CS 240: Advanced Breakpoint Settings Transcript

[00:00:00] Now that you know how to use the, the basic debugging features in IntelliJ , I want to show you some more advanced breakpoint settings that you can use.

Start visual description. The professor demonstrates how to use advanced breakpoint settings in IntelliJ. He starts by showing two breakpoints set in the code. End visual description.

[00:00:08] So I have two breakpoints set here.

[00:00:13] And the first thing, the first thing I'll show you is that you can get a list of all of your break points.

[00:00:18] So if you go to run view break points, you can see all of the break points that have been set from here. You can turn on exception, break points.

Start visual description. The professor shows the "Run" menu and selects "View Breakpoints" to display all the breakpoints set in the project. He explains how to turn on exception breakpoints. End visual description.

[00:00:28] So if you want to have um if you want to have your program pause, when a when an exception is thrown, you can just click that that's a breakpoint exception.

[00:00:38] Um Now it will pause for any breakpoint or I'm sorry, it will pause for any exception that occurs and you can even create specific exceptions. So, if you click here and select java exception breakpoints, I can enter any exception and then it'll break on, on specific exceptions.

[00:00:55] So that can be pretty useful.

[00:00:58] Um Some other things that you can do, there actually are several features on this, on this dialogue box here.

[00:01:03] And I want to show you two ways to get to it. So, I showed you first that you can come to uh run view breakpoints and get there.

[00:01:11] You can also get there from specific breakpoints. So, if I right click, I already showed you that you can specify conditions for breakpoints right here.

[00:01:20] Um Another thing I think that I didn't show you in the previous video was that you can enable or disable breakpoints from here.

[00:01:27] That can be useful because sometimes you don't want to delete a breakpoint, but you don't need it right now.

[00:01:32] You might need it later.

[00:01:33] So it can be useful to just disable it.

[00:01:36] And then you don't have to remember where you put it before.

[00:01:39] So that can be pretty useful.

[00:01:41] But from right here, you can see this more button and that will take you to this dialogue again.

[00:01:47] So here I can do some additional things with breakpoints that can be really useful.

Start visual description. The professor demonstrates how to specify conditions for a conditional breakpoint and how to log information when a breakpoint is hit. He shows the "Evaluate and Log" feature where custom code can be logged. End visual description.

[00:01:51] So one this is another place where I can specify a condition for a conditional breakpoint.

[00:01:57] I already talked about those. So, I won't go into detail on that.

[00:02:00] Um You can specify uh that different information gets logged.

[00:02:04] Usually when we're debugging, we're just stepping through code and, and looking at it and um and that's really all we need, but sometimes it can be useful to log some information about your breakpoints and then you can kind of dig through your logs later and see and see what's in there.

[00:02:20] So you can click that and then that will just give a message that indicates that a breakpoint was hit.

[00:02:26] You can print out a stack trace at each point where the or each time that the breakpoint is hit.

[00:02:33] And that can be pretty useful to see where you came from, and your code and you can even kind of have an arbitrary message. So, if I specify, evaluate and log I can type whatever code I want in here and whatever this code ends up returning is what will be logged.

[00:02:50] So you can take control of what your, what your log messages look like.

[00:02:54] So I don't find that I use that very often.

[00:02:56] I don't usually need to log information that breakpoints, but that can be useful.

[00:03:00] Um Another thing you can do, you can make it sometimes you only care about the first time you hit a break point.

Start visual description. The professor explains the "Remove Once Hit" feature, which allows a breakpoint to be removed after it is hit for the first time. He provides a scenario where this feature is useful. End visual description.

[00:03:07] So you can click this, remove once hit, then it will break the first time that it hits it.

[00:03:12] And then if, if you're in a loop or something, it won't keep breaking when it, when it hits there.

[00:03:16] So that can be useful.

[00:03:18] One of the very most useful advanced features is this disable until hitting the following break point.

[00:03:23] So I want to give you a kind of a scenario for that, that you've probably experienced before.

[00:03:27] So you can make it. So, a breakpoint will only pause your program if some previous breakpoint has been hit.

[00:03:35] That's useful for cases like if you have some code where you have, you're going to put a breakpoint in some code.

[00:03:42] And this often occurs with testing where you have some setup code that might actually call some function or method that also gets called from your test and you only care about it from the test.

[00:03:53] So an example of that could be if you're testing some database logic in your DAOs, you might have some code that does some inserts in a setup method to get your test to create some test data.

[00:04:07] And then in your test method, you might do some other things like maybe some updates or deletes or some additional inserts.

[00:04:14] Um In that scenario, you likely don't care about your breakpoint being hit in the setup method.

[00:04:21] Um But if you're testing your, your base database logic, then that logic is probably going to be hit both in the setup method that does your, your inserts to set up your test and in the code that you want to test.

[00:04:33] And maybe you only care about it once your test is actually running.

[00:04:37] So I run into that scenario a lot. And what I used to do is I would put a breakpoint in my test and disable or not even include the other breakpoint that I really care about.

[00:04:49] Wait till I hit my test one. Then I go in my code and find where I wanted to put the breakpoint that I really do care about, put that one in and then hit resume.

[00:04:58] That's kind of tedious.

[00:05:00] And so there's a better way, the better way is I can put the breakpoint in that I really care about and then put one in my test and then I can disable the one that I care about until the one in my test is hit.

Start visual description. The professor demonstrates how to disable a breakpoint until a specific previous breakpoint is hit. He provides an example scenario involving testing database logic. End visual description.

[00:05:14] That way. I don't have to pause in my testing and then go, go enable a breakpoint. So, I'll show you an example of that.

[00:05:23] So let's say, um let's re enable that one.

[00:05:29] OK? So, I've got these two breakpoints.

[00:05:31] And if you remember from this little um program, this ball clock program, there are two ways this can be executed, it can run for some number of minutes.

[00:05:40] And if so then the number of minutes will be specified on the command line, or it can just run until all the balls are back in the order that they started in.

[00:05:48] And let's say that I only care, I, I want to pause inside of the tick method, but I only care about that if it's executed um without a number of balls.

[00:05:57] So if it's executed in a state where it runs until all the balls are back in the same order, so what I can do is I can come in here and select this or turn on or create that breakpoint.

[00:06:10] And now this is the breakpoint that I really only care about if I came through this path in my code.

[00:06:16] So I can, I can easily make that happen.

[00:06:19] The way I can do that is go into my advanced breakpoint settings. First of all, um I only care about the breakpoint on line 90 if the one on line 64 is hit.

[00:06:30] So let's go to more and I can say disable until hitting the following break point and I'll say 64.

[00:06:41] So now what that means then is if I get to this tick method through this path, then this breakpoint doesn't count.

[00:06:51] But if I get there through this path, then it will pause when I get here.

[00:06:56] So that can be really useful.

[00:06:57] And this is kind of a made-up scenario that the more realistic scenario is the testing one where you have some setup code that might end up calling the function that has your breakpoint, and your test code does too. And it's the test code that you matter or that you care about.

[00:07:09] That's the this more likely scenario where you would care about that feature.

[00:07:16] OK? So, let's see what else we can do, we can make it. So once that breakpoint is hit, it will be disabled again. That's the default. So that means that um it would only be hit if the previous breakpoint was hit.

[00:07:31] And then once I do hit that breakpoint that I care about this, this one on line 90 it would be disabled once I hit it. If I don't want that, I can select leave enabled.

[00:07:41] Now, it will be enabled every time I hit it after that.

[00:07:45] So that can be pretty useful.

[00:07:47] There are just a couple of other features here that I want to tell you about.

[00:07:50] These, I use less of this one. I use um some but the others I use less often.

[00:07:55] So you have a pass count. So, let's set back that back to none.

[00:08:01] So sometimes you probably had this scenario before where you're, you're testing something in a loop and it's not the first few times iterations of the loop that you care about.

[00:08:09] It's maybe the 100th time.

[00:08:11] And so if you've ever just hit resume 50 times or 100 times or something until your, your program gets in the state that you care about, that's really tedious.

[00:08:20] I can avoid that by just clicking, pass count, typing 100. Now, the 1st 99 times that breakpoint is hit will be ignored by IntelliJ and on the 100th time, that's when it will pause.

Start visual description. The professor shows the "Pass Count" feature, which allows a breakpoint to be ignored for a specified number of hits before it pauses the program. He explains how this can be useful in loop testing. End visual description.

[00:08:32] So that can be really useful.

[00:08:34] Um The last few features here, I have some filters, and I really almost never use these.

[00:08:39] In fact, I'm not sure I ever have used them.

[00:08:41] I probably could use them and should use them some.

[00:08:45] Um But I have different ways to say whether a, a breakpoint should count or not, or whether my program should be paused in that breakpoint, probably the most useful of these three al although, as I said, I, I don't really use them but, but you could um probably the most useful the three is this one.

[00:09:03] If I select color filters, then I can specify that this breakpoint only counts if the function that contains it or the method that contains it was called by some specific method.

[00:09:16] So often you have multiple paths into a, a part of your code that you're trying to test.

[00:09:21] And you really only, you know that the bug is in one of those specific paths you can make it.

[00:09:25] So the breakpoint will only be hit from if it's called through that path.

[00:09:29] So that's what this color filter can do.

[00:09:32] Um Class filters are sometimes you have a breakpoint in some code in a parent class.

[00:09:38] And so that's inherited by multiple subclasses, maybe you only want the breakpoint to count if it's one of those spec one specific subclass, not all of them.

[00:09:47] And so you can specify that with a class filter and then finally instance filters.

[00:09:52] Um I'm not sure you would find much use for that one, but that allows you to specify that only a certain object that can, that contains the code.

[00:10:03] So you could have um you're putting your code in a class and there could be multiple instances of that class.

[00:10:08] And so your breakpoint is on some line of some class, maybe you only care about a particular instance um in your code and so you could specify that right here.

[00:10:20] Um So this, you probably need to, I'm, I'm not going to share all the details about how to um specify the syntax for these filters because you're not going to use that very often and you'd probably forget the details.

[00:10:32] What I'll show you instead is how you can find those um yourself, how you can look them up.

[00:10:43] And actually all of this advanced information you can find with a quick Google search, you can do IntelliJ breakpoints, click on this page and this starts with just the basic information that I've already shown shared with you mostly in the previous video.

[00:11:05] But if you scroll down here near the bottom, you get to this information about filters and how to set them. And it describes in words, some of what I've already described, it gives a few examples for um for doing different filters and, and how to use these breakpoints.

[00:11:24] So I invite you to spend some time reading that document and learning some more details about breakpoints.