

## The for Statement

The for statement provides a compact way to iterate over a range of values. Programmers often refer to it as the "for loop" because of the way in which it repeatedly loops until a particular condition is satisfied. The general form of the for statement can be expressed as follows:

```
for (initialization; termination; increment)  
{  
    statement(s)  
}
```

When using this version of the for statement, keep in mind that:

- The *initialization* expression initializes the loop; it's executed once, as the loop begins.
- When the *termination* expression evaluates to false, the loop terminates.
- The *increment* expression is invoked after each iteration through the loop; it is perfectly acceptable for this expression to increment *or* decrement a value.

The following program, [ForDemo](#), uses the general form of the for statement to print the numbers 1 through 10 to standard output:

```
public class ForDemo  
{  
    public static void main(String[] args)  
    {  
        for(int i=1; i<11; i++)  
        {  
            System.out.println("Count is: " + i);  
        }  
    }  
}
```

The output of this program is:

```
Count is: 1  
Count is: 2  
Count is: 3  
Count is: 4  
Count is: 5  
Count is: 6  
Count is: 7  
Count is: 8  
Count is: 9  
Count is: 10
```

Notice how the code declares a variable within the initialization expression. The scope of this variable extends from its declaration to the end of the block governed by the for statement, so it can be used in the termination and increment expressions as well. If the variable that controls a for statement is not needed outside of the loop, it's best to declare the variable in the initialization expression. The names *i*, *j*, and *k* are often used to control for loops; declaring them within the initialization expression limits their life span and reduces errors.

Code	OUTPUT
<pre> for(int a = 4;a &gt; 1;a--)  {     System.out.println(a); } </pre>	4 3 2
<pre> int anynum = 3;  for(int x = -1;x &lt;= anynum;x++) {     System.out.println(x); } </pre>	-1 0 1 2 3
<pre> for(int a = 1;a &lt;= 3;a++)    //outer loop {     for(int b = a;b &lt;= 3;b++)    //first inner loop     {         System.out.print(b);     }      if(a != 1)     {         for(int c = 1;c &lt; a;c++)    //second inner         {             System.out.print(c);         }     }      System.out.println(); } </pre>	123 231 312

On #2, notice how we can set the ending number to an *outside* variable, useful for controlling how many times the loop runs, for example, user input.

On #3, there is a nested for loop inside the for loop. The outer loop controls how many lines the loops executes: **The rows.** The inner loop controls how many in each of those lines the loop will run: **The columns.**

If you wanted to make a table in java, there has to be rows and columns. An outer for loops will move from row to row, while the inner loops controls each element in the row.