國立屏東大學 資訊工程學系 物件導向程式設計

4. Basic Input and Output

本章將透過簡單的範例程式，帶您瞭解Java語言的基本輸入與輸出處理。

4.1 Java SE API

過去當我們在學習C語言時，很大一部份的學習工作是關於functions。由於C語言提供許多已設計好的functions。我們可以使用這些function來完成許多的程式設計工作。同樣地，Java語言也提供許多事先設計好的程式碼，不過這些程式碼是以類別的型式存放在java class library(類別庫)裡。要把Java語言學習好，很大部份的功課應該放在這些類別的學習上。

<note tip> 伴隨JDK的安裝，我們已經取得了Java語言的class library(類別庫)。為方便學習，請不要忘記也下載JDK documentation。 </note>

Java語言的class library只是一個概念性的稱呼，其正式名稱為Java SE API。請參考figure 1。Java SE API是許多Java相關產品與技術的基石，任何一位Java Programmer都應該熟悉它。

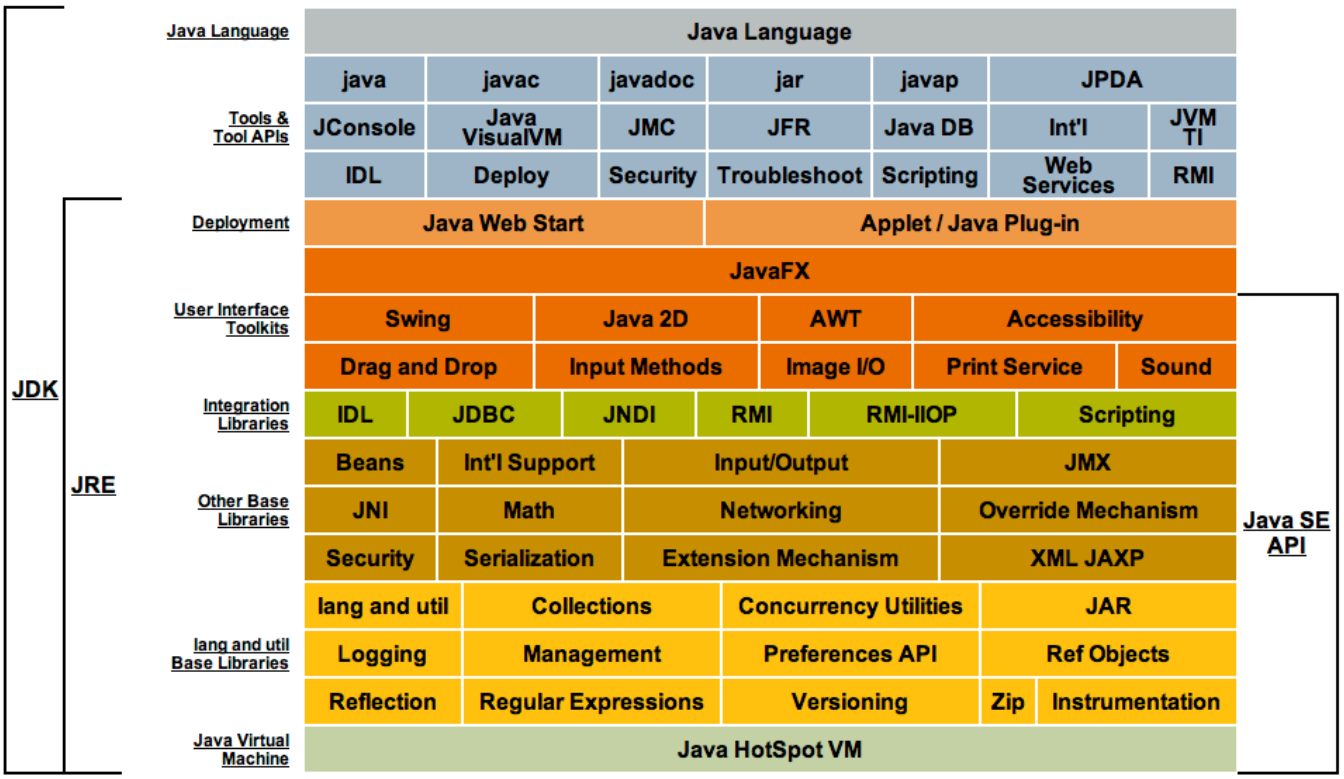

Fig. 1

在Java SE API中，大量的類別是以package方式管理的，每個package又可以擁有子package。就像是目錄還可以擁有子目錄一樣，我們使用。dot notation。來描述類別所屬的package。例如Vector是一個Java SE ADI中的類別，它被收錄在java package中的util子package中，因此其完整路徑名稱為

□java.util.Vector□(可讀做「在java裡面的util裡面的Vector□)□

在Java程式中，如果要使用某一個定義在Java SE API中的類別，例如Vector□你有以下兩種做法可以選擇：

1. 使用完整路徑名稱
    - 每次在程式中使用Vector 類別時，都以java.util.Vetor稱呼它
2. 以import載入package□並直接使用類別名稱
    - 先在程式開頭處以□import java.util.Vector;□□載入這個類別
    - 以後在程式中使用Vector 類別時，直接稱呼為Vector即可

若想要載入某一package中的所有類別，例如java.util內的所有類別，則可以□import java.util.*;□來達成，其中「*」為全部的意思。 此外□Java語言預設為將java.lang package內的所有類別載入，因此您不必再以□import java.lang.*;□來載入它們。

4.2 基本輸出

我們在前一章的HelloWorld.java中，已經使用了System.out.println()來輸出“Hello World!”字串。事實上，這個println()是out物件的一個method□用以將資料輸出到「標準輸出」(也就是console)□而out物件是屬於System這個類別的一個field□被宣告為java.io.PrintStream 類別的一個object□

<note tip>

關於method與field

我們將留待日後解釋了物件導向的概念後，再詳述何謂method與field□在此請先暫時將method與field視為C語言中的function與variable□

</note>

因此，要搞懂System.out.println()在做些什麼，應該要去查看定義在java.io.PrintStream的println() method□結果您有沒有發現有好多個println() method!□如table 1□

prototype	**meaning**
void println()	Terminates the current line by writing the line separator string.
void println(boolean x)	Prints a boolean and then terminate the line.
void println(char x)	Prints a character and then terminate the line.
void println(char[] x)	Prints an array of characters and then terminate the line.
void println(double x)	Prints a double and then terminate the line.
void println(float x)	Prints a float and then terminate the line.
void println(int x)	Prints an integer and then terminate the line.
void println(long x)	Prints a long and then terminate the line.
void println(Object x)	Prints an Object and then terminate the line.
void println(String x)	Prints a String and then terminate the line.

Tab. 1: println的多個版本

同一個method名稱，但提供不同參數的版本，讓我們可以針對不同資料呼叫同名的method來完成處理，這種在物件導向程式中允許同一個calss可以定義「同名異式」的method□就稱為□Method Overloading(方法重載)」。關於這點以後還會有詳細的說明。

你可以使用這個method來將各種資料輸出到各種地方，其中java.lang.System裡面的out就是java.io.PrintStrem類別的一個物件(你先暫時把它想像成out的型態是PrintStream)同時out已經幫我們建立好輸出到標準輸出的管道，因此透過使用java.lang.System.out來呼叫println()就可以將資料輸出到標準輸出並加上一個換行(也就是\n)

以下是一些範例，記住java.lang.*已經預設載入了，所以你可以直接使用System

```
System.out.println("Hello World!"); //可輸出一個字串

System.out.println("Hello " + "World!"); //兩個字串可以用+串接

System.out.println( 2+3 ); //輸出 5

System.out.println("" + 2 + 3 ); //輸出"23", 因為開頭有一個(空)字串，所以後面的資料會被轉換成字串
```

再查看一下文件，您還可以發現PrintStream類別還定義了print() method當然也是提供多個「同名異式」的版本print()與println()的差別在於println()輸出後會換行，但print()不會。

Java語言自JDK 1.5之後，還貼心(對C語言的程式設計師而言)地在PrintStream類別中提供了printf()其用法與C語言中的printf()一致(其用法可參閱C語言的格式化輸出與輸入)，因此下面這行程式碼也是正確的。

```
System.out.printf("The value of x is %5d.\n", x);
```

4.3 輸入

如果你熟悉C語言，您可能會想知道Java語言是否有提供類似C語言的scanf()的method? 很抱歉，截至目前為止(JDK 1.7)並沒有這樣的method或類似的作法。不過Java語言還是提供了好幾種方法，可以取得使用者自鍵盤的輸入。本節將介紹使用java.util.Scanner類別來取得使用者輸入的方法。

Scanner類別同樣是自JDK 1.5開始提供，可先宣告其物件並與標準輸入管道建立連結：

```
Scanner sc = new Scanner(System.in); //建立一個與System.in連結的Scanner類別的物件, 命名為sc
```

接下來，就使用其nextXXXX() method來取得使用者輸入的資料，其中XXXX表示不同的資料型態或類別，請參考table 2Scanner也提供對應於nextXXXX()的一組hasNextXXXX()用以檢查是否還有XXXX型態的資料可以被讀取。

prototype	**meaning**
BigDecimal nextBigDecimal()	Scans the next token of the input as a BigDecimal.
BigInteger nextBigInteger()	Scans the next token of the input as a BigInteger.
BigInteger nextBigInteger(int radix)	Scans the next token of the input as a BigInteger.
boolean nextBoolean()	Scans the next token of the input into a boolean value and returns that value.

prototype	**meaning**
byte nextByte()	Scans the next token of the input as a byte.
byte nextByte(int radix)	Scans the next token of the input as a byte.
double nextDouble()	Scans the next token of the input as a double.
float nextFloat()	Scans the next token of the input as a float.
int nextInt()	Scans the next token of the input as an int.
int nextInt(int radix)	Scans the next token of the input as an int.
String nextLine()	Advances this scanner past the current line and returns the input that was skipped.
long nextLong()	Scans the next token of the input as a long.
long nextLong(int radix)	Scans the next token of the input as a long.
short nextShort()	Scans the next token of the input as a short.
short nextShort(int radix)	Scans the next token of the input as a short.

Tab. 2: Scanner class's nextXXXX() methods

當不再使用Scanner取得使用者輸入時，請記得呼叫close() method將已連結好的輸入管道關閉。我們提供一下範例程式如下：

```
import java.util.Scanner;

public class ScannerExample
{
  public static void main(String[] args)
  {
      Scanner sc = new Scanner(System.in);

      System.out.print("Please input a number: ");

      int x = sc.nextInt();
      System.out.println("The value of x is " + x);

      System.out.print("Please input another number: ");

      int y = sc.nextInt();
      System.out.println("The value of y is " + y);

      sc.close();
  }
}
```

要特別注意的是，一但你將建立在System.in上的Scanner的物件關閉後，你也同時將System.in關閉了。日後若想再透過System.in來取得使用者的輸入時，就會發生Exception in thread “main” java.util.NoSuchElementException的例外錯誤。