Advanced Java 9

Course Overview

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.

This is a 4-day class

Who Should Attend

Programmers with prior Java 8 or 9 programming experience

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.

Course Outline

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

Advanced Java 9

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

Advanced Java 9

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