第四章　陣列　(Array) 處理

本章內容

- 4-1　認識陣列 (Array)
- 4-2　定義一維陣列
- 4-3　定義多維陣列
- 4-4　使用 Array 和 Arrays 類別

4-1 認識陣列 (Array)

- 儲存多個類似的資料時，若使用了太多的變數名稱來儲存類似的資料，這不僅會造成設定變數名稱的困難，在使用這些變數時，更會造成撰寫程式的不方便
- 陣列是一種可用來儲存「**相同資料類型**」的一種資料型態。使用陣列的最大好處是可以利用一個變數名稱並利用陣列的索引值來存取資料的內容。

4-1-1 陣列是參考型別的資料類型

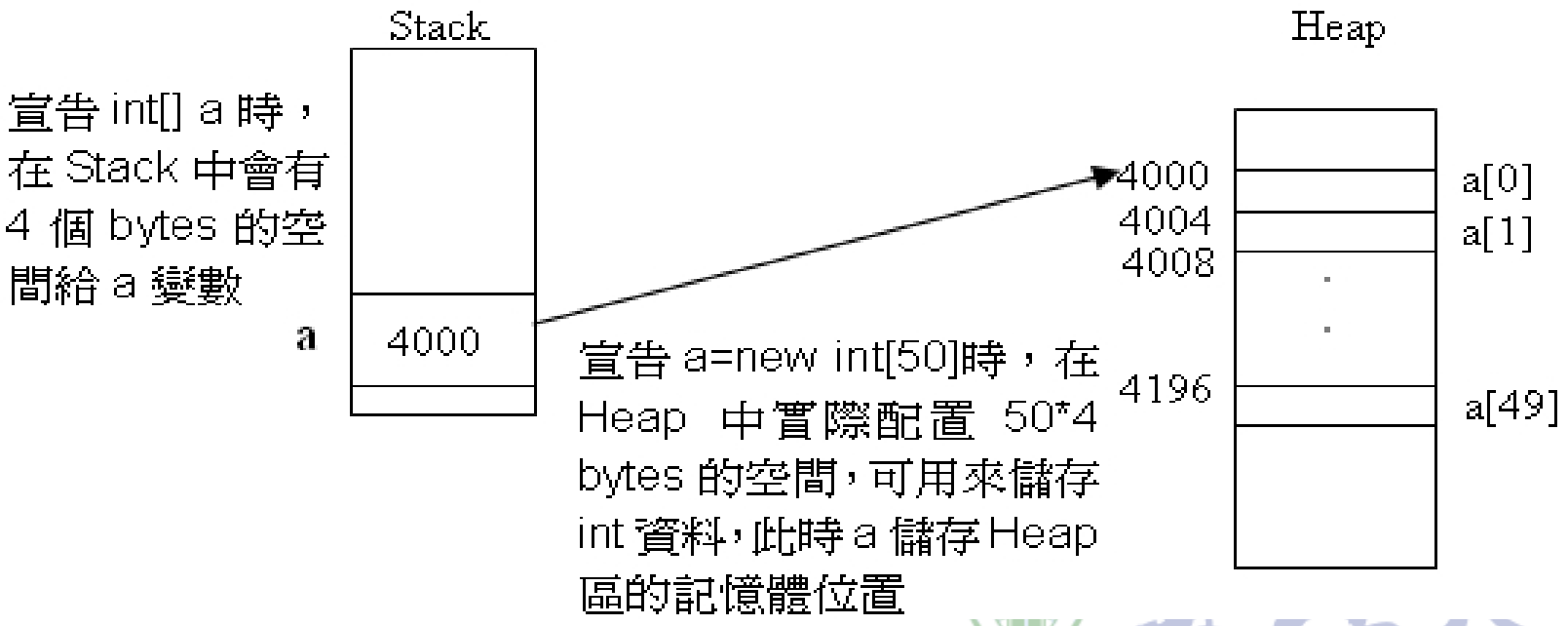
- 「參考資料型別」的資料型態可以是類別 (Class)，字串 (String)，陣列 (Array) 等資料類型。
- 宣告一個「參考資料型別」的變數僅只是宣告一個指向某個記憶體位置的變數，在沒有將物件實體化前，它並未被配置記憶體空間，所以不能直接用來儲存資料。
- 如果要為「參考資料型別」的變數配置記憶體空間，那我們必需將該變數「實體化」。在 Java 中，我們可以利用「 new 」關鍵字來將變數實體化 ：

```
int[] Student = new int[50];
```

4-1-1 陣列是參考型別的資料類型

- 參考型別變數的記憶體圖

int[] a = new int[50];



4-2 定義一維陣列

- 陣列具有維度，維度是指陣列中能夠儲存相關類型資料的種類數。例如：儲存班上學生數學成績的陣列即可視為是一維陣列。
- 定義陣列時，可以根據需求決定是否要在宣告時即實體化陣列。如果不要立即配置記憶體空間，我們可以如此定義：
  - 資料型態 [] 陣列名稱 ;　// int[] math_score;
  - 資料型態 陣列名稱 [];　// int math_score[];

4-2 定義一維陣列…

- 語法中的「元素個數」也可以稱為「**索引(index)**」。例如：將「math_score」變數配置空間以便儲存 50 個學生的數學成績，則語法為：
  - math_score = new int [50];
- 我們也可以在宣告陣列變數時，即將該變數實體化，語法如下：
  - 資料型態 [] 陣列名稱 = new 資料型態 [元素個數];
  - 資料型態 陣列名稱 [] = new 資料型態 [元素個數];

4-2-1 利用迴圈來控制陣列

- 使用陣列時，我們常會利用迴圈來簡化程式的寫作
  - 寫作方式如同：

```
for (i=0; i<3; i++) {
    math_score[i]);
}
```

4-2-2 使用「length」取得陣列的長度

- 我們可以使用「length」來取得某一陣列的長度，例如：

```
math_score.length          // 取得的長度值
```

- 利用迴圈處理陣列時，常會使用到「length」屬性。例如：

```
for (i=0; i<math_score.length; i++) {
  math_score[i]);
}
```

4-2-3 新的「foreach」使用方式－J2SE 5.0

- J2SE 5.0 提供了一項新的「foreach」語法可供我們更方便的使用陣列的內容。您可以將它視為「for」的加強版。如果使用於陣列上，語法如下：

```
for ( 陣列型別 變數名稱 : 陣列名稱 ) {
        依次處理「變數名稱」的敘述;
}
```

- 使用範例如同：

```
for (int element: math_score) {
        System.out.println(element);
}
```

- 「foreach」只能用來取出陣列中元素的內容，並不能用來設定元素的值。

4-2-4 一維陣列的初始化

- 陣列宣告時，也可以直接的初始化陣列的內容。您可以直接將陣列初始化的值置於「{」和「}」中，並使用「，」分隔每個值。語法如同：

  資料型態 [] 陣列名稱 = new 資料型態 [] { 值 1, 值 2, ...} ;

- 例如以下的範例：

```
char[] name = new char[] {'A', 'L', 'E', 'X'};
```

- 初始化陣列並同時指定值時，不需要在「[]」中指定陣列的大小，否則，程式會產生編譯錯誤。

4-2-4 一維陣列的初始化…

- 當您在宣告一個陣列時，就決定要立即初始化該陣列的內容，則 Java 也允許省略「 new 」關鍵字的使用。語法如同：

  資料型態 [] 陣列名稱 = { 值 1, 值 2, ...} ;

- 如果以這種方式來建構陣列，我們不需要特別的使用「 new 」關鍵字。例如以下的範例：

```
char[] name = {'A', 'L', 'E', 'X'};
```

4-2-5 參考型別的資料會由 JVM 自動初始化

- JVM 會負責初始化「參考型別」的資料內容。各資料型別的初始值如下表:

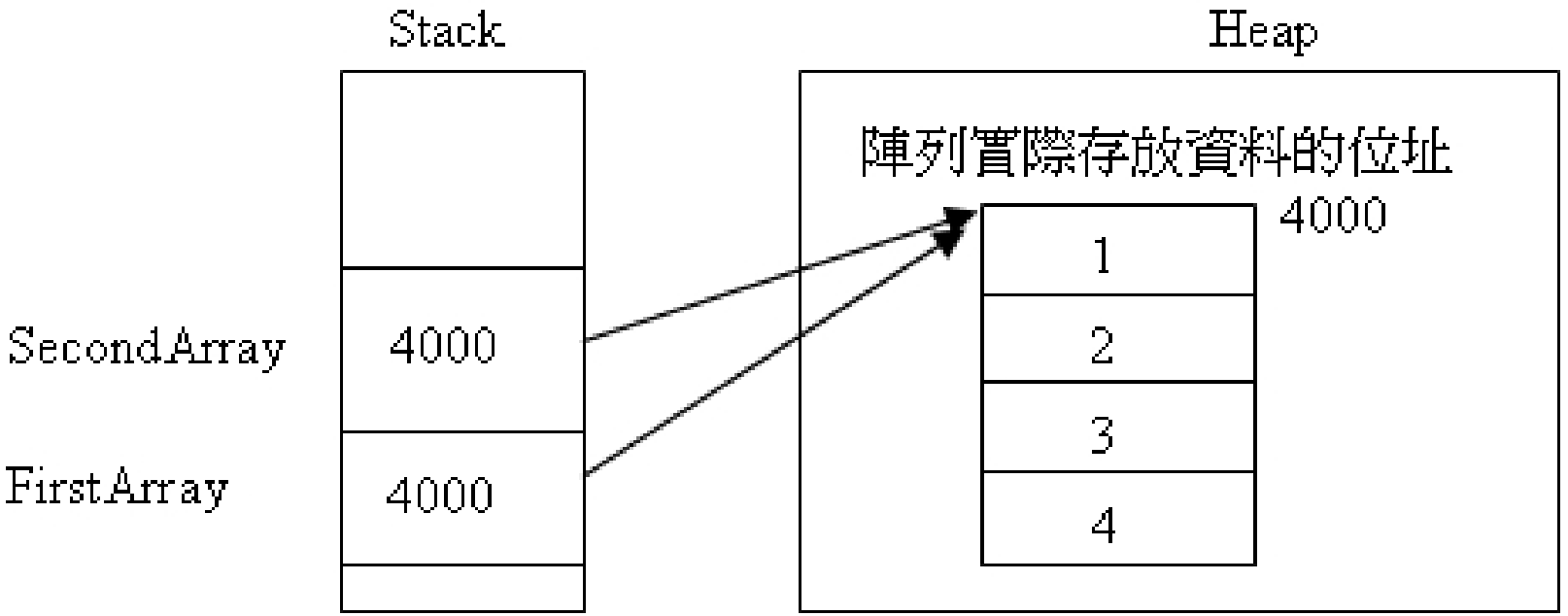
保留字	名稱	初始值
byte	位元組	0
short	短整數	0
int	整數	0
long	長整數	0L
float	浮點數	0.0F
double	倍精數	0.0D
char	字元	\u0000
boolean	布林值	false
All Other	其他的參考型別	null

4-2-6 使用「＝」運算子複製陣列的內容？

- 使用「參考型別」的資料類型時，如果將某一個「參考型別」的變數 a 的內容「指定 (=) 」給另一個「參考型別」的變數 b 時，這僅是代表兩個變數指向同一塊記憶體空間，而不是將一個物件的內容複製給另一個物件。 請參考次一頁的圖形:

4-2-7 使用「＝」運算子複製陣列內容?...

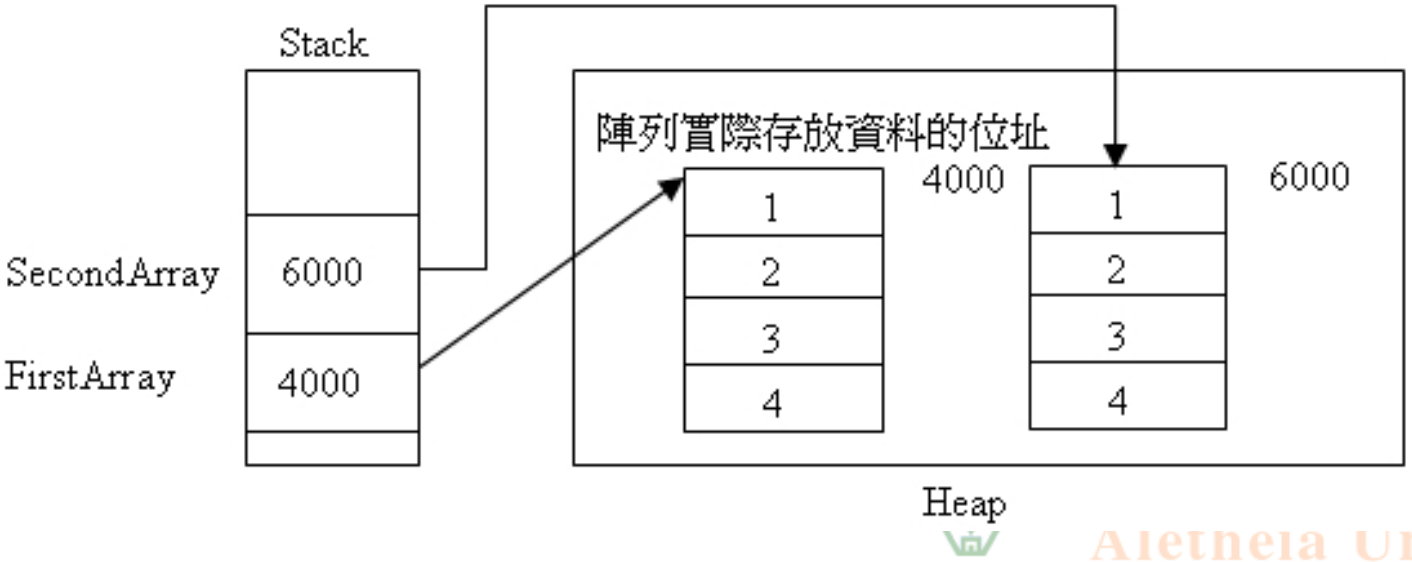
```
int[] FirstArray = {1, 2, 3, 4};
int[] SecondArray;
SecondArray = FirstArray;
```



4-2-8 使用「clone」複製陣列物件

- 我們可以延用該類別中的「clone()」方法來達成複製陣列的目的。該方法會產生並傳回某個物件的副本。

```
SecondArray = FirstArray.clone();
```



4-2-9 使用「arraycopy()」複製陣列內容

- 除了使用「Object」類別的「clone()」方法複製陣列之外，我們還可以使用「java.lang.System」類別下的「arraycopy()」方法來複製陣列，語法如下：

```
arraycopy(Object src, int srcPos, Object dest, int destPos, int length)
```

  - 「src」代表需要複製的來源陣列;
  - 「srcPos」代表來源陣列的開始索引值;
  - 「dest」代表要目的陣列;
  - 「destPos」代表目的陣列的開始索引值;
  - 「length」代表需要複製的資料長度

- 使用範例如同:

4-2-9 使用「arraycopy()」複製陣列內容

- 利用「**arraycopy()**」方法，我們可以將陣列中的某些，或是全部的元素複製到另一個陣列的任何位置中。
- 但使用時，您必需考慮到目的陣列的大小是否能容納原始陣列中需要被複製的元素個數。而您也必需要考慮來源陣列可取得的元素個數是否小於「length」的值
  - 「ArrayIndexOutOfBoundsException」和「ArrayStoreException」例外
- 「arraycopy」只能將來源陣列的內容複製到目的陣列中，它並不會為目的陣列配置記憶體空間。

4-2-10 重新改變陣列的大小？

- 在 Java 中，一旦產生陣列，並配置了記憶空間後，該陣列即無法改變大小。
- 如果將陣列變數重新指向另一個不同大小的陣列，則 Java 會重新指定另一個記憶空間來儲存新陣列的內容，而原先陣列的內容就無法再取回來。
- 原先存放值的記憶體空間會等待 JVM 自動啓動「**垃圾回收 (GC, Garbage Collection)**」機制時將它回收。

4-3 定義多維陣列

- 多維陣列可以視為多個一維陣列的組合。多維陣列的定義語法如下:

  型態 [] [] []... [] 陣列名稱 ;

  型態 陣列名稱 [] [] []... [];

- 如果要在定義陣列時，即配置記憶體空間，我們必需將陣列實體化，語法如:

  型態 [] [] []... [] 陣列名稱

  = new 型態 [第一維陣列的元素個數][第二維陣列的元素個數]...[第 n 維陣列的元素個數];

  型態 陣列名稱 [] [] []... []

  = new 型態 [第一維陣列的元素個數][第二維陣列的元素個數]...[第 n 維陣列的元素個數];

4-3-1 多維陣列的使用技巧

- 多維陣列也可以在宣告時即初始內容，以二維陣列為例，語法：

```
型態 [][] 陣列名稱 = new 型態 [][] {
                { 欄 1 值 , 欄 n 值 },          /* 代表第一列的內容 */
                { 欄 1 值 , 欄 n 值 },          /* 代表第二列的內容 */
                ....
                { 欄 n 值 , 欄 n 值 }};                  /* 代表第 n 列的內容 */
```

- 或是：

```
型態 [][] 陣列名稱 = {
                { 欄 1 值 , 欄 n 值 }, /* 代表第一列的內容 */
                { 欄 1 值 , 欄 n 值 }, /* 代表第二列的內容 */
                ....
                { 欄 n 值 , 欄 n 值 }};          /* 代表第 n 列的內容 */
```

4-3-2 多維陣列的使用技巧…

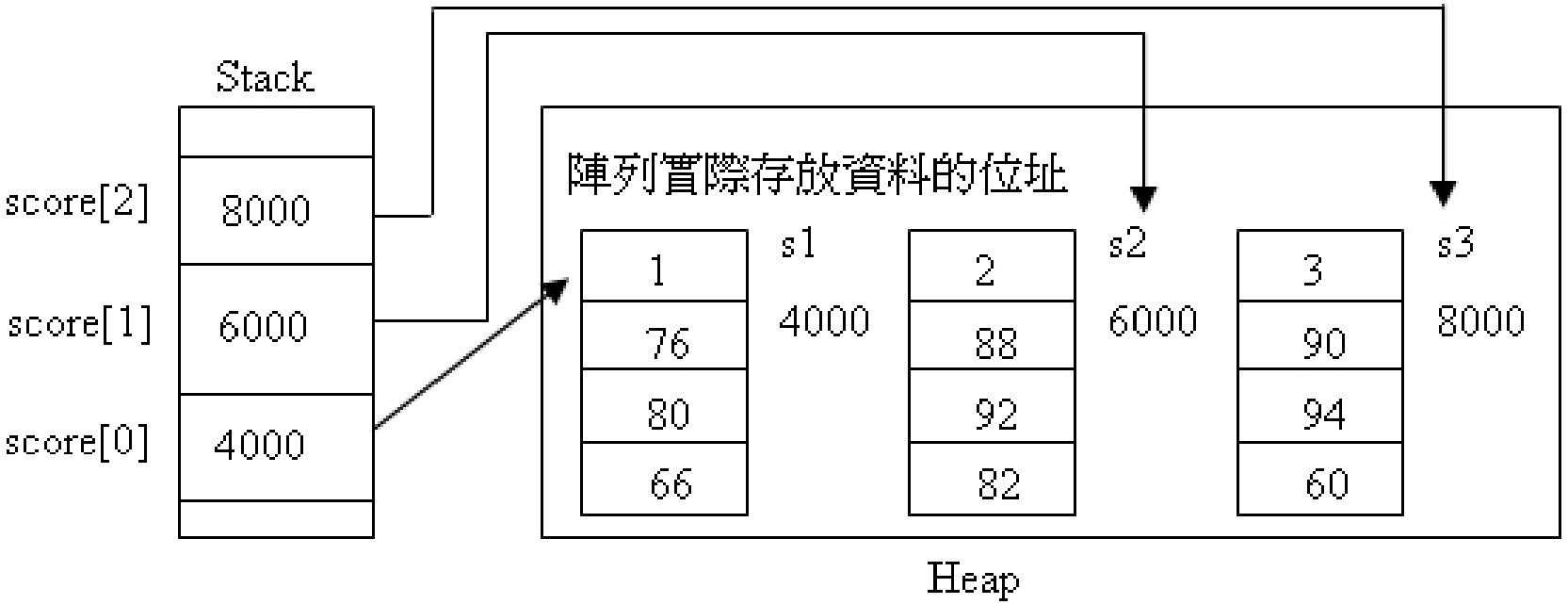
- 既然多維陣列可以視為多個一維陣列的組合。我們時常先宣告一個多維陣列的變數，再讓該變數中的元素指向另一個維度較小的陣列。

- 使用範例:

```
int[][] score = new int[3][]; // 準備指向另外三個一維陣列
int[] s1 = {1, 76, 80, 66};   // 宣告第一個一維陣列，並指定內容
int[] s2 = {2, 88, 92, 82};   // 宣告第二個一維陣列，並指定內容
int[] s3 = {3, 90, 94, 60};   // 宣告第三個一維陣列，並指定內容
score[0] = s1;
score[1] = s2;
score[2] = s3;
```

4-3-2 多維陣列的使用技巧…

- 記憶體空間配置情形:



4-3-2 多維陣列的使用技巧…

- 您可以使用 J2SE 5.0 新增的「foreach」語法來顯示多維陣列的內容，以二維陣列而言，語法的結構如下：

```
for ( 陣列型別 [] 變數名稱 A : 陣列名稱 ) {
    for ( 陣列型別 變數名稱 B : 變數名稱 A) {
      // 依次處理「變數名稱 B」的敘述 ; }
}
```

- 例如：

```
for(int[] firtArray : score){                    // 取得第一維陣列的元素
    for (int element : firtArray)                // 取得第二維陣列的元素
      System.out.print(element + "\t");
    System.out.println();
```

4-3-3 定義不規則陣列

- 在宣告多維陣列在時， Java 允許我們保留定義最後一維陣列大小的彈性。例如：

```
// 宣告一個二維陣列，第一維有 5 個元素
int [][] score = new int[5][];
int [] s1 = {1, 76, 80};  // 宣告一個一維陣列，並指定內容
score[0] = s1;                  // 將 s1 陣列中的內容指定給 score
int [] s2 = {1, 76, 80, 56}; // 宣告一個一維陣列，並指定內容
score[1] = s2;    // 將 s1 陣列中的內容指定給 score
```

- 在上述的範例中， score[0] 和 score[1] 指向不同長度的一維陣列

4-4 使用 Array 和 Arrays 類別

- Java 中提供了「 Array 」類別來協助陣列的相關問題。 Array 類別並沒有提供建構子，但 Array 類別中提供了相當多的方法可以用來設定或取得陣列的內容。
- Array 類別中提供的「 get(Object array, int index) 」方法可用來取出陣列中特定索引值的元素內容。
- Array 類別中也提供的「 set(Object array, int index) 」方法可用來設定陣列中特定索引值的元素內容。

4-4 使用 Array 和 Arrays 類別…

- 另一個更有用的類別是「Arrays」類別來協助陣列的相關問題。Arrays 類別也沒有提供建構子，但 Arrays 類別中提供了相當多的方法可以用來簡化陣列的操作，該類別是屬於「java.util」套件，使用前，您必需先引入該套件。語法如下：

```
import java.util.*;
```

4-4-1 判斷陣列的內容是否相同

- 陣列是一種物件，您不能使用指定運算子「＝」。如果您使用邏輯運算子「＝＝」，則是判斷兩個變數是否指向同一個陣列，而不是判斷兩個陣列的內容是否相同。
- Arrays 類別中提供了自己的 equals 方法來判斷兩個陣列的內容是否相同。該方法是屬於類別方法。使用的語法如下：

  Arrays.equals(陣列 A, 陣列 B)

4-4-2 陣列的排序

- Arrays 類別中提供「 sort 」方法來進行陣列的排序工作。「 sort 」方法具有多種的多載版本，最簡單的使用方式為:

  Arrays.sort(陣列名稱);

- 如果您不想使用 sort 方法來將陣列的內容做排序的處理，您必需利用巢狀迴圈來逐一的比對陣列中的每個元素的大小，在程式的寫作上，您會使用到 Swap 的觀念。

4-4-3 搜尋陣列中的元素

- Arrays 類別中提供的「 binarySearch 」方法可用來在陣列中搜尋特定的元素，如果可以在陣列中搜尋到特定的元素，則會傳回該元素的索引值，如果搜尋不到，則會傳回 -1 。「 binarySearch 」方法最簡單的使用方式為:

  Arrays.binarySearch(陣列名稱 , 型別 key);

- 如果您不想使用 binarySearch 方法來搜尋陣列的內容，您必需利用迴圈來逐一的比對陣列中的每個元素的值