Transcript: Building Class

>> Now let's look at a more complicated class, and in particular the process of building it. Suppose we want to create a class called loan, that represents a loan account. And it has a balance, interest, and maximum loan amount. And some more details, the balance starts at 0, interest rate is going to be 5%, unless otherwise set.

There's two types of loans, employees and customers. An employee can blow up to $100,000 and a customer, only up to $10,000. So to begin, we would declare the class in a new file, we'd open a new Java file here. And we enter a public class loan, open brace, skip a line, close brace and then we would save and compile this.

And now we have our class template so to speak or it's the stub for the class actually. Next we want to actually begin adding the class variables. We wanna add the instance variables to the class, so we take another look at the class description and see if we can figure out what those would be.

And we see that each loan has a balance, interest, and maximum loan amount. And the balance starts at 0, so these are gonna be our instance variables. So down below here we've got the code for that. And we've got a one line comment that says Fields- instance variables.

And so here they are coded up, private double balance = zero, interestRate and maxLoanAmount, all private doubles. And now let's take a look at what could be constants, we see that we've got a interest rate, we've got that default interest rate. And then we've got employees that can borrow up to 100,000 and customers that can borrow up to 10,000.

And so let's create a constants for those, so we've got private static final double DEFAULT_INTEREST. Notice in all caps, with the word separated with an underscore. So we'll set the default interest at .05%, and then CUSTOMER_MAX at 10,000 and EMPLOYEE_MAX at 100,000. And then a couple more constants, notice there's two loan types.

There's an employee loan type and a customer loan type. And we already know that customers can borrow 100,000, rather 10,000 and employees 100,000. And so we need a constant for the other two loan types there. And so here we've got down at the bottom EMPLOYEE_ACCOUNT = 0, and CUSTOMER_ACCOUNT = 1.

And these will be loan types, of course. We're gonna have a class loan, which we've declared here but a particular type of loan might be a customer account or employee account. So, now we need to build a constructor and that may look like as follows here. We've got an int parameter for account type and notice here, I've got a comment that says Fields go here.

So all of the fields that were declared on the previous slides there, would go here. We'll take a look at the actual file in just a moment but for the constructor, we need public and loan. It has to match the name of the class and then a single parameter, we'll call it a count type and it's an int.

And we'll just leave this constructor empty for the time being but eventually we'll need the fellow done with what we do with that account. So, now let's build some of the methods here. We need to create the skeleton cord or a stab if you will for each method or, and if it's simple we can just go ahead and complete it.

If it requires any code we'll just come back to that later. So the first one is a getter, get balance and notice it's very short, it's public double getBalance and this is just returning the value of the balance field so we just go ahead and complete that. And there was really not much in there except this one line.

And that regarding a setter here, here we're gonna set interest rate and it's a little more complicated. It says that we're gonna set the interest rate at double and it's gonna return true and set the interest only if the interest is a non negative. And so we do need to, we'll have a little code in there, so for now, let's just go ahead and return false here.

Next, let's look at the code for borrow, and total balance. Notice that borrow is fairly complicated, it says if an amount to be added to the loan will less than the max loan, then we add that to the loan amount and return true. Otherwise, we don't add anything to the loan amount and we just return false.

So for now, let's just return false and then for totalBalance, here we wanna return a double that represents the loan amount and with the interest added to it. And again, that'll be a calculation, so for now, let's just return 0. So at this point, we've got all of our stubs and let's take a look at them.

Transcript: Building Class

Here's the and I'm in the examples for this module and there's a folder called method stubs, and I've just double-clicked on loan and here it is. So there's not even any Java.comments here. So we've got public class loan and then here are our instance variables, balance, interest rate, and maximum amount.

And then the constants that we talked about default interest, customer max, employee max, employee accounting and customer account. And then here are the stubs for constructor, for the constructor, get balance we actually completed. And then we've got stubs for set interest rate, borrow, calculate total balance, and the two string method.

So at this point, we're actually we, we, this should compile, in fact we'll compile it here, everything compiles. So from from our end, the idea of constructing this, we've got it all sort of ready to go. We just haven't filled in the bodies, if you will. So if Web-CAT has a Skeleton Code assignment, we're ready to submit that.

And recall what's the Skeleton Code assignment does, it's going to really just check to see if you've got everything named correctly. The class has to be named correctly, all the methods have to be named correctly, have the correct return types, the correct parameters. And that's all that this is gonna check but it'll save you a submission if you say declare it a double parameter and it should have been an int.

It will check that or if it's returning a boolean type, and you're returning an int or a string or something, it will check that. So that's what you wanna do after you get the skeleton code built. So rather than building it all at once, it's much better looking at a project specification to go through and stub it all out.

And get the whole thing to compile and submit it to the Skeleton Code assignment if there is one. So next we'll actually fill in the bodies.