CS 240: Java Logging Transcript

This video shows a split screen of Professor Wilkerson on the right and a PowerPoint screen on the left. Any text displayed or action performed that is not verbalized will be included in italics as visual descriptions.

[00:00:00] **JEROD WILKERSON:** Logging is a really useful technique in software development, and I want to introduce this with an example that you've probably experienced.

[00:00:08] If you think about when you've written a fairly large program before and you've had a bug in it and you've been looking for the bug or something's not working quite right.

[00:00:17] I want you to just ask yourself, have you ever done this? Have you ever gone into your code and put a bunch of print statements in there? In Java, it would be system.out.println statements that you would put.

[00:00:27] In C#, it was probably cout statements.

[00:00:30] Have you ever done that, have you ever gone into a method or a set of methods and just scattered these print statements all over so you can see what's going on in your code and then ran your program to see what the output is so you can tell what's going on with it? I'm thinking that probably most of you have done that.

[00:00:47] Then once you find the defect or the bug, the problem, what do you do? You delete all those statements, and then you work on your code again and you find another problem.

[00:00:58] Now, what do you do? You put the statements back.

[00:01:00] Maybe some of the same ones or maybe different ones, maybe in different methods, but you end up in this cycle where you're going back and forth between putting print statements in and taking them out so you can find problems.

[00:01:11] Well, there’s a much better way, of course. There’s debugging.

[00:01:13] You can use debuggers, which can eliminate a lot of the need for this, but another technique that you can use is logging.

[00:01:21] Logging is a way to put the statements in and not have to take them out, not need to worry about taking them out.

[00:01:28] Java has built-in support for this and I need to explain some details for you to understand why we wouldn’t need to take them out when we’re through with them.

[00:01:38] I’ll get to that. These statements end up being really useful.

[00:01:43] They’re useful for developers who are debugging, as I just explained, but they’re useful in other cases too.

[00:01:49] They can be useful for system administrators to be able to look in a log and see what’s going on with the program and how it’s running.

[00:01:55] They can be really useful for customer support agents.

[00:01:59] If you have some software that a lot of people are using, you might have people that would have some problem and they would call in and say, “Hey, the software isn’t working.

[00:02:09] I don’t know what’s going on with it,” and the customer support agent can say, “Okay, well, what’s your name? When were you using it? When did you find the problem?” Then they might be able to find a file, or a log file, and find out what the program was logging, what messages it was sending at the point when the user had the problem.

[00:02:28] That can be really useful.

[00:02:30] The way this works is your program will send messages to loggers.

[00:02:36] We have these objects that are called loggers, and you create messages and send them through the loggers.

[00:02:44] Messages can have a level, and so in Java, they can have these levels: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST.

[00:02:52] The way this works is you create your log message at the level that indicates its severity.

[00:02:59] If it's a defect, a bug, and your program's going to crash, then you'd probably say that's SEVERE.

[00:03:05] If it's something where it's just debug information that normally we don't need to see, but once in a while we need to see it, it's probably one of these: FINE, FINER, or FINEST.

[00:03:13] You log the message at that level and then you never need to remove the statements; you just keep them in.

[00:03:20] You can turn loggers up and down to get them to ignore some messages and not others.

[00:03:29] For example, you might want to set your logger at WARNING or INFO so anything that's lower than that would not be logged, would be ignored.

[00:03:37] If you're having a problem, you could change it to be lower and then you'll see more messages, so that's the way it works.

[00:03:45] We have these loggers and loggers have a method for each level of message.

[00:03:52] A logger will have a method for each of the levels that I showed you, and loggers can be configured to omit or include messages at different levels based on the log level.

[00:04:05] You can set the log level to something.

[00:04:09] If we set it to ALL, then it’s going to log all messages.

[00:04:12] If we set it to OFF, it’s not going to log anything.

[00:04:15] If we set it to one of these, so for example, if we set it to CONFIG, then it will log anything that’s at that level and anything to the left.

Begin visual description. The list of levels of listed as referred to earlier, thus the levels to the left of CONFIG (in order of most left to closer to CONFIG) are SEVERE WARNING, and INFO. End visual description.

[00:04:27] Anything to the right it would ignore. *(In order of closer to CONFIG and most right, it would ignore FINE, FINER, and FINEST.)*

[00:04:29] That allows us to turn our loggers up or down and get more or fewer messages, more detailed messages, or fewer messages in our logs.

[00:04:39] We also have something called handlers.

[00:04:43] You have a logger, and it can have different handlers associated with it.

[00:04:49] Handlers determine where the messages actually go, what’s done with them.

[00:04:54] Each logger will have one or more handlers—you can have multiple—and here’s an example of some of the handlers.

[00:05:00] You can have a ConsoleHandler, which will send messages to the console.

[00:05:04] You can have a FileHandler that will send messages to some file, and you can even have a SocketHandler that would send messages over a network on some network socket. We might use that.

[00:05:14] There actually are programs that are made for catching and tracking and doing things with logs.

[00:05:21] There's a program called Logstash, for example, that you might access using a SocketHandler.

[00:05:26] Like loggers, handlers can have different levels that can be turned up or down.

[00:05:31] That would mean that we could have a logger that is set at some level and if it receives a message that is set low enough that it would not ignore that message, then it will pass that message on to any of its handlers.

[00:05:45] Then the handlers may or may not handle the message depending on what their log level is.

[00:05:50] That allows you to do things like, for example, you could have a console logger set at a pretty high level so you're not getting a lot of stuff in your console and have a file logger set at a lower level, so a lot of information is going into the file.

[00:06:04] That's pretty useful. Then finally, we have formatters.

[00:06:09] Each handler can have a formatter associated with it that determines what the messages look like when they are printed out.

[00:06:17] Normally, we want to use simple formatter—that's the default—and I'll show you how you still have a lot of control of what messages look like using a simple formatter.

[00:06:25] You might have some reason why you want to create XML logs, for example, and so there's an XMLFormatter.

[00:06:31] You can write your own, you could write a JSON formatter if you wanted to.

[00:06:36] That’s the basics, the high level overview of logging, and in the next videos, I’ll show you some of the specifics about how to make it work.