

## Java Programming: A Comprehensive Hands-On Introduction - 4 Days

### *Course 471 Overview*

#### **You Will Learn How To**

- Write, compile and execute Java programs
- Build robust applications using Java's object-oriented features
- Create robust applications using Java class libraries
- Develop platform-independent GUIs
- Read and write data using Java streams
- Retrieve data from a relational database with JDBC

#### **Course Benefits**

Java's unique architecture enables programmers to develop a single application that can run across multiple platforms seamlessly and reliably. In this hands-on course, you gain extensive experience with Java and its object-oriented features. You learn to create robust console and GUI applications and store and retrieve data from relational databases.

#### **Who Should Attend**

Anyone developing Java applications. Previous experience with a programming language such as C, JavaScript, PHP or COBOL is assumed. Familiarity with Web technologies and object concepts is helpful.

#### **Hands-On Training**

A series of hands-on exercises provides experience creating Java applications. Through an ongoing case study, you design and build an intricate desktop application modeled on a well-known Web site. Exercises include:

- Developing an object-oriented model with UML notation
- Creating Java objects and calling their methods
- Structuring data with the Java collections API
- Creating portable GUIs with Swing components
- Adding event handling to GUIs
- Retrieving data from a relational database with JDBC

# Java Programming: A Comprehensive Hands-On Introduction - 4 Days

## Course 471 Outline

### Introduction to Java Programming

#### Advantages of Java

- Platform independence
- Stand-alone applications and servlets

#### Structure of a Java program

- Compiling source code into bytecode
- Overview of class libraries

### Object-Oriented Programming with Java

#### The object paradigm

- Object-oriented (OO) programming
- Encapsulation, inheritance and polymorphism
- OO analysis and design: "Is a" and "Has a" relationships
- Designing an OO application step by step
- Diagramming object structure with Unified Modeling Language (UML)

#### Java's object-oriented features

- Instantiating objects from classes
- Aggregation and composition
- Extending existing classes
- Overloading and overriding methods

### Structure of the Java Language

#### Language syntax

- Declaring and initializing variables
- Statements and expressions
- Declaring and using arrays
- Upcasting, downcasting and autoboxing

#### Flow control

- Invoking methods and passing parameters
- Conditionals and loops
- Handling exceptions with **try** and **catch**

#### Defining classes

- Fields (instance data)
- Methods (functions)
- Abstract classes and interfaces
- Organizing classes with packages and visibility modifiers
- Composition vs. inheritance

### Building the components of a Java program

- Working with existing classes
- Leveraging generics with the collections API
- Extending base classes
- Developing new classes

- Compiling and debugging

- Java Integrated Development Environments (IDEs)

### Developing GUIs

#### Foundations of user interfaces

- Basic GUI widgets
- Event-driven programming
- Benefits of a portable windowing library

#### Java Foundation Classes (JFC)

- Advantages of lightweight Swing components
- Exploring the Swing component library
- Creating Swing components: buttons, text fields, drop-down lists
- Adding Swing components to containers
- Arranging Swing components using layout managers
- Dialogs and message boxes

#### Event handling

- Registering event handlers
- Inner classes and top-level classes

### Storing and Retrieving Data with File I/O

#### Java streams

- Streams, Readers and Writers
- Accessing files
- Catching and throwing exceptions
- Formatting text output

#### Files and directories

- Reading and writing files
- Creating, deleting and renaming files
- Obtaining directory and file information

### Working with Relational Databases

#### JDBC database access

- Leveraging the JDBC API
- Choosing database drivers
- Connecting to a database

#### Improving performance with prepared statements and stored procedures

- Submitting SQL statements
- Retrieving and processing results

### Java Development Tools

- Java Development Kit (JDK)
- Compiler (javac)
- Javadoc utility
- Java Archive (JAR) utility