

# Java 5 Programming For Experienced Developers

Duration:	5 days
Type:	intermediate

## Description

This is an intensive course designed to introduce experienced developers to the Java language. The course is designed for developers who will be moving to JEE development but can also be used as a refresher for delegates who already have some exposure to Java programming.

## Prerequisites

Delegates should have a minimum of two years programming experience, preferably in an object oriented environment.

## List of Modules

### Core Concepts

- Origins and goals of the Java language
- Bytecode and the Java Virtual Machine
- The classpath environment variable
- Packages and dynamic class loading
- Garbage collection in Java

### Basic Java Programming

- Primitive types and literal values
- Understanding Java references
- Converting numerical types
- String parsing with the *Scanner* class
- Pretty-printing with the *Formatter* class
- Iteration and selection in Java
- Equality of primitive and reference types
- Creating and manipulating arrays
- The costs of String concatenation

### Object Oriented Development Part 1

- Creating basic Java classes
- Choosing accessibility levels
- Overloading and overriding methods
- Overriding the *toString* method
- Comparing references using the *instanceof* operator
- Comparing references using *java.lang.Class* objects
- Inheriting from a base class
- Creating abstract and final classes

## Object Oriented Development Part 2

- Writing appropriate class constructors
- Private constructors and singletons
- Static and instance initialization blocks
- Top down class and object initialization
- Declaring and using interfaces
- Using inner and anonymous classes
- Implementing *equal* and *hashCode*
- Cloning and copy constructors

## Enumerations

- Why Java historically lacked enum support
- The typesafe enumeration design pattern
- Declaring enumerations in Java 5
- The link between enums and classes
- The *java.lang.Enum* base class
- Extending enums with new members
- Static methods added to enum types
- New collections which support enums

## Annotations

- Adding metadata to Java code
- The advantages of annotations
- Annotations verses configuration files
- Declaring Java 5 annotation types
- Understanding meta-annotations
- Adding methods to annotation types
- Defining default values for methods
- Discovering annotations with reflection

## Exception Handling in Java

- Introducing errors, runtime exceptions and checked exceptions
- Exceptions in constructors and finalizers
- Implementing an effective exception handling strategy
- The correct use of finally blocks
- Using Java 1.4 assertions

## The Collections API

- Introducing lists, sets and maps
- Using iterators and enumerations
- Choosing the right collection class
- Making collection objects thread safe
- Making collections objects read only
- Performance issues with collections
- Built in algorithms for collection objects
- Comparing objects in collections

## Generics in Java Part 1

- The evolution of Generics in Java
- Generics as a compile time construct
- Generics verses templates in C++
- Comparing Generics and inheritance
- Working with generic collection classes
- Introducing the *java.lang.Class* type
- Support for Generics in the Reflection API

## Generics in Java Part 2

- Declaring generic classes and methods
- Using the wildcard type in utility methods
- Defining constraints with bounded wildcards
- Adding generics to existing Java code

## Building Multi-Threaded Applications

- The Java memory model
- Creating threads by subclassing
- Creating threads via *Runnable*
- Safely interrupting another thread
- Synchronized methods and blocks
- Creating and avoiding deadlock
- Dangers when synchronizing methods
- Creating threads using *Executor* objects in Java 5
- Storing return values in *Future* objects in Java 5

## Java I/O

- Introducing input and output streams
- Introducing readers and writers
- Using *File* objects to resolve file paths
- Text based file I/O with buffered streams
- Binary based file I/O using data streams
- Serializing Java objects
- Customizing Java Serialization

## Networking in Java

- Connecting to servers via *Socket* objects
- Creating server loops with *ServerSocket*
- Building a simple HTTP web server
- Load balancing using worker threads
- Load balancing using asynchronous I/O