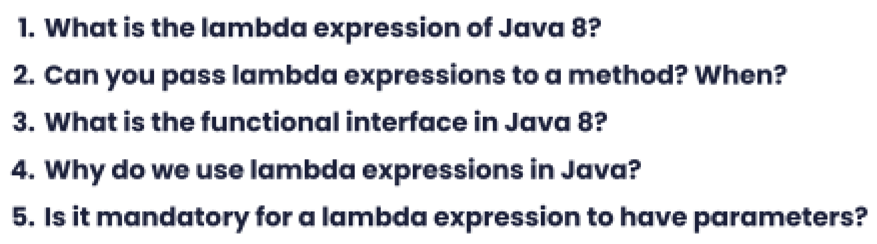
# day-023-lamda-expression



1. What is the lambda expression of Java 8?
2. Can you pass lambda expressions to a method? When?
3. What is the functional interface in Java 8?
4. Why do we use lambda expressions in Java?
5. Is it mandatory for a lambda expression to have parameters?

1. In Java 8, a lambda expression is a concise way to represent a functional interface, which is an interface with only one abstract method. The syntax for a lambda expression is "(parameters) -> {body}", where "parameters" are the input arguments for the method and "body" is the code to be executed. For example, a lambda expression for a method that takes two integers and returns their sum could be written as "(int a, int b) -> { return a + b; }".

2. Yes, lambda expressions can be passed as arguments to methods. This is because a lambda expression is essentially a function that can be treated like any other object in Java. You can pass a lambda expression to a method whenever the method's parameter type is a functional interface that matches the signature of the lambda expression. For example, if a method takes a parameter of type "Consumer<String>", which is a functional interface with one abstract method that takes a String argument and returns void, you can pass a lambda expression that takes a String parameter and does something with it.

3. A functional interface is an interface that has exactly one abstract method and is annotated with the "@FunctionalInterface" annotation. Java 8 introduced several new functional interfaces in the java.util.function package, such as Function, Predicate, and Consumer, that represent common function types. Functional interfaces provide a way to define a single method interface that can be implemented using a lambda expression, making it easier to write concise and expressive code.

4. Lambda expressions are used in Java to provide a concise way to represent functional interfaces. They allow developers to write shorter and more expressive code by avoiding the need to write verbose anonymous inner classes. Lambda expressions also make it easier to write code that is more modular and reusable, as they allow developers to pass functions as parameters to other methods, which can be especially useful in functional programming and stream processing.

5. No, a lambda expression doesn't have to have parameters. If a lambda expression doesn't take any arguments, you can simply leave the parentheses empty. For example, a lambda expression that returns a constant value of 42 could be written as "() -> 42".