CS 240: Lambda Expressions Overview Transcript

Start slide description.

Jerod Wilkerson presents a recorded lecture with the aid of a slide deck. All visual content is described in the audio.

End slide description.

[00:00:01] **JEROD WILKERSON:** Lambda expressions provide a way to specify code in a Java program that can be passed as parameters. It can be stored in variables. And basically any thing that you could do with a variable you can do with the lambda expression.

[00:00:17] So, it allows you to specify a block of code, basically, and the parameters that need to be passed to that block of code. So, if you're familiar with with threads in Java, you may or may not be, but it's similar to Runnables.

[00:00:31] If you're familiar with that, or if you're familiar with event handlers, when we have, well, let me back up to Runnables. So, if you have a thread, you need to have some code that can be executed in that thread, so that's a Runnable. So, that's similar to a Lambda expression.

[00:00:46] It's a chunk of code that's going to be executed later. And the same is true with event handlers.

[00:00:51] If we do anything with event programing, if we're building some kind of a gooey, we need to be able to have some way to specify a block of code that will be executed when some event happens.

[00:01:01] And Comparators are another example. We use Comparators in Java to sort things.

[00:01:09] So, you'll write the sort function ahead of time, so some block of code that can sort and then you pass that in to some method that's going to sort something when it's called.

[00:01:18] So, all of these things could be done before Java 8, before lambdas were added to the Java language, they were just a little bit tedious to do. In fact, I showed examples of how that's done in the generics lecture. But now with Lambda expressions, it's a lot simpler to create a block of code and pass it around and use it and call it later.

[00:01:41] So, just as I said it was always possible to do this. It was always possible to create a block of code.

[00:01:46] We typically did that by having some interface, and we create subclasses that implement the interface, and then we could pass those instances of the subclasses to functions that expected to receive the interface.

[00:02:01] So, that's the way it was done before, and it can still be done that way, but lambdas just provide a simpler kind of a shortcut notation. Basically what they do is they add the functional programing paradigm to Java.

[00:02:12] Functional programing wasn't a very big thing when Java was created, so that concept was added to Java later in Java version 8. So, here's an example of how that works. Let's start with an example the old way before we had lambdas, and then I'll show you how it changes with lambdas.

[00:02:30] So, right here we have this class "string length comparator." So, this is a class that implements the comparator interface and it can determine how it can be used in a sort function, so that's what the comparator interface is for.

[00:02:46] It allows you to create instances or implementing classes that classes that implement the interface that determine a sort function. So, the way that works

as comparator has one method compare and it takes two variables of the same data type.

[00:03:06] You can see we're using generics here. So, comparator is a generic class. So, when I create a specific comparator, I specify what the thing is that we're going to sort.

[00:03:17] So, in this case, it's strings. And so, that would mean that the compare method will take two strings. And what we need to do is return some integer that what the compare method is supposed to do is return an integer.

[00:03:29] It returns an integer that is less than zero. If the first thing should sort first, it returns zero If they're equal and have equal sort value and it returns a number greater than zero if the second item should actually be first.

[00:03:43] So, that's the way comparators work. And so, we just have to in this example, we're just going to sort by string length, so that's pretty easy to do because integer also has a comparator built into it.

[00:03:57] So, it already has a comparator method that will do the right thing if I passed two numbers, it will return a number less than zero, zero or greater than zero, depending on which one should sort first.

[00:04:07] So, all I really have to do to make this compare function work is call integers compare method and pass them the lengths of the two parameters and that will do the sort.

[00:04:18] So, now I have a comparator and so I can pass that in to some method or function that's going to do a sort for me, and that function can call that method when it needs to to determine the sort order.

[00:04:31] So, Java has on arrays, that sort method which will sort an array of whatever data type you want it to sort.

[00:04:39] So, in this example, we are calling arrays that sort, we're passing in an array of strings and we're passing the sort function right here, new string length comparator. So that's comparator that I had right here.

[00:04:52] So, that works and it's always worked in Java, but it's a little bit tedious because I have to create this class. I have to create a separate class to be the comparator, and then I have to pass an instance of it into the sort method.

[00:05:07] And a lot of that is simplified and that's really the purpose of lambdas is to simplify this situation. So, what I can do now and this arrays, that sort method takes two parameters.

[00:05:19] It takes a parameter, it takes some kind of an array, an array of some data type, and it takes something that implements comparator. So, with lambda expressions, I can use this shortcut notation for specifying the function or the algorithm for doing the sort.

[00:05:37] So, we look right here, I'm still passing in the string array and the code that is in bold is the lambda function. So, notice if we look back at the compare method right here, it needs to take two parameters first and second.

[00:05:51] And the reason I name them is so I can refer to them later in my code. I have the same need in a lambda expression, I need to be able to name my parameters so I can refer to them in my code. So, if we look right here, I have string first, string second as my parameter list.

[00:06:08] Here I have first and second as my parameter list. I can declare the data types in this parameter list, but I typically don't with lambda expressions because the compiler can infer them based on how I'm calling the code.

[00:06:20] So, I leave off the data types. Then I have this arrow notation and then I have my code to do the sort. So, that's just this line of code right here. Notice I don't have to have a return.

[00:06:32] I'll have more to say about that in a minute or in a future video, but that's built in. That's automatic. So, that is just very simplified syntax for doing the same thing I could do before.