Transcript: Reading Data from a File

>> Now let's take a look at reading data from a file. As you know, we've been using, using the Scanner to read text from System.in. Well, it turns out it's pretty easy to use the Scanner for, for reading a file. We do have to add a couple of import statements we need to import java.io.File, and also java.io.FileNotFoundException, and then of course Scanner as well.

In main, we're gonna need to add a throws clause and you see main here we've got public static void main String array args. And usually after that comes the body, but now we need to add in this throws clause, throws FileNotFoundException. Now, what this does, it indicates that that we're aware that this exception may be thrown when the Scanner is created on a file.

In other words we're gonna give it a file name and if we don't find that file name this exception gets thrown. But furthermore we're just planning to ignore it, just like we've ignored no pointer exception and so on. And if this exception is thrown just like these other exceptions we've had in the past the program is gonna end immediately.

Now, the reason that we have to actually throw this particular one is because it's a checked exception, all io exceptions fall in that class. And that means you either have to handle it, as in do something about it which we don't know how to do yet or you need to say in the method header that your method may throw that exception.

So that's what we're gonna do, nothing to worry about, just sorta proforma here, that you add a throws clause to main there. Now after we've got main and the throws clause, we're ready to create a Scanner object on a, on a new File object. And let's suppose fileName is a String, and you could actually put a String literal in there if you know the name of the file data.text or whatever you want to call it.

But here's the statement, it's scanner scanFile and this can be any name it's just the name of your Scanner, is a new Scanner. And instead of System.in, we've got new file in the file name or that the file name is, is text, either a String or its the actual name of the, literal name of the file there.

And this file will need to be in the same directory as your program or you'll need to actually put a full path in there. Easy just to put it in the same directory, all our examples will be that way. So, after we've got this Scanner set up on the file, we're ready to read in data.

And normally since we may not know how much data is in the file, we're gonna have a loop. And in the loop condition, we'll say, say something like scanFile.hasNext, and this returns true if there's any, anything else to read false, if we've, if, if we're at the end of the file.

So, use a file use a while loop here scanFilehasNext. And then inside the loop, we're gonna read the data and we can do that using our traditional Scanner methods just like we'd use with System.in. For instance we've named the Scanner scanFile, it would be scanFile.nextLine or scanFile.next, if we just wanted to read the next token.

ScanFile.nextInt, if we want to read the next token as an integer, scanFile.nextDouble and so on. Now, also in the body of the while loop, as, as we read the data in we usually store it some place, and/or we process it. So during the loop, we're reading in data and doing something with that data, and then of course the loop will eventually end when, when scanFile.hasNext is false.

And then it's a good idea, after the loop, the first statement after the loop to actually close that Scanner object on the file, and we do that the statement scanFile.close. Now, we're only reading so this particular statement isn't so important but if we were writing files which we will learn to do eventually in the study of Java, closing the file would actually be much more important.

Okay, next we're gonna look at some example programs that read in data from a file.