Data types and Variables

變數資料型態

Java Fundamental

Contents

- 變數命名規則
- 基本資料型態
- 空白與註解

Contents

- 變數命名規則
- 基本資料型態
- 空白與註解

Java的命名-識別字 (Identifier)

- Java的命名語法是使用識別字 (Identifier)命名。
- 識別字是指變數 (Variable) 、方法 (Method) 、類別(Class) 的名稱
- 除了變數的命名外，在程式中還需要使用在常數、變數、類別和方法命名，這些名稱都屬於識別字。

Java的命名-識別字 (Identifier)

- 識別字是使用英文字母開頭，不限長度的Unicode統一字碼字元的字串，包含字母、數字和底線”_”。例如：一些合法的名稱範例，如下所示：

  T, n, size, z100, long_name, helloWord, Test, apple

- 名稱區分英文字母的大小寫，例如：apple、Apple和APPLE屬於不同變數。
- 不能使用Java語法的「關鍵字」（Keyword）、保留字的布林值true或false和null。
- 名稱在「範圍」（Scope）中是唯一的。在程式中可以使用相同的變數名稱，不過各變數名稱需要在不同的範圍。

Identifier 識別字命名規則

- 識別字限以下列四種字元組成
  - Unicode letters： A-Z, a-z, 中文, …
  - Underscore：_
  - Dollar sign：$, ¥, £…
  - Digits：0,1,…9 (不能出現在起始字元)
- 不可為關鍵字(keywords)及保留字(Reserved words) 。
- 不能有標點符號、空白，或是 - 。
- 大小寫有別 Case sensitive 。
- 第一個字只能是 $ 或是 _ 或是 字母 (Letter),第二個字開始可以有數字。

識別字的命名慣例

◆為了程式撰寫的一致性和易於閱讀，Java習慣的命名原則，如下表所示：

識別字種類	習慣命名原則	範例
常數	使用英文大寫字母和底線”_”符號	MAX_SIZE、MIN_SIZE
變數	英文小寫字母開頭，如果 2 個英文字組成，第 2 個英文字以大寫開頭	size、screenSize、myAccountNumber
類別	英文大寫字母開頭，如果是 2 個英文字組成，第 2 個英文字也使用大寫開頭	LargeRoom、SmallRoom
函數（方法）	英文小寫字母開頭，如果是 2 個英文字組成，第 2 個英文字使用大寫開頭	pressButton、scrollScreen

識別字的命名慣例

- 類別(class)名稱
  - 名詞
  - 第一個英文字母大寫, 複合字用大寫區隔
  - Ex: SavingAccount
- 屬性(attribute)名稱
  - 名詞
  - 第一個英文字母小寫, 複合字用大寫區隔
  - Ex: userName

識別字的命名慣例

- 方法(method)名稱
  - 動詞
  - 第一個英文字母小寫, 複合字用大寫區隔
  - Ex: calculateInterest()
- 常數名稱
  - 利用 final 來修飾屬性成有常數
  - 全大寫
  - Ex: final double PI = 3.1415926

變數的命名

- 變數名稱不能為關鍵字或保留字。
- 變數名稱的第一個字元可以為Unicode字元(如: 中文字)、英文字母、 底線符號(_)或者金錢符號($、£ 、 ¥)，第二個字元以後，可以使用前述字元及數字字元。
- 避免在變數名稱中使用$ 。
- 在相同範圍下不能重複宣告一個變數名稱。

Java關鍵字和保留字

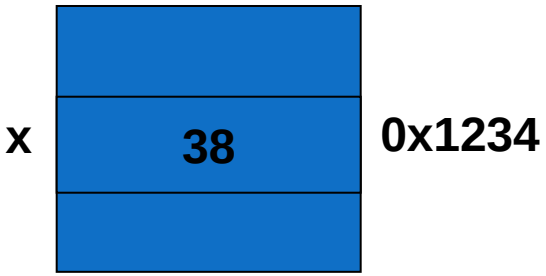
◆程式語言本身定義的識別字即是「關鍵字」或「保留字」。

Java的關鍵字				
abstract	assert	boolean	break	byte
case	catch	char	class	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

變數宣告

- 變數的目的是儲存程式執行中的一些暫存資料，程式設計者只需記住變數名稱，而且知道名稱表示一個記憶體位置中的資料，至於這個記憶體位置到底有哪裡？並不用傷腦筋，因為這是編譯程式的工作。
- 簡單的說，程式語言的變數是使用有意義的名稱代表數字的記憶體位址。
- 宣告變數的用意：
  - 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
  - 不會發生資料因無心而蓋掉的錯誤 (相同名稱的變數不能重複宣告) 。

int x = 38;



變數宣告語法

- Java變數宣告的語法，如下所示：

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

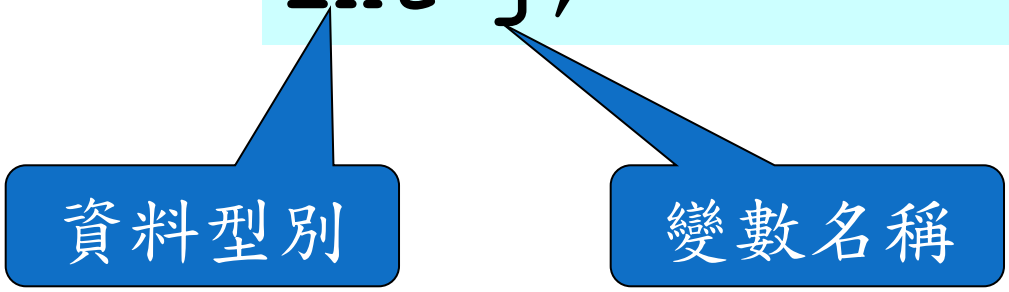
  資料型態　變數名稱;

- 宣告格式使用資料型態開頭，後面跟著變數名稱，目的是「配置一個宣告資料型態的變數」。

變數宣告-語法

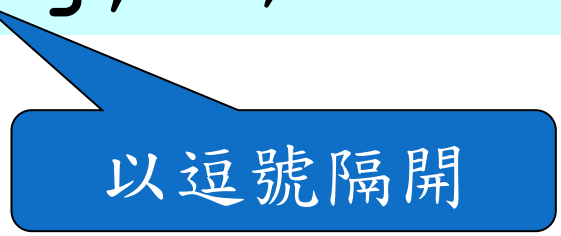
◆語法：

```
int j;
```



◆宣告多個變數：

```
int i, j, k;
```



變數宣告-設定初值

- 變數宣告單純只是表示保留記憶體空間，至於儲存的資料是什麼？可以在宣告時指定初始值，或是使用指定敘述在程式碼設定變數值。
- 在Java宣告變數且指定變數初值，如下所示：
  - int score = 85;
  - int i = 20;

指定敘述（Assignment Statement）

- 「指定敘述」（Assignment Statement）是在程式中存取指定變數的值，如果在宣告變數時沒有指定變數的初值，就可以使用指定敘述 "=" 等號指定變數值或更改變數值，其語法如下所示：
  - 變數 = 運算式;
- 在指定敘述的左邊是變數名稱，右邊是「運算式」（Expression），運算式可以是Java運算子和運算元組成的任何運算式。
- 簡單的說，指定敘述是「將右邊運算式的運算結果指定給左邊的變數」。
  - int score = 85;
  - int i;

    I=20;

指定敘述-範例

- 指定敘述的範例，如下所示：
  - balance = 6000;
- 上述程式碼指定變數balance的內容，運算式是數值6000

變數設值

- 語法：

```
i = 24;
```

將24設定給變數i

- 宣告並設值：

```
int a = 123;
int j = 520, k = 240;
```

變數宣告 (Variable Declaration)

<type> <identifier> [=initial_values];

Initialize literal values

Name of the variable

Primitive or reference type

- 資料型態
  - 適當記憶體儲存空間
- 變數名稱
  - 以該名稱來取得儲存值
- = 指定運算子
  - 右邊的值存到左邊的記憶體


```
01  public class VariableDemo {
02      public static void main(String[] args) {
03          int age = 18;
            char gender = 'M';
            System.out.print("年齡:");
            System.out.println(age);
07          System.out.print("性別:");
08          System.out.println(gender);
09
10          age = 20;
            gender = 'F';
            System.out.print("年齡:");
13          System.out.println(age);
14          System.out.print("性別:");
15          System.out.println(gender);
16      }
    }
```

宣告變數並賦值

取得變數值

變更變數值

跳脫字元 (Escape Character)

- 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某字元直接撰寫時，被Java當作特定意義的語法，會使用跳脫字元來表示。

跳脫字元	意義
\b	退一格(backspace)。
\f	跳頁(form feed，使用於印表機)。
\n	換行(newline)。
\r	返回(carriage return)，游標移至行首。
\t	水平跳格，相當於按鍵盤的Tab鍵。
\\	表示 \ 字元。
\’	表示’字元。
\”	表示”字元。
\uXXXX	以十六進位指定Unicode字元輸出
\XXX	以八進位指定Unicode字元輸出

字面常數（Literal Constant）

- 「字面常數」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。

- 常見的常數有：
  - 數值常數：

    例：12, 9.6, -456, -999.9 。
  - 字元常數：用單引號「 ‘」一前一後包起來。

    例： ‘j’ , ‘3’ , ‘我’
  - 字串常數：以雙引號「” 」一前一後包起來。

    例： “沒喝過Java,但我會用Java”

常數變數 (Constants) 的宣告與使用

- 常數變數是指在程式執行的過程中，值為固定不變的資料項目。
- 簡單的說，就是在程式中使用一個名稱代表一個固定值。
- Java的常數變數宣告和變數宣告相同，只需在前面使用final關鍵字，如下所示：

  語法：final *DataType VarName = InitialValue*;

```
final      type          name    [= value];
final   型別關鍵字    常數名稱 [= 常數數值];
```

```
final      float         pi      = 3.14159f;
```

- 一旦變數被宣告成 final，其值將不可再更改，若程式欲更改 final 變數的值將造成編譯時期錯誤(compile-time error)

```
final double PI = 3.1416;

PI = 2.345; // 會發生錯誤，因為 PI 是常數
```

常數變數 (Constants) 的宣告與使用

- final是常數宣告的保留字不可省略。
- 常數不一定要在宣告時就設定初始值，但一旦設定初值後，就不能更改。
  - 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
- 若要變更值時，只能變更初值。
- 避免不小心被更動。(若不小心在程式的其他地方變更其值時，會出現編譯錯誤訊息)

Contents

- 變數命名規則
- 基本資料型態
- 空白與註解

Java的資料型態

- Java語言是「強調型態」（Strongly Typed）程式語言，變數儲存的值需要指定資料型態。
- 資料型態的目的是告訴編譯程式宣告的變數準備儲存什麼樣的資料，而且不論如何存取變數值，基本上都不能更改變數的資料型態。
- Java的資料型態分為「基本」（Primitive）和「參考」（Reference）兩種資料型態：
  - 基本資料型態：
    - 變數有byte、short、int、long、float、double、char和boolean共8種資料型態。
  - 參考資料型態：
    - 變數值是一個記憶體位置，這個位置值是物件儲存的位置。
    - 變數有字串(String)、陣列(Array)、物件(Object)

Java的資料型態

- 基本型態(Primitive Type)
  - 整數
  - 浮點數
  - 字元
  - 布林值
- 參考型態(Reference Type)
  - 物件
  - 字串
  - 陣列

Java基本資料型別(Primitive Type)

資料類別	資料型態	位元數(bits)	資料範圍(Range)	範例	初始值
整數	byte	8	-128 ~ 127	2, -114	0
	short	16	-32,768 ~ 32,767	2, -32699	0
	int	32	-2^{31}~2^{31}-1	2, -147344778	0
	long	64	-2^{63}~2^{63}-1	2, -2036854708L	0L
浮點數	float	32	-3.4E+38 ~ 3.4E+38	99F, -32699.01F	0.0F
	double	64	-1.7E+308 ~1.7E+308	-111, 21E12	0.0D
布林值	boolean	1	只能有 true 或 false	true, false	false
字元	char (Unicode)	16	'\u0000' ~ '\uFFFF' 0 ~ 65535	'A', '3', '中', '\n','\u0063'	'\u0000'

- 溢位(Overflow) 錯誤: 當數值超過資料型態範圍的錯誤

整數

型別關鍵字	名稱	佔用記憶體	數值範圍
byte	位元組	8 bits	-128 ~ 127（$-2^7 \sim 2^7-1$）
short	短整數	16 bits	-32768 ~ 32767（$-2^{15} \sim 2^{15}-1$）
int	整數	32 bits	-2147483648 ~ 2147483647 （$-2^{31} \sim 2^{31}-1$）
long	長整數	64 bits	$-2^{63} \sim 2^{63}-1$

整數

- 八進位和十六進位的整數表示法：

```
int j = 012;          //八進位的變數值設定,是以數字0為開頭
int k = 0x2a;         //十六進位的變數值設定
int m = 0x1B0C;       //十六進位的變數值設定
```

- 長整數表示法：

```
long a = 123;         //以一般整數設定給長整數變數
long b = 123L;        //以長整數設定給長整數變數
```

整數

- 超出int的範圍，應在尾端加上l或L：

```
long a = 123456789123;                  //錯誤
long b = 123456789123L;                 //正確
```

- long型別的數值不可設定給int型別的變數：

```
int c = 123L;           //錯誤的設值
```

浮點數

型別關鍵字	名稱	佔用記憶體	數值範圍
float	單精數	32 bits	1.4E-45 ~ 3.4E+38 -1.4E-45 ~ -3.4E+38
double	倍精數	64 bits	4.9E-324 ~ 1.79E+308 -4.9E-324 ~ -1.79E+308

- float 的準確度到小數點下7位數，double到15位數，超出部份會被去掉(四捨五入)

  例：987654321.123456789f (float) ➔ 9.8765432E8

  987654321.123456789 (double) ➔ 9.8765432112345678E8 (四捨五入)

浮點數

- 浮點數預設型別為double，所以只要數值中出現小數點或E時，皆視為double 。
- double型別的值不能設定給float型別的變數。

```
float ft1 = 1.23f;
float ft2 = 1.234F;
float ft3 = 1.234;
float ft3 = 10E2;
```

錯誤

字元

- Java 採用 Unicode 編碼，每個字元佔 2 bytes，包括中文、日文、韓文、德文等。
- 字元以單引號（’ ）包起來表示。

```
char c = ‘A’;
char d = ‘字’;
```

- 字元變數的設值可以使用一般整數設定，範圍為 0~65535（0xffff） 。

```
char c = 26131;
```

- 因採用 Unicode 編碼，因此可採用Unicode表示法：
  - 以 ‘\uXXXX'表示一個字元，其中X為十六進位的數字，且必須為 4 位數。

```
char a = ‘\u0063’;   //與 ’c’ 相同；
char b = ‘\u0035’;   //與 ’5’ 相同；
char c = ‘\u004D’;   //與 ’M’ 相同；
char d = ‘\u0021’;   //與 ’!’ 相同；
```

布林值 (boolean)

- 布林值也叫真假值，不是真 (true) ，就是假 (false) 」。
- 宣告布林變數使用**boolean**關鍵字，設定其值時只能使用 true 或 false，或者運算結果為 true 或 false 的運算式，通常用來判斷儲存的資料內容為「真 (成立) 」或是「假 (不成立) 」。

```
boolean b1 = true;
boolean b2 = 18 > 17;
```

18大於17，故得true

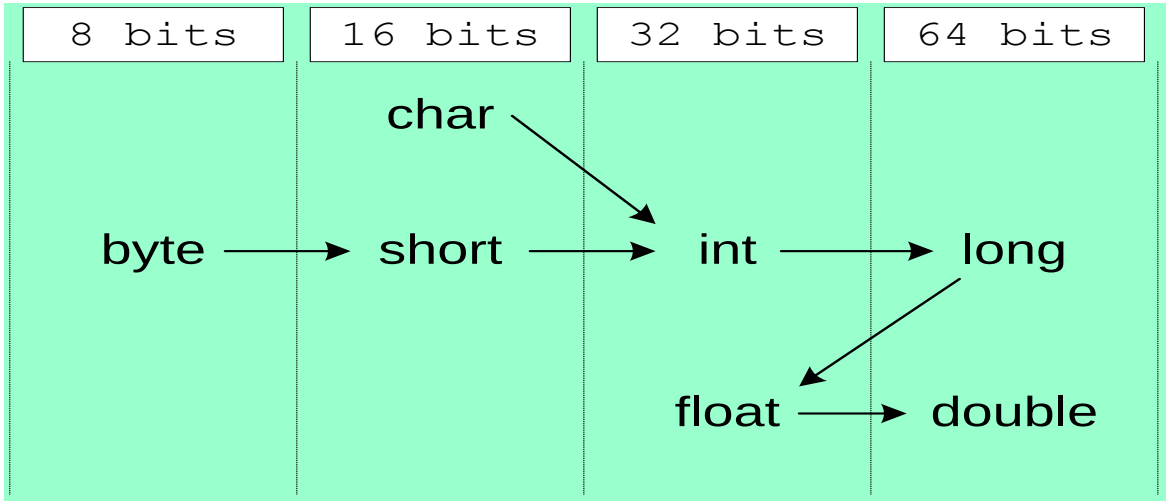
- 我們不能在布林變數中儲存0 、 1 或是其他的數值。
- 布林變數的預設值是「 false 」。

Java 資料型別的轉換

- 每一個變數宣告之後即有屬於自己的型別，往後只能指定給相同型別的變數儲存。
- 若欲指定給不同型別的變數儲存，就需進行型別轉換(Casting)。
- Java 提供資料型別轉換的機制，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。
- Java 語言中針對資料型別提供了2種資料型別轉換：
  - 隱含式的轉換(Implicit Casting)
  - 強制式的轉換(Explicit Casting)

隱含式的轉換 (Implicit Casting)

- 以較小的資料型別轉成較大的資料型別，也稱為自動轉型。
- 系統會根據程式的需要自動且適時地做轉型，並確保資料不會流失。
- 例如，將short轉為int或long，因後者的值範圍比前者大，所以可順利的轉換。
- 自動型別轉換圖解



隱含式的轉換 (Implicit Casting)

```
01  class TestPromotion {
02      public static void main(String[] args) {
03          int  i = 1; double d; float f;
04
05          d  =  i;
06
07          i  =  d;
08
09          f  =  2.5;
10      }
11  }
```

將 i 變數的內容值指派給 d。
合法，implicit Casting。

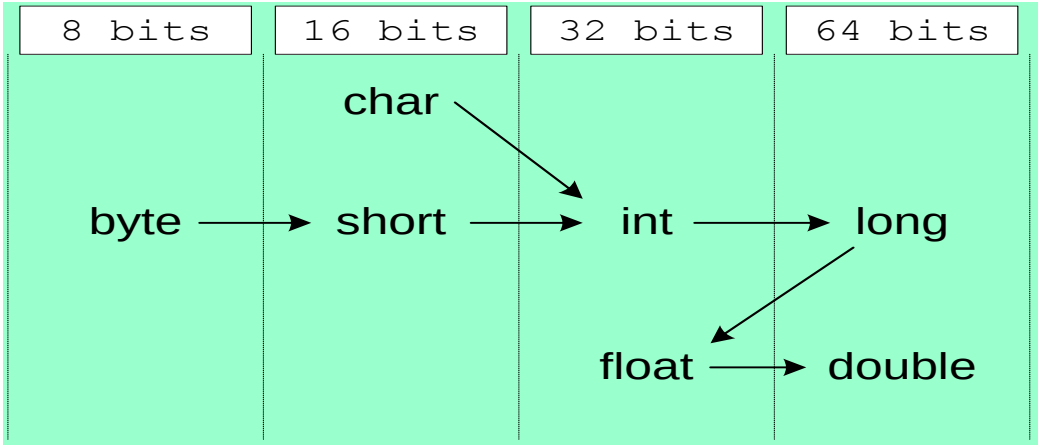
將 d 變數的內容值指派給 i。
不合法，程式需要作 Explicit Casting。 如：i = (int)d;

將 2.5 指派給 f。不合法，
在 Java 語言中 literal 浮點數預設的資料型別是 double 所以必須改寫成
f = (float)2.5; 或
f = 2.5f;

隱含式的轉換 (Implicit Casting)

- 自動型別轉換的限制
  - 同一類型的資料型態，小體積的型別值可以設定給大體積的型別變數，反之則否。

    例:int 可設定給 long, long 不可設定給int.
  - boolean型別不同於其它基本型別，不能和其它基本型別相互成為設定值。
  - char型別可以視為int型別，但所有整數型別皆不能視為char型別。
  - 所有整數型別皆可以設定給所有浮點數型別，反之則否。

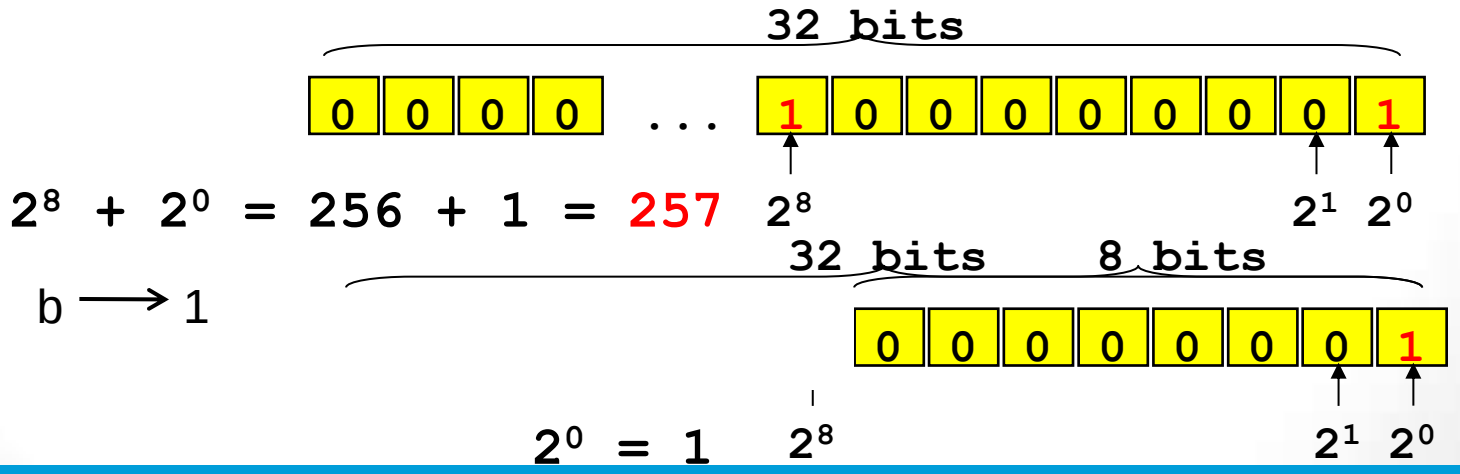


強制式的轉換 (Explicit Casting)

- 以較大的資料型別切割成較小的資料型別，則稱此為強制轉換。

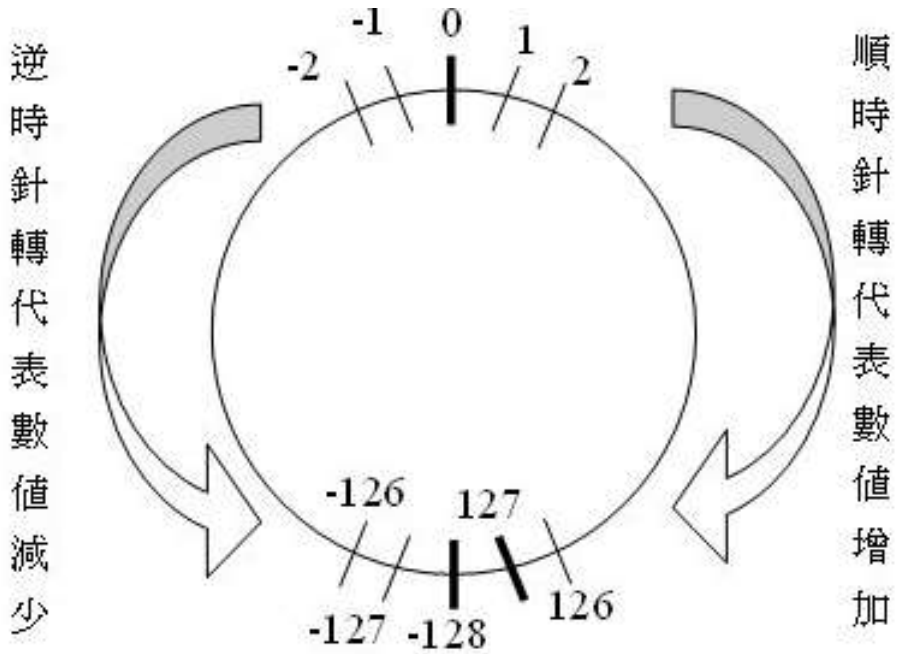
  例如：將 32 bit 的資料放到 16 bit 的資料中。
- 必須在程式中給予明確的指令，系統並不自動轉型。
- 強制轉換的語法

  變數A = (變數A的型別) 變數B
- Ex: byte b = (byte)257;
- 基本資料型別中的強制轉換機制，原則上就是切割資料。



Java 的溢位處理 (Overflow)

- 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

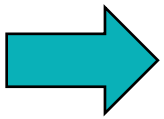
Contents

- 變數命名規則
- 基本資料型態
- 空白與註解

空白 White Space

- 程式設計師可在程式中加入任意數量的空白，包括 tabs、spaces、換行，以增加程式的可讀性

```
{int x; x=23*54;}
```



```
{
        int x;
        x = 23 * 54;
}
```

單行註解

- 單行註解

```
int shirtID = 0; //Default ID for the shirt

// The color codes are R= Red, B= Blue
```

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

多行註解

- 多行註解

```
/*
 this is a multi-line comment
*/
```

```
01  /*
02                  名稱:第一個java程式練習
03                  目的:在螢幕上顯示Hello World!
04  */
05  public class TraditionalCommentDemo {
06      public static void main(String[] args) {
07          System.out.println("Hello World!");
08      }
09  }
```

註解的用途及風格

- 註解的用途
  - 輔助程式設計人員閱讀程式
- 常見註解風格
  1. 變數宣告時使用註解,表明變數的作用

     Ex. int numberOfStudent; //學生的學號
  2. 類別和方法的最後一行加上單行註解,使區塊範圍明顯

```
Ex.    public class HelloWorld {
           public static void main(String[] args){
       …
           } //main 結束
} // HelloWorld 結束
```

註解的用途及風格

- 遇到暫時不想執行的陳述句,可用註解將其註銷,編譯器就不會去處理該指令

```
public class HelloWorld {
        public static void main(String[] args){
    //System.out.println(" Hello World !!! ");
    System.out.println(" 你好 !!! ");
        }
}
```

- 程式的開頭以標題的方式說明程式的名稱、目的

```
/*********************************
 *  變數宣告範圍                 *
 *   Variable Declaration Section  *
 *********************************/
```

Q & A

Exercise

- Which are Primitive data types? (Choose all that apply)

A. char

B. string

C. bit

D. int

E. boolean

F. short

G. long

Exercise

- Given:

```
1.  public class A {
2.    public static void main(String[] argv) {
3.           long x, y = 100;
4.      // insert code here
5.      System.out.println("x = " + x);
6.    }
7.  }
```

Which, inserted at line 4, will compile?(choose three or four)

A. x = 120;

B. x = 9123456789;

C. y = 200;

D. x = 120; y = 200;

E. x = y;

Exercise：Variables

- Example: Area.java

```
/* 計算圓的面積 */
1.   public class Area
2.   {
3.       public static void main (String args [])
4.      {
5.          final double PI = 3.1416;  //pi，近似值
6.          double r, area;
7.          r = 10.8;                          //圓的半徑
8.          area = PI * r * r;                 //計算面積
9.          System.out.println("Area of circle is: " + area);
10.     }
11. }
```

Exercise：Variables

- Int 型別可以表現的最大值為2147483647，請寫一程式，將int型別的最大值設定給變數a，再將a+1設定給變數b，然後將b的值印出來。

Exercise：Variables

- 請建立Java程式將下列的八和十六進位值轉換成十進位顯示：

  0277、0xcc、0xab、0333、0555、0xff

Q & A