

# Ultimate Java



## Abstract

Although Java is considered a relatively easy to use language, it has many sophisticated mechanisms and delicate points that are in many cases not fully utilized or even known to developers. A proper use of these mechanisms and "under the hood" structures greatly enhances code optimization and fine tuning.

This ultimate course focuses on these fine points and internal mechanisms and provides the "behind the scene" understanding of core Java libraries and the JVM internals. The knowledge gained in the course will significantly improve developers' ability to write more efficient and robust code.

The course includes many examples and hands-on exercises through which the material is demonstrated and practiced. The course is based on Java6 and provides a peek at Java7's new features.

## Target Audience

JavaSE/EE developers, team leaders and Architects.

## Prerequisites

2 years experience in Java Programming.

## Content:

### **Multi Threading and the JMM (4 hours):**

- Introduction.
- The Java Memory Model.
- Core Java threading functionality.
- Advanced Synchronization Mechanisms (`java.util.concurrent`).
- Best Practices in concurrent programming.
- A peek at Java7 fork/join library.

### **Garbage Collection (4 hours):**

- Introduction.
- The GC Anatomy and Algorithms.
- Monitoring the GC.



Reference Objects.  
HotSpot Command line flags.  
A peek at Java7 G1 algorithm.

### **Java Collections (4 hours)**

The core Data Structures (List, Set, Map).  
Understanding generics in and out.  
The util-concurrent Copy-on-Write collections.  
Queues, Dequeues and their Blocking versions.  
Overview of the Apache commons-collections framework.

### **Java IO/NIO (4 hours)**

Advanced Serialization concepts.  
Buffers.  
Channels.  
Non blocking IO.  
NIO design patterns.  
Overview of Apache Mina and Grizzly.  
A peek at Java7 NIO2.

### **Performance & Monitoring (8 hours):**

The JIT compiler and the HotSpot JVM.  
String Handling.  
Exceptions.  
JDBC Tracing.  
Avoiding synchronization using Atomic Classes.  
Performance Pitfalls.  
Profiling using the built-in VisualVM.  
Heap walking using JHat.  
Monitoring using JConsole.  
Agents and the Attach API.

**Duration:**        **3 days**