Transcript: Revisiting Assignment

>> Now let's take another look at assignment. Recall assignment simply takes a copy of a value and stores it in a variable. For primitive types we might have num1 equal of 38 and num2 equal to 96. Then if we have the assignment statement num2 = num1. We end up taking 38 and moving it down into num2, 38 is still in num1.

What there is a copy of it in num2. So we have something like this, we have 38 in two places and and that's what we expect. Now for reference assignment it's a bit different. Here assignment copies the address rather than the actual object itself. So, here we got name1 references, Steve Jobs, and name2 references, Steve Wozniak.

When we make the assignment, name1 equals name, rather name 2 equals name1, again, we're gonna take the address here for name one and we're gonna sign In it to name2, and then after we do that, we've got this situation. We've got both name1 and name2 referencing Steve Jobs.

Now when that happens, we have to wonder about what happened to the string Steve Wozniak. Well, if there's another variable referencing it, it's still around. However, if no reference, no variable is referencing the string Steve Wozniak, it's gone. And it's, just becomes a candidate for a garbage collection.

Now, let's take a look at this and in interactions. We'll just duplicate this, sort of see how that works there, so let's move, move over here. So we're gonna declare int num1 and let's let that equal to 38. So we see up on your work bench that num1 equal to 38 and now let's say, int num2 L equal to 96, and we got that there.

Now let's create our string. Now we got string name 1. Let's set set that to equal Steve Jobs. Now we need string name2, I'm gonna use the up up arrow and get what I just keyed in there back. I'm just gonna go over here and change name1 to name2.

And Steve and Jobs to Wozniak, okay? Now, when I when I no notice that name one is pointing at Steve Jobs let's give ourselves a little more room here. And notice that's object 112, it's a string, if we give ourselves some more room we see that. And now when I hit Enter over on the workbench, creating this new string name2.

It's Steve Wozniak and notice it's object 122. So it's two different objects there. Now if I do the assignment statements, I'll just skip a line here, and I'm gonna say in this case I'm gonna say num2 = num1. So over here I'm expecting this 38 to move down here to 96.

To where the 96 is. So when I hit Enter, sure now, we see that num2 is updated. This is now 38 and notice there's 38 in both places, that's the idea. Now if I do the same thing with the name2. Name2 = name1. Now here, notice, name1 is pointing at Steve Jobs object 112, and this is Wozniak object 122.

Now, when we're done here, both of them should be pointing at object 112, so that sort of goes away and we see that there. And notice the different notation if you will, and the on workbench here. For primitives it just uses the equal sign, so num 1 = 38, however for reference types its got the variable name and pointer to the actual object.

In this case, they're both pointing at the same object. This isn't two different objects with the name, Steve Jobs. It's both the same objects since it's both 112. So back to our slides here. This last slide simply captures what we did there. So in the beginning there we had 38 and 96 for the n types and then we over here, we, we declared the two string types, and, and we got those up here.

And notice, in this case running in, interactions, they got a signed object, 381 and 387. And then we step one more time, or we actually sorry, do the assignment. So this is num2 gets assigned num1 and name2 gets assigned name2, so over here we end up with them pointing at, at the same object if you will.

Next we'll talk about aliases.