

Array

陣列

Java Fundamental



Content

- ◆ 一維陣列
- ◆ 二維陣列
- ◆ 多維陣列

Content

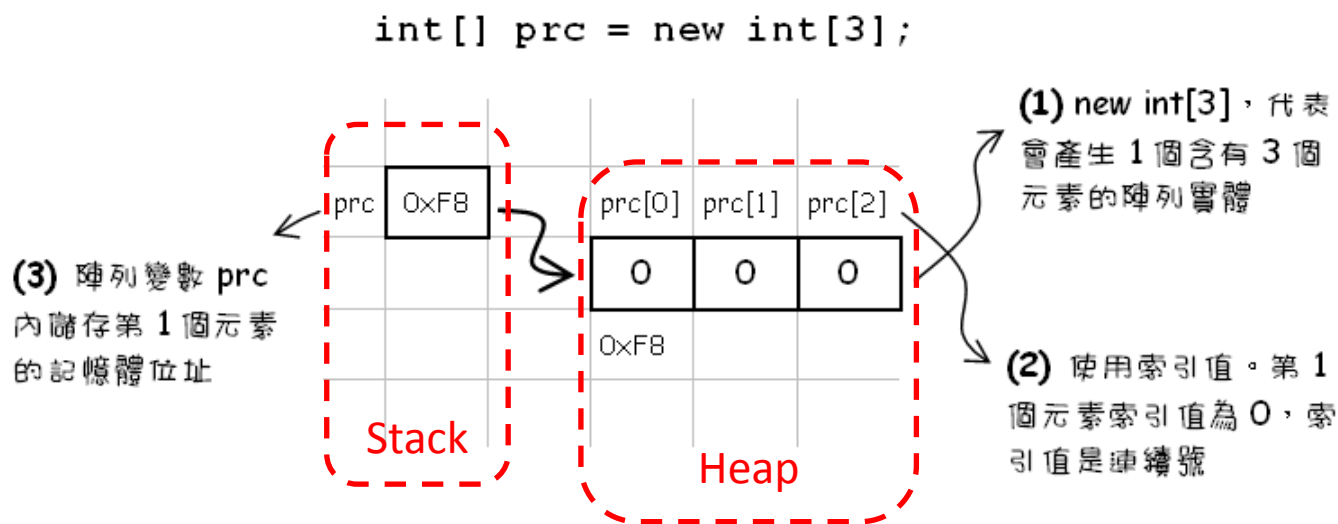
◆ 一維陣列

◆ 二維陣列

◆ 多維陣列

Array

- ◆ 處理的資料量越來越大，單一變數的處理模式已經無法滿足需求，所以引進了陣列的概念。
- ◆ 宣告一個陣列就是向系統要求一個區塊的記憶體空間來儲存一系列以上的資料。



Array

◆ 1維陣列的宣告與實體化(沒有初始化)

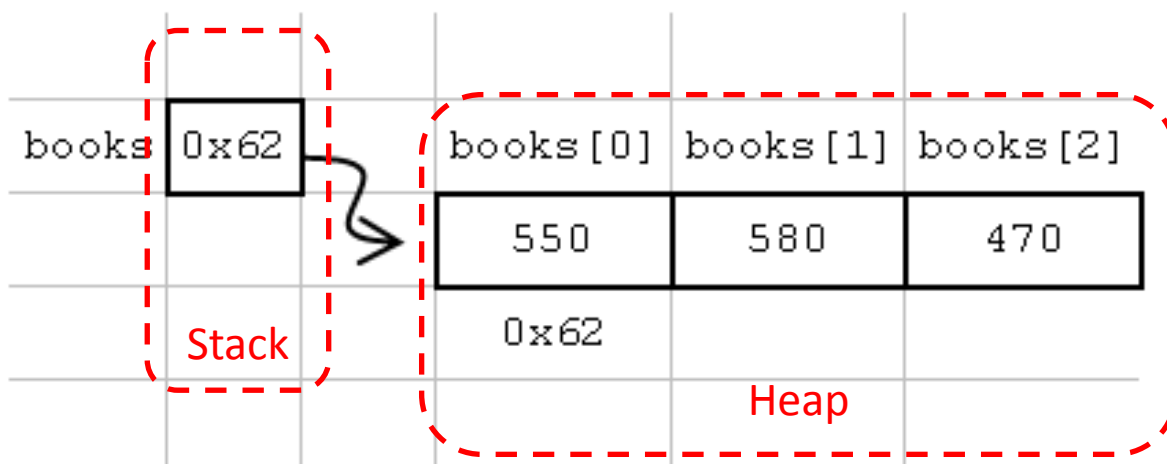
➤ `int[] books = new int[3];`

元素資料類型	初始值
int	0
short	0
byte	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
boolean	false
物件(object)	null

Array

◆ 1維陣列的宣告與實體化(有自訂初始化)

➤ `int[] books = {550, 580, 470};`



一維陣列 – 宣告

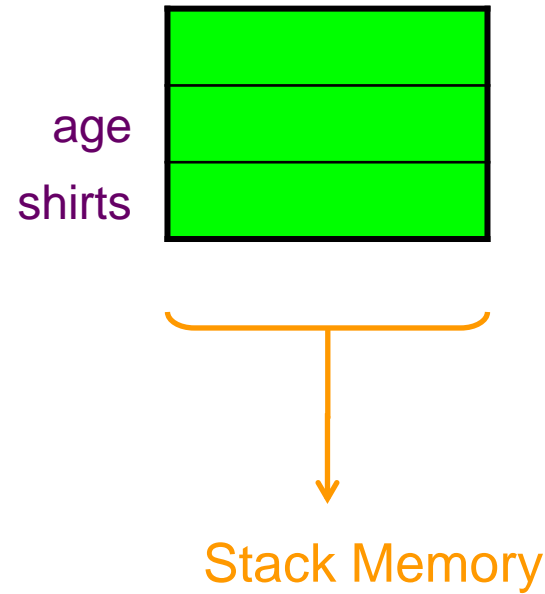
◆ 語法：

```
type [ ] array_name;
```

```
type array_name [ ];
```

Ex: int [] age;

Shirt shirts[];



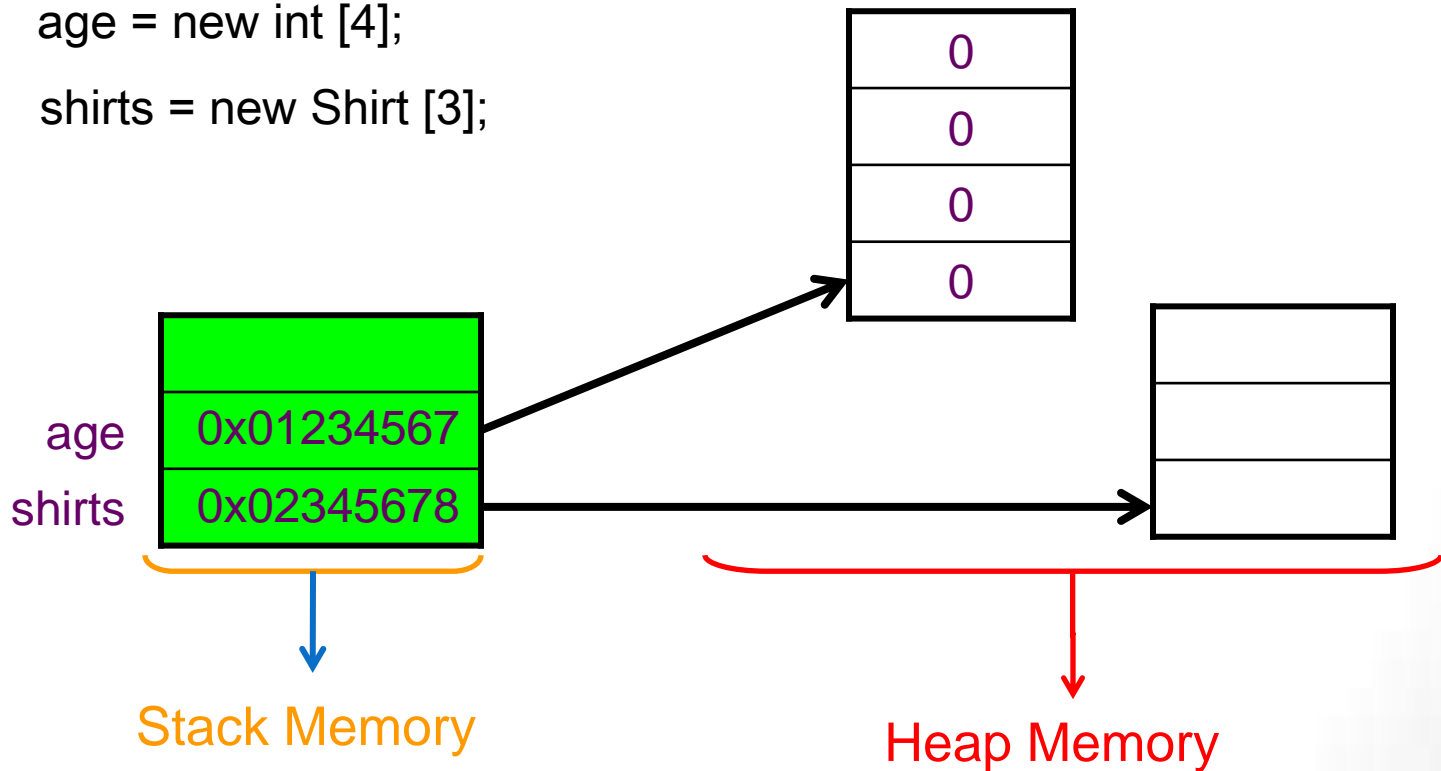
一維陣列 – 實體化

◆ 語法：

```
array_name = new type[length];
```

Ex: age = new int [4];

shirts = new Shirt [3];



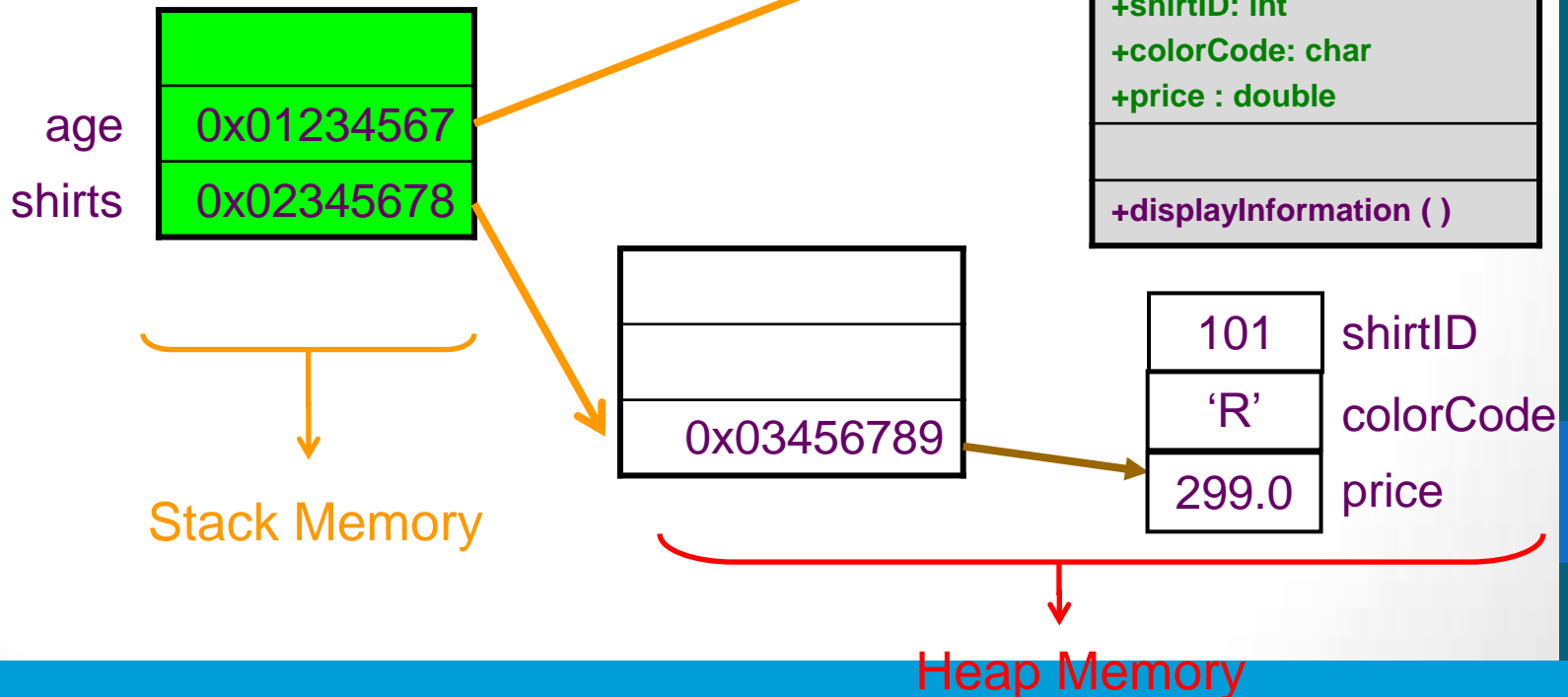
一維陣列 – 初始化

◆ 語法：

`array_name [index] = value;`

Ex: `age [2] = 3;`

`shirts [0] = new Shirt();`



Array

範例 ArrayDeclare.java

```
1. class ArrayDeclare{
2.     public static void main(String[] args){
3.         int[] books = new int[3];
4.         System.out.println("books[0] 預設值 = " + books[0]);
5.         books[0] = 550;
6.         books[1] = 580;
7.         books[2] = 470;
8.         //books[3] = 600; //ArrayIndexOutOfBoundsException
9.         System.out.println(books[0]);
10.        System.out.println(books[1]);
11.        System.out.println(books[2]);
12.    }
13. }
```

一維陣列的建構方式

宣告	<pre>int[] i; Shirt[] s;</pre>	<pre>int[] i = new int[3];</pre>	<pre>int[] i = new int[] {1, 3, 5}; int[] i = {1, 3, 5};</pre>
實體化	<pre>i = new int[3]; s = new Shirt[3];</pre>	<pre>Shirt[] p = new Shirt[3];</pre>	<pre>Shirt[] s = new Shirt[] { new Shirt(), new Shirt(), new Shirt() }</pre>
初始化	<pre>i[0] = 1; i[1] = 3; i[2] = 5; s[0] = new Shirt(); s[1] = new Shirt(); s[2] = new Shirt();</pre>	<pre>i[0] = 1; i[1] = 3; i[2] = 5; s[0] = new Shirt(); s[1] = new Shirt(); s[2] = new Shirt();</pre>	<pre>Shirt[] s = { new Shirt(), new Shirt(), new Shirt() }</pre>

Array

- ◆ 元素值取出是透過索引值，可以利用for迴圈與陣列長度變數來快速取值。

```
1. class Array_Access{
2.     public static void main(String[] args){
3.         int prc[] = new int[3];
4.         int prcLength = prc.length;
5.
6.         prc[0] = 200;
7.         prc[1] = 350;
8.         prc[2] = 250;
9.         //prc[3] = 150;
10.        System.out.println("共有幾本書? " + prcLength);
11.        System.out.print("書的價格為: ");
12.
13.        for(int i=0; i<prcLength; i++){
14.            System.out.print(prc[i] + " ");
15.        }
16. }
```

prc.length 會取得 prc 陣列的元素總數，此時會得到 3

存取索引值如果超過最大索引值會執行失敗，並產生 `ArrayIndexOutOfBoundsException` (索引值超過界限) 的異常情形

i=0 代表索引值從 0 開始；
i < prcLength 代表索引值最大到元素總數-1；
i++ 代表索引值是遞增的連續號

-----輸出-----

共有幾本書? 3

書的價格為: 200 350 250

for-each 迴圈

◆ for-each 迴圈(for-each)

- JavaSE 5.0所新增的語法。
- 簡化存取集合性物件的元素內容。
 - 集合性物件：陣列(array)或集合(collection)
- 執行時自動找下一個元素，直到全部擷取完畢為止
- 由 for-loop 實作出來的
- 讀取集合元素時不可修改元素內容

◆ for-each 語法

for (元素資料型別 區域變數名稱 : 元素集合名稱)

for-each 迴圈

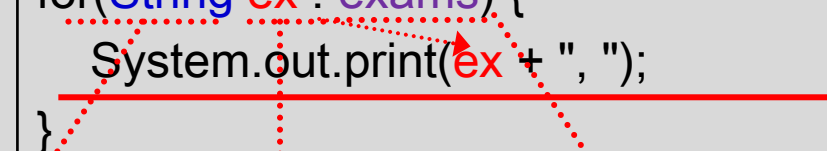
◆ for-loop

```
01 String[] exams = {"SCJP", "SCWCD", "SCMAD"};  
02 for(int i=0;i<exams.length;i++) {  
03     System.out.print(exams[i] + ", ");  
04 }
```

執行結果： **SCJP** , **SCWCD** , **SCMAD** ,

◆ for-each

```
01 String[] exams = {"SCJP", "SCWCD", "SCMAD"};  
02 for(String ex : exams) {  
03     System.out.print(ex + ", ");  
04 }
```



元素資料型別 區域變數名稱 元素集合名稱

執行結果： **SCJP** , **SCWCD** , **SCMAD** ,

Array

- ◆ 使用for~each取出陣列的每個元素值
- ◆ for~each迴圈會自動將陣列內的元素由第1個至最後1個依序傳出，所以只要宣告一個變數儲存陣列元素即可。

Array_Access_ForEach.java

```
1. class Array_Access_ForEach{
2.     public static void main(String[] args){
3.         int prc[] = {200, 350, 250};
4.
5.         System.out.print("書的價格為: ");
6.
7.         for(int price : prc)
8.             System.out.print(price + " ");
9.     }
10. }
```

for~each 是針對每一個元素，宣告 price 變數是為了依序儲存 prc 陣列內的元素值。for~each 迴圈每跑一次，就會將下個元素值存入 price 變數內，所以 price 變數代表的是元素值，並非索引值

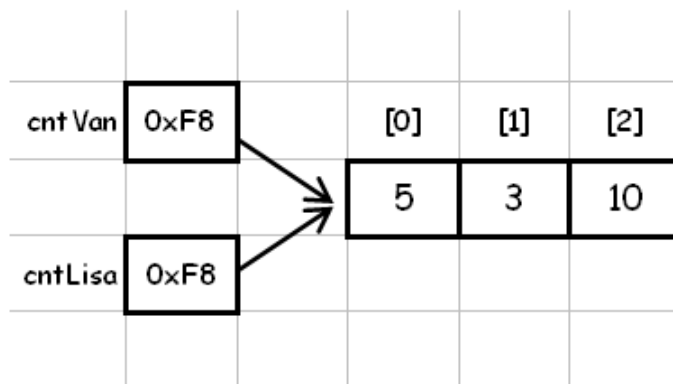
-----輸出-----

書的價格為: 200 350 250

Array

◆ 陣列指派(傳址)

- cntVan內存著**實體記憶體位址**，指派給cntLisa後，此二個陣列變數**指向同一個實體**。



Array

◆ 陣列傳址 範例

```
1. class Array_Assign{
2.     public static void main(String args[]){
3.         int[] cntVan = {5, 3, 10};
4.         int[] cntLisa = cntVan;
5.
6.         cntLisa[0] = 0;
7.         System.out.print("麗莎小姐將第一本書賣完後的存量: ");
8.         for(int count: cntLisa)
9.             System.out.print(count + " ");
10.        System.out.println();
11.        System.out.print("梵谷先生接手時書的存量: ");
12.        for(int count: cntVan)
13.            System.out.print(count + " ");
14.    }
15. }
```

麗莎小姐將第一本書賣完，所以該書存量為 0

cntLisa 與 cntVan 陣列變數指向同一個實體內容，所以會輸出相同結果

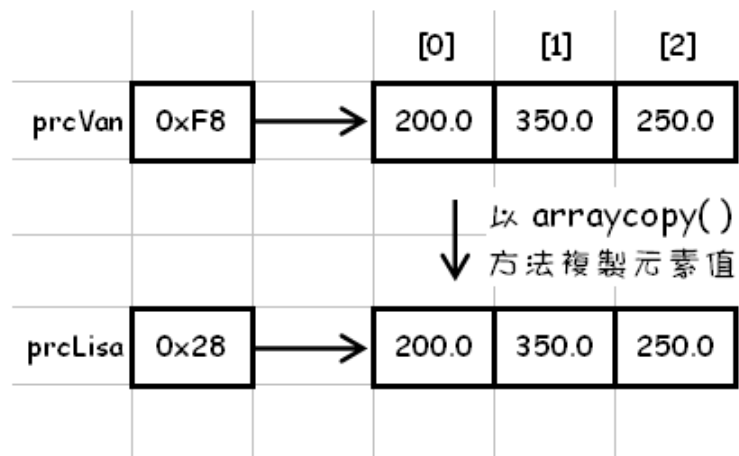
-----輸出-----

麗莎小姐將第一本書賣完後的存量: 0 3 10
梵谷先生接手時書的存量: 0 3 10

Array

◆ 陣列複製(傳值)

- 呼叫`arraycopy()`方法可以將`prcVan`陣列的所有元素值複製到`prcLisa`陣列內。
- 這二個陣列變數指向不同實體。



Array

◆ 陣列傳值 範例

```
1. class Array_Copy{
2.     public static void main(String args[]){
3.         double[] prcVan = {200.0, 350.0, 250.0};
4.         double[] prcLisa = new double[prcVan.length];
5.
6.         System.arraycopy(prcVan, 0, prcLisa, 0, prcVan.length);
7.         System.out.print("麗莎小姐將書籍八折出售: ");
8.         for(int i=0; i<prcLisa.length; i++)
9.             prcLisa[i]*=0.8;  → 將所有書的售價打八折後存回 prcLisa 陣列元素內
10.
11.         for(double price: prcLisa)
12.             System.out.print(price + " "); }
13.         System.out.println();
14.         System.out.print("梵谷先生書店的書籍售價: ");
15.         for(double price: prcVan)
16.             System.out.print(price + " "); }
17.     }
18. }
```

prcLisa與prcVan陣列變數指向不同的實體內容，所以當prcLisa陣列內的元素值改變並不會影響到prcVan的元素，因此輸出結果不同

-----輸出-----

麗莎小姐將書籍八折出售: 160.0 280.0 200.0
梵谷先生書店的書籍售價: 200.0 350.0 250.0

Array

◆ 陣列排序

- Java函式庫的「java.util.Arrays」類別的sort()方法提供了排序功能；
- 使用此功能必須在程式的第1行加上「import java.util.Arrays;」。

Array_Sort.java

```
1. import java.util.Arrays;
2.
3. class Array_Sort{
4.     public static void main(String args[]){
5.         int[] cntLisa = {5, 3, 10};
6.
7.         System.out.print("書籍依存量排序: ");
8.         Arrays.sort(cntLisa);
9.         for(int count: cntLisa)
10.             System.out.print(count + " ");
11.     }
12. }
```

→ 要使用排序功能，必須先 import

→ 呼叫 sort()方法並將要排序的陣列變數傳入即會做升冪排序(小至大)

-----輸出-----

書籍依存量排序: 3 5 10

Array

◆ 陣列搜尋

➤ 呼叫`Arrays.binarySearch()`方法即可回傳欲搜尋值在陣列內的索引，但要注意：

- 在執行搜尋前，必須先將該陣列排序。
- 如果該值不在該陣列內，則回傳負索引減1。

Array_Search.java

範例 `ArraySearch.java`

```
1. import java.util.Arrays;
2. class ArraySearch{
3.     public static void main(String[] args){
4.         int[] books = {550, 580, 470};
5.         Arrays.sort(books); // 搜尋前必須先排序
6.
7.         // 搜尋的值在陣列內
8.         int i1 = Arrays.binarySearch(books, 470);
9.         System.out.println("排序後書價 470 元的 index = " + i1);
10.
11.        // 搜尋的值不在陣列內
12.        int i2 = Arrays.binarySearch(books, 800);
13.        System.out.println("排序後書價 800 元的 index = " + i2);
14.    }
15. }
```

-----輸出-----

排序後書價470元的index = 0
排序後書價800元的index = -4

Content

◆ 一維陣列

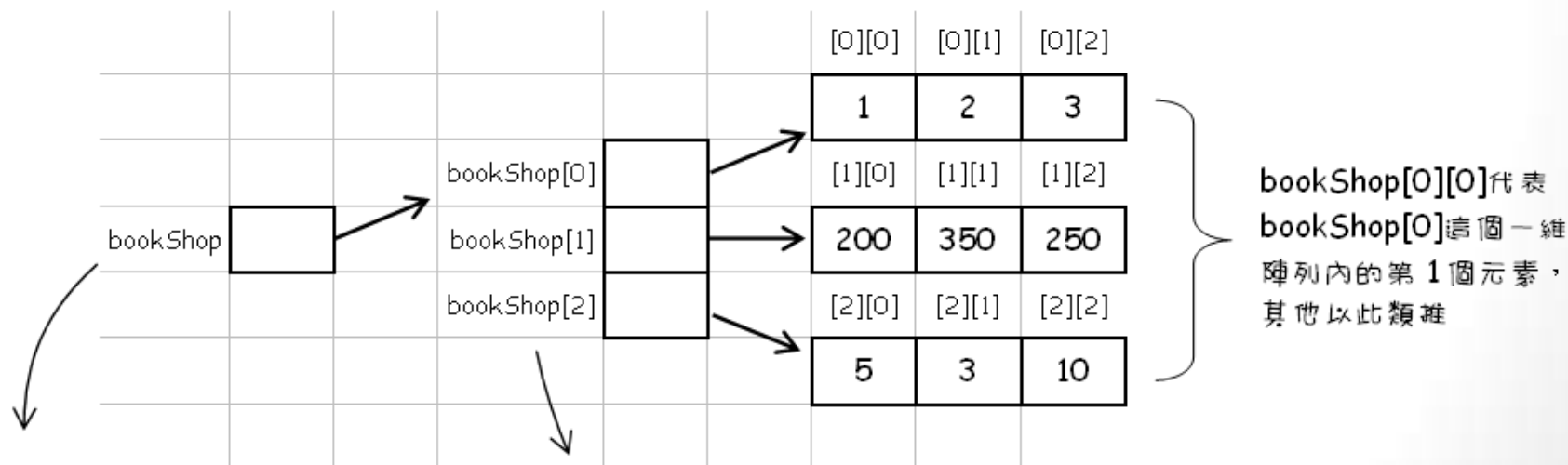
◆ 二維陣列

◆ 多維陣列

Arrays of Arrays

- ◆ 二維陣列，就是由多個一維陣列組成，也就如同一張表格；所以若要存取二維陣列內的元素，必須如同座標軸般指定列與欄。

➤ `int[][] bookShop = {{1, 2, 3},{200, 350, 250},{5, 3, 10}};`



bookShop 是二維陣列，
bookShop.length 為 3，
代表有 3 個一維陣列

bookShop[0]代表第 1 個一維陣列，其餘以此類推。
bookShop[0].length 為 3，代表有 3 個元素。

二維陣列 – 宣告

◆ 語法：

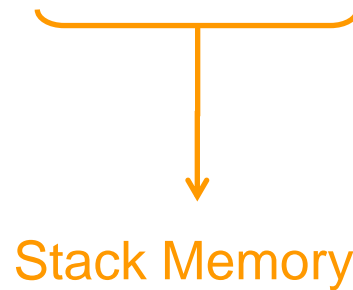
```
type [ ][ ] array_name;
```

```
type array_name [ ][ ];
```

Ex:

```
int [ ][ ] yearlySales;
```

yearlySales



陣列大小

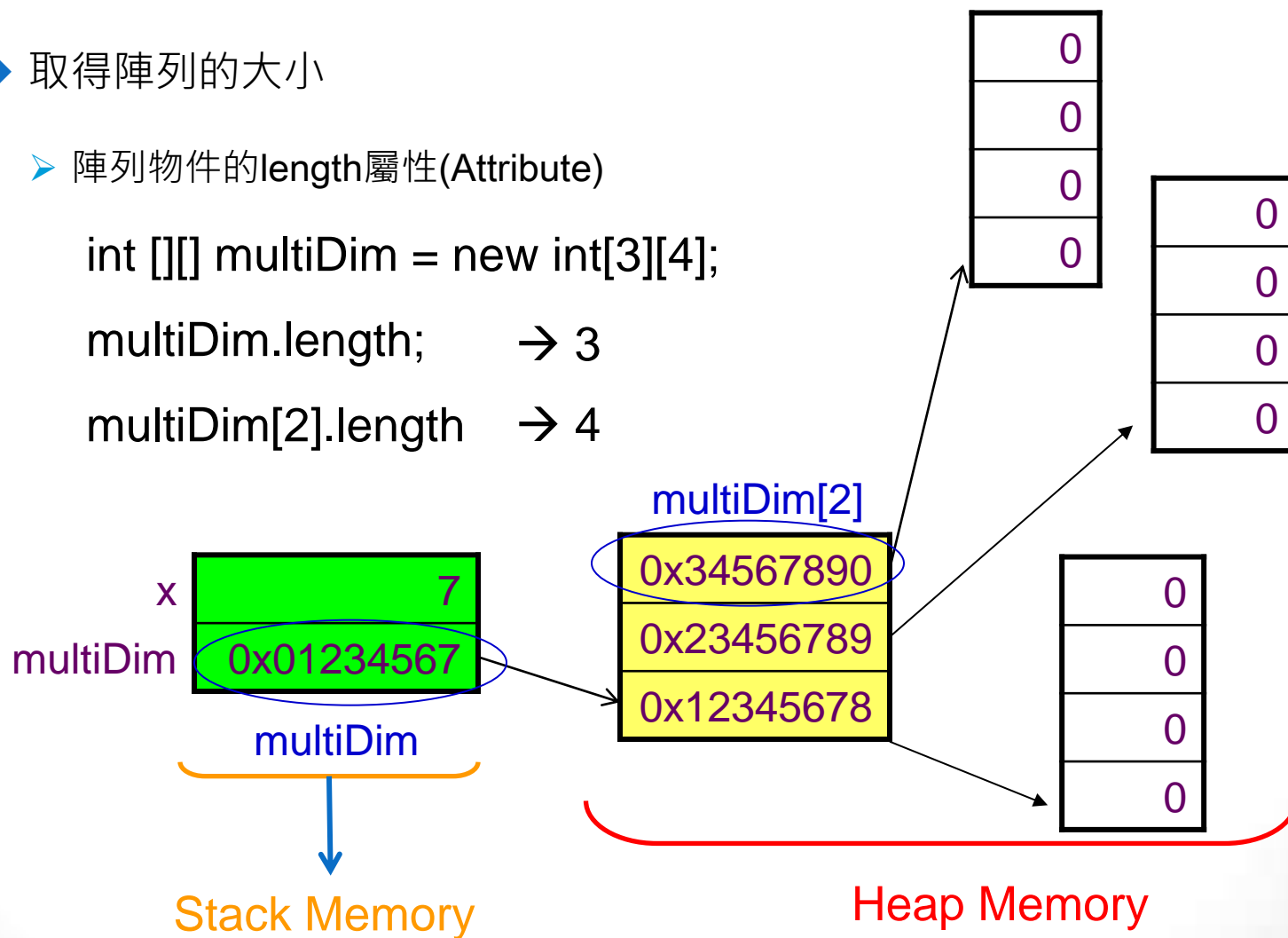
◆ 取得陣列的大小

- 陣列物件的length屬性(Attribute)

```
int [][] multiDim = new int[3][4];
```

```
multiDim.length;    → 3
```

```
multiDim[2].length  → 4
```



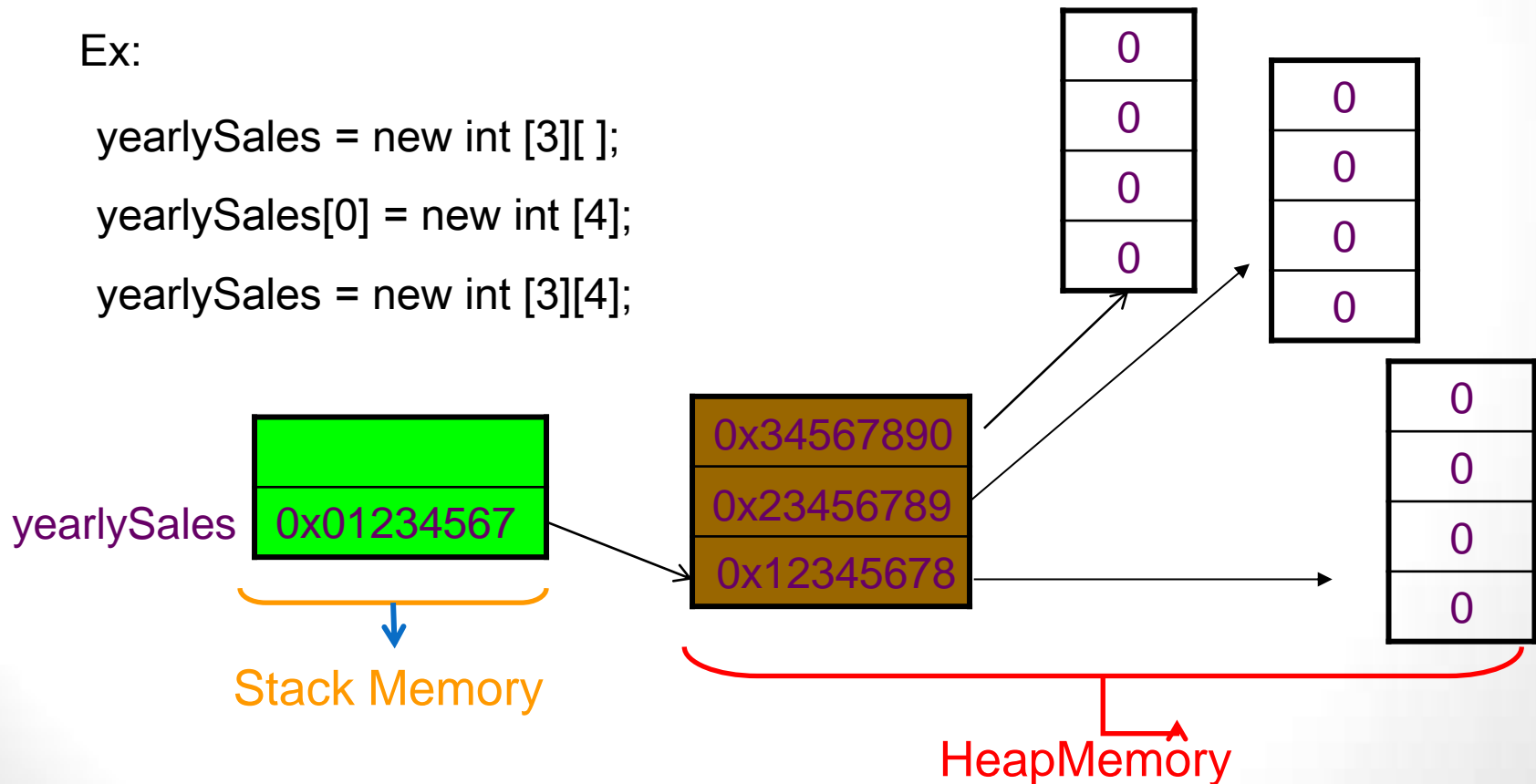
二維陣列 – 實體化

◆ 語法：

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

Ex:

```
yearlySales = new int [3][ ];  
yearlySales[0] = new int [4];  
yearlySales = new int [3][4];
```



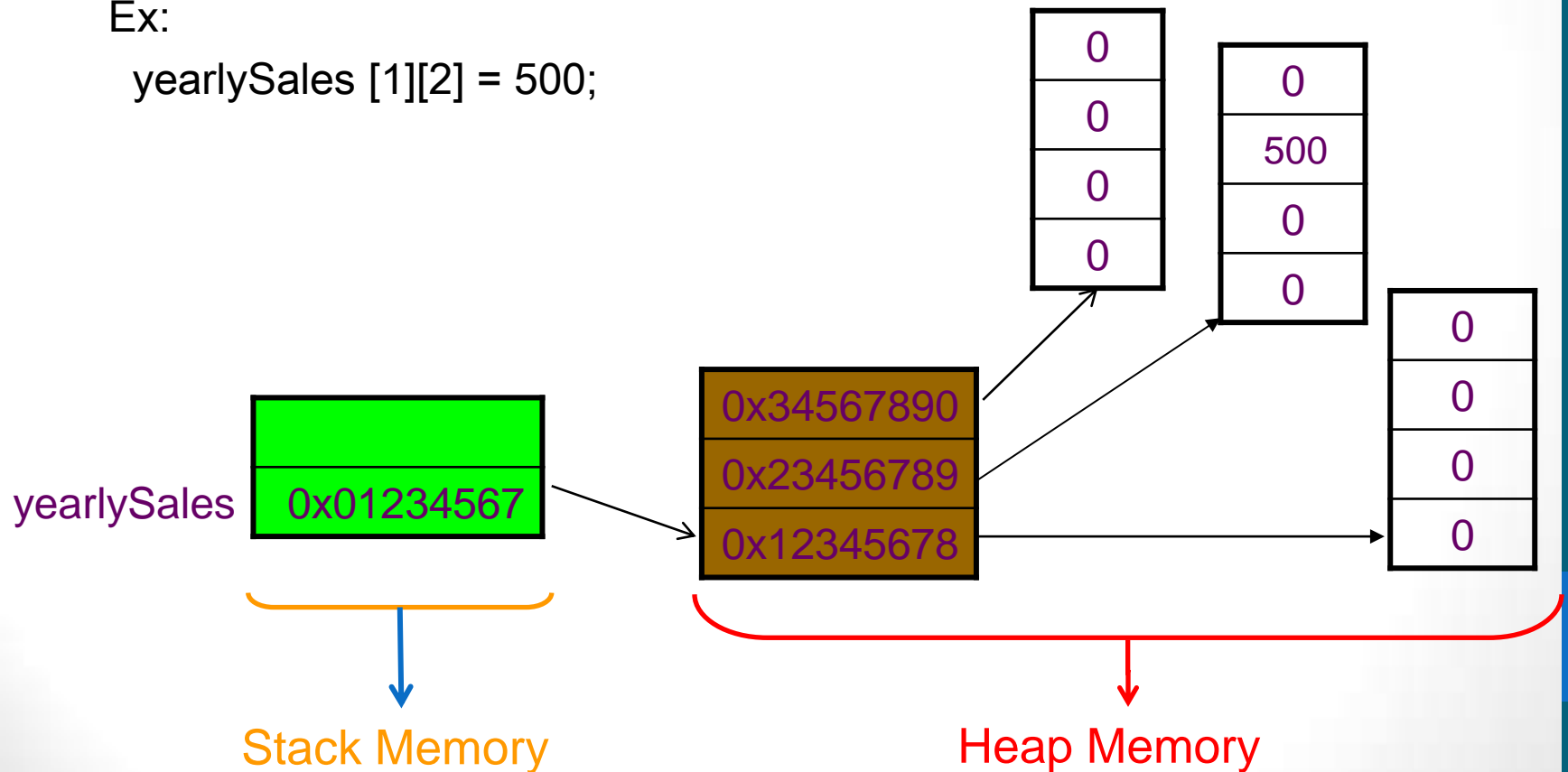
二維陣列 – 初始化

◆ 語法：

`array_name [number_of_arrays] [index] = value;`

Ex:

`yearlySales [1][2] = 500;`



Arrays of Arrays

```
1. class Array_2D{
2.     public static void main(String args[]){
3.         String[] title = {"編號", "價格", "庫存量"};
4.         int[][] bookShop = {{1, 2, 3},{200, 350, 250},{5, 3, 10}};
5.
6.         for(int i=0; i<bookShop.length; i++){
7.             System.out.print(title[i] + "\t");
8.
9.             for(int j=0; j<bookShop[i].length; j++){
10.                System.out.print(bookShop[i][j] + "\t");
11.            }
12.            System.out.println();
13.        }
14.    }
```

宣告一維字串陣列以
儲存標題名稱

i 遞增一次，就會跳
到下一個一維陣列

j 遞增一次，就會跳到
下一個元素

相當於按下 tab 鍵，
主要是為了對齊

-----輸出-----

編號	1	2	3
價格	200	350	250
庫存量	5	3	10

Content

◆ 一維陣列

◆ 二維陣列

◆ 多維陣列

Arrays of Arrays

◆ 多維陣列

- 一維陣列是一列；二維陣列是一個面；三維陣列就是一個立方體。要取出三維陣列內的元素值，一般會搭配3層巢狀式迴圈；
- 換句話說，幾維陣列就要搭配幾層巢狀式迴圈。

- 宣告與指派Example：

```
String[][][] school = new String[3][2][2];
```

```
String[][][] school = {{{"張一","李一"},"王一","陳一"},  
                        {"張二","李二"},"王二","陳二"},  
                        {"張三","李三"},"王三","陳三"}}};
```

Arrays of Arrays

◆ 多維陣列的初始化內容宣告：

- `int m[][] = { {1,2}, {1,2,3} };`
- 分別以括號 ({ }) 來表示不同階層的陣列

◆ 陣列的宣告是從左至右的方式來初始陣列大小：

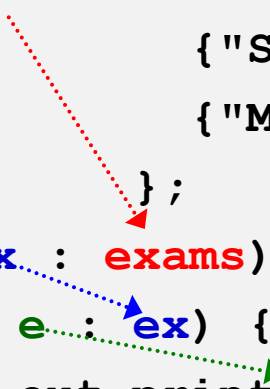
- `int m[][] = new int[3][];` 正確
- `int m[][] = new int[][3];`

此為錯誤的宣告方式，會產生編譯錯誤(Compile error)。

Arrays of Arrays

◆ 分析多維陣列 - 使用 for-each

```
01 String[][] exams = {  
02     {"SCJP", "SCWCD", "SCMAD"},  
03     {"MCSD", "MCAD", "MCDBA"}  
04 };  
05 for(String[] ex : exams) {  
06     for(String e : ex) {  
07         System.out.print(e + ", ");  
08     }  
09     System.out.println();  
10 }
```



執行結果：SCJP, SCWCD, SCMAD, MCSD, MCAD, MCDBA,

陣列邊界 Array Bounds

◆ 陣列索引值

- 由 0 開始到 `length-1`
- 存取陣列的索引值超過邊界範圍，會有執行時期錯誤：
`ArrayIndexOutOfBoundsException`
- 依序存取陣列元素

```
for (int i = 0; i < list.length; i++) {  
    System.out.println(list[i]);  
}
```

陣列長度不可改變

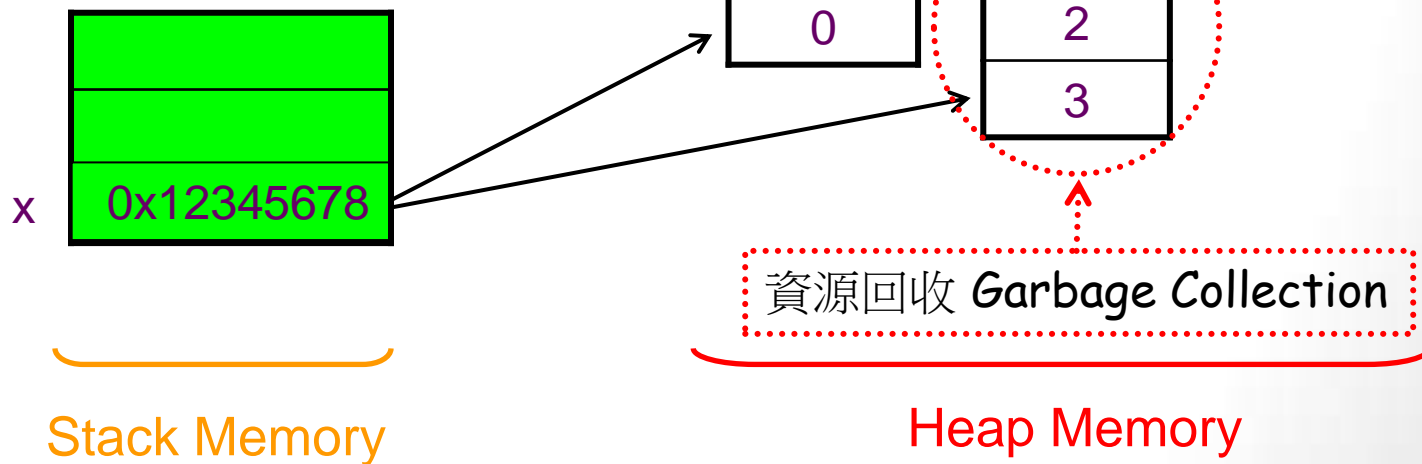
◆ 在Java中，陣列不被允許重新定義大小(Resize)

◆ 要改變陣列大小必須重新創建陣列

➤ 物件參考值會指向新創建出來的物件陣列實體

```
int [ ] x = {1,2,3};
```

```
x = new int[4];
```



Q & A