

## **Java EE Memory Management Performance**

### **Course Summary**

#### **Description**

This course is designed to introduce students to the details of how the Java virtual machine manages memory. The course will provide the students with in depth knowledge that will make troubleshooting and optimizing Java memory usage a breeze. The course is a mix of theory and hands on experimentation with the JVM designed to provide students with the skills to effectively optimize J2EE Applications. This course is designed to be taught in a virtual classroom.

#### **Topics**

- Low level Java
- Java Reference types (Strong, Soft, Weak, & Phantom)
- Garbage collection Algorithms
- Garbage collection implementations in different JVM's
- Profiling Java & J2EE Memory Usage
- Finding Memory Leaks
- Tuning GC Garbage collection
- Caching the wrong way
- Caching the right way
- Distributed caching
- Optimizing J2EE applications for memory usage

#### **Audience**

This course is designed for programmers & Architects who need to gain a deep understanding of the Java virtual machine and how it manages memory so that they can build highly scalable, distributed and reliable applications.

#### **Prerequisites**

Students should have experience with Java programming.

#### **Duration**

Three days

## Java EE Memory Management Performance

### Course Outline

- I. Low Level Java**
  - A. Overview of low level Java
  - B. JVM Class Loading Architecture
  - C. Reflection
  - D. Byte code enhancement
  - E. Thread local variables
  - F. Garbage collection
- II. Reference Types**
  - A. What are strong references?
  - B. What are weak references?
  - C. What are soft references?
  - D. What are phantom references?
  - E. Applications of weak, soft, and phantom references.
- III. Garbage Collection Algorithms**
  - A. History of Garbage Collection
  - B. Mark and Sweep
  - C. Memory fragmentation & Compaction
  - D. Tracing algorithms
  - E. Copying algorithms
  - F. Generational algorithms
  - G. Comparison of the various garbage collection algorithm approaches
  - H. Garbage Collection and performance
- IV. JVM GC Algorithms**
  - A. Over JVM GC implementations
  - B. Sun JVM GC implantations
  - C. IBM JVM GC implementations
  - D. Collecting data about the GC
  - E. Interpreting output from the GC
- V. Profiling Java Application Memory Usage**
  - A. Profiling with the Eclipse TPTP
- VI. Finding Memory Leaks**
  - A. How Java Memory Leaks Occur
  - B. How to use profiling techniques to find memory leaks
  - C. How to interpret GC collection output graphs to detect memory leaks.
- VII. Caching**
  - A. Leaking memory by implementing caches the wrong way
  - B. Caching worst practices
  - C. Caching best practices
  - D. Distributed Caching solutions
- VIII. Optimizing J2EE Memory Usage**
  - A. Optimization Process
  - B. GC Tuning
  - C. Sizing memory requirements fro a J2EE server
- IX. Integrated Case Study**
  - A. Hands on scenario for the students to apply their newly learned skills in an integrated fashion.