第二章 程式語言的概念

真理大學
Aletheia University

本章內容

- 2-1　程式的符號
- 2-2　變數的設定
- 2-3　資料型別
- 2-4　運算符號
- 2-5　Java 的物件概念

真理大學
Aletheia University

2-1 程式的符號

- 以我們在上一章執行過的程式來觀察，您會發現程式也只是一堆文數字、標記組合而成的檔案。
- 但和一般文字檔案不同的是：程式中的文字的組成包含了特定的原則，而這個原則就是程式撰寫的「**語法(Syntax)**」。依照正確語法排列而成的文字才能被Java 的編譯器所接受。
- 各種程式語言定義的語法並不會完全相同，但一般而言，程式語言都會包含以下的組成元件：「識別字」、「關鍵字」或是「保留字」、「資料常數」、「符號」等。

真理大學
Aletheia University

2-1-1 識別字 (Identifiers)

- 識別字是指變數 (Variable) 、方法 (Method) 、類別 (Class) 的名稱
- 識別字命名的原則有
  - 第一個字只能是 $ 或是 _ 或是 字母 (Letter)
  - 第二個字起可以有數字
  - 識別字中不能有標點符號、空白，或是－
  - Java 的關鍵字不可以當作是識別字
- 可不可以用中文當做是識別字的名稱?

真理大學 Aletheia University

2-1-2 Java 的關鍵字 (Keywords)

- 程式語言本身定義的識別字即是「關鍵字」或「保留字」。
- Java 的關鍵字

abstract	assert	boolean	break	byte	case
catch	char	class	const	continue	default
do	double	else	enum	extends	final
finally	float	for	goto	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				

真理大學 Aletheia University

2-1-3 資料常數 (Literals)

- 「**資料常數**」是指本身的定義不會再更改的資料內容，例如：數字「1」代表的意義不會再重新更改。
- Java 的資料常數類型：

資料類型	使用範例	使用方式
整數	100, 0, -100	不包含小數位
浮點數	124.2, 0.0, -365.0	包含小數位
真假	true, flase	使用保留字代表
字元	'A', 'b', '*', '5'	以單引號括起來的單一文數字
字串	"This is", "天使", "+-*/"	以雙引號括起來的文數字

真理大學 Aletheia University

2-1-4 符號 (SymbolIdentifiers)

- 程式中的符號多半是運算符號。
  - 例如： ＋、-、＊、/、＝...。
- 也可能是分隔符號。
  - 例如「小括號 （）」、「大括號｛｝」、「分號 （；）」、「句號 （.）」。

真理大學 Aletheia University

2-2 變數的設定

- 「**變數 (Variable)**」的主要目的是在記憶體中保留特定大小的空間，讓程式執行時可以暫時儲存資料。因為資料型態有許多種，並且可能需要使用到不同的記憶體大小。所以，在 Java 程式中，强制要求在使用變數之前必需先經過「**宣告 (Declare)**」。
- 變數宣告的語法如下：

  type name [= value] [, name [=value]...];

  - 「type」是資料的型別，
  - 「name」是變數的名稱，
  - 如果需要，您可以使用「＝」符號來指定變數的初始值。
- 而您也可以在指定型別後，同時宣告多個變數的內容，例如：以下宣告方式都是合法的變數宣告：

  - int days, months, years; // 同時宣告三個整數變數
  - int total; // 只宣告一個整數變數

真理大學 Aletheia University

2-2 變數的設定

- 變數的有效範圍 (Scope)
  - 有效範圍是指在程式中可以使用變數的地方
  - 宣告在方法中的變數稱為區域變數 (Local Variable)，也可以稱為 stack 變數、 automatic 變數、 temporary 變數
  - 區域變數是在執行區段中產生，離開區域後，區域變數即毀滅
  - 區域變數的有效範圍是由程式區域所決定，在進入程式區域時可以使用該變數，但離開程式區域後，就無法再使用區段中的變數
  - 內區域中可以使用外區段中的變數 (shadow)

真理大學
Aletheia University

2-3 資料型別

- Java 程式是屬於「**強型別 (strongly typed language)**」的程式語言。
- 雖然 Java 也提供自動轉型的功能，但並不是所有的資料型別都可以自由的轉換。
- 您在處理不同型別的資料運算時，最好是自行處理型別的轉換方式。
- Java 的資料型別可以概分為兩大類:
  - Reference Data Type：可以用來產生物件
  - Primitive Data Type：不能用來產生物件

真理大學
Aletheia University

2-3 資料型別

- Java 中定義了八個 Primitive Types，但我們可以將它們區分為四大類，除了這 8 個型別外，其餘的資料型別都是屬於「參考資料型別」。
- 「基本資料型別」變數所佔的記憶體大小並不相同，但「參考資料型別」的變數是用來儲存物件的記憶體位置，它們的大小都是 4 Bytes。
- 以最簡單的區分方式，我們可以這樣分別:
  - 「基本資料型別」變數儲存實際的內容，
  - 「參考資料型別」變數儲存著它代表的物件的記憶體位置。

真理大學 Aletheia University

2-3 資料型別

- Java 的基本資料型別的類型有:

分類	保留字	名稱	Byte	有效範圍
整數	byte	位元組	1	-128~127
	short	短整數	2	-32,768~32,767
	int	整數	4	-2,147,483,648 ~ 2,147,483,647
	long	長整數	8	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
浮點	float	浮點數	4	負值 -3.402823E38~-1.401298E-45 正值 1.401298E-45~3.402823E38
	double	倍精數	8	負值 -1.797693134E3.8~4.9406564584124E-324 正值 4.94.6564584E-324~1.797693134862E308
其他	char	字元	2	\u0000 ~ \uFFFF
	boolean	布林值	2	true , false

真理大學 Aletheia University

2-3-1 建立整數型別的變數

- 整數型別的變數是最常用到的一種變數，但此類型的變數內容不可以包含小數，否則會產生編譯時的錯誤。
- 例如，「int」型別的宣告方式:

```
int days=7;            // 宣告整數變數，並指定值
```

- 例如，「short 」型別的宣告方式:

```
short days = 32767;          // 宣告 short 變數，並指定值
```

- 例如，「byte 」型別的宣告方式:

```
byte a = 1;            // 宣告 byte 變數，並指定值
```

- 宣告時不可以超過該型別的最大或是最小值，例如以下的宣告及指定的值的內容，會使得程式產生編譯時的錯誤:

```
short days = 32768;                    // 超過 short 型別的 32767 的上限
byte a = -129;                 // 超過 byte 型別的 -128 的下限
```

真理大學 Aletheia University

2-3-1 建立整數型別的變數

- 長整數 (long) 型別可以容納較多位數的數值，除了上述的宣告方式之外，我們也可以在指定變數值時，再加上大寫的「L」或是小寫的「l」來直接指明這是一個長整數值。
- 例如以下的範例：

```
long value;                          // 宣告 long 變數，並指定值
value = 12345678987654321L; // 指定值
```

真理大學 Aletheia University

2-3-2 建立浮點型別的變數

- 浮點數包含整數和小數部份，例如：「3.14」或是「0.21」。
- 在 Java 中，浮點數的預設型別是「double」。
- 如果需要設定 float 型態的變數，可以在值的後方加上「f」或是「F」
- 宣告範例：

```
float value = 20.1f;          // 宣告 float 變數，並指定值
```

真理大學 Aletheia University

2-3-3 建立字元型別的變數

- Java 中是利用 2 個 Bytes 來表示所有的 Unicode 字元集，其目的是希望能有一套「**字元集 (character set)**」來表示不同語言中的符號。
- 請注意：字元變數不是指真正的字元，而是指字元集的索引。
  - 例如：在 Unicode 中，「A」是以「65」來表示的。
- 宣告字元型別變數的範例：

```
char A = 65;                    // 宣告 char 變數，並指定值
char B = 'B';                   // 利用單引號指定字元的值
```

真理大學 Aletheia University

2-3-3 建立字元型別的變數

- 對於一些非 ASCII 字元集或是無法由鍵盤輸入的符號，我們還可以使用 16 進位碼來表示字元，格式如同「/uHHHH」，「H」是 16 進位碼。宣告範例:

```
char x = '\u5496';
```

- Java 可以表示的字元符號可達 65,535 個，但對於一些有特殊語意的字元（稱為「**跳脫字元 (Escape Sequence)**」，我們要利用「反斜線」來搭配「跳脫字元」來顯示，如:

跳脫字元	Unicode	代表字元
\'	\u0027	單引號，原代表字元資料常數
\"	\u0022	雙引號，原代表字串資料常數
\\	\u005C	反斜線，原代表跳脫字元的起始符號
\n	\u000A	換行符號
\t	\u0009	Tab 鍵
\r	\u000D	Enter 鍵
\b	\u0008	Backspace 鍵

真理大學 Aletheia University

2-3-4 建立布林型別的變數

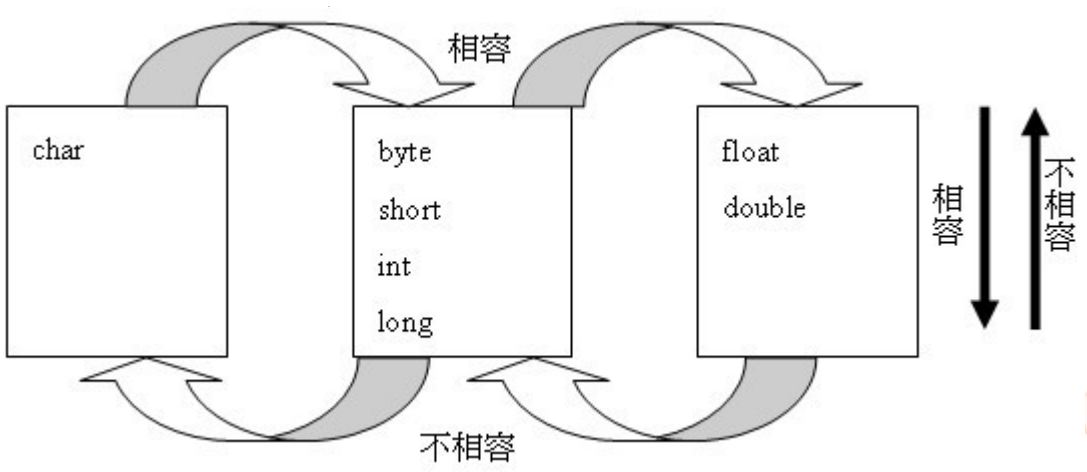
- Java 中的布林變數只能儲存「 true 」或是「 false 」，通常用來判斷儲存的資料內容為「真 (成立) 」或是「假 (不成立) 」。
- 特別注意的是：在 Java 中，我們不能在布林變數中儲存 0 、 1 或是其他的數值。布林變數的預設值是「 false 」。
- 宣告的範例如下：

```
// 宣告一個布林變數，初始值為「 false 」
boolean judge;
// 將「 true 」指定給「 judge 」變數，可以編譯成功
judge = true;
```

真理大學 Aletheia University

2-3-5 資料型別的轉換

- Java 語言中很重視資料的型別，對於基本資料型別 (Primitive Data Types) 而言，一旦宣告後，變數的大小是固定的，不可以隨意的轉換。
- 但 Java 也提供資料型別轉換的機制的，寫作 Java 程式時，我們可以利用 Java 的「**自動型別轉換 (automatic type conversion)**」，或是自行指定轉換的資料型別來進行資料的型別轉換工作。
- 型別轉換的



2-3-5 資料型別的轉換

- **自動型別轉換**
  - 在 Java 中，將一種型別的資料轉換成另一種型別的資料的動作稱為「**Casting**」。
- 如果在轉換時，資料型別符合轉換的規定，即使程式中並無特別指定資料型別需要轉換，Java 仍會自動的進行資料型別的轉換工作，這個機制稱之為「**隱含式轉換 (Implicit Casting)**」。
- 例如：

```
int salary = 25000;    // 定義整數變數，代表月薪；
float tax = 0.06F;     // 定義變數，代表稅率為 6%
double amount;                    // 繳稅總額
amount = salary * 12 * tax;  // 自動轉換型別
```

真理大學 Aletheia University

2-3-5 資料型別的轉換

- **顯式型別轉換 (Explicit Casting)**
  - 如果您要將某個值指定給變數，但該值的資料型別所佔的 Byte 數卻大於變數的資料料型別的 Byte 數，或是該值大於變數的資料型別所能容納的範圍時，這種行為我們稱之為「**縮小轉換 (narrowing conversion)**」。
  - 預設情況下，Java 不允許這種轉換，因為這會造成轉換後，數值資料的精準度 (precision) 降低。
- 例如:

```
int amount, salary = 12000;          // 設定「總額」變數為「 int 」型
double tax = 0.6;                    // 設定「稅額」為 6%
amount = salary * 12 * tax; // 會產生編譯時的錯誤
```

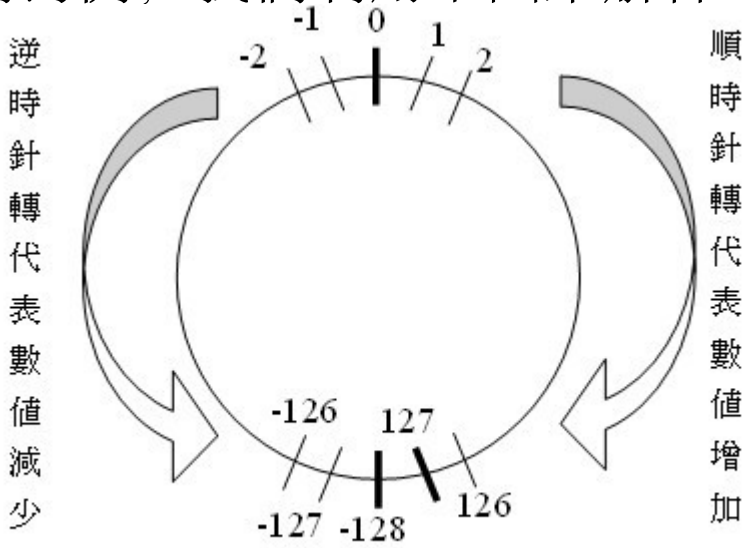
- 正確的作法是，我們必需明確的將資料型別轉換:

```
amount = (int) (salary * 12 * tax);   // 進行 Explicit Casting
```

真理大學 Aletheia University

2-3-6 Java 的溢位 (Overflow) 處理

- Java 程式在執行時，並沒有溢位的問題。
- 在 Java 中，當計算式中的數值超過了資料型別所能容納的範圍後，數字會產生「轉折」的現象，我們稱之為「**Modular**」。
- 以「byte」型別為例，我們利用下圖來解釋「Modular」的現象:



在數值的極限時，會產生反轉的情形

理大學
heia University

2-3-7 常數 (Constants) 的宣告

- 「**常數**」變數是指在程式執行過程中，變數的內容不可以改變的變數。這個定義是指：宣告常數變數時，必需直接指定初始值；或是在第一次用到這個變數時，我們必需指定值，而且，不可以再變更該變數的內容。在 Java 中，宣告常數變數必需加上「 final 」這個修飾詞 (modifier)。
- 宣告的語法如下：

```
final type name [= value];
```

- 以下的常數使用方式是合法的：

```
final int hours = 24;                    // 宣告時，直接指定常數變數的內容
```

- 或者，我們也可以這樣的使用：

```
final int hours;                         // 宣告一個常數變數
hours = 24;                              // 第一次可以指定常數變數的內容
```

真理大學 Aletheia University

2-4 運算符號

- 運算式是由「**運算子 (Operator)**」和「**運算元 (Operand)**」組合而成的。「運算元」代表資料的表示方式，而「運算子」決定了資料的運算方式，它又可以被稱為「運算符號」。
- 例如：以下的表示式中：

```
x = y + z;
```

  - 表示式中的「x」、「y」、「z」代表運算元，而「+」、「=」則是運算子。

- 程式中可以使用的運算子主要可以區分為「算術運算子」、「指定運算子」、「關係運算子」、「邏輯運算子」、「字串運算子」等類型。

真理大學 Aletheia University

2-4 運算符號

- 算術運算子

符號	說明	使用範例
+	加法運算	x + y;
-	減法運算，或是負號	x-y; -x
*	乘法運算	x * y
/	除法運算	x / y
%	餘數運算	x % y
++	遞增運算	x++; 等同 x = x + 1
--	遞減運算	x--; 等同 x = x - 1
<<	算數位元左平移	
>>	算數位元右平移，保留 sign	
>>>	邏輯位元右平移， unsign	

真理大學 Aletheia University

2-4-1 「++」和「--」運算子

- 「++」和「--」這兩個運算子會使得變數本身的內容自動增加或是減少 1 。
- 例如：運算式「x++」的結果等同於以下的運算式:

```
x = x + 1;
```

- 運算子置於變數之前或變數之後，對於變數本身的內容改變的時間並不相同。
  - 如果「++」或「--」放於變數之後，該變數會在整行運算式執行完成後，變數本身的值才會增加 1 。
  - 如果「++」或「--」放於變數之前，該變數本身的值會先增加 1，再執行運算式。

真理大學 Aletheia University

2-4-2 位元平移運算子

- 「<<」、「>>」、「>>>」也可以分類為位元算術運算子，這些運算子可以用來操控整數型別資料中個別位元的運算，使用的技巧在於二進位位元的移動。
- 「<<」稱為「算數左平移」，使用範例如同：

  6 << 1

  - 該行敘述計算的結果為「12」，請參考下表：

6 的二進位	0	0	0	0	0	1	1	0
左平移 1 個 bit	0	0	0	0	1	1	0	X
結果	$0 * 2^7$	$0 * 2^6$	$0 * 2^5$	$0 * 2^4$	$1 * 2^3$	$1 * 2^2$	$0 * 2^1$	$0 * 2^0$

真理大學 Aletheia University

2-4-2 位元平移運算子

- 「>>」稱為「算數右平移」，使用範例如同:

  6 >> 1

  – 該行敘述計算的結果為「3」，請參考下表:

6 的二進位	0	0	0	0	0	1	1	0	
右平移 1 個 bit		0	0	0	0	0	1	1	0
結果	$0 * 2^7$	$0 * 2^6$	$0 * 2^5$	$0 * 2^4$	$0 * 2^3$	$0 * 2^2$	$1 * 2^1$	$1 * 2^0$	

真理大學 Aletheia University

2-4-2 位元平移運算子

- 「 >>> 」稱為「邏輯右平移」，使用範例如同:

  6 >>> 1

  – 該行敘述計算的結果為「 3 」，請參考下表:

6 的二進位	0	0	0	0	0	1	1	0	
右平移 1 個 bit		0	0	0	0	0	1	1	0
結果	$0 * 2^7$	$0 * 2^6$	$0 * 2^5$	$0 * 2^4$	$0 * 2^3$	$0 * 2^2$	$1 * 2^1$	$1 * 2^0$	

真理大學 Aletheia University

2-4-3 指定運算子

- 指定運算子的使用方式不外乎是:

```
x = 2;
x = x + 2;
```

符號	說明	使用範例
=	將右邊的運算結果，指定給左邊的變數	x = 10;
+=	將左邊變數的值的加上右邊的值，再指定給左邊的變數	x += 1; 等同 x = x + 1
-=	將左邊變數的值的減去右邊的值，再指定給左邊的變數	x -= 1; 等同 x = x - 1
*=	將左邊變數的值的乘以右邊的值，再指定給左邊的變數	x *= 2; 等同 x = x * 2
/=	將左邊變數的值的除以右邊的值，再指定給左邊的變數	x /= 2; 等同 x = x / 2
%=	將左邊變數的值的除以右邊的值，再指定餘數給左邊的變數	x %=2; 等同 x = x %2
&=	將左邊變數的值和右邊的值相連，再指定給左邊的變數	x &= y; 等同 x = x&y
^=	位元互斥指定運算	x ^= y; 等同 x = x^y
<<=	算數位元左平移	x<<=2; 等同 x= x<<2
>>=	算數位元右平移	x>>=2; 等同 x= x>>2

2-4-4 關係運算子

- 關係運算子用來比較運算子左右的值是否相等。如果相等，則比較的結果代表「成立」，並傳回「true」值；如果不相等，則比較的結果代表「不成立」，並傳回「false」值。
- 例如：

```
10 >= 10;          // 運算的結果傳回「true」
```

符號	說明	使用範例
==	等於	x == 10;
!=	不等於	x != 10;
<	小於	x < 10;
<=	小於或等於	x <= 2;
>	大於	x > 10;
>=	大於或等於	x >= 10;
>>=	算數位元右平移	x>>=2; 等同 x= x>>2

真理大學 Aletheia University

2-4-5 邏輯運算子

符號	說明	使用範例
&	且 (AND) 運算	x & y;
|	或 (OR) 運算	x | y;
!	否 (NOT) 運算	!x;
^	XOR 運算	x ^ y;
&&	短路 (Short-circuit) 且運算	x && y;
||	短路 (Short-circuit) 或運算	x || y;
?:	if-then-else 三元運算子	a>10 ? x : y

真理大學 Aletheia University

2-4-5 邏輯運算子

- 「&」、「|」、「!」、「^」邏輯運算子的執行結果會傳回「true」或是「false」值，執行的結果可以用下表來表示：

A 敘述	B 敘述	A & B	A | B	!A	A^B
true	true	true	true	false	false
false	true	false	true	true	true
true	fasle	false	trus	false	true
false	false	false	false	true	false

- 上表中雙線框線的儲存格表示執行的條件。如果 A 敘述的執行結果為「true」，而且 B 敘述的執行結果也是「true」，則敘述「A & B」的執行結果會是「true」。

Aletheia University

2-4-5 短路運算子

- Java 提供了兩個很有趣的布林邏輯運算子，「 && 」和「 || 」。您可以把它當做是「 & 」和「 | 」的加強版。
- 如果運算子前的敘述的執行結果為「 false 」，則不需要再執行「 && 」後的敘述即可將「 false 」的結果回傳。 例如:

```
result = (a<5) && (b<10);        //result 的值為 false
```

  - 則「 b < 10 」敘述不會被執行。

- 「 || 」運算子的執行概念也是相同，如果運算子前的敘述的執行結果為「 true 」，則不需要再執行「 || 」後的敘述即可將「 true 」的結果回傳。

真理大學 Aletheia University

2-4-6 「？」三元運算子

- Java 程式包含了一個特殊的三元運算子「？」，某種程度上，它可以取代「if-then-else」敘述，是一種較有效率的寫作方式。
- 「？」一般的格式為：

```
expression ? result1 : result2;
```

- 敘述代表：「如果 expression 的敘述成立，則回傳「result1」的執行結果；如果 expression 的敘述不成立，則回傳「result2」的執行結果」。 ||」後的敘述即可將「true」的結果回傳。 例如：

```
result = x > 0 ? x : -x;
```

真理大學
Aletheia University

2-4-6 位元邏輯運算子

- 當「&」、「|」、「^」、「~」單獨使用於數值時，它們可以是位元邏輯運算子。位元邏輯運算子會對運算元的每一個位元單獨作運算。參考下表:

A	B	A | B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

真理大學 Aletheia University

2-4-6 位元邏輯運算子

- 以較簡單的 NOT (~) 運算子為例，該運算子會將數值的位元反向，例如：數字「30」的二進位可以表示成：

  00011110

- 在套用 NOT 運算後，得到的值為：

  11100001

- AND (&) 運算子只有在輸入的兩個位元都是「1」時，才會產生「1」的輸出位元，其餘的情形都[illegible]算式「30 & 15」的執行結果為「14。」

```
  00011110  30
&
  00001111  15
------------------------
  00001110  14
```

大學 University

2-4-6 位元邏輯運算子

- OR (|) 運算子只有在輸入的兩個位元都是「0」時，才會產生「0」的輸出位元，其餘的情形都會輸出「1」。例如：運算式「30 | 15」的執行結果為「31」

```
  00011110  30
| 00001111  15
------------------------
  00011111  31
```

- XOR (^) 運算子只有在輸入的兩個位元只有一個是「1」時，才會產生「1」的輸出位元，其餘的情形都會輸出「0」。例如：運算式「30 ^ 15」的執行結果為「17」

```
  00011110  30
^ 00001111  15
------------------------
  00010001  17
```

真理大學 Aletheia University

2-5 Java 的物件概念

- Java 是以物件為根基的物件導向程式語言，而物件使用的基本概念是：「物件使用前，必需先要實體化」。物件變數的宣告僅只是配置了存取物件的「**參考 (reference)**」，並未配置該物件的實際儲存空間，物件必需「實體化」後才會有記憶體空間 。
- 物件具有屬性和方法，例如：我們定義了陣列物件時，就可以使用「length」屬性來取得陣列的長度。
- 而物件也具有生命週期，這代表物件不再使用時，它原先所佔用的系統資源必需回收。
  - 但 Java 不允許程式直接的抓取並使用所需的記憶體，抓取記憶體的工作是由 JVM 完成的。
- Java 語言具有「**Garbage Collection**」的機制，JVM 會自行判斷物件是否不再被使用。

真理大學 Aletheia University

2-5-1 Wrapper Class 的使用

- 雖然「物件」的概念是 Java 的重心，但嚴格說來， Java 並不能算是一個純的物件導向程式語言。
  - 例如： Java 中的「 Primitive 」型別的資料並不是物件 。
- 但如果真的有必要，我們也可以將基本資料型別轉換成物件來處理。
- Java 在「 java.lang 」套件下提供了將「 Primitive 」型別包裝的「 **Wrapper Class** 」， 這些類別的作用就如同將「 Primitive 」型別「包裝 (wrap) 」起來，讓「 Primitive 」資料可以當做是物件來操作。
- 各個「 Primitive 」型別相對應的「 Wrapper Class 」請參考下

Primative	char	boolean	byte	short	int	long	float	double
Wrapper	Character	Boolean	Byte	Short	Integer	Long	Float	Double

真理大學 Aletheia University

2-5-2 Integer 類別

- Integer 類別對應到 int 這個基本資料型態，建構 Wrapper class 物件時，可以直接將 int 的值傳入 Wrapper class 物件中。
- 例如：下列的程式碼分別示範如何建立 int 變數和 Integer 物件：

```
int i = 123;              // 宣告一個 int 變數並指定值
// 宣告一個 Integer 物件並指定值
Integer i = new Integer(123);
// 宣告一個 Integer 物件並指定值
Integer i = new Integer("123");
```

真理大學
Aletheia University

2-5-2 Integer 類別

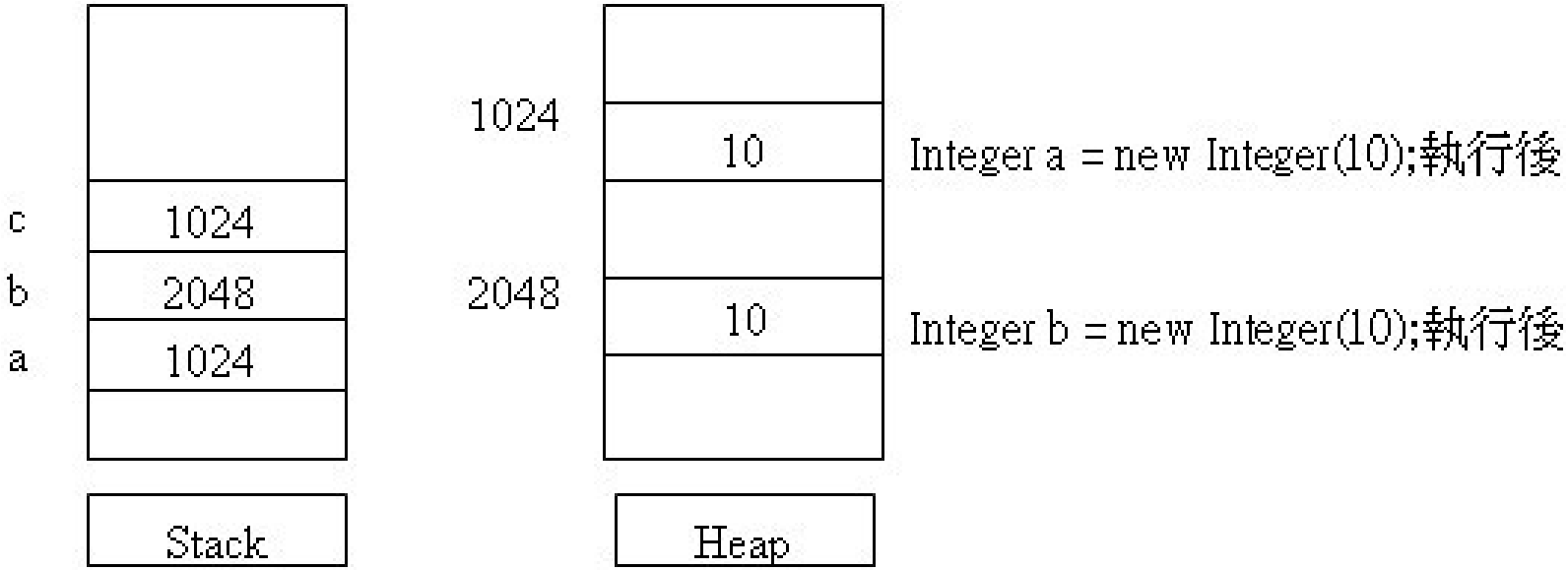
- 比較物件的相同性
  - 如果要比較兩個基本型態變數的大小，我們可以用「＞」、「＜」、「＝」等運算子。
  - 但這些運算子無法使用於物件上。在 Java 中，「**Object**」類別是所有類別基底類別，該類別中提供了「equals」方法來判斷兩個物件變數參考到的內容是否相同。
  - 這裏所謂的相同是指物件變數指向的記憶體位址儲存的內容，而不是物件變數中儲存的記憶體位址。

真理大學
Aletheia University

2-5-2 Integer 類別

- 例如以下的範例：

```
Integer a = new Integer(10);     // 產生一個物件，值為 10
Integer b = new Integer(10);     // 產生另一個物件，值也是 10
Integer c = a;                   //c 指向 a 物件
boolean same = c.equals(b);      //c 和 b 的內容相等
```



2-5-2 Integer 類別

- 另一個邏輯運算子「==」也可以用來判斷兩個物件變數的值是否相同。但使用「==」是用來判斷兩個物件變數是否指向同一個物件，只有在兩個物件變數是參考到相同的一個物件時，比較的結果才會是「true」。 例如：

```
Integer a = new Integer(10);          // 產生一個物件，值為 10
Integer b = new Integer(10);// 產生另一個物件，值也是 10
Integer c = a;                        //c 指向 a 物件
boolean same = (c == b);              //c 和 b 指向不同的物件
```

- 由於 c 和 b 是指向不同的物件，所以 same 的值會是 false 。

真理大學 Aletheia University

2-5-2 Integer 類別

- 要比較兩個 Wrapper Class 物件所存放的數值是否相同，我們可以使用「 equals 」方法。
- 如果要比較兩個 Wrapper Class 物件所存放的數值的大小，我們可以使用「 compareTo 」方法。
  - compareTo 會傳回一個整數值，
    - 如果該數值是 0 ，則代表兩個物件存放的數值相等。
    - 如果傳回大於 0 的數值，則代表呼叫此方法的物件大於傳入該方法的物件的值；
    - 如果傳回小於 0 的數值，則代表呼叫此方法的物件小於傳入該方法的物件的值。

真理大學
Aletheia University

2-5-3 Character 類別

- Character 類別對應到 char 這個基本資料型態。這個類別提供的方法大多是用來判斷字元的屬性。
- 建構 Character 物件時，需要傳入一個 char 型態的資料。該類別中常用到的類別等級的方法有 ：

方法	作用
static int digit(char ch, int radix)	將字元轉換為對應的 radix 進位的數字
static char forDigit(int digit, int radix)	將 radix 進位的數字轉換為對應的字元
static boolean isDigit(char ch)	判斷 ch 是否是數字字元
static boolean isLetter(char ch)	判斷 ch 是否是字母
static boolean isLetterOrDigit(char ch)	判斷 ch 是否是數字字元或是字母
static boolean isLowerCase(char ch)	判斷 ch 字母是否是小寫
static boolean isSpaceChar(char ch)	判斷 ch 字母是否是空白字元
static boolean isUpperCase(char ch)	判斷 ch 字母是否是大寫
static char toLowerCase(char ch)	將 ch 字母轉為小寫
static char toUpperCase(char ch)	將 ch 字母轉為大寫

真理大學 Aletheia University

2-5-4 Boolean 類別

- Boolean 類別對應到 boolean 這個基本資料型態。
- 建構 Boolean 物件時，您可以傳入一個 boolean 值或是傳入一個字串。如果您傳入的字串是「true」，則不論該字串是大寫或是小寫，Boolean 物件存放的值都是「true」，如果您傳入其他的字串時，則 Boolean 物件存放的值都是「false」。
- 參考以下的程式碼:

```
boolean b;
b = new Boolean(true);                  //b 為 true
b = new Boolean(false);                 //b 為 false
b = new Boolean("True");                //b 為 true
b = new Boolean("Qoo");                 //b 為 false
```

真理大學 Aletheia University

2-5-5 Byte 類別

- Byte 類別對應到 byte 這個基本資料型態。
- 建構 Byte 物件時，您可以傳入一個 byte 值或是傳入一個字串。如果您傳入字串的格式不符合 byte 格式時，程式執行時會產生 NumberFormatException 例外物件。
- Byte 類別中常用到的方法有 ：

方法	作用
`static byte parseByte(String s)`	將字串 s 轉換為 byte
`static Byte valueOf(byte b)`	將 b 轉為 Byte 物件

真理大學 Aletheia University

2-5-6 Boxing 與 AutoBoxing － J2SE 5.0

- 將 Primitive 資料轉換為相對應的 Wrapper 物件的操作稱為「 Boxing 」；轉換時，必需自行建構 Wrapper 物件。
- 而將 Wrapper 物件轉換為相對應的 Primitive 資料的操作稱為「 UnBoxing 」，使用時，必需指定相關的方法傳回需的值。
- 在 J2SE 5.0 新增了「 AutoBoxing 」的功能，可由 Complier 在「 Boxing 」時自動的產生 Wrapper 物件或是在「 UnBoxing 」時自動取出 Wrapper 物件中的值。例如：

```
int p = 600;
// 沒有使用 AutoBoxing，要自行建構 Wrapper 物件
Integer v1 = new Integer(p);
// 使用 AutoBoxing，由 Compiler 自動產生 Wrapper 物件
Integer v2 = p;
// 沒有使用 AutoBoxing，使用 intValue 方法取得值
int p2 = v1.intValue();
int p3 = v1;  // 使用 AutoBoxing，由 Compiler 自動取出值
```

真理大學 Aletheia University