# Introduction to Lambda Expressions in Java

## Introduction

In this tutorial, we will discuss

1. \*\*Lambda expressions\*\*,
2. their syntax, and
3. their usage.

This session focuses on the \*\*theory\*\* behind Lambdas in Java.

## What is a Lambda Expression?

- A \*\*Lambda expression\*\* is equivalent to **a function \*\*without a name**\*\*.

- That is why It is also known as an \*\***anonymous function**\*\* in Java.

- Like a regular method, a lambda has:

- Method parameters

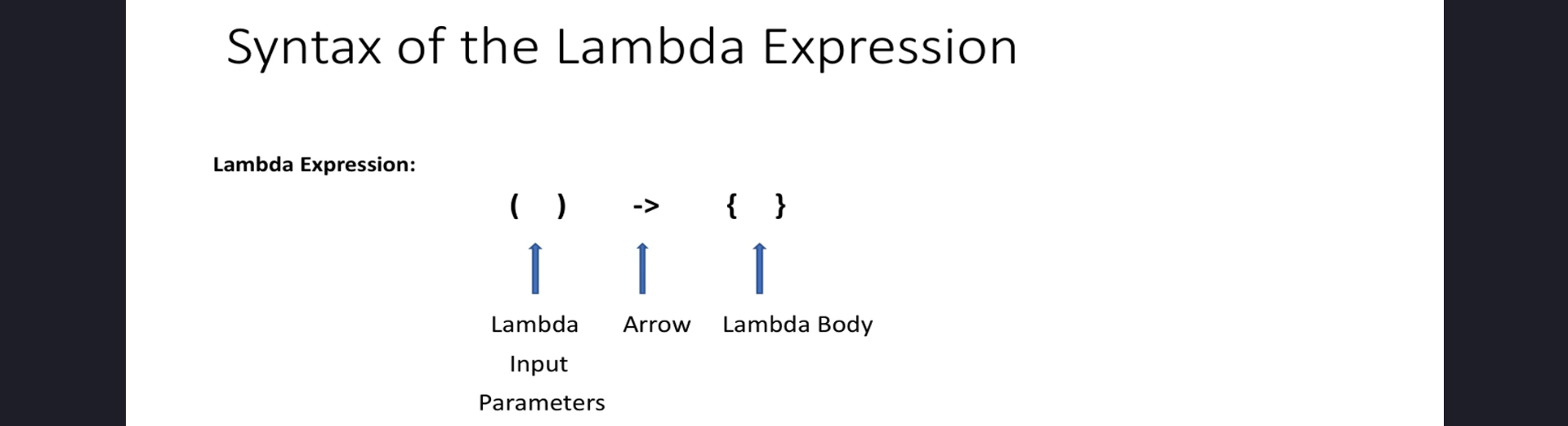
- Method body

- Return type

- Unlike regular methods, a \*\*Lambda is not tied to any class\*\*.

- It can be \*\***assigned to a variable**\*\* and **passed** like any other variable in Java.

## Lambda Expression Syntax

### \*\*Structure of a Lambda Expression:\*\*  


- \*\*Parentheses\*\* `( )` are used to pass input parameters.

- \*\*Arrow operator\*\* `->` separates input parameters from the method body.

- \*\*Curly braces\*\* `{ }` contain the method body, which defines the operation.

Example of a Lambda Expression:

(parameters) -> { method body }

## Usages of Lambda Expressions

### \*\*Implementing **Functional Interfaces**\*\*

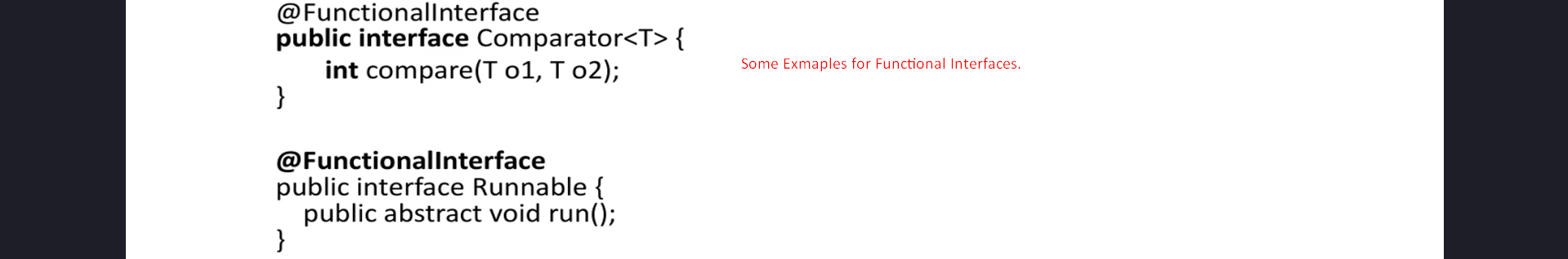
- The main purpose of Lambda expressions is to implement \*\*Functional Interfaces\*\*.

- Functional Interfaces are also called \*\***SAM (Single Abstract Method) Interfaces**\*\*.

- A \*\*Functional Interface\*\* is an interface that contains only **\*\*one\*\* abstract method**.

- These interfaces are annotated with **\*\*@FunctionalInterface\*\*.**

### \*\*Examples of Functional Interfaces Before Java 8\*\*

Some existing functional interfaces before Java 8 include:  


- \*\*Runnable\*\* (used for defining tasks in threads)

- \*\*Comparator\*\* (used for sorting collections)

## Next Steps

- In the next tutorial, we will implement \*\*Comparator\*\* and \*\*Runnable\*\* interfaces using Lambda expressions.

- We will also explore various use cases and benefits of Lambdas in real-world applications.

## Conclusion

- Lambda expressions provide a concise way to write \*\***anonymous functions**\*\* in Java.

- They are primarily used for \*\***implementing functional interfaces**\*\*.

- Java 8 introduced many built-in functional interfaces that can be used with Lambdas.

- Understanding Lambda expressions is essential for writing \*\*modern Java code\*\* efficiently.

### \*\*End of the Tutorial\*\*

Thank you for watching!