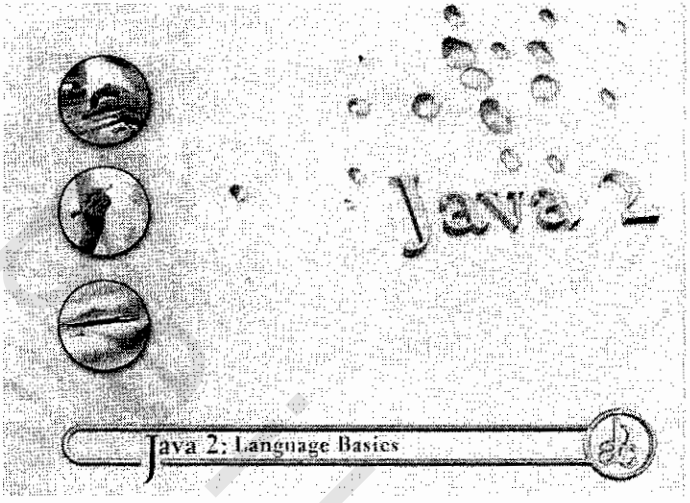
الفصل الثالث



جمل التحكم فى مسار البرنامج وجمل التكرار

Control Statements and Loops

فى هذا الفصل نشرح جمل التحكم فى مسار البرنامج Control Statements وجمل التكرار Loops وذلك من خلال النقاط التالية:

- التعامل مع المدخلات والمخرجات.
- أوامر المخرجات.
- أوامر الإدخال.
- بلوك الأوامر Block Statements.
- إظهار رسالة فى مربع حوار Dialog Box.
- جمل التكرار Looping .
- التكرار باستعمال for Loop.
- التكرار المتداخل Nested Loops.
- التكرار باستعمال while Loop.
- استخدام do ... while.
- الخروج من جمل التكرار Breaking Out Of Loops.
- جمل التحكم فى مسار البرنامج Control Statements.
- جملة if.
- جملة if ... else.
- مؤثر الشرط Conditional Operator.
- التركيب الشرطى switch ... case.

التعامل مع المدخلات والمخرجات:

- قبل تناول جمل التكرار ، نتناول باختصار جمل الإدخال والإخراج فى لغة Java التى تستخدم لطلب بعض البيانات من المستخدم (إدخال) أو لطباعة نتائج البرنامج للمستخدم (إخراج).
- والحقيقة أن هذه الجمل غير مستخدمة واقعياً وتستخدم فقط للتدريب عليها وذلك لأن الأكثر واقعية أن يتم الإدخال والإخراج من برامج النوافذ Windows Application من خلال أدوات مكتبة Swing مثلاً أو من صفحات مواقع الإنترنت Internet.
- وفى هذه المرحلة سنتناول هذه الأوامر للتدريب فقط.

أوامر المخرجات:

تستعمل الدالة println() فى طباعة المخرجات على الشاشة ، وبالطبع هى تطبع العبارات الحرفية String أو الرقمية مع ملاحظة وجود بعض المؤثرات التى تتحكم فى شكل المخرجات ، وتسمى هذه المؤثرات باسم Escape Sequence ، والجدول التالى يوضح هذه المؤثرات.

المؤثر	المعنى
\n	طباعة سطر جديد
\t	طباعة مسافة كبيرة Tab
\r	تعادل الضغط على الزر Enter
\\	طباعة الحرف \
\"	طباعة الحرف "

أوامر الإدخال:

تستخدم الدالة readLine() لإدخال قيمة حرفية String ، فإذا كان إدخال المستخدم عبارة عن قيم رقمية ، فيجب عليك تحويل القيمة الحرفية String إلي رقم عن طريق الدالة Integer.parseInt() التي تستقبل القيمة الحرفية String وتقوم بتحويلها إلي رقم من نوع int.

المثال التالى يوضح هذه النقطة.

مثال (1): أوامر الإدخال:

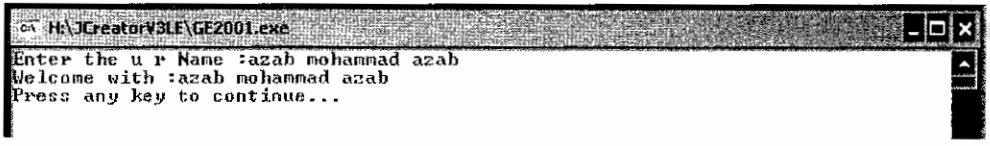
قم بتشغيل أحد برامج كتابة برامج Java ثم اكتب البرنامج التالى:

```
import java.io.*;
public class Echo
{
    public static void main (String[] args)throws IOException
{
  InputStreamReader inStream = new InputStreamReader(
System.in ) ;
     BufferedReader stdin = new BufferedReader( inStream );

     String name;
System.out.print("Enter the u r Name :");
    name = stdin.readLine();
System.out.println("Welcome with :" + name);
   }
}
```

شرح السطور:

يتم بناء فصيلة class التطبيق الرئيسية وفيها يتم تعريف متغيرات ثم طباعة العبارة Enter the u r Name ثم استقبال قيمة من المستخدم وهى اسمه وحفظها فى المتغير name ثم يتم طباعة قيمة هذا المتغير باستعمال الدالة println().

قم بتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (3-1).



(الشكل 3-1)

بلوك الأوامر Block Statements:

كما يوجد فى لغة C ما يسمى باسم بلوك Block ، كذلك يوجد فى لغة Java ، وهو عبارة عن أى مجموعة من الجمل أو الأوامر محصورة بين الأقواس {} كما بالشكل التالى:

```
{
   :
   :
   :
Statement 1
Statement 2
   :
   :
}
```

فى هذا الشكل يتم حصر جمل Statement 1 و Statement 2 بين الأقواس ليتعامل كبلوك Block.

ولكن السؤال ما الفائدة وراء إنشاء بلوك Block ؟

الفائدة هى إمكانية الإعلان عن متغيرات داخل البلوك Block ليس لها علاقة بما قبل أو بعد البلوك Block.

مثال (2): إظهار رسالة فى مربع حوار Dialog Box:

بالطبع أنت تتعجب من استعمالنا للدالة println() لطباعة عبارة على الشاشة ، وبالطبع ليس هذا هو الأسلوب العملى للبرمجة الحالية ، فالأسلوب الحالى هو استعمال مربعات الحوار Dialog Boxes ، وبالطبع سوف نتناول ذلك فى فصول منفصلة متكاملة تحت العنوان Swing components ، لكن حتى نلتقى بذلك ، تابع معنا السطور التالية التى توضح كيفية إظهار البيانات فى مربع حوار Dialog Box ببساطة كما فى الشكل (3-2).

Override.java DialogMsg.java *

```
1  import javax.swing.JOptionPane;
2  public class DialogMsg {
3
4      public static void main(String[] args)
5      {
6          JOptionPane.showMessageDialog(null, "Welcome\nto\nJava\nProgramming!" );
7      }
8  }
9
```

(الشكل 3-2)

- فى هذه السطور يتم استدعاء الدالة showMessageDialog() لإظهار الرسالة المطلوبة.
- نفذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (3-3).

(الشكل 3-3)

وكما ذكرنا سابقاً ، فسوف نتناول ذلك بالتفصيل فى فصول منفصلة تحت العنوان Swing Components.

جمل التكرار Looping :

- هى مجموعة من الجمل التى تستعمل لتكرار تنفيذ الأوامر أكثر من مرة ، ويوجد منها فى لغة Java الجمل التالية:

- جملة for.
- جملة while.
- جملة do ... while.

التكرار باستعمال for Loop:

تستعمل هذه الجملة لتكرار تنفيذ عملية أكثر من مرة ، وهى أبسط وأشهر أنواع جمل التكرار ، وتأخذ الصورة العامة التالية:

```
for ([initializers]; [expression]; [iterators])
  statement
```

فى هذه الصورة تأخذ جملة for الأجزاء التالية:

- for: الأمر نفسه المستعمل فى التكرار.
- initializers: القيمة الابتدائية التى يبدأ بها التكرار.
- expression: شرط التكرار الذى سيستمر التكرار بناء عليه طالما هذا الشرط صحيح.
- iterators: مقدار الزيادة.
- statement: هى الجملة المطلوب تنفيذها داخل التكرار ، ويتم استعمال الأقواس { } لتكوين البلوك Block الذى يتم تنفيذه داخل التكرار ، فإذا كانت جملة واحدة لا تحتاج إلى أقواس.
- لاحظ فصل أجزاء جملة for بالفاصلة المنقوطة Semi Colon (؛).

لا تنتهى جملة for أو جملة if بالفاصلة المنقوطة ؛ وإلا ستجد أن البرنامج يسير فى خطوات خطأ ، فمثلاً لا تكتب السطور التالية:

```
for (i =0; i<20; i++); //Logic Error
System.out.println ("Java –");
```

ففى هذه الحالة لا يتم تنفيذ الدالة println() عشر مرات ، بل يتم تنفيذ التكرار منفرداً بدون التأثير على الدالة println() ثم يتم تنفيذ الدالة مرة واحدة فقط.

- ويتم استعمال التكرار for بكثرة مع المصفوفات Arrays لتجنب التعامل مع كل عنصر على حدة كما سنري في الفصول القادمة.
- يتضح استخدام for من المثال التالي حيث نريد طباعة كلمة Java 20 مرة ، فنجد أنه بدلاً من كتابة 20 سطراً لتحقيق ذلك ، فإنه باستخدام جملة for نستطيع كتابة سطر واحد فقط لتنفيذ ذلك.
- ويتضح ذلك من السطر التالي:

```
for (int i = 1; i <= 5; i++)
System.out.println ("Java –");
```

مثال (3): جملة for:

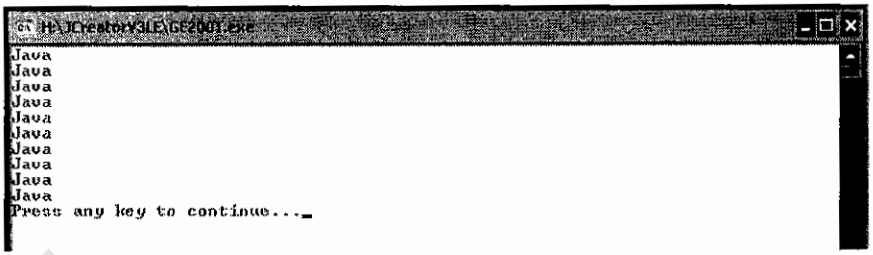
- لتوضيح النقاط السابقة قم بإنشاء برنامج جديد واكتب سطوره كما فى الشكل (3-4).

```
1   //import java.io.*;
2   class For1 {
3
4     public static void main(String[] args) {
5       // Create application frame.
6   int i;
7               for ( i = 1; i <= 10; i++)
8         System.out.println("Java ");
9
10    }
11  }
```

(الشكل 3-4)

شرح السطور:

- فى السطر رقم 6 تم الاعلان عن المتغير i من النوع int .
- فى السطر رقم 7 يتم استعمال جملة التكرار for للتكرار بداية من 1 إلى 10 حسب قواعد الجملة for التى شرحناها في النقاط السابقة.
- فى السطر رقم 8 يتم طباعة الكلمة Java داخل التكرار for وبالتالى يتم طباعة الكلمة Java 10 مرات.
- قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (3-5).



(الشكل 3-5)

مثال (4) : تغيير مقدار الزيادة:

يمكننا تغيير مقدار الزيادة عن طريق الصورة التالية:

```
for(i=0;i<20;i+=2)
```

في هذه الصورة يتم زيادة قيمة المتغير i بمقدار 2 كل مرة.

ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل (3-6).

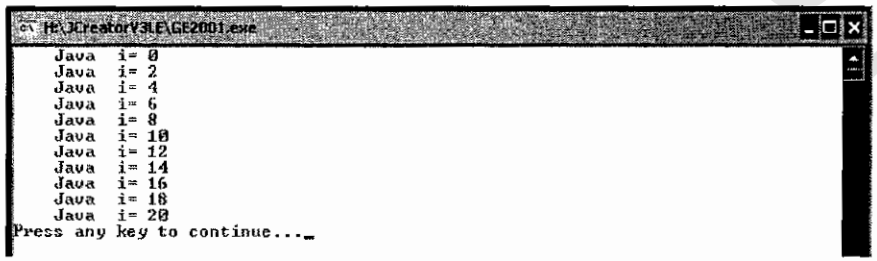
```
1  class For1 {
2
3    public static void main(String[] args) {
4      // Create application frame.
5  int i;
6              for ( i = 0; i <= 20; i+=2)
7        System.out.println("   Java  i= "+i);
8
9    }
10 }
```

(الشكل 3-6)

شرح السطور:

تم تغيير مقدار الزيادة إلى 2 بدلاً من 1 مع طباعة قيمة المتغير i كل مرة.

قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (3-7).



(الشكل 3-9)

مثال (5): الطباعة من الأكبر إلى الأصغر:

يمكننا أن نقوم بتنفيذ التكرار من القيمة الكبري إلي الصغري عن طريق الصورة التالية:

```
for(i=20;i>0;i--)
```

فى هذه الصورة يبدأ التكرار من القيمة 20 ثم يتناقص بمقدار 1 ، ويستمر التنفيذ طالما كانت القيمة الحالية للمتغير i أكبر من 0.

جرب تغيير ذلك فى المثال السابق ونفذه وشاهد النتيجة وسنترك هذا المثال كتمرين لك.

التكرار المتداخل Nested Loops:

وفيه يتم استعمال تكرار for داخل تكرار for أخر كما يلى:

```
for(r=0;r<5;r++)
for(c=0;c<6;c++)
System.out.println("r="+r+"c="+c);
```

مثال (6): التكرار المتداخل Nested Loops:

لتوضيح التكرار المتداخل Nested Loops قم بتعديل البرنامج السابق ليصبح كما فى الشكل (3-8).

```
1  public class For4 {
2
3      public static void main(String[] args) {
4          // Create application frame.
5  int r,c;
6       for(r=0;r<5;r++)
7       {
8         System.out.println("");
9
10
11 for(c=0;c<6;c++)
12 System.out.print("  r="+r+"   c="+c);
13 }
14    }
15 }
16
```

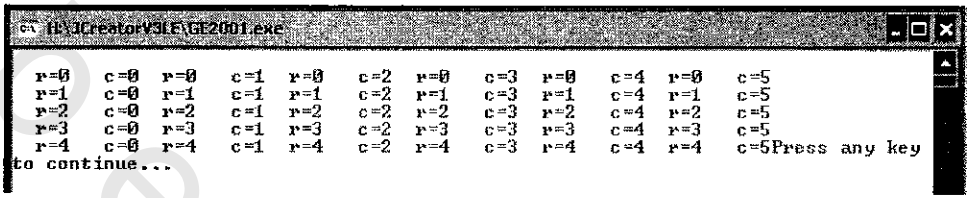
(الشكل 3-8)

شرح السطور:

يتم تنفيذ التكرار الأول بالمتغير r من القيمة 0 إلى القيمة 4 (خمس تكرارات) ،

وبداخل هذا التكرار يتم تنفيذ تكرار أخر من القيمة 0 إلى القيمة 5 (ست تكرارات) بالمتغير c ، وبالتالى يكون عدد مرات التكرار هى 6×5 أي 30 ، وحيث أننا نطبع قيمة متغيرين ، فلذلك نتوقع وجود 60 قيمة مطبوعة.

- قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (3-9).



(الشكل 3-9)

مثال (7) : جدول الضرب:

- من الأمثلة الواضحة جداً على استعمال التكرار المتداخل Nested Loops هو طباعة جدول الضرب المشهور.
- ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل (3-10).

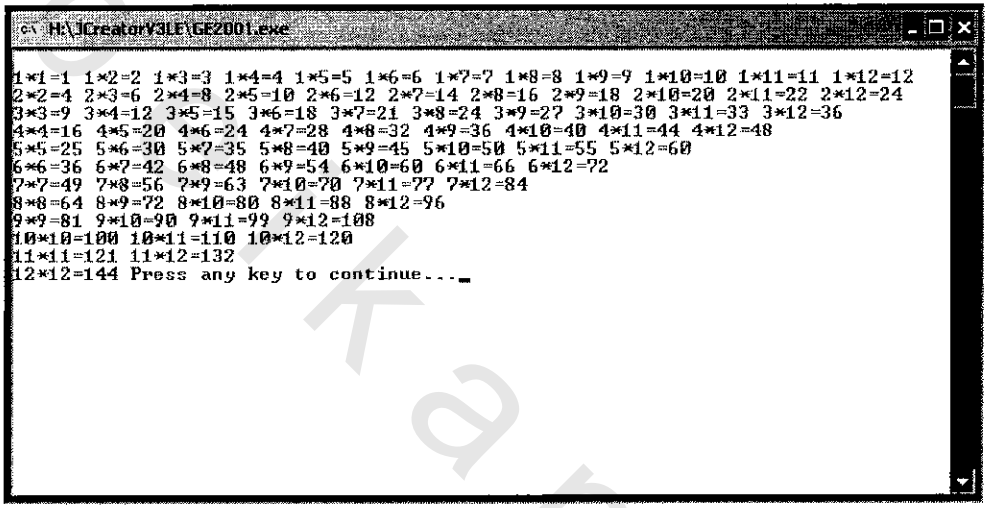
```
1  public class For4 {
2
3     public static void main(String[] args) {
4        // Create application frame.
5  int r,c;
6        for(r=1;r<13;r++)
7        {
8          System.out.println("");
9
10
11 for(c=r;c<13;c++)
12 System.out.print(r+"*"+c+"="+(r*c)+" " );
13
14 }
15    }
16 }
17
```

(الشكل 3-10)

شرح السطور:

- فى السطر رقم 6 يبدأ التكرار الأول من القيمة 1 إلى القيمة 12.
- فى السطر رقم 8 يتم طباعة سطر خالي.
- فى السطر رقم 11 يبدأ التكرار الثانى من القيمة 1 إلى القيمة 12.

- فى السطر رقم 12 يتم طباعة حاصل ضرب r*c ، وبالتالى يتم طباعة جدول لضرب للرقم 1 ثم يتم طباعة سطر جديد ثم جدول 2 ثم 3 وهكذا حتى ينتهى جدول الضرب مع ملاحظة أن التكرار الثانى للمتغير c يبدأ من القيمة التى يبدأ منها التكرار الأول حتى لا يحدث تكرار لنصف الجدول مرة أخرى.
- قم بترجمة Compile وتنفيذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل (3-11).



(الشكل 3-11)

مثال (8): التكرار باستعمال while Loop:

- تستعمل هذه الجملة لتكرار تنفيذ مجموعة من السطور عدد من المرات غير محدد مسبقاً ، ويتم إنهاء التنفيذ بناء على شرط ، ولذلك يفضل استعمالها فى حالة عدم معرفة عدد مرات التكرار.
- وتأخذ الجملة التركيب التالى:

```
while (expression)
statements
```

- فى هذه الصورة يتم اختبار الشرط *expression* ، فإذا كان صحيحاً يتم تنفيذ الجمل Statements ، ويستمر التنفيذ حتى يصبح الشرط غير صحيح.
- ولتوضيح ذلك تابع الجمل التالية:

```
int n = 1;

    while (n <= 10)
    {
      System.out.println (n);
      n++;
    }
```

شرح السطور:

- يتم استعمال التكرار while لطباعة قيمة المتغير n طالما كانت قيمة هذا المتغير أقل من أو تساوي 10 ثم يتم زيادة قيمة المتغير n.
- ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل (3-12).

```
1  public class For4 {
2
3    public static void main(String[] args) {
4      // Create application frame.
5  int n = 1;
6
7      while (n <=10)
8      {
9        System.out.println ("     n="+n);
10       n++;
11 }
12 }
13 }
```

(الشكل 3-12)

- قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (3-13).

```
n=1
n=2
n=3
n=4
n=5
n=6
n=7
n=8
n=9
n=10
Press any key to continue...
```

(الشكل 3-13)

استخدام do ... while:

- يستخدم هذا التكرار مثل while ولكن يختلف عنه فى أنه يبدأ أولاً بتنفيذ الجمل ثم يختبر الشرط الموجود فى while فى نهاية التركيب ، فإذا كان صحيحاً يعاد التكرار مرة أخرى وإلا يتوقف ، وبالتالى يتم تنفيذ الجمل مرة واحدة على الأقل حتى لو كان الشرط غير صحيح.
- تأخذ هذه الجملة الصورة العامة التالية.

```
do {
Statements
} while (Condition);
```

- فى هذه الصورة يتم تنفيذ الجمل Statements أولاً ثم يتم اختبار الشرط الموجود مع while ، فإذا كان صحيحاً يعود التكرار إلى do ويتم تنفيذ الجمل مرة أخرى وهكذا.

مثال (9): جملة do ... while:

- فى السطور التالية نعرض مثال بسيط لاستعمال do---while لطباعة مجموعة عبارات كما فى الشكل (3-14).

```
1  public class For4 {
2
3      public static void main(String[] args) {
4          // Create application frame.
5
6
7          int x = 1;
8          do {
9          System.out.println ("this is no "+x);
10         x++;
11         } while (x<=20);
12
13
14 }
15 }
```

(الشكل 3-14)

فى هذه السطور:

- في السطر رقم 1 يتم تعريف فصيلة جديدة بالاسم For4.
- ثم يبدأ التكرار فى السطر رقم 8 بالأمر do فيتم طباعة الرسالة this is no بالإضافة لقيمة المتغير x. لاحظ أن قيمة المتغير x أول مرة ستكون 1.

- ثم يتم زيادة قيمة x ، وفى السطر رقم 11 يتم استعمال باقى تركيب التكرار وهو الأمر while الذى يختبر قيمة المتغير x ، فإذا كانت أقل من 20 يعود التكرار مرة أخرى إلى أول التكرار في السطر رقم 9.

الخروج من جمل التكرار Breaking Out of Loops:

- يمكن استعمال الأمر break للخروج من التكرار قبل إنهاء الشرط الخاص به ويظهر كما فى السطور التالية:

```
int i =0;
while (i <50)
{
if (i >20) break;
System.out.println(i);
}
```

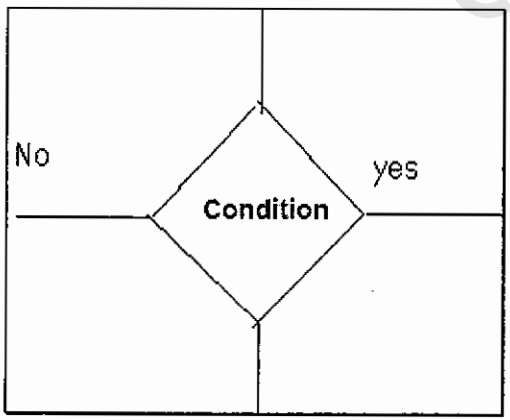
شرح السطور:

- يتم التكرار من 0 حتى 50 ولكن يوجد شرط في السطر الرابع يقوم بالخروج من هذا التكرار عندما تكون قيمة المتغير i أكبر من 20.

جمل التحكم فى مسار البرنامج Control Statements:

- هذه الجمل هي التي تحدد مسار البرنامج ، هل يسير في الخط الطبيعي أم يتفرع Branch إلي دالة Method فرعية أخري.

و يتضح ذلك من خريطة التدفق Flow Chart الموضحة في الشكل التالي :



- في هذا الشكل تلاحظ وجود جملة شرط تحدد مسار البرنامج فإذا تحقق الشرط يتم تنفيذ الجمل الموجودة في المسار yes ، وإذا لم يتحقق الشرط يتم تنفيذ الجمل الموجودة في المسار No.
- وفي النقاط التالية سنقوم بشرح جمل التحكم Control Statements التالية:
  - جملة if.
  - جملة if ... else.
  - جملة if ... else if ... else.
  - جملة switch ... case.

جملة if:

- تستخدم جملة if كما هي بدون else إذا كان هناك شرط نريد اختباره ، فإذا تحقق الشرط نقوم بتنفيذ جمل معينة وإلا يسير البرنامج في المسار الطبيعي ولا ينفذ هذه الجمل التي بداخل الشرط.
- تأخذ جملة if الصورة العامة التالية:

```
if (expression)
  statement1
else
  statement2]
```

- وجملة if تأخذ شكلين:

الشكل الأول:

```
if (Condition)
Statement
```

وهى صورة بسيطة لا تحتاج إلى أقواس {} ، وتستعمل عندما يكون الكود الذي نريد تنفيذه عند تحقق الشرط عبارة عن جملة واحدة.

الشكل الثاني:

```
if (condition )
{
Statements 1
Statements 2
}
```

وتستعمل هذه الصورة عندما يكون الكود الذي نريد تنفيذه عند تحقق الشرط عبارة عن أكثر من جملة واحدة.

مثال (10): جملة if:

```
 if (x>y)
System.out.println ("x is greater than y");
```

- فى هذا السطر يتم استعمال if لاختبار الشرط (x>y) ، فإذا كان صحيحاً (أى أن قيمة x أكبر من قيمة y) يتم تنفيذ الجملة التالية للشرط وهى طباعة العبارة الموجودة.
- فى السطر السابق يتم تنفيذ جملة واحدة كنتيجة للشرط ، ولكن إذا أردت تنفيذ أكثر من جملة ، فيجب استخدام الأقواس { } لتصبح بلوك Block فيتم تنفيذه كنتيجة للشرط بالشكل التالى:

```
if (x>y)
{
System.out.println ("x is greater than y");
System.out.println ("x is greater than y");
}
```

جملة if ... else:

- يستعمل هذا التركيب لاختبار شرط معين ، فإذا كان صحيحاً يتم تنفيذ ما بعد جملة if ، وإذا كان غير صحيح يتم تنفيذ ما بعد جملة else.
- تأخذ هذه الجملة الشكل العام التالي:

```
if (expression)
  statement1
else
  statement2
```

- في هذه الصورة يتم اختبار الشرط الموجود بجملة if ، فإذا كان الشرط صحيحاً ننفذ الجملة statement1 ، أما إذا كان الشرط غير صحيح فيتم تنفيذ الجملة statement2.

ولتوضيح ذلك تابع المثال التالي.

مثال (11): جملة if ... else:

```
1.    if (x>y)
2.    {
3.    System.out.println ("x is greater than y – ");
4.    System.out.println ("so you can go – ");
5.    }
6.    else
7.    {
8.    System.out.println ("x is less than y – ");
9.    System.out.println ("so you can not go – ");
10.   }
```

شرح السطور:

يتم اختبار الشرط (x>y) فإذا كان صحيحاً يتم تنفيذ الجمل التى فى السطور 3،4 وإلا يتم تنفيذ الجمل فى السطور 8،9.

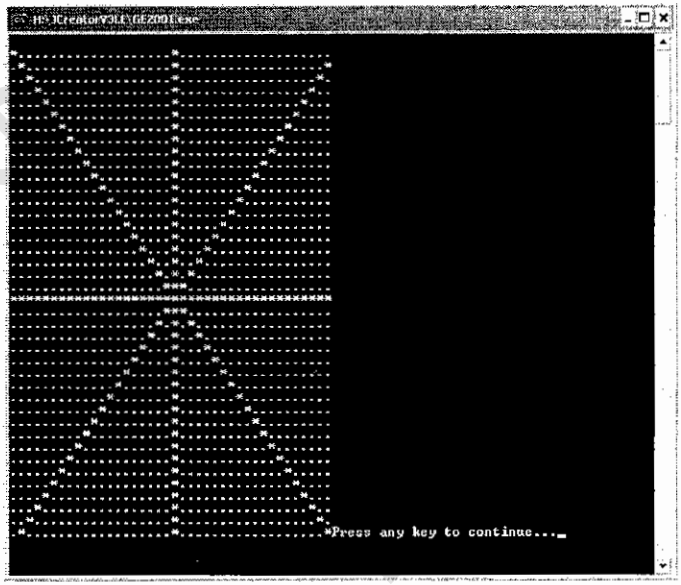
مثال (12): جملة if ... else والتكرار المتداخل Nested Loops:

قم بإعداد برنامج واكتب السطور التالية كما فى الشكل (3-15).

```
public class For4 {

    public static void main(String[] args) {
       // Create application frame.


    int r,c;
    for (r=0;r<40;r++)
    {

       System.out.println("");
    for (c=0;c<40;c++)
    {
    if ((r==c) || ( c==(40-r)) || r==20 ||c==20)
    System.out.print("*");
    else
    System.out.print(".");
    }
 }

 }
 }
```

(الشكل 3-15)

- هذا المثال يوضح كيفية استعمال التكرار المتداخل Nested Loops مع كيفية استعمال تركيب الشرط if ... else ، راجع الشروط السابقة وشاهد نتيجة التنفيذ ولاحظ كيف حقق هذا الشرط هذه النتيجة.
- قم بترجمة Compile وتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (16-3).



(الشكل 3-16)

مؤثر الشرط Conditional Operator:

بالإضافة لجملة if ، يوجد مؤثر الشرط الذى يحقق عمل الأمر if ... else بشكل بسيط كما يلى:

```
Condition ? true result: false result;
```

فى هذا السطر:

- Condition: هو الشرط المطلوب اختباره.
- true result: يتم تنفيذه إذا كان الشرط صحيحاً.
- false result: يتم تنفيذه إذا كان الشرط غير صحيح.

وبالتالى إذا كان الشرط Condition صحيحاً فيتم تنفيذ الجملة المكتوبة فى مكان result true وإلا يتم تنفيذ الجملة الأخرى false result ، ويظهر ذلك من المثال التالى:

```
k= (x > y) ? 10:20:
```

فى هذا المثال تأخذ k القيمة 10 إذا كانت x أكبر من y.

وتأخذ k القيمة 20 إذا كانت x أصغر من y.

وهذا المؤثر يكافئ جملة if ... else كما يلى:

```
if (x > y)
k=10;
else
k=20;
```

التركيب الشرطى switch ... case:

هذا التركيب يستعمل بدلاً من استعمال أكثر من جملة if ... else ، فمثلاً إذا كانت لديك قيمة وتريد اختبارها مع وجود احتمال أن تأخذ هذه القيمة أكثر من نتيجة ، فنستعمل لذلك التركيب switch ... case الذى يأخذ الشكل العام التالى:

```
switch (variable)
{
  case value 1:
    statement1
   break;
  case value 2:
    statement2
   break;
  default:
    statement
   break;
}
```

فى هذه الصورة:

- نبدأ بالكلمة switch وبداخلها المتغير *variable* الذى يمثل المتغير الذي نريد اختباره.
- الحالة الأولى case value 1: يتم تنفيذ الجملة statement1 إذا كانت قيمة المتغير variable تساوي القيمة value 1.
- الحالة الثانية case value 2: يتم تنفيذ الجملة statement2 إذا كانت قيمة المتغير variable تساوي القيمة value 2 ، وهكذا.
- وفى النهاية نجد جملة default التى يتم تنفيذها فى حالة عدم تحقق أى حالة من الحالات السابقة ، مع ملاحظة أن الكلمات المحجوزة Reserved Keywords فى هذه الصورة هى: switch, case, break, default.
- الكلمة switch: تستخدم لتحديد اسم المتغير الذي نريد اختباره.
- الكلمة case: تستخدم لتحديد الحالات المختلفة للمتغير.
- الكلمة break: تستخدم للخروج من التركيب الشرطى switch عند تحقق حالة معينة.
- الكلمة default : تستخدم لتنفيذ مجموعة من الجمل فى حالة عدم تحقق أى حالة.

مثال (13) : جملة switch ... case:

- يوضح هذا المثال كيفية استعمال التركيب switch ... case.
- اكتب البرنامج كما فى الشكل (3-17).

```
public class Switchcase {

    public static void main(String[] args) {
    int v=1;
       switch (v)
{
    case 0 :
     System.out.println ( "Zero");
    break;
    case 1 :
     System.out.println ( "One");
     break;
    case 2 :
     System.out.println ( "Two");
     break;
    case 3 :
     System.out.println ( "Three");
     break;
    case 4 :
     System.out.println ( "Four");break;
     case 5 :
     System.out.println ( "Five");break ;
    case 6 : System.out.println ( "Six");break;
    case 7 : System.out.println ( "Seven");break;
    case 8 : System.out.println ( "Eight");break;
    case 9 : System.out.println ( "Nine");break;
    default :System.out.println ( " ");break;
    }
    }}

```

(الشكل 3-17)

وعند التنفيذ تحصل على نتيجة التنفيذ التالية كما في الشكل (3-18).

```
H:\JCreatorV3LE\GE2001.exe
One
Press any key to continue...
```

(الشكل 3-18)

ملخص الفصل:

تعرضنا في هذا الفصل لشرح جمل التحكم Control Statements وجمل التكرار Loop.

في الفصل القادم سوف نتعرف - بإذن الله - علي الدوال Methods ، فتابع معنا الفصل القادم.