Java 陣列

一維陣列與二維陣列

1.陣列簡介

基本定義：(簡單說：陣列用於處理多個同種物件或基本型別變數)

1. 陣列(array)是一種「容器物件(container object)」，可以裝載「多個」且「單一型態」的「基本型別/參考型別」
2. 陣列裡的內容物，稱為成員(element)
3. 建立陣列時，必須指定「長度」，亦即 「成員數量」；一旦建立，長度就不能改變
4. 陣列的成員，使用數字化「索引 (index)」存取；第一個成員的index 為 0

- 陣列用於處理多個同種物件或基本型別變數

假設 :

班上有五位同學，可以宣告5個 int 變數記錄各自年齡:

ex :

int age1 = 30;

int age2 = 31;

int age3 = 54;

int age4 = 68;

int age5 = 20;

但如果有有200個同學? 甚至更多該如何處理? 太多變數將造成程式不易閱讀，這時候就是使用陣列的最佳時機!

2. 認識並建立一維陣列

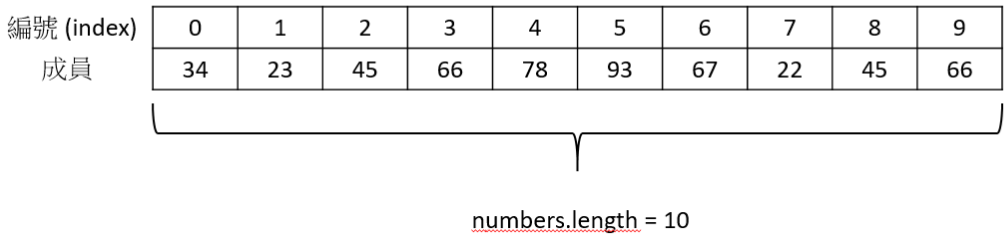
Java 提供陣列將「相同型態」的「物件或基本型別」的變數集中一起管理。

ex :

Array of int types :

12 34 56 78 90 57 32 79 60

- 陣列建立時要指定長度(length)，一旦建立後就不能改變。
- 建立完成後，使用 index 存取陣列成員



A . 第一部分 : 宣告 (Declaring)

語法 :

type [ ] array_identifier ;

// type：陣列的成員型別

// [ ] : 表示宣告陣列

// array_identifier : 陣列名稱

ex 1:

宣告成員為基本型別的陣列 :

char [ ] chars ;

int [ ] ints ;

ex 2:

宣告成員為參考型別的陣列 :

Shirt [ ] shirts;

String [ ] strings;

B. 第二部分 : 建構實例 (Instantiating)

語法 :

array_identifier = new type [length] ;

// array_identifier : 陣列名稱

// type : 陣列的成員型別

// length : 陣列長度

ex :

chars = new char [20] ;

ints = new int [5] ;

strings = new String [7] ;

shirts = new Shirt [3] ;

- Java 物件建構時，若屬於物件成員的實例變數 (Instance variable)未給值，Java將針對不同型別，給予不同預設 (default) 值 :

(1) 整數基本型別 (含 byte 、short、int 、long) : 0

(2) 浮點數基本型別 (含 float 、 double) : 0.0

(3) 字元基本型別 (為 char) : 空字元。用 ' ' 顯示，或 '\u0000'

(4) 邏輯基本型別 (為 boolean) : false

(5) 參考型別 : null

C . 第三部分 : 初始化 (Initializing)

語法 :

array_identifier [index] = value ;

// array_identifier : 陣列名稱

// index : 成員位置，由 0 開始，最大為「長度 -1 」

ex :

int [0] = 13 ;

int [1] = 23 ;

int [2] = 33 ;

int [3] = 43 ;

string [0] = "Hi 0" ;

string [1] = "Hi 1" ;

- 也可以將宣告、實例化、初始化，一起完成。但前提是程式碼不能分行
- 語法 :

  type [ ] array_identifier = { 成員以 "," 區隔 } ;

  ex :

  int [ ] ints = {13 , 23 , 33 , 43} ;

  string [ ] strings ={"Hi 0","Hi 1"} ;

  下面的程式碼是不行的

  int [ ] ints ;

  ints = {13,23,33,43} ;

3. 認識並建立二維陣列

A . 第一部分 : 宣告二維陣列

語法 :

type [ ] [ ] array_identifier ;

// array_identifier : 陣列名稱

// type : 陣列的成員型別

ex :

int [ ] [ ] rowColumns ;

B. 第二部分 : 實例化二維陣列

語法 :

array_identifier = new type [number_of_arrays] [length] ;

// array_identifier : 陣列名稱

// type : 陣列的成員型別

// number_of_arrays : 內含的一維成員陣列個數，不可維空

// length : 每個成員陣列的成員個數，可以為空

ex :

rowColumns = new int [3] [2] ;

C . 第三部分 : 初始化二維陣列

語法 :

array_identifier [index_1] [index_2] = value ;

//array_identifier : 陣列名稱

// index_1 : 指定成員陣列

// index_2 : 指定成員陣列裡的成員位置

ex :

rowColumns [0] [0] = 10;

rowColumns [1] [1] = 20 ;

rowColumns [2] [2] = 30 ;

- 也可以將宣告、實例化、初始化，一起完成
- 語法 :

  type [ ] [ ] array_identifier = { { }, { } ,....., {成員以 "," 區隔 } } ;

  ex :

  int [ ] [ ] int2d = { {13 , 23 } , { 33 } , { 4, 3 } } ;

4. 比較多維陣列的建立

範例 :

```
public class ArrayCreateTest {

    public static void main(String[] args) {
         int [] a1 = new int [5];          //宣告陣列時，[]可以在「型別」後方
         int a2 [] = new int [5];         // 宣告陣列時，[]也可以在「變數」後方
         //下面此行程式碼是錯的
```

```
        int a3 [5] = new int [];        // 宣告陣列時，"="左側的[]裡面不能加上長度，只能在右側的[]內
        int [][] a4 = new int [5][];      //宣告二維陣列時，第一層陣列必須有長度宣告
        int [] a5[] = new int [5][3];    //宣告二維陣列時，"="左右兩側的[]個數必須各自加總後為2。第二層的陣列長度非必要
        int [][][] a6 = new int [5][][]; //宣告三維陣列時，"="左右兩側的[]個數必須各自加總後為3
        int a7 [][][] = new int [5][3][2]; //用於宣告的[]可在變數前後，第一層陣列長度必填，其他層的陣列長度非必要

    }

}
```

- 關於「一維陣列的長度為必要」:

  (1) 陣列是容器物件，建立陣列就好像蓋房子供人居住

  (2) 建構房子的時候，幾個房間一定要事先確認，因為這和建物的主體結構有關，蓋好了就不能再改變

  (3) 每個房間有一個房客，房間編號由 0 開始，拜訪房客必須指定房間編號

  (4) 若是二維陣列，就像房客在自己房間內又隔了幾個房間再出租。不管幾維陣列，只有在蓋房子之初必須確定房間個數，因為關乎房子結構，所以第一層陣列的長度為必要

  ex :

  int [ ] [ ] ints = { { 0,1}, { 2 }, { 3, 4, 5 } } ;

  也可以解釋為何二維陣列的第二層陣列長度並未強制要求 :

  int [ ] [ ] ints = new int [3] [ ] ;

存取陣列內容

1. 陣列成員的基本存取

a. 若要將資料寫入陣列 :

chars [0] = 'Hi' ;

int [2] = 34 ;

strings [3] = "Hello" ;

b. 若要將資料讀出陣列 :

char c = chars [0] ;

int I = ints [2] ;

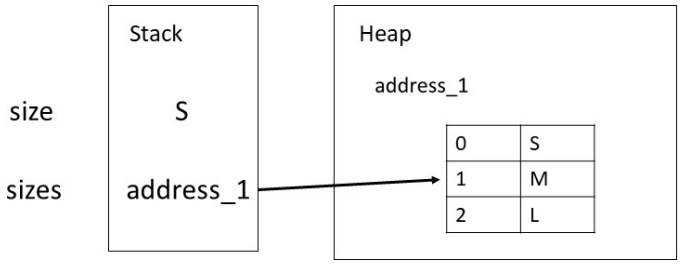
String s = strings [3] ;

2. 陣列成員為基本型別的記憶體配置

陣列成員為基本型別 :

char size = 'S' ;

char [ ] sizes = { 'S', 'M', 'L' } ;

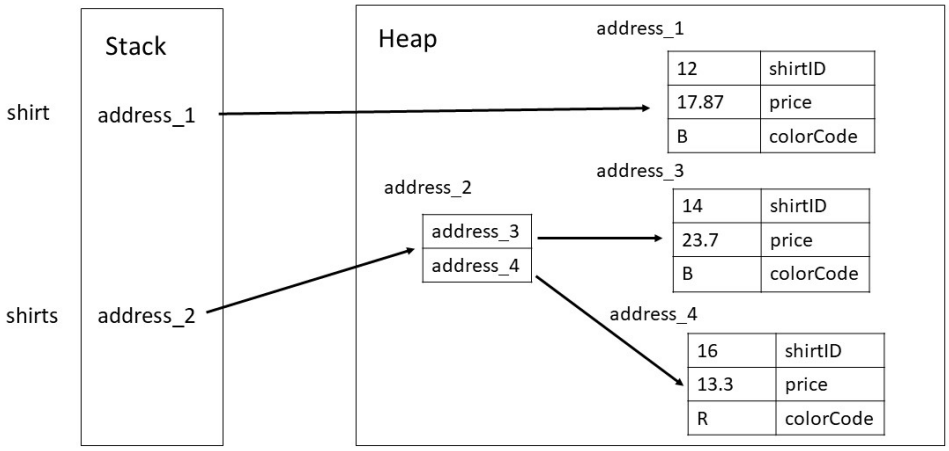


3. 陣列成員為參考型別的記憶體配置

- 參考型別的變數是指向 Heap 記憶體裡真實物件的遙控器，只有透過遙控器才能控制物件
- 陣列為參考型別時 :

  Shirt shirt = new Shirt () ;

  Shirt [ ] shirts = { new Shirt( ) , new Shirt( ) } ;



範例 :

```
public class ReferenceTypeArrayTest {
```

```
    public static void main(String[] args) {
        Shirts s1 = new Shirt();
        Shirt s2 = new Shirt();
        Shirt shirts [] = {s1,s2};

        //s1 和 shirts[0] 指向同一實例
        shirts[0].price = 100 ;
        System.out.println(s1.price);
        System.out.println(s1 == shirts[0]);

        //s2 和 shirts[1]指向同一實例
        s2.price = 200;
        System.out.println(s2.price);
        System.out.println(s2 == shirts[1]);

    }
}

/* Output :
 * 100.0
 * true
 * 200.0
 * true
 */
```

使用指令列的 args 陣列參數

一開始在學習 Java 時就已經接觸並使用陣列。

語法 :

public static void main (String [ ] args) {

    method_code_block

}

範例 1 :

```
public class CommandArgs {
    public static void main(String[] args) {
        System.out.println("args[0] is " + args[0]);
          System.out.println("args[1] is " + args[1]);

    }

}

/* Output :
 * args[0] is Hi
 * args[1] is Rebecca
*/
```

範例 2：

```
public class CommandArgsTest {
    public static void main(String[] args) {
        System.out.println("String is: " + (args[0] + args[1])); // 因為字串陣列取出的字串成員，相加等於字串相連
        int i1 = Integer.parseInt(args[0]);  // 使用Integer類別的parseInt()方法，將String轉換成int
        int i2 = Integer.parseInt(args[1]);
        System.out.println("Summary is : " + (i1+i2)); // 字串轉換成int後，相加即為一般整數加法

    }

}

/* Output :
 * String is: 100200
 * Summary is : 300
*/
```

使用ArrayList類別

1. 陣列的缺點

陣列無法自動增加長度，若有需要，必須自己：

(1) 記錄每個加入陣列的元素的索引

(2) 追蹤並記錄陣列長度

(3) 若長度不足，則建立一個足夠長度的新陣列，並將原陣列成員逐一複製過去，再捨棄原陣列

2. ArrayList 類別簡介

陣列並非唯一可以儲存資料的容器物件。陣列受限於先天的限制，讓 ArrayList類別也是選項之一：

(1) ArrayList類別只存放參考型別的物件，不接受基本型別；但可以改用基本型別的包覆類別

(2) ArrayList類別有許多方法可以管理成員物件: add( )、get( )、remove( )、indexOf( )

(3) 在建構ArrayList物件時不需要設定長度大小，當需要加入更多成員物件時，ArrayList將自動成長

(4) 可以在建構ArrayList物件時設定「initial capacity」，但非必要

- 「initial capacity」的概念

  A. StringBuilder 和 ArrayList 類別在建構時都可以指定「initial capacity」，但非必要

  B. StringBuilder 類別可以隨意增加字串內容，ArrayList類別可以隨意增加物件成員；看似相當方便，但其實這兩個類別用來裝載物件的容器物件，還是「陣列」！因為陣列的規定是一旦建立長度即不能改變，所以StringBuilder和ArrayList類別作法是：

  (a) 是先建立預設長度的陣列

(b) 若持續加入字串或物件導致超出內建陣列長度範圍，則 Java將自動建立更長的新陣列，並將原陣列成員都複製一份移轉過去

後，刪除舊陣列

C. 陣列在新舊交替時將大量使用CPU和記憶體資源，因此若事先可以預知長度，Java 建議在一開始建構 StringBuilder 和 ArrayList 物件時就指定適當的初始長度，亦即「initial capacity」

3. ArrayList使用範例

```
import java.util.ArrayList;
public class ArrayList {

    public static void main(String[] args) {
        ArrayList myList;
        myList = new ArrayList ();
        myList.add("S1");  // ArrayList 內依序加入"S1"、"S2"、"S3"、"S4"等四個成員
        myList.add("S2");  // ArrayList 內依序加入"S1"、"S2"、"S3"、"S4"等四個成員
        myList.add("S3");  // ArrayList 內依序加入"S1"、"S2"、"S3"、"S4"等四個成員
        myList.add("S4");  // ArrayList 內依序加入"S1"、"S2"、"S3"、"S4"等四個成員
        myList.remove(0);  // 指定位置移除第一個，也就是"S1"
        myList.remove(myList.size()-1); //指定位置移除最後一個，也就是"S4"
        myList.remove("S3");       // 指定名稱，直接移除"S3"
        System.out.println(myList);
    }

}

/* Output :
 * [S2]
*/
```

4. 使用泛型指定ArrayList的成員型態

ArrayList類別可以放入基本型別之外的任何物件。使用「泛型 (generic)」則可以在宣告ArrayList的一開始，就限定成員型態

- 「泛型」的符號為「<>」，在 Java 5 的時候導入。

  範例：

  ArrayList list = new ArrayList ( ) ;

- Java 7 後，認為不需要前後兩個「<>」都載明成員型別，因此移除後方「<>」內的型別

  ArrayList list = new ArrayList <> ( ) ;

- ArrayList 使用泛型後，若試圖放入非指定型態的物件，將無法通過編譯