AL GHURAIR UNIVERSITY
College of Computing

CSC 209 JAVA I

Week 4, Control Statements: Part 2

Objectives:

- **The essentials of counter-controlled repetition.**
- **To use the `for` repetition statements to execute statements in a program repeatedly.**

Examples:

Example 1

Essentials of Counter-Controlled Repetition

This Example uses the `while` repetition statement to formalize the elements required to perform counter-controlled repetition. Counter-controlled repetition requires

1. a **control variable** (or loop counter)
2. the **initial value** of the control variable
3. the increment (or decrement) by which the control variable is modified each time through the loop (also known as **each iteration of the loop**)
4. the **loop-continuation condition** that determines whether looping should continue.

To see these elements of counter-controlled repetition, consider the application of Fig. 1, which uses a loop to display the numbers from 1 through 10.

Figure 1. Counter-controlled repetition with the `while` repetition statement.

```
1  // Fig. 5.1: WhileCounter.java
2  // Counter-controlled repetition with the while repetition statement.
3
4  public class WhileCounter
5  {
6     public static void main( String args[] )
7     {
8        int counter = 1; // declare and initialize control variable
9
10       while ( counter <= 10 ) // loop-continuation condition
11       {
12          System.out.printf( "%d ", counter );
13          ++counter; // increment control variable by 1
14       } // end while
15
16       System.out.println(); // output a newline
17    } // end main
18 } // end class WhileCounter
```

```
1 2 3 4 5 6 7 8 9 10
```

In `main` of Fig. 1 (lines 6 17), the elements of counter-controlled repetition are defined in lines 8, 10 and 13. Line 8 declares the control variable (`counter`) as an `int`, reserves space for it in memory and sets its initial value to `1`. Variable `counter` could also have been declared and initialized with the following local-variable declaration and assignment statements:

```
int counter; // declare counter
counter = 1 ; // initialize counter to 1
```

Line 12 in the `while` statement displays control variable `counter`'s value during each iteration of the loop. Line 13 **increments the control variable** by 1 for each iteration of the loop. The loop-continuation condition in the `while` (line 10) tests whether the value of the control variable is less than or equal to `10` (the final value for which the condition is `true`). Note that the program performs the body of this `while` even when the control variable is `10`. The loop terminates when the control variable exceeds `10` (i.e., `counter` becomes `11`).

for Repetition Statement

Java also provides the `for` repetition statement, which specifies the counter-controlled-repetition details in a single line of code. Fig.2 reimplements the application in Fig.1 using `for`.

Figure 2. Counter-controlled repetition with the `for` repetition statement.

```
1  // ForCounter.java
2  // Counter-controlled repetition with the for repetition statement.
3
4  public class ForCounter
5  {
6     public static void main( String args[] )
7     {
8        // for statement header includes initialization,
9        // loop-continuation condition and increment
10       for ( int counter = 1; counter <= 10; counter++ )
11          System.out.printf( "%d  ", counter );
12
13       System.out.println(); // output a newline
14    } // end main
15 } // end class ForCounter


1 2 3 4 5 6 7 8 9 10
```

The application's `main` method operates as follows: When the `for` statement (lines 10 and 11) begins executing, the control variable `counter` is declared and initialized to `1`. Next, the program checks the loop-continuation condition, `counter <= 10`, which is between the two required semicolons. Because the initial value of `counter` is `1`, the condition initially is true. Therefore, the body statement (line 11) displays control variable `counter`'s value, namely `1`. After executing the loop's body, the program increments `counter` in the expression `counter++`, which appears to the right of the second semicolon. Then the loop-continuation test is performed again to determine whether the program should continue with the next iteration of the loop. At this point, the control variable value is `2`, so the condition is still true (the final value is not exceeded) thus, the program performs the body statement again (i.e., the next iteration of the loop). This process continues until the numbers 1 through 10 have been displayed and the `counter`'s value becomes `11`, causing the loop-continuation test to fail and repetition to terminate (after 10 repetitions of the loop body at line 11). Then the program performs the first statement after the `for` in this case, line 13.

Note that Fig. 2 uses (in line 10) the loop-continuation condition `counter <= 10`. If the programmer incorrectly specified `counter < 10` as the condition, the loop would iterate only nine times. This mistake is a common logic error called an **off-by-one error**.

H.W Exercise:

Factorials are used frequently in probability problems. The factorial of a positive integer n (written n! and pronounced “n factorial") is equal to the product of the positive integers from 1 to n. Write an application that evaluates the factorials of the integers from 1 to 5. Display the results in tabular format. What difficulty might prevent you from calculating the factorial of 20?

Submit your answer as

Algorithm
Flowchart
Java Source code
Capture image of the output screen