Transcript: *While* Statements

>> Now let's take a look at the details of the while statement. Or as it's commonly called, while loop. A while statement will continually execute as one statement or a block of statements as long as its condition, which is a Boolean expression, is true. That is, it repeats the statement or block until the condition becomes false.

The general form of the while looks like this. Reserve word while. And then that's followed in parentheses by a Boolean expression. And this could have variables in it or it could just be a method call that returns a Boolean value. And then this is followed by one statement or, as we're gonna do, a block of statements.

Actually works with one statement, but our coding style and good programming practice says that we wanna use a block. So, we'll have an open parenthesis and then the code to perform, that's to be performed during each iteration, followed by a close parenthesis. So it's simply while, boolean expression, all of the code and then close brace.

Now any variables, that are in this boolean expression need to be initialized before the loop. Compiler will catch this if they're not. And then usually, the code in the loop alters the value of these expressions, rather of the variables, so that the condition eventually becomes false and the loop will terminate.

Otherwise you'll end up with an infinite loop. So let's look at an example. Here we've got a little code that prints all the numbers from one to ten. Now, as you know, we could just have separate print statements, ten print statements. So and print, one, two, three, and so on.

In this case, we wanna use the while statement. So up front we initialize int count to 1, and then we have while in our boolean expression, the condition that we're after, is long as count less than or equal to 10 we're gonna go through this block of code.

And in the block we simply print out count, System.out.println(count);. And then we increment count. So, count++ is the line of code that alters the variable up in the boolean expression. And that's important, because count less than or equal to 10, should eventually become false. So, after we go through this one time with count equal 1, we hit the closing brace, and it goes back increm, after counts been incremented.

And now, count will be two, that's still less than or equal to 10, and we repeat the block. And that simply continues until that boolean expression is false. Now, the debugger is a useful tool for seeing the control flow in loops. So, let's, let's take a look at count 1.

So here's our code, and again, it's what we saw on the slide. Int count = 1, equals 0 and then here's our loop that we're gonna continue through as long as count is less than or equal to 10. Now if I just run this. We see the output there, 1 through 10, and after the loop terminated, we actually printed the the stream done.

So let's set a break point and we'll just step through this and see it go through the loop in slow motion, if you will. All right, so we're about to set count equal to 1. So I'm gonna step. And now we see over in the debug variables tab count is equal to 1.

And I'm in the loop, that was we actually check the condition there to see if count is less or equal to 10. It was, so we go into the loop. And just as a side not here, if the first time you try to enter a while if this condition is false, it just skips the loop.

So this loop is gonna iterate zero or more times. It actually may not go through the code. So, here's what we print out to count, and there's 1, you see down here in the run IO tab. And we increment count. So count is now 2. And then we go through the loop again and I'm just continually stepping here.

We see count is now 6, 7, 8, 9, and now it is 10. Now, we're asking here in the expression, is 10 less than or equal to 10? Well, it's true, because we're actually checking for equality as well. So we're gonna go through one more time, and now count is 11.

So when we step here, we should actually jump beyond the loop to that print statement on line 22, and there we are. And one more step, we print done and we're finished. So, that's just a, a little, look at loop. You should practice with these loops, on your own.

They're quite simple, just remember there's an expression here at the top. And as long as it's true you're gonna continually execute the block there. So now let's look at a little more detailed example. Little more I guess I should say complicated example. Here we've got a NumberSet class which has int fields for low and high, and these are tended intended to hold positive integers, so we're just gonna use positive numbers here.

And the NumberSet class includes two methods, findEvensBetween and findCommonDivisors. findEvensBetween just returns a string that includes all the even numbers between the values of low and high inclusive. FindCommonDivisors returns a string that includes all the positive common divisors of low and high. So we're just looking at the two numbers and determine which are their common divisors.

Now, each of these methods requires a while loop to perform its work. And so, we're gonna look at the the details of these in particular, findEvensBetween, and then in the source code, we'll look at both methods there. So here we've got findEvensBetween. Now detail, again we wanna return a string that includes all the even numbers between the values of the low and high fields, inclusive.

And so here's our strategy, we're gonna begin with a candidate number that's equal to low. And then we're going to, have ourselves a while loop that, with a boolean expression is, while the candidate is less than or equal to high, we're gonna go through the loop. And, and, we've got an if statement in there that says if candidate, is divisible by 2, then we want a concatenate candidate.

Along with the space we'll see, to the end of the string that's gonna be returned. Then we increment candidate, and we go back and see if candidate is still less than or equal to high. And the, by doing this we loop through checking all of the values between low and high.

And we're essentially just concatenating, building a string consisting of all the ones that were divisible by 2. Eventually the loop ends because we're incrementing candidate each time. And then it jumps out. And then, and and then we would return the string that we, we did. Now the other method findCommonDivisors, we'll talk about that as we look at the source code.

So, here's NumbersSet. I'm gonna move this down a little bit so we can see it a little more. Notice we have two fields, low and high. And then we've got a constructor that takes in two ints, number1 and number2. And the both these should be positive integers. And then we check to see which one is less than, the other.

If number1 is less than number2 then it gets assigned to low. And number2 goes to high, otherwise we reverse them. The idea here is in the constructor we want low to be the lower value and high to be the upper value, of course. And we've got to get, or forget low or get high.

Then here's our little routine for finding EvensBetween. So, we initialize the string. This will be the string that we return, called evens. And notice we initialize it to an empty string. This is a string of length zero. And then we, declare int candidate equal low. So again, we're gonna start with a low number.

And then a while loop that says while candidate is less than or equal to high, we wanna do this block of statements here. And there's really just a, an if statement in the increment. And so our if statement says, if candidate remainder two, equals zero. So recall this is the remainder operator.

And we just divide candidate by two and look at the remainder. So if there's no remainder, it was divisible by two, meaning it's an even number. So if that's true, we simply say evens plus equal candidate and a space. So what happens here, candidate, which is a, a number, get's added to evens.

And then we actually concatenate a space with that. So we'll have a space between each one, if you will. Now, looking at and we simply go through the loop until this get's to a candidate exceeds high. And we, we return evens. So, let's go ahead and look at common divisors, as well, while we're here.

Here, we've got two positive numbers. And, again, one is low and one is high. And we want to find the common divisors. And a common divisor just means when or a divisor. Here it means that when you divide by zero, the remainder is zero. So when you divide the number by zero.

So, in this case, we've got the while. We set the divisor to one, and we're gonna get, let it range up to low. So, the divisors that are in common between these two can, you know, range from one up to low. Anything greater than low, of course, will never divide low.

And so, that's what we're after there. So, in our if statement, we say, low remainder divisor, so we divide low by divisor and see if the remainder equals zero. And, high divided by the same divisor. Let me see if it's equal to zero. If they're both, both equal to zero, then it's a common divisor, if you will.

And again, we do the same thing. We just add divisor to the string here, common divisors, and, and add a space, in with that. And then we increment divisors. Notice it started at 1, and so it's gonna go up, it's gonna loop through this until we get to low.

And then after it exceeds low, of course, we'll stop. Fact, lets set a, we'll set a break point in in find evens, and we'll just take a look at that. And let's go ahead and set a break point also in find common divisors. OK, and then let's go to number drivers number set driver.

And here we're just creating a number set. This example, 3 and 15. And then here's one, 10 and 20. And then, in this statement here at line 14, number set 1. That's 3 and 15, we call findEvensBetween. And then we print those out. We print the answer out there.

And then, in these next ones we actually just call the method. Here's wh, here we're calling number set 1 dot find common divisors. We're actually calling that just in the print statement. Then we do that for number set two and, and so on. Let's just run this and look at the output.

So here we've got, evens between 3 and 15. And we see they are 4, 6, 8, 10, 12, and 14, sort of like we'd expect, hopefully. And then the common divisors of 3 and 15 are just 1 and 3. These are the numbers that divide, the positive numbers that divide 3 and 15 evenly, if you will without a remainder.

And, and then here's a 10, and 20. Here's the, even numbers between 10 and 20, 10, 12, 14, 16, 18, 20, and then the common divisors are 1, 2, 5, and 10. So each of these numbers, 5 goes into 10 evenly, and 5 also goes into 20 evenly, and so on, so, those are common divisors.

So now let's run this, with a break point set. In fact, let me set, yeah. Let's, I think this will stop for us here, where we want it. All right, so here we are. We're looking at, we're looking at 3 and 15. Those are our, our two high and low values, if you will.

And so let's go through her and see how evens grows. I'm gonna step. And so the first thing it's looking at is candidate three. And it wants to know if this is, even or not. It's, it's less than high. And is it even? Well, three we know is not even so it didn't get added.

And so now we're looking at candidate four, four is even, so it should get added. And so we see evens now includes 4 and a space, which is what we intended there. And we'll loop again, 5 didn't get added, how about 6? 6 gets added, there it is.

And we just loop through here and I'll just pull this out. And you can see that evens are growing there. Each time we add an even, there's one there, there's 10. And this is gonna be 12 gets added. And so on, OK? 14 and that should be it since we're only going to 15.

So now we're gonna go out of the loop and return evens. And we're back. Let's make that go away. And we should see answer here set to all of those even numbers. And then we print those out, and so on. And now I'm gonna go to the next break point, and this is us, program is now looking, for the common divisors of 3 and 15.

And again, we set the, the first candidate divisor, if you will, to 1. We begin with 1, and we're gonna check up through low. And so our while loop runs, divisor less than or equal to low, and we're gonna go through and check to see if, again, low divided by divisor, if that remainder is zero, and high divided by divisor, if that remainder is zero.

If it is, then we just add divisor to the common divisors string, and also add a string there. Okay, so let's, let's step. And so we're looking at 1. 1 is a common divisor, it gets added. Now we're looking at 2, 2 is not a common divisor of 3 and 15, so it gets skipped.

And now we're up to 3. It is, OK. But notice low here is 3, and so divisor's 4, so we're about to leave the loop. And then we just simply return common divisors, and it gets printed out in the statement there. Ok, I'll leave it as an exercise for you to explore this.

Simply set break points and, and step through this, and, and the, the focus should be on studying these while loops here. The one that, does common divisors and the one that does evens. Now just to reiterate here when we were concatenating the string, our candidate here is a, an integer.

And we're concatenating it with a space. So you know the result of concatenating an int and a, and a string is gonna be a string. And so, then that gets, concatenated with evens there. And same thing down here, we concatenated the space with divisor and so the result of that addition, so to speak, really concatenation, is that we end up with a string with a number space in it and then that gets concatenated to common divisors.

OK, hope this has been useful. We will look forward to seeing you in the next segment.