COP 2250 – Java Programming I - Chapter 5 - Loops

# Loops

* Many computer programs require a block of code to be executed more than once.
* This can be done with loops.
* Java has three looping statements. All of them evaluate a Boolean expression to determine if the loop should perform another **iteration**, or cycle.
  1. while
  2. do-while
  3. for

# The while Loop

* A while loop will cycle as long as a conditional expression inside ( ) evaluates as true.
* Note that this is a pretest loop. If the condition is initially false, the entire loop is skipped.
* General form:

while( Boolean expression )

// **single** statement when expression is true

But it is recommended that you enclose even a single statement in a { } block as below:

while ( Boolean expression ) {

// statements **block** to repeat when expression is true

}

* Care must be taken to ensure that the Boolean expression will eventually test as false, else an **endless**, or infinite, loop results.

Try RepeatAdditionQuiz

Try GuessNumberOneTime

Try GuessNumber

Try SubtractionQuizLoop

# Exercise

Write a program to print the multiples of 7 between 90 and 60. Use a while loop and progress down from 90 to 60. Don’t delete this when done, we will add to it in Exercise 3.

# Counting Loops

* In programming, a counter is a variable that is **initialized to zero and incremented** every time that a certain condition is true.
* Integer types are always used for counter variables.
* Counters can be used to count the number of iterations that a loop runs through.
* A counter can be used with if to cause a block of code to execute based on the condition.

# Exercise 3

Modify Exercise 2. Display the multiples of 7 between 90 and 60 as before, but add a counter to count the number of multiples. Report the count after the loop ends.

Sentinel-Controlled Loops

* A while loop condition can be set to watch for a special value, called a **sentinel**, from input.
* Typically, the sentinel is used to stop the loop.

Try SentinelValue

# Exercise 4

Write an interactive program that asks the user to enter a number, and then displays the square root of the number. The program should continue until the user enters zero or a negative number. If the user does, the program should terminate. Use a sentinel.

**do-while Loops**

* The general format is: **do {**

**//statements to repeat**

**} while (test condition);**

* Note that this is a posttest loop. The test condition is at the end of the construct.
* This means that a do-while loop **always executes at least once**.
* Note also that it **ends with a semicolon**.

Try TestDoWhile

### **Exercise**

Write a Java program that uses a do-while loop to display the numbers from 1 – 10, all on one line of output.

# The for Statement

* The general syntax is:

**for(initialization expression; test condition; modifying expression)**

**// single statement to repeat**

* **Only the very next line** is repeated, but a block inside braces is recommended.
* To repeat a block of statements, enclose the block within { }.
* Many Java shops will demand this as a coding convention.

**for(initialization expression; test condition; modifying expression){**

**// multiple statements to repeat**

**}**

# Omitting Expressions in for Statements

* Any/all of the for loop expressions in the first line can be omitted, but there will always be two semicolons in this line.
* An infinite for loop can be created with:

**for ( ; ; ) {**

**// statements to repeat**

**}**

* Java allows you to declare the initialization variable inside a for statement.
* Instead of:

**int i;**

**for(i = 0; i < myNums.length; i++) {**

… you can use:

**for(int i = 0; i < myNums.length; i++) {**

* This latter form is recommended unless you need the variable later in your code (since it is not visible outside of the loop).

# Commas In for Statements

* Both the initialization and modifying expression may have commas.
* This can be done when a loop has **several counters** to initialize and update.
* Each expression must be capable of standing alone.

# Exercise

# Try this code. Copy-paste it into a class named LoopFor in Eclipse. Edit the package statement.

// LoopFor.java

package yourpackage;

public class LoopFor {

public static void main (String [] args) {

//int j;

for ( int i = 0, j = 10; i <= 10; i++, j-- ) {

System.out.println ( "i = " + i + ", j = " + j );

}

}

}

Which Loop to Use?

* See page 174-175.

### **Nested Loops**

* Nesting occurs when a loop contains other loop(s).
* Nested loops are useful for displaying **tabular** data.
* Try MultiplicationTable

### **Exercise**

Use nested for loops to generate this output. Note: **\t** is the tab escape code.

1.0 1.1 1.2 1.3 1.4 1.5

2.0 2.1 2.2 2.3 2.4 2.5

3.0 3.1 3.2 3.3 3.4 3.5

4.0 4.1 4.2 4.3 4.4 4.5

5.0 5.1 5.2 5.3 5.4 5.5

**Exercise**

Write a Java program that uses an endless loop to continuously dump Pythagorean triplets (with no number being greater than 100) to the screen. Repeats and multiples are okay. Create random numbers with the Math class. Pythagorean triplets are values that can make a right triangle, such as 3,4,5 and 5,12,13 etc. You can stop the endless loop in Eclipse by clicking the tiny red square icon.

**Minimizing Numeric Errors**

* Using floating-point types to control loops can cause approximation errors

Try TestSum

Try GreatestCommonDivisor

Try FutureTuition

Try Dec2Hex

Keywords break and continue

# The break Statement

* This statement will immediately **halt** an executing loop.
* Use it to prematurely break out of a loop when some condition becomes true inside the loop.
* Sometimes you will want several exit points for a loop.
* A selection statement is commonly used to conditionally execute the break statement.

### **The continue Statement**

* This statement causes a loop to **skip** to the end of the loop body.
* If the loop test condition is still true, the loop starts the next iteration.

Try TestBreak

Try TestContinue

Try Palindrome

Try PrimeNumber