AUBURN UNIVERSITY

Transcript: Default Rounding in Java

>> Now lets talk about default rounding in Java. Java uses something called half- even rounding for formatting, and that means it rounds toward the nearest neighbor, unless both neighbors are equal distance, in which case it rounds toward the even neighbor. This is also called bankers rounding. Essentially, what this means is if you're rounding a digit, if the digit is below five, it's gonna round down.

If it's above five, it rounds up. If it's exactly five, then it's going to round toward the even neighbor. And this mean that it's going to round up half the time and round down half the time thus bankers rounding. Let's look at some examples. Here we've got the value 12.374 and we're gonna use the decimal format pattern #.##.

That means we want two decimal places. Here, notice we've got three. And so we're gonna lose this last digit, 4, and since 4 is less than 5, this becomes 12.37. In our next example, we've got 12.376. Six is above five and so that's gonna round up to 12.38.

Now in this example, 12.375, we've got exactly five. And so we're gonna round toward the even, the nearest even neighbor. And in this case, 8 would be an, the even neighbor. And so, we round the 12.38. In the next example, 12.125, here again, we've got 5. And so, we want to round toward the even neighbor and that turns out to be 2.

So in this case, we round down. Now in this next example, it looks like it didn't quite work according to that, our our rule here because the 5, should've rounded to the 4. But it actually rounded up to, 12.35. Well it turns out that many numbers can't be represented exactly in binary.

And 12.35 or 12.345 is one of those. And and what appears to be equidistant really isn't, fact the digit being rounded is not exactly 5. So this is what it, it actually is. Notice down below here that this. This 12.345 is really got some leftover stuff, again because it can't be represented exactly in in binary.

And so it's not 5 it's above 5, and so the fact that it's above 5 means it rounds up. So 345 the fraction there rounds to 0.35 rather than rounding down to 34 as as it should. Now let's look at this in an example. And we'll look at this in the viewer.

This figure down below here the actual calculation came from the viewer so let's take a look at that. So in this example we're gonna go through and and we're gonna let X= 12.374 12.376 these are from the table on the slide. And then here's 0.375 and 0.125. And we'll see that these get represented exactly and round like we would think.

And then here we've got this number 12.345 and we'll see that it's not exactly what it appears to be. And then we'll finish up with a one tenth, which looks like a pretty simple number to represent. But, turns out it's not so simple. So let's run this in the canvas.

There's a canvas built in for you. Now, if you want to build your own canvas, I'm gonna go ahead and step, and we'll get this on the canvas. So we're looking at 12, 374, 12.374, and you can see it's not exactly right. And we had to use some rounding there to get it.

Gonna move this up a little. Now do a couple things, one is if you wanted to build your own canvas. First time you open a program without a canvas, you would actually get a, a blank canvas like so. And you just need to drag X onto the canvas and, and you would have essentially the same canvas we have now, except that you're looking at this in the the default view.

You wanna change the viewer to numeric. We're looking at the numeric viewer, and then there you would have it. Then if you save this canvas, the next time you run the program if you run in canvas, the this will pop to the top. If you run in debug mode and click the little canvas button, you can get this up as well.

I'm gonna go ahead and close this, since we've already got one. Just wanted to show you how to get one. One other thing, we're about to print some output. And so we're printing that 374 out and to get a little more room down below we can change the pattern here of our desktop.

And I'm gonna switch this so that I see all the way across. You can see the pattern that this switches to. It just switches back and forth and gives us a little more room down here. We we're printing things out. And I'll just use the step arrow on the canvas.

And now there's a 376. And again, that's a, that, that was above, and so, that rounded up to, to 12.38, if you will. I'm gonna step again. And now, we've got 375, and you can see it on the canvas. That's actually represented exactly. And so this is the case where it's gonna round to the even neighbor.

So this should round up, in our case. And we see it printed out down below, it did. So that 5 digit they're rounded toward it's e nearest even neighbor which is an 8. So let's go down and look at 12.125 and here we see it's represented exactly and so in this case we got 5 at the end that we're gonna lose and it's gonna round towards the even neighbor which is 2, it's already even.

And so in this case, it rounds down. So if we print something out there, we'll see that, okay. And then here's 12.uh 345. This is the one that we were showing on the slide. And it looks like it ends in five, and the rounding, this is that, can't be represented exactly in batters.

So when we look at the rounding, there's the decimal version. As we see this 6 out here would round up to 1, and then the 1 rounds down. So we lose it all. We end up with 12.345 just using Java default rounding for floating point numbers. But the fact is, this, this 5 here, is really more than 5.

And so that's why this is gonna round up to, 12.35 instead of round toward the 4, 4, it's not exactly 5. So we'll step again and see that. And there it is. And again, it, it rounded to 5, because that wasn't exactly 5. It rounded up. So the last example is 0.1.

You'd think 0.1, which is, a, an easy number to represent in decimal. It turns out it's not so easy in binary. So here's 0.1, and notice here is the mantissa. The mantissa happens to be a repeating bicebel, it's called, and I'm going to just slide over here and we will see, it's, it's, 1001.

Ea each one of the, four binary digits that make up a hex digit you see. All the way out to the end and then it rounds up. So that 1, this last 1 actually rounded to a 2, and so we end up with 1010 in that last place.

And so again we've got a, a, a number here that doesn't quite, fit like you'd think, but the rounding turns it into, for our purposes, 0.1. You can think of an analogy like 2/3s in decimal. 2/3s is really .666 forever. But finally at the end, if you've got a finite number of space, as we do in this representation up here in in our computer, that 6, that last 6 rounds to a 7.

Well, that's essentially what happened here with these binary digits, it, it got rounded. And and we go with that. But, that's just a, just to show you that, there is a is some potential round off but a, it's not anything we need to worry about. But just wanted to demonstrate if you've got if you're round ending up rounding with in your decimal format.

If you goes up sometimes and down sometimes with with when it ends in 5, this this is why it happens.