Transcript: *for* Statement

>> Now let's consider the for statement, or for loop. It's similar to a while loop, in that the condition is checked prior to entering the body. But the for loop is particularly well suited for iterating a specific number of times through the loop, or over a range of values.

The basic form looks like this, it's for and then we've got an initialization. This is all in parentheses there, semicolon followed by a termination and this would be a Boolean expression. And again, this is checked before each iteration. Initialization is done before the first iteration. And then, finally, we've got the increment.

And that's actually performed after each iteration. So it's for some initialization of a variable like i, and then as long as say i is less than equal to 10, you're gonna go through the loop and each time you go through the loop, you would increment. Similar to what we've done in a lot of our while loops, but this collects all the information right here in the for header, if you will.

So suppose we wanted to calculate the sum of all numbers from 1 to n, this is one of those instances where we're gonna go through the, loop for a, particular number of times. So we would need to initialize a sum to 0, set up an index to count from 1 to n.

Since we want to go through this loop n times. And then on each iteration we would add the current index to sum, and then we would increment the index. So we could do this in a while loop or we could do it in, in, in a for loop.

Eventually we would wanna break out of the loop of course when the index exceeds n. So let's look at how we would do this in a for statement. Here, we would set n equal to 5. This is gonna be how many times we wanna go through the loop.

Recall we're gonna sum up 1 through n, 1, 2, 3, 4, 5, dot dot to n. And then we set sum equal to 0. And then we say for i, for int i, equal 1. As long as i is less than or equal to n, we're gonna go through this loop.

And each time we go through the loop, we'll increment i by 1. And in the body of the loop, we're just adding i to sum and therefore, we're keeping a running total. Eventually, i is greater than n and we break out of the loop. So let's look at an example.

This example does addition and also multiplication of of a range of numbers. So this is AddMultiply, this is in the Examples folder of course. So looking at this, we're going to, here is the for loop for adding a 1 to n. And again, we set sum equal to 0.

And then all of the nice information about the loop is collected right here in the for header, if you will. So, for int i = 1, as long as i <= n. We're gonna go through this loop and we'll increment by one after each iteration. So again, sum gets incremented or added, i gets added to sum there each time we go through.

This statement sum plus equals is equivalent to sum = sum + i. And then after the loop, we'll, we'll actually print out the sum. And his is the equivalent while loop. Study these. These are, these are very similar. We, we've got but in the while loop, we actually set the index outside the loop and we increment inside the loop, whereas in the for, all this is collected up front.

and you know sort of immediately from looking at the for header there, what you're gonna be doing in the loop. And other things in this, we were actually multiplying, here we're multiplying k = 1 as long as k <= n, and this is using a long instead of an int for, for the product itself.

And then down here, we're using a, looks like we're using a, a double for the product. And we'll see how multiplying gets out of hand pretty quickly. Because the numbers get large. All right, let's run this. And we've got a canvas here, let's open the canvas. And we'll see how these, these, these run here.

So I'm going to just run down to the first break point there. Let's see, we need to move that break point to there. There we go. So we'll just run down to there and we're waiting for input here. So we're gonna do this from 1 to let's do it from 1 to 25.

We'll add up those numbers. There's 1 to 25. And then this is us going through the loop. I'm gonna move this over a little bit. In fact, we'll just let this, play in the canvas. We're looping pretty fast and now we're doing it for the while loop version of it.

These should be the same, of course. And then here's the product version. Notice this number got big and it even turned to negative, so that was bad. This is not gonna be a good thing. Here's the double. This will be a more accurate version of it. And so on.

In fact, here, here is what what we've got here the and the exact numbers here looking at it. Just using some decimal format things, and big decimal. But point is, we we were able to go through these loops pretty quickly and, and sum these things up. Now let's write one more time and we'll run it in the canvas again.

And let's go ahead and run down that first break point. And let's give it 100 this time. And we'll launch away. So this time, notice though, we're gonna get the, the sums are gonna be OK, because adding, we don't get too big numbers. But the products are gonna get out of hand here, and we'll see them go negative and all sorts of things, which is bad.

So you do have to pay attention if you're gonna have large numbers, you may wanna use doubles. So here's the equivalent while, and we're just looping through this until j exceeds 100. It's at 30 well, you see it in both places here. There's at 47, 40, 50, and so on.

We'll just watch this loop through. And we're about to reach 100. And then the product, this is where we'll see things sort of get out of hand, if you will. So there goes the product, and notice it's turning negative and positive, this is all what we call overflow, and eventually just become zero.

So this is not a good thing, we need to probably use a double here which'll let us have a, you know, floating point representation, so at least we'll get 15 significant digits or so. And notice, when it got big enough we went to what we'd call scientific notation.

And what this is, it's a and an exponent. And we'll see that down here in the bottom when it, when it prints out in our run IO. And you can see what the number is over here in the, in the canvas. It's, it's a relatively large number there.

So eventually it finished, and we can see the, the sum there from 1 to 100. That worked both in the for statement, and the while version was 5,050 for each. And then the product of multiplying the numbers 1 up to 100 using a long, which is 64 bits, whereas an int would be 32.

We ended up with a 0 there because of the it overflowed actually. And then when we changed to a double, we ended up with 9.33 and and so on and an expon, exponent of 157. So this was actually, a more accurate number. And in Java, you can use big decimal and things like that to get, even a more accurate version if you, if you were interested in that.

And there you see some of these over here. So this was just a little demo of, of the for and while to go over a range of numbers or through, through loop a a specific number of times. So let's go back to our slides. Just to sum up.

This is what a, a for loop looks like, and this is the equivalent while loop. And again, they're very similar. They do the same thing. The difference is that for the while loop, the initialization here, j = 1, and the increment are sort of not in the header.

You've got to do those. And some place in here, this, since j is our index, we need to increment j. And, and the for loop does a nice job of collecting all these. So here we've got the initialization, the Boolean, which we of course had in the while as well.

And then we also have the increment. And so, going into the for loop, you sort of know what's gonna be expected in the body no matter how long or how large the body is there.