Document Generated: 11/14/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Intermediate

Course Duration: 1 Day

Migrating Java 11 to Java 21 | Java 21 New Features and Skills (TT2136)



About This Course:

There have been a lot of changes in the Java programming language since the release of Java 11 in September of 2018. Migrating Java 11 to Java 21 is a one day course that focuses on the changes that will be most noticable for Java developers who will make the transition from Java 11 to Java 21.

This course blends expert instruction and practical, hands-on experience, focusing on bridging the gap between your current Java knowledge and the latest version. Understanding that many developers are now transitioning projects from older versions of Java, this course is your key to making that shift both quickly and effectively. Throughout the course, Throughout the course, you'll dive deep into Java 21's key features, such as Records for efficient data management, enhanced text manipulation with String and Text Blocks, the introduction of Sealed Classes, and the refinements in Switch Expressions and Pattern Matching. These updates are not mere enhancements but pivotal tools that will streamline your coding practices and make your development process more efficient. The real-world labs enable you to quickly pick up useful practical skills you'll be able to apply seamlessly into your work.

Whether you're upgrading existing projects or embarking on new ones, the skills and insights gained here will empower you to use Java 21 efficiently and innovatively. This course is more than just an upgrade in your Java knowledge; it's a comprehensive enhancement of your development skills, preparing you to tackle modern challenges with renewed expertise and confidence.

Course Objectives:

- The difference between LTS and non-LTS versions
- What are the (dis)advantages of preview features
- To implement data objects using the new record type
- Improve memory consumption through the use of compact Strings
- The use of the new strip and inBlank methods of the String class
- Define and use multi-line text blocks
- Use switch expressions to assign value
- Understand the concept of pattern matching
- Apply pattern matching for instanceof
- Use pattern matching in switch statements
- Sealed classes and interfaces
- Enhancements made to the try-with-resources construct
- Review the Java modular system
- Explore virtual Threads

Audience:

- This is an intermediate- level Java programming course, designed for experienced Java 11 developers who wish to get up and running with Java 21 immediately. Attendees should have a working knowledge of developing Java 11 applications.

Prerequisites:

- This course is not for non-developers, or developers new to Java.

Course Outline:

1. Versions and Features

- Understand the key changes in Java versions beyond Java 11.
- Explore improvements in Java runtime and garbage collection.
- Familiarize with new language features such as switch expressions and text blocks.
- Learn about new APIs and enhancements including HttpClient and the Foreign Function and Memory API.
- Recognize the impact of the Java Modular System on application structure and encapsulation.
- Tutorial: Importing code Using Maven
- Lab: Defining Modules
- Lab: Modules and the ServiceLoader (optional)

2. Records

- Understand the purpose and structure of data objects in Java.
- Recognize the role of IDE tools in generating code for data objects.
- Learn about the Java record class and its advantages for data representation.
- Implement interfaces with records to enhance functionality.
- Create and use canonical, alternative, and compact constructors in records.
- Apply annotations to records.
- Lab: Records

3. String and Text Blocks

- Define whitespace in Java and its significance in code readability and input parsing
- Identify and utilize Java methods for handling whitespace in strings.

- Explain the evolution of whitespace definitions due to Unicode updates.
- Utilize `trim()`, `strip()`, `stripLeading()`, and `stripTrailing()` methods for string whitespace management.
- Differentiate between `isEmpty()` and `isBlank()` methods for string content validation.
- Implement text blocks for improved multi-line string readability and formatting.
- Employ the `formatted` method for dynamic string creation.
- Use the `lines()` method to process multi-line strings efficiently.
- Recognize the impact of compact strings on memory efficiency.
- Lab: Text Blocks

4. Sealed Classes

- Understand the concept of Java interface contracts.
- Explore the limitations of defining types for interface contracts.
- Analyze the use of final and package-private constructors in Java.
- Grasp the significance of sealed types in managing subclassing.
- Recognize the advantages of sealed classes for robust software design.
- Lab: Sealed Classes

5. Switch Expressions

- Recognize how switch statements simplify decision-making in code.
- Differentiate between traditional switch statements and switch expressions.
- Comprehend the concept of fall-through semantics in switch statements.
- Apply the yield statement to produce values in switch expressions.
- Create exhaustive switch expressions to handle all potential input scenarios.
- Lab: Switch Expressions

6. Pattern Matching

- Understand the enhancements to the instanceof operator in Java 16.
- Learn how pattern matching simplifies type checking and casting.
- Identify limitations of switch statements prior to Java 21
- Explore the role of pattern matching within switch statements.
- Recognize the importance of exhaustive switch statements.
- Understand the role of enum constants in switch expressions.
- Discover how sealed classes affect switch statement requirements.
- Lab: Pattern Matching

7. Virtual Threads

- Understand the concept and purpose of virtual threads in Java.
- Describe how virtual threads improve concurrency and scalability.
- Identify the difference between platform threads and virtual threads.
- Learn how to create and manage virtual threads using Java APIs.
- Implement thread-local variables with virtual threads.
- Utilize CompletableFuture with virtual threads for asynchronous tasks.

- Explore the use of InheritableThreadLocal for sharing data with child threads.
- Lab: Virtual Threads

Additional Topic: Time Permitting

These topics will be included in your course materials but may or may not be presented during the live class depending on the pace of the course and attendee skill level and participation.

1) More Updates

- Sequenced Collections
- private methods in interfaces
- The forRemoval and since attributes of the Deprecated annotation
- Multi-release JAR files
- Javadoc updates
- Lab: Creating a Multi-Release Jar file (optional)