

# 初探 Java

TYIC桃高資訊社

# 第一個 Java 程式

寫好了第一個 Java 程式  
但是自己在寫什麼  
自己也不知道

```
01 // class 名稱必須跟檔案名稱一樣
02 public class Main {
03
04     // Java 程式需要一個主方法(main 方法)，程式從這裡開始執行
05     public static void main(String[] args) {
06
07         // 在 Java 中，使用 System.out.println() 來輸出資料
08         System.out.println("Hello, World!");
09
10     }
11 }
```



java

# 第一個 Java 程式

第 1、4、7 行：

"//" 表示是單行註解(Comment)

程式執行會忽略 "//" 和該行後面的所有文字

```
01  // class 名稱必須跟檔案名稱一樣
02  public class Main {
03
04      // Java 程式需要一個主方法(main 方法)，程式從這裡開始執行
05      public static void main(String[] args) {
06
07          // 在 Java 中，使用 System.out.println() 來輸出資料
08          System.out.println("Hello, World!");
09
10      }
11  }
```



java

# 第一個 Java 程式

第 2 行：`public`、`class` 都是保留字(Reserved word)  
有著特定功能，之後的課程會說到  
"`Main`" 是類別(`class`)名稱



```
01 // class 名稱必須跟檔案名稱一樣
02 public class Main {
03
04 // Java 程式需要一個主方法(main 方法)，程式從這裡開始執行
05 public static void main(String[] args) {
06
07 // 在 Java 中，使用 System.out.println() 來輸出資料
08 System.out.println("Hello, World!");
09
10 }
11 }
```



java

# 第一個 Java 程式

第 5 行：`public`、`static`、`void` 也都是保留字

"`main`" 是方法名稱，"`args`" 是一個參數(`parameter`)

"`String[]`" 是 `args` 參數的型別(`type`)，之後的課程會說到

```
01 // class 名稱必須跟檔案名稱一樣
02 public class Main {
03
04     // Java 程式需要一個主方法(main 方法)，程式從這裡開始執行
05     public static void main(String[] args) {
06
07         // 在 Java 中，使用 System.out.println() 來輸出資料
08         System.out.println("Hello, World!");
09
10     }
11 }
```



java

# 第一個 Java 程式

第 8 行："System.out.println()" 是一個方法(method) 用來輸出小括號裡面放的是要輸出的東西(引數argument)，這裡放的是「"Hello, World!"」，所以會輸出 "Hello, World!"

```
01 // class 名稱必須跟檔案名稱一樣
02 public class Main {
03
04     // Java 程式需要一個主方法(main 方法)，程式從這裡開始執行
05     public static void main(String[] args) {
06
07         // 在 Java 中，使用 System.out.println() 來輸出資料
08         System.out.println("Hello, World!");
09
10     }
11 }
```



java



# 第一個 Java 程式

在 Java 程式碼中

每一個陳述式(statement)後方都要一個分號，且一定要單獨一行  
在此程式中，第 8 行就是一個陳述式

```
01 // class 名稱必須跟檔案名稱一樣
02 public class Main {
03
04     // Java 程式需要一個主方法(main 方法)，程式從這裡開始執行
05     public static void main(String[] args) {
06
07         // 在 Java 中，使用 System.out.println() 來輸出資料
08         System.out.println("Hello, World!");
09
10     }
11 }
```



java

# 註釋

我們在第一個程式中說過，"**//**" 表示是單行註解

程式會忽略 "**//**" 和該行後面的所有文字

還有另一種註解是多行註解

程式會忽略夾在 "**/\***" 和下一個 "**\*/**" 中間的所有文字

```
System.out.println("會輸出(沒有被單行註解)");  
// System.out.println("不會輸出(被單行註解)");  
System.out.println("會輸出(多行註解前)");  
/*  
System.out.println("不會輸出(被多行註解)");  
System.out.println("不會輸出(被多行註解)");  
System.out.println("不會輸出(被多行註解)");  
*/  
System.out.println("會輸出(多行註解後)");
```

java

```
會輸出(沒有被單行註解)  
會輸出(多行註解前)  
會輸出(多行註解後)
```

output



# 基本輸出

我們在第一個程式中說過

"**System.out.println()**" 是一個用來輸出東西的方法且會換行

如果不想換行可以使用 "**System.out.print()**" 方法

能輸出的也不只文字，如：'a'、2147483647、3.14159、true

```
01 public class Main {
02     public static void main(String[] args) {
03         System.out.println("a");
04         System.out.println('a');
05         System.out.println("2147483647");
06         System.out.println(2147483647);
07         System.out.println("3.14159");
08         System.out.println(3.14159);
09         System.out.println("true");
10         System.out.println(true);
11     }
12 }
```



java

```
[ a
[ a
[ 2147483647
[ 2147483647
[ 3.14159
[ 3.14159
[ true
[ true
```

output

觀察每兩行有什麼差別？

# 基本輸出

```
01 public class Main {
02     public static void main(String[] args) {
03         System.out.println("a");
04         System.out.println(a);
05         System.out.println(2147483647);
06         System.out.println(2147483647);
07         System.out.println(3.14159);
08         System.out.println(3.14159);
09         System.out.println(true);
10         System.out.println(true);
11     }
12 }
```



java

```
[ a
[ a
[ 2147483647
[ 2147483647
[ 3.14159
[ 3.14159
[ true
[ true
```

output

顯而易見的，程式碼奇數行有一對雙引號，而偶數行沒有  
這是因為奇數行和偶數行括號裡的東西的資料型態不一樣的關係  
使用一對雙引號 "" 夾起來的才是字串(**String**)，其餘則不是  
這與資料型別(**Data type**)有關

# 基本資料型別(primitive data types)

Java 中總共有 8 種基本資料型態：

byte	short	int	long	float	double	char	boolean
直接表示，值的範圍為-128到127的整數	直接表示，值的範圍為-32768到32767的整數	直接表示，值的範圍為 $-2^{31}$ 到 $2^{31}-1$ 的整數	整數後方加L表示，值的範圍為 $-2^{63}$ 到 $2^{63}-1$ 的整數	小數後方加f表示，值的範圍約為 $3.4\text{E}-38$ 到 $3.4\text{E}+38$	直接表示，值的範圍約為 $1.7\text{E}-308$ 到 $1.7\text{E}+308$	用一對單引號 '' 表示，單引號裡面只能放一個字	直接表示，值只有true和false，分別代表「真」與「假」
-1 24	-2222 1024	-2147 83648	999999L -77777L	6.0734f -2.887f	-2284.1 3.55555	'a' 'c'	true false

像這樣直接寫下來的叫做字面常數(literal constant)，是值(value)的一種

# 數字

8種基本資料型態中，整數表示的有 `byte`、`short`、`int`、`long`

這四種不只可以直接以十進位表示，也可以加些東西用其他進位表示

二進位(**Binary**)：在二進位數字前加上"`0b`"，如"`0b101`"

八進位(**Octal**)：在八進位數字前加上"`0`"，如"`0777`"

十六進位(**Hexadecimal**)：在十六進位數字前加上"`0x`"，如"`0xF4`"

8種基本資料型態中，數字表示的除上面四種整數還有`float`和`double`

這六種不只可以直接表示，還可以在數字之間(含十六進位下A-F)

加上下劃線("`_`"，**underscore**)讓數字更容易閱讀

如：`"0b1_0_1"`、`"0_777"`、`"1912_01_01"`、`"0xF_4"`

# Char

Char 在電腦內部實際上是儲存一個 0 ~ 65535 的整數

所以 Char 也是數字的一種

這 0 ~ 65535 的整數當中有許多數字各自對應了一個字元

而這個對應是根據UTF-8的基本多文種平面(Basic Multilingual Plane，簡稱BMP、0號平面、Plane 0)來決定

當中除了完全兼容 ASCII(American Standard Code for Information Interchange，美國標準資訊交換碼)

還有新增中日韓統一表意文字，也就是常見的漢字

以及拉丁字母、特殊字元、中日韓符號和標點、康熙部首等

# ASCII

ASCII 是相當重要的編碼

其中包含了英文字母、數字符號、特殊符號、控制字元  
當中較為重要的是：

32：空格(space)

48：0

65：A

97：a

數字 0-9、英文 a-z、A-Z 皆可直接按照順序推下去

# ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

# 變數(Variable)

在 Java 中可以宣告(**declare**)變數，宣告的方式有兩種：

```
資料型別 變數名稱; // 第一種，未初始化
```

```
資料型別 變數名稱 = 值; // 第二種，已初始化
```

java

第一種只是宣告變數，沒有初始化(**Initialization**)，使用前一定要初始化

第二種是宣告變數，並初始化變數，且值的資料型別必須和變數相同

兩種都是陳述式，所以皆須單獨一行，且結尾須有個分號

已經宣告過的變數不可以再宣告。舉例：

```
byte a;  
short b = 0;  
int c = 2147_4836_47;  
long d = 29999999999L;
```

java

```
float e = 1.414f;  
double f = 6.8;  
char g = 'z';  
boolean h = true;
```

java



# 變數

在 **Java** 中，賦值(指定，**assign**)給變數的方式如下：

```
變數名稱 = 值;
```

java

若變數還沒有初始化，則這個陳述式就是初始化變數

若變數已初始化，則這個陳述式就是重新賦值給變數，

且值的資料型別必須和變數相同

賦值可以是陳述式也可以是表達式。舉例：

```
a = 2;
```

```
b = 4;
```

```
c = -2147_4836_48;
```

```
d = 999999999999999L;
```

java

```
e = 0.99999f;
```

```
f = 0.999999999999999;
```

```
g = ' '; // 空白也是一個字元
```

```
h = false;
```

java


# 變數

變數代表一個值

所以任何可以填值的地方都可以填變數

舉例：

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println(2);  
        int a = 2;  
        System.out.println(a);  
        a = 0;  
        System.out.println(a);  
        System.out.println(a = 3); // 賦值作為表達式  
        System.out.println(a);  
    }  
}
```

  
java

```
2  
2  
0  
3  
3      output
```

# 變數命名規則

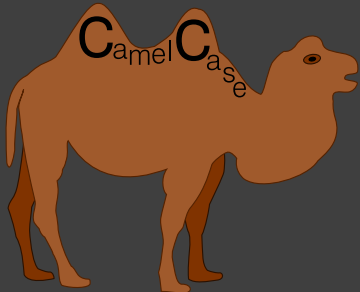
在 **Java** 中，變數命名「一定要」遵守以下規則：

1. 只能由 **a-z**、**A-Z**、**0-9**、**\$**、**\_** 組成
2. 開頭不能是數字
3. 不能是保留字

在 **Java** 中，變數命名「盡可能」遵守以下規則：

1. 名稱要有意義，避免 **a**、**b**、**c** 這種名稱，除非是臨時變數
2. 使用小駝峰式命名法(**lowerCamelCase**)

如：**apple**、**applePen**、**penPineappleApplePen**



大駝峰式命名法  
(**Pascal**命名法)：  
每個單字首字母大寫  
其餘小寫  
且每個單字中間直接連接

小駝峰式命名法：  
第二個單字起每個單字  
首字母大寫其餘小寫  
且每個單字中間直接連接



# 常數(Constant)

若在宣告變數時加上**final**，則在初始化後不可以被重新賦值：

```
final 資料型別 變數名稱; // 第一種，未初始化
```

```
final 資料型別 變數名稱 = 值; // 第二種，已初始化
```

java

其餘用法與變數完全一，舉例：

```
class Main {  
    public static void main(String[] args) {  
        final int a;  
        a = 10;  
        System.out.println(a);  
        a = 100; // Compile error: variable a might already have been assigned  
        System.out.println(a);  
    }  
}
```

java

# 常數命名規則

在 **Java** 中，常數命名規則基本上與變數命名規則一樣  
但建議使用

蛇行命名法(`snake_case`、`lower_case_with_underscores`)  
的變種

**SCREAMING\_SNAKE\_CASE**(**UPPER\_CASE\_WITH\_UNDERSCORES**)：

每個字母都大寫

且每個單字之間用下劃線連接

如：**PEN**、**APPLE\_PEN**、**PEN\_PINEAPPLE\_APPLE\_PEN**、**PI**

# 命名規則

如果沒有遵守命名規則...

```
final double abcde = 3.14_159;  
final int nINeTYniNe = 99;  
System.out.println(abcde);  
System.out.println(nINeTYniNe);
```

java

遵守命名規則後：

```
final double PI = 3.14_159;  
final int ninetyNine = 99;  
System.out.println(PI);  
System.out.println(ninetyNine);
```

java

# 運算(operation)

只有基本型別可以進行運算

每個運算都由運算元(operand)及運算子(operator)組成

且每個運算都會返回一個結果

以加法運算為例：

```
6 + 8
```

java

其中 6 和 8 為運算元，為參與運算的值

"+" 為運算子，表示運算的類型

運算元和運算子中間的空格可省略，但不省略較易閱讀

運算元的數量及型別，視運算的類型而定

返回結果的型別，視運算的類型和運算元的型別而定

# 數學運算

顯然的，數學運算只有數字才能用，但因 `char` 內部也是儲存數字  
所以 `char` 也可以參與數學運算，且會自動轉為 `int` 進行運算

byte	short	int	long	float	double	char	boolean
直接表示， 值的範圍為 -128 到 127 的整數	直接表示， 值的範圍為 -32768 到 32767 的整數	直接表示， 值的範圍為 $-2^{31}$ 到 $2^{31}-1$ 的整數	整數後方 加L表示， 值的範圍為 $-2^{63}$ 到 $2^{63}-1$ 的整數	小數後方 加f表示， 值的範圍約為 $3.4\text{E}-38$ 到 $3.4\text{E}+38$	直接表示， 值的範圍 約為 $1.7\text{E}-308$ 到 $1.7\text{E}+308$	用一對 單引號 '' 表示， 單引號裡面 只能放 一個字	直接表示， 值只有true 和false， 分別代表 「真」與 「假」
-1 24	-2222 1024	-2147 83648	999999L -77777L	6.0734f -2.887f	-2284.1 3.55555	'a' 'c'	true false



# 一元數學運算

運算名稱	正數運算	負數運算
格式	+運算元	-運算元
功能	把數字加上正號 = <b>Do Nothing</b>	把數字變 為相反數
結果型別	與運算元型別相同	
範例	<b>+1</b> <b>+(-2)</b>	<b>-1</b> <b>-(-2)</b>

# 二元數學運算

運算名稱	加法運算	減法運算	乘法運算
格式	運算元1 + 運算元2	運算元1 - 運算元2	運算元1 * 運算元2
功能	運算元1 + 運算元2	運算元1 - 運算元2	運算元1 × 運算元2
結果型別	數字型別順序：double>float>long>int>short>byte 結果型別為兩個運算元中數字型別順序大的		
範例	1 + 2 5 + -9	1 - 2 -5 - 9	1 * 2 -5 * -9

# 二元數學運算

運算名稱	除法運算	取餘運算
格式	運算元1 / 運算元2	運算元1 % 運算元2
功能	運算元1 ÷ 運算元2	運算元1 ≡ 結果 (mod 運算元2)
結果型別	兩個運算元中 數字型別順序大的	
範例	4 / 2 17 / -9	1 - 2 -5 - 9

# 溢位(Overflow)

如果數字超過了可以儲存的範圍  
那麼數值就會發生溢位

# 一元運算

一元運算是指只有一個運算元的運算

邏輯 否定(NOT)運算	負數運算	正數運算	位元(Bitwise) 否定運算
boolean	byte、short、int、 long、float、double		byte、short 、int、long
真(true)變假 假(false)變真	把數字變 為相反數	把數字加上正號 = Do Nothing	把二進制的0改成1、1改成0 = 取 1 補數
!運算元	-運算元	+運算元	~運算元
!false !true	-1 -(-2)	+1 +(-2)	~2 ~-3

# 二元運算

二元運算是指有兩個運算元的運算

邏輯 且(AND)運算	邏輯 或(OR)運算	邏輯 或(OR)運算	
boolean	boolean	boolean	
都為真即為真 否則為假	有一真即為真 否則為假	有一真即為真 否則為假	
!運算元	!運算元	!運算元	
!false !true	!false !true	!false !true	