第 1 章 ： Java是什麼

1－1、Java的簡介

- Java的產生
- Java的歷史
- Java的特性

1－2、第一個Java程式

- Java的程式發展步驟
- 範例程式及發展步驟

1－3、Java程式的基本架構

1 - 1 、 Java的簡介- Java的產生

- Java語言是美國昇陽電腦公司(Sun Micro System) 所發展出的程式語言。
- 自1991年的綠色計畫(Green Project) 開始，Java的影響力就一直不斷的持續上升。
- Sun Microsystem於2010年被Oracle Corporation 收購。並在獲得歐盟和美國相關機構的審批後完成。

1 - 1 、 Java的歷史

Java的發展歷史及時間請參考:

- http://www.oracle.com
- https://www.oracle.com/java/technologies/

1 - 1 Java的簡介 - Java的特性(1)

簡單(Simple)

- 程式碼結構簡單清晰，易於理解和維護。
- 類別和方法的設計簡單，每個類別方法只負責一個功能。
- 程式碼避免使用過於複雜的結構和算法，以便提高可讀性和維護性
- 使用標準庫和框架。

1 - 1 Java的簡介- Java的特性

物件導向(Object-Oriented)

- 物件導向是資訊界廣為使用的觀念及技術。
- 物件導向許多良好的特性，例如:
  - 封裝性(Encapsulation) 可以使物件的介面定義明確
  - 繼承性(Inheritance)可以增加軟體的可再用性(Reusability)
  - 方法複寫(override)
- 實現物件導向的概念及其各種良好的特性是Java的設計理念之一

1 - 1 、 Java的簡介- Java的特

分散式(Distributed)

- 電腦網路的發展使得資訊應用朝向分散式的環境發展。
- Java具有網路功能的程式庫(Library)，其中包含與如HTTP、URL等TCP/IP網際網路通訊協定整合的能力。
- 利用Java並配合網路程式庫所開發的程式，能輕易的在網路上開啟、連結並使用物件方便的使用各種網路資訊。

1 - 1 Java的簡介-Java的特性

強韌性(Robust)

- 由Java所撰寫出的程式要能在各種情況下執行，而且必需具有高的穩定性。
- 例如:Java在制訂時即加入了能排除記憶體被覆寫(Overwriting Memory)和資料毀損(Corrupting Data)的相關處理機制 。

1 - 1 Java的簡介 Java的特性

安全性(Secure)

- Java設計用於網路及分散性的環境中，所以安全性是一個很重要的考量。

- Java擁有從簡單到複雜的安全保護措施，能較有效地防止病毒的侵入和破壞行為的發生。

1 - 1 、 Java的簡介 Java的特性

可攜性(Portable)

- Java的各組成程式庫及以Java環境所產生的程式碼都具備有相當好的可攜性，例如:
  - **Java Runtime** 可以很容易的在各種不同電腦平台上建立Java的環境。
  - Java的編譯器就以Java所寫成，可以移植到各種平台。
  - Java程式庫中更定義了可移植的介面，可以在 Linux、Windows和Mac各種平台上的實作使用介面。Java發展出來的程式具有好的可攜性，可以在各種電腦上執行。

1 - 1 、 Java的簡介 Java的特性

直譯式特性(Interpreted)

- Java具有一個直譯器(Interpreter)可以直接在任何已安裝直譯器的機器上執行Java位元碼(Byte Code)

1 - 1 、 Java的簡介 Java的特性

高效能(High Performance)

- Java在執行時能將位元碼迅速地轉換成機械碼(Machine Code)再加以執行。

- 雖然Java增加了許多的特性及功能且是在執行時才把位元碼轉換成機械碼，但它的執行效能和以C與C++等傳統語言所開發出來的程式幾乎與沒有分別。

1 - 1 、 Java的簡介-Java的特性

多執行緒(Multithreaded)

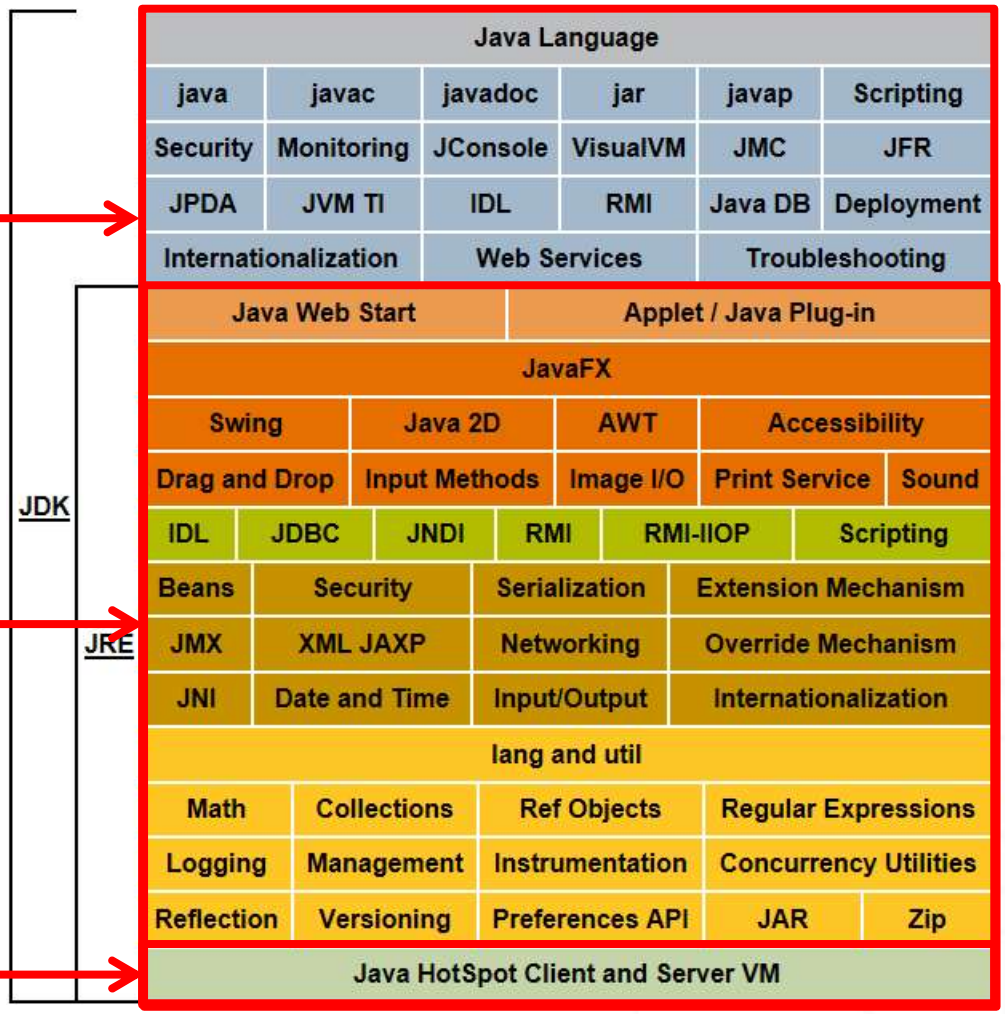
- 在Java中，一個執行緒就代表一個執行工作。
- 在我們周遭的真實世界中，許多工作會同時發生而需要同時間進行。
- 若要如同真實世界一般，設計一個同時處理許多事件的Java程式，就要把Java制定成具有多執行緒的能力。

Java SE JDK架構

JDK Java標準版開發工具：
提供編譯器，及相關的開發、執行、測試等工具

JRE Java執行環境：
提供Java應用程式相關資源與執行程式的環境
如標準函式庫、Java虛擬機器

JVM Java虛擬機器：
將編譯過的Java程式轉換為平台相依的原生碼

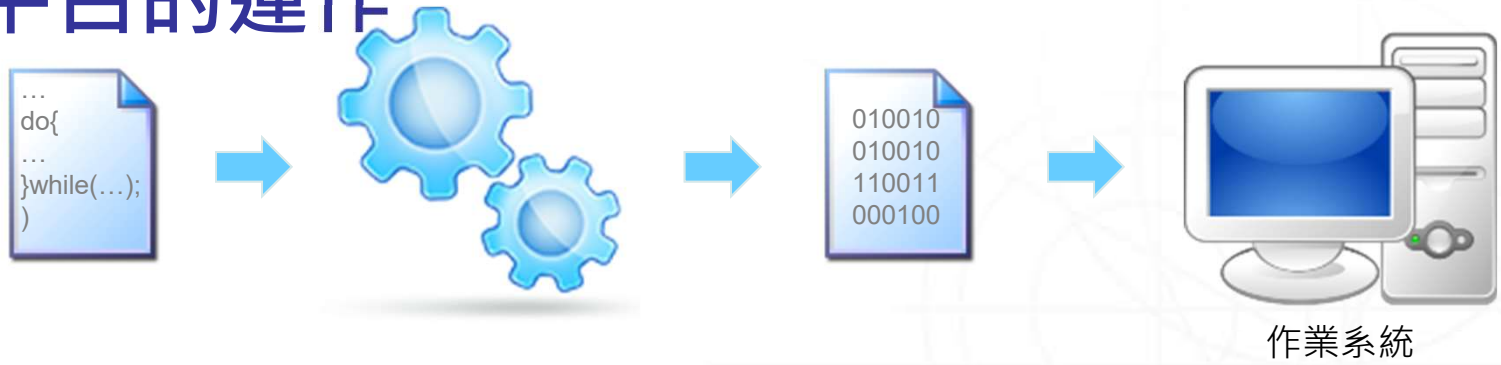


第一個Java程式

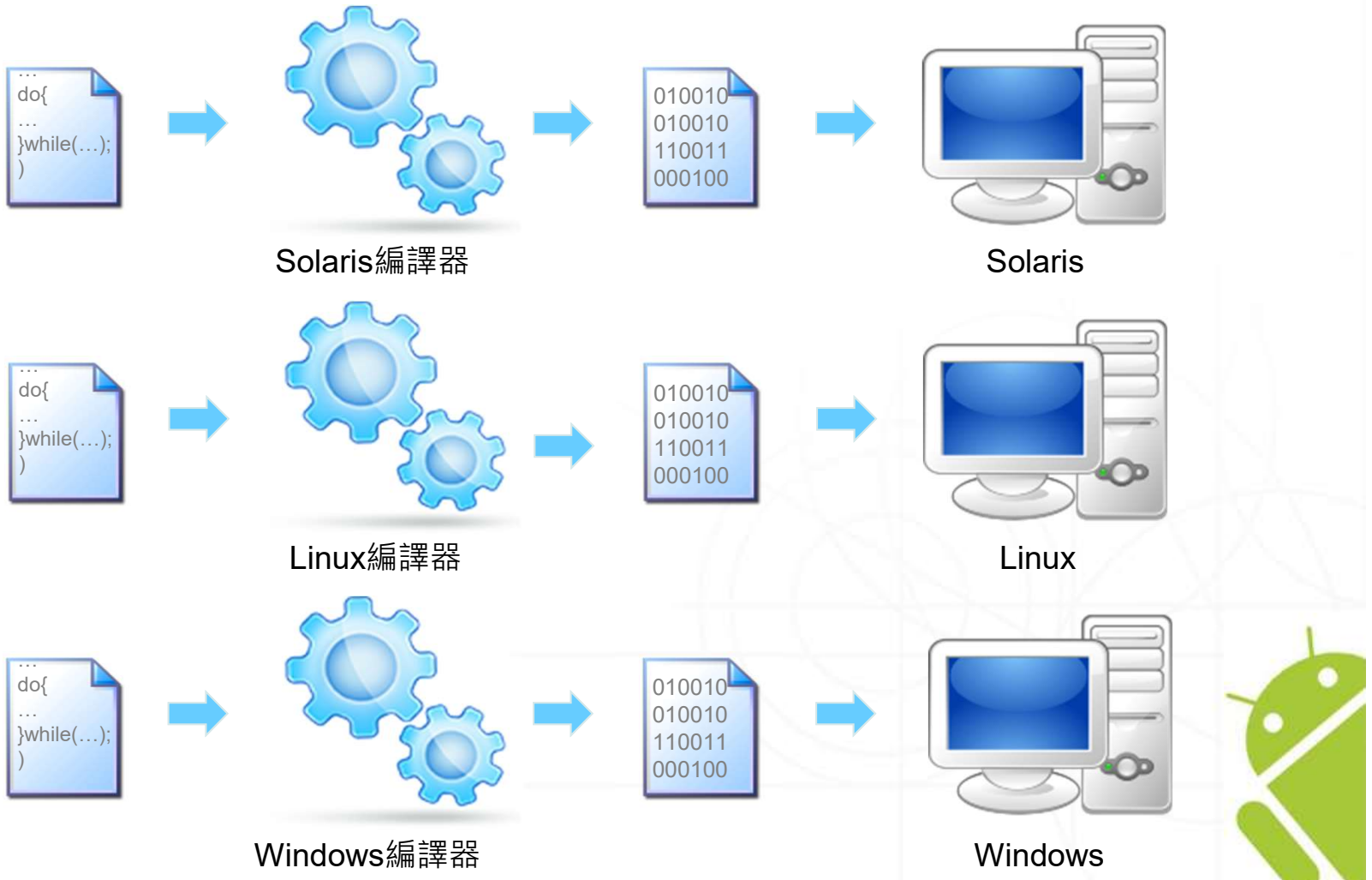
```
01 //檔名: HelloWorld.java
02 public class HelloWorld {
03
04     public static void main(String[] args) {
05         System.out.println("Hello!World");
06     }
07 }
```

認識JVM

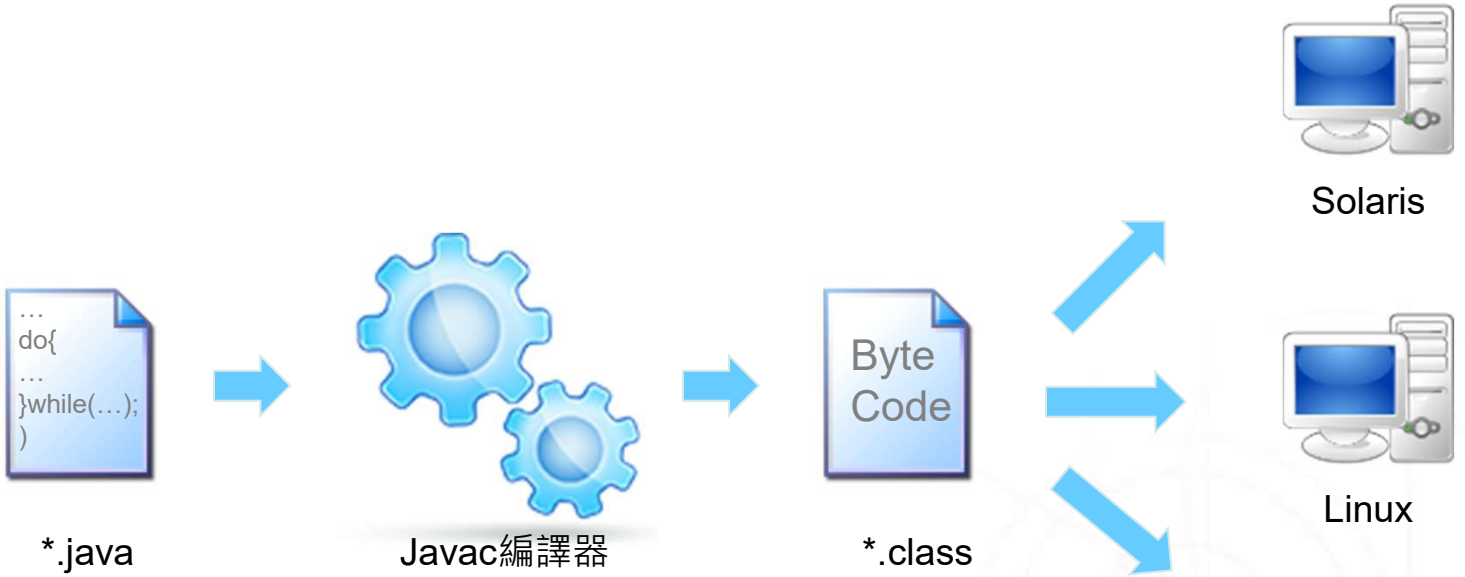
- Java Virtual Machine，Java虛擬機器
- 目的編譯一次可在不同作業系統執行的跨平台的運作



認識JVM


…
do{
…
}while(…);
)
Solaris編譯器
010010
010010
110011
000100
Solaris
Linux編譯器
Linux
Windows編譯器
Windows

認識JVM



撰寫一次，到處執行(write once, run anywhere)

第一個Java程式-（對應例範例開發步驟）

- 步驟一：撰寫程式
  - 我們可以使用編輯器vscode，其它只要是能多編輯文字檔的工具也可以使用，但請注意不要使用會加入控制訊息在檔案中的編輯工具(如MS-WORD)。
- 步驟二：編譯程式
  - 以編譯器將原始檔轉換成位元碼檔Bytecode。
- 步驟三：執行程式
  - 以執行器執行Bytecode。延續步驟二的環境，執行我們的範例程式。若成功在螢幕上會出現 "Java is a good language!"，
- 步驟四：除錯及修改程式（Debug）
  - 若沒有就是不成功。請進入除錯及修改的流程，檢查我們執行的這三個步驟是否正確。

Java程式的基本架構-（範例程式）

```
01  /** This program display a sample
02  String on computer screen !
03  */
04  class FirstApp{
05        public static void main(String[ ] args){
06       System.out.println( “Java is a good language” );
07  }
08 }
```

Java程式的基本架構-（範例程式說明）

```
01  /** This program display a sample
02  String on computer screen !
03  */
```

程式中第1~3行是程式的註解部份。所謂程式的”註解”電腦不會去執行它，它是用來給程式設計者作為說明的地方，日後設計者本人或其它人員在查看原始碼時，可以由註解瞭解程式寫作時的一些情形。

Java程式的基本架構-（範例程式說明）

```
04  class FirstApp{
```

- 第4行: class FirstApp{ 這一行是在宣告（告訴編譯器）從這一行是一個名字為FirstApp “類別”的開始，這個類別會以一個配對”}” 作為結束(在範例程式中為第八行)。

Java程式的基本架構-（範例程式說明）

```
05        public static void main(String[ ] args){
```

- 第5行:public static void main(String[] args){
- 方法(Method)是附屬在類別中，使電腦依方法所撰寫的命令完成指定的工作。
- 一個類別中可以包含有許多個方法，所以執行程式的類別內容，就要宣告這方法。

Java程式的基本架構-(範例程式說明)

```
06      System.out.println( “Java is a good language” );
```

- 第 6 行 : System.out.println(“Java is a good language”);
- 它是包含在類別FirstAPP中的方法main中，所以會在一開始執行這個應用程式時會被執行起來。
- 它會將包含在” “中的字串（範例中為 “Java is a good language” ）輸出在系統的標準輸出（在個人電腦上若沒有更改就為電腦螢幕）。

第 2 章、程式語言基本觀念及資料

1、變數與資料型態

2、常數

3、運算式、運算子和運算元

1、變數與資料型態

- 變數名稱(簡稱變數Variable):
  - 資料項目的名稱。
- 變數型態或稱為資料型態(Data Types):
  - 所儲存資料的型式。
- 變數宣告（Declare）:
  - 預先說明變數的名字、型態及大小。
  - 把“變數”想像成日常生活中用來存放東西的箱子，存放的內容就是資料的值。
  - 在使用箱子前，我們需要先知道箱子的名字、大小，箱子是用來裝甚麼樣子的東西，然後取得所需要數量的箱子，這個過程就是變數宣告。

關鍵字(Keyword)

- 在宣告變數時，變數名稱不可以用到系統已內定的保留字(Reserved Words)。

- 保留字是在電腦語言中保留下來，用以代表特殊意義的用字。程式設計時要避免把保留字當成變數名稱，如此才不會與系統產生混淆導致程式執行時發生錯誤。

Java程式所使用的保留字

abstract	double	int	strictfp	do
boolean	else	interface	super	while
break	extends	long	switch	volatile
byte	final	native	synchronized	short
case	finally	new	this	static
catch	float	package	throw	import
char	for	private	throws	instanceof
class	goto	protected	transient	default
const	if	public	try	
continue	implements	return	void	

變數宣告的語法

- 【語法】 資料型態　變數名稱[=初始值]；
  - 用來告訴電腦在程式中所會用到變數的三件事情:
    - 型態
    - 名稱
    - 〔初始值〕

    (註:語法說明內的中括號[ ]，在習慣上表示撰寫程式時，可以存在或不存在的部分)
- 【語法說明】
  - 變數宣告中的“資料型態”，是用以指定變數所要儲存之資料值的類型;
  - “變數名稱”是指定變數的名字，如Payment；
  - “[=初始值]”是可以存在或不存在的部份

基本資料型態

- Java語言預先定義好可以直接使用的資料型態，稱為”基本資料型態“(Primitive Data Types)。
- 相對於基本資料型態有一種非基本的參考資料型態，會在後續的章節中介紹。
- Java語言有八種基本資料型態,分成三類:
  - 整數(integer)
  - 浮點數(floating numbers)
  - 其他(others)

基本資料型態表

基本資料型態名稱	佔用記憶體長度	最小值	最大值	預設值
整數(integer)				
byte（位元組）	8-bit	-128	+127	(byte)0
short（短整數）	16-bit	-2^{15}	$+2^{15}-1$	(short)0
int（整數）	32-bit	-2^{31}	$+2^{31}-1$	0
long（長整數）	64-bit	-2^{63}	$+2^{63}-1$	0L
浮點數(floating number)				
float（浮點數）	32-bit	IEEE754	IEEE754	0.0f
double（倍精數）	64-bit	IEEE754	IEEE754	0.0d
其他(others)				
char（字元）	16-bit	‘\u0000’	‘\uFFFF’	‘\u0000’ (null)
boolean（布林變數）	1-bit	-	-	false

Java基本資料型態說明(1、整數)

第1類、整數

包含byte(位元組)、short（短整數）、int（整數）、long（長整數)。可以包含正整數、負整數及0，由所佔用的記憶體大小，對應出可以表示數值極限的大小。

- byte:
  - 的大小為8位元，用來存放8個位元長度的數字。
  - 所以可表示2的8次方種可能值，對應出來的值為自最小值為-128到最大值為127共256個值，其中預設值為0 。
- short、int、long:
  - 可以存放整數，差異為該資料變數的存放空間大小不同，空間大者可以表示的範圍值較大，但在儲存及執行時所佔用的電腦空間就較大且執行時會耗用較多的時間。

字面值 (Literal value)

- `System.out.println(100);`
- 這 100 就是所謂的 字面值(Literal value)
- 也就是一般俗稱的常數(Constants)

- 整數有四種型態: byte short int long
- 這 100 算是哪一種整數型態的字面值?

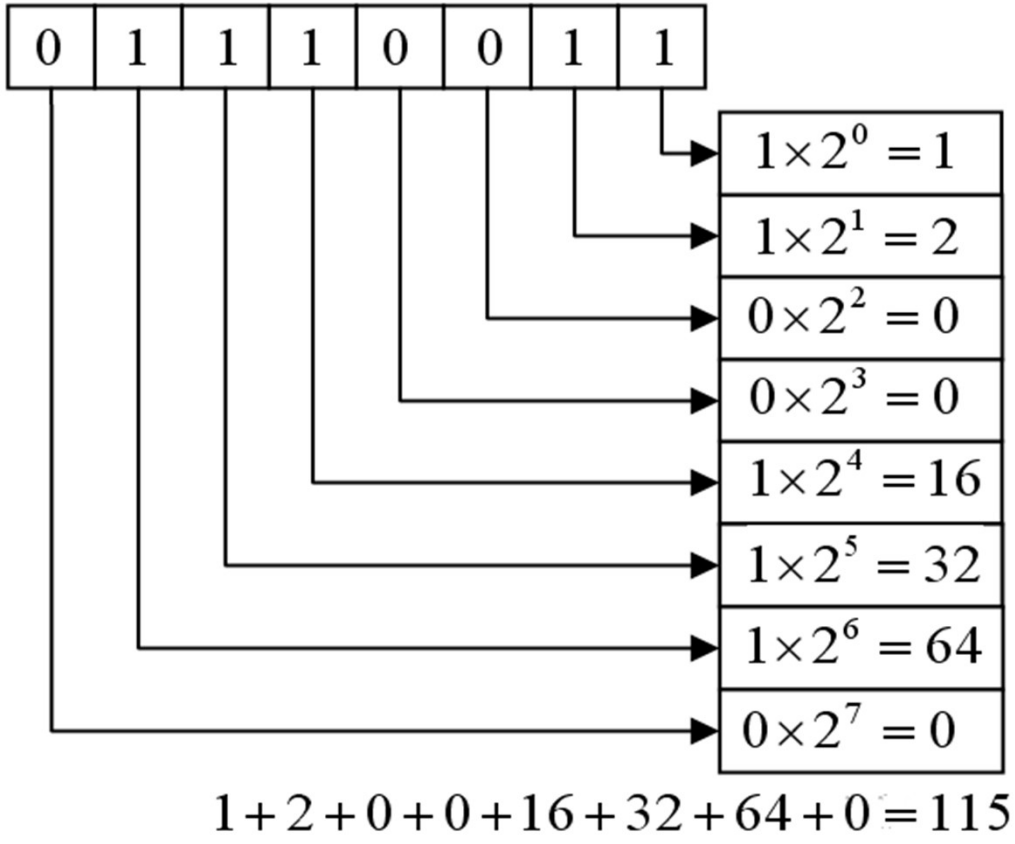
Java 只有提供兩種整數字面值的寫法

100	為 int 字面值
100L	為 long 字面值
3.14	為 double 字面值
3.14F	為 float 字面值

十進位、二進位、十六進位

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

二進位轉十進位



$$1+2+0+0+16+32+64+0=115$$

Java基本資料型態說明(2、實數)

第2類、浮點數

表示含有小數點數值的資料，包含有float（浮點數）、double（倍精數）二種資料型態。

- float(浮點數-floating point):
  - 用來表示含有小數數值的方法
- Double(雙精密度的浮點數):
  - 通常在科學運算上需要用到數值很大或是精確度很高的資料，float型態的資料不能符合這項需求時，就可以使用double的資料型態增加變數可表示及計算的準確度。

Java基本資料型態說明(3、其它)

包含有有char（字元）及boolean（布林）兩種。

- char(字元):
  - 存放十六位元的國際碼（unicode）內碼
  - 當Java程式在程式中要求將此資料項目以字元型態（char）表現出來時，電腦會透過一個對照表將所代表字形列印出來。
  - char佔記憶體的儲存空間大小為16位元，最小值為“\u000 ”；最大值為“\uFFFF”，也就是國際碼（unicode）的最小值與最大值。
- boolean(布林型態):
  - 表示一個條件成立與否的情形，所以它能表示的值只有代表成立的真（true）及不成立的偽（false）兩種，
  - 也可代表boolean變數或資料經過“邏輯運算”- 及(and)、或(or)、否定(not)等所得到的值。
  - 也可以代表“比較運算”-大於(>)、小於(<)及等於(==)等的運算結果。

用 Unicode編碼 來指定字元

```
System.out.println( 'A' );
System.out.println( '\u41' ); //效果同上
```

Ex2_1 Java基本資料型態範例

```
class DemoVariables {
  public static void main(String args[]) {
        // integers
        byte largestByte = Byte.MAX_VALUE;  //位元最大值
        short largestShort = Short.MAX_VALUE;          //短整數最大值
        int largestInteger = Integer.MAX_VALUE;        //整數最大值
        long largestLong = Long.MAX_VALUE;  //長整數最大值
        // real numbers
        float largestFloat = Float.MAX_VALUE;          //浮點數最大值
        double largestDouble = Double.MAX_VALUE;       //倍精數最大值
        // other primitive types
        boolean aBoolean = true;  //布林運算值為真
        // display them all
        System.out.println("The largest byte value is " + largestByte);
        System.out.println("The largest short value is " + largestShort);
        System.out.println("The largest integer value is " + largestInteger);
        System.out.println("The largest long value is " + largestLong);
        System.out.println("The largest float value is " + largestFloat);
        System.out.println("The largest double value is " + largestDouble);
        System.out.println("The value of aBoolean is " + aBoolean);
    }
}
```

Java基本資料型態範例結果

【執行結果】

```
The largest byte value is 127
The largest short value is 32767
The largest integer value is 2147483647
The largest long value is 9223372036854775807
The largest float value is 3.4028235E38
The largest double value is 1.79769313486231570E308
The value of aBoolean is true
```

列印控制字元

說明	Unicode designation	Character string
Backspace	BS	\b
Backslash	\	\\
Carriage Return	CR	\r
Double Quote	"	\"
Form Feed	FF	\f
Horizontal Tab	HT	\t
Single Quote	'	\'
Unicode Character	0x####	\u####

2 - 3、常數

常數是指在程式執行的過程中，值為固定不變的資料項目。例如，一年中的月份數為12，每天的時數為24，都可以視為常數。在Java中是以保留字“final”來宣告常數。

【語法】 final 型態　常數名稱[=初始值]；

【語法說明】

- final是常數宣告的保留字不可省略
- 常數的型態可以是基本資料型態
- 常數名稱命名時的規則和變數的一樣，並且要注意不可使用保留字
- 初始值可以在宣告時決定給或不給，但無初始值時表示在程式中會給定一個值，而且只會有一次設定的機會，設定後就不可以再改變

Ex2_2　常數範例程式

```
class DemoConstants {
        public static void main(String args[]) {
        final int workhour=40;
        final int workday;
        workday=5;
        System.out.println("In the future, labors must work " +  workhour
   + "
        hours in " + workday + " days.");  // 在螢幕上顯現出內容
    }
}
```

【執行結果】

```
In the future, labors must work 40 hours in 5 days.
```

範例 Ex2_3 計算 30 天工資

```
01.     class DemoConstants2{
02.             public static void main(String args[]) {
03.                     final int daypayment=3000;
04.                     final int workday=30;
05.                     final int money=daypayment*workday;
06.                     System.out.println("I earn " +  daypayment + "
   everyday,
                        so I earn " + money + " every " + workday +
   "days.");
07.                             // 在螢幕上顯現出內容
08.             }
09..    }
```

執行結果：

```
I earn 3000 everyday, so I earn 90000 every 30 days.
```

2 - 4 、 運算式、運算子和運算元

- 運算元(Operand):
  - 表示作用的對象。程式段落Payment=Payrate*Hours中， Payrate、Payment、Hours等資料變數。
- 運算子(Operator):
  - 表示作用的方式。如“=”（等號）、“*”（乘號）等。
- 運算式(Expression):
  - 經由運算子，將由常數或變數所代表的運算元經由運算子組合而成的一列式子。
- Java的運算子分為:

（1）設定(assignment)運算子
（2）算數(arithmetic)運算子
（3）比較(compare)運算子
（4）邏輯(logic)或稱為布林(boolean)運算子四類來說明。

運算子合併精簡使用

運算子	使用方式	相當作用的方式	備註
+=	運算元 1 += 運算元 2	運算元 1 = 運算元 1 + 運算元 2	加
-=	運算元 1 -= 運算元 2	運算元 1 = 運算元 1 - 運算元 2	減
*=	運算元 1 *= 運算元 2	運算元 1 = 運算元 1 * 運算元 2	乘
/=	運算元 1 /= 運算元 2	運算元 1 = 運算元 1 / 運算元 2	除
%=	運算元 1 %= 運算元 2	運算元 1 = 運算元 1 % 運算元 2	取餘數
&=	運算元 1 &= 運算元 2	運算元 1 = 運算元 1 & 運算元 2	位元及
|=	運算元 1 |= 運算元 2	運算元 1 = 運算元 1 | 運算元 2	位元或
^=	運算元 1 ^= 運算元 2	運算元 1 = 運算元 1 ^ 運算元 2	位元互斥或

Ex2_4　運算子合併精簡使用範例

```
class DemoOperators {
    public static void main(String args[]) {
        int a=20,b=3,c=5;
        c *=5;
        b +=(2*c)+1;
        a %=((c-=18)-1);
        System.out.println("a=" + a + ",b=" + b + ",c=" + c
  );
    }
}
```

【執行結果】

```
a=2,b=54,c=7
```

Ex2_5 DemoOperators2.java

```
01.class DemoOperators2 {
02.  public static void main(String args[]) {
03.     int a=75,b=15,c=25;
04.     System.out.println("a=" + a + ",b=" + b + ",c=" + c );
05.     c *=3;
06.     System.out.println("c *=3");
07.     System.out.println("a=" + a + ",b=" + b + ",c=" + c );
08.     a %=b+5;
09.     System.out.println("a %=b+5");
10.     System.out.println("a=" + a + ",b=" + b + ",c=" + c );
11.     b ^=((a &= 18) + (c |= 12));
12.     System.out.println("b ^=((a &= 18) + (c |= 12))");
13.     System.out.println("a=" + a + ",b=" + b + ",c=" + c );
14.     }
15.}
```

執行結果：

a=75, b=15, c=25

c *=3

a=75, b=15, c=75

a %=b+5

a=15, b=15, c=75

b ^=((a &= 18) + (c |= 12))

a=2, b=94, c=79

遞增、遞減運算子

- “++”（遞增)和“--”（遞減），隱含有運算完後“設定”的意義。
- 如x++表示將變數x的值加一後設定回變數x。
- “前置”表示執行運算式前將運算元先加一或減一。
- “後置”表示先執行運算式，再將運算元加一或減一。

遞增與遞減運算子

```
int age = 10;
int var = age++;
```

相當於：

```
var = age;
age = age + 1;
```

```
int age = 10;
int var = ++age;
```

相當於：

```
age = age + 1;
var = age;
```

遞增、遞減運算子

運算子	使用方式	說明
++（後置遞增）	運算元++	執行運算式後，再將運算元加一
++（前置遞增）	++運算元	將運算元先加一，再執行運算式
--（後置遞減）	運算元--	執行運算式後，再將運算元減一
--（前置遞減）	--運算元	先將運算元減一，再執行運算式

算數運算子

- 一般的數學運算，例如：加、減、乘、除等。

表2-8 算數運算子

運算子	使用方式	說明
+	運算元1 + 運算元2	運算元1 加上 運算元2
-	運算元1 - 運算元2	運算元1 減去 運算元2
*	運算元1 * 運算元2	運算元1 乘上 運算元2
/	運算元1 / 運算元2	運算元1 除以 運算元2
%	運算元1 % 運算元2	運算元1 除以 運算元2的餘數

Ex2_6 算數運算子範例

```
class DemoMath {
     public static void main(String args[]) {
          int x,y,z,q;
          int I=9;
          int mod;
          int J=3;
          x=I+J;     //x=12
          y=I-J;     //y=6
          z=I*J;     //z=27
          q=I/J;     //q=3
          mod=19%5; //mod=4
          System.out.println("x=" + x + ",y=" + y + ",z=" + z + ",q="
   + q+ ",mod=" + mod);
       }
}
```

【執行結果】

```
x=12,y=6,z=27,q=3,mod=4
```

比較運算子

- 比較運算子是用來比較兩個運算元的大小（可以是變數、常數、算式或是它們的組合)以決定其間關係是否成立，結果是一個只包含“真”或是“偽”的“布林值”。
- 若關係成立，用真（true）表示，反之用偽（false）表示，比如說之5>3是真，10<7是偽。

比較運算子

運算元	使用方式	以下狀況發生時，回應true
>	運算元1 > 運算元2	運算元1 大於 運算元2
>=	運算元1 >= 運算元2	運算元1 大於或等於 運算元2
<	運算元1 < 運算元2	運算元1 小於 運算元2
<=	運算元1 <= 運算元2	運算元1 小於或等於 運算元2
==	運算元1 == 運算元2	運算元1等於 運算元2
!=	運算元1 != 運算元2	運算元1 不等於 運算元2

Ex2_7 比較運算子範例

```
class DemoCompare {
    public static void main(String[] args) {
        int x = 27;
        int y = 36; //宣告變數
        System.out.println("Variable values as follows...");
        System.out.println("    x = " + x);
        System.out.println("    y = " + y);
        System.out.println("Greater than...");
        System.out.println("    x > y is " + (x > y)); //大於

    }
}
```

【執行結果】

```
Variable values as follows...
    x = 27
    y = 36
```

邏輯運算子

- 在真實的世界中，我們常會需要同時判斷幾個條件的組合是否成立，得到的結果可能會決定後續動作的執行，此時就需要用到邏輯運算子以組合不同的條件。
- 邏輯運算子的運算元及判斷的結果，也都是只有真（成立時）或偽（不成立）的布林值，表2-9列出Java語言使用的邏輯運算子：

邏輯運算子

名稱	運算子	使用方式	結果為真偽之情形及相關說明
條件及	&&	op1 && op2	op1 及 op2 均為真時，結果為真； op1 為真之條件下才對 op2 取右值
條件或	||	op1 || op2	op1 或 op2 任一為真時，結果為真； op1 為真之條件下才對 op2 取右值
非	!	! op	op 值為偽時，結果為真
及	&	op1 & op2	op1 及 op2 均為真時，結果為真
或	|	op1 ! op2	op1 或 op2 任一為真時，結果為真
互斥或	^	op1 ^ op22	op1、op2 二值相同時 (均為真或均為偽)，結果為真

邏輯運算子

在表示的習慣上，我們會把運算元所有真(T)及偽(F)的組合及邏輯運算的結果，對應在一個表中，稱為真偽值表(truth table)，如表2-10所示。

邏輯運算真偽值表

運算元1	運算元2	運算元1 && 運算元2	運算元1 || 運算元2	! 運算元1
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

Ex2_8 邏輯運算子範例

```
class DemoLogic {
    public static void main(String args[]) {
        boolean x,y;
         int i=10;
         x= (i >5) && (3>2);
         y=( i <5) && (3<2);
         System.out.println("x=" + x + ",y=" + y);
    }
}
```

【執行結果】

x=true，y=false

2_10 邏輯運算子複雜判斷範例

```
class DemoPrice{
public static void main(String args[]){
        int age=53;
int price=0;
        if(age<18) price=50;
        if(age>=18 && age<60) price=100;
        if(age>=60) price=70;
System.out.println("The price is "+ price + " dollars.");
 }
```

【執行結果】

```
The price is 100 dollars.
```

位元處理運算子

- 在位元層次處理資料:也就可以想像成先將資料成二進位，再在二進位位元的每個位址上進行 AND、OR、XOR 等運算。如： 3 AND 6 可轉成二進位 (011)2 & (110)2 = (010)2 為 2 。

- 例如：典型的電腦控制程式，常以每個位元代表一個開關的狀態，一個位元組即可表示 8 個開關，可經過另一個位元組形成的條件選擇要處理的開關，如 (00001001)2，即選擇了第 1 及 4 號開關，再和狀態位元組，如 (00001111) 表示開關現在狀況為第 1~4 號為開，5~8 號為關，即可使用 (00001001)&(00001111)=(00001001) 找到第 1 及 4 號開關狀態，而 5~8 號開關則不考慮的結果。

低階邏輯的運算

運算元	使用情形	說明
&	op1 & op2	位元階層的及運算
|	op1 | op2	位元階層的或運算
^	op1 ^ op2	位元階層的互斥或運算
~	~ op	求 op 的補數

Ex2_12 型態轉換範例

```
class DemoTypeConverse {
      public static void main(String[] args) {
          int x=3;
          double y=2.0;
          x=x+y;
          System.out.println(x);
      }
}
```

【執行結果】

1 error

(因為X為整數，而Y為雙精密浮點數)

Ex2_13 強迫型態轉換範例

```
class DemoManulTypeConverse {
    public static void main(String[] args) {
      double x=3.0;
      int y=2;
      x=x+(double)y;
      System.out.println(x);
    }
}
```

【執行結果】

```
5.0
```

[第2章習題]

- 1.假設天然瓦斯計費標準如下:若度數<20，瓦斯費為150元;若度數<=20，瓦斯費為(度數-20)*4+100。設計一Java程式運算式來表示應收的費用。

閏年的判斷

是否為閏年?

[條件1] 四年一閏，百年不閏

[條件2] 四百年閏

其中一個條件符合就是閏年！

```
int  year = 2012;
System.out.print( year + " is leap year : ");
System.out.println(
    (year % 4 == 0 && year % 100 != 0) ||
     year % 400 == 0
);
```

三元運算子 (ternary operator)

它是一個三元運算子 (ternary operator)

運算元1 ? 運算元2 : 運算元3

運算元1 是一個 布林運算式(boolean expression)，
值為 true 運算結果為 運算元2
值為 false 運算結果為 運算元3

[範例]三元運算子

```
int j = 10;
int k = 20;
int max = (j>k) ? j : k ;
System.out.println("max = " + max);


int  eng  = 90;
int  math = 90;

System.out.println(
    (eng>=90 && math>=90) ? "錄取" : "不錄取"
);
```