



J01-08

建立並使用 Array (陣列)

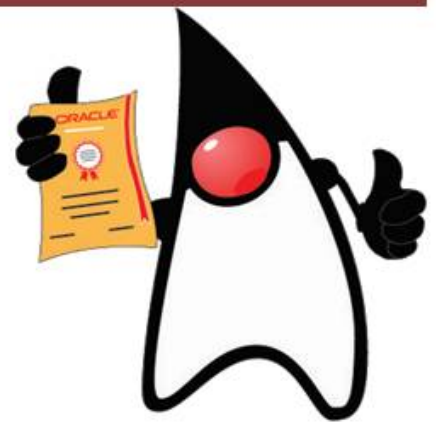
曾瑞君 (Jim_Tzeng)

學習目標

- 陣列 (one-dimensional & two-dimensional)
 - Declare (宣告)
 - Instantiate (建構實例)
 - initialize (初始)
- 存取陣列內容
- 使用 command-line 的 args 陣列
- 使用 ArrayList




Edited by Ruei-Jiun Tzeng



1/4

陣列 (one-dimensional & two-dimensional)

- declare (宣告)
- instantiate(建構實例)
- initialize (初始)



陣列 簡介

- 陣列是一種可以持有多個單一型態的物件或基本型別的「Container Object (容器物件)」
- 建立陣列時，必須指定長度；一旦建立，長度就不能改變。
- 陣列裡的內容物，稱為 element(成員)。
- 陣列的 element，使用數字化的「index」存取；第一個 element 的 index 為 0。



One-Dimensional Arrays

- 若班上有5位同學，或許可以宣告5個 int 變數記錄各自年齡：

```
int age1 = 30;  
int age2 = 31;  
int age3 = 30;  
int age4 = 31;  
int age5 = 30;
```

- 但若有500個？甚至更多個時該如何處理？

Creating One-Dimensional Arrays

- Java 提供 Array (陣列)，可以將相同型態的物件或是基本型別集中一起管理，如：

Array of **Shirt** types:



Array of **int** types:

12 34 56 78 90 57 32 79 60

Array Indices and Length

- 陣列建立時要指定 length，建立後不能改變。
- 使用 index 存取陣列成員。
- 陣列：`numbers`

編號
(index)->

成員->

0	1	2	3	4	5	6	7	8	9
34	23	45	66	78	93	67	22	45	66

`numbers.length = 10`

Declaring (宣告) a One-Dimensional Array

- 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

- 宣告成員為基本型別的陣列：

```
char [ ] chars;
```

```
int [ ] ints;
```

- 宣告成員為參考型別的陣列：

```
Shirt [ ] shirts;
```

```
String [ ] strings;
```


Instantiating (建構實例) a One-Dimensional Array

- 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

- 範例：

```
status = new char [20];    // 成員初始值為 \u0000 (空字元)
ages = new int [5];        // 成員初始值為 0
names = new String [7];   // 成員初始值為 null
shirts = new Shirt [3];    // 成員初始值為 null
```

Initializing (初始化) a One-Dimensional Array

- 語法：

```
array_identifier[index] = value;
```

array_identifier：陣列名稱

index：成員位置，由0開始

- 範例：

```
ints[0] = 13;  
ints[1] = 23;  
ints[2] = 33;  
ints[3] = 43;
```

```
strings[0] = "Hi 0";  
strings[1] = "Hi 1";  
strings[2] = "Hi 2";  
strings[3] = "Hi 3";
```

Declaring, Instantiating, and Initializing 一次完成

- 語法：

```
type [ ] array_identifier = { 成員以 “,” 區隔 };
```

- 範例：

```
int [ ] ints = {13, 23, 33, 43};  
String [ ] strings = {"Hi 0", "Hi 1", "Hi 2", "Hi 3", };
```

- 錯誤示範：

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

Describing Two-Dimensional Arrays

- 多維陣列的概念，在於
 - 第一層陣列裡的成員是第二層陣列 (二維)
 - 第二層陣列裡的成員是第三層陣列 (三維)
 - ...
 - 第N-1層陣列裡的成員是第N層陣列 (N維)
- 二維陣列個概念，可以用「表格」來描述。如指定 第?列、第?行的 element：

Declaring a Two-Dimensional Array

- 語法：

```
type [ ][ ] array_identifier;
```

- 範例：

```
int [ ][ ] rowColumns;
```

Instantiating a Two-Dimensional Array

- 語法：

```
array_identifier  
    = new type [number_of_arrays] [length];
```

- 範例：

```
rowColumns = new int [3][2];
```

	Column 0	Column 1
Row 0		
Row 1		
Row 2		



Initializing a Two-Dimensional Array

- 範例：

```
rowColumns [0] [0] = 10;  
rowColumns [1] [1] = 20;  
rowColumns [2] [0] = 30;
```

	Column 0	Column 1
Row 0	10	
Row 1		20
Row 2	30	



```
public class ArrayCreateTest {  
  
    public static void main(String[] args) {  
  
        int[] a1 = new int[5];  
        int a2[] = new int[5];  
        // int a3[5] = new int[];  
  
        int[][] a4 = new int[5][];  
        int[] a5[] = new int[5][3];  
  
        int[][][] a6 = new int[5][][];  
        int a7[][][] = new int[5][3][2];  
    }  
}
```






小秘訣

關於「第一維陣列的長度為必要」：

- 陣列是容器物件，建立陣列就好像蓋房子供人居住。
- 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。
- 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。

- 
-
- 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，多維陣列由此類推。不管是幾維陣列，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，之後的隔間再出租，可以只是木造隔間，不需特別限制每個二房東要區隔的房間數，所以這樣的二維陣列是合法、合理的：

```
1 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：

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



2/4

存取陣列內容



存取陣列成員

- 寫入：

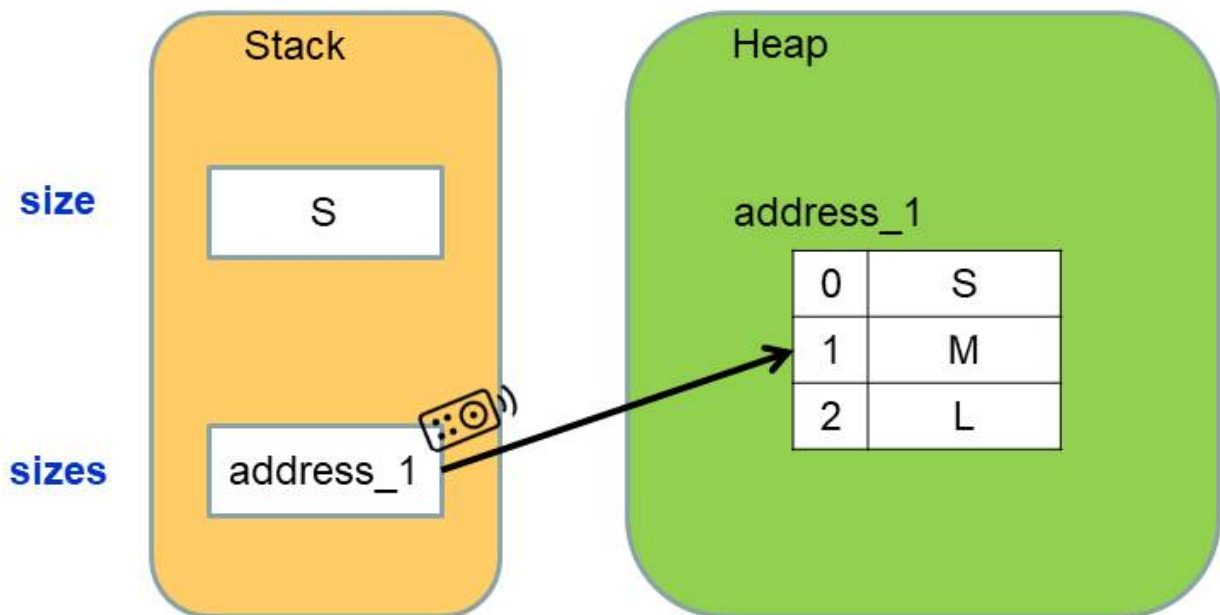
```
chars [0] = 'H';  
ints [2] = 34;  
strings [3] = "Hi";
```

- 讀出：

```
char c = chars [0];  
int l = ints [2];  
String s = strings [3];
```

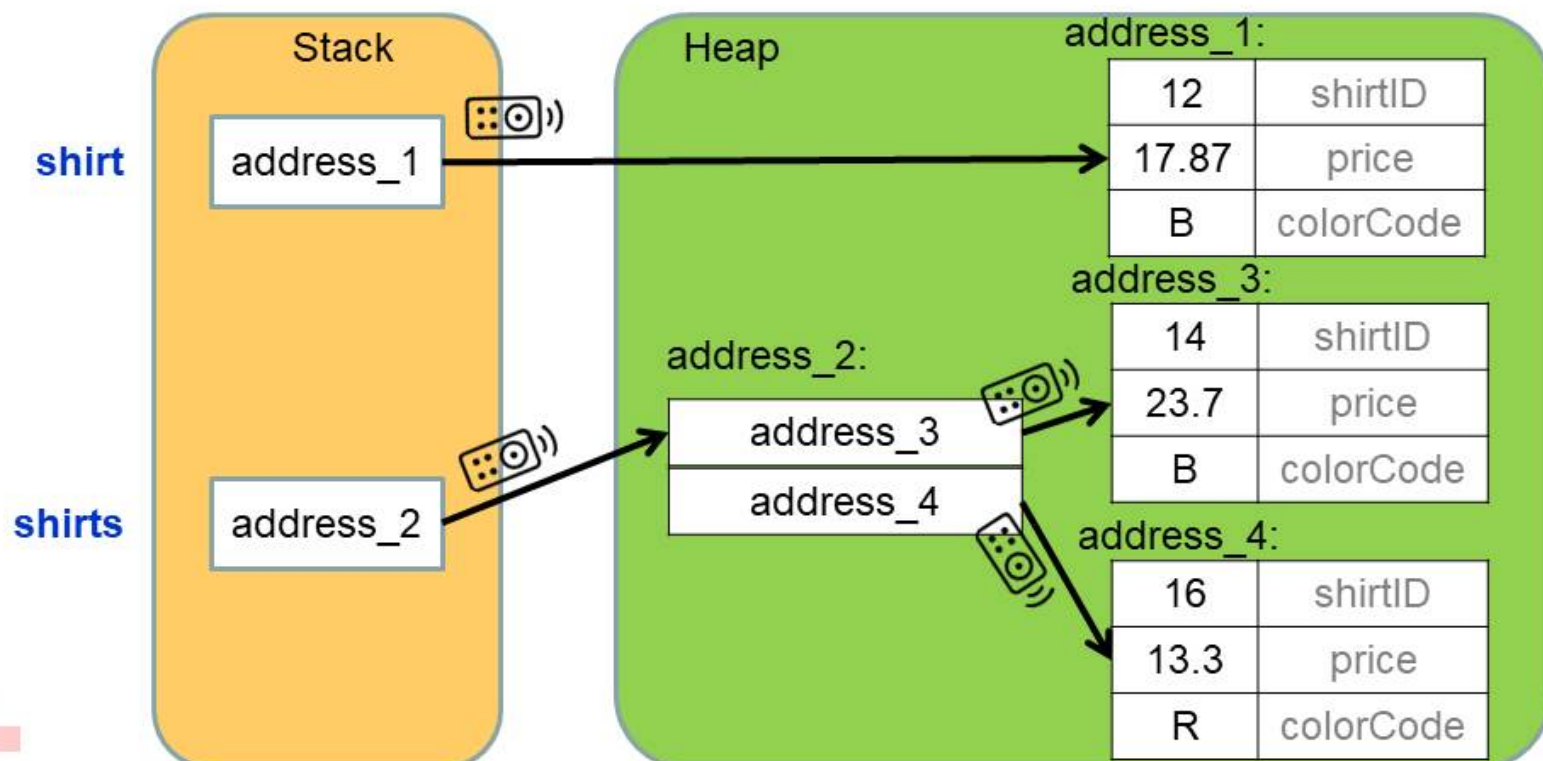
陣列成員為基本型別


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




陣列成員為參考型別

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





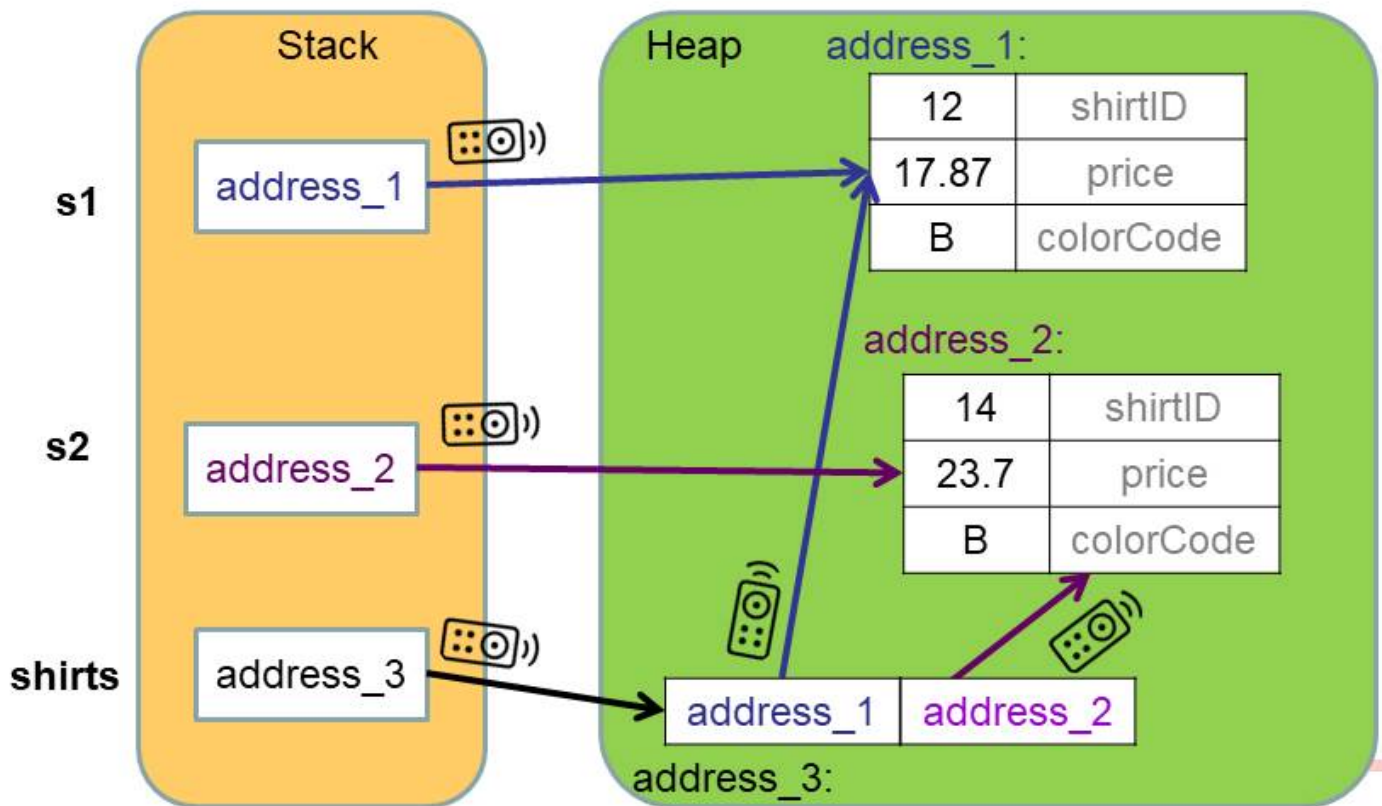
```
public class ReferencedTypeArrayTest {  
    public static void main(String[] args) {  
  
        Shirt s1 = new Shirt();  
        Shirt s2 = new Shirt();  
        Shirt shirts[] = {s1, s2};  
  
        //s1 和 shirts[0] 指向同一實例  
        shirts[0].price = 100;  
        System.out.println(s1.price);  
        System.out.println(s1 == shirts[0]);  
  
        //s2 和 shirts[1] 指向同一實例  
        s2.price = 200;  
        System.out.println(shirts[1].price);  
        System.out.println(s2 == shirts[1]);  
    }  
}
```



```
100.0  
true  
200.0  
true
```

陣列成員為參考型別

```
Shirt s1 = new Shirt();  
Shirt s2 = new Shirt();  
Shirt shirts[] = {s1, s2};
```





3/4

使用 **command-line** 的 **args** 陣列



在 main 方法中使用 args 陣列

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

- 在 command line 中執行：

```
javac Test.java  
java Test Hi Jim!!
```

- 在 command line 後傳入的所有字串，會以空白為區隔，轉成字串陣列傳入main 方法。得到結果為：

```
args[0] is Hi  
args[1] is Jim!!
```



轉換 String 到其他型別

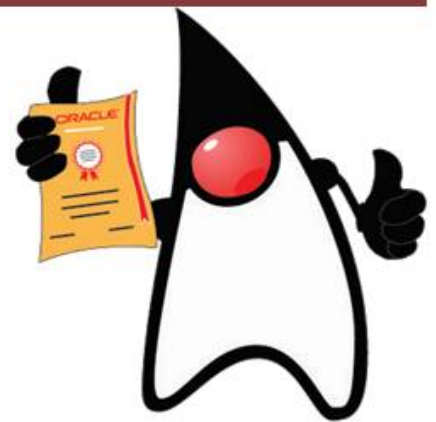
```
public class Test {  
    public static void main (String args[]) {  
        System.out.println("Summary is: " + (args[0] + args[1])); // 字串連接  
        int i1= Integer.parseInt(args[0]);  
        int i2 = Integer.parseInt(args[1]);  
        System.out.println("Summary is: " + (i1 + i2)); // 數字相加  
    }  
}
```

- 在 command line 中執行：

```
javac Test.java  
java Test 4 5
```

- 即便傳入數字，依然以字串陣列傳入 main 方法。必須自己使用“Integer.parseInt()”將String轉換回int：

```
Summary is 45  
Summary is 9
```



4/4

使用 **ArrayList**



陣列的缺點

Array (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 index。
- 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。



The ArrayList Class

- Array (陣列) 並非唯一可以儲存資料的容器物件，ArrayList 類別也是選項之一。
- ArrayList 有許多方法可以管理成員：add()、get()、remove()、indexOf()，及其他。
- 在建構 ArrayList 實例時不需要註明“size”：
 - 當需要加入更多 elements 時，ArrayList 將自動成長
 - 可以註明“initial capacity”，但非必要
- ArrayList 只能存放 參考型別，不允許 基本型別。

Class Names and the Import Statement

- 開發 java 程式時，根據功能屬性會分類到不同 package 中。
- 屬於語言基礎的類別，都放在「`java.lang`」package 中，如 `String`、`Math`、`System`、`Integer` 等。這類類別可以直接使用，無須載明類別出自的 package。
- `ArrayList` 出自 package「`java.util`」。要使用必須二擇一：
 - 1) 使用完整類別名稱 (含 package)：`java.util.ArrayList`
 - 2) 使用 `import`：`import java.util.ArrayList`

```
import java.util.ArrayList;
public class Test {
    public static void main (String args[]) {
        java.util.ArrayList list;
    }
}
```

Working with an ArrayList

```
public static void main(String[] args) {  
  
    ArrayList list;           // 宣告  
  
    list = new ArrayList();    // 建構實例  
  
    list.add("S1");            // 初始化  
    list.add("S2");  
    list.add("S3");  
    list.add("S4");  
  
    list.remove(0);            // 修改  
    list.remove(list.size()-1);  
    list.remove("S3");  
  
    System.out.println(list);  // [S2]  
}
```

使用泛型(Generic)指定成員型態

- ArrayList 可以放入基本型別之外的任何物件。使用泛型，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。語法範例為：

```
ArrayList<String> list = new ArrayList<String>();
```

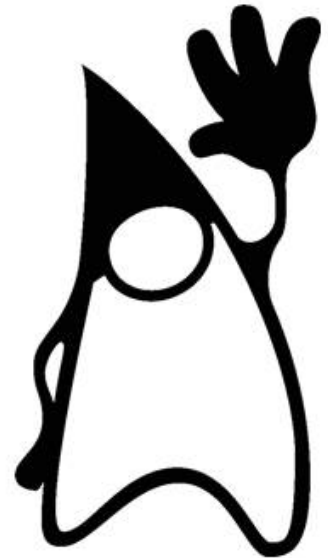
- Java 7之後，認為不需要前後兩個<>都載明成員型別，因此可以使用：

```
ArrayList<String> list = new ArrayList<>();
```



END ~~

Thank you!!



Edited by Ruei-Jiun Tzeng