JAVA’S LAMBDA

Tyler O’Riley

CSD405

06/28/2023

To discuss Java’s Lambda and how it is utilized in Java code, it is important to first begin reviewing the type of code it is utilized for. Interfaces are a fundamental tool for Java that allow a user to pull various methods and data from other Java users for blueprints to concepts they may not know how to execute themselves. In concept this can be easy to understand as you define the implemented program as Implement (interface name) rather than how you name normal classes as public class (class name).

This type of Java code follows the idea of abstraction where we are using the signatures of the methods and data from the implemented information which can be useful in hiding the implementation from the user. The signatures are separate from the main class, but the methods of these signatures still need to be created within your class in order to run them. Once these methods have been written you can then assign an object to run the methods and produce results based on the information you implemented from the referenced data. The main benefits of using this form of Java are achieving states of abstraction, effective inheritance, and loose coupling by separating the signatures from where the methods are stored. The downside of this approach is the amount of code that has to be written to utilize implementation. One way Java has sought to address this issue is the introduction of Java’s Lambda.

Lambda expressions in Java are designed to provide shorthand syntax for implementing an interface without the need for excessive code. As you can recall above, there are multiple stipulations for using an interface with your Java program. An instance of the method needs to be created, an object to implement the method must be created, and supporting code for structure of your class must be made to ensure it runs correctly (Ex. @Override). The usage of lambda can depend on how and what you are passing. With methods that aren’t passing parameters, most syntax can be subtracted from your code as a simple arrow sign using dash and greater than (->) is the lambda to pass the method. This symbol disregards the need for void and public and all other syntax as it is simply telling Java to return the information based on what the interface is able to do.

This style of Java coding can be very effective in reducing the amount of code otherwise needed and depending on the size of your project this can assist in saving a lot of time and effort. The downside to lambdas that needs to be taken into consideration is the fact it is shorthand coding. To avoid the need of scrolling back and forth to see what your lambdas are referencing, it becomes more critical to enter proper notes to assist others who may review your code. The lambda is fantastic in giving your short, more straight forward code, but can hinder others who may be working on the same project as you within a business function.

Where is it appropriate to use a lambda if it has the potential of being a detriment? You can use the lambda whenever you feel appropriate, but I did find a few examples of where it becomes a life saver. When creating collections inside Java, the amount of code needed can start to pile up fast. Implementation of a lambda in this instance can serve to reduce the amount needed immensely. In this situation you may find your code easier to read than difficult. The reason being is how much information your eyes must review to understand the big picture of your program. Although you are using shorthand, if organized and notated effectively, you find your program more readable and easier to understand.

Lambda’s are one of Java’s newer features and assists in making the language easier to write to help make it more effective against languages that pride themselves on simplicity such as Python. There are multitudes of ways to use Python Lambda’s it will just be a matter of review to see which form best assists for any one situation. My attached code shows this lambda used in the context of a simple implementation of non-parameter method building.

SOURCES

“Java Lambda Expressions - Javatpoint.” *Www.Javatpoint.Com*, www.javatpoint.com/java-lambda-expressions. Accessed 28 June 2023.

“Lambda Expressions in Java - Full Simple Tutorial.” *YouTube*, 7 Feb. 2022, www.youtube.com/watch?v=tj5sLSFjVj4&t=202s.

“Java Interface Tutorial - Learn Interfaces in Java.” *YouTube*, 10 Oct. 2019, www.youtube.com/watch?v=kTpp5n\_CppQ.

S, Ravikiran A. “What Is Java Interface and Why It’s Needed?” *Simplilearn.Com*, 27 Sept. 2022, www.simplilearn.com/tutorials/java-tutorial/java-interface#:~:text=and%20project%20performance.-,What%20is%20Interface%20in%20Java%3F,in%20Java%20to%20achieve%20abstraction.