الفصل الخامس

٥

المصفوفات

مقدمة

المصفوفة تعنى في البرمجة Array وهي مكان أو متغير يحتوي على أكثر من قيمة أو عنصر من نفس النوع (لا يمكن إنشاء مصفوفة تتضمن أكثر من نوع) فيمكن إنشاء مصفوفة أرقام أو مصفوفة أسماء وكل عنصر Element له فهرس أو مفتاح Index يدل عليه يبدأ هذا الفهرس من الصفر إذا لم نقم بتحديده،

وتختلف عن المتغير أن المصفوفة تحتوي على قيم كثيرة في حين أن المتغير يحتوي على قيمه واحده وقد تكون غير محدده.

بفرض وجود أسماء لطلاب ونريد الترتيب الأبجدي لتلك الأسماء فلا يمكن تحقيق ذلك بسهوله من خلال المتغيرات ولكن يمكن ترتيبها عند وضعها في مصفوفة حيث تحجز المصفوفة عناوين متسلسلة في الذاكرة ويتوقف حجم كل جزء على مقدار حجم النوع الذي عرفت به المصفوفة

مزايا المصفوفة

-تقليل عدد المتغيرات المتشابهة، فمثلاً إذا كنا نريد تعريف 10 متغيرات نوعهم int نقوم بتعريف مصفوفة واحدة تتكون من 10عناصر.

-تطوير الكود: عند تخزين المعلومات داخل مصفوفة فإنه يمكن تعديلهم، ومقارنتهم أو استدعائهم كلهم مرة واحدة بكود صغير جداً باستخدام الحلقات.

-يمكن الوصول لأي عنصر من خلال index

المشكلة الرئيسة في الـ Array أن حجمها ثابت فعند تعريف مصفوفة وتحديد عدد عناصرها لا يمكن تغيير حجمها فيما بعد، أي لا يمكن حذف عنصر أو إضافة عنصر جديد (هذه المشكلة تم حلها من خلال إطار تخزين متطور يسمى Collection)

خطوات إنشاء مصفوفة

الخطوة الأولى : الإعلان عن متغير المصفوفة

يمكن تحقيق ذلك بطريقتين:

الطريقة الأولى: كتابة نوع عناصر المصفوفة ثم كتابة اسم المصفوفة ثم وضع الأقواس [ ] حيث تدل تلك الأقواس على التعامل مع مصفوفة وليس متغيرات

مثال

```
String frindname [ ];
```

الطريقة الثانية: كتابة نوع المصفوفة ثم الأقواس [ ] ثم اسم المصفوفة

مثال

```
String [ ] frindname;
```

الخطوة الثانية : إنشاء عنصر المصفوفة

يمكن تحقيق ذلك بطريقتين:

الطريقة الأولى :استخدام الأمر new والذي يعمل على إنشاء عنصر أو Object حيث يحجز في الذاكرة مكان لـ Object جديد ويتطلب ذلك تحديد عدد عناصر المصفوفة

مثال

```
String age=new string [10]
```

حيث أن:

age : متغير من نوع مصفوفات حرفية

10 : عدد عناصر المصفوفة ، ويمكن كتابة السطر السابق على سطرين كما يلي:

```
String [ ] age;
Age=new string [10];
```

الطريقة الثانية: إنشاء عنصر المصفوفة بدون استخدام الكلمة new من خلال إعطاء المصفوفة قيمة ابتدائية وبالتالي لا نحتاج لتحديد عدد العناصر حيث نقوم بوضع العناصر نفسها داخل المصفوفة كما يلي :

```
String [ ] names={ "Ahmed" ," Ali" , "Nada" , "Nora" }
```

الخطوة الثالثة : التعامل مع عناصر المصفوفة (تخزين البيانات داخل المصفوفة والتعامل معها)

يتم التعامل مع عناصر المصفوفة بكتابة رقم عنصر المصفوفة بين أقواس المصفوفة [ ] وذلك بالصيغة التالية:

```
Array name [ No ]
```

حيث أن :

Name name : اسم المصفوفة

No : ترتيب العنصر فى المصفوفة وتجدر الإشارة إلى أن ترتيب العناصر في المصفوفة تبدأ من القيمة 0

الخلاصة

توجد طريقتين لإنشاء مصفوفة:

الطريقة الأولى: انشاء المصفوفة وتمرير القيم اليها فيما يعد.

الطريقة الثانية: انشاء المصفوفة وتمرير القيم اليها مباشرةً عند تعريفها

أولا : انشاء المصفوفة وتمرير القيم اليها فيما يعد

الصيغة Syntax

```
arrayRefVar = new datatype[size];
```

حيث أن:

arrayRefVar : اسم المصفوفة التي يجب أن تكون معرفة سابقاً.

new :تقوم بتوليد قيم أولية لجميع عناصر المصفوفة وإعطائهم القيمة صفر كقيمة أولية.

datatype :نوع القيم الأولية التي سيتم توليدها.

size :عدد عناصر المصفوفة.

ثانيا : انشاء المصفوفة وتمرير القيم اليها مباشرةً عند تعريفها

```
arrayRefVar = { value0, value1, ..., valuek };
```

arrayRefVar: إسم المصفوفة التي يجب أن تكون معرفة سابقاً.
value0 ، value1 ، valuek : القيم التي نعطيها للمصفوفة.

القيمة الافتراضية صفر تكون حسب نوع المصفوفة

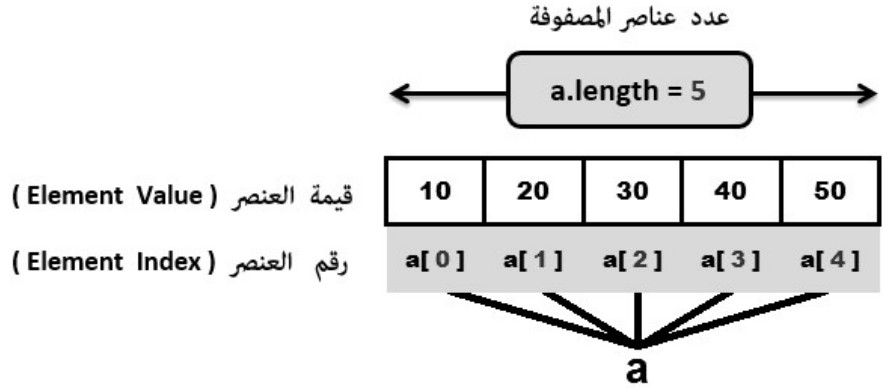
-إذا كان نوع المصفوفة **int** أو **long** فإن القيمة الافتراضية التي ستعطى لجميع عناصر المصفوفة هي **0**
-إذا كان نوع المصفوفة **double** أو **float** فإن القيمة الافتراضية التي ستعطى لجميع عناصر المصفوفة هي **0.0**
-إذا كان نوع المصفوفة **String** فإن القيمة الافتراضية التي ستعطى لجميع عناصر المصفوفة هي **null**

شكل المصفوفة في الذاكرة

نفترض أننا عرفنا مصفوفة نوعها **int** بالاسم **a** وتتكون من ٥ عناصر

```
int[ ] a = { 10, 20, 30, 40, 50 };
```

يمكن تصور شكل المصفوفة a في الذاكرة كالتالي:



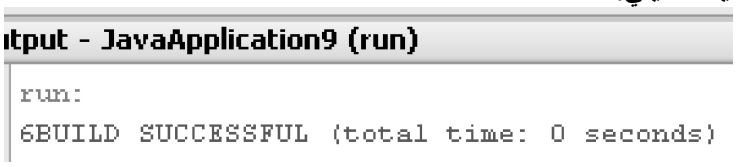
بما أن المصفوفة تتكون من ٥ عناصر تم إعطاء العناصر أرقام **index** بالترتيب من ٠ إلى٤ ، وأصبح عدد عناصر المصفوفة = ٥ و هو ثابت لا يمكن تغييره فيما بعد في الكود ، ويمكن استخدام أرقام الـ **index** للوصول الى جميع عناصر المصفوفة سواء لعرض القيم أو لتغييرها.

-الدالة Length

تستخدم لحساب عدد عناصر المصفوفة (طول المصفوفة)

```
1  public static void main(String[] args) {
2   int x;
3  int[] e={2,5,3,7,1,9};
4  x=e.length;
5  System.out.print(x);
6  }
```

ويظهر ناتج التنفيذ كما يلي:



نلاحظ أن قيمة X تساوى 6 وهى تدل على عدد الخانات المحجوزة للمصفوفة

مثال:

تعريف مصفوفة، وتغيير قيمة العنصر الأول ثم استعراضها من خلال الـ index الخاص بالعنصر، ثم استخدام الدالة length لطباعة عدد عناصر المصفوفة.

```
1  public class Main {
2    public static void main(String[] args) {
3   int[ ] a = { 10, 20, 30, 40, 50 };                  // تعريف مصفوفة تتكون من ٥ عناصر
4    a[0] = 99;                                         // تغيير قيمة العنصر الأول
5   System.out.println("a[0] = " + a[0]);               // عرض قيمة العنصر الأول
6   System.out.println("a.length = " + a.length);      // عرض عدد عناصر المصفوفة
7    }
8    }
```

نتيجة التنفيذ

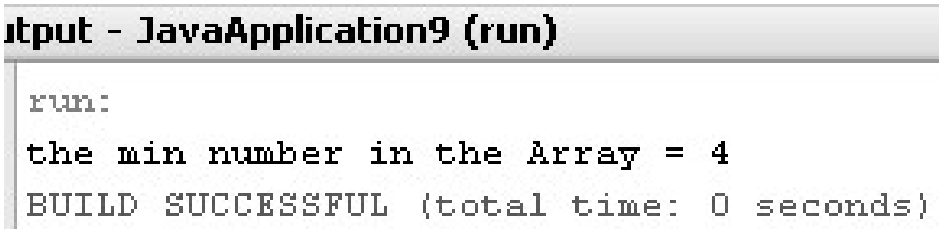
```
a[0] = 99
a.length = 5
```

مثال:

إيجاد أصغر قيمة من بين عدة قيم داخل المصفوفة

```
1  public static void main(String[] args) {
2  int[] a={12,6,4,33,7};
3  int min = a[0];
4  for(int i=0;i<a.length;i++){
5  if(a[i]<min)
6  min=a[i];
7  }
8  System.out.println("the min number in the Array = " + min );
9  }
10 }
```

ويظهر ناتج التنفيذ كما يلي:



ويمكن في البرنامج الواحد كتابة أكثر من Method بالإضافة إلى الـ Method الرئيسي ويتم استدعاء كل Method عند الحاجة وذلك من خلال الـ Methodالرئيسي

مثال:

إيجاد أكبر قيمة من بين عدة قيم داخل المصفوفة

```
1  public static int getMax(int[] a)
2  {
3     int max = a[0];
4     for(int i=0;i<a.length;i++){
5      if(a[i]>max)
6       max=a[i];
7  }
8  return max;
9  }
10 public static void main(String[] args)
11   {
12      int[] b={4,10,11,17,3,8};
13      int max=getMax(b);
14    System.out.println("the max number in the Array = " +
15 max );
16 }
   }
```

ويظهر ناتج التنفيذ كما يلي :

```
tput - JavaApplication10 (run)
run:
the max number in the Array = 17
BUILD SUCCESSFUL (total time: 0 seconds)
```

ملحوظة

public static int getMax(int[] a)

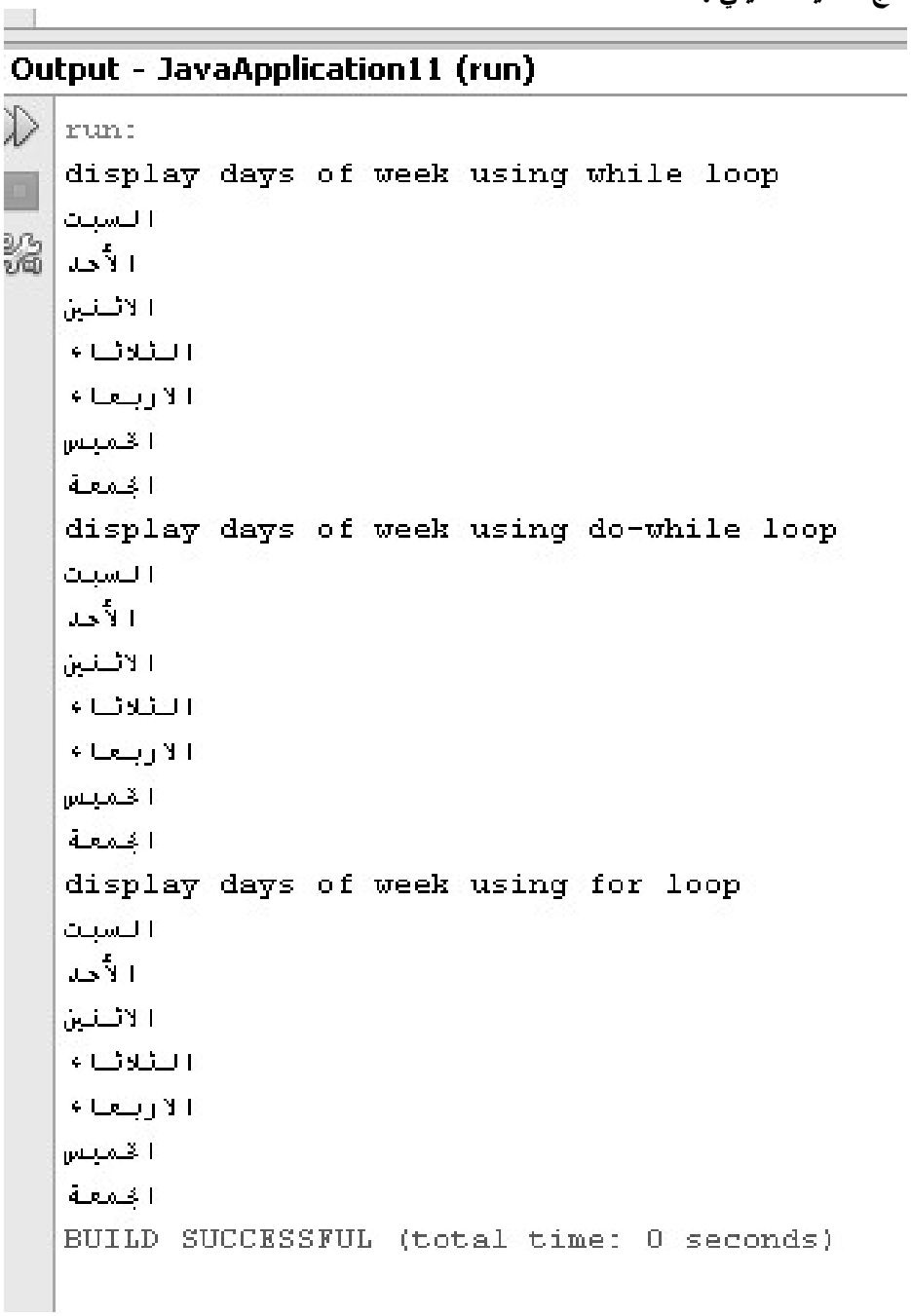
حيث أن :

نوع المتغير أو الـ object حتى يمكن إرساله إلى method وهنا من نوع مصفوفة أعداد حقيقة (int) ويسمى ذلك الجزء المعامل Parameter	**القيمة التي يرجعها الـ method وهى هنا من النوع int**

مثال :
استخدام المصفوفات مع الحلقات التكرارية (for - Do while– While)لطباعة أيام الأسبوع

```
1    public static void main(String[] args)
2   {
3     //Declare and initialize String array of the week
4   String[]days={"السبت","الأحد","الاثنين","الثلاثاء", "الأربعاء" ,
5   "الخميس","الجمعة"};
6    //Display days of the week using while loop
7   System.out.println("display days of week using while loop");
8         int counter=0;
9         while(counter<days.length)
10        {
11           System.out.println(days[counter]);
12           counter++;
13        }
14    //Display days of the week using do-while loop
15  System.out.println("display days of week using do-while loop");
16        counter=0;
17        Do
18       {
19           System.out.println(days[counter]);
20           counter++;
21        }
22  while(counter<days.length);
23        //Display days of the week using for loop
24    System.out.println("display days of week using for  loop");
25        for(counter=0;counter<days.length;counter++)
26        {
27           System.out.println(days[counter]);
29        }
30     }
    }
```

يظهر ناتج التنفيذ كما يلي :



مثال شامل

-تعريف مصفوفة فارغة نوعها int بالاسم numbers تتكون من ٥ عناصر ، عرض قيمها الأولية باستخدام حلقة ، إعطاء قيمة لكل عنصر فيها ، عرض جميع قيمها من جديد باستخدام حلقة

```
1   public class Main {
2           public static void main(String[] args) {
3               // 1- تعريف المصفوفة
4                  int[ ] numbers = new int[5];
5                  // 2- عرض قيم جميع عناصرها
6                  for(int i=0; i<numbers.length; i++)
7                  {
8                  System.out.println("numbers[" + i + "]: " + numbers[i]);
9                  }
10                 // 3- تغيير قيم جميع عناصرها
11                 numbers[0] = 1000;
12                 numbers[1] = 2000;
13                  numbers[2] = 3000;
14                 numbers[3] = 4000;
15                 numbers[4] = 5000;
16                  System.out.println();
17                 // 4- عرض قيم جميع عناصرها
18                 for(int i=0; i<numbers.length; i++)
19                 {
20                 System.out.println("numbers[" + i + "]: " + numbers[i]);
21                 }
           }
   }
```

يظهر ناتج التنفيذ كما يلي:

```
numbers[0]: 0
numbers[1]: 0
numbers[2]: 0
numbers[3]: 0
numbers[4]: 0
numbers[0]: 1000
numbers[1]: 2000
numbers[2]: 3000
numbers[3]: 4000
numbers[4]: 5000
```

المصفوفة متعددة الأبعاد

تتكون من أكثر من صف وأكثر من عمود ويتم الإعلان عنها من خلال اسم المصفوفة ونوعها وتحديد عدد الصفوف والأعمدة

```
Int WZ [ ] [ ] = New int [ 4] [ 5 ] ;
```

المثال (قبل السابق) للإعلان عن مصفوفة بالاسم WZ ذات بعدين من النوع int وقد تم استخدام الأمر new لتحديد عدد الصفوف (٤) وعدد الأعمدة (٥) ، ويتم تسجيل أو التعامل مع أي عنصر بتحديد عدد الصف والعمود

فمثلا لوضع قيمة في العنصر الموجود في الصف الثاني والعمود الثالث نكتب الكود التالي :

```
WZ [2] [3] = 50;
```

كما يمكن الإعلان عن المصفوفة متعددة الأبعاد بوضع القيم فيها مباشرة كما يلي :

```
int[][] a = {{1,2,4,7},{-1,5,8,9}};
```

وللوصول لقيمة خانة في المصفوفة نكتب رقم الصف وقم العمود التي تقع فيهم تلك القيمة

مثال

```
int[][] a= {{3,5,6,9},{8,12,9,14}};
 int w = a[0][3];
```

نلاحظ أن قيمة w=9

3	5	6	9
8	12	9	14

الحلقة foreach

تسمح هذه الحلقة بالمرور على جميع عناصر المصفوفة دون الحاجة لتعريف عداد وتحديد أين يبدأ وأين ينتهي، لا يتم كتابة الكلمة foreach بل نكتب for فقط ، ويظهر الاختلاف بين for ، foreach فقط بين القوسين ( ) كما هو موضح في الصيغة التالية:

```
for( element: array ) {
   // statements
}
```

element : متغير نقوم بتعريفه داخل الحلقة وإعطاؤه نفس نوع المصفوفة التي نضعها بعد النقطتين لأنه في كل دورة سيقوم بتخزين قيمة عنصر من عناصرها لذلك يجب وضع نوعه مثل نوعها.

array : المصفوفة المراد الوصول لجميع عناصرها.

statements : جميع الأوامر الموضوعة في الحلقة وهي تتنفذ في كل دورة.

تقوم الحلقة بالمرور على جميع عناصر المصفوفة بالترتيب من العنصر الأول إلى العنصر الأخير وفي كل دورة تقوم بتخزين قيمة العنصر في المتغير الذي تم تعريفه.

مثال :

برنامج يعرض قيم جميع عناصر مصفوفة باستخدام الحلقة foreach

```
1   public class Main {
2   public static void main(String[] args) {
3   // تعريف المصفوفة
4   int[] numbers = { 1000, 2000, 3000, 4000, 5000 };
5   // element في المتغير numbers في كل دورة سيتم تخزين قيمة عنصر من عناصر المصفوفة
6   for(int element: numbers)
7   {
8   System.out.println(element); // element عرض القيمة التي تم تخزينها في المتغير
9   }
10  }
11  }
```

يظهر ناتج التنفيذ كما يلي :

```
1000
2000
3000
4000
5000
```

دوال جاهزة للتعامل مع المصفوفات في الجافا

الكلاس Arrays هو كلاس جاهز في الجافا يحتوي على دوال نوعها static تطبق على جميع أنواع البيانات البدائية وتستخدم للترتيب، البحث، المقارنة، ولإعطاء قيم لعناصر المصفوفة

١- الدالة equals()

الغرض منها: مقارنة مصفوفتان نمررهم لها مكان الباراميترين (a1 ، a2) ، وترجع true إذا كانتا متساويتين في الحجم والمحتوى ، غير ذلك ترجع false (في حال إذا كانت المصفوفتان لا تحتويان أي قيمة - أي قيمتهما = null - وتم مقارنتهما سيكون الجواب true لأنهما متساويتين في الحجم والمحتوى)

الصيغة:

```
public static boolean equals(Object[] a1, Object[] a2)
```

باراميترات:مكان الباراميترات a1 ، a2 نمرر مصفوفتين من أي نوع.

```
1  import java.util.Arrays;
2  public class Main {
3        public static void main(String[] args) {
4         Object arr1[] = { 0, 1, 2, 3, 4, 5};
5         Object arr2[] = { 0, 1, 2, 3, 4, 5};
6        Object arr3[] = { 0, 1, 2, 3};
7       // سترجع true لأن المصفوفتين متساويتين فى الحجم و الطول
8         System.out.println( Arrays.equals(arr1, arr2) );
9       // سترجع false لأنهما ليستا متساويتين
10        System.out.println( Arrays.equals(arr1, arr3) );
11        }
12 }
```

يظهر ناتج التنفيذ كما يلي :

```
true
false
```

٢-الدالة sort()

الغرض منها: ترتب قيم المصفوفة التي نمررها لها مكان الباراميتر a بشكل تصاعدي ascending(لا ترجع قيمة)

```
1   import java.util.Arrays;
2   public class Main {
3           public static void main(String[] args) {
4           Object arr1[] = { 4, 1, 3, 5, 2 };
5          Object arr2[] = { 'c', 'b', 'a', 'd', 'e' };
6           Arrays.sort(arr1); // ترتيب قيم المصفوفة arr1
7          Arrays.sort(arr2); // ترتيب قيم المصفوفة arr2
8           // عرض قيم المصفوفة arr1
9           System.out.print("arr1: ");
10          for(Object element: arr1) {
11         System.out.print( element + " ");
12        }
13          System.out.println();
14          // عرض قيم المصفوفة arr2
15          System.out.print("arr2: ");
16         for(Object element: arr2) {
17         System.out.print( element + " ");
18         }
19      }
20  }
```

يظهر ناتج التنفيذ كما يلي :

```
arr1: 1 2 3 4 5
arr2: a b c d e
```

٣-الدالة fill()

تستخدم لوضع قيمة أولية في جميع عناصر المصفوفة (تلك الدالة لا ترجع بقيمة)

الصيغة:

public static void fill(Object[] a, Object val)

مكان الباراميتر a نمرر المصفوفة التي يتم وضع قيم أولية لعناصرها ، ومكان الباراميتر val نمرر القيمة التي سيتم وضعها لهم.

```
1  import java.util.Arrays;
2  public class Main {
3      public static void main(String[] args) {
4      Object arr1[] = new Object[5];
5     Object arr2[] = { 1, 2, 3, 4, 5};
6      Arrays.fill(arr1, "java"); // java تبديل القيم null التي كانت موجودة بالقيمة
7      Arrays.fill(arr2, 100); // 100 تبديل جميع الأرقام التي تم وضعها في المصفوفة بالقيمة
8     // arr1 عرض قيم المصفوفة
9       System.out.print("arr1: ");
10      for(Object element: arr1) {
11      System.out.print( element + " ");
12      }
13      System.out.println();
14     // arr2 عرض قيم المصفوفة
15      System.out.print("arr2: ");
16     for(Object element: arr2) {
17      System.out.print( element + " ");
18          }
19       }
20 }
```

يظهر ناتج التنفيذ كما يلي :

```
arr1: java java java java java
arr2: 100 100 100 100 100
```

٤-الدالة arraycopy()

تستخدم لنسخ محتوى مصفوفة ووضعه في مصفوفة أخرى.
عند استدعاء هذه الدالة يجب تحديد ما يلى:

- المصفوفة المراد نسخ قيم منها (source)
- العنصر المراد أن تبدأ عملية النسخ من عنده (starting position in the source)
- المصفوفة المراد نسخ القيم اليها (destination)
- العنصر المراد أن تبدأ عملية اللصق عنده(starting position in the destination)
- عدد العناصر المراد نسخها من المصفوفة الأولى إلى المصفوفة الثانية (length)

الصيغة Syntax

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

الباراميترات

- src : نمرر المصفوفة التي سيتم نسخ قيم منها.
- srcPos : نمرر رقم أول عنصر index في المصفوفة src تبدأ عملية النسخ من عنده.
- dest نمرر المصفوفة التي ستتخزن فيها القيم التي تم نسخها.

destPos : نمرر رقم index أول عنصر في المصفوفة dest تبدأ عملية النسخ عنده.
length : نمرر عدد العناصر التي سيتم نسخها من المصفوفة الأولى إلى المصفوفة الثانية.

مثال

```
1  import java.util.Arrays;
2   public class Main {
3      public static void main(String[] args) {
4       int arr1[] = { 10, 20, 30, 40, 50, 60 };
5       int arr2[] = new int [6];
6       // نسخ جميع قيم المصفوفة الأولى ووضعها في المصفوفة الثانية
7       System.arraycopy(arr1, 0, arr2, 0, 6);
8       // عرض جميع قيم عناصر المصفوفة الثانية
9       System.out.print( "arr2 = " );
10      for(int i=0; i<arr2.length; i++)
11      {
12      System.out.print( arr2[i] + ", " );
13              }
14       }
   }
```

يظهر ناتج التنفيذ كما يلي :

```
arr2 = 10, 20, 30, 40, 50, 60,
```