CS 240: Use of Lambdas in Existing Java Classes Transcript

[00:00:00] **INSTRUCTOR:** Yeah, one of the things I've showed you in the earlier videos is how to create methods that take lambdas as parameters, but most of the time when you use lambda functions, you won't be doing that.

[00:00:12] Most of the time you'll just be calling methods that already exist in the Java classes that you get when you install Java or in some library that somebody else created.

[00:00:23] So, most of the time, you're just passing lambdas to some method that already was created to accept them.

[00:00:32] When lambdas were added to the Java language, lots of methods were added to existing classes, especially in the collections API that take lambdas as parameters, and that just made it easier to do a lot of things we need to do with different existing classes.

[00:00:47] So, here's an example of that.

Start slide description. Following content is shown:

Use of Lambdas in Existing Java Classes

Use of Lambda functions has been integrated throughout the Java API, especially in the collections API.

Example:

```
List<Integer> intList = new ArrayList<>();

intList.addAll(Arrays.asList(23, 5, 10, 71, 100, 1203, 4, 7, 748));

intList.removeIf(x -> x >= 100);
```

for (Integer value : intList) {

System.out.print(value + " ");

}

Prints:

23 5 10 71 4 7

End slide description.

[00:00:49] So, we have, in this example, actually I think I'll show you this example from my code editor, instead of the slide.

[00:00:59] So here, we are creating an array of integers.

[00:01:05] So, the first thing we're doing is we're creating an array list, and then we're adding a bunch of numbers to it.

Start visual description. The instructor creates a lambda array on the code editor. End visual description.

[00:01:14] Now, when you first look at this code, this is a little bit of a tangent.

[00:01:17] But when you first look at this code, you might think, why don't I just say list integer in list equals, and then move this code up to here and not create a new array list and then add stuff to it. That seems to make sense, but it won't quite work because this arrays dot as list is a method that makes it really easy to create a list of something, but the list is not, the list is immutable.

[00:01:40] It can't be changed.

[00:01:42] And so, the whole point of this example is going to be to remove some things from the list which can't be done with a list just created this way.

[00:01:49] So, what I need to do instead is I need to create my own list that is mutable, that can be changed, and then I'll just add the elements to it using this syntax.

[00:01:59] So, I now have a list of integers.

[00:02:02] And the list class is one of many classes that had methods added to it when lambda functions were added to the language.

[00:02:10] So, in this case, what we wanna do is we want to remove all of the elements from this list that are greater than 100, that their number is greater than or equal to 100.

[00:02:24] So, I could do that with a for loop or something, but I don't need to because I have a remove-if method now.

[00:02:30] So, I can just call remove-if and pass some lambda function.

[00:02:34] This is a predicate function.

[00:02:36] And in any case where it's true, this remove-if method will iterate through the elements of the list and check the predicate on each element.

[00:02:44] And in any case where that predicate is true, it'll remove the element.

[00:02:48] So, what I'll end up with is just the elements here that are less than 100.

[00:02:54] So, let's run this example to see that that is what we get.

[00:03:00] OK.

[00:03:01] So, I have created my list.

[00:03:03] I've called remove-if.

[00:03:05] And now, I have this for loop that will just print the values that are not removed.

[00:03:10] And notice that's all the values that are less than 100.

[00:03:13] So that's just one of many, many examples of how to use lambdas with existing classes that have already been written.