Transcript: Wrapper Classes

>> The java.lang package contains wrapper classes that correspond to each primitive type plus the reserved word void. As you can see from the list the wrapper class is essentially the same as the primitive, except its got a capital letter in most cases. For instance, byte, the primitive type, has the wrapper class that just begins with a capital B, two exceptions, and that's int.

Int has a wrapper class called Integer with a capital I, and the other exception is char its Wrapper Class is the entire word Character with a capital C. Notice Void down here also has a Wrapper Class with a capital V. The following declaration creates an Integer object which represents the integer 40 as an object rather than a primitive type.

Notice we declare age to be of a type Integer, this is the wrapper class, and we use the the new operator with the constructor Integer and we give it a 40. And that creates that object. Now, a primitive variable of type Int would not have methods. We know that, however type Integer, the class, does have methods.

For example, byteValue returns the corresponding byte value of an int. The doubleValue returns a corresponding double value of the int, the int is what's wrapped up, keep in mind. And of course, it has a toString method that would get you the string version of, of the, of the underlying int that's inside there.

Now Java does provide autoboxing and, and, and unboxing, they're both automatic and this is very convenient. So autoboxing is the automatic conversion of a primitive value to the corresponding wrapper object. For example, if we declared Integer obj;, this obj now can reference an integer object. And then we declared int num = 42, we could make the assignment obj = num.

Now num is an int, but during the assignment, the appropriate integer object is created and assigned to obj. The reverse conversion is called unboxing, and this is automatic as well. So if we made the assignment, num = obj, num of course is an, is a primitive, a primitive event and obj is the object integer.

Well, it's actually unboxed and, and the underlying value inside obj is assigned to num. So it's actually pretty convenient to go back and forth between these. Now, wrapper classes have useful constants and static methods, as well. For example, the integer class contains a MIN_VALUE and a MAX_VALUE. And that is the smallest and largest int values that can be represented in Java.

Integer.MAX_VALUE, you see there is, a long number, that's actually 2 to the 32, and the min value, is minus 2 to the 32. The and notice we get at these using the, the class names and since these are constants, so it's integer.MAX_VALUE, and integer.MIN_VALUE. The Integer and double classes contain methods that convert a number stored as a string in to a numeric value for example int i = Integer.parseInt, and we send that a string.

Here I have sent it the string literal ("1234"), and Integer.parseInt, would convert that ("1234") string, into the integer value, 1,234. Here's an example with a double. If we were to do Double.parseDouble on the string 12.34. Again, that's a string, we can't do arithmetic on that. But if we needed to do arithmetic, we could convert it to a double with this Double.parseDouble.

And and then we would actually have the value double. These two methods are used frequently, especially with reading input as you know, all input comes in from the keyboard as a string. And if you're reading a, a file in, it typically comes in as text and so if there's numbers there and you need to convert them to a corresponding int or double so that you can do arithmetic.

These are the mess, methods that you would use. All right, let's look at some examples.