CS 240: Classes and Objects – The "Object" Super Class Transcript

[00:00:00] **PROFESSOR RODHAM:** Now, the next thing we're going to do is we're going to learn how to write the "toString", the "equals," and the "hashCode" methods on a class in Java.

[00:00:10] Now, one thing to know about Java is that every class in Java inherits from a class named "object." This is how you say that in Java: the "Person" class extends the "Object" class.

Start visual description. Professor Rodham types "Person extends Object {" on the third line of the coding page. End visual description.

[00:00:27] Every class in Java either directly or indirectly inherits from the "Object" class. So, the "Object" class is the root of the inheritance hierarchy. And even if I don't say that in my code, that is still true. So, Java will make the "Person" class inherit from "Object" even if I don't say that explicitly.

[00:00:47] Now, if we go to the Java documentation, we can see, if we go to the documentation for the java.lang, L-A-N-G, package.

Start visual description. Professor Rodham navigates a webpage that lists Java packages and their descriptions. End visual description.

[00:01:01] If we scroll down through the list of classes in that package, we will find a class named "Object." And so, this is the root of the inheritance hierarchy. As we said, the advantage of having every class in Java inherit from the same superclass is that you can have a set of methods that every class has.

[00:01:29] So, you can depend on these methods always being there and available, and that's the real advantage of having a common parent class. So, whatever methods we have on the object class, they are available on every class in Java.

[00:01:44] In this case, let's just look at some of the methods that are on the object class. A simple one that we'll be looking at today is called "toString." The "toString" method is called if you want to convert an object to a string.

[00:01:59] Just imagine taking the variable values that are inside an object and just encoding those in a string and then returning the string. That's what the "toString" method does. If you ever need to convert an object to a string for debugging purposes, so you can print it out or so you can write it to a file or whatever your purpose may be, just call the "toString" method on the object, and that'll give you back a string representation of it.

[00:02:22] Now, by default, the "toString" method doesn't really do very much that's useful. By default, it just returns, I think, a string containing the name of the object class and also the address of the object in memory, and that typically is not very useful.

[00:02:38] If you want to make the "toString" method useful, you need to override it, and we'll learn how to do that in a couple of minutes here. Another method that's on the "Object" base class is called "equals." The "equals" method is used to compare objects for equality, and we'll talk more about that very soon as well. In Java, there's two ways to compare objects for equality.

Start visual description. Professor Rodham navigates a webpage that lists various modifiers and their types, the methods to be adopted in specific instances, and the descriptions of the methods. End visual description.

[00:03:05] You can use the double-equal operator, which compares the objects by address or you can call the "equals" method, which will typically compare the objects by value or by the data that's inside the objects. We'll talk about that more. Another method we have here is called "getClass." For any object in the Java program, you can walk up to it and you can call the "getClass" method.

[00:03:30] And that method returns an object of type “Class”. Java actually has a class named “Class”, which is really confusing. You can see “getClass” returns a class “Object.” And a class “Object” has a lot of information about the object’s class inside of it. For example, the class “Object” would tell you the name of the class.

[00:03:50] It would tell you the names and types of all the fields inside the class. It would tell you what methods are on the class and what their parameters are, and return values and so forth.

Start visual description. Professor Rodham places cursor on “Class” while navigating the “ALL CLASSES” webpage. End visual description.

[00:04:01] It would tell you what classes does this class inherit from? All kinds of information, most of which we don’t really need to know about in this class. But in general, you can learn a lot about an object simply by calling “getClass.” We’ll use that in a few minutes. Another method that every object has on it is called “hashCode.”

[00:04:24] If you want to take your objects in your program and put them in a hash table, you need to override the “hashCode” method, perhaps, and we’ll talk about what that means and how to do that shortly.

Start visual description. Professor Rodham places cursor on “hashCode” while navigating the “ALL CLASSES” webpage. End visual description.

[00:04:36] Just remember, every class in Java inherits from a common ancestor named “Object.” And all the methods that are on this class are universally available. There’s lots of other ones that we’re not going to talk about now.