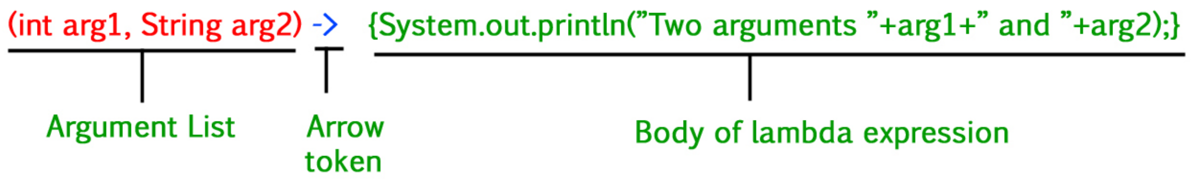
Jason Palmeri
Assignment 9
3/29/2022
CSD405

Lambda

Lambda was added in Java version 8, and is a “short block of code which takes in parameters and returns a value” (w3). Similar to methods, lambda expressions take in parameters and execute code with them, with the main difference between the two being methods require names, while lambda expressions do not need a name and can be implemented right in the body of a method. W3 explains the simple syntax of a lambda expression as “parameter -> expression”, or if you want to use more than one parameter: “(parameter1, parameter2) -> expression”. “Expressions are limited. They have to immediately return a value, and they cannot contain variables, assignments or statements such as `if` or `for`” (w3). While you can not do these complex operations with the example syntax, you can create a code block for your lambda expression:


(int arg1, String arg2) -> {System.out.println(”Two arguments ”+arg1+” and ”+arg2);}
Argument List
Arrow token
Body of lambda expression

Lambda expressions can be stored in variables, if the variable’s type is an interface which has one method. With this solution the lambda expression should have the same number of parameters and the same return type as the method. A useful implementation of Lambda expressions in Java is to use them to instantiate functional interfaces. “The compiler will allow us to use an inner class to instantiate a functional interface; however, this can lead to very verbose code. We should prefer to use lambda expressions” (Baeldung). We can instantiate a functional interface using Lambda expressions like:
myClass x = parameter -> parameter

As I was searching around for more information on Lambda in Java, I asked myself why even use Lambda? Are they any benefits to using it, or are there more drawbacks? Tutorialspoint had a great list of reasons on why we use Lambda expressions:

- **Enables functional programming**
  - “All new JVM based languages take advantage of the functional paradigm in their applications, but forced programmers to work with **Object-Oriented Programming** till lambda expressions came. Hence lambda expressions enable us to write **functional code**” (Raja)
- **Readable and concise code**

    - “People have started using lambda expressions and reported that it can help to remove a huge number of lines from their code” (Raja)
- **Easy-to-Use API’s and Libraries**
    - “An API designed using lambda expressions can be easier to use and allow support for other API’s” (Raja)
- **Enables support for parallel processing**
    - “A Lambda expression can also enable us to write **parallel processing** because every processor is a multi-core processor nowadays” (Raja)

At the end of the day, Lambda expressions seem like a great way to write code in Java, and being able to write readable code is great, as well as having to write less of it, but everything usually has a downside, so I found an article on jrebel.com that had a list of cons when it comes to lambda expressions. They start off thesection by saying how in the real world we “use Maven or Gradle-based projects for managing dependencies. Usually you use an IDE in order to develop your application…” (JRebel). They then give two important cons to using Lambda expressions:

- **Dependency Management**
    - JRebel talks about how it can be hard using Lambdas with libraries and APIs because you don't always have the source you are using, and with a concise syntax like lambda you will likely have to navigate through API documentation for quite some time trying to figure out what you are looking at.
- **Useability without an IDE**
    - JRebel talks about how IDEs really do help us as developers alot to figure out things that are happening in our code as we write it, with error messages and instant spell checking.

JRebel finishes off by saying “The big paradox of lambdas is that they are so syntactically simple to write and so tough to understand if you’re not used to them. So if you have to quickly determine what the code is doing, the abstraction brought in by lambdas, even as it simplifies the Java syntax, will be hard to understand quickly and easily” (JRebel).

Citations

Kaushal, S. (2018, August 13). *Lambda expressions in java 8*. GeeksforGeeks. Retrieved March 31, 2022, from https://www.geeksforgeeks.org/lambda-expressions-java-8/

Baeldung, B. (2021, May 15). *Lambda expressions and functional interfaces: Tips and best practices*. Baeldung. Retrieved March 31, 2022, from https://www.baeldung.com/java-8-lambda-expressions-tips

Oracle. (n.d.). *Lambda expressions*. Lambda Expressions (The Java™ Tutorials > Learning the Java Language > Classes and Objects). Retrieved March 31, 2022, from https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

W3. (n.d.). *Java Lambda*. Java Lambda expressions. Retrieved March 31, 2022, from https://www.w3schools.com/java/java_lambda.asp

Raja. (2019, December 26). Why we use lambda expressions in Java? Retrieved April 2, 2022, from https://www.tutorialspoint.com/why-we-use-lambda-expressions-in-java

JRebel. (2014, March 4). *Pros and cons of lambdas in java 8*. JRebel by Perforce. Retrieved April 2, 2022, from https://www.jrebel.com/blog/pros-and-cons-of-lambdas-in-java-8#:~:text=The%20big%20paradox%20of%20lambdas,to%20understand%20quickly%20and%20easily.