Advanced Java 9

Price	Duration	Delivery Methods
$2,796.00	4 Days	VILT, Private Group

This course provides an in-depth treatment of the many significant Java 9 features and updates, with the goal of demonstrating how these features can be used to improve the performance and functionality of Java applications.

Who Should Attend

Programmers with prior Java 8 or 9 programming experience

This class is not currently scheduled.

Contact us and we will help you get the training you need!

Course Objectives

Students who attend this course will leave armed with new skills to leverage Modules, scale applications into multi-core environments and improve the performance of Java 9 applications. This course will teach students everything they need to successfully master and implement the latest features and benefits of Java 9 and become a more effective Java 9 developer.

Agenda

1 - REVIEW OF WHAT IS NEW IN JAVA 9

- Overview of 'smaller' Java 9 topics
- Java versioning
- The JDK/JRE file structure
- Deprecation
- The jdeprscan tool
- Multi-Release JAR Files
- HTML 5 compliant JavaDoc
- Exercise: Creating a MRJar

2 - 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 (Java 9)
- Type inference in anonymous classes (Java 9)
- @SafeVargs (updates in Java 9)
- Default and static methods in interfaces (Java 8)
- Private methods in interfaces (Java 9)
- Exercise: Try-With-Resources

3 - WHY JIGSAW?

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

4 - 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
- Exercise: Defining Modules

5 - THE MODULE DESCRIPTOR

- Define module requirements
- Explain qualified exports
- Open modules for reflection
- Use ServiceLoader
- The provides and uses keywords
- Exercise: Modules and the ServiceLoader
- Exercise: Using Reflection on modules

6 - WORKING WITH MODULES

- Being backwards compatible

- The ModulePath and ClassPath
- Unnamed Modules
- Automatic Modules
- The JLink tool
- Exercise: Migrating to modules

7 - JSHELL

- Introduction to JShell
- Running Expressions in JShell
- Importing packages
- Defining methods and types
- Using the JShell editor
- Save and loading state
- Exercise: Working with JShell

8 - OTHER NEW JAVA 9 FEATURES

- Enhancements on the Optional class
- Improvements made in the Process API
- The Stack-Walking API
- The HTTP2 Client
- The Multi-Resolution API
- Exercise: Working with Native processes
- Exercise: HTTP Clients

9 - PERFORMANCE OPTIMIZATIONS

- Performance in Java 9
- Compact Strings
- String deduplication
- Ahead-Of-Time Compilation
- Hotspot Diagnostic commands
- The G1 Garbage collector
- Variable and Method Handles

10 - MULTITHREADING

- Principles of Multithreading
- Creating a Threaded Class
- Basic Features of the Thread Class
- Thread Scheduling
- Thread Synchronization
- Exercise: Simple Thread Class
- Exercise: Simple Runnable Class

11 - CONCURRENT JAVA

- Concurrent Locks are Explicit and Flexible
- Executor Interfaces Provide Thread Management
- Challenges for Concurrent Use of Collections
- Concurrent Collections
- Atomic Variables Avoid Synchronization
- Exercise: Working with Concurrent Java
- Exercise: Sleeping Threads
- Exercise: Safe Data Access
- Exercise: Producer/Consumer

12 - JAVA 8 CONCURRENCY UPDATES

- The common thread pool
- Atomic variables
- LongAdder and LongAccumulator
- CompletableFuture
- Non-blocking asynchronous tasks
- Exercise: CompletableFuture

13 - INTROSPECTION AND REFLECTION

- Reflection classes
- Introspection
- Dynamic invocation of methods
- Using annotations
- Type annotations
- Receiver parameter
- Exercise: Introspection and Reflection
- Exercise: Reflection Server

14 - REFERENCE OBJECTS

- List the kinds of object references available in Java
- Introduce Weak, Soft and PhantomReference
- Explain the ReferenceQueue

15 - OBJECTS, DECLARATIONS, AND INITIALIZATIONS

- Abstraction and Responsibilities
- Low Coupling
- Programming principles
- Inheritance

16 - EXCEPTIONS

- Proper use of Exceptions
- Managing state in exceptional situations

- Checked vs. Unchecked Exceptions

17 - PROFILING AND BENCHMARKING

- List and describe the two types of benchmarks
- Describe the criteria that should be considered when constructing a benchmark plan
- Name the three most useful targets for profiling
- List four common tools/techniques for profiling
- Describe two strategies for improving performance as a result of profiling data
- List and explain the five most common problem areas for good performance with Java

18 - PROFILING TOOLS

- Use the JDK to collect runtime profiling data
- Successfully read the profiling data generated by the JDK to detect performance bottlenecks
- Instrument your own code to collect method execution time data
- Exercise: Using the JVM Profiling Tools and Visual VM

19 - CODE OPTIMIZATION TECHNIQUES

- List three potential problems with strings
- List two ways to optimize loops
- Describe the advantages of private and final methods
- List two advantages of collections over vectors and hashtables
- List 4 other code and system optimizations
- Exercise: Code Optimizations

20 - CODE OPTIMIZATION MYTHS

- Debunk several myths of Java performance tuning
- Synchronization trade-offs
- Setting methods to be final
- String is not always bad
- Revisit the fundamentals of Java code performance
- How to detect a performance myth
-

21 - DESIGN OPTIMIZATION TECHNIQUES

- List five ways to optimize Java program design
- Exercise: Design Optimizations