Transcript: Part 2-Building Class

>> Now we need to complete the constructor and methods for our class, so let's take a look at the constructor first. Recall it's got the same, name as the class, so it's public Loan, and it takes in an accountType int. And, the purpose of it is to set the default interest, and then set the maximum loan amount, based on that parameter coming in.

So notice the first thing it does is set the interest rate to DEFAULT_INTEREST, that was our, constant for that. And then it checks to see if the parameter coming in, accountType, is equal to EMPLOYEE_ACCOUNT. If it is, it's sets the maximum loan amount to EMPLOYEE_MAX. Otherwise, it sets the max amount to the CUSTOMER_MAX.

So that's really all there is in the constructor. Now we're able to test the constructor in interactions. You can instatiate, a couple of objects, here's the lines of code for that. Loan customer = new Loan, and we're sending it Loan.CUSTOMER_ACCOUNT, and then we could create another one. Loan, empl for employee is a new Loan, and, and, its parameter is loan.EMPLOYEE_ACCOUNT.

And then, we could open viewers, or place these on the on the canvas, and we would see, these objects here for customer and employee. Now let's talk about the other, methods a bit, we've got a getBalance method and a setInterest method. In general, the get methods don't change the state, they only return a value of one of the fields.

And we've already completed, getBalance, we just added return balance. Set methods, and we'll look at the, code for several of these. They often have a boolean return that, indicates whether or not the input was valid, in other words, actually set the field. It comes back true if it sets the field, and false, for example, if it doesn't set the field.

And so once we get these written, you can test them, in interactions, as you did the other. In fact, here's an example here, for setInterestRate. Recall, the interest rate has to be, greater than 0, greater or equal to 0, for it to set it. And so here, we've, created a Loan object, and, this is a customer account.

And then we call setInterestRate with -1, and we expect it to come back false. And we could also actually check the interest rate in, in, our loanObj to see if it, changed, it shouldn't change. Here we call

setInterestRate with the value 2, and that should come back true because it's greater than or equal to 0, and there should be an updated interest rate there.

Then finally, here's one where we set the interest rate to 0.5, this is checking to make sure it takes a double. These two up here were ints, and, it, it takes a double. When you give an int, the int gets promoted to a double, during that transfer there.

So let's look at, let's look at the source code from the, for these, and this is in, Loan.java. Now we were looking at, the, just the, the skeleton code here, so let, let's, find the actual code. This is in the, folder method stubs, we need to go up a level.

And, here we'll see loan and, I'll just double click on that. And this opens the one with all the code in it. So here's our constructor which we looked at already. It just, sets the, loan to, the default interest rate, and which type of account, either employee or customer, it sets, the max for those two.

And then, here we've got the getBalance, it just returns balance. And then here's setInterestRate, this has a boolean return type. And notice the interestRateIn comes in, and we check to see if that's greater or equal to 0. If that's true, then we actually set the interest rate to the, parameter, interestRateIn, and we return true.

Otherwise, down here in the else block, we just return false, notice we did not set the interest rate. So it's up to the user to determine whether it came back true or false, and whether the interest rate was set or not. And then here's borrow, borrow also has a boolean return type.

Takes an amount of money, and then checks, in an if statement to see if the balance plus the amount is less than or equal to the max loan amount. If that's true, it updates the balance, adds the amount to it, that becomes the new balance, if you will, and returns true.

Otherwise, it just returns false and, and the balance did not get updated. And then here we've got a totalBalance, where we actually calculate the balance plus interest. And notice, it's just the expression balance times 1 plus interest rate. And, and that will give us a double that gets returned.

And finally here's the toString method, and, we've got a couple of, decimal formats, here. One for the, loan amount, and another one for the interest, and, and so on there. In fact, let's, let's go back to our slides and we'll talk about, one other thing there. Here, in the toString is an example of not repeating code.

It didn't repeat code, notice we're printing down here the total balance, and then we, well, first we, print out the loan amount, just balance. And then we wanna print out total balance which includes the interest rate. And we call, to get the total balance, we're calling the totalBalance method.

Now, instead of calling that method we could have just put the arithmetic expression in there, balance, times 1 plus interest rate. But you don't wanna do that, you want to have the calculation all in one place. And since the method is calculating it, you, you wanna make a call to the method here.

Two reasons for that, one is that if you need to change it, you change it in one place. And the other is, if it is in multiple places, it could be wrong in some place. And if it's only in one place, it's either gonna be always right or always wrong, and once it's right, it is right.

So again, if you've got, already got a method that calculates a, an item, you wanna call that. Now, we also need a driver program, the Loan class is not intended to be run as an application, it doesn't have a main method. So, it's, it's intended to be used by another program, and, and essentially we test it with a driver program.

And so, we'll need to create that, and, and this program will need to, set up a loan, as in create, a loan. And, and and then borrow some amount and, and display the information. That would be a sort of a simple case, driver, if you will. And this is, we've called this LoanCalculator, so let let's take a look at that.

So LoanCalculator is in the same class here, I double click on it, and notice I've got a project opened for it, so here's our LoanCalculator. We've got main, and main declares a few variables, and then we're gonna read in, the type of loan we wanna make, C for customer or E for employee.

And then we'll, actually, key in an amount after, well, after we key in the type, we're gonna create the loan. We'll see if loanType, charAt(0) was e, that's what we just read in from the, from the keyboard. If it was e,

we'll create a new loan, of, EMPLOYEE_ACCOUNT, otherwise, we'll create a loan for CUSTOMER_ACCOUNT.

So we assume it's gonna be, if it's not e, we assume it was c. And then down here, we're gonna enter an amount that we want to borrow. And we'll get that in and, parse it into a double, so we've got loanAmount that's a double. And then this is, interesting here, this may, you may not have seen this before.

But here we're going to call the method in the if expression. So we're saying if loan.borrow(loanAmount), and, and that's gonna call the method. And recall, the borrow method comes back true or false and, if it comes back true, then we actually, we're able to borrow something, and we're gonna print out the loan.

If it comes back false, that's because we were trying to borrow too much, we'll just print out the line, amount exceeds limit. So let's, let's, take a run at this, I think we have a little canvas for it. We'll open the canvas there, and we'll see what we've got, so I'm just gonna step, Let's step down here, and it says, enter, you've got to enter an amount for, I'll put it up here, for the loan.

For, E for employee or C for customer. Let's do a customer, and then we'll step some more, so there's the loan type. And then this is the if statement where we're gonna create an employee account or a customer account. Since this is a C, not an E, it's gonna go down here and create a, a loan that's for a customer account.

And there's our loan, I just dragged this out of the, debug tab onto here. So, so there's our loan, now we, in the main program, says here, enter an amount. Can't quite see all of this, but we're, entering an amount. Well, let me step, step, there we go, enter the amount that you'd like to borrow.

So this is a customer, let's go ahead and enter, we'll enter, $9000, OK, and hit Enter. And then we'll step, and here is where we check to see if, the, amount borrowed, when we go to loan.borrow, it's gonna check to see if, if we've, exceeded our amount here.

So I'm gonna step in here and, there we are, we're actually checking to see if, balance plus amount. And you can see amount here is $9,000 over in the, debug tab. And that doesn't exceed the loan amount, so we should be OK here. So I'm just gonna step, and so this is gonna increase the balance.

We can see that increase over here in our, on, on the canvas for balance. We can also open up this and see it increase there. So now we have a balance of $9000, and we're gonna return true. So we're back here, and since this is true, we're gonna go ahead and print out a loan.

And, when we, that calls the toString for loan, and we got $9000 and the total balance with 5% interest is, $9,450. Okay, so one more step and and we complete, let's run it again. We'll run it in the canvas again, and, this time let's run, let's go down here, and we'll key in, let's key in an employee loan of, And we'll key in the amount of $250,000, now, that should not let us borrow that amount so, let's hit Enter here.

And then, we'll step down here and, now here's where we calling the if statement. And, so I'm gonna step into this, the if statement, rather, is calling, loan.borrow, and it's coming back true or false. Let me just move that up a little bit, and so I'm gonna step in.

And here we are, wants to know balance plus amount is less than maxLoanAmount. Well, maxLoanAmount we see is 100,000, and this is, 250,000, so we know that's not gonna work. And so one step, and we're down to return false. And so notice we, the, the balance did not change, we didn't actually get to borrow anything.

And it, is gonna return, back to main, and so one step, this, this is now false. And so we're gonna end up in the else, and just print out, amount exceeds the limit. So you can play with this, program some more, test it out, and so on. It's a, you know, a nice little driver program to test the, the Loan class.

Now, just in summary, we've covered a lot here. And a lot of OOP topics, object oriented programming concepts, and so on. About how to write your own class, from which you can create objects, and, these are very important. In fact, they allow programs to consist of multiple classes now.

You'll have a driver class with the main method, and then one or more classes that define objects, that'll be used in the program. Now, large, complex programs may have hundreds of classes, maybe even thousands of classes. So, this is a very important concept for us to, sort of build up to.

And, you can also, in addition to your own classes, of course you have use of all the classes in the Java class library. You've been using some of those, scanner and so on. So, we have a lot, lot that we can do now in our programs. We can write our own classes and also use the class library.

So I wanna say that this particular, set of concepts in this module is, is critical for your success going forward. So you wanna be sure that you understand this.