CS 240: Function Lambda Variables Transcript

[00:00:00] **INSTRUCTOR:** Most of the time when you're using lambdas, you're using a lambda expression as a parameter to some method call.

[00:00:06] But you can also use lambda expressions as variables.

[00:00:10] So, you could actually just create a variable that holds a lambda expression, and then you can pass that variable around, use it like you would use any other variable.

[00:00:18] So, here is syntax for that.

Start slide description. Following content is shown:

```
public class LambdaVariable {

Predicate<String> strLenPredicate = s -> s.length() > 10;

}
```

- Creates a predicate function (a function with a test method that returns a boolean) that in this case returns true if the String's length is > 10.
- The data type of the lambda expression is inferred by context (by the type of the variable to which it's being assigned—in this case Predicate<String>).
- Can be passed as a parameter to any method that takes a Predicate<String> parameter.
- Can be invoked by calling its test(String) method.
- Predicate is one of the 43 interfaces in the java.util.function package (you know what method to call by finding the abstract method in the API docs).

End slide description.

[00:00:20] So, if I have a class called lambda variable, it doesn't matter what my class is called, I can have a variable with a lambda type.

[00:00:30] So, the data type needs to be a functional interface.

[00:00:34] So that's any interface that fits the definition of a functional interface, which means it's an interface that has exactly one abstract method.

[00:00:41] So, predicate happens to be one of the 43 that was added when when lambdas were added to the Java language.

[00:00:48] So, predicate is a function that takes one parameter and returns true or false.

[00:00:56] And so, it's generic.

[00:00:58] Most of the function variables that were added to the functional interfaces that were added to Java are generic.

[00:01:07] So, I can specify the data type of the parameter that I'm passing in.

[00:01:11] So, here we're saying that we're creating a string predicate.

[00:01:14] So, it takes a string as a parameter, and it returns a Boolean.

[00:01:18] So, then I need to give like any variable, I need to give the variable a name so I can refer to it later, and then I set that equal to some lambda expression.

[00:01:28] So, here, the lambda expression, the data type of this lambda expression is known by it can be inferred based on what it's being assigned to you.

[00:01:36] So, it's being assigned to a predicate, so the JVM can tell this is a predicate lambda function.

[00:01:42] So, since it only takes one parameter, the parentheses are optional. So, I've left those off.

[00:01:48] So that's saying, this is a function that if you give it a string S, it will determine if that string's length is greater than 10.

[00:01:57] If it's greater than 10, it'll return true. Otherwise, it'll return false.

[00:02:04] So, this can be this predicate, now that I've created it, now that I've created this predicate variable, it can be passed to any method that accepts a predicate.

[00:02:14] So, it's not the most common thing to do.

[00:02:16] It's more common that you just declare the the lambda as you call the method, but you could have a reason to create some variable that you're gonna pass later.

[00:02:24] And so, that's what this does. Now, predicate happens to have one method called the methodist test.

[00:02:32] And so that means that if I create this predicate, this string length predicate, I would be passing it to some function that's expecting it, and that function would have a test method that it could call it we call a test method on this lambda function.