




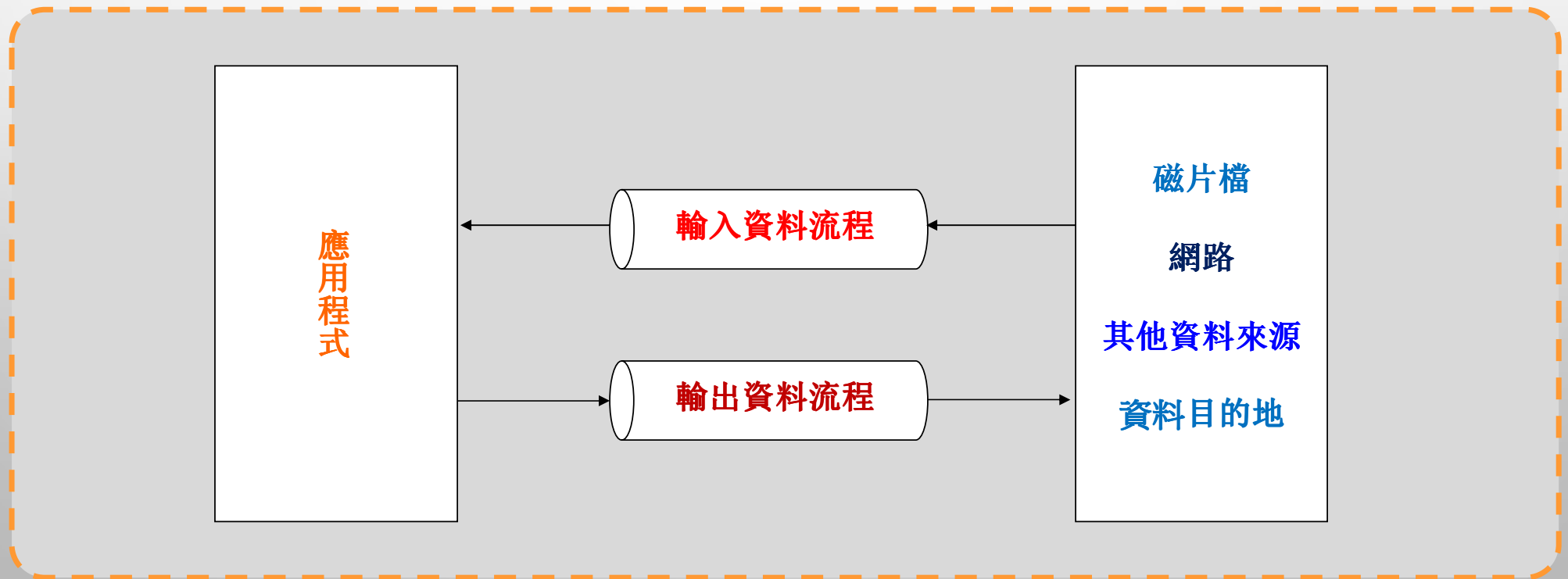
輸入輸出

學習目標

- 瞭解串流與輸入輸出的關係
 - **File**類別
 - 常用I/O串流類別
 - 物件串行化
 - **NIO**
- 

串流

- 串流代表資料的序列
- 輸入串流表示從一個來源讀取資料
- 輸出串流表示向一個目標寫入資料



串流的分類

串流根據處理資料類型:

- 位元組串流。
- 字元串流。

串流根據資料的流向:

- 輸入串流。
- 輸出串流。

串流根據處理資料功能:

- 實體串流:對資料不做任何處理，只完成基本的讀寫操作。
- 裝飾串流:在實體串流的基礎上，提供更高級的功能。
例如:指定編碼。

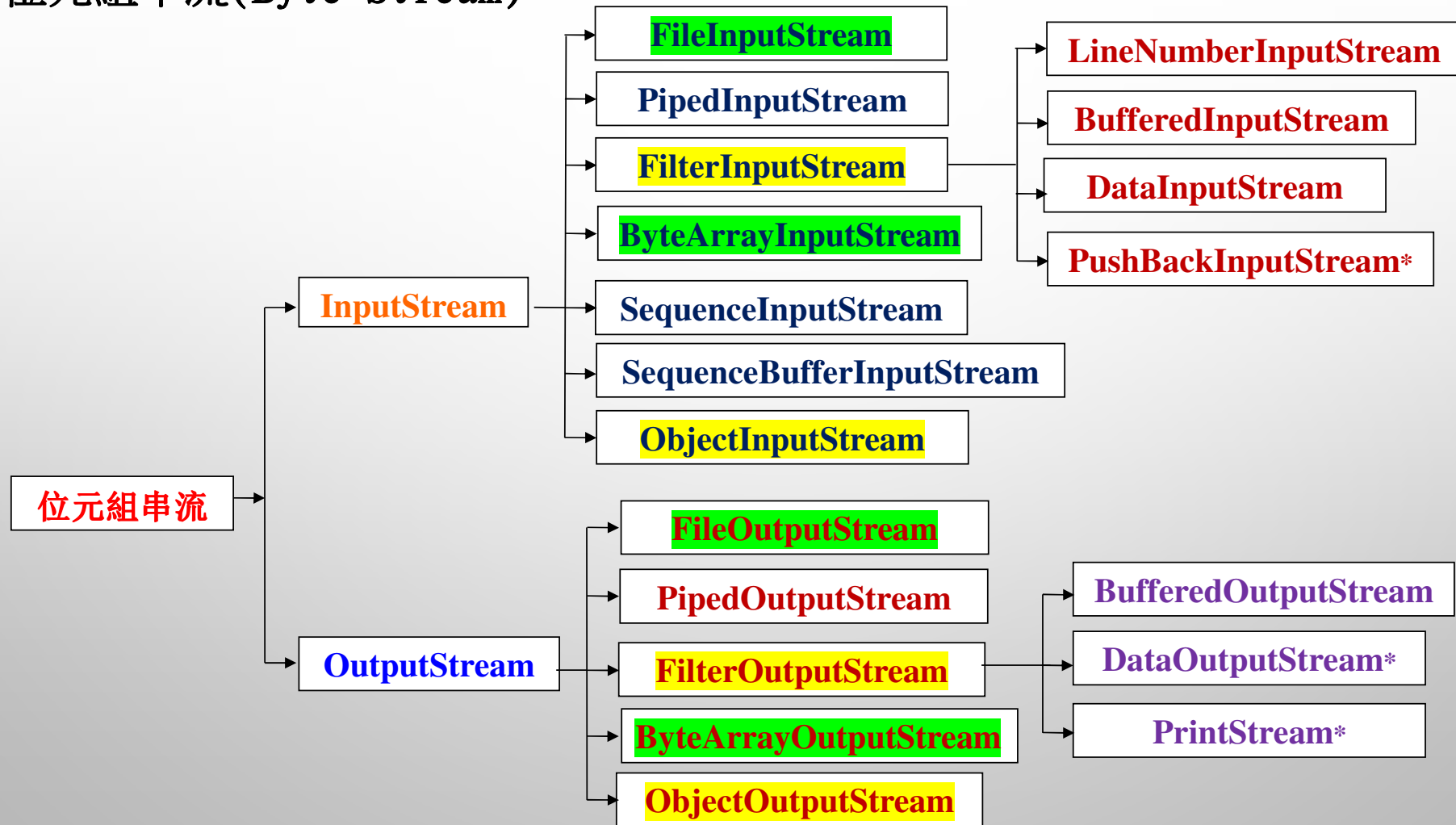
串流的分類

1. 位元組串流(Byte Stream)

- 在資料傳輸過程中以位元組為單位進行輸入和輸出。
- 適用於傳輸各種類型的檔或資料。
- 在位元組輸入串流中，InputStream 類別是所有的輸入位元組串流的父類別，它是一個抽象類別。
- 在位元組輸出串流中，OutputStream 是所有的輸出位元組串流的父類別，它是一個抽象類別。
- 裝飾串流
- ObjectInputStream、FilterInputStream、ObjectOutputStream、FilterOutputStream 和所有的子類別都是裝飾串流。
- 實體串流
- ByteArrayOutputStream、FileOutputStream 。

串流的分類

1. 位元組串流(Byte Stream)

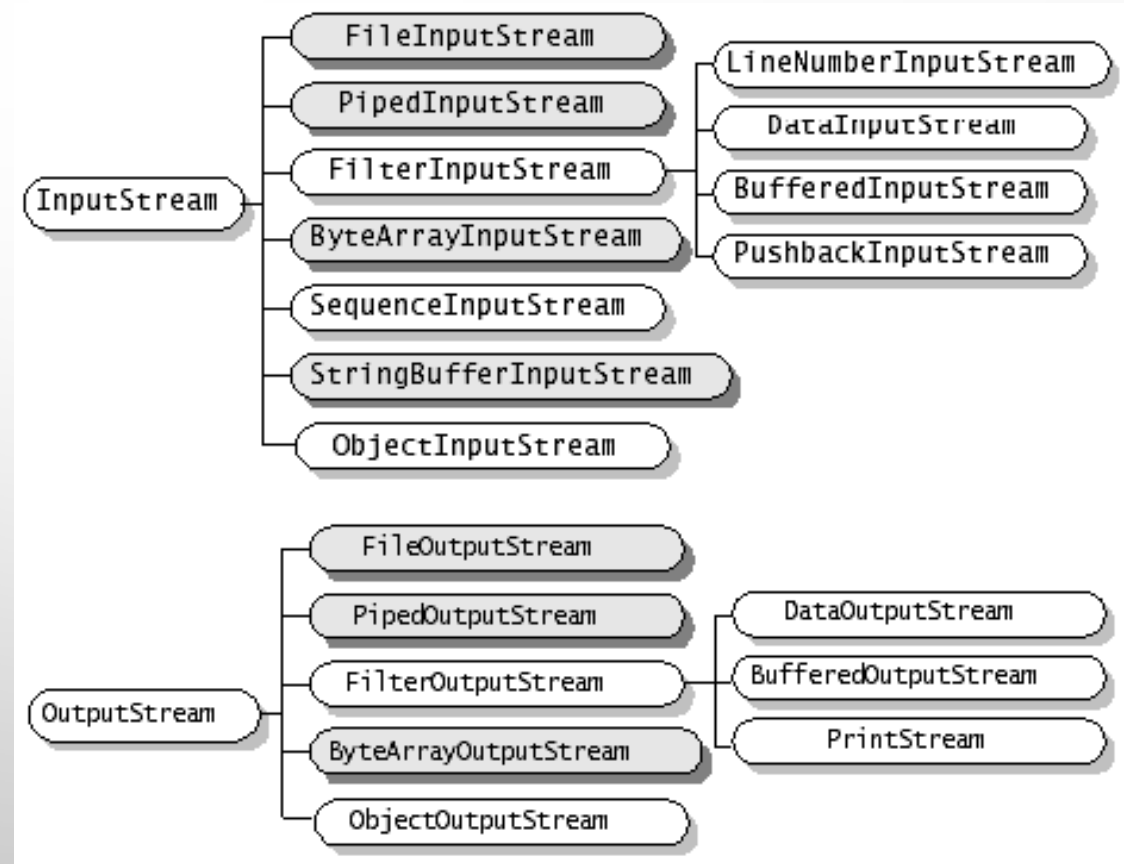


Overview of I/O Streams(串流)

- Byte Streams

- 用來處理 8 位元資料
，如：執行檔、圖檔
和聲音檔

- `InputStream` 和 `OutputStream` 是所有 `byte streams` 的 `abstract superclasses`



Standard In / Out Streams

- 標準輸入、輸出
 - 指的是 `System` 類別中的 `System.out` 和 `System.in` 子類別
 - `System` 類別不屬於 `java.io` 套件，而是屬於 `java.lang` 套件
 - 已經大量使用 `System.out.println()` 以及 `System.out.print()`
 - `System.out` 是 `PrintStream` 類別的物件，而 `PrintStream` 是 `OutputStream` 的子類別。
 - `System.in` 是 `InputStream` 類別的子類別物件。

Standard In / Out Streams

- **System.out**

- 在 Java 中，將字串輸出到螢幕顯示就是開啟 **System.out** 標準輸出的 **OutputStreamWriter** 串流，可以使用 **BufferedWriter** 來加速資料處理。如：

```
BufferedWriter output = new BufferedWriter(  
    new OutputStreamWriter(System.out));
```

- 如此便可以使用 **BufferedWriter** 中繼承自 **Writer** 類別中的 **write()** 方法來輸出字串到螢幕上。

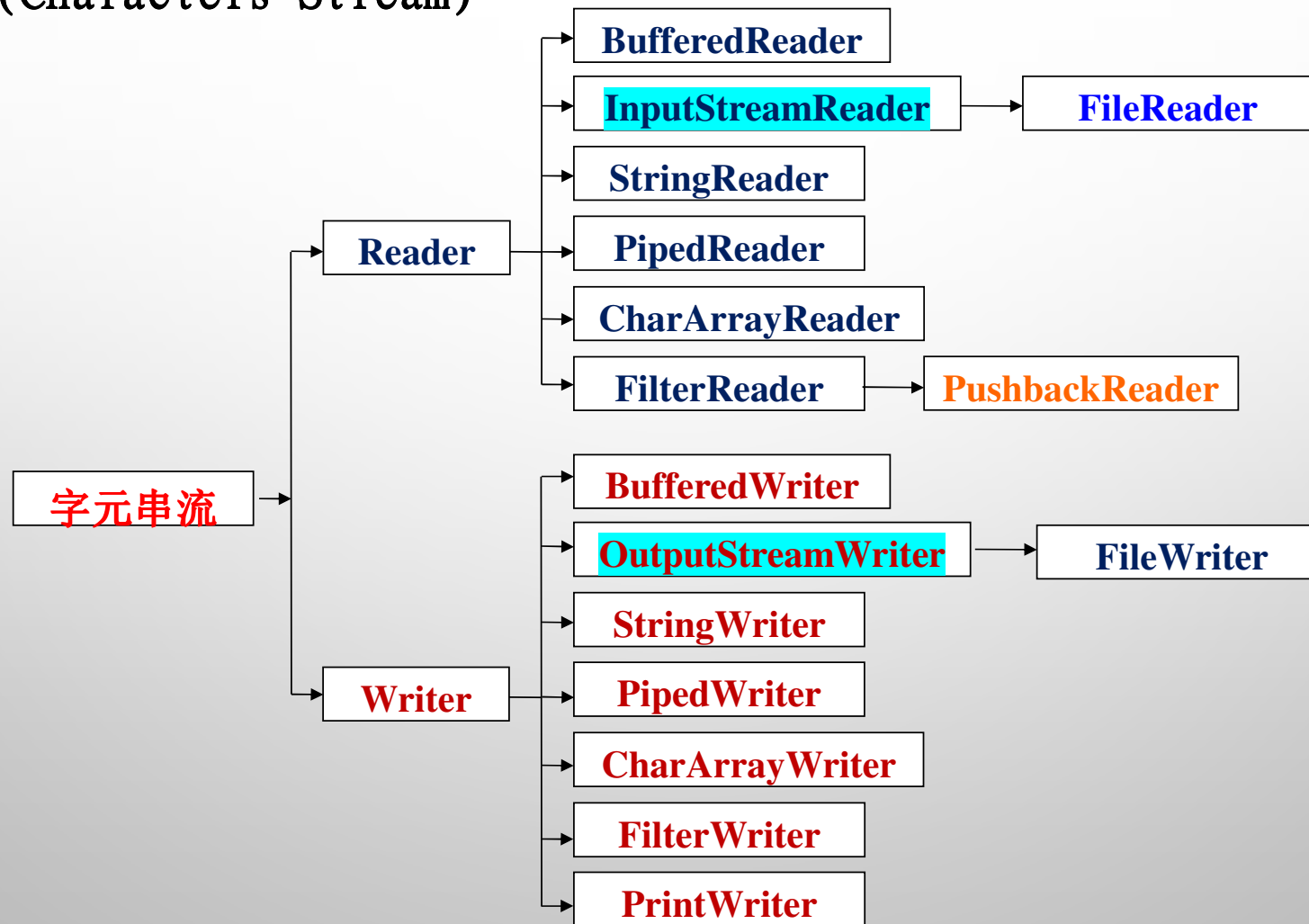
串流的分類

2. 字元串流(Characters Stream)

- 在資料傳輸過程中以字元為單位進行輸入和輸出。
- 一個字元佔用兩個位元組，因此字元串流只適用於字元類型資料的處理。
- 在字元輸入串流中，Reader 是所有的輸入字元串流的父類別，它是一個抽象類別。
- InputStreamReader 是一個連接位元組串流和字元串流的橋樑，它使用指定的字元集讀取位元組並轉換成字元。
- 其FileReader子類可以更方便地讀取字元檔，也是常用的Reader串流物件。
- 在字元輸出串流中，Writer是所有的輸出字元串流的父類別，也是一個抽象類別。

串流的分類

2. 字元串流(Character Stream)



檔案物件與檔案屬性

檔案用檔別類別的物件表示，有了檔物件就可以獲取檔的屬性。

類型	方法名	功能
	File(String filename)	在當前路徑下，創建一個名字為filename的檔案
	File(String path, String filename)	在給定的path路徑下，創建一個名字為filename的檔案
String	getName()	獲取此檔（目錄）的名稱
String	getPath()	獲取路徑名字串
String	getAbsolutePath()	獲取路徑名的絕對路徑名字串
long	length()	獲取檔案的長度。如果表示目錄，則返回值不確定
boolean	canRead()	判斷檔是否可讀

檔物件與檔案屬性

類型	方法名	功能
boolean	canWrite()	判斷檔案是否可寫
boolean	canExecute()	判斷檔案是否執行
boolean	exists()	判斷檔案（目錄）是否存在
boolean	isFile()	判斷檔案是否是一個標準檔
boolean	isDirectory()	判斷檔案是否是一個目錄
boolean	isHidden()	判斷檔案是否是一個隱藏檔
long	lastModified()	獲取檔案最後一次被修改的時間

讀取給定檔案的相關屬性，如果該檔案不存在則建立該檔案2-1

```
package ch10_1;
import java.io.*;
import java.util.Scanner;

public class ch10_1 {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        System.out.println("請輸入檔案名，例如：d:\\1.png");
        String s=scanner.nextLine();
        File file=new File(s);
        System.out.println("檔案名："+file.getName());
        System.out.println("文件大小為："+file.length()+"位元組");
        System.out.println("檔所在路徑為："+file.getAbsolutePath());

        if (file.isHidden())
        {
            System.out.println("該檔是一個隱藏檔");
        }
        else
        {
            System.out.println("該檔不是一個隱藏檔");
        }
    }
}
```

讀取給定檔案的相關屬性，如果該檔案不存在則建立該檔案2-2

```
if (!file.exists())
{
    System.out.println("該文件不存在");
    try
    {
        file.createNewFile();
        System.out.println("新檔創建成功");
    }
    catch(IOException e){}
}
```

結果

請輸入檔案名，例如：d:\1.png

1.txt

檔案名：1.txt

文件大小為：0位元組

檔所在路徑為：C:\Java\work\ch10_1\1.txt

該檔不是一個隱藏檔

目錄

Java把目錄作為一種特殊的檔案進行處理，它除了具備檔案的基本屬性如檔案名、所在路徑等資訊以外，同時也提供了專用於目錄的一些操作方法。

類型	方法名	功能
boolean	mkdir()	創建一個目錄，並返回創建結果。成功返回 true ，失敗（目錄已存在）返回 false
boolean	makedirs()	創建一個包括父目錄在內的目錄。創建所有目錄成功返回 true ，如果失敗返回 false ，但要注意的是有可能部分目錄已創建成功
String[]	list()	獲取目錄下字串表示形式的檔案名和目錄名
String[]	list(FilenameFilter filter)	獲取滿足指定篩檢程式條件的字串表示形式的檔案名和目錄名

目錄

類型	方法名	功能
File[]	listFiles()	獲取目錄下檔案類型表示形式的檔案名和目錄名
File[]	listFiles(FileFilter filter)	獲取滿足指定篩檢程式檔案條件的檔案表示形式的檔案名和目錄名
File[]	listFiles(FilenameFilter filter)	獲取滿足指定篩檢程式路徑和檔案條件的檔案表示形式的檔案名和目錄名

列出給定目錄下的所有檔案名，並列出給定副檔名的所有檔案名2-1

```
package ch10_2;
import java.io.*;
import java.util.Scanner;

public class ch10_2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("請輸入要訪問的目錄：");
        String s = scanner.nextLine();//讀取待訪問的目錄名
        File dirFile = new File(s);//創建目錄檔物件
        String[] allresults = dirFile.list();//獲取目錄下的所有檔案名
        for (String name : allresults)
            System.out.println(name);//輸出所有檔案名
        System.out.println("輸入檔副檔名：");
        s = scanner.nextLine();
        Filter_Name fileAccept = new Filter_Name();//創建檔案名過濾對象
        fileAccept.setExtendName(s);//設置過濾條件
        String result[] = dirFile.list(fileAccept);//獲取滿足條件的檔案名
        for (String name : result)
            System.out.println(name);//輸出滿足條件的檔案名
    }
}
```

列出給定目錄下的所有檔案名，並列出給定副檔名的所有檔案名2-2

```
class Filter_Name implements FilenameFilter
{
    String extendName;
    public void setExtendName(String s)
    {
        extendName = s;
    }
    public boolean accept(File dir, String name)
    { //重寫介面中的方法，設置過濾內容
        return name.endsWith(extendName);
    }
}
```

結果

請輸入要訪問的目錄：

C:\Java\work

.metadata

ch10_1

ch10_2

輸入檔副檔名：

txt

檔案的操作

1.創建檔案File類別的方法：

```
public boolean createNewFile()
```

例如，要想在D碟根目錄下創建立一個hello.txt檔，則
首先由File類別創建一個檔案物件：

```
File file = new File( “D:\\” ,” hello.txt” );  
file.createNewFile(); 建立檔案。
```

檔案的操作

2.刪除檔File類中的方法：

```
public boolean delete()
```

刪除hello.txt檔可以使用語句：

```
file.delete();
```

檔案的操作

3.運行可執行檔

需要使用`java.lang.Runtime`類別。

- 首先利用`Runtime`類別的靜態方法創建一個`Runtime`對象：
- `Runtime ec=Runtime.getRuntime();`
- 然後用`ec`呼叫方法：
- `Process exec(String command);`
- 方法執行本地的命令，參數`command`為指定的系統命令。
- `ec.exec()`

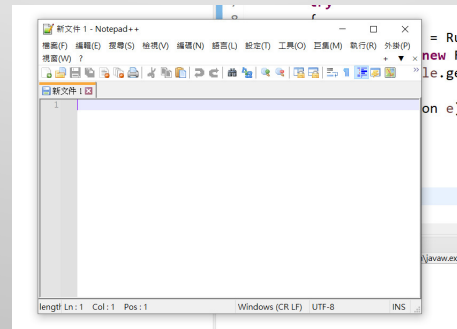
呼叫命令打開記事本

```
File file = new File("C:\\windows\\system32","notepad.exe");
```

```
package ch10_3;  
import java.io.File;
```

```
public class ch10_3 {  
    public static void main(String[] args) {  
        try  
        {  
            Runtime ec = Runtime.getRuntime();  
            File file=new File("C:\\Program Files\\Notepad++","notepad++.exe");  
            ec.exec(file.getAbsolutePath());  
        }  
        catch (Exception e){}  
    }  
}
```

結果



Scanner類別與檔案

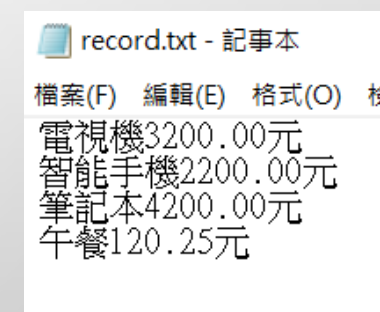
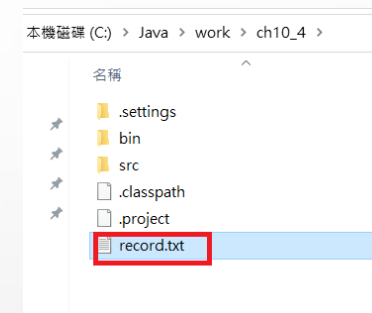
- 利用Scanner類別的物件還可以從檔案中讀取資料:
`Scanner input=new Scanner(檔案類別物件);`
- 創建的Scanner類別的物件使用`read()`方法即可從檔中讀資料。
- 讀資料時默認以空格作為資料的分隔標記。
- `public Scanner useDelimiter(String pattern)`
- 宣告scanner的分隔模式，指定分隔符號。
- `hasNextDouble()`
- 返回scanner的下一個標記是有效的double值,此方法返回true。

購物清單：電視機3200.00元，智能手機2200.00元，筆記本4200.00元，午餐120.25元。統計該次購物共花費多少？該購物清單存放在一個檔record.txt中。

```
Package ch10_4;
Import java.io.*;
Import java.util.*;
Public class ch10_4 {
    public static void main(String[] args) {
        File file = new File("record.txt");// 檔案物件
        Scanner reader = null;// 宣告Scanner物件
        double price, total = 0;
        try {
            reader = new Scanner(file);// 產生實體時file可能不存在，所以要放在try中
            reader.useDelimiter("[^0-9.]+");// 數據間分隔符號

            while (reader.hasNextDouble())// 是否還有數
            {
                price = reader.nextDouble();// 有，讀出並相加
                total = total + price;
            }
            System.out.println("總花費：" + total + "元");
        } catch (Exception e) {
        }
    }
}
```

先建立record.txt再執行



結果： 總花費：9720.25元

位元組串流

抽象類別**InputStream**和抽象類別**OutputStream**是所有位元組串流類別的根類別，其他位元組串流類都繼承自這兩個類別。

1.位元組輸入串流**InputStream**

- 位元組輸入串流的作用是從資料登錄源（例如從磁片、網路等）獲取位元組資料到應用程式（記憶體）中。

位元組串流

1. 位元組輸入串流InputStream

類型	方法名	功能
int	read()	從輸入串流中讀取下一個位元組，返回讀入的位元組資料；如果讀到末尾，返回-1
int	read(byte b[])	從輸入串流中讀取一定數量的位元組保存到位元組陣列中，並返回實際讀取的位元組數
int	read(byte b[],int off,int len)	從輸入串流中讀取最多len個位元組，保存到陣列b中從off開始的位置，並返回實際讀入的位元組數；如果off+len 大於b.length，或者off 和len中有一個是負數，那麼會拋出IndexOutOfBoundsException 異常
long	skip(long n)	從輸入串流中跳過並丟棄n個位元組，並返回實際跳過的位元組數

位元組串流

1. 位元組輸入串流InputStream

類型	方法名	功能
void	close()	關閉輸入串流，釋放資源。對串流的讀取完畢後呼叫該方法以釋放資源
int	available()	返回此輸入串流可以讀取（或跳過）的估計位元組數
void	mark(int readlimit)	在輸入串流中標記當前的位置。參數readlimit為標記失效前最多讀取的位元組數。如果讀取的位元組數超出此範圍則標記失效
void	reset()	將輸入串流重新定位到最後一次呼叫mark 方法時的位置
boolean	markSupported()	測試此輸入串流是否支援 mark 和 reset 方法。只有帶緩存的輸入串流支援標記功能

位元組串流

2.檔案位元組輸入串流類**FileInputStream**

在進行位元組輸入串流操作時，常使用**InputStream**類別的子類別**FileInputStream**，實現簡單的檔案資料讀取。

FileInputStream類別的常用構造方法：

```
public FileInputStream(File file) throws FileNotFoundException
```

```
public FileInputStream(String name) throws FileNotFoundException
```

- 通過給定的**File**物件和檔案建立位元組輸入串流物件。
- 在創建輸入串流時，如果檔案不存在或出現其他問題，會拋出**FileNotFoundException**例外，所以要注意捕獲。

位元組串流

2.檔案位元組輸入串流類FileInputStream

位元組輸入串流讀取資料步驟：

- 設定輸入串流的資料來源
- 建立指向資料來源的輸入串流
- 從輸入串流中讀取資料
- 關閉輸入串流

從磁片檔中讀取指定檔並顯示出來2-1

```
package ch10_5;
import java.io.*;
import java.util.Scanner;

public class ch10_5 {
    public static void main(String[] args) {
        byte[] b=new byte[1024];//設置位元組緩衝區
        int n=-1;
        System.out.println("請輸入要讀取的檔案名:(例如:d:\\hello.txt)");
        Scanner scanner =new Scanner(System.in);
        String str=scanner.nextLine();//獲取要讀取的檔案名
        try
        {
            FileInputStream in=new FileInputStream(str);//創建位元組輸入串流
            while((n=in.read(b,0,1024))!=-1)
            {//讀取檔內容到緩衝區，並顯示
                String s=new String (b,0,n);
                System.out.println(s);
            }
            in.close();//讀取文件結束，關閉文件
        }
    }
}
```

從磁片檔中讀取指定檔並顯示出來2-2

```
        catch(IOException e)
        {
            System.out.println("檔案讀取失敗");
        }
    }
}
```

結果

:

請輸入要讀取的檔案名:(例如:d:\hello.txt)

C:\Java\work\ch10_4\record.txt

電視機3200.00元

智能手機2200.00元

筆記本4200.00元

午餐120.25元

位元組串流

3.位元組輸出串流

位元組輸出串流的作用是将位元組資料從應用程式（記憶體）中傳送到輸出目的地，如外部設備、網路等。

位元組輸出串流OutputStream的常用方法

類型	方法名	功能
void	write(int b)	將整數 b 的低 8 位寫到輸出串流
void	write(byte b[])	將位元組陣列中資料寫到輸出串流
void	write(byte b[], int off,int len)	從位元組陣列 b 的 off 處寫 len 個位元組資料到輸出串流
void	flush()	強制將輸出串流保存在緩衝區中的資料寫到輸出串流
void	close()	關閉輸出串流，釋放資源

位元組串流

4.檔案位元組輸出串流類**FileOutputStream**

- 在進行位元組輸出串流操作時，經常使用的是**OutputStream**類別的子類**FileOutputStream**
- 用於將資料寫入**File**或其他的輸出串流。

FileOutputStream類的常用構造方法：

`public FileOutputStream(File file) throws IOException`

`public FileOutputStream(String name) throws IOException`

`public FileOutputStream(File file, boolean append) throws IOException`

`public FileOutputStream(String name, boolean append) throws IOException`

位元組串流

4.檔位元組輸出串流類FileOutputStream

位元組輸出串流寫資料步驟：

- 設定輸出串流的目的地
- 創建指向這目的地輸出
- 向輸出串流中寫入資料
- 關閉輸出串流

位元組串流

4.檔案位元組輸出串流類FileOutputStream

- 在完成寫操作過程中，系統會將資料暫存到緩衝區中，緩衝區存滿後再一次性寫入到輸出串流中。
- 執行close()方法時，不管緩衝區是否已滿，都會把其中的資料寫到輸出串流。

第一步建立檔案與寫入資料，第二步追加寫入3-1

```
package ch10_6;
import java.io.*;
import java.util.Scanner;

public class ch10_6 {
    public static void main(String[] args) {
        String content;//待輸出字串
        byte[] b;//輸出位元組流
        FileOutputStream out;//檔案輸出流
        Scanner scanner = new Scanner(System.in);
        System.out.println("請輸入檔名：(例如，d:\\hello.txt)");
        String filename=scanner.nextLine();
        File file = new File(filename);//創建檔案物件
        if (!file.exists())
        { //判斷檔是否存在
            System.out.println("檔案不存在，是否創建?(y/n)");
            String f =scanner.nextLine();
            if (f.equalsIgnoreCase("n"))
                System.exit(0);//不創建，退出
            else
            {
```

第一步建立檔案與寫入資料，第二步追加寫入3-2

```
        try
        {
            file.createNewFile();//創建新文件
        }
        catch(IOException e)
        {
            System.out.println("創建失敗");
            System.exit(0);
        }
    }

    try
    {
        //向檔案中寫內容
        content="Hello";
        b=content.getBytes();
        out = new FileOutputStream(file);//建立文件輸出流
        out.write(b);//完成寫入操作
        out.close();//關閉輸出流
        System.out.println("檔案寫入操作成功!");
    }
    catch(IOException e)
    {
        e.getMessage();
    }
}
```

第一步建立檔案與寫入資料，第二步追加寫入3-3

```
try
{ //向檔案中追加內容
    System.out.println("請輸入追加的內容：");
    content = scanner.nextLine();
    b=content.getBytes();
    out = new FileOutputStream(file,true); //創建可追加內容的輸出串流
    out.write(b); //完成追加寫操作
    out.close(); //關閉輸出串流
    System.out.println("檔案追加寫入操作成功！");
    scanner.close();
}
catch(IOException e)
{e.getMessage();}
}
```

請輸入檔名：（例如，d:\hello.txt）

結果

:

hello.txt

檔案不存在，是否創建？(y/n)

y

檔案寫入操作成功！

請輸入追加的內容：

1234

檔案追加寫入操作成功！

本機磁碟 (C:) > Java > work > ch10_6

名稱

.settings

bin

src

.classpath

.project

hello.txt

hello.txt - 記事本

檔案(F) 編輯(E) 格式(O)

Hello1234

字元串流

- 字元串流通常用於文字檔的傳輸。
- 抽象類**Reader**和抽象類別**Writer**是所有字元串流類別的根類別，其他字元串流類都繼承自這兩個類別。
- 其中一些子類別還在傳輸過程中對資料做了進一步處理以方便用戶的使用。

字元串流

1.字元輸入串流Reader

字元輸入串流**Reader**是所有字元輸入串流類的父類別，實現從資料來源讀入字元資料。常用方法有：

類型	方法名	功能
int	read()	從輸入串流讀取單個字元
int	read(char[] cbuf)	從輸入串流讀取字元保存到陣列 cbuf 中，返回讀取的字元數，如果已到達串流的末尾，則返回 -1
int	read(char[] cbuf,int off,int len)	從輸入流讀取最多 len 個字元保存到字元陣列 cbuf 中，存放的起始位置在 off 處。返回：讀取的字元數，如果已到達串流的末尾，則返回 -1

字元串流

1. 字元輸入串流Reader

類型	方法名	功能
long	skip(long n)	跳過n個字元。 返回： 實際跳過的字元數
void	mark(int readAheadLimit)	標記串流中的當前位置
void	reset()	重置該流
boolean	markSupported()	判斷此串流是否支持 mark() 操作
void	close()	關閉該串流， 釋放資源

字元串流

2.檔案字元輸入串流**FileReader**

- 進行字元輸入流操作時，經常使用的是**Reader**類別的子類別**FileReader**，用於從輸入串流讀取資料。

FileReader類的常用構造方法：

public FileReader(File file) throws FileNotFoundException

public FileReader(String name) throws FileNotFoundException

- 通過給定的**File**物件或檔案名創建字元輸入串流。
- 在創建輸入串流時，如果檔案不存在或出現其他問題，會拋出**FileNotFoundException**例外。

字元串流

3.字元輸出串流類別Writer

字元輸出串流**Writer**用於將字元資料輸出到目的地。類別**Writer**的常用方法：

類型	方法名	功能
void	write(int c)	將整數 c 的低16位寫到輸出串流
void	write(char[] cbuf)	將字元陣列中資料寫到輸出串流
void	write(cbuf[],int off,int len)	從字元陣列 cbuf 的 off 處開始取 len 個字元寫到輸出串流
void	write(String str)	將字串寫到輸出串流
void	write(String str,int off,int len)	從字串 str 的 off 處開始取 len 個字元資料寫到輸出串流
void	flush()	強制將輸出串流保存在緩衝區中的資料寫到輸出
void	close()	關閉輸出串流，釋放資源

字元串流

4.檔案字元輸出串流**FileWriter**類別

- **FileWriter**類別和位元組流**FileOutputStream**類相對應，實現字元的輸出操作，實現方法也基本相同。**FileWriter**類別的常用構造方法：

`public FileWriter(File file) throws IOException`

`public FileWriter(String name) throws IOException`

`public FileWriter(File file, boolean append) throws IOException`

`public FileWriter(String name, boolean append) throws IOException`

字元串流

4.檔案字元輸出串流FileWriter類別

- 如果第二個參數為 **true**，則將字元寫入檔案末尾處，而不是寫入檔案開始處。
- 如果檔案不存在或出現其他問題，會拋出IOException例外。

利用檔案串流實現檔案的複製功能2-1

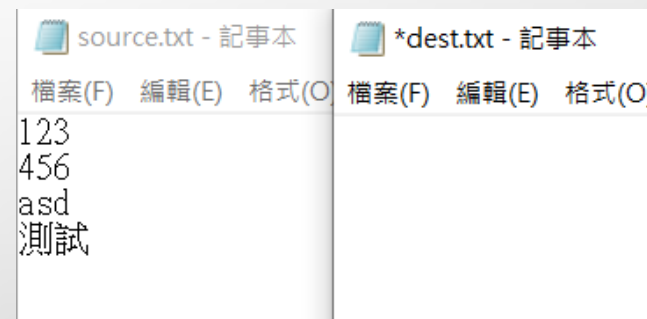
```
package ch10_7;
import java.io.*;
import java.util.Scanner;

public class ch10_7 {
    public static void main(String[] args) throws IOException
    {
        Scanner scanner = new Scanner(System.in);
        System.out.println("請輸入來源檔案名和目的檔案名，中間用空格分隔");
        String s = scanner.next();// 讀取來源檔案名
        String d = scanner.next();// 讀取目的檔案名
        File file1 = new File(s);// 創建來源檔案物件
        File file2 = new File(d);// 創建目的檔案物件
        if (!file1.exists()) {
            System.out.println("被複製的文件不存在");
            System.exit(1);
        }
        InputStream input = new FileInputStream(file1);// 創建來源檔案流
        OutputStream output = new FileOutputStream(file2);// 創建目的檔案流
        if ((input != null) && (output != null)) {
            int temp = 0;
```

利用檔案串流實現檔案的複製功能2-2

```
        while ((temp = input.read()) != (-1))
            output.write(temp); // 完成資料複製
    }
    input.close(); // 關閉原始檔案流
    output.close(); // 關閉目的檔案流
    System.out.println("檔複製成功!");
}
```

前



後



結果

:

請輸入原始檔案名和目的檔案名，中間用空格分隔

source.txt dest.txt

檔複製成功！