Paul Romer

CSD340

Module 9 Assignment

Lambda

Lambda expressions are not only a feature in Java, but in a lot of other programming languages as well. Lambda expressions, also known as anonymous functions, were brought to Java in Java 8, in March of 2014. Wikipedia defines an anonymous function as a “function definition that is not bound to an identifier (Wikipedia)”. A way to define a function without assigning it to a variable. These are used in Java when working with Functional Interfaces. When a method within an Interface doesn’t have an implementation, it’s called an abstract method, and when an Interface has one abstract method, we call it a Functional Interface. Lambda expressions give us a way to define the function of the abstract method of a Functional Interface.

Instead of initializing the class and passing in an object that contains a specific implementation of an abstract method, Lambdas give us a shortcut which allows us to pass in the specific implementation of the method directly. They also give us the ability to turn the Lambda into the object.

There are a few ways to structure of a Lambda Expression. The simplest is parameters within parentheses, followed by the arrow operator, and the body or code to be run within brackets - (parameters) -> body;. This structure `(x, y) -> x + y;` is a lambda expression that takes two integers as inputs and returns the sum. A lambda can also be defined including a block of code within the body that will be run. The structure of this one is (x, y) -> {x+y;}. If there is only one statement, the parentheses are not required. In this example, the return statement isn’t required because the compiler understands that if there is only one statement in the body, it should be returned.

One of the biggest benefits of using Lambda Expressions is that it helps the developer create cleaner and more simple code. Instead of needing to define a class that implements the method from the interface, creating an instance of that class, and then being able to use it, Lambdas can do all of that in one line. They can be assigned to variables. They provide a way to define a function without creating a class.

The included java files give an example of using Lambda Expressions. It includes a MathOperation interface which is a Functional Interface. Calculator.java then defines the interface using a Lambda Expression, defining that it returns the sum of the two integers that are passed into the function. This Lambda is assigned to `sum`. Then sum.operation is called, and the resulting sum is returned, assigned to `result` and printed to the terminal.

While Lambda Expressions are a powerful tool for the Java developer, they have some limitations. First, they can only be used with functional interfaces. That means that we can’t use Lambda Expressions with classes or objects that have multiple abstract methods. Since Lambdas don’t have names, it can be harder to understand their purpose at a glance. This can also make it difficult to debug as the error may display a generic name <lambda> instead of a function name. Because of this, it is best practice to avoid putting complex logic in lambdas.

Lambda expressions in Java are commonly used with action listeners. Action listeners are used to respond to button clicks, menu selections, etc. The oracle resource in the sources section has a list of quite a few Functional Interfaces in Java where lambda expressions could be used. They are also used in data processing and multithreading.

In conclusion, lambda expressions in Java are a powerful feature that offers multiple benefits. They help the Java developer write more conciseness and flexible code. By following best practices of using good naming and avoiding logic that is too complex, using Lambda expressions can be an essential tool for any Java developer working with Functional Interfaces like ActionListeners, or building UI, working with data, or multithreading.

Sources:

<https://www.youtube.com/watch?v=tj5sLSFjVj4&list=WL&index=6&t=97s>

<https://en.wikipedia.org/wiki/Anonymous_function>

<https://www.w3schools.com/java/java_lambda.asp>

<https://www.programiz.com/java-programming/lambda-expression>

<https://www.geeksforgeeks.org/lambda-expressions-java-8/>

<https://www.youtube.com/watch?v=LEJ1kGHSXdA&t=278s>

<https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html>