Lesson 17: The while & do-while Loops

The *while* loop is basically the same as the *for*-loop except the **initializing** and **step** expressions are **not** part of the *while*-loop basic structure. In the following code we show the basic structure (skeleton) of the *while*-loop:

```
while( j <= 79 )
{
      … some code that we want repeated…
}
```

We notice in the above code that the only part similar to the *for*-loop is the **control expression** *j <= 79*. The **initializing** and **step** expressions are **absent**. As with the *for*-loop, the *while*-loop keeps repeating **as long as the control statement is *true***.

Summing numbers:

Now, let's actually do something with a *while*-loop. We will begin with a *for*-loop that sums the numbers from 3 to 79 and then perform this same task with a *while*-loop:

```
int sum = 0, j;
for (j = 3; j <= 79; j++)
{
      sum = sum + j;
}
System.out.println(sum); //3157
```

An equivalent *while*-loop:

Here's a *while*-loop that does the same thing:

```
int sum = 0;
int j = 3;      //initializing expression…not part of loop.
while (j <= 79)//control expression:fundamental
                //part of loop
{
      sum = sum + j;
      j++;      //step expression…we have to remember to
                //put this in.
                //It's not part of the basic "skeleton"
                //of a while-loop.
}
System.out.println(sum);
```

The *do-while* loop:

A *do-while* loop is exactly the same as a *while*-loop except the control expression is at the **bottom** of the loop rather that at the **top** as it is with the *while*-loop. Following is the skeleton of a *do-while* loop:

```
do
{
      …some code that gets repeated…
}
```

```
while( j<= 79);
```

Note that *while* is not inside the braces. Also, notice the **semicolon**. It is a common mistake to leave it off.
We will now re-implement the *for*-loop above that sums from 3 to 79 as a *do-while* loop:

```
int sum = 0;
int j = 3; //initializing expression
do
{
     sum = sum + j;
j++; //step expression
}while (j <= 79); //control expression
System.out.println(sum); //3157
```

What's the difference?

The main difference between the *while* loop and the *do-while* loop is **where** the test for staying in the loop is made (the control expression).

while-loop → test is at the **top** of the loop
do-while-loop → test is at the **bottom** of the loop

The *break* statement:

If *break* is encountered inside a loop, the loop terminates regardless of the status of the control statement. Code execution continues with the first line of code following the loop structure.

The *continue* statement:

If *continue* is encountered inside **any** loop (*for*, *while*, or *do-while*), all remaining code in the loop is skipped for this particular iteration; however, looping continues in accordance with the control expression.
This is illustrated with the following code:

```
int j = 0, boxer =11;
while(j <10)
{
     j++;
     if (j != 5)
     {
          continue;
     }
     boxer = boxer + j;
}
System.out.println(boxer); //16
```

No braces:

If a *while* loop has no braces then it is understood that **only** the very next line of code (or structure such as another loop, *switch*, or *if* structure) is to be iterated (repeated).

Consider the following code examples:

```
while(control expression) ➔
     pk = pk +2;
     x = 97;
```

```
while(control expression)
     {
          pk = pk +2;
     }
     x = 97;
```