Transcript: do-while Statement

>> Now let's consider the do-while statement. Also called the do-while loop. It's similar to a while loop except the Boolean expression is evaluated at the end of the loop. The do-while is a post-test loop, whereas the while statement that we've seen before is a pre-test loop. So this means that the body of the do-while loop is always going to be executed at least once, regardless of whether the Boolean expression is true.

The general form is the reserved word do. And then this is followed by the loop body enclosed in a block. So we have open brace and eventually a closed brace. And now, as you enter the loop the code in the body will always be executed, again, at least once.

And then the Boolean expression is evaluated here. And if it's true, you loop back and loop and, and do their body again. And you continue to do this of course until the while is, is false. So a good use of the do-while is evaluating user input. Suppose the user is to enter either a y or an n, this would be for yes or no.

And you wanna repeat the request until you actually get a y or an n. Well, here's the code that would do that. We set up the scanner on System.in and then we, we've got a string to retrieve the the input. It's y or n. And then we've got this do-while loop.

And so we say do, and then inside, the first thing we wanna do inside the loop is print out Continue? Enter y or n, this is where we're asking the user. And then we read the next line. We trim it up and this is in case the user enter y space, space.

We're gonna accept that as OK, the trim will take off that trailing white space. And then we say, yorN, yOrN, that's the string that we read in here. While it's not equal to y and not equal to n, we wanna loop. And so, we'll keep looping as long as what was read in is not equal to y and not equal to n.

But once we gets, get a y or an n we'll continue on. So let's look at the the code for this in our environment. So here, here is the code we just saw there on the slide. And now I'm going to run this in debug mode, So we're, we're declaring yorN, we're gonna set that equal to an empty string.

And then next step and now we print out enter a y or n. And the next statement is actually waiting for input. So I'm going to key in a y here. In fact, I'll key in y and some spaces since we're gonna trim it up there. And when I hit Enter and then we step again.

We'll see over here that y or n is y. And so, this statement here is gonna be false. Because I wanna know if it's not y and not n, well that's false, it is y. And so when we step, we go on through and notice in the print statement, we're using our conditional operators.

Our ternary operator, and we're saying here yorN equals y, if that's true we're gonna say yes, y for yes. Otherwise we'll say n for no. And we know it's one of these because we got out of the while statement that ensured at least one of them. And so I'll step and there we print out the selection was y for yes.

So, let's end this and we'll rerun it in debug mode. So this time we go down here and I'm going to enter an a. And we get to the while and this is gonna be false. Rather it's gonna be true. It's not equal to y and not equal to n because it's equal to a.

And so when we step, we actually go back through the loop and we ask again. And this time, let's give it a z. Then again, we go back through the loop and so on. This time let's give it, an n. And we should get out of a loop this time because that's acceptable.

The while is asking is it not a y and not an n, that's gonna be false because it is an n we see over here in the debug tab. And so when I step, we get out of the while loop. And again, we print, this time the conditional here, n, for no.

The selection was n for no, okay? Now let's run this, just run it. Not in the debugger, and you'll see how fast this works. So it's wanting it in, and I'm gonna key in a 1, a 2, a 3, a 4, a c, an s, all these are not y and not, n.

Eventually though I key in something it's looking for, like a y and we get out of the loop and we finish. So let's look at the second example. It's similar, except in this case we've decided to put a maximum number of tries here. Up here we have a, a constant static final int max tries=3.

So it's essentially the same logic, but a bit more to actually count the tries there. And we've got a Boolean variable is valid and that's what we're checking down here. And, again if it's a y or an n we'll say it's valid to true. And as long as it's not true, it wasn't valid, we continue to loop.

And then we have an if statement where if it's not valid, we count the tries there. And once the tries = MAX_TRIES, we actually print out too many attempts aborting and we break out of the loop. So this actually is an example of using the break statement to get out of a while loop as well.

So, I'll just run this and, and we'll see how this works. I'll let you step through it in debug mode to, to actually test it on your own there. So I'll key in 1 and it says here Error:1 was entered. That's, this right here. It wasn't too many tries, but it wasn't valid.

So here we printed out Error: and whatever the user entered and it was a 1. And then, was entered, as you see there. So now let's enter a z. We say Error: z was entered. And we're gonna try one more time with a, with an x and it says too many attempts aborting.

So this was when, Max, tries=MAX_TRIES, and actually abort it there. So, just keep in mind the, the do while loop is a great loop. Especially if you wanna go through the code at least once, which usually you do, if you're getting input from a user. And then you might continue to ask, like we did here in, in this example.

And also the more simple version of this where we weren't counting tries. You know, we continue to go through the loop until we get a valid, entry, so to speak.