CS 240: Code Style Checking Transcript

[00:00:00] In some previous videos, I, we talked about writing quality code and how you can format your code.

[00:00:06] And there are a lot of things you can do to make your code more readable and, and easy to understand.

[00:00:11] Uh There's also tool support that can help you with that.

[00:00:15] So there are tools that will format your code for you.

[00:00:19] So um what they do is compare your code to a coding standard and then they can reformat it according to that standard.

Start visual description. The professor demonstrates how tools can compare your code to a coding standard and reformat it accordingly. The screen shows a code editor with a piece of code that is being automatically reformatted to match the coding standards. End visual description.

[00:00:27] So some of these tools are built into editors like Intel and I'll show you that in a minute.

[00:00:34] Um some of these tools are independent and are intended to be part of a build process.

[00:00:38] So sometimes we will generate code or, or we'll, we'll have an automated process where for example, if you push a get commit, you could have a process that will take the code that you just changed and run it through a set of steps to see if it's OK.

[00:00:55] So it could compile it, it could run it through an automated style checker, for example, to determine if your, if your code um conforms to the team's coding standards and then it could run the test and if any of those things fail, then your code doesn't get um pushed to production or, or it doesn't move on to whatever the next step is.

[00:01:15] So for that reason, because we have these automated build tools that can be useful to have style checkers that can run from the command line.

[00:01:22] So we have, we have not only tools that can run from ID ES like Intelli J, but we have uh several tools check style is a popular one that can be integrated into some kind of an automated build process.

[00:01:35] Um That's one that's used a lot with java.

[00:01:37] We have prettier which is used in JavaScript and typescript code.

[00:01:43] Um This can be integrated into um IDEs and then as I mentioned, we have the built in code formatter.

[00:01:52] So I will show you now um some things about the IntelliJ code format.

Start visual description. The professor shows an example of bad code with improper spacing and formatting. The screen displays the code editor with highlighted sections of the code that need reformatting. End visual description.

[00:01:58] OK? So here is an example of some bad code.

[00:02:01] Um There are quite a few things wrong with it, but we'll mostly focus on the spacing.

[00:02:06] And I want to show you how you can use Intelli J to fix this and the built in Intel code editor.

[00:02:12] So there are a few different ways you can do this.

[00:02:15] There's a menu option that we can select that will reformat the code and depending on what I have selected, that will determine what code gets reformatted.

[00:02:23] So if I have a file open in the editor and I select some code in in that file, then if I go to code reformat code, it will just reformat the code that's highlighted.

Start visual description. The professor demonstrates how to use the 'Reformat Code' option in IntelliJ. The screen shows the code editor with a file open, and the professor selects a portion of the code and uses the menu option to reformat it. The code is then shown in its reformatted state. End visual description.

[00:02:36] So there we go. It reformed that code according to a set of rules.

[00:02:40] If I have an editor file open and I don't have text selected, it will reformat the whole file.

[00:02:50] So there we just reformatted the whole file.

[00:02:52] But notice I have another file here that also needs to be reformatted.

[00:02:56] So I'm going to undo this change and show you another thing.

[00:03:04] So now we're back to having two files of bad code in this project.

[00:03:08] So if I select the project or the source directory, let's just select the project and then I do code reformat code, it will reformat all of the code.

[00:03:22] So now when I run that, it not only reformatted this file, but it reformatted this file as well if you remember from before this code was all on one line.

[00:03:30] So that's a really nice way to be able to make sure your code conforms to a standard.

[00:03:36] And there is a, a built-in standard in IntelliJ that it's using to format this code, and you can change that if you want. So, I'm using a Mac. So, I go to um Intel J settings, I think on Windows, it's file properties or something like that.

[00:03:52] Um But if I go into settings, I can let's close that. So you can tell how to get there. I can open up editor and then let's see, code style and then specify the language.

Start visual description. The professor navigates through the IntelliJ settings to customize the code formatting options. The screen shows the settings menu where the professor adjusts various formatting options such as tab size, spaces, and blank lines. End visual description.

[00:04:06] And now I have a lot of things I can do to specify what the format should be.

[00:04:11] So I can determine what the tab size should be.

[00:04:15] Whether we use the tab character or not.

[00:04:17] If this is not selected, then it will format with spaces.

[00:04:20] Um And you can see that there are a lot of different things I can specify here about spaces wrapping, how to use blank lines, what to do with imports, whether we want to have wildcard imports or not.

[00:04:32] So I won't go through all the details, but you can see from this, there's a lot you can do to affect the formatting so you can just make your changes here, hit, um apply or just hit. OK? And then the formatter will format code according to what you just specified in here.

[00:04:50] Um There’s also a way to uh integrate check style with this and I’m not going to go into details of how to do that.

[00:04:58] But if you have a automated build process that’s using check style, um actually, it probably should be a better way to do this.

[00:05:05] Um At least with this version of Intel J, the current version of IntelliJ.

[00:05:09] Um I think what you have to do is just go in here and change all the settings to match your check style format. So that’s what you would normally want to do.

[00:05:17] You, you would have a maybe a build process that is going to automatically check the formatting.

[00:05:22] But as an individual developer, you want to be able to make sure your code is going to pass those checks.

[00:05:28] So you want to make your settings here in your ID E match whatever your automated build process is going to expect, you can do that just by coming in here and changing the settings to whatever you want them to be.