Module III
定數、變數、常數與運算子

1. Java定數
2. Java基本資料型別
3. Java變數
4. Java常數
5. 各種運算子介紹
6. 晉升與轉型

Java定數 (Literal) (1/5)

- 格式固**定**不變的**數**值就是定數
- 多數人把定數稱之為字面常量或是字面量
- Java定數可分為以下四種：
  - 整數定數
  - 浮點數定數
  - 布林值定數
  - 字元定數

Java定數 (Literal) (2/5)

- 根據整數長度大小，Java提供了四種整數資料類型：

  byte, short, int, long

- Java整數定數可以用十進位、八進位與十六進位制表示(Java 7以後可以用二進位制表示)
- 十進位即為我們平常使用表示，如100，12345
- 八進位需為0開頭，如0123，0567
- 十六進位需為0x或0X開頭，如0x8E，0X8E
- 二進位需為0b或0B開頭，如0b1001，0B1001
- 為了可閱讀性，Java 7開始，可以使用底線(_)來分隔整數定數

  如會計人員會使用1,000,000表示一百萬，在Java裡我們可以

  1_000_000作為表示

Java定數 (Literal) (3/5)

- 根據浮點數長度大小，Java提供了兩種浮點數資料類型：

  float, double

- 1.23，3.14等為我們常見的浮點數定數，若是在後面加上f或F，則可以讓此浮點數定數變成一個float類型，如1.23f或1.23F

- Java浮點數定數另有科學記號表示法，如1.23e-10(1.23×10^{-10})，多用作工程或電腦專用浮點數，e大小寫均可，代表10的次方數

Java定數 (Literal) (4/5)

- Java提供了一種布林值定數資料類型：

  boolean

- Java使用**true**與**false**來代表布林值定數

- 換作日常生活就是我們常說的「真」與「假」、「對」與「錯」、「Yes」與「No」

- 對電腦來說，1就是true，0就是false

Java定數 (Literal) (5/5)

- Java提供了一種字元值定數資料類型：

  char

- 字元值定數為一個Unicode字元，或是在一對單引號裡的特殊字元

  (又稱為轉義序列(Escape Sequence))

- 一般字元表示方式如：’a’, ‘o’, ‘%’, ‘我’, ‘A’

- 轉義序列表示方式如：’\n’ (換行)，’\\’ (\)，’\”’ (“)

- Java允許使用Unicode表示法，如 ’我’在unicode編號為6211，則可以使用’\u6211’表示

Java轉義序列 (補充1)

- 轉義序列(Escape Sequence)
  - \’ ：單引號
  - \” ：雙引號
  - \\ ：反斜線
  - \n ：換行
  - \t ：tab鍵
  - \b ：倒退一格
  - \f ：換頁
  - \r ：return鍵 (Enter鍵)

編碼法 (補充2 1/2)

- 編碼法(Encoding)：將文字編碼成數字，再以0與1表示這些數字
- 電腦世界常見的編碼法：

ASCII

American Standard Code for Information Interchange
(美國國家標準資訊交換碼)
長度為1 byte
拉丁字母(歐美國家常用)
範例：'A' (= 65)，'a' (= 97)

Big5

大五碼：大千、倚天、國喬、零壹
正體中文(台灣、香港常用)
長度為2 bytes
範例：'乙' = 1010010001000001 (0xA441)

Unicode

萬國碼(Universal Code)
世界上所有字母 (全世界通用)
長度為2 bytes (UTF-8, UTF-16)或4 bytes (UTF-32)
範例：'乙' = 1000111001011001 (0x4E59)

編碼法 (補充2 2/2)

- UTF-8編碼介紹：
  - Universal Transformation Format (通用轉換格式)
  - 原ASCII使用1 byte儲存 (u0000 ~ u07FF)
  - 拉丁文、希臘文、希伯來文與阿拉伯文使用2 bytes儲存 (u0080 ~ u07FF)
  - 中、日、韓 (簡稱CJK)漢字或字母使用3 bytes儲存 (u0800 ~ uFFFF)
  - 其它字母使用4 bytes儲存 (u10000 ~ u10FFFF)
  - **因為省空間且與ASCII相容，故為現今最常使用的編碼**

- UTF-16無法與ASCII相容，需經過轉換，所以較少用

- UTF-32為最標準的Unicode格式，一律使用4 bytes儲存，除非要顯示罕用語言與字母，否則不太使用

Java基本資料型別 (1/2)

- 基本資料型態 (Primitive Data Types)

型態名稱	大小	範圍	說明	初始值
整數型態				
byte	8 bits	$-2^7 \sim 2^7-1$ (-128 – 127)	位元組整數	0
short	16 bits	$-2^{15} \sim 2^{15}-1$	短整數	0
int	32 bits	$-2^{31} \sim 2^{31}-1$	整數	0
long	64 bits	$-2^{63} \sim 2^{63}-1$	長整數	0L
浮點數型態				
float	32 bits	$-3.4*10^{38} \sim 3.4*10^{38}$	7位小數	0.0F
double	64 bits	$-1.8*10^{308} \sim 1.8*10^{308}$	15位小數	0.0(D)
其它型態				
boolean	1 bit	true, false	布林	false
char	16 bit / Unicode格式	$0 \sim 2^{16}-1$ (0-65535) (\u0000 - \uffff)	字元	'\u0000'

Java基本資料型別 (2/2)

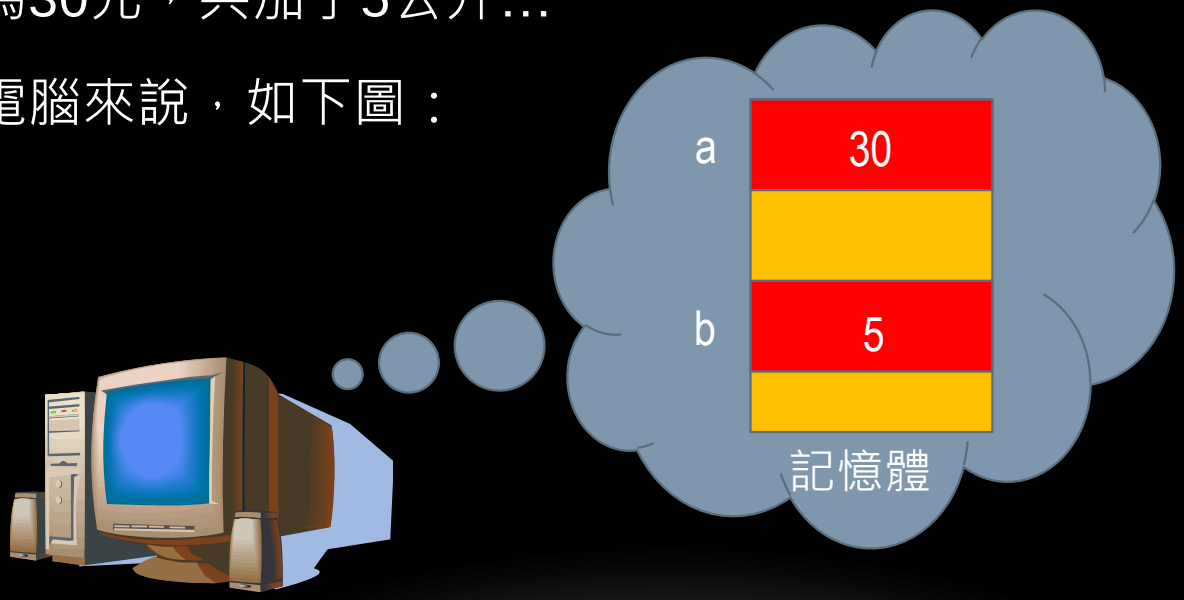
- Java基本資料型別初始值指定 (Initialization)

型態名稱	初值設定
整數型態	
byte	byte i = 1;
short	short i = 1;
int	int i = 1;
long	long i = 1L;
浮點數型態	
float	float i = 1.1F;
double	double i = 1.1; double i = 1.1D;
其它型態	
boolean	boolean b = true; boolean b = false;
char	char c = 'A'; char c = '\u0041'

範例：TestPrimitiveType.java

什麼是變數

- 凡計算前必先記憶！
- 每公升油為a元，共加了b公升，我們可以計算出a x b元，假設一公升為30元，共加了5公升...
- 對電腦來說，如下圖：



變數命名規則

- **每個變數名字都代表著記憶體中某個記憶體位址**
- 變數名稱：

  - 可以用**A-Z**, **a-z**, **0-9**, **_**(底線), **$**，長度不限制

  - 第一個字不可以是數字

  - 大小寫不同 (case-sensitive)
- **不能是關鍵字 (keyword) 或稱作保留字**
- **變數、方法名稱以小寫開頭，類別名稱為大寫開頭**

  - 如果名稱由不同字組成，建議後面字的首字元為大寫，如myName

Java關鍵字與保留字

- Java Programming Language Keywords

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

- goto與const也為關鍵字，但不可以在Java裡使用
- Reserved literal words：**null**, true and false

變數種類與有效範圍

- 基本觀念

- 區域變數 (Local variables)：
  - 宣告在方法(method)內
  - 區域變數只能在它們被宣告的同方法內存取
  - 又稱為automatic, temporary或stack variables
- 實體變數 (Instance variables)：
  - 宣告在方法之外，類別之內，**且沒有static修飾子**
  - 實體變數可被類別內任何非static方法存取
  - 又稱member variables(成員變數)、attribute variables(屬性變數)
- 注意：
  - 加上static修飾子為類別變數，也稱為靜態變數
  - 詳見修飾子章節的static部份

變數宣告與初始化 (1/2)

- 基本觀念
  - 變數被使用前皆需要有初值，否則編譯時會有錯誤訊息
  - 基本的宣告方式：
    - <資料型態> <變數名稱>;
    - 如：int i; int i, j , k;
  - 宣告之後，再指定初始值 (區域變數才可以)
    - 如：int i;
      i = 0;
  - 宣告同時指定初始值：
    - <資料型態> <變數名稱> = <初始值>;
    - 如：int i = 0; int i = 0, j = 1, k = 2;
- 區域變數
  - 區域變數在宣告時，Java不會自動給予初始值，因此在被存取前必需先自行指定初始值
- 實體變數
  - 實體變數在宣告時，Java會自動給予初始值，不可以在宣告後又再指定初始值，編譯會有錯誤

變數宣告與初始化 (2/2)

- 實體變數的預設初始值：

變數型態	值
byte	0
short	0
int	0
long	0L
float	0.0F
double	0.0(D)
boolean	false
char	‘\u0000’
All reference types	null

Java常數

- Java常數：**常**常要用到的**數**，因為常用到，所以不能像變數一樣變來變去

- 常數(Constant)**通常全部大寫**，且習慣使用**底線 _** 來分開字組，例如：MY_NAME

- 一個**變數**宣告為**final**，表示這個變數在初始值化後，不得再變更其值，也就是常數(Constant)，例如：final double PI = 3.14;

運算子

- 運算子 (operator)可對一個以上的運算元(Operand)進行運算動作
- 運算子執行運算之後將回傳值，而其回傳值型態視運算元而定

運算子名稱	
算術運算子	+, -, *, /, %
遞增遞減運算子	++, --
指定運算子	=, +=, -=, *=, /=, %=
關係運算子	<, <=, >, >=, ==, !=
條件運算子	&&, ||, !
位元運算子	&, |, ^, ~
移位運算子	<<, >>, >>>
三元運算子	? :

算術運算子

- 算術運算子 (Arithmetic Operators)又稱為二元運算子

算術運算子	用法	說明
+	a + b	1+1為數字相加 “1” + 1為串接相加
-	a – b	a 減 b
*	a * b	a 乘 b
/	a / b	a 除 b (5.0/2 = 2.5 ; 5/2 = 2)
%	a % b	取a 除以b 後的餘數 (7%2 = 1 ; 9.6%3.5 = 2.6)

- 若兩個運算元的位階不相等，則運算完後的回傳值會與位階高者相同
- 若兩個運算元為基本型別，至少會轉換成int型別

範例：TestArithmeticOP.java

遞增遞減運算子

- 遞增(++)與遞減(--)運算子通常又稱為一元運算子
- 遞增與遞減運算子分為**後置型**與**前置型**

遞增遞減運算子	用法	說明
(後置型)++	a++	a = a + 1
(後置型)--	a--	a = a - 1
(前置型)++	++a	a = a + 1
(前置型)--	--a	a = a - 1

- 後置型 (a++)：**先取值後再遞增**
- 前置型 (++a)：**先遞增後再取值**
- 如：int a = 3;

```
System.out.println(a++);   //3        System.out.println(++a);   //4
System.out.println(a);     //4        System.out.println(a);     //4
```

範例：TestInDecrementOP.java

指定運算子

- 指定運算子 (assignment operators)是將**右邊**運算完成的結果**指定給左邊**的變數保存起來
  - “=”並非數學上”等於”的意思，而是”**指定**”的意思
  - 因此右值的位階不可以高過左值的位階
  - **位階高低順序：double > float > long > int > short > byte**

指定運算子	用法	相當於
=	a = 2	a = 2
+=	a += 2	a = a + 2
-=	a -= 2	a = a -2
*=	a *= 2	a = a * 2
/=	a /= 2	a = a / 2
%=	a %= 2	a = a % 2

範例：TestAssignOP.java

關係運算子

- 關係運算子 (Relational operators)用來比較兩個變數的值，其回傳值結果是一個布林值(true / false)

關係運算子	用法	回傳true/false結果
<	a < b	a是否小於b
<=	a <= b	a是否小於等於b
==	a == b	a是否等於b
!=	a != b	a是否不等於b
>=	a >= b	a是否大於等於b
>	a > b	a是否大於b

範例：TestRelationalOP.java

條件運算子

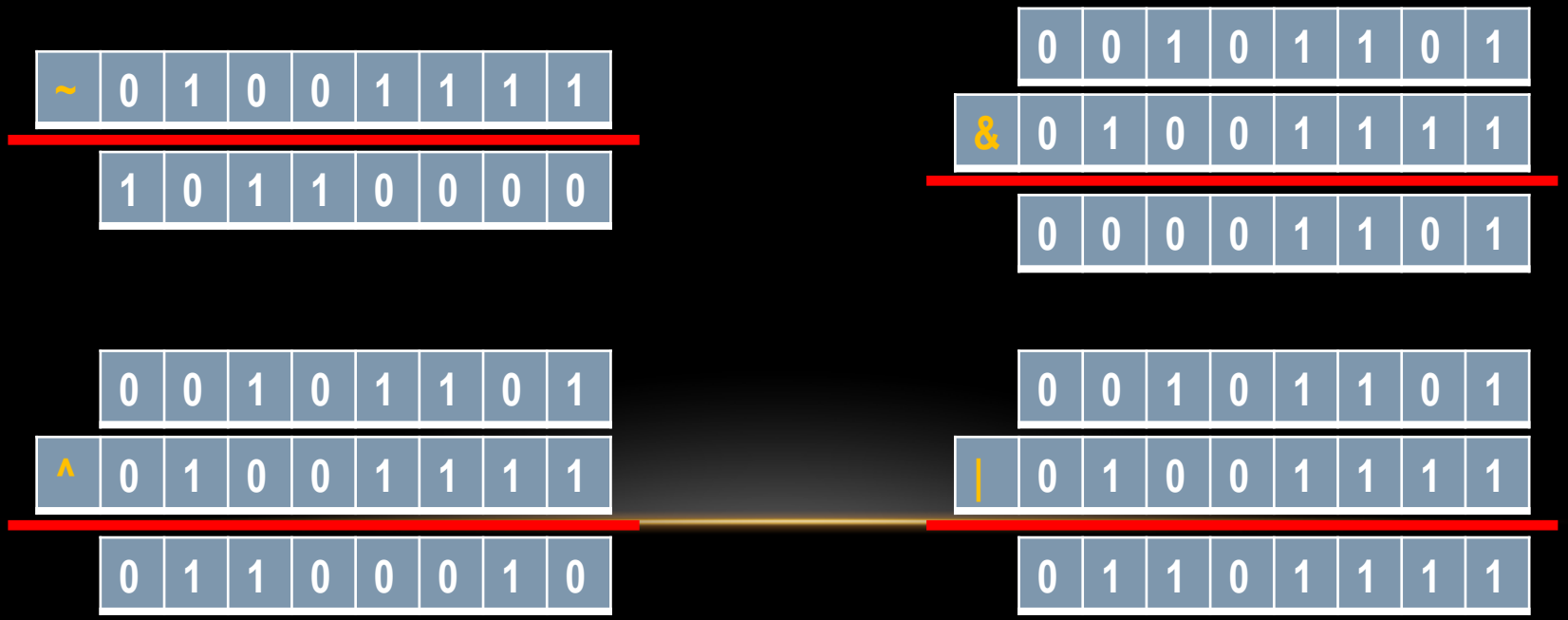
- 條件運算子 (Conditional operators)是將兩個布林值合併起來的運算子，運算結果仍為布林值

條件運算子	用法	回傳true/false結果
&&	a && b	a和b都是true才回傳true
||	a || b	a和b只要有一方為true就回傳true
!	!a	傳回相反的布林值

範例：TestConditionalOP.java

位元運算子

- 位元運算子 (Bit operators)： &(and), |(or), ^(xor)可用在整數的位元運算
- ~ (位元反轉運算子)：用於整數，將位元1變成0，0變成1



範例：TestBitOP.java

移位運算子

- 移位運算子 (Shift operators)用在整數型態的位元移位運算

移位運算子	用法	功能
>>	a >> b	將a向右移動b個位元 (具正負值)
<<	a << b	將a向左移動b個位元 (具正負值)
>>>	a >>> b	將a向右移動b個位元 (不具正負值向右移位，以0位元補進)

- 移位運算子的右邊參數，若超過型態本身的位元數，則以餘數作為移位的次數

範例：TestShiftOP.java

三元運算子

- 三元運算子 (Ternary operators)是一個簡略的if – else敘述

三元運算子	用法	功能
? :	a ? b : c	如果條件a為true，執行運算式b 如果條件a為false，執行運算式c

- 例：

```
if ( x > y )
        a = x + 100;
else
        a = y + 100;
```

相當於： a = (x > y) ? x + 100 : y + 100;

範例：TestTernaryOP.java

運算子優先順序

Operators	Associative
++ -- ~ !	**R to L**
* / %	L to R
+ -	L to R
<< >> >>>	L to R
< > <= >= instanceof	L to R
== !=	L to R
& ^ |	L to R
&& ||	L to R
<boolean_expr> ? <expr1> : <expr2>	**R to L**
= += -= *= /= %= <<= >>= >>>= &= ^= |=	**R to L**

晉升與型別轉換

- 晉升 (Promotion)

  - 指較小的資料型別(等號的右邊)自動晉升為較大的資料型別(等號的左邊)

- 型別轉換 (Typecasting)

  - 指較大的資料型別轉換成為較小的資料型別
  - 必須使用強制轉換
  - 語法：(target type) value

  例：int x = 1;

  ```
  double y = 2.2;
  y = x + 1;        // 晉升
  x = (int)y + 1;  // 型別轉換
  ```

範例：TestCast.java

進制轉換 (補充3)

- 四種進制：
  - 二進制(Binary) : 0, 1
  - 十進制(Decimal) : 0 - 9
  - 十六進制(Hexadecimal) : 0 - 9, A, B, C, D, E, F
  - 八進制(Octal 已較少使用) : 0 - 7
- 二進制、十進制、十六進制互轉
  - 如$1000001_2 \rightarrow 65_{10} \rightarrow 41_{16} \rightarrow 101_8$
- 整數常數
  - 十進制 : 65
  - 十六進制 : 0x41 (第一個字元是零)
  - 八進制 : 0101 (第一個字元是零)

二進位	十進位	十六進位
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

章節整理

- 可以清楚定數、變數與常數的意義跟不同之處
- 瞭解Java八大基本資料型別(Primitive Data Types)
- 撰寫程式時，變數的命名規則與注意事項
- 謹記變數宣告時的初始值，會因區域變數或實體變數而有所不同
- 瞭解各位運算子的使用方式與執行優先順序
- 小心！型別轉換時，需確認資料型別是否相符，以免造成轉型失敗