# Why Java 8 is Important?

## Introduction

In this video, the importance of Java 8 and why Java developers should invest time in learning its features is discussed. Java is one of the most popular and widely accepted languages globally. Despite its widespread use, Java creators introduced major changes in Java 8 to keep up with **technological advancements**.

## Why Did Java Introduce Functional Features?

Java 8 introduced functional features such as:

- \*\*Lambdas\*\*

- \*\*Streams\*\*

- \*\*Optionals\*\*, and more

These features were added due to the increasing complexity in handling concurrency and parallelism. Modern devices, including smartphones, have multiple processors, but prior to Java 8, utilizing these processors required managing threads manually.

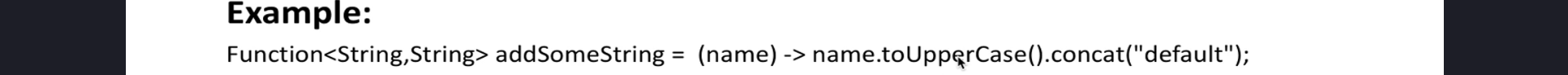
Working with threads is complex and requires careful handling. Java 8 simplifies concurrency operations by allowing developers to write functional programming code, making it easier to utilize multiple processors efficiently.

## Benefits of Functional Programming in Java 8

Functional programming in Java 8 brings several advantages:

- \*\***Creation of Immutable Objects**\*\*: Helps in reducing errors in Java applications, as mutable objects are a common source of bugs.

- \*\***Concise and Readable Code**\*\*: Functional programming allows writing cleaner and more maintainable code.

- \*\***First-Class Functions**\*\*: Methods and functions can be treated as variables, assigned to variables, and passed as parameters to other functions, something that was not possible before Java 8.  


## Lambda Expressions

Java 8 introduced the \*\*lambda syntax\*\*, which allows defining functions concisely. Below is an example demonstrating lambda expressions:

- The input is a string that undergoes an uppercase transformation followed by concatenation.

- The logic is assigned to a variable, `addSomeString`, which can be used like any other variable in Java.

This is one of the key benefits of functional programming in Java 8.

## Imperative vs. Declarative Programming

### \*\***Imperative Programming**\*\*

This style focuses on \*how\* to achieve a result by specifying step-by-step instructions.

- Involves modifying the state of objects (mutability).

- Used in traditional object-oriented programming.

Example: If you want to achieve a certain objective, you explicitly define each step needed to reach that goal.

### \*\*Declarative Programming\*\*

This style focuses on \*what\* result is needed, without worrying about how the result is computed.

- Emphasizes immutability.

- Similar to SQL queries where data retrieval logic is abstracted.

Example: In SQL, we specify \*what\* data we need from tables, without worrying about the underlying algorithm used to fetch it. Java 8 promotes a declarative programming approach using built-in functions.

## Conclusion

Functional programming in Java 8 leverages declarative programming, enabling developers to write efficient, readable, and bug-free code. The course will cover the differences between imperative and declarative styles with practical examples in the upcoming tutorials.

This marks the end of the theory section. Thank you for watching!