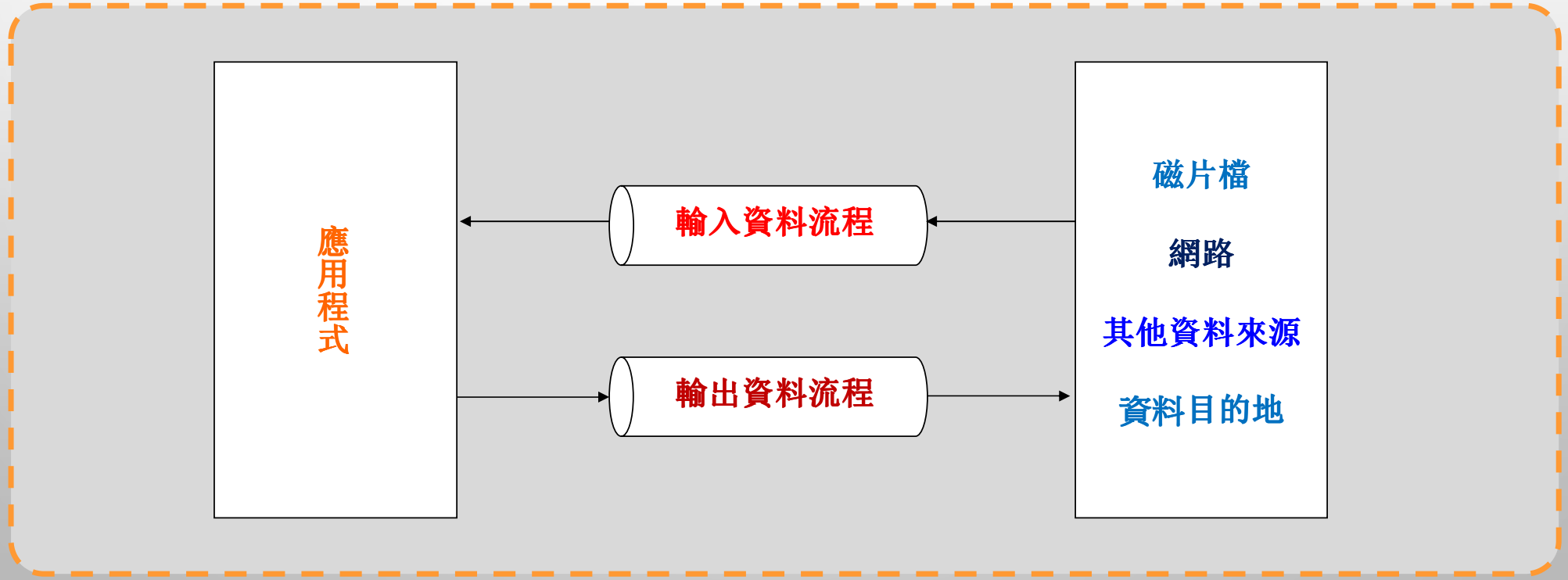
輸入輸出

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。

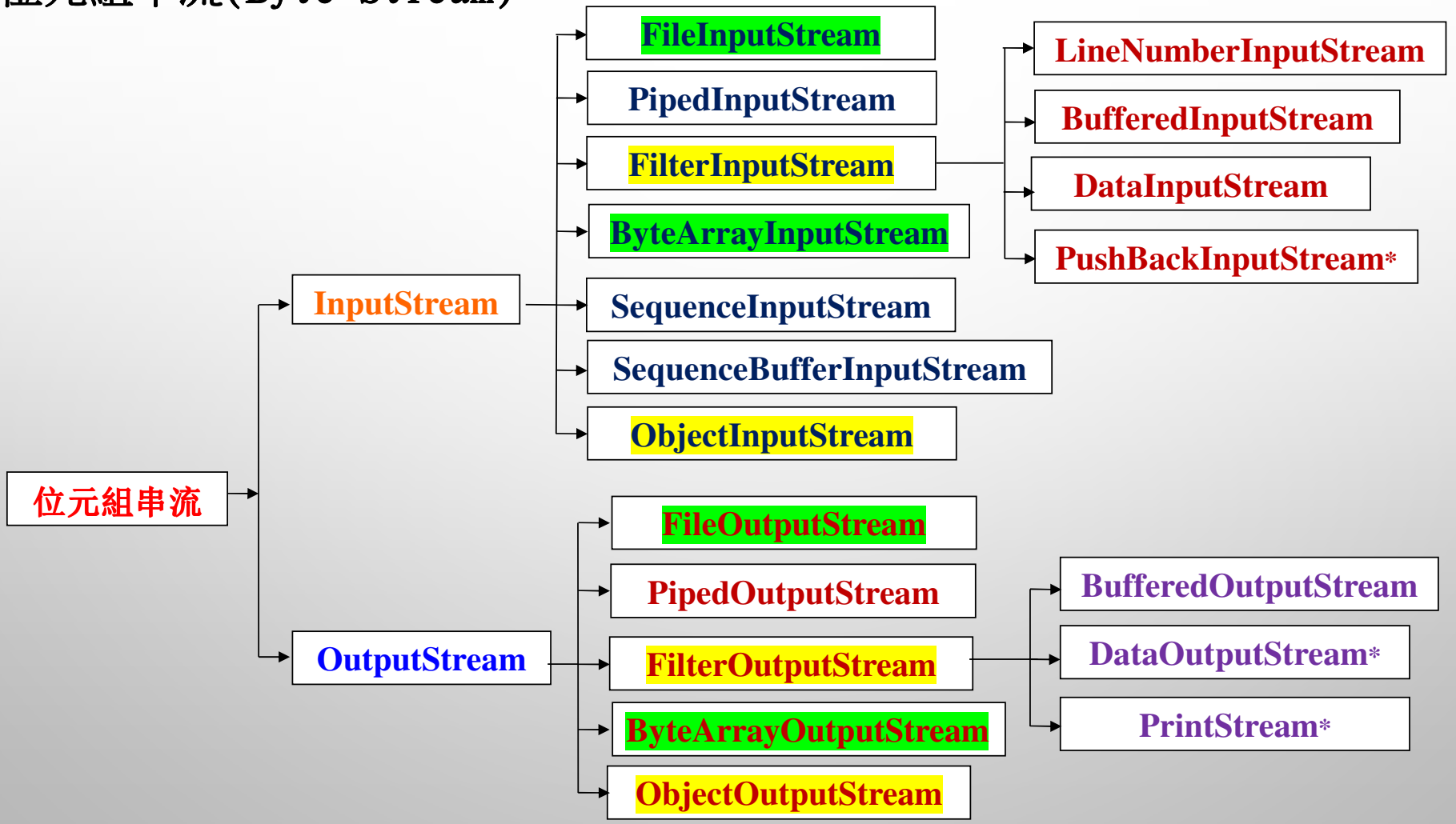
串流的分類

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。
- 裝飾器串流
- ObjectInputStream、FilterInputStream、ObjectOutputStream、FilterOutputStream 和所有的子類別都是裝飾器串流。
- 實體串流
- ByteArrayOutputStream、FileOutputStream 。

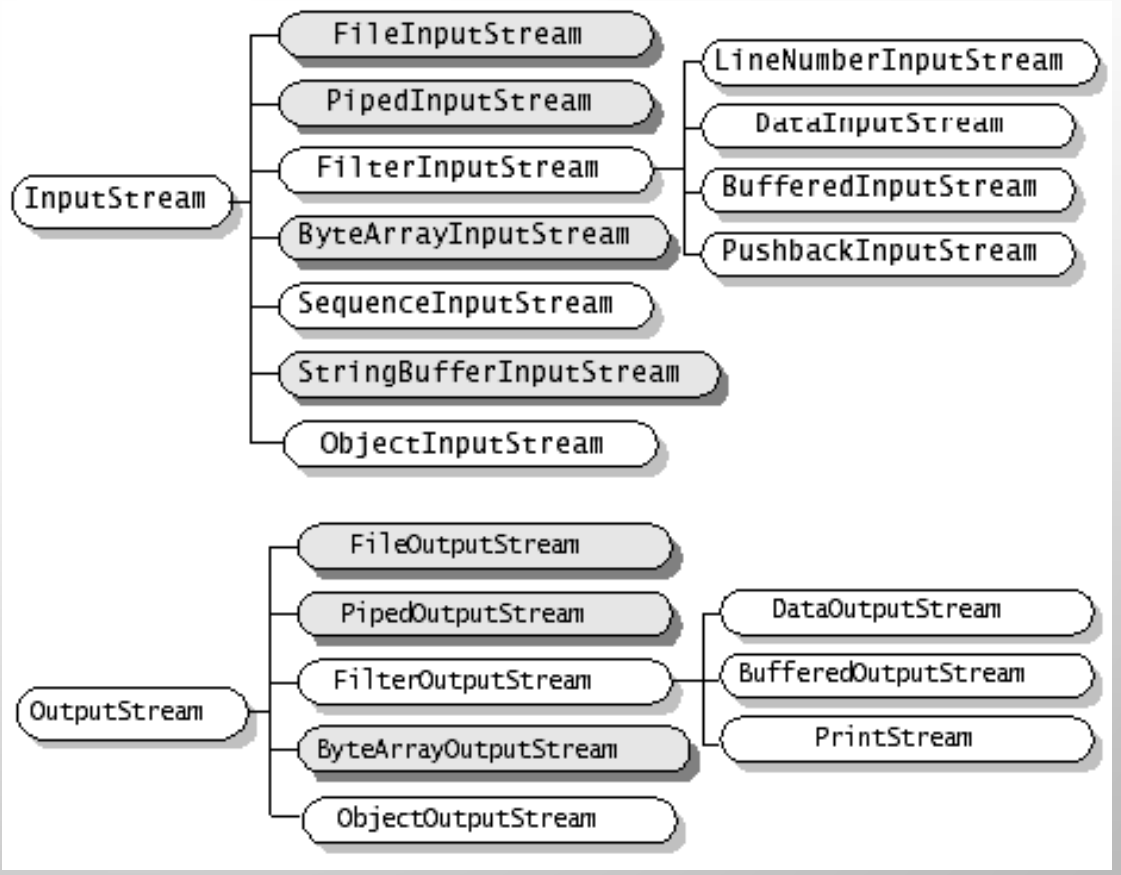
串流的分類

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。

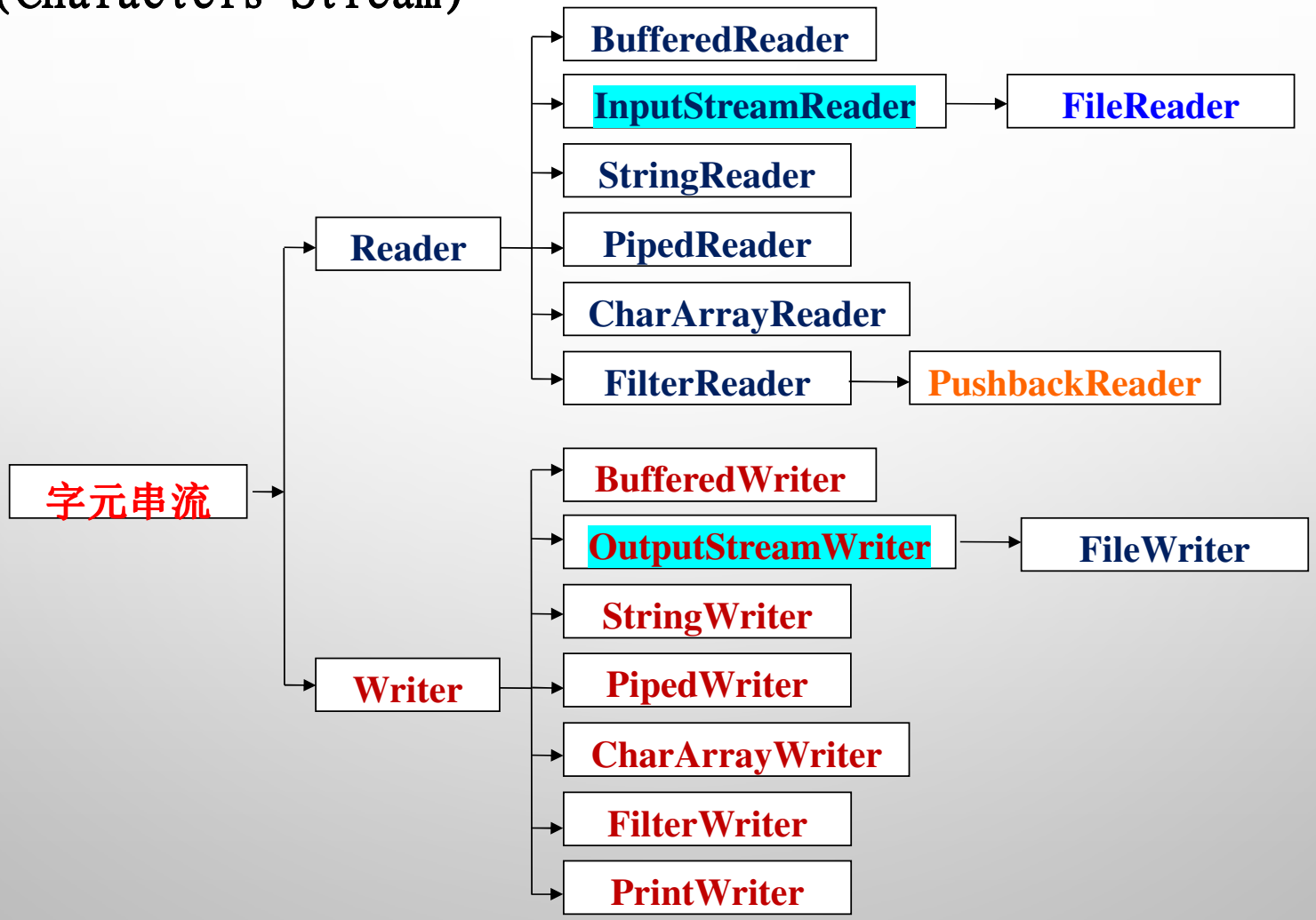
串流的分類

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(Characters 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()	獲取此檔案（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()	判讀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("檔案所在路徑為："+file.getAbsolutePath());

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

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("該檔案不存在");
               try
                 {
                   file.createNewFile();
                   System.out.println("新檔案創建成功");
               }
               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	mkdirs()	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);//創建目錄檔案物件
        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("輸入檔案副檔名:");
            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。

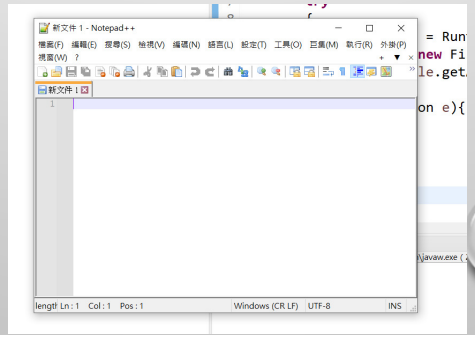
- 首先利用Runtime類別的靜態方法創建一個Runtime對象：
- Runtime ec=Runtime.getRuntime();
- 然後用ec呼叫方法：
- Process exec(String command);
- 方法執行本地的命令，參數command為指定的系統命令。
- ec.exec()

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。

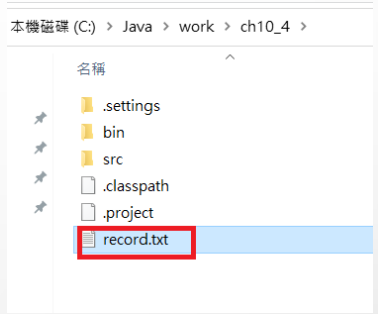
- public Scanner useDelimiter(String pattern)
- 宣告scanner的分隔模式，指定分隔符號。

- 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將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 FileOutnputStream(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.檔案位元組輸出串流類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

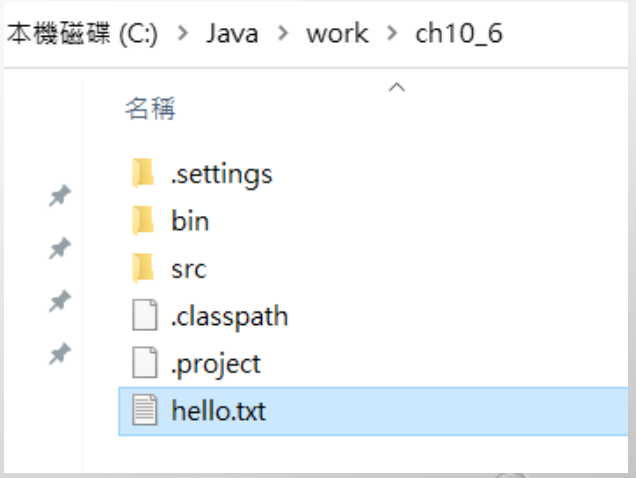
```
            try
            {
                file.createNewFile();//創建新文件
            }
                catch(IOException e)
            {
                System.out.println("創建失敗");
                System.exit(0);
            }
        }
}

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();}
```

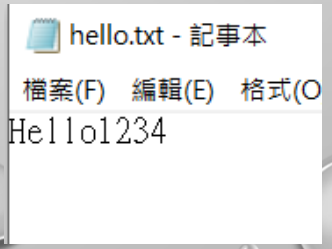
第一步建立檔案與寫入資料，第二步追加寫入3-3

```
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();}
}
}
```

結果：

```
請輸入檔名：（例如，d:\hello.txt）
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！
```





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類別的常用構造方法：

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;
```

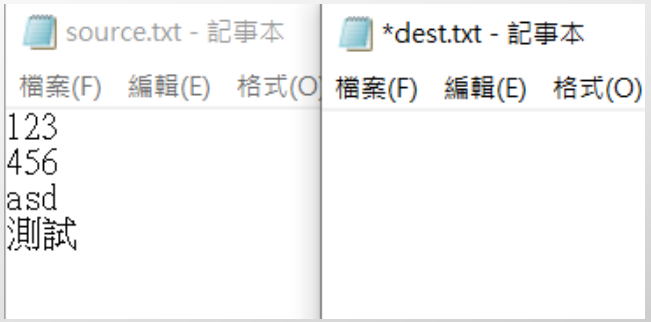
利用檔案串流實現檔案的複製功能2-2

```
            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();// 關閉目的檔流
    System.out.println("檔複製成功！");
  }
}
```

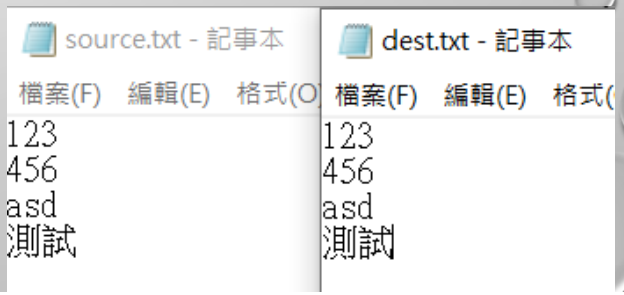
結果：

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！

前



後



資料登錄串流

- 資料串流程是Java提供的一種裝飾器類別串流。
- 建立在實體串流基礎上，讓程式不需考慮資料所占位元組個數的情況下就能夠正確地完成讀寫操作。
- DataInputStream類別和DataOutputStream類別分別為資料登錄串流類別和資料輸出串流類別。

資料登錄串流

- 資料登錄串流DataInputStream類別允許程式以與機器無關方式從底層輸入串流中讀取基本 Java 資料類型。

DataInputStream類別的常用方法：

類型	方法名	功能
	DataInputStream(InputStream in)	**使用指定的實體串流 InputStream 創建一個DataInputStream。**
boolean	**readBoolean()**	**讀取一個布林值**
byte	**readByte()**	**讀取一個位元組**
char	**readChar()**	**讀取一個字元**
long	**readLong()**	**讀取一個長整型數**
int	**readInt()**	**讀取一個整數**

資料登錄串流

DataInputStream類別的常用方法：

類型	方法名	功能
short	readShort()	讀取短整數型
float	readFloat()	讀取Float數
double	readDouble()	讀取Double數
String	readUTF()	讀取UTF字串
int	skipBytes(int n)	跳過並丟棄n個位元組，返回實際跳過的位元組數

資料輸出串流

- 資料輸出串流DataOutputStream類別允許程式以適當方式將基本Java資料類型寫入輸出串流中。

DataOutputStream類別的常用方法：

類型	方法名	功能
	DataOuputStream (OutputStream out)	創建一個新的資料輸出串流，將資料寫入指定基礎輸出串流
void	writeBoolean(Boolean v)	將一個布林值寫出到輸出串流
void	writeByte(int v)	將一個位元組寫出到輸出串流
void	writeBytes(String s)	將字串按位元組（每個字元的高八位元丟棄）順序寫出到輸出串流中
void	writeChar(int c)	將一個 char 值以2位元組值形式寫入輸出串流中，先寫入高位元組
void	writeChars(String s)	將字串按字元順序寫出到輸出串流
void	writeLong(long v)	將一個長整型數寫出到輸出串流

資料輸出串流

DataOutputStream類別的常用方法：

類型	方法名	功能
void	writeInt(int v)	將整數寫出到輸出串流
void	writeShort(int v)	將短整型數寫出到輸出串流
void	writeFloat(float v)	將Float數寫出到輸出串流
void	writeDouble(double v)	將Double數寫出到輸出串流
void	writeUTF(String s)	將字串用UTF_8編碼形式寫出到輸出串流
int	size()	返回寫到資料輸出串流中的位元組數
void	flush()	清空輸出流，使所有緩衝中的位元組被寫出到串流中

將資料寫入檔案中再讀出後顯示出來2-1

```
package ch10_8;
import java.io.*;

public class ch10_8 {
    public static void main(String[] args) {
    File file=new File("data.txt");
    try
    {
       FileOutputStream out=new FileOutputStream(file);
       DataOutputStream outData=new DataOutputStream(out);
       outData.writeBoolean(true);
       outData.writeChar('A');
       outData.writeInt(10);
       outData.writeLong(88888888);
       outData.writeFloat(3.14f);
       outData.writeDouble(3.1415926897);
       outData.writeChars("hello,every one!");
    }
    catch(IOException e){}
```

將資料寫入檔案中再讀出後顯示出來2-2

```
    try
    {
      FileInputStream in=new FileInputStream(file);
      DataInputStream inData=new DataInputStream(in);
      System.out.println(inData.readBoolean());//讀取boolean資料
      System.out.println(inData.readChar());//讀取字元資料
      System.out.println(inData.readInt());     //讀取int資料
      System.out.println(inData.readLong());    //讀取long資料
      System.out.println(+inData.readFloat()); //讀取float資料
      System.out.println(inData.readDouble()); //讀取double資料
      char c = '\0';
      while((c=inData.readChar())!='\0')//讀入字元不為空
      System.out.print(c);
    }
    catch(IOException e){}
  }
}
```

結果:

```
true
A
10
88888888
3.14
3.1415926897
hello,every one!
```

緩衝串流

- 緩衝串流是在實體I/O流基礎上增加一個緩衝區，應用程式和I/O設備之間的資料傳輸都要經過緩衝區來進行。
- 緩沖串流分為緩衝輸入串流和緩衝輸出串流。

- 緩沖輸入串流
  將從輸入串流讀入的位元組/字元資料先存在緩衝區中，應用程式從緩衝區讀取資料；

- 緩沖輸出串流
  在進行資料輸出時先把資料存在緩衝區中，當緩衝區滿時再一次性地寫到輸出串流中。

緩衝串流

- 使用緩衝串流可以減少應用程式與I/O設備之間的訪問次數，提高傳輸效率；
- 可以對緩衝區中的資料進行按需訪問和預處理，增加訪問的靈活性。

緩衝輸入串流

- 緩衝輸入串流包括
- 位元組緩衝輸入串流BufferedInputStream類別
- 字元緩衝輸入串流BufferedReader類別。

1.位元組緩衝輸入串流BufferedInputStream類別

- 先通過實體輸入串流（例如FileInputStream類別）物件逐一讀取位元組資料並存入緩衝區，應用程式則從緩衝區中讀取資料。

BufferedInputStream類別

- 位元組緩衝輸入串流構造方法：
- public BufferedInputStream(InputStream in)
- public BufferedInputStream(InputStream in,int size)
- Size-指定緩衝區的大小。
- BufferedInputStream類繼承自InputStream，所以該類別的方法與InputStream類別的方法相同。

BufferedReader類別

2.字元緩衝輸入別流BufferedReader類別

- 與位元組緩衝輸入別流BufferedInputStream類別在功能和實現上基本相同。
- 它只適用於字元讀入。

BufferedReader類別

字元緩衝輸入別流構造方法：

public BufferedReader(Reader in)

public BufferedReader(Reader in,int sz)

- BufferedReader類別繼承自Reader，所以該類別的方法與Reader類別的方法相同。
- 按行讀取的方法：String readLine()。
- 該方法返回值為該行不包含結束符的字串內容，如果已到達流末尾，則返回 null。

緩衝輸出串流

緩衝輸出串流包括位元組緩衝輸出串流BufferedOutputStream類別和字元緩衝輸出串流BufferedWriter類別。

1.位元組緩衝輸出串流BufferedOutputStream類別
完成輸出操作時，先將位元組資料寫入緩衝區，當緩衝區滿時，再把緩衝區中的所有資料一次性寫到底層輸出串流中。

構造方法：
public BufferedOutputStream(OutputStream out)
public BufferedOutputStream(OutputStream out,int size)
BufferedOutputStream類繼承自OutputStream，所以該類別的方法與OutputStream類別的方法相同。

緩衝輸出串流

2.字元緩衝輸出串流BufferedWriter類別

- 與位元組緩衝輸出串流BufferedOutputStream類別在功能和實現上是相同的。只適用於字元輸出。

BufferedWriter類別的構造方法：

public BufferedWriter(Writer out)

public BufferedWriterr(Writer out,int sz)

- BufferedWriter類別繼承自Writer，所以該類別的方法與Writer類別的方法相同。
- 行分隔符號的方法：String newLine()行分隔符號字串由系統屬性 line.separator 定義。

指定檔案寫入內容並讀出該檔內容3-1

```
package ch10_9;
import java.io.*;
import java.util.Scanner;

public class ch10_9 {
    public static void main(String[] args) {
        File file;
        FileReader fin;
        FileWriter fout;
        BufferedReader bin;
        BufferedWriter bout;
        Scanner scanner = new Scanner(System.in);
        System.out.println("請輸入檔案名，例如d:\\hello.txt");
        String filename = scanner.nextLine();
        try
        {
          file = new File(filename);//創建檔案物件
          if (!file.exists())
          {
            file.createNewFile();//創建新文件
            fout = new FileWriter(file);//創建檔案輸出串流對
          }
```

指定檔案寫入內容並讀出該檔內容3-2

```
else
    fout = new FileWriter(file, true);//創建追加內容的檔案輸出串流物件

    fin = new FileReader(file);//創建檔案輸入串流
    bin = new BufferedReader(fin);//創建緩衝輸入串流
    bout = new BufferedWriter(fout);//創建緩衝輸出串流
    System.out.println("請輸入資料，最後一行為字元'0'結束。");
    String str = scanner.nextLine();//從鍵盤讀取待輸入字串
    while (!str.equals("0"))
    {
        bout.write(str);//輸出字串內容
        bout.newLine();//輸出分行符號
        str = scanner.nextLine();//讀下一行
    }
    bout.flush();//刷新輸出串流
    bout.close();//關閉緩衝輸出串流
    fout.close();//關閉檔輸出串流
    System.out.println("文件寫入完畢！");

    //重新將檔內容顯示出來
    System.out.println("檔案" + filename + "的內容是：");
```

指定檔案寫入內容並讀出該檔內容3-3

```
            while ((str = bin.readLine()) != null)
                System.out.println(str);//讀取檔內容並顯示
                bin.close();//關閉緩衝輸入流
                fin.close();//關閉檔輸入流
            }
            catch (IOException e)
            {e.printStackTrace();}
        }
    }
```

結果:

```
請輸入檔案名，例如d:\hello.txt
hello.txt
請輸入資料，最後一行為字元'0'結束。
test data
0
文件寫入完畢！
檔hello.txt的內容是：
test data
```

隨機串流

- 隨機串流RandomAccessFile類別創建的物件既可以作為輸入串流，也可以作為輸出串流，因此建立一個隨機串流就可以完成讀寫操作。
- RandomAccessFile類別是java.lang.Object根類別的子類別。
- RandomAccessFile類別的實例物件支援對隨機訪問檔案的讀取和寫入。
- 隨機串流可以用於多執行緒下載或多個執行緒同時寫資料到檔案，便於快速完成訪問。

RandomAccessFile類別

該類別中既有讀取操作的方法，也有寫操入作的方法。常用方法：

序號	類型	方法名	功能
1		RandomAccessFile(String name, String mode)	參數name為待訪問的檔案名，參數mode為讀寫模式，常用的值有：“r”以唯讀方式打開檔案，如果進行寫入操作會產生例外；“rw”：以讀寫方式打開檔案
2		RandomAccessFile(File file, String mode)	
3	int	read()	從檔案中讀取一個資料位元組並以整數形式返回此位元組
4	int	read(byte[] b)	從檔案中讀取最多b.length 個資料位元組到b陣列中，並返回實際讀取的位元組數

RandomAccessFile類別

RandomAccessFile類別方法

序號	類型	方法名	功能
5	int	read(byte[] b, int off, int len)	從檔案中讀取len個位元組資料到b陣列中。off為位元組在陣列中存放的位址
6	XXX	readXXX()	從檔案中讀取一個XXX類型資料，XXX包括：boolean，byte，char，short，int，long，float，double
7	void	write(int b)	寫入指定的位元組
8	void	write(byte[] b)	寫入位元組陣列內容到檔案
9	void	writeXXX (XXX v)	寫入指定類型資料到檔案，XXX包括：boolean，byte，char，short，int，long，float，double

RandomAccessFile類別

RandomAccessFile類別方法

序號	類型	方法名	功能
10	void	writeChars(String s)	寫入字串到檔案
11	void	writeUTF(String s)	按UTF_8編碼形式將字串寫入到檔案
12	long	getFilePointer()	獲取檔案的當前偏移量
13	void	seek(long pos)	設置檔案的指標偏移量
14	long	length()	獲取檔案的長度

RandomAccessFile類別

RandomAccessFile類別方法

序號	類型	方法名	功能
15	void	setLength(long newLength)	設置此檔案的長度。 如果 length 方法返回的檔案的現有長度大於 newLength 參數，則該檔案將被截短。超過則加大檔案，多出部分沒有內容
16	int	skipBytes(int n)	跳過輸入的 n 個位元組並丟棄跳過的位元組
17	void	close()	關閉檔案串流，釋放資源

以隨機串流完成檔的讀寫操作2-1

```
package ch10_10;
import java.io.IOException;
import java.io.RandomAccessFile;
public class ch10_10 {
    public static void main(String[] args) {
        try
        {
          RandomAccessFile file = new RandomAccessFile("file", "rw");
          file.writeInt(10);// 占4個位元組
          file.writeDouble(3.14159);// 占8個位元組
          //下麵長度寫在當前文件指標的前兩個位元組處，可用readShort()讀取
          file.writeUTF("UTF字串");
          file.writeBoolean(true);// 占1個位元組
          file.writeShort(100);// 占2個位元組
          file.writeLong(12345678);// 占8個位元組
          file.writeUTF("又一個UTF字串");
          file.writeFloat(3.14f);// 占4個位元組
          file.writeChar('a');// 占2個位元組
          file.seek(0);// 把檔指針位置設置到檔起始處
          System.out.println("—從file檔案起始位置開始讀數據—");
```

以隨機串流完成檔的讀寫操作2-2

```
            System.out.println(file.readInt());
            System.out.println(file.readDouble());
            System.out.println(file.readUTF());
            //將檔指標跳過3個位元組，本例中即跳過了一個boolean值和short值
            file.skipBytes(3);
            System.out.println(file.readLong());
            // 跳過檔中“又是一個UTF字串”所占位元組，
            //注意readShort()方法會移動檔指標，所以不用加2。
            file.skipBytes(file.readShort());
            System.out.println(file.readFloat());
            file.close();
        }
        catch (IOException e)
        {
            System.out.println("檔案讀寫錯誤！");
        }
    }
}
```

結果:

```
—從file檔起始位置開始讀數據—
10
3.14159
UTF字串
12345678
3.14
```

字串亂碼的處理

- 由於編碼格式不同當進行字串讀取的時候，有時會出現“亂碼”的現象。
- 對字符串重新進行編碼可以解決亂碼問題
- 先讀字串：
- String str = in.readLine();
- 再將字串恢復成標準位元組陣列：
- byte [] b=str.getBytes(“iso-8859-1”);
- 最後將位元組陣列按當前機器的預設編碼重新轉化為字串：
- String result=new String(b);
- 指定編碼類型，可以直接給出繁體中文編碼類型：
- String result=new String(b,”BIG5”);

物件串流

- 物件串流是在實體串流基礎上，通過對物件資料的處理和變換，實現物件的永久保存和讀取。
- ObjectInputStream和ObjectOutputStream分別是物件輸入串流類和物件輸出串流類別，它們也是InputStream和OutputStream類別的子類別。
- 通過物件輸出串流，可以把物件寫入到檔案或進行網路傳輸。
- 物件輸入串流類可以從檔案或網路上，把讀取的資料還原成物件。
- 要想實現物件的傳輸，待傳輸的物件要先進行序列化處理，才能保證物件能準確地保存和讀取。

物件的序列化

- 物件的序列化是指把物件轉換成位元組序列的過程，而把位元組序列恢復為物件的過程稱為物件的反序列化。
- 一個類別如果實現了java.io.Serializable介面，這個類別的實例（物件）就是一個序列化的物件。
- Serializable介面中沒有方法。實現了該介面的物件進行輸出時，JVM將按照一定的格式（序列化資訊）轉換成位元組進行傳輸和存儲到目的地。
- 物件輸入串流從檔或網路上讀取物件時，會先讀取物件的序列化資訊，並根據這一資訊創建物件。

物件輸入串流與物件輸出串流

1.物件輸入串流ObjectInputStream類別
實現物件的輸入操作。

構造方法：
public ObjectInputStream(InputStream in)

類別中的方法：
Object readObject()從ObjectInputStream串流中讀取物件。

物件輸入串流與物件輸出串流

2.物件輸出串流ObjectOutputStream類別實現物件的輸出操作。

構造方法：
public ObjectOutputStream(OutputStream out)
類中的方法：void writeObject(Object o)將指定對象o寫入ObjectOutputStream串流中。

創建一個可序列化類別，將該類別的物件寫入到檔案中。
用物件輸入串流讀取並顯示物件資訊3-1

```
package ch10_11;
import java.io.*;
import java.util.Scanner;

public class ch10_11 {
    public static void main(String[] args) {
        try {
             File file;
             FileInputStream fin;
             FileOutputStream fout;
             ObjectInputStream oin;
             ObjectOutputStream oout;
             Scanner scanner = new Scanner(System.in);
             System.out.println("請輸入檔案名，例如d:\\foo.txt");
             String filename = scanner.nextLine();
             file = new File(filename);// 創建檔案物件
             if (!file.exists())
                file.createNewFile();// 創建新檔案
                fout = new FileOutputStream(file);// 創建檔案輸出串流
                oout = new ObjectOutputStream(fout);// 創建物件輸出串流
                Person person = new Person("張三", 20);
```

創建一個可序列化類別，將該類別的物件寫入到檔案中。用物件輸入串流讀取並顯示物件資訊3-2

```
            oout.writeObject(person);
            oout.close();// 關閉物件輸出串流
            fout.close();// 關閉檔案輸出串流
            System.out.println("物件寫入完畢！");
            System.out.println("檔案" + filename + "的內容是：");
            Person object;
            fin = new FileInputStream(file);// 創建檔案輸入串流
            oin = new ObjectInputStream(fin);// 創建物件輸入串流
            try {
                object = (Person) oin.readObject();
                System.out.println("讀取的物件資訊：");
                System.out.println("用戶名：" + object.getName());
                System.out.println("年齡：" + object.getAge());
            } catch (ClassNotFoundException e) {
                System.err.println("讀取物件失敗！");
            }
                oin.close();// 關閉物件輸入串流
                fin.close();// 關閉檔案輸入串流
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

創建一個可序列化類別，將該類別的物件寫入到檔案中。
用物件輸入串流讀取並顯示物件資訊3-3

```
class Person implements Serializable// 物件序列化
{
    private static final long serialVersionUID = 1234567890L;
    String name;
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

結果:

```
請輸入檔案名，例如d:\foo.txt
foo.txt
物件寫入完畢！
檔案foo.txt的內容是：
讀取的物件資訊：
用戶名：張三
年齡：20
```

NIO與IO的區別

- NIO即New IO，這個套件是在JDK1.4中提出，NIO的效率要比IO高很多。
- 在Java API中提供了兩套NIO，一套是針對標準輸入輸出NIO，另一套就是網路程式設計NIO。
- java.io是位元組串流或字元串流進行操作，不在緩衝區。
- java.nio傳輸的資料都存在緩衝區中。
- java.io傳輸是阻塞的，即在開始讀/寫操作之前執行緒一直處於阻塞狀態，不能做其他的事情。
- java.nio是非阻塞的，即執行緒不需要等待資料全部傳輸結束就可以做其他的事情。

NIO的主要組成部分

- NIO提供了三個重要的組成部分：緩衝區Buffers，通道Channels，選擇器selector。初始的io類別提供緩衝和雙向操作等功能。

1.緩衝區Buffers

- 緩衝區Buffer用於緩存待發送/已接收的資料。
- 緩衝區子類別：ByteBuffer、CharBuffer、DoubleBuffer、FloatBuffer、IntBuffer、LongBuffer、ShortBuffer等。
- Buffer抽象類別是它們的父類別，定義了緩衝區訪問的相關屬性和方法。

NIO的主要組成部分

1.緩衝區Buffers

- Buffer的基本屬性有三個：position（位置）、limit（限制）、capacity（容量）。這三個屬性相當於緩衝區中的三個指標標記，用於指出訪問位置，訪問範圍，最大容量等資訊。

NIO的主要組成部分

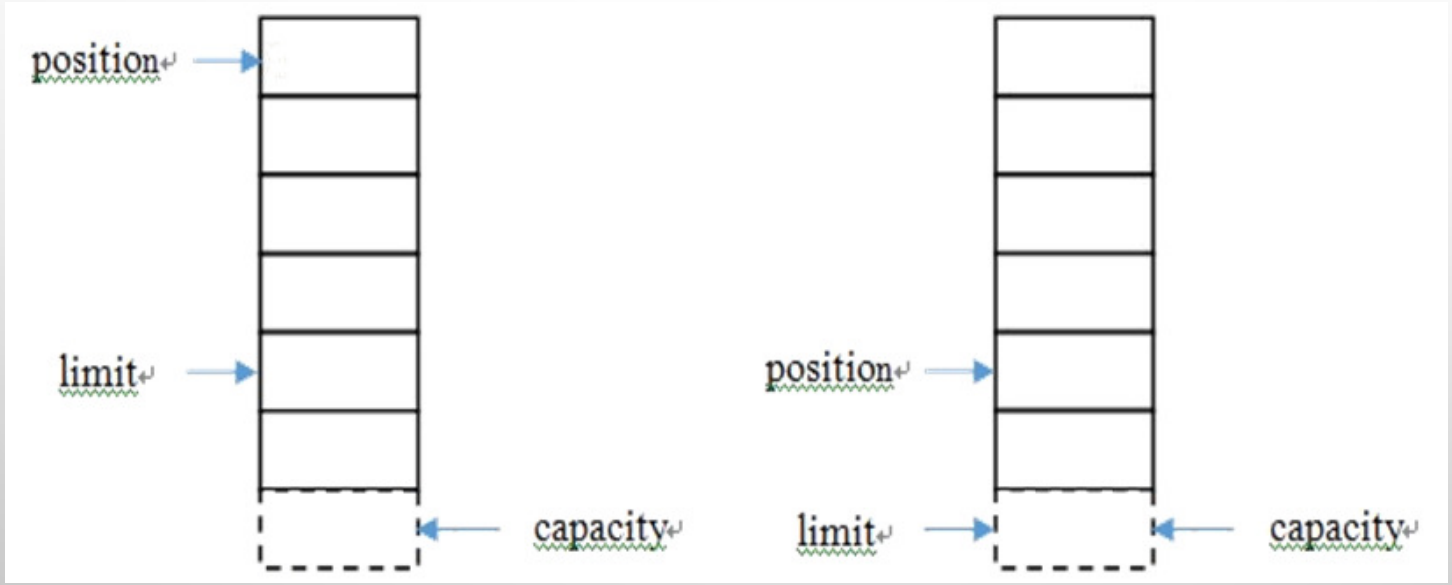
1.緩衝區Buffers

三個屬性說明：

- position：下一個要讀/寫的資料位置，可以從0開始。該值不能大於limit。
- limit：不可以讀/寫的資料起始位置。該值不能大於capacity。
- capacity：緩衝區容量，一旦設定不能修改。
- 三者的大小關係：0≤position≤limit≤capacity

NIO的主要組成部分

1. 緩衝區Buffers



讀取模式　　寫入模式

NIO的主要組成部分

2.通道Channels

- 通道Channels用於創建緩衝區與外部資料來源的連接通道，並實現資料傳輸。

常用的Channel類：

- FileChannel、DatagramChannel、SocketChannel、ServerSocketChannel。利用這幾個通道類，不僅可以實現檔案的傳輸，也可以實現網路TCP、UDP資料包的傳輸。

NIO的主要組成部分

3.選擇器selector

- 選擇器selector 可以讓一個單執行緒多個 Channel。
- 這種應用在一些特殊情況下回非常方便，比如網路聊天室。
- 要使用Selector，首先Selector註冊Channel，然後調用它的select()方法。這個方法會一直阻塞到某個註冊的通道有事件就緒。一旦這個方法返回，執行緒就可以處理這些事件，如讀入新資料等。

Buffers

Buffers緩衝區，一共有10個類別，Buffer類別是其他類別的父類。這裡只介紹兩個類Buffer和ByteBuffer。

1.Buffer類別

- Buffer類別作為緩衝區類的根類別，重點定義了緩衝區的結構和基本方法。

Buffer類別方法

- public final int capacity()
  返回此緩衝區的容量。當創建一個緩衝區後，其容量就固定不變了。

- public final int position()
  返回此緩衝區的position指針位置。該值表示下一個可處理的
- 資料位置。該值初始為0，隨著讀寫操作自動後移。

- public final Buffer position(int newPosition)
- 設置緩衝區新的position指針位置。新值必須是非負數，不能大於當前限制值。
- 如果該緩衝區設置了標記，並且標記位置大於新位置，則該標記被丟棄。

Buffer類別方法

- public final int limit()
  返回緩衝區的limit限制值。表示當前讀/寫操作的最大緩衝區範圍。

- public final Buffer limit(int newLimit)
  設置此緩衝區的新限制值。新限制值必須為非負且不大於此緩衝區的容量。

- public final Buffer clear()
  清除此緩衝區。該方法通常在通道準備讀取資料到緩衝區時先行呼叫。

Buffer類別方法

- public final Buffer flip()
- 反轉此緩衝區。該方法通常在準備將緩衝區中的資料寫入到通道時先行呼叫。

- public final Buffer rewind()
- 重置此緩衝區。該方法通常用於重新完成讀/寫操作。

- public final int remaining()
- 返回當前位置與限制之間的元素數。用於返回緩衝區中的剩餘元素數量。

Buffer類別方法

- public abstract boolean isReadOnly()
- 判斷此緩衝區是否為唯讀緩衝區。

- public final Buffer mark()
- 在此緩衝區的當前position位置設置標記。

- public final Buffer reset()
- 將此緩衝區的position值重置為以前標記的位置。

ByteBuffer類別

- ByteBuffer類別用於定義一個以位元組為單位的緩衝區，實現資料存儲和訪問。

ByteBuffer類別方法

返回類型	方法名	方法功能
static ByteBuffer	allocate(int capacity)	分配一個新的位元組緩衝區。
CharBuffer	asCharBuffer()	創建一個位元組緩衝區作為char緩衝區的視圖。
ByteBuffer	asReadOnlyBuffer()	創建一個新的唯讀位元組緩衝區，共用此緩衝區的內容。
ByteBuffer	compact()	壓縮此緩衝區，將緩衝區當前位置與其限制之間的位元組複製到緩衝區的開頭。

ByteBuffer類別方法

返回類型	方法名	方法功能
byte	get()	讀取該緩衝區當前位置的位元組，然後增加位置。
ByteBuffer	put(byte b)	將給定位元組寫入當前位置的緩衝區，然後增加位置。
ByteBuffer	slice()	創建一個新的位元組緩衝區，其內容是此緩衝區內容的共用子序列。其容量和限制是此緩衝區中剩餘的位元組數。
ByteBuffer	wrap(byte[] array)	將一個位元組陣列封裝到緩衝區中。一方的內容修改會影響另一方。

Buffer簡單應用

```
package ch10_12;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
public class ch10_12 {
    public static void main(String[] args) {
        ByteBuffer buffer1 = ByteBuffer.allocate(40);
        ByteBuffer buffer2 = ByteBuffer.allocate(40);
        String str1 = "Java NIO reader";
        byte [] b1 =str1.getBytes();
        char[] str2 = "Java NIO writer".toCharArray();
        CharBuffer cbuffer = buffer2.asCharBuffer();
        for (byte b:b1){
            buffer1.put(b);
        }
        System.out.print("str1=");
        System.out.println(str1);
        System.out.print("str2=");
        System.out.println(str2);
        System.out.print("b1=");
        System.out.println(b1);
        System.out.print("buffer1=");
        System.out.println(buffer1);
    }
}
```

結果:

```
str1=Java NIO reader
str2=Java NIO writer
b1=[B@2f92e0f4
buffer1=java.nio.HeapByteBuffer[pos=15 lim=40 cap=40]
```

Channels類別

Channels通道提供了與通道有關的若干個介面和類別。

1.Channels類別

- Channels類別定義了支援java.io包的串流類別與nio套件的通道類別的交互操作的靜態方法。

返回類型	方法名	方法功能
ReadableByteChannel	newChannel(InputStream in)	構造從給定串流讀取位元組的通道。
WritableByteChannel	newChannel(OutputStream out)	構造一個將位元組寫入給定串流的通道。
InputStream	newInputStream(ReadableByteChannel ch)	構造從給定通道讀取位元組的串流。

Channels類別方法

返回類型	方法名	方法功能
OutputStream	newOutputStream(WritableByteChannel ch)	構造將位元組寫入給定通道的流。
Reader	newReader(ReadableByteChannel ch, String csName)	根據給定的字元集編碼構造一個來自給定位元組通道的讀字元串流。
Writer	newWriter(WritableByteChannel ch, String csName)	根據給定的字元集編碼構造一個寫入給定位元組通道的寫字元串流。

從鍵盤讀取字串顯示在螢幕上，輸入“exit”結束

```
package ch10_13;
import java.nio.ByteBuffer;
import java.nio.channels.*;
public class ch10_13 {
    public static void main(String[] args) throws Exception {
        System.out.println("請輸入資料若輸入\"exit\"則結束");
        ReadableByteChannel in = Channels.newChannel(System.in);// 創建一個讀通道
        WritableByteChannel out = Channels.newChannel(System.out);// 創建一個寫通道
        ByteBuffer buff = ByteBuffer.allocate(1024);// 創建一個1024位元組的位元組緩衝區
        while (in.read(buff) != -1) {// 將讀通道的資料讀到緩衝區
            buff.flip();// 翻轉緩衝區
            String str = new String(buff.array()).trim(); // trim()刪除空白字元
            if (str.equals("exit")) {// 若輸入"exit"則結束
                in.close();
                out.close();
                break;
            }
            System.out.print("輸入資料=");
            out.write(buff);// 將緩衝區的資料寫入到寫通道
            while (buff.hasRemaining()) {// 查詢緩衝區是否還有剩餘資料
                out.write(buff);
            }
            buff.clear();// 清空緩衝區，準備寫入下一批資料
        }
    }
}
```

結果:

```
請輸入資料若輸入"exit"則結束
223
輸入資料=223
```

FileChannel類別

2.FileChannel類別

FileChannel類別用於創建可以用於讀，寫，映射和操作檔案的通道。

FileChannel類別方法

返回類型	方法名	方法功能
static FileChannel	Open(Path path,OpenOption... options)	打開或創建文件，並返回文件通道
int	Read(ByteBuffer dst)	從該通道讀取到給定緩衝區的位元組序列。
long	size()	返回此通道文件的當前大小。
int	Write(ByteBuffer src)	從給定的緩衝區向該通道寫入一個位元組序列，返回寫入的位元組數。

顯示所讀文字檔內容，並向檔中寫入讀取檔的時間

```
package ch10_14;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.util.Date;

public class ch10_14 {
    public static void main(String[] args) throws Exception {
        RandomAccessFile file = new RandomAccessFile("data.txt", "rw");// 建立檔物件
        FileChannel channel = file.getChannel();// 建立文件通道
        ByteBuffer buf = ByteBuffer.allocate(128);// 設置位元組緩衝區
        System.out.println("文件大小：" + channel.size());// 顯示檔大小
        while (channel.read(buf) != -1) {// 讀取檔內容並顯示
            buf.flip();// 準備緩衝區讀取
            String str = new String(buf.array()).trim();
            System.out.println(str);
        }
        Date time = new Date();
        String newData = "read file time is:" + time;
        buf.clear();// 準備緩衝區寫入
        buf.put(newData.getBytes());// 向緩衝區寫入當前時間
        buf.flip();// 準備緩衝區讀取
```

顯示所讀文字檔內容並向檔中寫入讀取檔的時間

```
            while (buf.hasRemaining()) {
                channel.write(buf);// 讀取緩衝區內容送入通道
            }
            channel.close();// 關閉通道
        }
    }
```

結果:

文件大小：61
伊尼舍林的女妖
白蓮花大飯店
劫婚大作戰