Transcript: Object Oriented Programming

>> Let's finish up our introductory set of notes here with a few more words about object-oriented programming. OOP is a programming world-view, in which things in the real world are modeled as software objects. And objects is really just an abstraction for a real world item. And it's implemented as an encapsulation of private data and methods, or operations that work on the data.

We might look at that graphically as a cup of circles here. Inside you see the data represented by squares and it's encapsulated, surrounded by methods or operations. We would think of a, b, and c as separate operations or methods. That work on this data or, or, call other objects that, that do things, and so on.

For instance, we may have several objects that communicate with each other by calling methods. This is an example here of c calling j. And the y method of this third object is calling the h method there, or vice versa, h is calling y, and so on. So they're really communicating by sending methods, or rather messages, or calling each other's methods.

Now a class is how we actually define objects. We just saw objects communicating with each other, but the way we define them is with a class. And it's a description of an entire category, if you will, or group of objects. And, again, the idea is to model real world items by describing their data and their operations.

You might think of a class, as an example of this. A class named GamePlayer. And it might have data for Level, Speed, Health points. Things about the GamePlayer attributes, if you will, its state. And then operations or methods for Run, Jump, Pick up item, and so on. Then we would model the class graphically, it might look like this.

Inside we've got the data items Level, Speed, Health points, and so on. And then they're surrounded by the methods and the only way to get at the data, or state, is, is via the methods. So, the -- this data again is encapsulated. Once we have the class we can create individual objects.

So, here we might create Player1, this is an instance of the class, if you will, or an object of the class. We might also have Player2. And notice that they each have the data items Level, Speed, Health points, and so on, and, and the methods. But the, the, data here is different for each one.

So, that these are unIque data items for each one. So, each player's state can be different depending on, of course, how they're playing the game, the methods it get called, and so on. And, we'd probably have another object here called environment where the players are interacting with the environment, and so on.

One other major feature, if you will, of object-oriented programming is inheritance. And it allows us to derive classes from other classes. Suppose we had an Employee class, and we, we wanted to derive a Staff class, a Faculty class, maybe a StudentWorker class. The idea is Employee has data and it has methods.

And when we derive the Staff class it inherits, thus the term inheritance the methods and data from the parent class. We think of Staff as a child class, or a subclass, of the Employee class. And we might extend that or derive another class called Advisor, and it inherits from Staff and Employee.

So, the idea is this is reused, we're reusing the stuff there. And when we extend or derive a class we just have to add the things that are unique to, to that class to make it a separate class there. So, that, that talks about a couple of major items, this idea of a class, and objects that come from it, and, and then inheritance.

All this sorta leads to the idea of class libraries that provide lots of reuse. And so, in Java we have the API and, again, class libraries. These are sets of classes designed to be reusable components by other programs. And, in Java we got the API, the Application Programming Interface.

And there are literally thousands of classes in there ready for us to use. And it's organized into packages such as the awt, io, that would be java.io. This other is a windowing toolkit. java.lang is a general purpose package. java.net for doing network type stuff. For example, the system class which we've already been using, to print to the screen that's in java.lang.

And as we go on here, we'll use lots of other classes from the API. Again, there's many of them, and, and they've already been tested and they're ready to go.