Java Loops

3.1 Loops

3.1.1 `for()`

Many tasks within a program are repetitive, such as prompting for data, counting values, and so on. The `for` loop allows the execution of a block of code for a given control function. The following is an example format; if there is only one statement in the block then the braces can be omitted. Figure 3.1 shows a flow chart representation of this statement.

```
for (starting condition;test condition;operation)
{
     statement block
}
```

where :

`starting condition` – the starting value for the loop;
`test condition` – if `test condition` is TRUE the loop will continue execution;
`operation` – the operation conducted at the end of the loop.

Displaying ASCII characters

Program 3.1 displays ASCII characters for entered start and end decimal values. Sample run 3.1 displays the ASCII characters from decimal 40 (‘(’) to 50 (‘2’). The type conversion `(char)` is used to convert an integer to a `char`.

 Java program 3.1

```
public class chap3_01
{
   public static void main (String args[])
   {
   int start,end,ch;
      start=40; end=50;
      for (ch=start;ch<=end;ch++)
           System.out.println((int)ch+" "+(char)ch);
   }
}
```

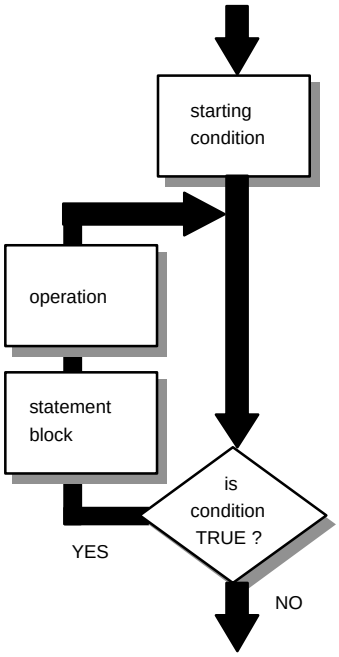

Figure 3.1 Flow chart representation of the `for` statement

Sample run 3.1

```
C:\java\src\chap2>java chap3_01
40 (
41 )
42 *
43 +
44 ,
45 -
46 .
47 /
48 0
49 1
50 2
```

Simulation of a mathematical equation

The program in this section will simulate the results of the equation:

$$y = 3x^2 - 12x - 1$$

for values of x from 0 to 100 in steps of 10. Program 3.2 gives a Java program which implements this. Test run 3.2 shows a sample run of the program. It can be seen that the value of x varies from 0 to 100, in steps of 10.

Java program 3.2

```
public class chap3_02
{
   public static void main (String args[])
   {
   double x,y;
      System.out.println("X   Y");
      for (x=0;x<=100;x+=10)
      {
           y=3*(x*x)-12*x-1;
           System.out.println(x+" "+y);
      }
   }
}
```

Sample run 3.2

```
C:\java\src\chap2>java chap3_02
X   Y
0.0 -1.0
10.0 179.0
20.0 959.0
30.0 2339.0
40.0 4319.0
50.0 6899.0
60.0 10079.0
70.0 13859.0
80.0 18239.0
90.0 23219.0
100.0 28799.0
```

Boolean logic

Program 3.3 is an example of how a Boolean logic function can be analysed and a truth table generated. The `for` loop generates all the required binary permutations for a truth table. The Boolean function used is:

$$Z = \overline{(A.B) + C}$$

A schematic of this equation is given in Figure 3.2. Test run 3.3 shows a sample run. The above equation is implemented in Java with:

```
z=~((a & b) | c)        // not ( (a and b) or c)
```

and as `z` is a 16-bit integer then to just show the first bit of the value then the following bit mask is used:

```
z=~((a & b) | c) & 1;  // mask-off least-significant bit
```

which will only display the least-significant bit of the operation.

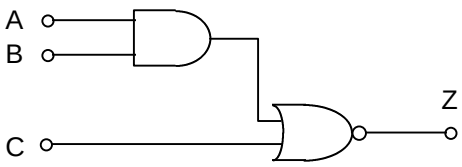

Figure 3.2 Digital circuit

Java program 3.3

```
public class chap3_03
{
   public static void main (String args[])
   {
   int a,b,c,z;

      System.out.println("A B C Z");

      for (a=0;a<=1;a++)
      for (b=0;b<=1;b++)
      for (c=0;c<=1;c++)
      {
           z=~((a & b) | c) & 1;
           System.out.println(a + " "+b +" "+ c +" "+ z);
      }
   }
}
```

Sample run 3.3

```
C:\java\src\chap2>java chap3_03
A B C Z
0 0 0 1
0 0 1 0
0 1 0 1
0 1 1 0
1 0 0 1
1 0 1 0
1 1 0 0
1 1 1 0
```

3.1.2 while( )

The `while()` statement allows a block of code to be executed while a specified condition is TRUE. It checks the condition at the start of the block; if this is TRUE the block is executed, else it will exit the loop. The syntax is:

```
while (condition)
{
   :        :
   statement block
   :        :
}
```

If the statement block contains a single statement then the braces may be omitted (although it does no harm to keep them).

3.1.3 `do...while()`

The `do...while()` statement is similar in its operation to `while()` except that it tests the condition at the bottom of the loop. This allows *statement block* to be executed at least once. The syntax is:

```
do
{
     statement block
} while (condition);
```

As with `for()` and `while()` loops the braces are optional. The `do...while()` loop requires a semicolon at the end of the loop, whereas the `while()` does not.

Figure 3.3 shows a flow chart representation of the `do...while()` and the `while()` loops. In both loops a TRUE condition will cause the statement block to be repeated.

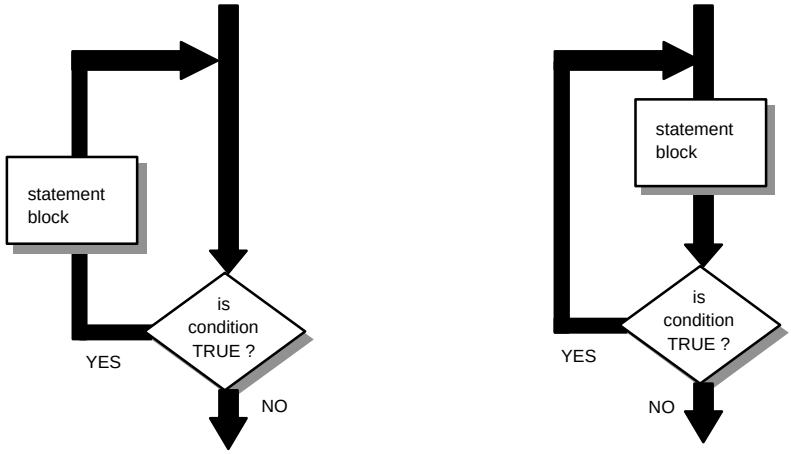

Figure 3.3 `while()` and `do...while()` loops

3.1.4 Conversion from decimal to octal

Octal numbers uses base eight. To convert a decimal value to an octal number the decimal value is divided by 8 recursively and each remainder noted. The first remainder gives the least significant digit and the final remainder the most significant digit. For example, the following shows the octal equivalent of the decimal number 55:

```
8 | 55
  |  6  r 7  <<< LSD (least significant digit)
  |  0  r 6  <<< MSD (most significant digit)
```

Thus the decimal value `55` is equivalent to `67o` (where the `o` represents octal). Program 3.4 shows a program which determines an octal value for an entered decimal value. Unfortunately, it displays the least significant digit first and the most significant digit last, thus the displayed value must be read in reverse. Test run 3.4 shows a sample run.

📖 Java program 3.4

```
public class chap3_04
{
   public static void main (String args[])
   {
   int val,remainder;

      val=55;
      System.out.println("Conversion to octal (in reverse)");
      do
      {
           remainder=val % 8;   // find remainder with modulus
           System.out.print(remainder);
           val=val / 8;
      } while (val>0);
   }
}
```

💻 **Sample run 3.4**

```
Conversion to octal (in reverse)
76
```

3.2 Exercises

3.2.1 Write a program which prints all the characters from ‘0’ (zero) to ‘z’ in sequence using a `for` loop.

3.2.2 Enter Program 3.1 and use it to complete Table 3.1.

3.2.3 Write a program which lists the square of the values from 1 to 10. A sample run in shown in Sample run 3.5.

3.2.4 Write a program which displays the squares, cubes and fourth powers of the first 10 integers. A sample run in shown in Sample run 3.6.

Table 3.1 ASCII characters

Value	*Character*
34	
35	
36	
37	
38	
64	
65	
66	
67	
68	
69	
70	

Sample run 3.5

```
Value   Square
1       1
2       4
3       9
4       16
5       25
6       36
7       49
8       64
9       81
10      100
```

Sample run 3.6

```
Number  Square  Cube  Fourth
----------------------------
1       1       1     1
2       4       8     16
3       9       27    81
   etc
```

3.2.5 Write a program which displays the *y* values in the formulas given below and with the given *x* steps.

Equation	*Range of x*
(i) $y = 4x + 1$	0 to 50 in steps of 5
(ii) $y = \sqrt{x} - 1$	1 to 10 in steps of 0.5
(iii) $y = 5x^2 + 3x - 2$	–5 to 5 in steps of 0.5

A sample run in shown in Sample run 3.7.

Sample run 3.7

```
EQUATION y=4x+1, x goes from 0 to 50 in steps of 5
 x     y
 0     1
 5    21
10    41
15    61
  etc
```

3.2.6 Java program 3.5 gives a program which displays the sine of a number from 0° to 90° in steps of 10° and Sample run 3.8 gives a sample run. Modify it so that it determines the cosine of an angle from 0° to 90° in steps of 10°. (Hint: use `cos()` method instead of `sin()`.)

Java program 3.5

```
import java.lang.Math;

public class chap3_05
{
   public static void main (String args[])
   {
   double val;

      System.out.println("Val\tSine");
      for (val=0;val<=90;val+=10)
      {
         System.out.println(val+"\t"+
      Math.sin(val*3.14/180)); // convert to radians
      }
   }
}
```

Sample run 3.8

```
C:\java\src\chap3>java chap3_05
Val     Sine
0.0     0.0
10.0    0.17356104045380674
20.0    0.34185384854620343

    ::   ::

80.0    0.9846845901305833
90.0    0.9999996829318346
```

3.2.7 Modify Program 3.3 so that it determines the truth table for the following Boolean equation:

$$Z = \overline{(A + B).C}$$

Table 3.1 Truth table

A	*B*	*C*	*Z*
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

6.4.8 Write a program to convert from decimal to binary (base 2). A sample run is shown in Sample run 3.9 with a value of 42. Hint: Modify Program 3.4 so that the operator value is 2 rather that 8.

Sample run 3.9

```
The value in binary is (in reverse) 0100001
```

6.4.9 Write a program to convert from decimal to hexadecimal (base 16). A sample run for a value of 42 is shown in Sample run 3.10. Program 3.6 shows a sample outline of the program.

Sample run 3.10

```
The value in hex is (in reverse) A2
```

Java program 3.6

```
public class chap3_06
{
   public static void main (String args[])
   {
   int val,remain;

      val=55;
      System.out.println("Conversion to hex (in reverse)");
      do
      {
         remain=val % 16;   // find remainder with modulus
         if (remain<10) System.out.print(remainder);
         else if (remain==10) System.out.print('A');
         else if (remain==11) System.out.print('B');
                 etc
         val=val / 16;
      } while (val>0);
   }
}
```