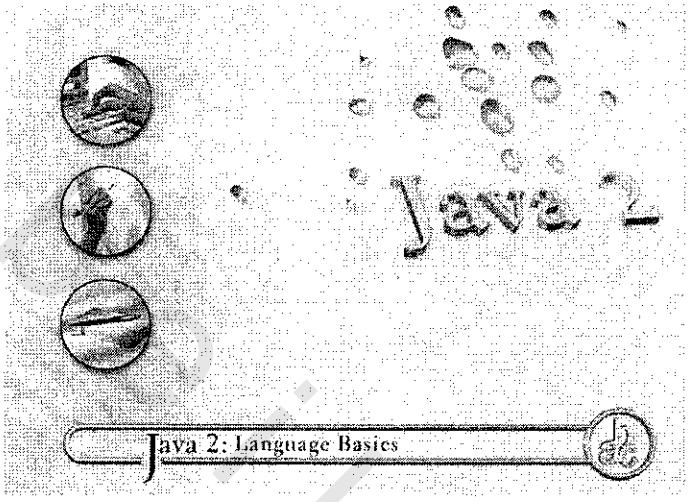
الفصل الخامس



المصفوفات Arrays

فى هذا الفصل نشرح موضوعاً لا تخلو منه لغة برمجة وهو المصفوفات Arrays وذلك من خلال النقاط التالية:

- **معنى المصفوفة Array.**
- **الإعلان عن متغير مصفوفة Declaring Array Variable.**
- **تعريف هدف من المصفوفة Creating Array Object.**
- **تعريف عنصر المصفوفة Array وإعطاؤها قيم ابتدائية Initial Values.**
- **التعامل مع عناصر المصفوفة Accessing Array Elements.**
- **استعمال أوامر التكرار for مع المصفوفة Array.**
- **نسخ المصفوفات Arrays وطول المصفوفة Array.**
- **إيجاد أكبر قيمة والبحث عن قيمة داخل المصفوفة Array.**
- **ترتيب عناصر المصفوفة Sorting Array.**
- **المصفوفات متعددة الأبعاد Multi-Dimensional Array.**
- **الإعلان عن مصفوفة متعددة الأبعاد Multi-Dimensional Array.**
- **هل يمكن تغيير طول المصفوفة Array؟**

معنى المصفوفة Array:

- المصفوفة Array هى مجموعة من العناصر من نفس النوع ، فيمكن إنشاء مصفوفة Array أرقام أو مصفوفة أسماء (حرفيات) أو أي نوع ، المهم أنه يوجد مجموعة من العناصر من نفس النوع ، ولا يمكن إنشاء مصفوفة Array من أكثر من نوع مثل الأرقام والحرفيات (integers , String) فى نفس المصفوفة Array.
- هناك كثير من العمليات يصعب تحقيقها بدون استعمال المصفوفات Arrays مثل ترتيب (فرز) Sort مجموعة من القيم تصاعدياً أو تنازلياً ، حيث أنه من الأسهل وضع هذه القيم فى مصفوفة Array ثم القيام بترتيب العناصر فى المصفوفة Array.
- ولإنشاء مصفوفة Array نقوم بعمل ثلاثة خطوات هى:

1- الإعلان عن متغير المصفوفة Array Variables.

2- تعريف (إنشاء) عنصر المصفوفة Array object.

3- تخزين القيم أو البيانات داخل المصفوفة والتعامل معها.

الإعلان عن متغير مصفوفة Declaring Array Variable:

ويتم ذلك بطريقتين هما:

الطريقة الأولى:

```
String ourteam [ ];
int codes [ ];
```

فى هذه الطريقة يتم كتابة نوع عناصر المصفوفة Array فى البداية (String , int) ثم كتابة اسم المصفوفة (our team, codes) ثم وضع الأقواس [ ] ، والفرق بين المصفوفة Array والمتغير العادى هو الأقواس [ ] التى تدل على أن المتغير هو مصفوفة Array.

الطريقة الثانية:

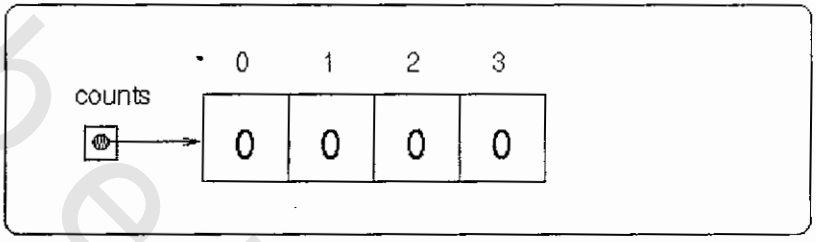
```
String [ ] ourteam;
int [ ] codes;
```

فى هذه الطريقة يتم كتابة نوع المصفوفة Array ثم الأقواس بالشكل String [ ] ثم يليهم اسم المصفوفة Array وهى ourteam كما هو واضح فى السطر الأول.

بالمثل في السطر الثانى نجد نوع المصفوفة Array ثم الأقواس بالشكل int [ ] ثم يليهم اسم المصفوفة وهو codes كما هو واضح فى السطر الثاني.

وفى الحالتين تم الإعلان عن متغير من نوع مصفوفة Array.

والشكل التالى يوضح كيف تمثل المصفوفة Array فى الذاكرة كما يوجد فى الشكل التالى.



تعريف هدف من المصفوفة Creating Array Object:

بعد الإعلان عن متغير مصفوفة Array ، يتم تعريف (إنشاء) هدف object من المصفوفة Array بطريقتين:

- استعمال كلمة new لإنشاء الهدف object.
- تعريف هدف object من المصفوفة Array مع إعطائها قيم ابتدائية Initial Values بدون استعمال new.

استعمال new:

يتم استعمال new لتعريف عنصر object كما يلى:

```
String []jobs = new String [10];
```

وفى هذا السطر يتم تعريف متغير بالاسم jobs من نوع مصفوفة حرفيات Array of String وعدد العناصر 10 ، وبالطبع يمكن كتابة هذا السطر على سطرين كما يلى:

```
String [ ] jobs;
jobs = new string [10];
```

وعند استعمال new مع المصفوفة Array لابد من تحديد عدد العناصر مثل 10.

تعريف عنصر المصفوفة Array وإعطاؤها قيم ابتدائية Initial Values:

يمكن تعريف عنصر مصفوفة Array بدون استعمال new وذلك بإعطاء المصفوفة Array قيم ابتدائية Initial Values ، وبالتالى لا نحتاج لتحديد عدد العناصر لأننا

نضع العناصر نفسها كما يلى:

```
String [ ] jobs = {"Eng", "Doc", "Tech"};
```

فى هذا السطر تم تعريف مصفوفة حرفيات Array of String بالاسم jobs ثم تم إعطاؤها قيم ابتدائية Initial Values ، وفى هذه الحالة لا نحتاج لتحديد عدد العناصر لأن العناصر موجودة بالفعل.

فى حالة استعمال new ، يتم تسجيل قيم افتراضية لعناصر المصفوفة Array حسب نوعها ، فمثلاً مصفوفة القيم الرقمية تأخذ القيم 0 ومصفوفة القيم المنطقية boolean تأخذ القيم false ومصفوفة العناصر Objects تأخذ القيم null.

التعامل مع عناصر المصفوفة Accessing Array Elements:

يتم التعامل مع عناصر المصفوفة Array بكتابة رقم عنصر المصفوفة Array بين أقواس [ ] كما هو واضح في السطر التالي:

```
arrayName [ index ]
```

وفى هذه الصورة نجد أن arrayName يمثل اسم المصفوفة Array وindex يمثل ترتيب العنصر فى المصفوفة Array مع ملاحظة أن ترتيب عناصر المصفوفة Array يبدأ من القيمة 0 ، وتقوم لغة Java باختبار قيمة ترتيب العنصر قبل التعامل معه للتأكد من وجود ترتيب هذا العنصر ضمن عناصر المصفوفة Array وليس خارجها ، ولا يمكن تسجيل قيمة فى عنصر غير موجود ضمن عناصر المصفوفة Array (كما يحدث فى لغة C/C++).

ولتوضيح ذلك تابع المثال التالى:

```
String [ ] myArray = new String [20];
myArray [20] = "Java lang";
```

فى السطر الأول يتم الإعلان عن مصفوفة Array بالاسم myArray من نوع حرفيات String وحجز 20 عنصر باستعمال new.

فى السطر الثانى يتم تسجيل العبارة Java lang فى المصفوفة myArray فى العنصر رقم 20 ، وهذا السطر يعطى خطأ عند تنفيذ البرنامج وذلك لأن هذه المصفوفة Array

تبدأ من العنصر رقم 0 وتنتهى عند العنصر 19 وبالتالي يكون عدد العناصر 20 عنصراً لذلك لا يوجد عنصر رقمه 20 ، وهذا الخطأ من أكثر الأخطاء الشائعة عند التعامل مع المصفوفات Arrays.

ولمعرفة عدد عناصر المصفوفة Array يمكن استعمال المتغير length كما فى السطر التالى:

```
int L= myArray.length;
```

فى هذا السطر يتم حساب عدد عناصر المصفوفة Array باستعمال المتغير length ثم يتم حفظ هذا العدد فى المتغير L.

مثال (1): إنشاء المصفوفة Array:

فى هذا المثال نحاول تجميع النقاط السابقة عن المصفوفة Array وتوضيحها في البرنامج الموضح كما فى الشكل (5-1).

```
1  public class Myfriends {
2
3      String [ ] Names = { "khaled", "Alaa", "Amr", "Omr"};
4  void printNames ( )
5  {
6    int I = 0;
7    System.out.println (Names [I]);
8    I++;
9    System.out.println (Names [I]);
10   I++;
11 System.out.println (Names [I]);
12   I++;
13   System.out.println (Names [I]);
14   }
15   public static void main (String args [ ])
16   {
17
18   Myfriends FD= new Myfriends ( );
19   FD. printNames ();
20   }
21   }
```

(الشكل 5-1)

شرح السطور:

- فى السطر رقم 1 يتم تعريف فصيلة البرنامج (Class) بالاسم Myfriends.
- فى السطر رقم 3 يتم تعريف مصفوفة حرفيات (Array of String) بالاسم Names

مع إعطائها قيم ابتدائية Initial Values عبارة عن مجموعة أسماء.

- فى السطر رقم 4 يتم تعريف دالة Method بالاسم printNames( ).
- فى السطر رقم 6 يتم الإعلان عن متغير I بقيمة ابتدائية Initial Value تساوي 0.
- فى السطر رقم 7 يتم طباعة عنصر المصفوفة Array رقم I أى العنصر رقم 0 الذي يحمل الاسم Khaled.
- فى السطر رقم 8 يتم زيادة قيمة المتغير I بمقدار 1 لتصبح I تساوى 1.
- فى السطر رقم 9 يتم طباعة عنصر المصفوفة Array رقم I أي العنصر رقم 1 الذي يحمل الاسم Alaa ، وبالمثل باقي السطور حتى السطر رقم 13.
- فى السطر رقم 14 تنتهى الدالة printNames( ) التى تقوم بطباعة الأسماء.
- فى السطر رقم 15 تبدأ الدالة الرئيسية main( ) التى يبدأ التنفيذ منها.
- فى السطر رقم 18 يتم تعريف هدف object بالاسم FD من نوع الفصيلة Myfriends.
- فى السطر 19 يتم استدعاء الدالة printNames( ) ، وبالتالى تقوم الدالة Method بطباعة الأسماء الموجودة داخل المصفوفة Names وتحصل على قائمة بالأسماء كما فى نتيجة التنفيذ.
- اكتب هذا البرنامج ونفذه تحصل على قائمة بالأسماء التى تم حفظها فى المصفوفة Array.

التعامل مع عنصر المصفوفة Array:

فى المثال السابق تم طباعة عناصر المصفوفة Array داخل الدالة printNames() عنصر بعنصر ، بينما لا يناسب ذلك الحالات العملية ، فلو فرضنا أن عدد عناصر المصفوفة كان 100 عنصراً ، فستضطر لكتابة 100 سطر لطباعة العناصر ، ولذلك يكون الحل الأمثل هو استعمال جمل التكرار Loops كما يلى فى النقاط القادمة.

استعمال أوامر التكرار for مع المصفوفة Array:

لاحظنا فى المثال السابق كيف تم التعامل مع عناصر المصفوفة Array عن طريق تحديد ترتيب عنصر المصفوفة Array كما فى السطر التالي:

Names[3]

أى العنصر رقم 4 ، ولكن أشرنا إلى أنه يصعب أو يستحيل استعمال هـذه الطريقـة مـع المصفوفات Arrays الكبيرة ، ولا بد من استعمال جملة التكرار for بدلاً من ذلك كمـا يتضح من المثال التالي.

مثال (2): استعمال أوامر التكرار for مع المصفوفة Array:

قم بتعديل البرنامج السابق ليصبح كما فى الشكل (5-2).

```
1  public class Myfriends {
2
3      String [ ] Names = { "khaled", "Alaa", "Amr", "Omr"};
4  void printNames ( )
5  {
6     int i ;
7     for (i=0;i<4;i++)
8     System.out.println (Names [i]);
9       }
10    public static void main (String args [ ])
11    {
12
13    Myfriends FD= new Myfriends ( );
14    FD. printNames ();
15    }
16    }
```

(الشكل 5-2)

شرح السطور:

- الجديد فى هذه السطور عن برنـامج المصـفوفات Arrays التقليـدى هـو اسـتعمال التكرار for Loop.
- فى السطر رقم 7 يتم التكرار من 0 حتى أقل من أربعة وهو عدد عناصر المصفوفة Array.
- فى السطر رقم 8 يتم طباعة عناصر المصفوفة عـن طريـق Names[i] وبالتـالى يـتم طباعة الأسماء ، وذلك يوفر إعادة كتابة أمر الطباعة وذلـك باسـتعمال تكـراره مـع for بعدد عناصر المصفوفة Array.
- وعند تنفيذ البرنامج تحصل على نفس النتيجة السابقة.

نسخ المصفوفات Arrays:

يمكن استعمال متغير مصـفوفة Array للإشـارة إلى مصـفوفة Array أخـرى كمـا فـى السطور التالية:

```
double[] a = new double[3];
double[] b = a;
```

فى هذه السطور يتم الإعلان عن مصفوفة a ثم الإعلان عن مصفوفة b واستخدام متغيرها للإشارة إلى نفس المصفوفة a .

طول المصفوفة Array:

توفر المصفوفات Arrays المتغير length لمعرفة طول المصفوفة Array ، ويتم استعماله كما فى السطور التالية:

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

فى هذه السطور يتم الاستفادة من الخاصية length الموجودة فى المصفوفة Array لتحديد طولها أى عدد عناصرها بدلاً من تحديد هذا الطول برقم ، ثم يتم القيام بنسخ عناصر المصفوفة a إلى المصفوفة b.

إيجاد أكبر قيمة:

يمكن استعمال المصفوفات Arrays لإيجاد أكبر قيمة من مجموعة قيم وذلك كما فى السطور التالية:

```
double max = A[0];
for (int i = 1; i < A.length; i++)
{
        if (A[i] > max)
            max = A[i];
}
```

فى هذه السطور يتم وضع قيمة أول عنصر من المصفوفة Array فى المتغير max بافتراض أنه أكبر عنصر ، ثم يتم استعمال جملة التكرار for فى المرور على جميع عناصر المصفوفة Array ، وفى كل مرة يتم مقارنة قيمة المتغير max بعنصر المصفوفة Array الحالى ، فإذا كانت القيمة الموجودة فى المصفوفة Array أكبر من قيمة المتغير max ، فإنه

يتم مساواة قيمة المتغير max بالقيمة الموجودة في المصفوفة Array بحيث ينتهى التكرار وقد تم وضع قيمة أكبر عنصر فى المتغير max.

البحث عن قيمة داخل المصفوفة Array:

من العمليات المشهورة للمصفوفات Arrays القيام بالبحث عن قيمة داخل مجموعة قيم ، والسطور التالية توضح ذلك.

```
static int find(int[] A, int N)
{
        for (int index = 0; index < A.length; index++)
         {
            if ( A[index] == N )
               return index;  // N has been found at this index!
         }
         return -1;
}
```

فى هذه السطور يتم إنشاء دالة Method تستقبل اسم المصفوفة Array والقيمة المطلوب البحث عنها ، ثم يتم استعمال جملة التكرار for للمرور على العناصر ومقارنتها بالقيمة المطلوب البحث عنها ، فإذا وجدت يتم إعادة ترتيب هذه القيمة فى المصفوفة Array وإلا يتم إعادة القيمة 1-.

ترتيب عناصر المصفوفة Sorting Array:

من العمليات المشهورة أيضاً ، استعمال المصفوفات فى ترتيب مجموعة قيم ، ولتوضيح ذلك تابع السطور التالية:

```
static void selectionSort(int[] A)
{
    for (int lastPlace = A.length-1; lastPlace > 0; lastPlace--)
    {
       position lastPlace;
       int maxLoc = 0;  // Location of largest item seen so far.
```

```
        for (int j = 1; j <= lastPlace; j++)
        {
           if (A[j] > A[maxLoc])
           {
              maxLoc = j;
           }
        }
        // Swap largest item with A[lastPlace]
        int temp = A[maxLoc];
        A[maxLoc] = A[lastPlace];
        A[lastPlace] = temp;
     } // end of for loop
}
```

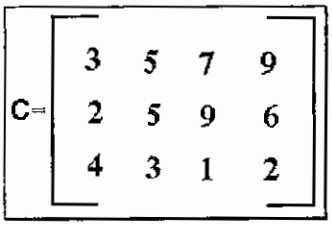
المصفوفات متعددة الأبعاد Multi-Dimensional Array:

فى الفقرات السابقة تم شرح مصفوفة البعد الواحد One-Dimensional Array والتى تتكون من صف واحد أو عمود واحد ، وتأخذ الشكل التالى:

$$A = \begin{bmatrix} 3 \\ 6 \\ 5 \\ 7 \\ 9 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 6 & 5 & 7 & 9 \end{bmatrix}$$

فى الشكل توجد المصفوفة A التى تتكون من 5 صفوف وعمود واحد والمصفوفة B التى تتكون من 5 أعمدة وصف واحد ، وكما أشرنا يتم التعامل مع عناصر المصفوفة Array باسم المصفوفة Array وترتيب العنصر ، فمثلاً العنصر الأول الموجود فى المصفوفة A - الذي قيمته (3)- يتم الإشارة إليه بالشكل A[0] ، وهكذا. كل ذلك بالنسبة لمصفوفات ذات اتجاه أو بعد واحد One-Dimensional.

ولكن هناك نوع أخر من المصفوفات Array الذى يسمى متعدد الأبعاد Multi-Dimensional ويتكون من أكثر من صف وأكثر من عمود كما بالشكل التالى:



في هذا الشكل توجد مصفوفة Array بالاسم C عبارة عن 3 صفوف و4 أعمدة ، لذلك تسمى مصفوفة 4*3.

ويتم الإشارة أو التعامل مع أى عنصر بتحديد رقم الصف ورقم العمود الذى يقع فيه هذا العنصر ، فمثلاً العنصر الأول فى المصفوفة – الذي قيمته 3- يشار إليه كما يلى:

```
C [0] [0]
```

وذلك لأنه يقع فى الصف رقم 0 والعمود رقم 0 ، فى حين أن العنصر الثانى فى نفس الصف الأول يشار إليه بالصيغة:

```
C [0] [1]
```

لأنه يقع فى الصف 0 فى العمود 1 ، وهكذا بالنسبة لباقى العناصر.

الإعلان عن مصفوفة متعددة الأبعاد Multi-Dimensional Array:

ويتم ذلك باسم المصفوفة Array ونوعها وتحديد عدد الصفوف وعدد الأعمدة كما يلى:

```
int XY [ ] [ ] = new int [5] [6];
```

فى هذا السطر تم الإعلان عن مصفوفة Array بالاسم XY ذات بعدين ، وهي من النوع int ، ثم تم استعمال new لتحديد عدد الصفوف بالعدد 5 والأعمدة بالعدد 6.

ويتم تسجيل أو التعامل مع أى عنصر بتحديد الصف والعمود ، فمثلاً لوضع قيمة فى العنصر الموجود فى الصف الثانى الذي ترتيبه (1) والعمود الثالث الذي ترتيبه (4) نكتب السطر التالى: (تذكر أن أول صف وأول عمود ترتيبه صفر)

```
XY [1] [4] = 90;
```

وللتعامل مع المصفوفة ذات البعدين Dimensional Array-2 ، تستعمل اثنتين من جمل التكرار المتداخلة باستعمال nested for loop.

مثال (3): التكرار المتداخل مع المصفوفات Nested Loops with Arrays:

فى هذا المثال نوضح كيفية استعمال التكرار المتداخل Nested Loops مع المصفوفات Arrays وكيفية التعامل مع عناصر المصفوفات ثنائية الأبعاد Dimensional Arrays-2.

اكتب سطور هذا البرنامج كما فى الشكل (5-3).

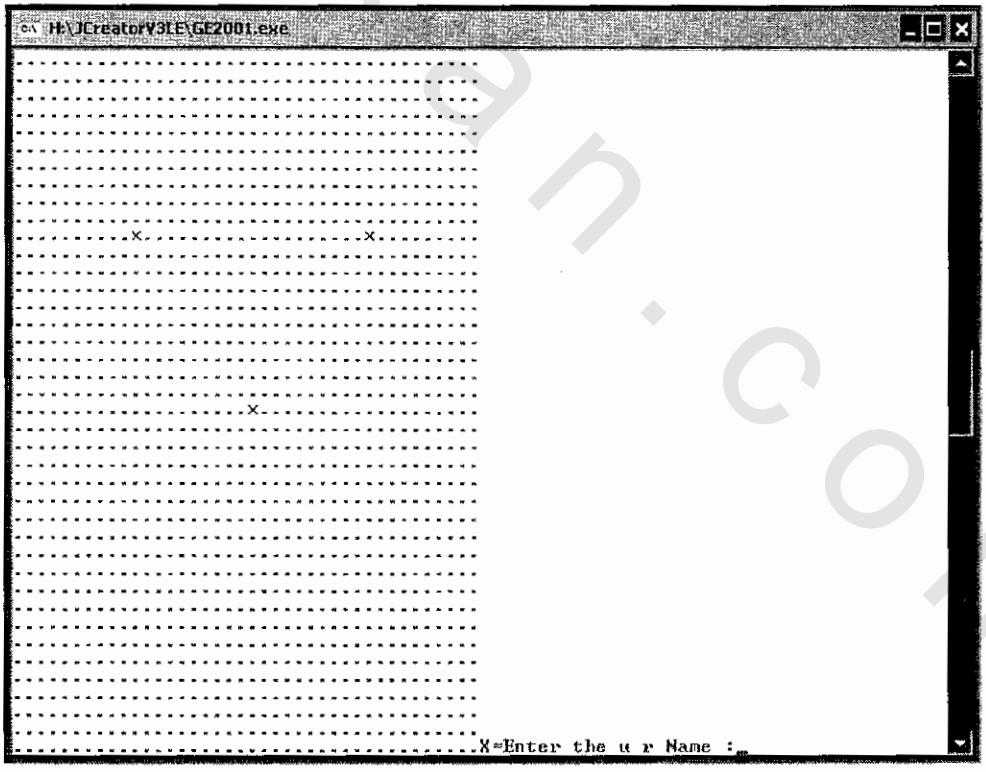
```
1  import java.io.*;
2  public class Myfriends {
3
4      public static void main (String args [ ]) throws IOException
5
6      {
7         int x,y;
8          InputStreamReader inStream = new InputStreamReader( System.in ) ;
9        BufferedReader stdin = new BufferedReader( inStream );
10
11       String inData;
12   char  [ ][ ]  arr=new char [40][40];
13     int r=0,c=0;
14
15     for (r=0;r<40;r++)
16     for (c=0;c<40;c++)
17     arr[r][c]='.';
18   do
19   {
20
21   for (r=0;r<40;r++)
22   {
23     System.out.println ("");
24
25     for (c=0;c<40;c++)
26     System.out.print (arr[r][c]);
27     }
28        System.out.print ("X=");
29        System.out.print("Enter the u r Name :");
30     inData = stdin.readLine();
31     x = Integer.parseInt( inData );
32
33        System.out.print(" Y=");
34         inData = stdin.readLine();
35       y = Integer.parseInt( inData );
36      arr[x][y]='x';
37    }while (x>0);
38    }
39 }
```

(الشكل 5-3)

شرح السطور:

- فى السطر رقم 12 تم الإعلان عن مصفوفة Array من نوع حروف وعدد صفوفها وأعمدتها 40.
- فى السطور 15 و16 و 17 يتم ملء عناصر هذه المصفوفة Array بالحرف (.) وذلك باستخدام التكرار المتداخل Nested Loop مع المصفوفة Array.

- فى السطر رقم 18 يبدأ تكرار do.
- فى السطور من 21 الى 27 يتم طباعة عناصر المصفوفة Array وهى طباعة 40×40 من النقط مما يعطى شكل ورقة رسم بيانى.
- فى السطر رقم 31 يتم استقبال قيمة المتغير x.
- فى السطر رقم 35 يتم استقبال قيمة المتغير y.
- فى السطر رقم 36 يتم وضع الحرف x فى عنصر المصفوفة Arrayالمحدد بالصف x والعمود y ، ثم يعاد طباعة المصفوفة Array بالشكل الجديد ، مما يؤدى إلى طباعة ورقة رسم بيانى عليها هذه النقطة ، ويستمر ذلك حتي يقوم المستخدم بإدخال قيمة سالبة.
- نفذ البرنامج لتحصل على برنامج رسم بيانى يستقبل منك النقطة (r,c) المطلوب توقيعها علي الرسم البياني ، ثم يقوم البرنامج برسم علامة x فيها وهكذا حتى تدخل قيمة سالبة كما فى الشكل (5-4).



(الشكل 5-4)

مثال (4): المصفوفة متعددة الأبعاد Multi-Dimensional Array:

فى هذا المثال يتم توضيح كيفية التعامل مع مصفوفة متعددة الأبعاد Multi-Dimensional Arrays.

اكتب البرنامج كما فى الشكل (5-5).

```
1  public class MultiArray {
2    final static int ROWS = 10;
3    final static int COLS = 5;
4    public static void main(String[] args) {
5      int rCount;
6      int cCount;
7      int totalSize;
8      byte[][] twodimarray= new byte[ROWS][COLS];
9      rCount = twodimarray.length;
10     cCount = twodimarray[COLS].length;
11     totalSize = rCount * cCount;
12     System.out.println("No of Rows is :    " + rCount);
13     System.out.println("No of cols is : " + cCount);
14     System.out.println("Total size:       " + totalSize);
15     byte[][] Array2 = new byte[5][];
16     Array2[0] = new byte[2];
17     Array2[1] = new byte[2];
18     Array2[2] = new byte[4];
19     Array2[3] = new byte[8];
20     Array2[4] = new byte[3];
21     byte[][] smallArray = { { 01, 02, 03, 04 }, { 20, 21, 22, 23 },
22       { 30, 31, 32, 33 }, { 40, 41, 42, 43 }, };
23     System.out.println(smallArray[1][2]);
24   }
25 }
```

(الشكل 5-5)

شرح السطور:

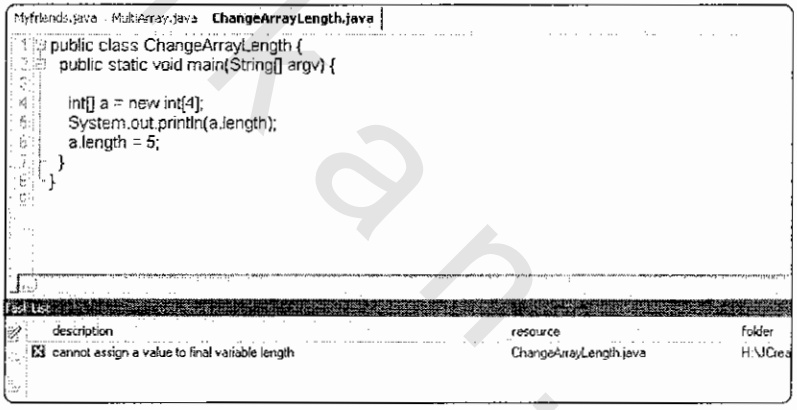
- فى السطر رقم 8 تم الإعلان عن مصفوفة ذات بعدين 2-Dimensional Array بعدد 10 صفوف و8 أعمدة.
- فى السطر رقم 9 تم إيجاد طول المصفوفة Array باستعمال الخاصية length ثم تم طباعة النتيجة ثم تم وضع بعض القيم فى بعض العناصر كما فى السطور من 16 إلى 20.
- فى السطر رقم 21 تم تعريف مصفوفة Array مع إعطائها قيم ابتدائية Initial Values ثم تم طباعة أحد هذه العناصر.

- بالطبع لا يؤدى البرنامج وظيفة قوية ، فهو فقط لتوضيح بعض عمليات المصفوفات Arrays.
- نفذ البرنامج لتحصل على النتيجة كما فى الشكل (5-6).

(الشكل 5-6)

هل يمكن تغيير طول المصفوفة Array؟

لمعرفة ذلك اكتب البرنامج التالى ونفذه كما فى الشكل (5-7).



(الشكل 5-7)

فى السطر رقم 6 نحاول تغيير طول المصفوفة Array ولكن يعطى المترجم Compiler رسالة خطأ دلالة علي عدم إمكانية تغيير طول المصفوفة Array.

ملخص الفصل:

- تعرضنا في هذا الفصل لشرح المصفوفات Arrays.
- في الفصل القادم سوف نتعرف - بإذن الله - علي أحد أهم مبادئ البرمجة موجهة الهدف Object-Oriented Programming وهو الفصائل Classes والأهداف Objects ، فتابع معنا الفصل القادم.