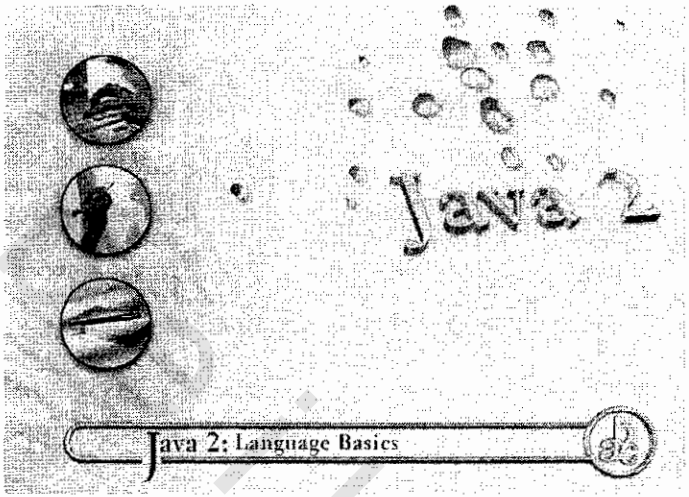
الفصل الرابع



Methods أول برنامج

فى هذا الفصل سوف نتناول موضوع الدوال Methods وذلك من خلال النقاط التالية:

- موجز سريع عن تاريخ البرمجة.
- لغات البرمجة.
- البرمجة التركيبية Structured Programming.
- ما هى الدوال Methods.
- لماذا نحتاج إلى الدوال Methods.
- أنواع الدوال Methods.
- دوال من غير قيم مرتجعة Methods with No Returning Values.
- الدالة الرئيسية ()main.
- استدعاء الدالة لنفسها Recursion.
- شرط إنهاء استدعاء الدالة Base Case.

موجز سريع عن تاريخ البرمجة:

- فى بداية عصر تصنيع الكمبيوتر كانت تتم عملية البرمجة عن طريق مجموعة كبيرة من المفاتيح Switches ، وكان كل مفتاح يمثل معلومة صغيرة ، وكان القائمون على عملية البرمجة لا بد وأن يكونوا من العلماء والفنيين فى الإلكترونيات والرياضيات ، وكانت برمجة برنامج صغيرة عملية شاقة ومجهدة.
- ولكن مع تطور الكمبيوتر وانتشاره ، زادت الحاجة إلى برامج أكبر ولغات أكثر مرونة ، ومن هنا زادت العملية تعقيداً.

لغات البرمجة:

- ومن أوائل اللغات التى ظهرت كانت لغة الأسمبلى (التجميع) Assembly ، ولكنها كانت لغة صعبة للغاية وكان من يتعامل معها لابد وأن يكون من الفنيين.
- ثم تلى ظهور لغة Assembly لغات تستخدم مفردات اللغة الإنجليزية مثل لغة فورتران Fortran التي تنتمى إلى لغات المستوى العالى ، وأدى هذا إلى تصنيف لغات البرمجة إلى فئتين هما:

1- لغات المستوى المنخفض Low Level Languages مثل الأسمبلى Assembly.

2- لغات المستوى العالى High Level Languages مثل الفورتران Fortran ولغة البيسك Basic الشهيرة وهى أصل لغة Visual Basic ، وظهر أيضاً كثير من اللغات الأخرى.

- ومع ظهور لغات المستوى العالى Level Languages High ، لم يصبح العلماء والفنيين فقط هم الذين يقومون بالبرمجة فقط ، بل أصبح الكثير من الناس يستطيعون أن يقوموا بتلك العملية لأن تلك اللغات تتعامل بمفردات اللغة الإنجليزية المفهومة.
- وأدى تطور لغات البرمجة إلى ظهور الكثير من المفاهيم ، وكان من أهم تلك المفاهيم هو البرمجة التركيبية Structured Programming.

البرمجة التركيبية Structured Programming:

- قبل ظهور البرمجة التركيبية Structured Programming ، كان المبرمجون يقومون بكتابة البرنامج كله قطعة واحدة ، وكان ذلك يؤدى إلى صعوبات كثيرة فى التعامل مع الكود ويؤدى أيضاً إلى صعوبة فهم الكود والأخطاء الكثيرة.
- ولكن البرمجة التركيبية Structured Programming غيرت ذلك حيث يقوم المبرمج بتقسيم البرنامج إلى أجزاء (مقاطع) ويعطى لكل جزء اسم خاص به ثم يقوم بعد ذلك باستدعاء اسم ذلك الجزء (فقط ليقوم البرنامج بتنفيذ ذلك الجزء) فى الدالة الرئيسية ( )main بحيث يقوم كل جزء بأداء مهمة محددة.

كما ذكرنا من قبل ، يتم تقسيم البرنامج إلى دالة رئيسية main( ) method ودوال methods فرعية ويتم استدعاء الدوال methods الفرعية داخل الدالة الرئيسية ( )main ، وعملية الاستدعاء تسمى Calling

- ولابد أن تكون تلك الدوال Methods مرتبة فى ترتيب منطقى مترابط.
- إذن نستطيع أن نخرج من المناقشة السابقة بأن البرنامج فى البرمجة التركيبية Structured Programming هو مجموعة من الدوال Methods ، والدالة Method تتكون من مجموعة من البيانات والأوامر.
- ثم مع تطور حركة البرمجة وزيادة التعقيد فى البرامج ، كان من اللازم تطوير مفهوم البرمجة التركيبية Structured Programming ، وجاء ذلك التطوير فى شكل جديد وهو البرمجة بالأهداف Object Oriented Programming أو اختصاراً OOP ، وتفيدنا البرمجة بالأهداف OOP فى أننا نستطيع تمثيل الأشياء المحيطة بنا تمثيل حقيقي وهو ما سيتضح خلال الفقرة القادمة.

ما هى الدوال Methods:

- الدالة Method هى مجموعة من سطور البرمجة التى تتحد معاً لتحقيق وظيفة معينة ، ويمكنك أن تقول أنها برنامج فرعى يؤدى وظيفة محددة.
- وهى تتصف بما يلى:
  1. برنامج فرعى يمكن استدعائه Call أكثر من مرة.
  2. هى بلوك Block من الكود لتحقيق وظيفة محددة.
  3. وهى غالباً ما تحتوى على مدخلات Inputs ومخرجاتOutputs (ولكن ليس بالضرورى).
  4. أحيانا تسمى Subroutines or functions.

لماذا نحتاج إلى الدوال Methods:

(1) جعل البرنامج مركب Modularity:

بالطبع يصبح البرنامج كبير ويحتوى على عدد كبير من السطور ، وبالتالى من الأفضل تقسيم هذا البرنامج إلى أجزاء صغيرة واستدعائها Call عند الحاجة ، وبالتالى يتم تقسيم وظائف البرنامج إلى دوال Methods.

(2) إعادة الاستخدام Code Reuse:

من أشهر قواعد البرمجة التى حققتها الدوال Methods نظرية كتابة الكود مرة واحدة ثم إعادة استخدامه أكثر من مرة ، وهى من القواعد المشهورة والمهمة جداً ، وهو اتجاه تطوير البرمجة من أول إنشائها وحتى الآن وفى المستقبل ، ومن أول ما حقق هذا المفهوم وجود الدوال Methods.

(3) إخفاء الأوامر Encapsulation:

من فوائد الدوال Methods هو إنشاء أجزاء تحقق وظائف معينة ، وعند الاحتياج إلى وظائف هذه الدوال Methods ، يتم استعمالها دون الرجوع إلى سطور هذه الدوال Methods ، وسوف تتضح هذه النقطة أكثر مع درس Objects and Classes.

مثال (1): الدوال Methods:

فى هذا المثال يتم إنشاء دالة Method بسيطة لتوضيح فقط كيفية إنشاء واستدعاء Call الدالة Method ، ولتحقيق ذلك اكتب السطور كما فى الشكل (4-1).

```
1  public class Method1 {
2      public void Hello()
3      {
4          System.out.println("Hello from Method 1 ");
5      }
6      public static void main(String[] args)
7      {
8      Method1 Obj1=new Method1();
9      Obj1.Hello();
10     }
11 }
12
```

(الشكل 4-1)

شرح السطور:

- فى السطر رقم 2 تم إنشاء دالة Method جديدة بالاسم Hello().
- فى السطر رقم 4 وداخل هذه الدالة Method يتم طباعة العبارة Hello from Method 1.
- فى السطر رقم 8 يتم تعريف هدف Object من الفصيلة Class ، وهذا ما سوف نتناوله فى فصول البرمجة باستعمال الأهداف OOP.
- فى السطر رقم 9 يتم استدعاء الدالة الجديدة Hello().
- نفذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (4-2).

```
H:\JCreatorV3LE\GE2001.exe
Hello from Method 1
Press any key to continue...
```

(الشكل 4-2)

- فى هذا الشكل تم استدعاء الدالة Method وطباعة العبارة المحددة.

أنواع الدوال Methods:

- بعد إنشاء دالة Method بسيطة وتناول كيفية الإنشاء وكيفية الاستدعاء Call ، تعال معنا نستكمل معاً باقى عناصر هذا الموضوع وذلك بتناول أنواع الدوال Methods.
- يتم تحديد نوع الدالة Method بنوع القيمة المرتجعة Return Type من الدالة Method ، فإذا كانت الدالة Method تقوم بإرجاع قيمة من نوع int ، تصبح الدالة Method من النوع int ، وإذا كانت الدالة تقوم بإرجاع قيمة من النوع char ، تصبح الدالة Method من النوع char وهكذا ، ولتوضيح ذلك تابع معنا الامثلة التالية.

مثال (2): إرجاع قيمة من دالة Returning a Value from Method:

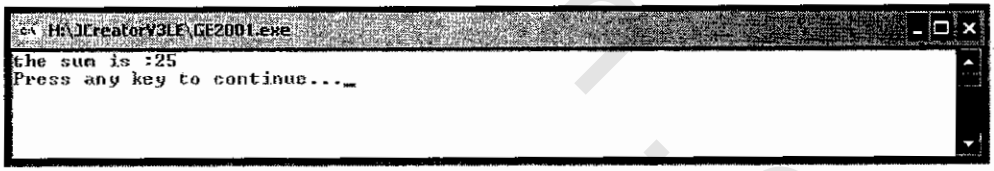
المثال التالى يوضح كيفية إنشاء دالة Method من النوع int ، ولتوضيح ذلك اكتب المثال كما فى الشكل (4-3).

```
1  public class Method1 {
2      public int Sum(int a ,int b )
3      {
4  int s;
5  s=a+b;
6  return s;
7          }
8      public static void main(String[] args)
9      {
10       int r;
11     Method1 Obj1=new Method1();
12     r=Obj1.Sum(10,15);
13
14       System.out.println("the sum is :"+r);
15     }
16 }
```

(الشكل 4-3)

شرح السطور:

- فى السطر رقم 2 يتم إنشاء الدالة Sum() من النوع int ، لأنها تقوم بإرجاع قيمة من نوع int ولها معاملان Parameters من النوع int.
- فى السطر رقم 5 يتم جمع قيم المعاملين Parameters ووضع النتيجة فى المتغير s.
- فى السطر رقم 6 يتم استعمال الأمر return لإعادة قيمة المتغير s ، وبالتالى تعود قيمة المتغير s عند استدعاء Call هذه الدالة Method.
- فى السطر رقم 12 يتم استدعاء Call الدالة Sum() مع إرسال القيمتين 10 و 15 كمعاملات Parameters للدالة Method ، وبالتالى يتم إرسال هاتين القيمتين إلى الدالة Sum() التى تقوم بجمعهما وإعادة النتيجة التى توضع فى المتغير r.
- وفى النهاية يتم طباعة قيمة المتغير r وهى القيمة المرتجعة Return Value من استدعاء الدالة Method Call.
- ترجم Compile ونفذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل (4-4).



(الشكل 4-4)

مثال (3): دالة تعيد قيمة حرفية String Method:

وهذا المثال يستعمل التركيب switch لاختيار الأرقام (... ,2 ,1 ,0) وتحويلها إلى العبارة المقابلة ، فمثلاً نريد تحويل الرقم 1 إلى one وهكذا ، وهو يشابه فكرة تفقيط الشيكات.

ولتحقيق ذلك قم بكتابة السطور كما فى الشكل (4-5).

```
1  public class Method2 {
2
3        String Convfunc (int v) {
4  switch (v)
5     {
6  case 0 : return "Zero";
7
8  case 1 : return "One";
9  case 2 : return "Two";
10 case 3 : return "Three";
11 case 4 : return "Four";
12 case 5 : return "Five";
13 case 6 : return "Six";
14 case 7 : return "Seven";
15 case 8 : return "Eight";
16 case 9 : return "Nine";
17    default : return "";
18 }
19    }
20  public static void main(String[] args)
21    {
22    Method2 CN= new Method2();
23   String wordNum = CN. Convfunc (1);
24    System.out.println (wordNum);
25
26     wordNum = CN. Convfunc (2);
27    System.out.println (wordNum);
28
29     wordNum = CN. Convfunc (7);
30    System.out.println (wordNum);
31    }
32 }
```

(الشكل 4-5)

شرح السطور:

- فى السطر رقم 1 يتم إنشاء فصيلة Class بالاسم Method2.
- فى السطر رقم 2 يتم إنشاء دالة Method بالاسم Convertfunc.
- فى السطور من 5 إلى 16 يتم اختبار قيمة المتغير v الذى تستقبله الدالة Method وإعادة أحد الكلمات المقابلة ، وبالتالى إذا كان الرقم 0 تعيد الدالة Method كلمة Zero وإذا كان 1 تعيد one وهكذا.
- فى السطر رقم 17 نجد الكلمة default التي تقوم بإرجاع نص خالي إذا لم تتحقق أى حالة.
- فى السطر رقم 20 تبدأ الدالة الرئيسية ()main التى تتولى سير البرنامج.
- فى السطر رقم 22 يتم تعريف هدف object اسمه CN من نوع الفصيلة Method2.
- فى السطر رقم 23 يتم تعريف المتغير WordNum من نوع String الذي يقوم بتخزين نتيجة استدعاء الدالة ()Convertfunc مع إرسال القيمة 1 كمعامل

Parameter فتقوم الدالة Method بتحويل 1 إلى One ، وبالمثل يتم تحويل الرقمين 2 ,3 وتصبح النتيجة one two three.

- فى السطر رقم 24 و 27 و 30 يتم طباعة محتوى المتغير wordNum وهو one ثم two ثم three.
- قم بكتابة البرنامج كما فى الشكل ثم قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (4-6).

(الشكل 4-6)

دوال من غير قيم مرتجعة Methods with No Returning Values:

- كما اتفقنا يتم تحديد نوع الدالة Method حسب القيمة المرتجعة Return Value ، فماذا إذا كانت الدالة Method لا تعيد قيمة... فى هذه الحالة تسمى الدالة Method بلا عائد أو void ، وهى الدوال Methods التى تقوم بتنفيذ عملية محددة بدون إعادة نتيجة مثل دوال الرسم ودوال الصوت ...
- وهذا ما تم فى المثال الأول حيث تم إنشاء دالة Method لا تعيد قيمة .. فقط تطبع رسالة وبالتالى كان نوعها void.

الدالة الرئيسية main():

هى أول دالة Method يتم تنفيذها فى البرنامج ، ويتم تنفيذها بدون استدعاء Call وتأخذ الشكل التالى:

```
public static void main (String[] args)
```

وتتوفر فيها الصفات التالية: نوعها void أى لا تعيد قيمة.

الاسم main وهو خاص بها ولا تأخذ اسماً أخر.

String[] arg ويعنى أن للدالة Method معاملاً Parameter من نوع مصفوفة حرفيات Array of String وتمثل معاملات Parameters البرنامج.
وكلمة public وكلمة static سوف نتناولها فى فصل المعدلات Modifiers.

مثال (4): دوال من غير قيم مرتجعة Methods with No Returning Values:

فى هذا المثال يتم إنشاء دالة Method بسيطة لاستقبال معامل Parameter عبارة عن كلمة أو جملة وطباعتها.
وهذه الدالة Method توفر عليك السطر

```
System.out.println(message);
```

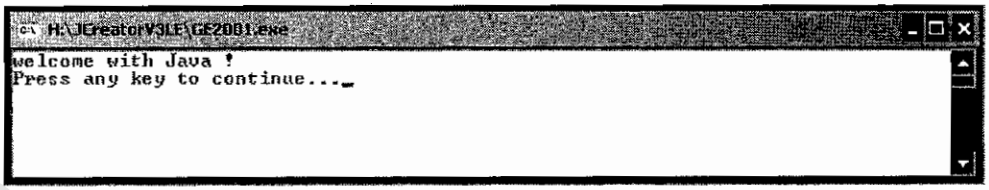
إبدأ برنامج جديد واكتب به سطور البرنامج كما فى الشكل (4-7).

```
1  public class PrintMsg {
2
3      public static void printMsg(String message)
4      {
5          System.out.println(message);
6      }
7
8      public static void main(String[] args)
9      {
10         //calling the method
11         printMsg("welcome with Java !");
12     }
13
14 }
```

(الشكل 4-7)

شرح السطور:

- تم إنشاء دالة Method بالاسم printMsg() من النوع void حيث أنها لا تعيد أي قيمة ، ولها معامل Parameter واحد من النوع String يمثل العبارة المطلوب طباعتها ، ثم تم استدعاء Call هذه الدالة Method وإرسال العبارة Welcome with Java كمعامل Parameter مما يؤدى إلى طباعة الرسالة.
- قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (4-8).



(الشكل 4-9)

مثال (5): مقارنة رقمين:

- فى هذا المثال يتم إنشاء دالة Method تستقبل رقمين وتقارن بينهما وتعيد الرقم الأكبر.
- ولتوضيح ذلك اكتب السطور كما فى الشكل (4-9).

```
1  import java.io.*;
2  public class Laragnum {
3
4  public static double larger(double x,double y)
5  {
6     double largNum;
7       if (x >= y)
8          largNum= x;
9       else
10         largNum = y;
11
12      return largNum;
13 }
14    public static void main(String[] args) throws IOException
15    {
16 double firstNumber, secondNumber;
17 BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
18 System.out.print("first number: ");
19 firstNumbe = Double.parseDouble(stdin.readLine());
20 System.out.print("second number: ");
21 secondNumber = Double.parseDouble(stdin.readLine());
22 System.out.println("the Larg num is: "+larger(firstNumber,secondNumber));
23
24
25    //
26  }
27 }
```

(الشكل 4-9)

شرح السطور:

- تم إنشاء الدالة larger() من النوع double لأنها تعيد قيمة من النوع double حيث تستقبل قيمتين من النوع double ثم تقارن بينهما وتعيد القيمة الأكبر.

وعند تنفيذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل (4-10).

(الشكل 4-10)

مثال (6): بعض الدوال Methods الشائعة:

في هذا المثال يتم إعداد ثلاثة دوال Methods لتحقيق عمليات حسابية معروفة وهى:

1- إيجاد مربع قيمة Square Value.

2- إيجاد مكعب قيمة Cubic Value. 3- جمع قيمتين.

ولتوضيح ذلك اكتب السطور كما فى الشكل (4-11).

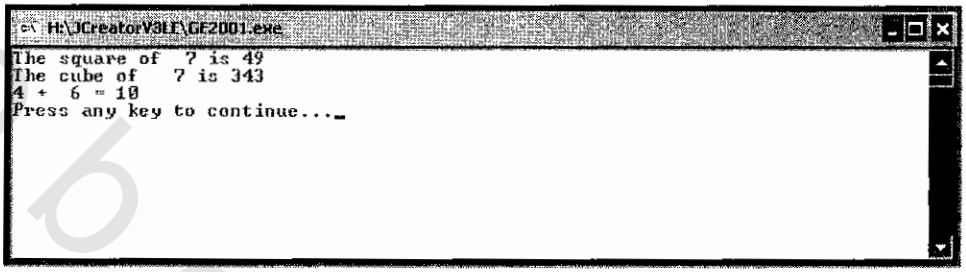
```
public class Calc {

public static int square(int num)
    {
        return num * num ;
    }

    public static int cube(int num)
    {
        int result= num * num * num;
        return result;
    }
    public static long add(long num1,int num2)
    {
        long additionResult = num1 + num2;
        return additionResult;
    }
    public static void main(String[] args)
    {
        int sq,cb,number,number1,number2;
        number = 7;
        number1 = 4;
        number2 = 6;
        sq = square(number);
        cb = cube(number);
        System.out.println("The square of  "+number+" is "+sq);
        System.out.println("The cube of  "+number+" is "+cb);
        System.out.println(number1+" +  "+number2+" = "+add(number1,number2) ) ;
    }


}
```

(الشكل 4-11)

شرح السطور:

تم إنشاء ثلاثة دوال Methods بالأسماء cube(), square(), add() تستقبل معاملات

Parameters وتعيد القيمة المطلوبة ، وعند التنفيذ تحصل على نتيجة التنفيذ كما فى الشكل (4-12).

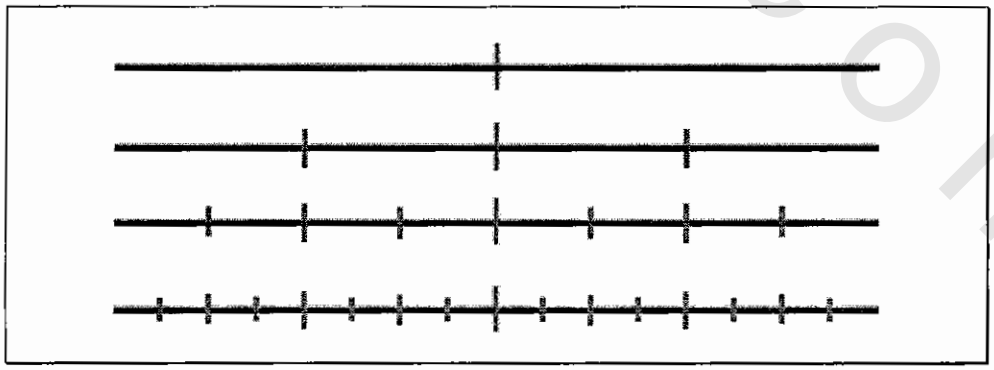


(الشكل 4-12)

استدعاء الدالة لنفسها Recursion:

- هى خاصية موجودة فى الدوال Methods مفيدة جداً وتفيد فى كثير من المشاكل المعقدة ، ومعناها استدعاء الدالة Method لنفسها ، وكثير من المسائل والمشاكل التى يتم حلها بعدد كبير من السطور فى البرمجة يتم حلها بعدد سطور أقل بكثير عن طريق تطبيق فكرة استدعاء الدالة لنفسها Recursion.
- وتقوم الفكرة على محاولة تحديد الجزء الممكن تكراره لتحقيق الهدف.
- فكرة استدعاء الدالة Method لنفسها تبنى على عدة أجزاء:

1- تقسيم المشكلة أو المسألة بالضبط كأن لديك خط وتريد تقسيمه الى أجزاء ، فأولاً يتم تقسيم الخط إلي نصفين ثم يتم تقسيم كل نصف إلى 8 أقسام متساوية كما فى الشكل (4-13).



(الشكل 4-13)

بالتالى يمكن التعبير عن ذلك كما يلى:

```
Divide(1) = 1*2=2
Divide(2) = 2*2=4
Divide(3) = 4*2=8
Divide(4) = 8*2=16
```

والصورة العامة:

```
Divide(N) = Divide(N-1)*2
```

شرط إنهاء استدعاء الدالة Base Case:

- هو الشرط الذى يتوقف عنده التكرار.
- السؤال الآن هو كيف يتم تحقيق ذلك فى لغة Java.
- يتم ذلك ببساطة بكتابة الدالة Method واستدعائها Call داخل سطورها.

مثال (7): استدعاء الدالة لنفسها Recursion:

- لو فرضنا أن لدينا الشكل التالي.

```
* * * * *
* * * *
* * *
* *
*
```

- والمطلوب إيجاد العدد الكلى لنقاط المثلث عند أى نقطة.
- عند التفكير فى إعداد الدالة Method التى تتولى ذلك ، يتم التفكير فى نقطتين:
  1- شرط إنهاء استدعاء الدالة Base Case.
  2- المعادلة التى على أساسها يتم إعداد الصيغة Formula.
- لو حاولنا مراجعة المسألة وتجربة إنشاء جدول بقيم تجريبية ، فيمكننا إنشاء الجدول التالى:

عدد الصفوف # of rows	عدد نقاط المثلث Total # of Pins in Triangle	الصيغة Formula
1	1	Base case
2	3	CountPins(2) = 2 + CountPins(1)
3	6	CountPins(3) = 3 + CountPins(2)
4	10	CountPins(4) = 4 + CountPins(3)
5	15	CountPins(5) = 5 + CountPins(4)
6	21	CountPins(6) = 6 + CountPins(5)

وبالتالى تكون الصيغة العامة هى

```
CountPins(N)  = N + CountPins(N-1)
```

وبالتالى يتم تنفيذ البرنامج كالتالي (بفرض أن عدد الصفوف كان 4).

```
CountPins(4) = 4 + CountPins(3)
    = 4 + ( 3 +CountPins(2) )
    = 4 + ( 3+ (2 +CountPins(1) ) )
    = 4 + ( 3+ (2 + 1) ) )
    = 10
```

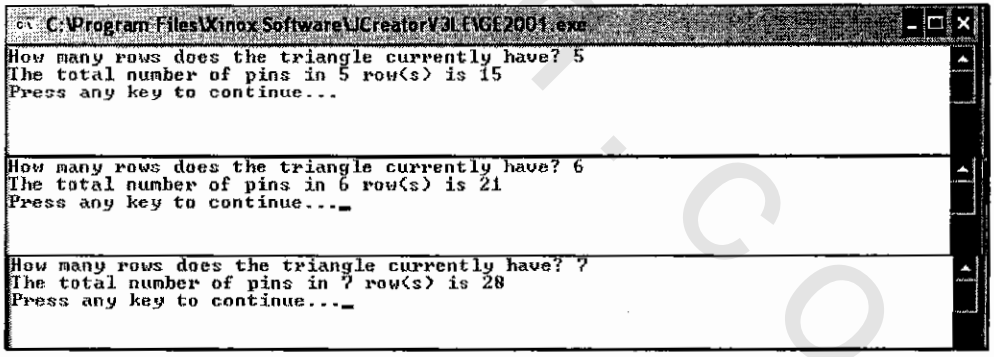
ولتحقيق ذلك يمكن كتابة المثال كما فى الشكل (4-14).

```
public class Test1 {

    static int CountPins(int row)
    {
      if (row == 1)
        return 1;
      else
        return (CountPins(row-1)+row);
    }
    public static void main(String[] args)  throws IOException
    {

      int currentRows;
      String inData;

      InputStreamReader inStream = new InputStreamReader( System.in );
      BufferedReader stdin = new BufferedReader( inStream );
      System.out.print("How many rows does the triangle currently have? ");
      inData = stdin.readLine();
      currentRows = Integer.parseInt( inData );

      System.out.println("The total number of pins in "+currentRows+" row(s) is "+CountPins(currentRows));

    }
}
```

(الشكل 4-14)

شرح السطور:

- فى السطر رقم 4 تم إنشاء دالة Method بالاسم CountPins() لها معامل Parameter واحد من نوع int يمثل رقم الصف المطلوب حساب عدد نقطه.
- فى السطر رقم 6 يتم اختبار قيمة هذا المعامل Parameter ، فإذا كان 1 ، فهذا يعني أن هذا هو الصف الأول وعدد نقطه 1 ، لذلك يتم إعادة القيمة 1 ، وإلا يتم استدعاء Call الدالة Method مرة أخرى ولكن بعدد أقل.
- هذا الاستدعاء Call يؤدى إلى استدعاء Call الدالة Method مرة أخرى أكثر من مرة حتى يتم حساب عدد النقط وتصل إلى الصف الأول.
- فى السطر رقم 20 يتم استقبال رقم الصف من المستخدم.
- فى السطر رقم 21 يتم استدعاء Call الدالة Method مع إرسال رقم الصف كمعامل Parameter للدالة Method ، وتقوم الدالة Method باستدعاء Call نفسها حتى تحسب عدد نقط الصف ثم يتم طباعته.
- وعند تجربة تنفيذ البرنامج أكثر من مرة تحصل على النتائج التالية كما فى الشكل (4-15).



(الشكل 4-15)

مثال (8): حساب المضروب Factorial:

- من الأمثلة المشهورة جداً لنظرية استدعاء Call الدالة Method لنفسها ، مسألة حساب مضروب رقم Factorial ، حيث تلاحظ فيها تكرار العملية أكثر من مرة ، ويتضح ذلك مما يلى:

قواعد التفكير:

1- يجب تحديد شرط إنهاء استدعاء الدالةBase Case :

كما هو معروف أن قيمة مضروب 0 هو 1 ، ولذلك نعتبر هذا هو الأساس.

2- الصيغة العامةThe Formula :

يمكن أن يتم حساب قيمة مضروب Factorial أى رقم بالصيغة التالية.

```
Fact(n) = n*(n-1)*(n-2) *..... *1
```

ولتوضيح ذلك تابع الجدول التالي.

الرقم	Fact(n)	الصيغة Formula
0	1	Base Case
1	1	Fact(1) = 1 * Fact(0)
2	6	Fact(2) = 2 * Fact(1)
3	24	Fact(3) = 3 * Fact(1) * Fact(0)
4	120	Fact(4) = 4 * Fact(3) * Fact(2) * Fact(1)
5	720	Fact(5) = 5 * Fact(4) * Fact(3) * Fact(2) * Fact(1)

من هذا الجدول نجد أن الصيغة العامة هى:

```
Fact(n) = n * Fact (n-1)
```

ويمكن كتابة سطور الدالة Method التى تحقق ذلك كما فى السطور التالية.

```
import java.io.*;
public class RecursiveFactorial
{
   static int Fact(int n)
   {
      if (n==0)
         return 1;
      else
```

```
        return (n*Fact(n-1));

    }
public static void main(String[] args) throws IOException
{
   int num;
   String inData;
InputStreamReader inStream = new
InputStreamReader(System.in);
BufferedReader stdin = new BufferedReader( inStream );
System.out.print("Please enter n: ");
inData = stdin.readLine();
num = Integer.parseInt( inData );
System.out.println("The factorial of "+num+" is "+Fact(num));
 }
}
```

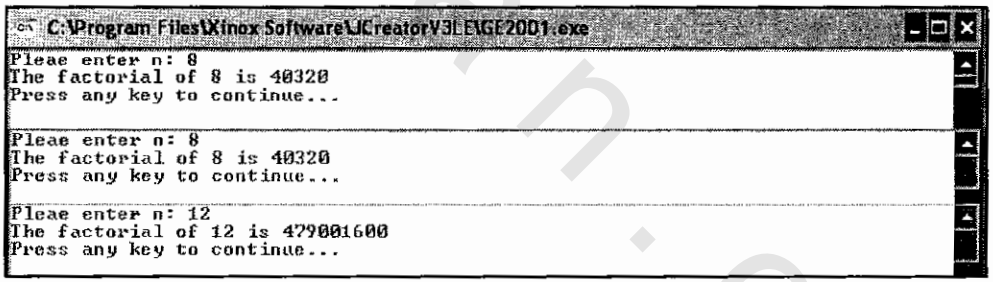
قم بكتابة البرنامج كما يوجد فى الشكل (4-16).

```
1   import java.io.*;
2   public class Test1 {
3
4
5       static int Fact(int n)
6       {
7           if (n==0)
8               return 1;
9           else
10              return (n*Fact(n-1));
11
12      }
13  public static void main(String[] args) throws IOException
14  {
15     int num;
16     String inData;
17
18  InputStreamReader inStream = new InputStreamReader(System.in);
19     BufferedReader stdin = new BufferedReader( inStream );
20  System.out.print("Pleae enter n: ");
21     inData = stdin.readLine();
22     num = Integer.parseInt( inData );
23
24  System.out.println("The factorial of "+num+" is "+Fact(num));
25
26  }
27  }
```

(الشكل 4-16)

شرح السطور:

- فى السطر رقم 5 يتم إنشاء دالة Method بالاسم Fact() لحساب المضروب Factorial ، لها معامل Parameter من النوع int ، وهو المتغير الذى يستقبل الرقم المطلوب حساب مضروبه Factorial.
- فى السطر رقم 7 يتم اختبار قيمة الرقم المطلوب حساب مضروبه Factorial ، فإذا كان 0 يتم إعادة القيمة 1 وينتهى الأمر ، وإلا يتم الاستدعاء Call مرة أخري حتى يتم حساب قيمة المضروب Factorial.
- فى السطور 21 و 22 و 23 يتم استقبال الرقم المطلوب حساب مضروبه Factorial ثم يتم استدعاء Call الدالة Method وحساب المضروب Factorial ثم يتم طباعة النتيجة.
- وعند تنفيذ البرنامج تحصل على النتائج التالية كما فى الشكل (4-17).



(الشكل 4-17)

ملخص الفصل:

- تعرضنا في هذا الفصل لشرح الدوال Methods وكيفية استخدامها.
- في الفصل القادم سوف نتعرف - بإذن الله - علي المصفوفات Arrays ، فتابع معنا الفصل القادم.