( java.io 函式庫 )"寫到檔案
資料長度 : 26 bytes.
c:\JavaClass>
```



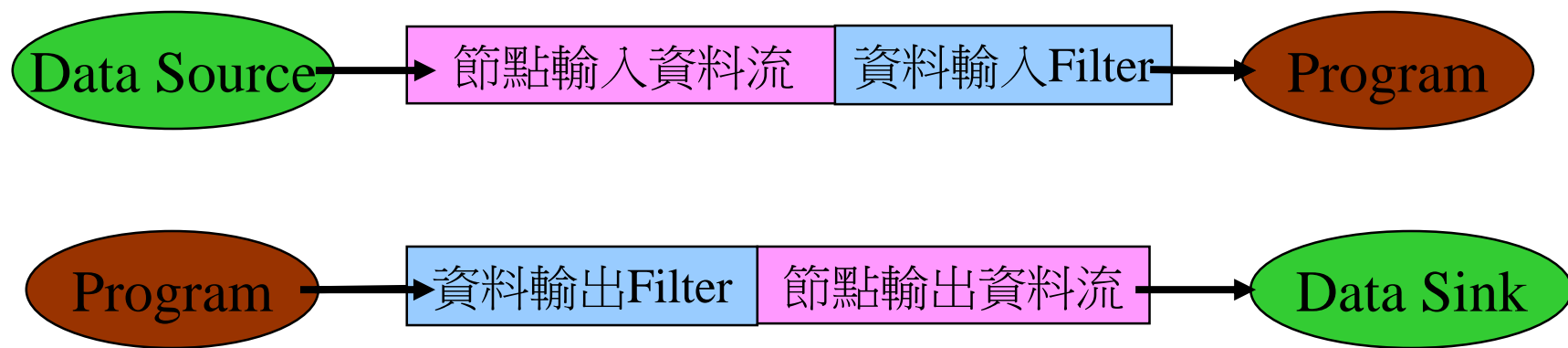
```
outputFile2 - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
Java 輸入與輸出 ( java.io 函式庫 )
```

# 課程大綱

- 1) Java 資料輸出入處理架構
- 2) 節點資料流 Node Stream
- 3) 資料流串接
  - 處理資料流 **Processing Stream**
  - 特定資料型態處理
    - **DataInputStream & DataOutputStream**
  - 紀錄行號 **LineNumberReader**
  - 物件序列化 **Serialization**
    - **Serializable interface**
    - **ObjectOutputStream & ObjectInputStream**
- 4) Java檔案結構

# 資料流串接

- 將節點資料流串接特殊的處理資料流,以便使用特定的方法來存取資料
- 處理資料流 **Processing Stream**

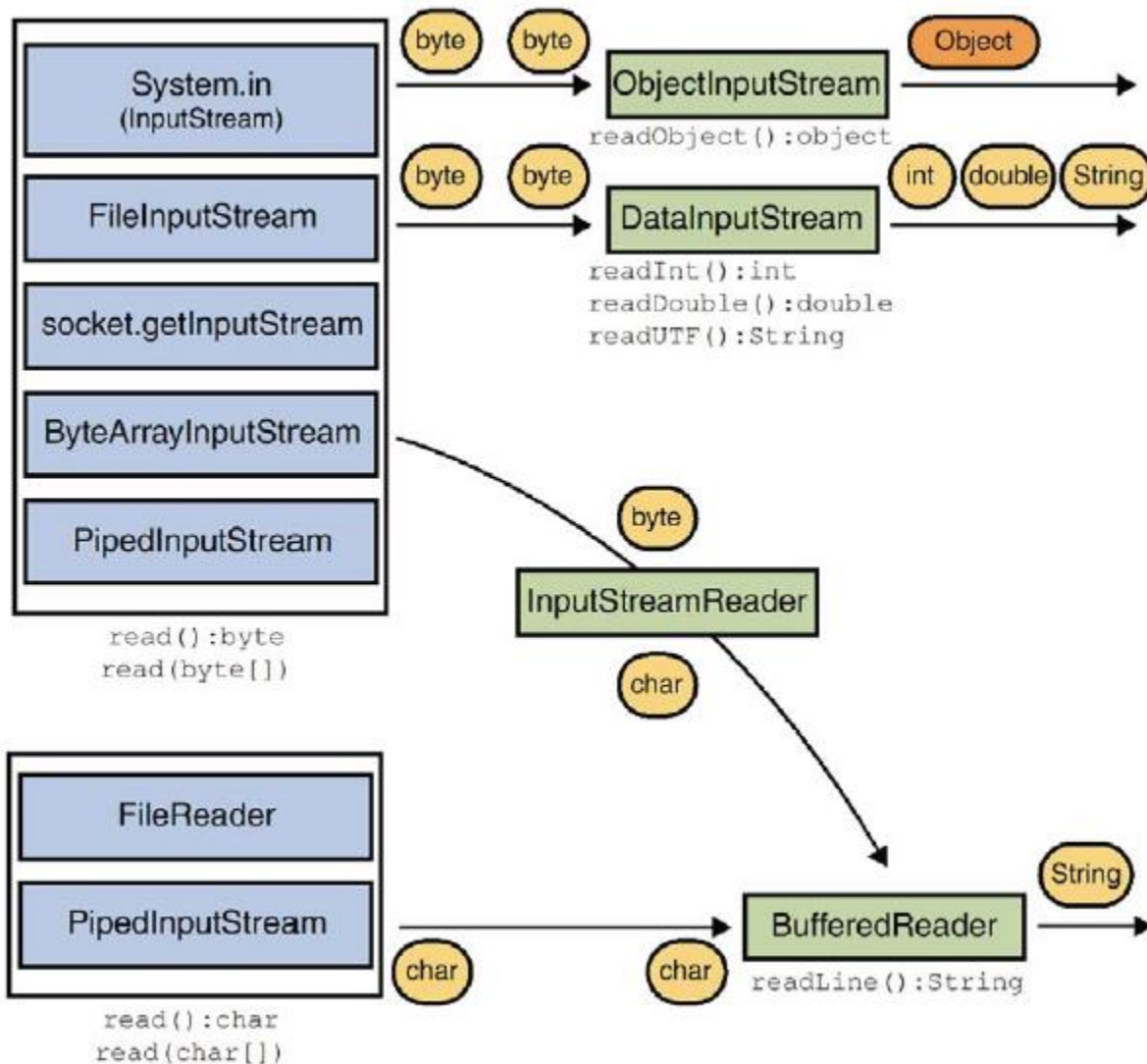




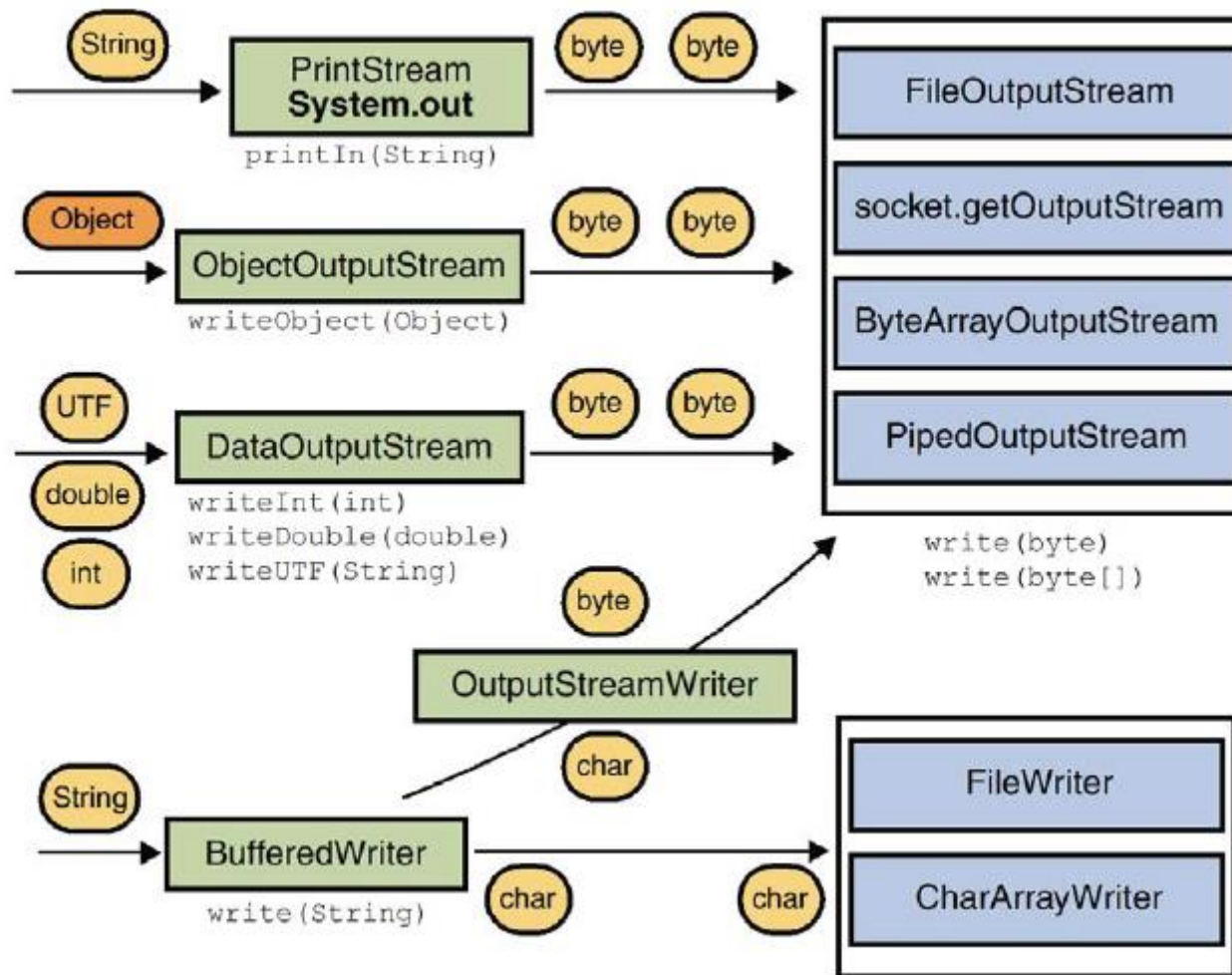
# 處理資料流 Processing Stream 類別

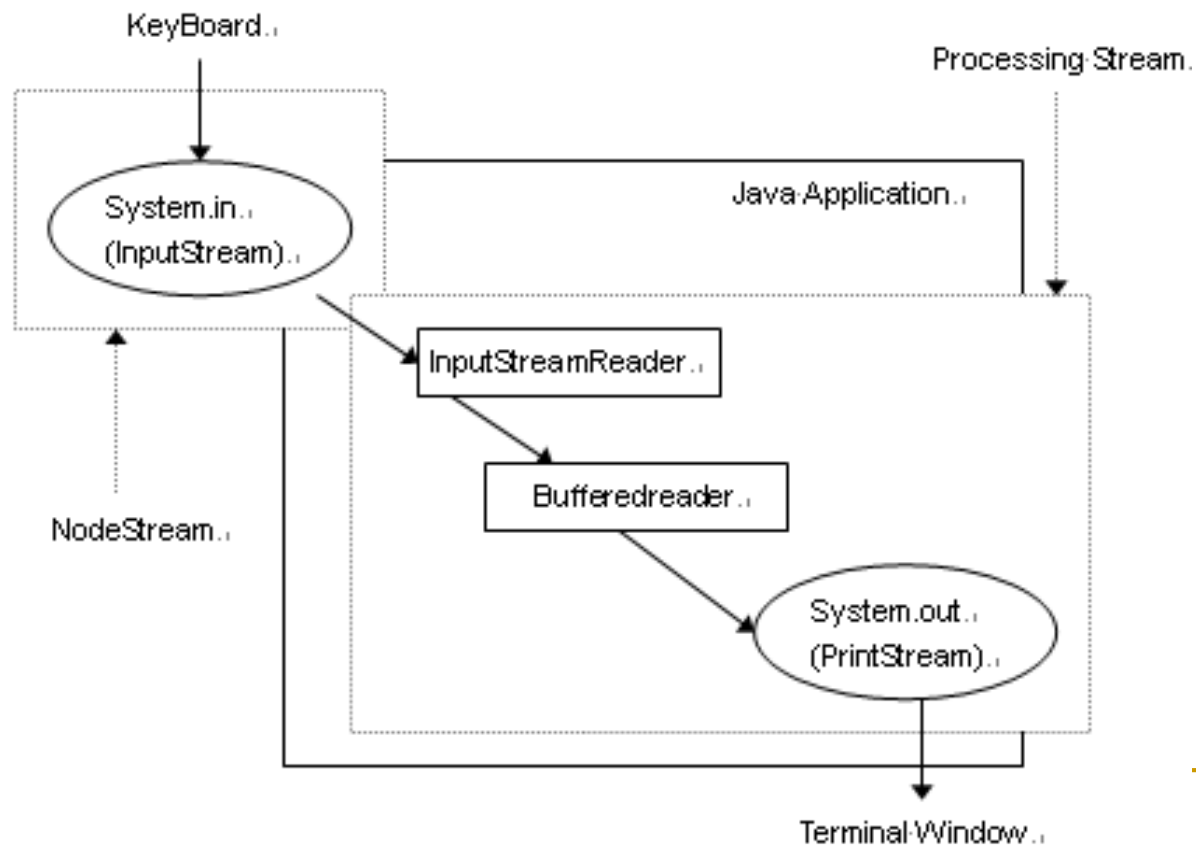
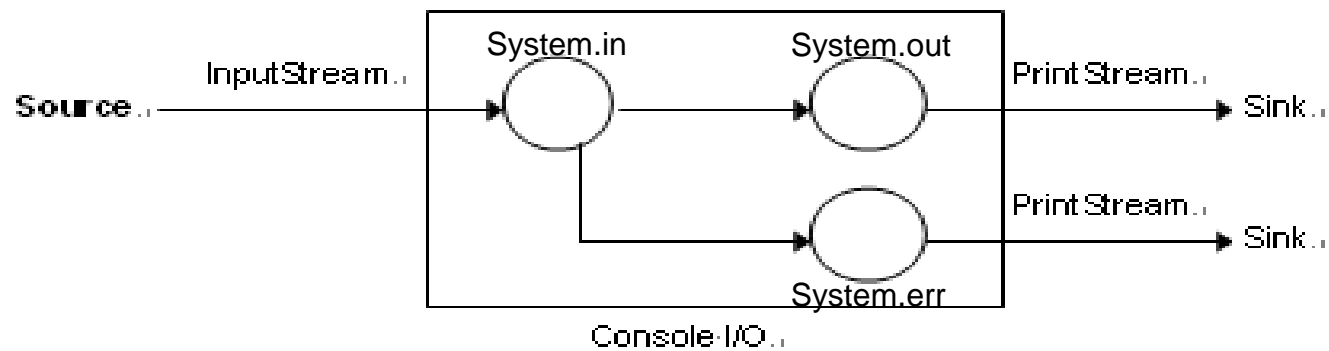
種類	位元組	字元
緩衝(Buffer)	<b>BufferedInputStream</b> <b>BufferedOutputStream</b>	<b>BufferedReader</b> <b>BufferedWriter</b>
計錄行數	<b>LineNumberInputStream</b>	<b>LineNumberReader</b>
資料預覽	<b>PushbackInputStream</b>	<b>PushbackReader</b>
列印輸出	<b>PrintStream</b>	<b>PrintWriter</b>
字元與位元組轉換		<b>InputStreamReader</b> <b>OutputStreamWriter</b>
物件序列化	<b>ObjectInputStream</b> <b>ObjectOutputStream</b>	
特定資料型態存取	<b>DataInputStream</b> <b>DataOutputStream</b>	

# 輸入資料流串接



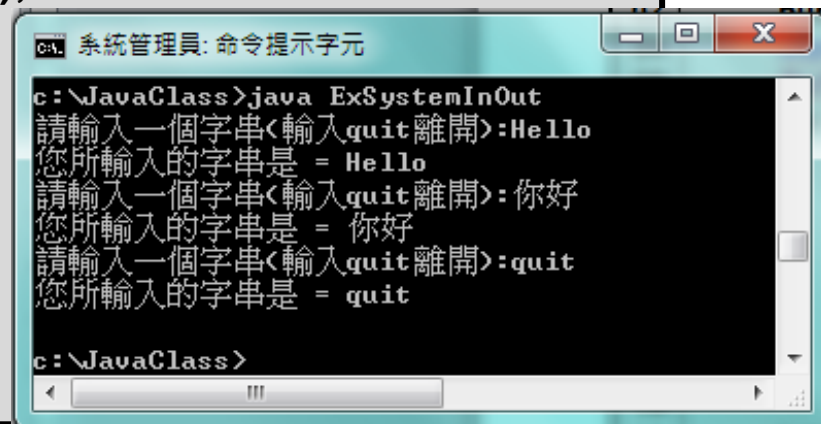
# 輸出資料流串接





# System.in & System.out

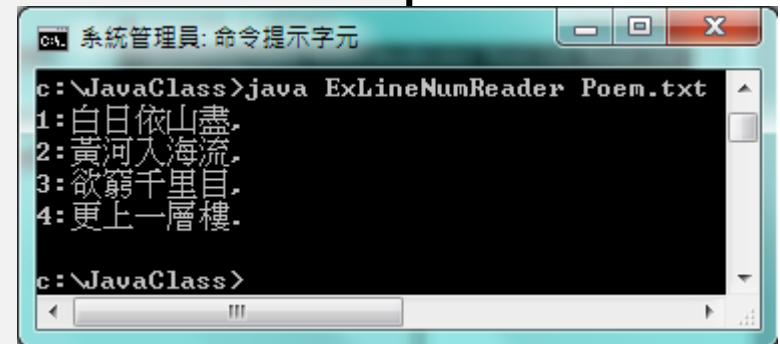
```
01 import java.io.*;
02 public class ExSystemInOut {
03     public static void main(String[] args) {
04         InputStreamReader in = null;
05         BufferedReader br = null;
06         try {
07             in = new InputStreamReader(System.in);
08             br = new BufferedReader(in);
09             String s;
10             do {
11                 System.out.print("請輸入一個字串(輸入quit離開):");
12                 s = br.readLine();
13                 System.out.println("您所輸入的字串是 = " + s);
14             } while (!(s.equals("quit")));
15         } catch(IOException e) {
16         } finally {
17             try {
18                 br.close();
19             } catch(IOException e){}
20         }
21     }
22 }
```



```
C:\JavaClass>java ExSystemInOut
請輸入一個字串<輸入quit離開>:Hello
您所輸入的字串是 = Hello
請輸入一個字串<輸入quit離開>:你好
您所輸入的字串是 = 你好
請輸入一個字串<輸入quit離開>:quit
您所輸入的字串是 = quit
C:\JavaClass>
```

# LineNumberReader 範例

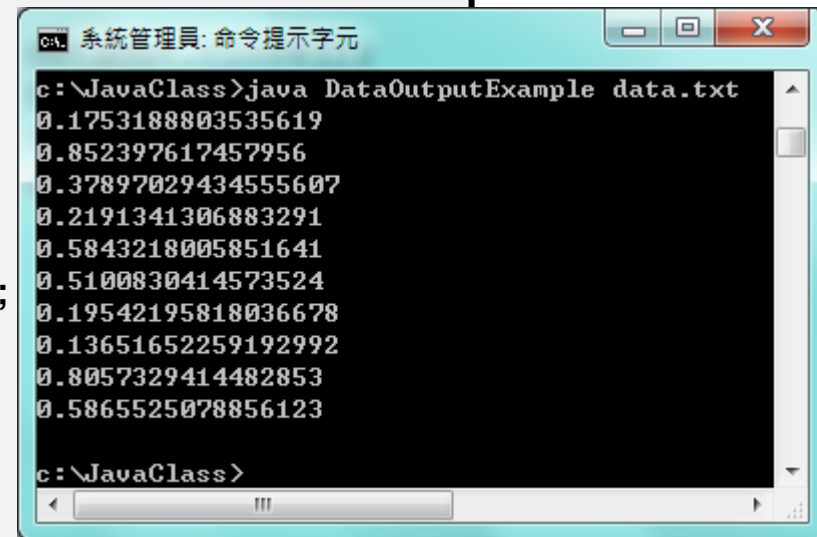
```
01 import java.io.*;
02 public class ExLineNumReader {
03     public static void main(String argv[]) {
04         try {
05             FileReader fin = new FileReader(argv[0]);
06             LineNumberReader lnr = new LineNumberReader(fin);
07
08             String str = lnr.readLine();
09             while (str != null) {
10                 System.out.println(lnr.getLineNumber()+":"+str);
11                 str = lnr.readLine();
12             }
13
14             lnr.close();
15             fin.close();
16         } catch (IOException e) {
17             System.err.println(e);
18         }
19     }
20 }
```



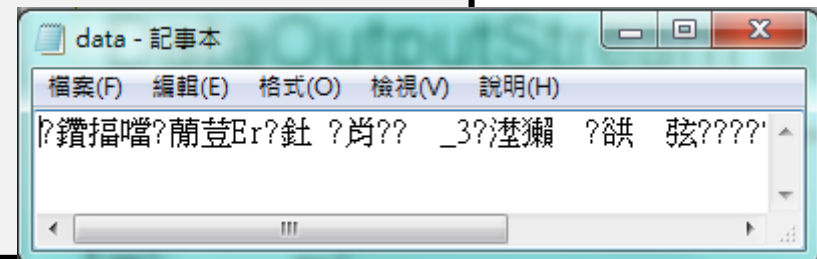
```
C:\JavaClass>java ExLineNumReader Poem.txt
1: 白日依山盡
2: 黃河入海流
3: 欲窮千里目
4: 更上一層樓
C:\JavaClass>
```

# DataOutputStream 範例

```
01 import java.io.*;
02 public class DataOutputExample {
03     public static void main(String argv[]) {
04         try {
05             FileOutputStream fout = new FileOutputStream(argv[0]);
06             DataOutputStream dataOut = new DataOutputStream(fout);
07
08             double data;
09             for(int i=0; i<10; i++) {
10                 data = Math.random();
11                 System.out.println(data);
12                 dataOut.writeDouble(data);
13             }
14
15             dataOut.close();
16             fout.close();
17         } catch (IOException e) {
18             System.err.print(e);
19         }
20     }
21 }
```

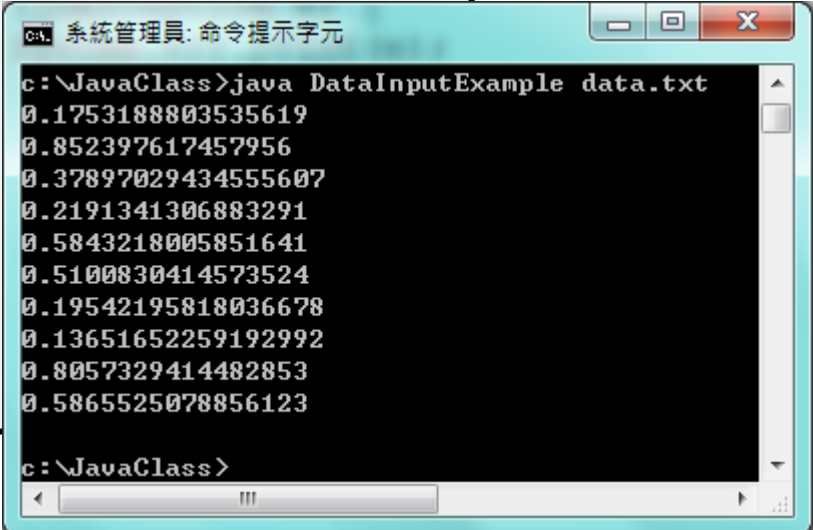


```
c:\JavaClass>java DataOutputExample data.txt
0.1753188803535619
0.852397617457956
0.37897029434555607
0.2191341306883291
0.5843218005851641
0.5100830414573524
0.19542195818036678
0.13651652259192992
0.8057329414482853
0.5865525078856123
c:\JavaClass>
```



# DataInputStream 範例

```
01 import java.io.*;
02 public class DataInputExample {
03     public static void main(String argv[]) {
04         try {
05             FileInputStream fin = new FileInputStream(argv[0]);
06             DataInputStream dataIn = new DataInputStream(fin);
07
08             while(fin.available() > 0)
09                 System.out.println(dataIn.readDouble());
10
11             dataIn.close();
12             fin.close();
13         } catch (IOException e) {
14             System.err.print(e);
15         }
16     }
17 }
```



```
系統管理員: 命令提示字元
c:\JavaClass>java DataInputExample data.txt
0.1753188803535619
0.852397617457956
0.37897029434555607
0.2191341306883291
0.5843218005851641
0.5100830414573524
0.19542195818036678
0.13651652259192992
0.8057329414482853
0.5865525078856123
c:\JavaClass>
```



# 永續性 Persistence

## ■ 永續性 Persistence

- 將資料以可永久保存的形式儲存
  - 檔案 / 資料庫...等
- 記憶體中的資料為非永續性資料
  - JVM關閉後,資料即不存在

## ■ 序列化(Serialization)與反序列化(Deserialization)

- 物件資料輸出(序列化位元組),儲存於外部媒體(檔案)
- 日後可由外部媒體(檔案)載入,將物件還原

# 物件序列化(Serialization)

## ■ 物件序列化步驟

### □ 類別需 implements Serializable Interface

- 標記介面 Marker interface：只有介面宣告,內部完全沒有任何常數或方法宣告

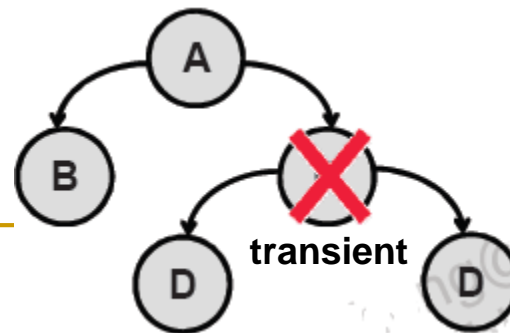
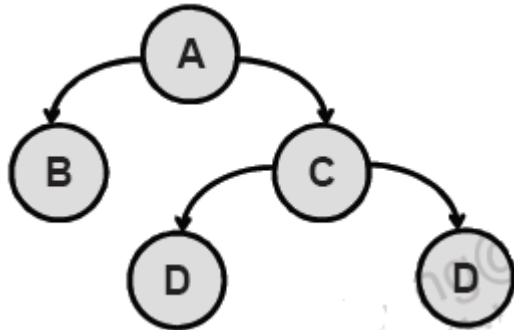
### □ ObjectOutputStream 將物件內容輸出

### □ ObjectInputStream 將物件內容載入

# 物件序列化(Serialization)

## ■ 物件序列化特性

- 序列化時,僅物件屬性資料被保留於外部媒體
- 類別屬性及**transient**屬性不會被輸出
  - 某屬性不想被序列化,加上**transient**修飾子
    - 特定作業系統相關的暫時資料
  - 物件反序列化(還原)時,該屬性為預設值
- 屬性為參考型別物件時,該物件也會被序列化
  - 序列化機制會尋訪整個物件結構,將屬性輸出至外部媒體
  - 該類別也需實作**Serializable**介面
    - **NotSerializableException**



# 物件序列化(Serialization)

## ❑ serialVersionUID

- 於序列化的過程中輸出
- 反序列化時,用以檢查載入的類別與檔案中物件是否相容

❑ InvalidClassException

- 自行宣告 serialVersionUID

```
private static long serialVersionUID = 42L;
```

- 若類別中未宣告 serialVersionUID

❑ Compiler 依類別內容自動運算提供

❑ 不同的Compiler實作,相同類別可能運算出不同的  
serialVersionUID

# ObjectOutputStream 類別

建構子	說明
<code>ObjectOutputStream(OutputStream out)</code> throws <code>IOException</code>	建立一個輸出至指定OutputStream的ObjectOutputStream物件

常用方法	傳回值	說明
<code>writeObject(Object obj)</code> throws <code>IOException</code>	<code>void</code>	寫一個物件到指定的OutputStream
<code>writeBoolean(boolean val)</code> throws <code>IOException</code>	<code>void</code>	寫一個boolean值到指定的OutputStream
<code>writeChar(char val)</code> throws <code>IOException</code>	<code>void</code>	寫一個char值到指定的OutputStream
<code>writeInt(int val)</code> throws <code>IOException</code>	<code>void</code>	寫一個int值到指定的OutputStream
<code>writeDouble(double val)</code> throws <code>IOException</code>	<code>void</code>	寫一個double值到指定的OutputStream
<code>writeChars(String str)</code> throws <code>IOException</code>	<code>void</code>	寫一個字串到指定的OutputStream

# ObjectInputStream 類別

建構子	說明
<code>ObjectInputStream(InputStream in)</code> throws <code>IOException</code>	建立一個由指定InputStream的讀資料的ObjectInputStream物件

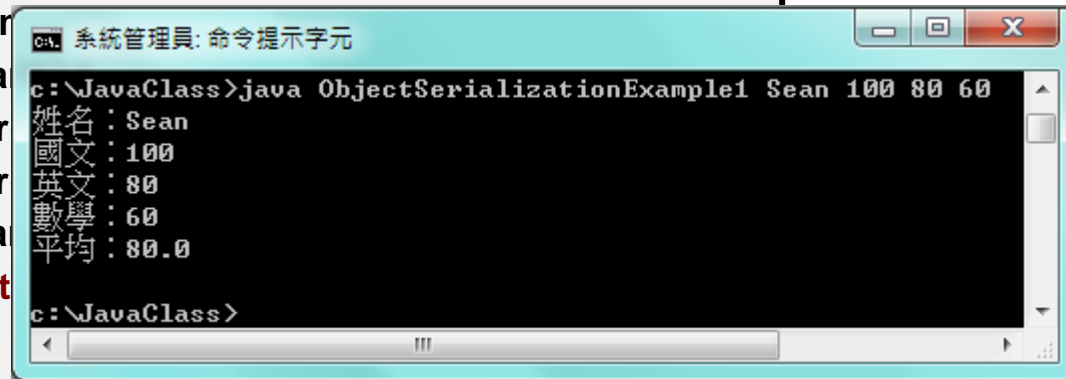
常用方法	傳回值	說明
<code>readObject()</code> throws <code>IOException</code> , <code>ClassNotFoundException</code>	<code>Object</code>	由指定InputStream的讀一個物件
<code>readBoolean()</code> throws <code>IOException</code>	<code>boolean</code>	由指定InputStream的讀一個boolean值
<code>readChar()</code> throws <code>IOException</code>	<code>char</code>	由指定InputStream的讀一個char值
<code>readInt()</code> throws <code>IOException</code>	<code>int</code>	由指定InputStream的讀一個int值
<code>readDouble()</code> throws <code>IOException</code>	<code>double</code>	由指定InputStream的讀一個double值

# 物件序列化範例

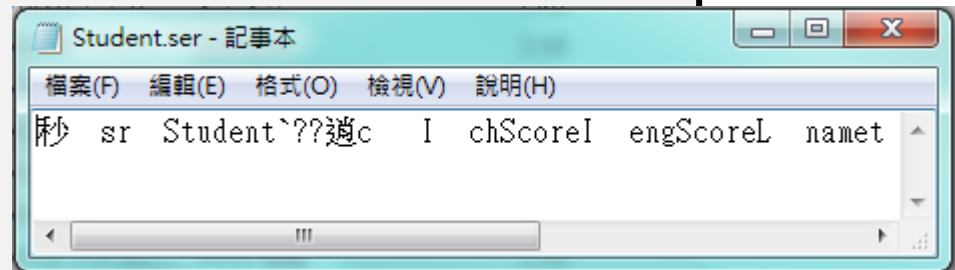
```
01 import java.io.*;
02 public class Student implements Serializable {
03     private String name;
04     private int chScore, engScore;
05     private transient int mathScore;
06     public Student(String n, int c, int e, int m) {
07         name = n;
08         chScore = c;
09         engScore = e;
10         mathScore = m;
11     }
12     public double avgScore() {
13         return (chScore+engScore+mathScore)/3.0;
14     }
15     public void printData() {
16         System.out.println("姓名 : "+name);
17         System.out.println("國文 : "+chScore);
18         System.out.println("英文 : "+engScore);
19         System.out.println("數學 : "+mathScore);
20         System.out.println("平均 : "+avgScore());
21     }
22 }
```

# 物件序列化範例

```
01 import java.io.*;
02 public class ObjectSerializationExample1 {
03     public static void main(String[] args) {
04         int c = Integer.parseInt(args[0]);
05         int e = Integer.parseInt(args[1]);
06         int m = Integer.parseInt(args[2]);
07         Student st = new Student(c, e, m);
08         st.printData();
09         try {
10             FileOutputStream fout = new FileOutputStream("Student.ser");
11             ObjectOutputStream objOut = new ObjectOutputStream(fout);
12
13             objOut.writeObject(st);
14             objOut.close();
15             fout.close();
16         } catch (IOException ie) {
17             System.err.println(e);
18         }
19     }
20 }
```



```
c:\JavaClass>java ObjectSerializationExample1 Sean 100 80 60
姓名: Sean
國文: 100
英文: 80
數學: 60
平均: 80.0
c:\JavaClass>
```



```
Student.ser - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
秒 sr Student'??道c I chScoreI engScoreL namet
```



# 物件序列化範例

```
01 import java.io.*;
02 public class ObjectSerializationExample2 {
03     public static void main(String argv[]) {
04         try {
05             FileInputStream fin = new FileInputStream("Student.ser");
06             ObjectInputStream objIn = new ObjectInputStream(fin);
07
08             Student st = (Student)objIn.readObject();
09             st.printData();
10
11             objIn.close();
12             fin.close();
13         } catch (Exception e) {
14             System.err.println(e);
15         }
16     }
17 }
```



系統管理員: 命令提示字元

```
c:\JavaClass>java ObjectSerializationExample2
姓名 : Sean
國文 : 100
英文 : 80
數學 : 0
平均 : 60.0

c:\JavaClass>
```

# 課程大綱

- 1) **Java** 資料輸出入處理架構
- 2) 節點資料流 **Node Stream**
- 3) 資料流串接
- 4) **Java**檔案結構

# 檔案結構 – java.io.File 類別

建構子	說明
File(String pathname)	字串可以是路徑字串,資料夾名稱字串,檔名字串
File(String parent, String child)	parent是路徑字串,child是檔名字串
File(File parent, String child)	Parent是另一個File實體代表路徑,child是檔名字串

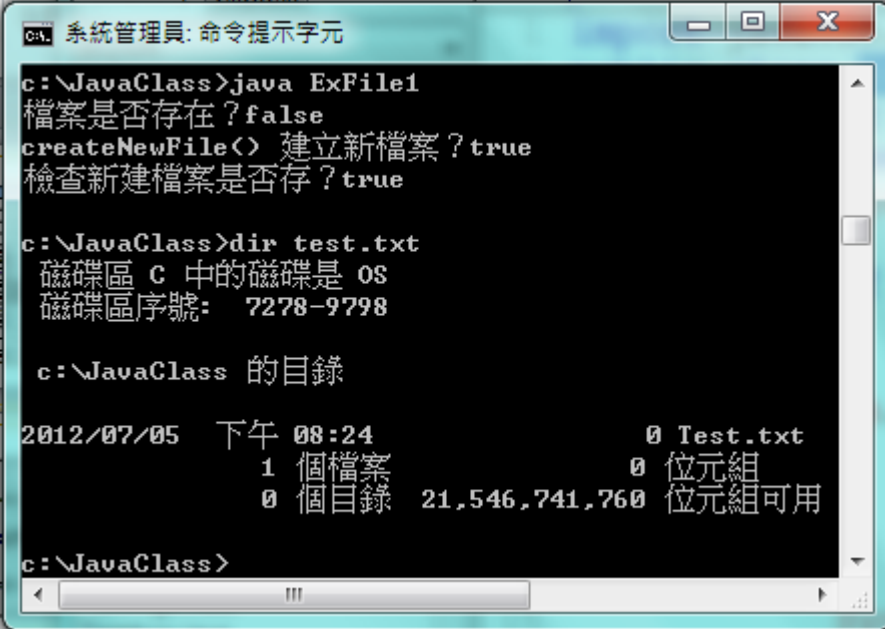
常用方法	傳回值	說明
createNewFile()	boolean	在File()中所指定的目錄下建立新檔
createTempFile(String prefix, String suffix)	static File	在File()中所指定的目錄下建立暫時檔 prefix：檔名；suffix：副檔名
createTempFile(String prefix, String suffix, File directory)	static File	在自行指定File()的目錄下建立暫時檔 prefix：檔名；suffix：副檔名；directory：File目錄
mkdir()	boolean	建立目錄，若指定的目錄不存在則將無法建立(通常會因為parent目錄不存在而失敗) 建立成功：true；建立失敗：false
mkdirs()	boolean	建立目錄，若指定的目錄不存在則會先自行建立所需的parent目錄，之後再建立所要建立的目錄。 建立成功：true；建立失敗：false

# java.io.File 類別

常用方法	傳回值	說明
getName()	String	取得檔案名稱
getPath()	String	取得路徑
getAbsolutePath()	String	取得絕對路徑
getParent()	String	取得上層路徑
renameTo(File dest)	boolean	修改檔名或目錄路徑
lastModified()	long	取得檔案最後修改時間
length()	long	取得檔案大小(bytes)
delete()	boolean	刪除指定的檔名或目錄路徑
exists()	boolean	判斷檔案或目錄路徑是否存在
canWrite()	boolean	判斷檔案是否可寫入
canRead()	boolean	判斷檔案是否可讀取
isFile()	boolean	判斷是否為檔案
isDirectory()	boolean	判斷是否為目錄
isAbsolute()	boolean	判斷是否為絕對路徑

# java.io.File 範例

```
01 import java.io.*;
02 public class ExFile1{
03     public static void main(String[] args) {
04         File f = new File("Test.txt");
05         System.out.println("檔案是否存在？" + f.exists());
06
07         if (!f.exists()) {
08             try {
09                 System.out.print("createNewFile() 建立新檔案？");
10                 System.out.println(f.createNewFile());
11             } catch (IOException e) {
12                 System.out.println(e);
13             }
14
15             System.out.println("檢查新檔案");
16         }
17     }
18 }
```



系統管理員: 命令提示字元

```
c:\JavaClass>java ExFile1
檔案是否存在？false
createNewFile() 建立新檔案？true
檢查新建檔案是否存？true

c:\JavaClass>dir test.txt
磁碟區 C 中的磁碟是 OS
磁碟區序號: 7278-9798

c:\JavaClass 的目錄

2012/07/05 下午 08:24                0 Test.txt
                        1 個檔案                0 位元組
                        0 個目錄    21,546,741,760 位元組可用

c:\JavaClass>
```

# java.io.File 範例

```
01 import java.io.*;
02 public class ExFile2 {
03     public static void main(String[] args) {
04         String path = args[0];
05         File f1 = new File(path);
06         String[] fileList = f1.list();
07         for(int i=0;i<fileList.length;i++) {
08             File f2 = new File(path + fileList[i]);
09             if (f2.isDirectory())
10                 System.out.println(fileList[i] + "
11                 else
12                 System.out.println(fileList[i] + "
13         }
14     }
15 }
```



The screenshot shows a Windows Command Prompt window titled "系統管理員: 命令提示字元". The command executed is `c:\JavaClass>java ExFile2 C:\`. The output lists various system directories and files, each followed by a status in Chinese: "是目錄." (is directory) or "是檔案." (is file). The output is as follows:

```
c:\JavaClass>java ExFile2 C:\
$RECYCLE.BIN : 是目錄.
BDE32 : 是目錄.
Beceem : 是目錄.
Boot : 是目錄.
bootmgr : 是檔案.
BOOTSECT.BAK : 是檔案.
Config.Msi : 是目錄.
Dell : 是目錄.
dell.sdr : 是檔案.
Documents and Settings : 是目錄.
ETAX : 是目錄.
FIND_EULA_PATH : 是目錄.
gdiplus.dll : 是檔案.
hiberfil.sys : 是檔案.
Intel : 是目錄.
JavaClass : 是目錄.
MSOCache : 是目錄.
pagefile.sys : 是檔案.
PerfLogs : 是目錄.
Program Files : 是目錄.
Program Files (x86) : 是目錄.
ProgramData : 是目錄.
System Volume Information : 是目錄.
Temp : 是目錄.
Users : 是目錄.
WINDOWS : 是目錄.

c:\JavaClass>
```