05.04 Virtual Lecture Notes: for Loops

Counting with a `for` Loop

It is often necessary to execute a loop a specific number of times. This is easily accomplished as indicated in the following example. When a loop counts from a low number to a higher number, it is referred to as ascending.

```
< 1>   public class CountingWithForLoops
< 2>   {
< 3>       public static void main(String [] args)
< 4>       {
< 5>           int theTerminator = 10;
< 6>
< 7>          for(int counter=1;counter<=theTerminator;counter++)
< 8>           {
< 9>               System.out.println("counter = " + counter);
<10>
<11>           }//end of for loop
<12>       }//end of main method
<13>   }//end of class
```

Type the code for the CountingWithForLoops class into BlueJ. Compile the program and fix any errors, then run the program and observe the output. Carefully study the following line-by-line explanation of the program.

Lines	
< 1>	declares a class name CountingWithForLoops.
< 2>	opening curly brace to start the class (matches with closing curly brace on line <13>).
< 3>	the **`main()`** method where program execution begins.
< 4>	opening curly brace to start the **`main()`** method (matches up with line <12>).
< 5>	declares and initializes **`theTerminator`** variable for the loop.
< 6>	white space to improve program readability.
< 7>	**`for`** loop, which initializes counter, tests the condition, and increments the counter.
< 8>	opening curly brace starting the **`for`** loop code block (matches up with line < 11>).
< 9>	prints the value of the **`counter`** variable.
<10>	white space to improve program readability.
<11>	closing curly brace ending the **`for`** loop code block (matches up with line < 8>).
<12>	closing curly brace to end the **`main()`** method (matches up with line < 4>).
<13>	closing curly brace to end the class (matches with opening curly brace on line < 2>).

Study the syntax of the **`for`** loop carefully. Notice that one statement on line <7> handles the three elements of every iterative structure.

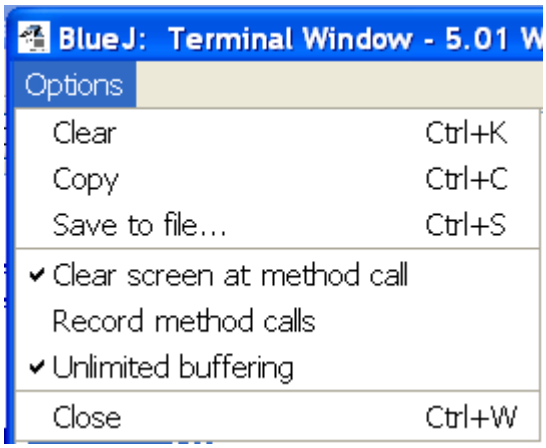
- **`int counter = 1`** initializes the counter variable.

- **`counter <= theTerminator`** tests the **`boolean`** condition to determine when to stop the loop.
- **`counter++`** increments the counter variable each time through the loop.

Type in the code with BlueJ and observe the performance and output of the simple counting program.

Modifications

1. Run the program again after changing the initial value of the **`theTerminator`** variable to various larger numbers. Record the amount of time (in seconds or minutes) a while loop takes to count to 100, 1,000, 10,000, etc.



The default option for the BlueJ Terminal Window does not clear the screen before each new execution. Change these settings to start with a fresh uncluttered screen for each run of your program.

Initially, the Terminal Window is set to display about 25 rows. If your output scrolls off the top of the screen, select Unlimited buffering form the Options menu.

2. Programmers often start loop counting variables at 0, but there are many times when you will need to start at a different number.

    - What happens if you change the initial value of the **`counter`** variable to a higher number, or a lower (negative) number?
    - What happens if you accidentally start the **`counter`** variable at a number larger than **`theTerminator`** variable?

3. Modify the program so that it counts backwards. What is the shortcut operator for decrement?

4. Modify the program so that it increments/decrements by a different integer (e.g., 2, 3, 15, etc.). You will no longer be able to use a shortcut increment operator, but what about the += or -= operators?

5. Change the **`counter`** and **`theTerminator`** variables to **`doubles`** and the values to decimals. Also, change the incrementation to a small decimal value (e.g., .5, .1, .01). Be aware that sometimes counting with decimals can lead to problems due to the way **`doubles`** values may be approximated.

6. What happens if you delete the curly braces around the block of code within the **`for`** loop?

7. What happens if you delete the parentheses around the condition in the **`for`** statement?

8. What happens if you delete one or more semicolon(s) within the **`for`** statement?

9. What happens if you add a semicolon to the end of the **`for`** statement, after the closing parentheses?

10. What happens if you don’t declare the type of the counter variable?