Chapter 5 Loops

A loop can be used to tell a program to execute statements repeatedly; meaning those particular lines of code will be repeated until a certain condition is met and then it will stop the loop and continue to the next line of code.

Remember that a program is executed in consecutive sequence from the beginning to the end of the program.

Loops are constructs that control repeated executions of a block of statements that are within the program.

The parts of the loop construct:

loop body: the part of the loop that contains the statements to be repeated

iteration: a one-time execution of a loop body

loop-continuation-condition: aka loop control variable

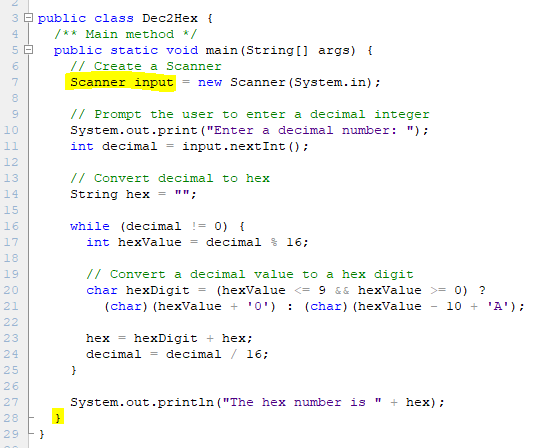
* a Boolean expression that controls the execution of the body;
* it is evaluated each time to determine if the loop body is to be executed;
* executes only if the continuation-condition is true
* must always appear inside the parentheses

The braces enclosing the loop body may be omitted only if the loop body contains one or no statement.

Scope of variables: the ability to access/reference a variable

The variables that we have been using so far are declared inside a method, then the scope of those variables are from where the variable is declared until the closing { brace.

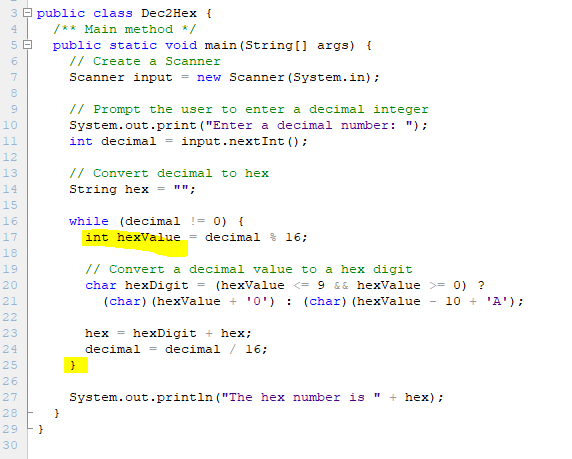
For example:



The Scanner variable input, it is declared on line 7 and can be accessed/referenced through line 27, the closing } is on line 28.

If a variable is declared inside a loop – its scope is only in that loop (it cannot be referenced to outside the loop).

For example:



The int variable, hexValue, is declared on line 17 and the closing } brace is on line 25; meaning hexValue can only be accessed from line 17 until line 24.

When you are developing programs:

* think before coding – think how to solve the problem without writing a program
  + create your algorithm
  + create your flowchart
* code incrementally – code one step at a time
  + Implement one line of code a time, compile the program, if necessary, run the program to see what that line of code does. If it is correct, then add the next line and continue to do so until the algorithm/flowchart is completed.
  + Keep checking to make sure you are solving the problem correctly

When the certain parts of the code need to be repeated, create a loop to do so.

Loop Design Strategies:

1. identify the statements that need to be repeated
   1. ensure those statements will evaluate correctly
2. encase those statements inside { } braces
   1. place the { prior to the first line of code that needs to be repeated
   2. place the } after the last line of code that needs to be repeated
3. wrap them in a simple loop

int num = 1; // this is ”priming the pump” by setting the loop control variable (lcv) to a value

while(num == 1){// test the lcv in such a way that the loop will execute just once

statements;…

num++; //modify the lcv to reach the test statement to become false

}

1. code the loop-continuation-condition: modify the loop completed in step 3 to how you need the code to execute for the program
   1. Make sure that the loop-continuation-condition can terminate! Or you will have an infinite loop!
   2. Be careful of the off-by-one error: making the loop execute an extra time or one time less than it should

The different kind of loops:

**counter**-controlled loop: a loop that is executed an exact number of times

* must eventually become false so that the loop can terminate
* not to become false makes an infinite loop (a loop that runs forever)

**Common error**: executing a loop 1 more or 1 less time. Called *off-by-one error*.

**Sentinel**-controlled loop: a loop’s execution that is controlled by a sentinel value

Sentinel Value: a special value that controls a loop; usually used as the signal to end the loop

**Caution**: do not use floating-point values for equality checking in a loop control. Because floating-point values are approximated for some values, using them could result in imprecise counter values and inaccurate results.

Java provides 3 types of loop statements:

1. while loops
2. do-while loops
3. for loops

**while Loop**

* executes statements repeatedly while the condition is true
* Checks the test for true first, if the test evaluates to true, then statements are executed
  + Pretest Loop
* while is a keyword

**do-while Loop**

* Same as the while loop, but performs the all the statements first, then checks if the condition is true
  + Posttest Loop
* Executes statements repeatedly as long as the loop-continuation-condition is true
* Use the do-while loop if you have statements inside the loop that must be executed *at least once*
* do is a keyword
* be careful with the syntax when using this loop

**for Loop**

* Concise syntax of a counter-controlled loop
  + for(int i = 0; i < 10; i++)
* Performs the initial action once

int i = 0;

* + initial-action: initializes the control loop variable
  + The control variable must be declared inside the control structure or prior to the loop
  + If the control variable is only for the loop, it is good practice to declare and assign it here
  + This can be a list of zero or more variables that are separated by commas
* Test condition, if the test evaluates to true, *then* executes the statements in the loop body

i < 10;

* + Pretest Loop
  + The loop-continuation-condition is a Boolean expression
  + It is evaluated after the initial-action and at the beginning of each iteration
  + When the test evaluates to true = executes loop body
  + When the test evaluates to false = terminates the loop
* Adjusts the loop-control variable – also called the *control variable*

i++

* + action-after-each-iteration: a statement that adjusts the control variable
  + has to make the loop-continuation-condition become false, otherwise it makes an infinite loop
* Repeats as long as the loop-continuation-condition is true
* for is a keyword

**Which loop to use?**

* The while and for loops are pretest loops
* The do-while is a posttest loop
* All are expressively equivalent
* for: when the number of repetitions is known
* while: when the repetition number is not known, or using a sentinel value
* do-while: when the statements will be executed at least once, and then maybe repeated

**Common errors:**

* adding a semicolon to the end of the for control structure
* forgetting the semicolon at the end of the while statement on the do-while loop

**Nested Loops:** consist of an outer and one or more inner loops.

Each time the outer loop is repeated; the inner loops are reentered, and are started anew.

**Keywords break and continue:**

break: terminates the loop

continue: ends the current iteration and program control goes to the end of the loop body

The difference – break terminates the loop, continue terminates the iteration

I will tell you this, if you are using break and continue in your loops, then you have not fully developed the logic. It is ok to use these keywords to develop the code, but you need to develop the logic fully of what needs to be accomplished and remove those keywords from the finished code.