

This intensive, hands-on course explores advanced Java™ 5.0 Standard Edition language features and packages. Students will learn to parse XML documents using the JAXP API. Multi-threaded applications will be covered in detail including concepts such as deadlocks and race conditions. Students will also learn how to utilize more advanced I/O capabilities with object serialization and low-level file I/O with the java.nio package. Client/server applications will be written utilizing both the java.net and java.rmi packages. Additional topics on JNI, performance tuning, and advanced RMI are included as appendices for further study.

Audience: Java programmers who wish to increase their depth of knowledge in Java programming and explore the uses of the various advanced packages.

Prerequisites: *Intermediate Java Programming* or equivalent experience is required.

Number of Days: 3 days

- 1. Processing XML with Java – JAXP**
 - The Java API for XML Processing
 - Introduction to SAX Parsing
 - SAXParser and JAXP
 - SAX Event Methods
 - Introduction to DOM
 - Parsing DOM with JAXP
 - The DOM API
 - Validation
 - Transformation
- 2. Introduction to Threads**
 - Non-Threaded Applications
 - Threaded Applications
 - Creating Threads
 - Thread States
 - Runnable Threads
 - Coordinating Threads
 - Interrupting Threads
 - Runnable Interface
 - ThreadGroups
- 3. Thread Synchronization and Concurrency**
 - Race Conditions
 - Synchronized Methods
 - Deadlocks
 - Synchronized Blocks
 - Thread Communication — wait()
 - Thread Communication — notify()
 - Java 5.0 Concurrency Improvements
- 4. Advanced I/O - Object Serialization**
 - Thread-Aware Collections
 - Executor
 - Callable
 - What is Serialization?
 - Serializable Objects
 - Writing an Object
 - Reading an Object
 - Handling Exceptions
 - Customizing Serialization
 - Controlling Serialization
 - Versioning
- 5. Advanced I/O – New I/O**
 - The java.nio package
 - Buffers and Channels
 - Buffer Implementations
 - Buffer Methods
 - ByteBuffer Methods
 - FileChannel
 - File Locking
 - MappedByteBuffer
 - Transferring Data between Channels
 - Character Sets
- 6. Reflection**
 - Introduction to Reflection
 - The Class Class
 - The reflect Package
 - Constructors
 - Fields

- Methods
- Exception Handling and Reflection
- JavaBeans
- Dynamic Programming
- 7. Networking with Sockets**
 - Clients and Servers
 - Ports, Addresses and Protocols
 - The Socket Class
 - Communication Using I/O Servers
 - The ServerSocket Class
 - Concurrent Servers
 - The URL Class
 - The URLConnection Class
- 8. Remote Method Invocation**
 - Distributed Applications
 - Stubs
 - Steps to Create a Remote Object
 - An RMI Client
 - An RMI Server
 - RMI Classes and Interfaces
 - Class Distribution
 - RMI Utilities
 - Parameter Passing and Serialization
- 9. Java Naming and Directory Interface (JNDI)**
 - Naming and Directory Services
 - Namespaces and Contexts
 - Naming Operations
 - Bindings
 - Attributes
 - Directory Operations
 - DNS Lookups with JNDI
 - JNDI in J2EE
- 10. Java Performance Tuning**
 - Is Java Slow?
 - Don't Optimize Until You Profile
 - HotSpot Virtual Machine
 - Garbage Collection Concepts
 - Garbage Collection Generations
 - Garbage Collection in Java 5.0
 - Object Creation
 - String, StringBuffer, and StringBuilder
 - Synchronized
 - Inline methods
 - Tuning Collections

- 11. Appendix A - Advanced RMI**
 - Client Callbacks
 - Dynamic Class Loading
 - Activation
 - Activatable Objects
 - Registering Activatable Objects
 - Security and Activation
 - JNDI and RMI Registry
 - RMI-IIOP
- 12. Appendix B - Native Methods**
 - Overview of Java Native Methods and JNI
 - How to Create and Use Native Methods
 - Native Method Declaration
 - Using javah
 - Creating the Implementation Code
 - Compilation
 - Distribution
 - Using the Native Methods
 - JNI
 - Passing Arguments
 - Calling Java Methods in Native Code
 - JNI Signatures