Migrating Java 8 to Java 17 | Java 17 New Features and Skills - TT2138

Jumpstart your Java 17 Skills | Explore New features, Versioning, Jigsaw, JShell, Concurency, Performance Enhancements and More

Duration: 2 Days
Skill Level: Introductory
Available Format: Instructor-Led Online ; On Public Schedule

Two-day, hands-on fast-track course geared for developers who have prior hands-on experience working with Java 8, who need to quickly get up and running the latest features introduced in Java 17.

What You'll Learn

Overview

Migrating from Java 8 to Java 17 is a two day, hands on fast-track course geared for developers who have prior hands on experience working with Java 8, who need to quickly get up and running the latest features introduced in Java 17. Throughout the course students learn the best practices for taking advantage of the new Java Module system as well as other new features in this major update to the Java programming language.

The Java 11 update introduced major changes to the core language, including new features such as the Java Module system , As well as several small enhancements to the language as part of the Milling Project Coin project . Project Coin was introduced during the development of Java 7 to introduce small language changes. Milling Project Coin introduced several enhancements that did not make it into Java 7. This course provides a fast-pace, high level overview of the some of the lesser-known languages changes that were introduced over the years. Several of these small changes have laid the foundation for the enhancements made in Java 9, 10 and 11.

Objectives

This 'skills-centric' course is about 50% hands-on lab and 50% lecture, designed to train attendees in core next-level Java development skills, coupling the most current, effective techniques with the soundest industry practices. Our engaging instructors and mentors are highly experienced practitioners who bring years of current "on-the-job" experience into every classroom.

- Migrate existing Java 8 applications: Take all the steps needed to migrate your existing Java 8 application to Java 17
- Mastering Records in Java 17: Gain a thorough understanding of how to utilize Records in Java 17 for efficient data handling, enabling you to simplify data modeling and ensure more concise and readable code.
- Advanced String and Text Block Management: Develop the ability to effectively use the new String and Text Block features in Java 17. This includes mastering methods like strip(), isBlank(), and repeat(), as well as understanding the nuances of indentation in text blocks for improved text manipulation.
- Implementing Sealed Classes and Interfaces: Acquire the skills to implement sealed classes and interfaces in Java 17, enhancing your ability to create more secure, type-safe, and maintainable code.
- Utilizing Enhanced Switch Expressions and Pattern Matching: Learn to effectively use enhanced switch expressions and pattern matching for instanceof in Java 17. This skill is crucial for writing more concise and error-resistant code.
- Efficient Exception Handling Techniques: Master the enhanced exception handling features in Java 17, such as the improved try-with-resources statement and more informative NullPointerExceptions, to write more robust and reliable Java applications.
- Applying Modern Java Development Techniques: Develop the ability to apply these new Java 17 features in real-world scenarios. This includes integrating these skills into existing projects, enhancing your capacity to upgrade and modernize Java applications efficiently.
- Specific Java 17 features that are covered in the course including: Switch Expressions; Text blocks; Pattern matching for instanceof; Introducing records as carrier of immutable data

In addition to several exercises that are done during class does this training also includes a project to be completed after class. Students will take an existing Java 8 application written using JUnit, Project Lombok, JAXB and JavaFX and update it to run on Java 11**. In several individual steps students will:**

- Take the existing application and make it run on Java without modifying the source code
- Update the application, converting it to a proper Java Module
- Update the libraries to the latest Jakarta version
- Utilize the ServiceLoader mechanism to decouple the components
- Use JLink to build a custom runtime environment

Throughout these exercises students will run into some of the challenges that might be encountered while going through the process of updating an existing application from Java 8 to Java 11

Audience

This is an intermediate- level Java programming course, designed for experienced Java 8 developers who wish to get up and running with Java 17 immediately. Attendees should have a working knowledge of developing Java 8 applications.

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

Pre-Requisites

Attendees should have a working knowledge of developing Java 8 applications.

Agenda

Please note that this list of topics is based on our standard course offering, evolved from typical industry uses and trends. We'll work with you to tune this course and level of coverage to target the skills you need most. Topics, agenda and labs are subject to change, and may adjust during live delivery based on audience skill level, interests and participation.

Session: Java 8 to Java 17

Lesson: Versions and Features

- Overview of Java versions since Java 8
- (Non) LTS releases
- Preview features
- Java Language Specifications
- Java Specification Requests
- Java Enhancement Proposals

Lesson: Milling Project Coin

- Changes made to the language since Java 6

- Multi-catch
- Using effectively final variables in try-with-resources
- Suppressed exceptions
- Binary literals
- Reserved underscore
- Type inference in anonymous classes
- @SafeVargs (updates in Java 9)
- Default and static methods in interfaces
- Private methods in interfaces
- Tutorial: Importing Exercises into IntelliJ 2022 (Community Edition)
- Lab: Try-With-Resources

Lesson: Local Variable Type Inference

- Explain type inference
- Infering types of local variables
- The var reserved type name
- Benefits of using var
- Backward compatibilty
- Lab: Variable type inference

Lesson: Using Strings in Java 11

- Working with Strings
- Discuss the definition of whitespace in Java
- Introduce the new strip() methods of the String class
- The isBlank() and repeat() methods introduced in Java 11
- Using the lines() method to construct a Stream instance using a String
- Compact strings
- String deduplication

Lesson: String and Text Blocks

- Discuss the definition of whitespace in Java
- Introduce the strip() methods of the String class
- The isBlank() and repeat() methods introduced in Java 11
- Using the lines() method to construct a Stream instance using a String
- Compact strings
- Introducing Text Blocks
- Indentation in text blocks

- Lab: Text Blocks

Lesson: Records

- Data objects in Java
- Introduce records as carrier of immutable data
- Defining records
- The Canonical constructor
- Compact constructors
- Lab: Records

Lesson: Switch Expressions

- Switch Expressions
- Using yield
- Discuss switch fall through
- Lab: Switch Expressions

Lesson: Pattern Matching

- Pattern Matching for instanceof
- Scope of variable when using pattern matching
- Refining Patterns in switch
- Dominance of pattern labels
- Lab: Pattern Matching

Lesson: Sealed Classes

- Introduce sealed classes
- The sealed and permits modifier
- Sealed interfaces
- Sealed classes and pattern matching
- Lab: Sealed Classes

Lesson: Exception Handling

- Enhanced try-with-resources
- Helpful NullpointerExceptions
- Excluding parameter names in error messages
- Lab: Helpful Nullpointers

Lesson: Updates to Collections and Streams

- Introduce enhancements to the Collection API
- Unmodifiable Collections
- Introduce new functionality of the Stream API
- Explain new intermediate Stream operations (takeWhile, dropWhile)
- Collecting Stream elements into list using the new toList method
- Lab: Updated Streams

Lesson: More Updates

- Private methods in interfaces
- The forRemoval and since attributes of the Deprecated annotation
- Multi-release JAR files
- Javadoc updates
- Class-Data Sharing
- Application Class-Data
- CompactNumberFormat
- Lab: Creating a Multi-Release Jar file

Session: The Java Module system (Jigsaw)

Lesson: Why JigSaw?

- Problems with Classpath
- Encapsulation and the public access modifier
- Application memory footprint
- Java 8's compact profile
- Using internal JDK APIs

Lesson: Introduction to the Module System

- Introduce Project Jigsaw
- Classpath and Encapsulation
- The JDK internal APIs
- Java 9 Platform modules
- Defining application modules
- Define module dependencies
- Implicit dependencies
- Implied Readability
- Exporting packages
- Lab: Defining Modules

Lesson: The Module Descriptor

- Define module requirements

- Explain qualified exports
- Open modules for reflection
- Use ServiceLoader
- The provides and uses keywords
- Lab: Modules and the ServiceLoader
- Lab: Using Reflection on modules (optional)

Additional Topics: 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.

Lesson: Migrating an Application

- Migrating a java 8 application to Java 11
- Replacing libraries removed from the Java Standard Edition
- Explore the challenges

Lesson: Java 9 Concurrency Updates

- Brief overview of Concurrency in Java
- Overview of CompletableFuture (Java 8)
- Subclassing the CompletableFuture
- The default Executor
- New Factory methods
- Dealing with time-outs
- Lab: Completables (optional)

Evening Review / Homework

- Lab: Flight Application (approx 30 minutes)
- Lab: Migrating from Java 8 to Java 11 (approx 45 minutes)
- Lab: Migrating from Java 8 to Java 11 (part 2) (approx 30 minutes)
- Lab: Migrating from Java 8 to Java 11 (part 3) (approx 45 minutes)

For More Information

Please contact us or call 844-475-4559 toll free for more information about our training services (instructor-led, self-paced or blended), coaching and mentoring services, public course enrollment or questions, partner programs, courseware licensing options and more.