SQLabs
המכללה הגבוהה להייטק

Advanced Java– 40 Hours

Overview

This course teaches advanced concepts in Java. It teaches the more advanced aspects of Java system. It covers important topics such as memory management, collections, reflection, JVM internals, optimization, class loaders, references, NIO, multi-threading and more. The course will also present the most common pitfalls associated with the JAVA frameworks - memory leaks, garbage collection issues, using Java in scenarios where it is not comfortable.

Intended Audience

- Experiences Java programmers who would like become experts in the Java landscape.
- Java operations people who would like to know how to tune the JVM or find issues production Java systems.

Prerequisites

- At least one year experience in programming Java.

At the end of the course

- Participants will gain a better understanding of
  - how the JVM works.
  - how to use containers better.
  - how garbage collection works
  - the range of tools at their disposal when programming Java in order to protect themselves againt multi-threaded race conditions.
  - the Java memory model
  - the Java compile time and run time optimzer

Outline

- Containers
  - Available data structures and their performance tradeoffs
  - Using generics with data structures
    - Variance
    - Wildcards
    - Erasure
    - Runג€• time safe collections
  - Best practices
    - Iterators
    - Comparable/Comparator
    - The Collections class
    - Synchronized collections
  - Specific collections
    - ConcurrentHashMap
    - CopyOnWrite collections
    - Queue and Deque
    - ConcurrentSkipList
  - Primitive type collections as alternative
  - Issues
    - Modification during iteration
    - incorrect implementation of equals()
    - uncontrolled access to Map keys
  - Connection to Garbage collection

- Java Serialization
  - What is serializtion?
  - Backward compatibilty
  - Class versions
  - Non serializable base classes
  - Replacing serialized objects

- Java Multi Threading
  - Why threads?
  - Parallelism vs Concurrency
  - Theading concepts
  - Thread design
    - inheritance
    - Runnable implementation
  - Basic problem
    - stopping a thread
    - suspending a thread
    - thread communication
    - thread pooling
    - thread safe classes
    - daemon threads
    - finallization hooks
    - thread local
  - JVM and threading

        - threads in the JVM
        - Kernel threads vs user threads
        - performance issues
        - Exceptions from threads
        - affinity
        - Java memory model (volatile and other optimizations)
        - Double checked locking
        - Nested monitor lockout
    - Advanced synchronization mechanisms
        - Semaphores
        - Thread pools
        - Blocking queues
        - Timers
        - Futures
        - Read-Write locks
        - Atomic variables
        - Conditions and Synchronizers

- Java New I/O, New I/O2 [NIO, NIO.2]
    - Buffers
    - Channels
    - Selectors
    - Charsets

- Formatting in Java
    - How formatting works?
    - advanced features

- Java memory management
    - How garbage collection works?
    - Specifications and algorithms
    - Debugging garbade collection events
    - How to abuse the JVM and detect it.
    - What is a Java memory leak and how to find it?
    - Tuning the garbage collector.
    - Using object pools effectively to avoid mm issues.
    - Using native object containers to avoid mm issues.

- References
    - Why references?
    - Weak references
    - Soft references
    - Phantom references
    - Alternatives to references

- Reflection and Class loading
  - Dynamic proxy classes
  - Annotations, reflective access and compileɹ€• time processing
  - The classɹ€• loading mechanism
  - Classɹ€• loaders hierarchy, defining custom class loaders
  - Classes and Garbage Collection
  - Java class format, verification, class manipulation
  - Instrumenting Java classes at loadɹ€• time
  - Invoking the compiler from Java program

- Optimization
  - Low / High level Java Optimization issues
  - Compiler optimizations, runtime optimization and JIT compilers
  - Java problematic areas
  - Optimization tools

- Functional programming
  - Callback Functions
  - Async Programming
  - Invoke Dynamic
  - Predicate, Function, Consumer, Supplier
  - Default methods for interfaces
  - Using LAMBDA expressions
  - Using method references
  - Stream API
  - Parallel streams, common pool and dedicated pools

- Extra chapters if time permits
  - Java Internationalization
  - JVM Internals
  - JNI

מבין לקוחותינו: