CS 240: The Scanner Class Tokenizing Input Transcript

[00:00:00] The scanner class is another way to read files.

[00:00:03] And this is useful when you need to tokenize the input.

[00:00:06] Um So the scanner can read one token of characters at a time.

[00:00:10] And it’s probably easiest to explain what a token is by showing you an example.

[00:00:18] So here we have a file of words, and you can see that it is separated by white space, various kinds of white space.

Start visual description. The professor demonstrates how the scanner class can read a file of words separated by various kinds of white space, such as spaces and new lines. The professor shows an example file and explains that tokens are the characters we care about, separated by characters we don't care about. End visual description.

[00:00:25] It has spaces and new lines.

[00:00:27] And so tokens are the characters that we care about uh separated or delimited by the characters that we don’t care about.

[00:00:35] So the scanner class can read in just the words and skip over the white space.

[00:00:40] So these words are the tokens.

[00:00:43] So here’s an example of that and I’ll, I think I’ll run the example first and then I’ll show it, show it to you.

[00:00:49] So here you can see that this printed out just the words.

[00:00:53] So the way that works in this example, we create an instance of our scanner example class, and we check to make sure we have a parameter.

[00:01:02] So um the parameter is just the file name that we want to read and it’s probably OK, probably a good idea to just review that to show you how to, how to specify parameters.

Start visual description. The professor demonstrates creating an instance of the scanner example class and checking for a parameter, which is the file name to read. The professor shows how to specify parameters in a run configuration and explains the process of calling the process file method with the file path. End visual description.

[00:01:12] So I have a run configuration here.

[00:01:14] The way to get a run configuration.

[00:01:16] The easiest way is to just run a class that has a main method and then it’ll create a run configuration for you.

[00:01:22] Then once you have that you can edit it. So, you go into edit and here um I’ve just specified as a command line parameter, the text file that we want to read.

[00:01:33] OK. So, we expect to find that as a command line parameter, then we call process file and we pass the, the path, the parameter that we received.

[00:01:43] We create a file object with that, and we use that as a parameter to the scanner constructor.

[00:01:48] So we just create an instance of scanner.

[00:01:50] Once we have a scanner, we can create a Y loop. And in the while loop, we just say we has next. So as long as there's a next token, we will read the token by calling next, that gives us the next string or the next um the next word in that file and then we print it out.

[00:02:10] So the scanner is a lot like an input stream or a reader, but it will tokenize the data as you read it, which you don't always need.

[00:02:17] But when, when you need that, it's really useful, we can also control what the delimiter are.

[00:02:25] So by default, notice, I haven't specified anything about what the delimiter should be.

Start visual description. The professor demonstrates how to control delimiters in the scanner class using regular expressions. The professor explains that by default, the delimiter is any form of white space, but it can be changed to include comments or other characters. The professor shows an example of specifying a custom delimiter with a regular expression. End visual description.

[00:02:29] So by default, the delimiter is any form of white space, but we can change that. So here we have a similar file except it also has a comment that we also want to skip over.

[00:02:42] So we want to include that in our, in our delimiter, so we can do that with a regular expression.

[00:02:48] So in this example, the main method is the same.

[00:02:53] Yeah, and we're passing in the file path, creating a file object.

[00:02:57] But notice once I create the scanner before calling has next on it, I specify the delimiter.

[00:03:03] So I'm saying that I don't want to use the default delimiter and you can specify one by regular expression.

[00:03:09] This isn't really a regular expression class, but I'll go ahead and interpret this regular expression for you.

[00:03:14] And I think 235 would be the class where you would learn that in more detail.

[00:03:19] Um But what this regular expression says is it, first of all, it has two sets of parentheses with an or between it. So, it means that either this or this is the delimiter and then that those two things are combined and another set of parentheses with a plus sign.

[00:03:39] So what that means is anything that matches this or this, any number of times.

[00:03:44] So that would mean any number of spaces or any number of comments or any combination of comments and spaces would all be considered delimiters.

[00:03:51] So the way this regular expression works is first of all, the pound sign says, um something that starts with a pound. So, a line that starts with a pound sign and then has any number of characters. So this is the specifies characters that are not new lines.

[00:04:09] So that says anything that's not a new line, any number of times followed by a new line.

[00:04:14] So that's saying that a comment starts with a pound has some number of characters in it that are not new lines and ends with a new line.

[00:04:20] And that's what we can see right here.

[00:04:22] Pound some number of characters and there's a new line there.

[00:04:25] So that's one thing that would be considered a delimiter.

[00:04:28] And this is just a shorthand for saying any white space is also a delimiter.

[00:04:33] So now that we have that we can, first of all, I'll go to the run configuration for this file and show you that. Now we're reading text file two, which is this one that has the comment.

[00:04:49] And when I print it out, it's still just going to print the words and it will skip over the comment and the whitespace.

[00:04:57] Another thing that you can do with scanners is in these two examples, I read from a file, but you can also specify that a scanner should read from either a reader or an input stream.

[00:05:08] Usually when you're using a scanner you're reading text data, so you would normally want to use a reader.

Start visual description. The professor demonstrates how to use the scanner class to read from a reader or an input stream, instead of a file. The professor sets up a file reader, attaches a buffered reader to it, and then attaches the scanner to the buffered reader. The professor explains that this allows efficient reading and tokenizing of data from any source. End visual description.

[00:05:12] So here notice that we've set up a file reader, attach the buffered reader to that and then attach the scanner to the buffered reader.

[00:05:19] So that allows us to read tokens efficiently.

[00:05:21] Um The buffered reader, as I mentioned in a previous video will allow you to read a file efficiently.

[00:05:28] And then the scanner will tokenize whatever data it’s reading from the buffered reader.

[00:05:32] And we’re still using that delimiter that I showed you in the previous example.

[00:05:36] So now if we run this, we get the same output, it’s tokenizing the data and doing it from a reader, which really would allow you to tokenize data from any source because you can attach a reader or an input stream to basically any source of data.