變數的命名規則

1. 變數名必須是一個合格的識別字➔開頭的第一個字必須是 英文字母、$、_、中文字
2. 變數名不能是關鍵字或保留字
3. 變數名稱區分大小寫 Tom 與 TOM 不同
4. 變數被定義的那個區段內，在定義變數的那個敘述句後才是變數的有效範圍的開始

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

變數的宣告方式

```
public static void 變數的宣告() {
    int x;
    int a, b, c;

    int m = 1;
    int p = 1, q = 2, r = 3;
}
```

```
public static void 變數命名規則() {
    int a123;
    int $123;
    int _abc;
    int 中文;

//      int 123;   //不可全部都是數字
//      int ~abc;  //不可用 ~
//      int #abc; //不可用 #
//      int abstract ; //用到關鍵字
    }
```

```
public static void 兩數交換() {
    int var1 = 100, var2 = 200;
    int tmp;
    System.out.println("交換前 var1 = " + var1 + "  var2 = " + var2);

    tmp = var1;
    var1 = var2;
    var2 = tmp;

    System.out.println("交換後 var1 = " + var1 + "  var2 = " + var2);
}
```

```
交換前 var1 = 100  var2 = 200
交換後 var1 = 200  var2 = 100
```

變數分類一➔型態提供者

1. 系統提供的型別

<1>. 基本型別➔byte · short · int · long · char · float · double · boolean ➔基本型態變數

<2>. 類別 (class)➔ 抽象單位 ➔物件變數

(1). 包裝類別➔Byte · Short · Integer · Long · Character · Float · Double · Boolean

(2). 字串類別➔ String · StringBuffer · StringBuilder

(3). 其他類別➔Scanner · System .....

<3>. 列舉(enum)➔ 抽象單位➔一組常數值➔列舉值

2. 使用自訂的型別

<1>. 其他類別 (class)

<2>. 列舉型別 (enum)

基本型別➔數值 · 字元 · 布林

型別	位元數	修飾字	範圍
byte	8 bits		-128 至+127
short	16 bits		-32 · 768 至+32 · 767
int	32 bits		-2 · 147 · 483 · 648 至+2 · 147 · 483 · 647
long	64 bits	L · l	-9 · 223 · 372 · 036 · 854 · 775 · 808 至+9 · 223 · 372 · 036 · 854 · 775 · 807
char	16 bits		無號的短整數 0~65535
float	32 bits	F · f	
double	64 bits	D · d	
boolean			true 或 false 列舉型別

```
public static void 基本型別() {
    //數值
    byte a1 = 127;
    short a2 = 32765;
    int a3 = 2147483647;

    long b1 = 2147483647;
    //long b2 = 2147483648; //已超出整數範圍 4byte
    long b3 = 2147483649L; //會真正得到 8 byte

    float c1 = 4.5F;
    float c4 = 5; //沒有小數,可以不加F
    double c2 = 5.6d;
    double c3 = 5.6;
    //字元
    char d1 = 'A';
    char d2 = 65535; // char 可以當 無號的短整數
    char d3 = 65;
    System.out.println("d3= " + d3);  //A
    //布林
    boolean e1 = true; //列舉型別
}
```

物件型別➔八個包裝類別

基本型別	包裝類別	
byte	Byte	1. 字串轉數字
short	Short	2. 數字轉字串
int	Integer	3. 數字轉包裝物件
long	Long	4. 包裝物件轉數字
float	Float	5. 字元轉大寫
double	Double	6. 字元轉小寫
char	Character	
boolean	Boolean	

```
public static void 八個包裝類別() {

    int a1 = Integer.parseInt("123"); //123
    double a2 = Double.parseDouble("123.456"); //123.456

    String b1 = Integer.toString(123); //"123"
    String b2 = Double.toString(123.456); //"123.456"

    char c1 = Character.toUpperCase('a'); //'A'
    char c2 = Character.toLowerCase('A'); //'a'
}
```

物件型別➔字串類別

型別	
String	不適合做字串運算
StringBuffer	適合做字串運算➔多工環境
StringBuilder	適合做字串運算➔單工環境➔速度快

```
public static void 字串類別() {
    String str1 = new String("abc");
    //或
    String str2 = "abc";
    str2.concat("xyz");
    System.out.println("str2=" + str2); //abc

    str2 = str2.concat("xyz");
    System.out.println("str2=" + str2); //abcxyz

    StringBuffer str3 = new StringBuffer("abc");
    str3.append("xyz");
    System.out.println("str3=" + str3); //abcxyz

    StringBuilder str4 = new StringBuilder("abc");
    str4.append("xyz");
    System.out.println("str4=" + str4); //abcxyz
}
```

物件型別➔一般類別(系統提供，自訂)

Scanner(String source)
建構一個新的 Scanner，它產生的值是從指定字元串掃瞄的。

String **next**()
尋找並返回來自此掃瞄器的下一個完整標記。

```
public static void 其他類別_系統提供() {
    Scanner s = new Scanner("ab,cd").useDelimiter(",");
    System.out.println(s.next()); //ab
    System.out.println(s.next()); //cd
}
```

```
public static void 其他類別_使用者自訂() {
    Human1 人1 = new Human1();
    人1.名字 = "賴玉珊";
    人1.身高 = 162;
    人1.體重 = 50;
    System.out.println("人1=" + 人1); //位址
    System.out.println("人1.名字=" + 人1.名字); //值
}
```

```
人1=source.Human1@659e0bfd
人1.名字=賴玉珊
```

```
class Object {

    public native int hashCode();

    public boolean equals(Object obj) {
        return (this == obj);
    }

    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
    //
    //
}
```

```
public class Human1 {

    public String 名字;
    public int 身高;
    public int 體重;

    public void 吃(int x) {
        System.out.println("吃" + x + "碗飯");
    }

    public void 跑(int y) {
        System.out.println("跑" + y + "公里");
    }
}
```

```
public class Human1 {

    public String 名字;
    public int 身高;
    public int 體重;

    public void 吃(int x) {
        System.out.println("吃" + x + "碗飯");
    }

    public void 跑(int y) {
        System.out.println("跑" + y + "公里");
    }

    public String toString(){
        return 名字 + " " + 身高 + " " + 體重;
    }
}
```

列舉型別➔(系統提供，自訂)

```java
public enum ColorX {

    BLUE, ORANGE , RED, YELLOW
}
```

```java
public enum Fruit {

    APPLE,BANANA,ORANGE
}
```

```java
public static void 列舉型別1() {

    int opt1=2; //1=藍色 2=橘色 3=紅色 4=黃色
    接收1(opt1);
}

public static void 接收1(int opt1) {

    if (opt1 == 1) {
        System.out.println("藍色");
    }
    if (opt1 == 2) {
        System.out.println("橘色");
    }
    if (opt1 == 3) {
        System.out.println("紅色");
    }
    if (opt1 == 4) {
        System.out.println("黃色");
    }
}
```

```java
public static void 列舉型別2() {

    ColorX opt1= ColorX.ORANGE;
    接收2(opt1);
}

public static void 接收2(ColorX opt1) {

    if (opt1 == ColorX.BLUE) {
        System.out.println("藍色");
    }
    if (opt1 == ColorX.ORANGE) {
        System.out.println("橘色");
    }
    if (opt1 == ColorX.RED) {
        System.out.println("紅色");
    }
    if (opt1 == ColorX.YELLOW) {
        System.out.println("黃色");
    }
}
```

```java
public static void 列舉型別() {
    boolean x = true;
    System.out.println(x);

    ColorX color = ColorX.ORANGE;
    System.out.println(color);
}
```

```
true
ORANGE
```

變數的分類二➔變數內容能否變化

1. 常數值➔只能放 = 右邊➔123 · "abc" · 'a'
2. 變數➔int x➔小寫
3. 常數的變數➔簡稱常數 ➔final double PI=3.14159

<1>. 在 java 程式中，在定義變數時若使用了 final 修飾字，就是定義 "常數"

<2>. 其特點就是經過初始化後不能再重新設定其值

<3>. 常數的變數名稱習慣用大寫的英文字

```
public static void 變數分類二_內容能否變化() {
    //等號右邊是常數
    int x1 = 4;
    String x2 = "abc";
    char x3 = 'A';
    boolean x4 = true;

    //變數
    int y1 = 4;
    y1 = 5;
    y1 = 6;
    //常數的變數
    final double PI = 3.14159;
    //PI = 3.14;    //錯誤，不可再次設定其值
    final int NUM; // final 區域變數，定義時允許不初始化
    NUM = 18;
    //NUM = 20;
}
```

變數的分類三➔變數的內容

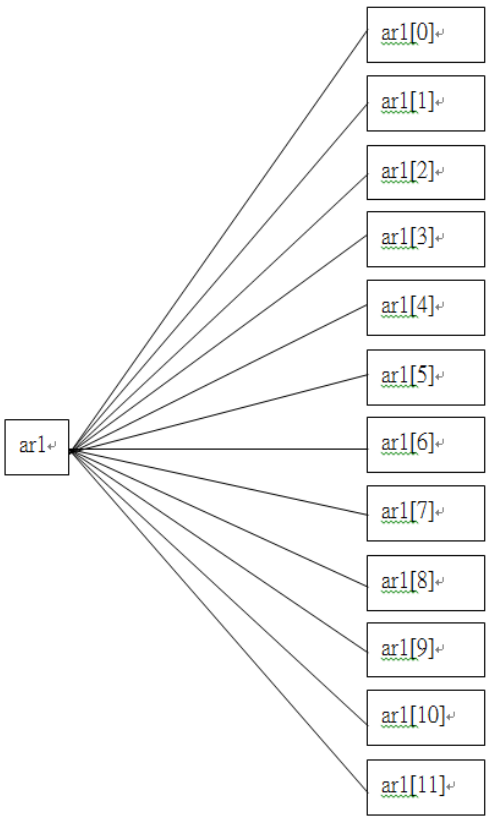
1. 值➔8 個基本型態(數值，字元，布林)，列舉 (常數串)
2. 址➔8 個基本型態[]，參考型別，參考型別[]

```
//基本型態,基本陣列,包裝型別,字串型別,其他
    public static void 變數分類三_值與址() {
        int a1 = 123;
        double a2 = 75.6;
        char a3 = 'A';
        boolean a4 = true;
        System.out.println("a1=" + a1); //值 123
        System.out.println("a2=" + a2); //值 75.6
        System.out.println("a3=" + a3); //值 'A'
        System.out.println("a4=" + a4); //值  true
        ////////////////////////////////////
        int[] ar1 = new int[3];
        ar1[0] = 10;
        ar1[1] = 20;
        ar1[2] = 30;
        System.out.println("ar1=" + ar1); //址  [I@659e0bfd
        ////////////////////////////////////
        Integer b1 = new Integer(123);
        Double b2 = new Double(123.45);
        System.out.println("b1=" + b1); //值 ,特 123
        System.out.println("b2=" + b2); //值 ,特 123.45
        ////////////////////////////////////
        String c1 = new String("abc");
        StringBuffer c2 = new StringBuffer("abc");
        StringBuilder c3 = new StringBuilder("abc");
        System.out.println("c1=" + c1); //值 ,特 "abc"
        System.out.println("c2=" + c2); //值 ,特 "abc"
        System.out.println("c3=" + c3); //值 ,特 "abc"
        ////////////////////////////////////
        Human1 人1 = new Human1();
        人1.名字 = "賴玉珊";
        人1.身高 = 162;
        人1.體重 = 50;
        System.out.println("人1.名字=" + 人1.名字); //值,特 "賴玉珊"
        System.out.println("人1=" + 人1); //值,特 "賴玉珊 162 50"==>toString()覆寫
    }
```

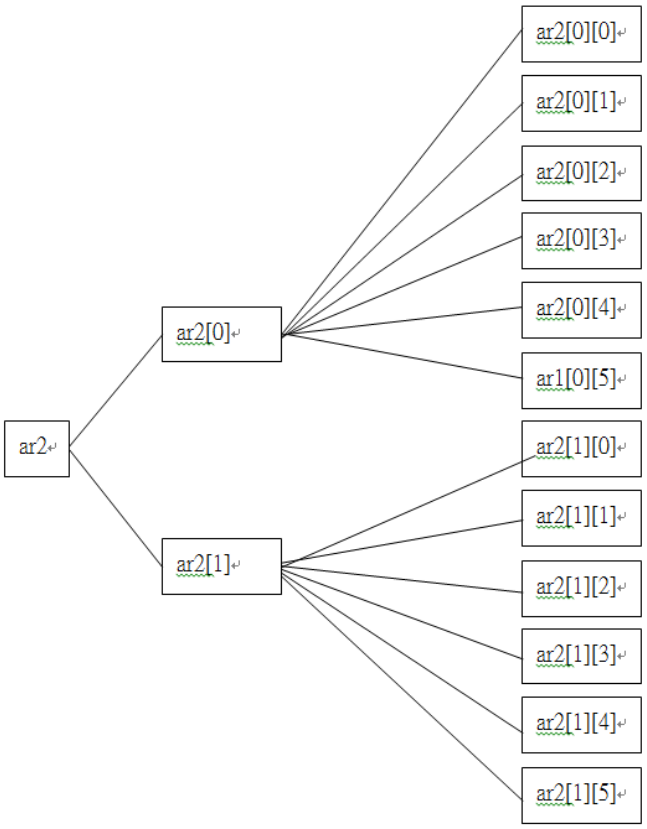
```java
public static void 變數分類三_值與址_注意() {
    Integer[] ar1 = new Integer[3];
    ar1[0] = new Integer(123);
    ar1[1] = new Integer(456);
    ar1[2] = new Integer(789);
    System.out.println("ar1=" + ar1); //址 [Ljava.lang.Integer;@659e0bfd
    ////////////////////////////////////////////////
    String[] ar2 = new String[3];
    ar2[0] = new String("aa");
    ar2[1] = new String("bb");
    ar2[2] = new String("cc");
    System.out.println("ar2=" + ar2); //址 [Ljava.lang.String;@2a139a55
    System.out.println("ar2[0]=" + ar2[0]); //值->特 aa
    ////////////////////////////////////////////////
    Human1[] ar3 = new Human1[2];
    ar3[0] = new Human1();
    ar3[1] = new Human1();
    ar3[0].名字 = "賴玉珊";
    ar3[0].身高 = 162;
    ar3[0].體重 = 50;
    ar3[1].名字 = "張小燕";
    ar3[1].身高 = 150;
    ar3[1].體重 = 60;
    System.out.println("ar3=" + ar3); //址 [Lsource.Human1;@15db9742
    System.out.println("ar3=" + ar3[0]); //值->特 賴玉珊 162 50
    System.out.println("ar3=" + ar3[1]); //值->特 張小燕 150 60
    System.out.println("ar3[0].名字=" + ar3[0].名字); //值->特 賴玉珊
    System.out.println("ar3[0].名字=" + ar3[1].名字); //值->特 張小燕
}
```

陣列結構

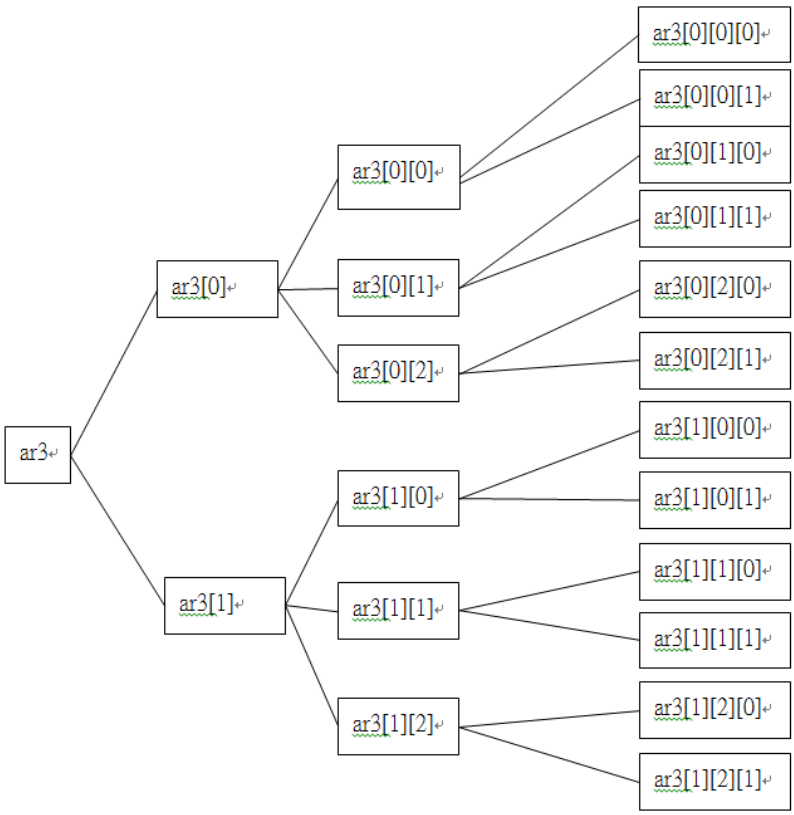
一維陣列➔int[ ] ar1=new int[12]



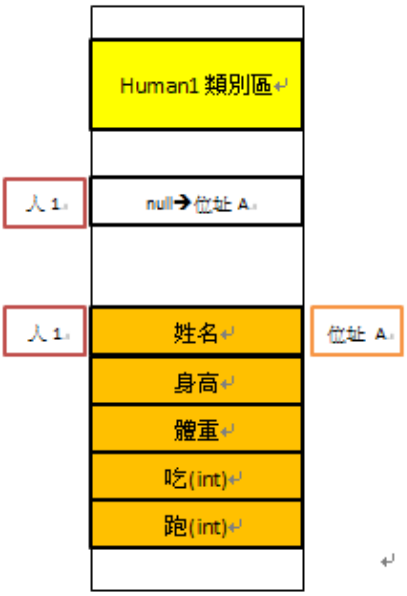
二維陣列➔int[ ][ ] ar2=new int[2][6]



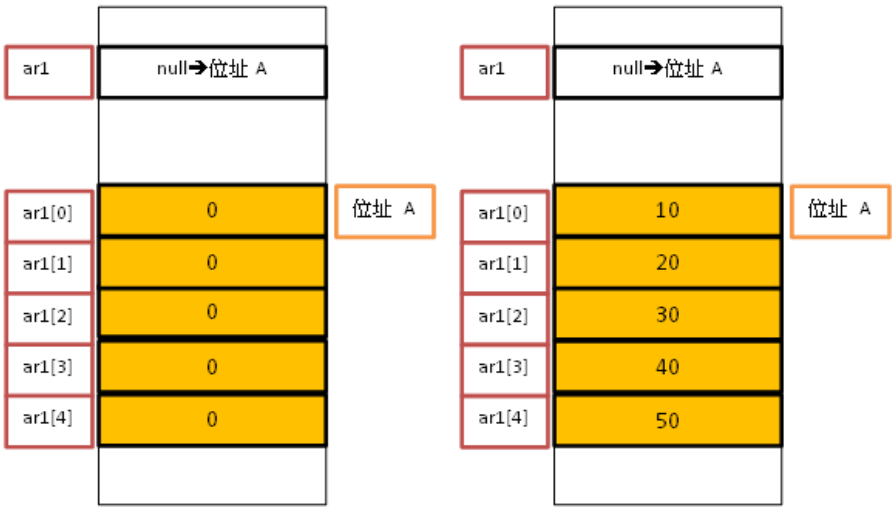
三維陣列➔int[ ][ ][] ar3=new int[2][3][2]



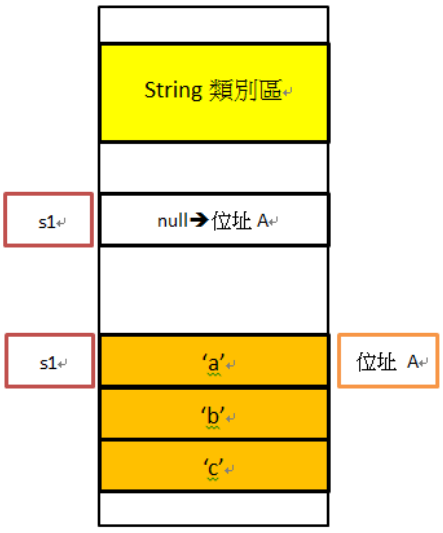
Class➔ Human1 人 1=new Human1()



Array➔ int [] ar1=new int[5] 或 int[] ar1={10，20，30，40，50}



String➔ String s1=new String(“abc”) 或 String s1=” abc”



變數的分類四➔作用範圍　生命週期

1. 全域變數➔可加 public，static ➔整個專案的類別皆可使用，所以必須加上　“封裝”

   <1>.　類別等級➔static

   <2>.　物件等級

2. 區域變數 ➔區域變數名可以跟全域變數名一樣 (會有警告訊號)

3. 區塊變數➔ if，for，switch➔區塊變數名不可以跟區域變數名一樣

注意➔系統會自動初始化的變數 ➔[全域變數]，[全域陣列，區域陣列，區塊陣列]

➔系統不會初始化的變數，不強制 user 初始化 ➔[區域變數，區塊變數]，[區域常數，區塊常數]

➔系統不會初始化的變數，且強制 user 自行初始化，不初始化 會 Compiler 錯誤 ➔[全域常數]➔延遲定義

※ 當一個 變數 尚未儲存任何資料，我們稱為 “未初始化” ，未初始化的變數是不能取得的 ※

例如 System.out.println(x)　或　x=x+1➔ 都需要 取得 x 的值 ➔若 x 的值沒被初始化就不能取用

作用範圍

```
public class VarDemo1 {

    public Human1 p1 = new Human1();
    public int[] a1 = new int[5];
    public String s1 = new String("abc");
    public int x1;
    public double y1;
    public final int z1 = 10;
    // z1=10 ; //全域的常數 , 要馬上設定初始值

    public void s1(int p) {
        Human1 p2 = new Human1();
        int[] a2 = new int[5];
        String s2 = new String("abc");
        int x2;
        double y2;
        final int z2;
        z2 = 10;// 區域的常數可以在下面設定初始值
        if (z2 > 10) {
            Human1 p3 = new Human1();
            int[] a3 = new int[5];
            String s3 = new String("abc");
            int x3;
            double y3;
            final int z3 = 10;
            // System.out.println(x3); //區塊變數 , 沒有設定初始值 , 不能取用

            //不管 陣列宣告在那裡 , 系統都會給初始值
            System.out.println(a1[0]); //全域
            System.out.println(a2[0]); //區域
            System.out.println(a3[0]); //區塊
        }
        System.out.println(x1);
        // System.out.println(x2); //區域變數 , 沒有設定初始值 , 不能取用
        // System.out.println(x3); //區塊變數 , 超出範圍
    }
}
```

```java
public class VarDemo2 {

    public static float var = 99.99F;

    public static void 變數的有效範圍() {

        System.out.println("var = " + var);
        //錯誤，尚未定義出 i 變數
       // System.out.println("i = " + i);

        int i = 1000;
        System.out.println("i = " + i);
    }
}
```

```java
public class VarDemo3 {

    public static String s1 = "gogo";
   //public String s1 = "ok";

    public static void fun1() {
        String s1 = "abc";
        int x = 200;
        //byte x = 120;
        float y = 33.91F;
        System.out.println("s1 = " + s1);
        System.out.println("x = " + x);
        System.out.println("y = " + y);

        fun2();
    }
    public static void fun2() {
        int x = 10;
        char y = 'C';
        System.out.println("in fun1(), x = " + x);
        System.out.println("in fun1(), y = " + y);
        System.out.println("in fun1(), s1 = " + s1);
    }
}
```

生命週期

```java
public class VarDemo4 {

    public static int x ; //回自動初始化成 0

    public static void main(String[] args) {
        //測試全域變數
        a1();
        System.out.println("--->x= " + x);
        a2();
        System.out.println("--->x= " + x);
        //測試區域變數
        a3();
        a3();
        a3();
    }
    public static void a1() {
        x = x + 10; //全域變數
    }
    public static void a2() {
        x = x + 20;
    }
    public static void a3() {
        int y = 0;
        y = y + 1;
        System.out.println("--->y= " + y);
    }
}
```

```java
public class VarDemo5 {

    public int a = 4;                   //全域 , 物件等級 , 基本形態 , 變數
    public final double b = 5.5;        //全域 , 物件等級 , 基本型態 , 常數
    public static double c;             //全域 , 類別等級 , 基本型態 , 變數
    public VarDemo1 d;                  //全域 , 物件等級 , 參考型態 , 變數(類別)
    public static VarDemo1 e;           //全域 , 類別等級 , 參考型態 , 變數(類別)
    public int[] f = new int[5];        //全域 , 物件等級 , 參考型態 , 變數(陣列)
    public String g = "abc";            //全域 , 物件等級 , 參考型態 , 變數(字串)

    public void abc(int p) {             //區域 , 基本型態 , 參數 ( 變數 )
        int total =0  ;                  //區域 , 基本型態 , 變數
        final double k = 4.6;            //區域 , 基本型態 , 常數
        VarDemo1 l = new VarDemo1();     //區域 , 參考型態 , 變數(類別)
        double[] m = new double[10];     //區域 , 參考型態 , 變數(陣列)
        String n = "xyz";                //區域 , 參考型態 , 變數(字串)

        for (int i = 0; i < 10; i++) {   //區塊 , 基本型態 , 變數
            total = total + i;
        }
        if (total > 4.5) {
            VarDemo1 o = new VarDemo1();  //區塊 , 參考型態 , 變數(類別)
            String q = new String("abc"); //區塊 , 參考型態 , 變數(字串)
            final double sum = 0;         //區塊 , 基本型態 , 常數
        }
    }
    public static void xyz() {
        int r = 0;                       //區域 ,  基本形態 , 變數
    }
}
```

二進位表示法與底線(java 7.0 版 才有)

```java
public static void 進位表示法與底線() {
    int a1 = 70;
    int a2 = 070;
    int a3 = 0x70;
    int a4 = 0b0100_0110;
    System.out.println("a1=" + a1); //70
    System.out.println("a2=" + a2); //56
    System.out.println("a3=" + a3); //112
    System.out.println("a4=" + a4); //70

    int b1 = 1234_5678;
    double b2 = 123.456_789;
    System.out.println("b1=" + b1); //12345678
    System.out.println("b2=" + b2); //123.456789

    long c1 = 0b0000101001110011L;
    long c2 = 0b0000_1010_0111_0011L;
    System.out.println("c1=" + c1); //2675
    System.out.println("c2=" + c2); //2675
}
```

```java
public static void 底線_注意() {

    //int x1 = _52;
    //int x2 = 52_;
    //float x3 = 3_.1415F;
    //float x4 3._1415F;
    //long x5 = 999_99_9999_L;
    //long x6 = 999_99_9999L_;
    //int x7 = 0_x52 ;
    //int x8 = 0x_52;
    //int x9 = _0x52;
    int x10 = 5_______2;
    int x11 = 0x5_2;
    int x12 = 0_52;
    int x13 = 05_2;
    int x14 = 5_2;
}
```

1. 數字的前後
2. 點的前後
3. 修飾字的前後
4. 0x、0b 的前後，中間 但 8 進位的 0 例外

都不能放 底線

讓變數之值正確被辨識的修飾符號

1. 當我們利用數字來指定變數之值時，除了它必須是範圍內的值之外，有些值必須樣加上特定的修飾符號，編譯器才能正確辨識它，否則編譯時會有錯誤發生
2. 可加修飾符號 的型態有 long、float、double
3. 當 long 變數之值若超過 int 的範圍 例如大於 2147483647 時若不加 "L" 編譯器就會顯示 integer number too large
4. double 型態可以不加，但 float 一定要加
5. 修飾符號只是用來跟編譯器溝通的符號，並非值的一部份

```java
public static void 修飾字() {
    long a1 = 2000;
    long a2 = 2147483647;
    long a3 = 2147483648L;   // 超出整數範圍 才需 加上 L

    float b1 = 4.5F; // float 一定要加 F
    float b2 = 5;
    System.out.println(b2);

    double c1 = 5.8; // double 可加 D 也可不加
    double c2 = 6.5D;
}
```

資料型態的轉換➔基本資料型態 與 參考資料型態

1. 基本轉基本➔ long 轉成 int
2. 參考轉參考➔同型態可互轉，不同型態不能互轉除非有(繼承)關係
3. 參考轉基本➔字串 轉成 基本型態➔”2100” (2100) ➔特例
4. 基本轉參考➔基本型態 轉成 字串➔(2100) “2100” ➔特例

注意 ➔ 參考型態與基本型態之間的轉型只限 String，其餘不行

注意 ➔ int 變數型態的值 100 要設定給 short 變數型態的值➔會產生編譯錯誤

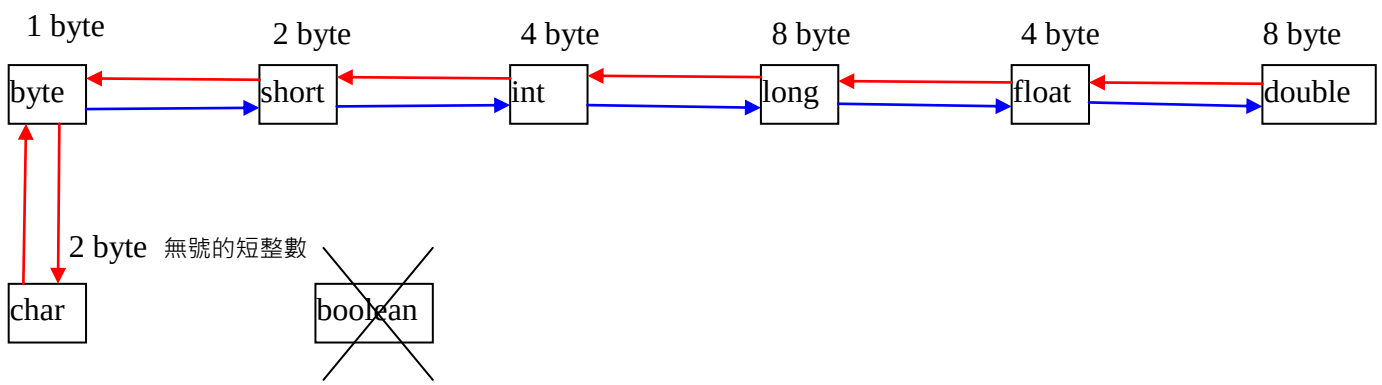
轉型的種類

1. 自然轉型➔小轉大➔不需要經過我們處理，系統自動幫我們做
2. 強迫轉型➔大轉小 ➔ (欲轉成的型態) 值

基本轉基本

int 的 100 0000 0000 0000 0000 0000 0000 0110 0100 short 的 100 0000 0000 0110 0100	0010 0000 0100 0000 0000 0000 0110 0100 ➔有 1 的地方會被截掉 0000 0000 0110 0100

圖



```java
public static void 基本轉基本_轉型() {
    int a1 = 123;
    long a2 = 2147483648L;
    int a3 = 123;
    long a4 = 2147483648L;
    a2 = a1;        //自然轉型
    a3 = (int) a4;  // 強迫轉型
    System.out.println(a3);
    //--------------------------------
    long b1 = 2147483648L;
    float b2 = 4.5f;
    long b3 = 2147483648L;
    float b4 = 4.5f;
    b2 = b1;         //自然轉型
    b3 = (long) b4;//強迫轉型
    System.out.println(b3);// 得到 4 , 小數會被捨掉
    //--------------------------------
    byte c1 = 4;
    char c2 = 'A';
    byte c3 = 4;
    char c4 = 'A';
    c1 = (byte) c2;  //byte 與 char 要 互相轉型
    c4 = (char) c3;  //byte 與 char 要 互相轉型
    //--------------------------------
    byte d1 = 127;   //特權
    short d2 = 5;  //特權
    char d3 = 6;   //特權
    //short d4 = 123L; //沒特權
    //short d5 = d1 + d2; //沒特權 ==>只要是整數全都當成 int==>是班長
    //--------------------------------
    int e1;
    e1 = (int) (4 + 5.5F + 6.8D);  //運算時會轉成型態裡面最大的那一個
}
```

```java
public static void 浮點轉整數() {

    double price = 499 * 0.78;
    int pay = (int) price;
    System.out.println("打 78 折後: " + price);
    System.out.println("取整數: " + pay);
}
```

```
打 78 折後: 389.22
取整數: 389
```

參考轉參考➔除非兩個參考型態間有繼承關係，否則不能轉型

```java
public class VarDemo6 {

    public int x;
    public int y;

    public void xyz() {

    }
}
public class VarDemo7 extends VarDemo6 {

    public int a;
    public int b;

    public void abc() {

    }
}
```

```java
public static void 參考轉參考_轉型() {
    VarDemo6 a = new VarDemo6();
    VarDemo6 b = new VarDemo6();
    a = b;
    ////////////
    VarDemo6 c1 = new VarDemo6();  //父
    VarDemo7 d1 = new VarDemo7();  //子
    c1 = d1;   //有繼承關係 小轉大
    VarDemo6 c2 = new VarDemo6();  //父
    VarDemo7 d2 = new VarDemo7();  //子
    d2 = (VarDemo7) c2; //有繼承關係 大轉小

    ////////////////////////////////////////////
    String s1 = "abc";
    String s2 = "xyz";
    StringBuffer s3 = new StringBuffer("abc");
    s1 = s2;
    // s1 = s3;  //不同型態 , 又沒有繼承關係
    ////////////////////////////////////////////////
    int[] ar1 = new int[5];
    int[][] ar2 = new int[2][3];
    // ar1 = ar2;  //一維對二維不行
    // ar2 = ar1;  //一維對二維不行
    ar1 = ar2[0];  //址
    ar1[0] = ar2[0][0];  //值
}
```

轉型_參考轉基本(字串轉 int)➔利用內建包裝類別作轉型-wrapper class

1. Java 的字串是屬於一種 “類別” 像 “2100” 是一個 String
   不能用 “自動” 或 “強制” 轉型來達成
2. 使用一些特殊的內建類別---Integer 或 Double 或 Long 或 Boolean 等所提供的成員函數(已被建成 static)，所以可以直接拿來用
   用法➔類別名.方法，大寫的英文字母，與基本的資料型別很像，這些特殊的類別稱為 wrapper class 放在 java.lang 內

```
public static void 字串轉數字() {

    int a1 = Integer.parseInt("2100");  //String 轉成 int
    double a2 = Double.parseDouble("88.8");  //String 轉成 double

    System.out.println("a1 = " + a1);
    System.out.println("a2 = " + a2);
}
```

轉型_基本轉參考➔使用 String 類別的 valueOf()方法，因為它是一個多載(overload)的函數所以可傳入的型態可以有 int，long....

```
public static void 數字轉字串() {
    String str1, str2, str3, str4;

    str1 = Integer.toString(1000); // int 轉成 String
    str2 = Character.toString('k'); // char 轉成 String
    str3 = Double.toString(199.77); // double 轉成 String
    str4 = Boolean.toString(false); // boolean 轉成 String

    str1 = String.valueOf(1000); // int 轉成 String
    str2 = String.valueOf('k');  // char 轉成 String
    str3 = String.valueOf(199.77); // double 轉成 String
    str4 = String.valueOf(false); // boolean 轉成 String
}
```

static String	**valueOf**(boolean b) 返回 boolean 參數的字元串表示形式。
static String	**valueOf**(char c) 返回 char 參數的字元串表示形式。
static String	**valueOf**(char[] data) 返回 char 陣列參數的字元串表示形式。
static String	**valueOf**(char[] data, int offset, int count) 返回 char 陣列參數的特定子陣列的字元串表示形式。
static String	**valueOf**(double d) 返回 double 參數的字元串表示形式。
static String	**valueOf**(float f) 返回 float 參數的字元串表示形式。
static String	**valueOf**(int i) 返回 int 參數的字元串表示形式。
static String	**valueOf**(long l) 返回 long 參數的字元串表示形式。
static String	**valueOf**(Object obj) 返回 Object 參數的字元串表示形式。

輸入

```java
public static void scanner1() {
    int x;
    Scanner s1 = new Scanner(System.in);
    System.out.print("輸入數字=");
    x = s1.nextInt();
    System.out.println(x);
}
```

```java
public static void scanner2() {
    String x;
    //可以輸入中文
    Scanner s1 = new Scanner(System.in,"big5");
    System.out.print("輸入字串=");
    x = s1.next();
    System.out.println(x);
}
```

```java
public static void 輸入框與訊息框1() {
    String msg;
    msg = JOptionPane.showInputDialog(null, "請寫下留言：");
    JOptionPane.showMessageDialog(null, "您所輸入的留言是" + msg);
}
```


```java
public static void 輸入框與訊息框2() {
    int msg;
    msg = Integer.parseInt(JOptionPane.showInputDialog(null, "請寫下數字："));
    JOptionPane.showMessageDialog(null, "您所輸入的數字是" + msg);
}
```

視窗程式