Extreme Java Programming
BT123

40 hours

Course Outline:

When Java programmers wish to achieve deeper knowledge of the language, two options are available:

- deep dive into the java programing language
- deep dive into the java runtime

This course is a deep dive into the runtime of the java application – where developers learn how their code is materialized and executed on the target machine. This raises the level of understanding – how multi-threaded programming works, how GC is effected by the code, reflection, performance impact, profiling – and language patterns to be used for each topic.

Upon completion of the course, developers will have deeper understanding of the execution environment and will know what kind of Java programming language constructs to use.

Who should attend the course:

Java developers with moderate experience.

Prerequisites:

Java programming.

Course Contents:

Module 1 - Multi-threading programming

- Parallel execution vs. concurrent execution
- The problems set of multi-threading programming
- Even more problems with the introduction of
  - Multi core CPU
  - L1 L2 L3 caches
- Cross process, Cross node.

Module 2 - Threads Synchronization

- The problem: when threads operate on class members of the same class
- The standard solution: synchronization
  - Types of synchronizations
    - internal on this, internal on object, external
- The modern soliton: explicit Locks
  - Standard lock, read lock, write lock, XXX

Module 3 - Threads coordination

- The problem: how to coordinate the execution of threads
- The must know: wait notify join
- Advanced thread coordination abstracts to the rescue
- Pitfalls to watch when programming threads coordination

Module 4 - Threads communication

- The problem: do threads see
- Java memory model: the (mostly) unknown factor
- Synchronized to the rescue
- Or is it volatile to the rescue

Module 5 - JRE internals

- Where is my data? Heap VS. Off heap VS. stack
- Where is my code? Interpreted code, compiled code, just in time
- Is really my code that is running? runtime optimizations general discussion
- Where is my code really running?
  - Java stack VS. OS stack
  - multi core and core affinity

- How my code is organized @runtime?
  - class loaders, manipulating class loaders
  - How to package my code? Jars in wars VS. jars outside wars
  - class meta data && reflection
  - Java 9 modularization (hell or bless?)

Module 6 - Garbage Collection

- Basics: why to GC , how to GC – general approach
- Different strategies of garbage collections
- Performance considerations
- Writing garbage collection friendly code

Module 7 - Profiling the runtime

- Why should I profile?
- Which approach? Sampling VS profiling
- What support is there? Profiling support on the JRE over the Java versions
- Heap analysis
- Threads analysis
- Other aspects to check
- From Jconsole to jvisualvm to flight control

Module 8 - Spring

- Why is it that Spring VS JEE
- Dependency Injection as enforcement of code modularization
- The power of templates (jdbc, soap, rest, activeDirectory etc)
- The singleton concept, and the impact on multi-threading programming