Transcript: Character Sets

>> Now, let's consider character sets. A character set is an ordered list of characters and, and a character represents a unique number. A char variable in Java can store any character from the Unicode character set. And, Unicode characters use 16 bits to represent a character. And this allows for 65,536 unique characters, essentially 2 to the 16.

It's an international character set, and it contains symbols and characters from many of the world languages. Think of the Asian languages like, Japanese and, and Chinese, Korean and, and so on. When you have time you might experiment with char, and string literals, in interactions and see how they work.

Turns out you can concatenate a char and a string, or a string and a char. The ASCII character set is older and smaller than Unicode, but still quite popular. ASCII stands for American Standard Code for Information Interchange. It's only 255 characters or 256, I guess. It start, it goes 0 to 255, it's 8 bits, an 8-bit character set.

And it includes all of the things we find on our keyboards. Upper case letters, lower case, punctuation, the digits, special symbols, control characters, and so on. Now let's look at operator precedence. We talked about that a little bit, but, as you know, operators can be combined into complex expressions.

Here's one that's slightly more complex, maybe, than we've been looking at. Here we've got result = total + count / max- offset. Now operators have a precedence, which determines the order in which they're evaluated. Multiplication, division, and remainder are evaluated before addition, subtraction, and string concatenation. We learned that a long time ago, probably in elementary school.

And arithmetic operators with the same precedence, are evaluated from left to right. But we can use parentheses to force a particular order. So for this expression that we just looked at, notice we've got addition, division and subtraction. And the order of precedence says, when, when this is evaluated, we're gonna do the division before the addition and subtraction.

So, first thing that would happen here is, count would be divided by max. And then we would go back and work left to right, because addition and subtraction are the same order of precedence. And so we would say total plus, that result that we got from the division, and then finally, we would subtract offset.

Here is an example of doing addition among five different operands. And we are adding them all together, a + b + c + d + e. The order that they're done is the first one, the second one, third and fourth, and you see we just work left to right.

Here we've got multiplication and division thrown in. They're done before addition and subtraction. So here we would add, we would multiply b times c. Then we would divide d by e, and then we would go back and, and do the addition. We would add a to that result we got earlier.

And then finally, we would subtract that d divided by e result of the rest of it there. Here's an example where we throw in an parenthesis around addition. So, even though addition is done usually after multiplication or division. Here we would do it first, then we would go back, and looks like the next order of precedence is gonna be our division.

Followed by the remainder, and then finally, we would do the subtraction. In this last example we have lots of parentheses. And so when, when these are nested we do the inside, the inside parentheses set first. So here we would subtract e from d. Then we would add c to that, multiply by b, and finally all that would be divided into a.

So let's revisit assignment a little bit. The assignment operator itself, it's, it's an operator that takes a right side and, and assigns it to the left side, if you will. It has a lower precedence than the arithmetic operators. So in this situation here, we would do the division, do the multiplication, add the two results there, and then we would take all of that, and do the assignment operator, so it would be done last.

Now, let's revisit the increment and decrement operators. They can be applied in postfix form, as in count++. And, in this case, if there's an expression being evaluated that contains count++. Or think of count++ by itself the old value is used, and then the increment takes place. Notice the ++ is after count, and that's the way you can think about that.

We're gonna use the old value, and then we're gonna increment. The prefix form is ++count. In this case, the increment takes place first, and then we use a new value in the expression. When used as part of a larger expression, these two forms can have different effects, and we'll see that here in an example of program.

Transcript: Character Sets

Let's look at the increment operator example. This is a simple little program where we're going to, we've got count set to 0 up at the front. And then we're gonna add 10, to count++. We're just gonna print out 10 + count++ =, but the arithmetic we're gonna go, do is 10 + count++.

And then the second one, we're gonna do 10 + ++count. And so one is the, first one is the postfix, and the second one is the prefix operator. And then we'll finish up with just a ++count and a count++ is individual statements. We'll see that those are essentially the same.

So let's run this in the debugger. First thing we do is set count to 0. I'm gonna step, so over in the Variables tab we see count = 0. And then we're gonna print out -- print that out, count = 0. And now at line 15, when we execute this, we're going to add 10 to count++.

Since this is a postfix operator, it's going to use the old value of count to add to 10, and then it's gonna increment count. So we're gonna add 10 to 0 essentially, and come out with 10, but count is gonna go up by 1. So when I step, we see that down in the bottom here, 10 + count ++ = 10.

So we use the old value, but then count went up by 1. So in this next statement, here we're going to say, 10+ ++count. So this says, take count and let's increment it first. And then we'll use it in the evaluation of this expression here. So count is currently 1, ++count is gonna add 1 to that, so we're gonna have 2.

So when this statement is executed we're gonna add 10 + 2, and get 12. We see that down in the Run I/O window. And then notice here, we've got ++ count. And so, that's just gonna add 1 to count. We see it's 2 right now. So when I step, it goes to 3.

In the postfix version, the same result is there. Now the reason this was the case is, these were sort of standalone statements, rather than being part of a large expression. Say, on the left hand side of a, of a, an assignment statement, or in a print statement. If we go down to Interactions, we can visit this a little bit more here.

So, count equals 4, so if I say count ++. Now this is an expression. I'm not putting a semicolon on the end. So when I hit Return, I should get the old value of count, which is 4, and then count is gonna go up by 1. So notice we printed 4 here in Interactions, but over in the Variables tab, count went up by 1, so it's 5 now.

If I say ++count, again this is just an expression. In this case, we're gonna add 1 to count. Count is currently 5 so, we are going to add 1 to it and get 6. And so we'll print out 6, and it will become 6 over in the Variables tab.

OK, you'll just need to experiment with this a little bit to get the hang of the, prefix and, and postfix. In Interactions, you can do this and, and see it pretty easily though. All right, let's say a few more words about assignment operators. There are many of those in Java.

And they look like this, so it's really just a shortcut form, if you will. +=, for instance, if we say x += y. We're gonna say x + y, and the result then equals x. And so that's, that's what we've got the equivalent to. And you can look down through these, they're all, all like that.

Here's x /= y, and that's essentially x / y and the result. And then is assigned back to x, and you, and you see that over on the right there. Now the right side, right hand side of one of these can be a complex, a bit. And when that's the case, the entire right hand side has evaluated first.

And then the result is combined with the original value. So here's an example here result /=, and we've got total-MIN % num. And so this is gonna be equivalent to result being divided by this whole bit here on the right. And so essentially, we're gonna need to put parentheses around that, to make sure it's divided by that whole thing.

And then the result is assigned back to the variable result. And so this is equivalent to this. Essentially just what you think, the only thing that's sort of you need to be careful of is, there are implied parentheses around all of this, to make sure it all gets done there.