JSEADV - Advanced Java SE Programming

Overview:
This course teaches how to develop more advanced Java SE applications using Eclipse and also explores Java SE topics that provide the finishing touches when building professional Java applications using Eclipse. Students will be introduced to Java Beans and Java Database Connectivity (JDBC). Students will learn about internationalising their Java applications, using Java reflection, multi-tier architecture, using threads for multitasking, and networking. In addition, students will learn about Parsing XML documents using both SAX and DOM methodologies. This course also covers how to take advantage of distributed objects, add security, use the Java Native Interface (JNI) and utilise Test-Driven Design through JUnit.

Target Audience:
This course is intended for programmers who are familiar with Java and want to learn about the advanced features of Java.

Pre-requisites:
This course is intended for students who have attended our JSEINT - Introduction to Java SE Programming course or have equivalent Java SE experience (6-12 months professional coding experience).

Module 1: JavaBeans
- JavaBeans
- Overview
- Structure
- Life Cycle
- Properties
- Making a Bean

Module 2: JDBC
- Java Database Connectivity
- Types of Drivers
- Making a Connection
- Statements
- Execute, Update and ResultSets
- Prepared Statements
- Stored Procedures
- Transactions
- Connection Pooling

Module 3: Internationalisation
- Internationalisation
- Locale
- User Interface design
- Resource Bundles
- Other Local Customs
- How Java Represents Characters
- Text Files

Module 4: The Reflection API
- Dynamic Facilities In Java
- What this dynamic aspect implies
- Considerations
  - The Class class
  - Examining Classes
  - Reflection Allows
  - Reflection Also Allows
  - When not to use Reflection
- Examining Classes
- Manipulating Objects
- Creating Objects
- Getting Field Values
- Setting Field Values
- Arrays

Module 5: Architecture
- Architecture
- Tiered Architectures
- Presentation Layer
- Middle Tier
- Model View Controller
- Extensible Markup Language
- XML

Module 6: Threads
- Threads
- Overview of Threads
- Threads in Java Programming
- Write a Runnable Class
- Create Threads
- Another Way of Creating Threads
- Two Ways of Creating Threads
- States in a Thread s Lifetime
- JVM Scheduler
- Control and Schedule Thread
- Executor Interface
- Using the Executor
- Callable
- ExecutorService Object
- Future Object
- Executor, Future, Callable
- Coordinating the Concurrency of Multiple threads
- Synchronisation
- How Does the Object Lock Flag Work
- Using the synchronised keyword
- The implication of synchronisation
- Coordinating Thread Cooperation
- wait() and notify()
- Results
- Deadlock

Module 7: Networking
- Java Networking
- URL Connections
- InetAddress
- Socket Classes
- Simple Clients and Servers
- Multithreaded Servers
- UDP Sockets

Module 8: Parsing XML with SAX
- Obtaining a Parser SAX
- How it Works
- Core SAX2 Handler Classes
- SAX2 DefaultHandler
- SAX Events
- Ignorable Whitespace
- XML Reader Interface
- XMLReader Features
- XMLReader Factory
- Prepare SAX Parser Object
- Parse XML with SAX Steps
- Define an Event Handler
- Prepare SAX Parser Object
- Define an Event Handler
- Set a SAX Handler
- Parse XML Document
- Parse XML Factory
- Define an Event Handler
- Define an Event Handler
- Using characters()
- Define an Event Handler ErrorHandler interface
- Parse XML Document
- Simple SAX Parser
- Run the SAX Application
- EntityResolver
- Locator
- Document Locator

Duration: 5 days
Module 9: Parsing XML with DOM
- DOM
- Limitations of SAX
- XML as an Object Model
- Nodes
- The Basic Node Types
- Less Common Node Types
- Node Interface
- Document Interface
- NodeList Interface
- Element Interface
- Attr Interface
- Text Interface
- DOM Parsing
- Parse XML with DOM Steps
- Prepare DOM Parser Object
- Parse XML Document
- Parse Exceptions
- Writing DOM

Module 10: Distributed Objects
- Serialisation
- Externalisable
- Remote Method Invocation
- Steps to implement RMI
- RMI Remote Object Registry
- RMI Dynamic class loading
- RMI and Applets

Module 11: Security
- Java Security
- Attacks and Dangers
- Overview of JDK Security Features
- Basic Concepts of Computer Security
- Encryption
- Using the MessageDigest Class
- Using the Signature Class
- Java Security Architecture
- JDK 1.0 Security Model Sandbox
- JDK 1.1 Security Model Trusted Signed Code
- JDK 1.2 Security Model Security Policy
- JDK 1.4 Security Enhancement
- Protection Domains and Security Policies
- ProtectionDomain Class
- Permission Classes
- Using Permission Classes
- Policy Class
- Policy Configuration File
- AccessController Class
- SecurityManager Class
- Using the SecurityManager Class
- Dynamic Class Loader
- Loader Classes
- Java Security Tools
- Using Java Security Tools Code Signing

Module 12: JNI
- JNI - Java Native Interface
- JNI Architecture
- Calling C Functions
- The Header File
- Passing Simple Parameters
- Mapping Java Types
- Calling Java Methods
- The Invocation API
- Exception Handling
- Native Exception to Java
- Java Exception In Native Code

Module 13: JUnit
- What is JUnit?
- Who uses JUnit?
- Why JUnit?
- The xUnit Philosophy
- Test-Driven Design
- A JUnit Test
- Running the Tests
- Swing-based Test Runner
- Text-based Test Runner
- JUnit Basics
- assertTrue, assertEquals
- assertSame, assertNotSame
- The Failure Message
- The Test Class
- The Test Method
- The Test Suite
- JUnit Design
- Testing Strategies
- Specific Techniques
- Testing simple Java classes
- Testing with Databases
- Testing Web Applications
- JUnit with Ant
- JUnit with Eclipse
- Create a Test Case
- Test Case “Stubs”
- Running Tests
- Eclipse Test Runner Icons
- Rerun an Individual Test
- Failure Trace
- Debug with JUnit
- Test Suite Wizard