Transcript: Local Data

>> Now let's summarize some things about local data, constructors and constants. Let's begin with local data. As you know, the local variables are declared within a method. And formal parameters, they're also declared in the method header, these actually become automatic local variables. And, and jGRASP in the debug tab, you would see these as arguments, but they come into being when you enter the methods just like local variables do.

And then, when the method finishes, all the local variables are destroyed, in including the formal parameters. Now keep in mind, instance variables are different from local data. In that they exist, as long as an object of the class exist, that is there is a variable referencing the object.

You can think of the declaration of the string. If we said, string name is a new string and gave it, gave it a name. The name, as long as name exist and is pointing at the string, well, the instance variables in the string being the value, the character value of the string, it, it would exist.

Now let's talk about constructors a bit. Remember very important, a constructor has no return type, not even void. A common mistake is to actually give a constructor a return type, and if you do that, it's no longer a constructor, but it's just a method. That happens to have the same name as the class.

So you don't want to do that. If you, if you've written a constructor, and your constructor doesn't compile or it's not working properly, you're calling it and it's not working first thing to check, make sure you haven't get, you've you don't have a return type. Now, if the programmer does not define a constructor in a class, the class gets a default constructor, that has no parameters.

And we saw an example of that in our MessageCalc class. This is where we had the method calc and we and we passed in two ints and a string. Notice this is a class, it has a single method but there's no constructor. Yet in the method example we actually call a constructor right here on line 13.

We declared an object of type MessageCalc and then we said new MessageCalc. This is the parameterless constructor and then that allowed us to call the count method. So down here on line 16, we could say obj. That's what we, that was the object we created. Obj.calc, and and send it, the parameters there.

Now let's say a few words about Constant Fields. Recall a class constant can be declared using the static and final reserved words. And we usually put the name in all caps and if it's made up of individual words, we usually separate the words by an underscore. For example, we might have the constant MAX_AMOUNT.

So, max is in all caps and amount another word there, we, we put an underscore between those. Since we can't use the the camel case. So here we've got a a constant and we set it equal to 5. Now, in general, you don't want to use magic numbers in your code.

And so, by magic number, I mean something like this. Here we've got an if statement and it says number greater than 5. Well someone reading that won't know what 5 really is perhaps. So it would be much better to use a constant. 5 can't change in your program, it's a literal.

So it would be much better to actually say something like if number greater than max amount, where max amount is the constant 5. Now, unlike variables, constants can be public without violating encapsulation, so again these are fields in your class. If you want them used outside the class, then you just declare them public and that's not an issue.

In fact we've used some of those. For example, the constant PI in the Math class. Here we might say double circumference = Math.PI * diameter where Math.PI, PI, capital P-I is that constant and, it's declared in the math class and so that's how we get at it. So we we reference it using the name of the class and the dot operator and then the constant.

And so on. Now when you define a constant in your own class, and you actually use it inside the class, say MAX_AMOUNT, if this is used inside the class, we don't have to put the class name like we do up here. But if you're exporting the constant, if it's a client program using a public constant, you will need to use the class there.

So an example of that might be like this, here we've got a student class that has two type of students, undergrad and graduate students. And, here we've declared two constants, GRADUATE = 0 and UNDERGRAD = 1. And then in a, in a, in the client programs, say we declared a student object, an instance of the student class.

Instead of saying studentObj.setStudentType (0), instead of that we, it'd be much better to say studentObj.setStudentType (Student.GRADUATE). That would make a lot more sense for someone reading the code, or if it was an undergraduate, you'd say Student.UNDERGRAD.