Extreme Java - Concurrency Performance for Java 8

Duration 3 day(s) (XJ-CONC-J8-03)

Learn and apply new essentials on core Java topics

Official Training

Zenika exclusivity

Description

The "Extreme Java - Concurrency Performance for Java 8" course is the most intense learning experience you will ever get. It is aimed at the busy Java professional who would like to learn essential information about core Java topics. Each of the sections has been thoroughly researched by the author, Dr Heinz Kabutz, famous in 135 countries for his invention of "The Java Specialists' Newsletter". Throughout the course we use the new Java 8 syntax for lambdas and streams, in order to make the code more readable. Not only will you learn about threading, performance, compare-and-swap non-blocking constructs, garbage collectors and so many other topics that you can apply in your work, but at the same time you will get familiar with Java 8. In addition, we also cover all the relevant constructs that we find in Java 8, such as StampedLock, LongAdder, parallel streams and many others. In our outline below, we show you all that we will cover during our training. You will get an opportunity to try out what you learn with carefully thought out exercises. These will help you to really understand what is going on.

Goals

- Learn how to truly understand Java concurrency

Public

- Developer

Prerequisites

- Preferably a formal qualification in computer science or related field
- At least two years of professional Java programming

Structure

40% Theory, 60% Practice

Program

Introduction

- Welcome To The Course
  - How we deal with questions
  - Exercises with partial solutions
  - Certificate of Training
- History of concurrency
  - New supercomputers
  - Moore's Law
  - Hardware impact of concurrency
- Benefits of threads
  - Programming is easier
  - Better throughput
  - Simpler modeling
- Risks of threads
  - Safety vs liveness
  - Safety hazards
  - Using basic synchronized
  - Caching of fields
  - Code reordering
  - Annotations for Concurrency
  - Class annotations
  - Field annotations
- Threads are everywhere
  - Threads created by JVM
  - Threads created by frameworks
  - Timer
  - Servlets and JavaServer Pages
  - Remote Method Invocation (RMI)
- Short Java 7 and 8 Primer
  - Underscores in integral literals
  - Generic type inference
  - Lambdas
  - Method References
  - Streams
  - Primitive Streams

Thread Safety

- Introduction to Thread Safety
  - Synchronization and shared data
  - Your program has latent defects
- Atomicity
  - Byte code generated by simple count++
  - Demonstration of broken servlet
  - Compound actions
  - Check-then-act
  - Read-write-modify
- Sharing Objects
  - Visibility
  - Synchronization and visibility
  - Reason why changes are not visible
  - Making fields visible with volatile
  - Volatile flushing
  - Thread confinement
  - Unshared objects are safe
  - Ad-hoc thread confinement
  - ThreadLocal
  - Stack confinement
  - Immutability
  - Immutable is always thread safe
  - Definition of immutable
  - Final fields
- Designing a thread-safe class
  - Encapsulation
  - Primitive vs object fields
  - Thread-safe counter with invariant
  - Post-conditions
  - Pre-condition
  - Waiting for pre-condition to become true

Building Blocks

- Synchronized collections
  - Old Java 1.0 thread-safe containers
  - Synchronized wrapper classes
  - Locking with compound actions
- Concurrent collections
  - Scalability
  - ConcurrentHashMap
  - Class annotations
  - Additional atomic operations
  - Java 8 ConcurrentHashMap
  - CopyOnWriteCollections
- Blocking queues and the producer-consumer pattern
  - How BlockingQueues work
  - Java implementations of BlockingQueue
  - ArrayBlockingQueue
  - Circular array lists
  - LinkedBlockingQueue
  - PriorityBlockingQueue
  - DelayQueue
  - SynchronousQueue
  - TransferQueue
  - Deques
  - ArrayDeque
  - LinkedBlockingDeque
  - ConcurrentLinkedDeque (Java 7)
  - Work stealing