

Java 6 Programming Fundamentals for Non-OO, Procedural (C, COBOL, 4GL) Programmers

Duration: 5 days

Description

An **introductory** comprehensive hands-on Java training course geared for developers who have little or no prior working knowledge of object-oriented programming languages (such as those working on (C, COBOL, 4GL, etc.) Throughout the course students learn the best practices for writing great object-oriented programs in Java 6, using sound development techniques, new improved features for better performance, and new capabilities for addressing rapid application development. Special emphasis is placed on object oriented concepts and best practices. In addition to the normal exercises that are liberally sprinkled throughout the course, there is a case study that covers the entire spectrum from use cases to object-oriented design to implemented classes. This case study supplements the course and can be used during and after the course as a reference and a tool for reviewing and practicing what was learned in class.

Audience

Experienced developers with little or no Object-Oriented background (C, COBOL, Mainframe, 4GL) who wish to get up and running with Java, or who need to reinforce sound Java coding practices, immediately.

Prerequisites

Attendees should have a working knowledge of developing software applications.

Topics

JAVA: A FIRST LOOK

Lesson: Using the JDK

- Setting Up Environment
- The Development Process
- Locating Class Files
- Compiling Package Classes
- Source and Class Files
- Applications and Applets

Lesson: Writing a Simple Class

- Classes in Java™
- What Is a Class?
- Defining the Class
- Class Modifiers
- Class Instance Fields

- Instance Fields Diagram
- Primitives vs. Object References
- Creating Objects
- The main Method
- Using the Dot Operator
- Writing Output

Lesson: The Java™ Platform

- Defining Java
- Java Provides Several Platforms
- Note on Terminology
- Java SE 6
- Java SE Development Kit (JDK)
- Executing Programs
- Lifecycle of a Java Program
- Responsibilities of JVM

Java 6 Programming Fundamentals for Non-OO, Procedural (C, COBOL, 4GL) Programmers

- Java is Dynamic: The Runtime Process
- Primary Areas of the JVM Runtime
- Garbage Collection
- Documentation and Code Reuse
- JavaDoc Provides Documentation Delivery
- In-Line Comments are Translated into HTML Rendering
- Working with Java in Your Environment
- Type Abstraction – Grouping as Supertype
- Polymorphism
- Polymorphism Diagram

INTRODUCTION TO OO CONCEPTS

Lesson: OO Programming

- The Object Oriented Way
- Real-World Objects
- Classes and Objects
- Examples of Classes and Objects
- Classes and Objects Diagram
- Object Behavior
- Methods and Messages

Lesson: Classes and Objects

- Concepts, Entities, Classes & Objects
- Classes
- Responsibilities and Operations
- Abstractions and Responsibilities
- Instantiation
- Instances
- Objects Provide a Service
- Messages and Objects
- Encapsulation
- Layered Architecture

Lesson: Inheritance, Abstraction, and Polymorphism

- Encapsulation
- Inheritance
- Method Overriding
- Aggregation

INTRODUCTION TO OO ANALYSIS AND DESIGN

Lesson: Introduction to Modeling, UML, and Design

- Software Design
- What is OO?
- The Goal of OOD
- Benefits of OO
- Three Object-Oriented Themes
- Building Models
- Why Build a Model?
- Notation
- What is UML?
- Domains
- The Process of OO Analysis and Design
- OOAD Process: Requirements Capture
- OOAD Process: Analysis
- OOAD Process: Domain Design
- OOAD Process: Detailed Design
- OOAD Process: Architectural Design

Lesson: UML

- Class Diagrams
- Terms and Concepts
- Class Diagram Compartments
- Adding Attributes to the Class Notation
- Adding Methods to the Class Notation
- Visibility
- Instance and Static Scope
- Class Diagrams with Package Names
- Stereotypes
- UML Class and Instance Icons

Java 6 Programming Fundamentals for Non-OO, Procedural (C, COBOL, 4GL) Programmers

- Drawing Dependencies
- Associations
- Associations Can Be Bi-directional
- Navigability
- Multiple Associations Between Two Classes
- Whole/Part Associations
- Composition
- Generalization/Specialization Relationships
- Scanner - File Source
- Scanner - Getting Input
- Scanner - Testing for Tokens
- Scanner - Patterns for Tokens
- Formatter
- Formatter – Probable First Encounters
- Formatter – StringBuffer

GETTING STARTED WITH JAVA

Lesson: Adding Methods to the Class

- Instance Methods
- Passing Parameters Into Methods
- Returning a Value From a Method
- Overloaded Methods
- Overloaded Methods Diagram
- Constructors
- Defining a Constructor
- Optimizing Constructor Usage

Lesson: Language Statements

- Operators
- Comparison and Logical Operators
- Looping: The for Statement
- Looping: The while Statement
- Looping: The do Statement
- Continue and Break Statements
- The switch Statement

Lesson: Using Strings

- Strings
- String Method
- String Equality
- StringBuffer
- Strings, StringBuffer, and StringBuilder
- StringTokenizer
- Scanner

Lesson: Specializing in a Subclass

- Extending a Class
- The extends Keyword
- Casting
- Overriding Superclass Methods
- Method Overriding Diagram
- Calling Superclass Methods from Subclass
- The Object Class
- The equals Method
- Default Constructor
- Implicit Constructor Chaining
- Passing Data Up Constructor Chain
- A Common Programming Mistake
- Editing Tools in Your IDE

ESSENTIAL JAVA PROGRAMMING

Lesson: Fields and Variables

- Fields vs. Variables
- Data Types
- Default Values
- Block Scoping Rules
- Using this
- Final and Static Fields
- Static Variable Diagram

Lesson: Using Arrays

- Arrays
- Accessing the Array
- Multidimensional Arrays

Java 6 Programming Fundamentals for Non-OO, Procedural (C, COBOL, 4GL) Programmers

Lesson: Static Methods and Fields

- Static Fields
- Simple Example of Static Fields
- Static Methods

Lesson: Java Packages

- The Problem
- Packages
- Class Location of Packages
- The Package Keyword
- Importing Classes
- Executing Programs
- Visibility
- Java Naming Conventions
- Packages Diagram
- Refactoring in Your IDE

ADVANCED JAVA PROGRAMMING

Lesson: Inheritance and Polymorphism

- Polymorphism
- Polymorphism: The Subclasses
- Treating Derived Classes as the Superclass
- Casting to the Derived Class
- Using instanceof For Downcasting
- Upcasting vs Downcasting
- Calling Superclass Methods From Subclass
- The final Keyword

Lesson: Interfaces and Abstract Classes

- Separating Capability from Implementation
- Abstract Classes
- Shape as an Abstract Class
- Polymorphism With Abstract Classes
- Interfaces
- Implementing an Interface
- Extending Interfaces
- Polymorphism With Interfaces

- Type Checking
- Abstract Classes vs. Interfaces
- Interfaces Diagram

Lesson: Exceptions

- What is an Exception?
- Exception Architecture
- Handling Exceptions
- The Throwable Class
- The try Block; The catch Block
- The finally Block
- Full Example of Exception Handling
- Generalized vs. Specialized Exceptions
- Overriding Methods
- Creating Your Own Exceptions
- Throwing Exceptions
- Re-throwing an Exception
- Checked vs. Unchecked Exceptions
- Debugging in Your IDE

JAVA DEVELOPER'S TOOLBOX

Lesson: Utility Classes

- Wrapper Classes
- The Number Class
- Numbers and Strings
- Big Decimal
- Random Numbers
- Decimal Formatting
- The Date Class

Lesson: Vector and Hashtable

- The Vector Class
- Creating and Using a Vector
- Java Collections Methods in Vector
- Hashtables
- Understanding How Hashing Works
- Creating and Using a Hashtable
- Performing Lookups

Java 6 Programming Fundamentals for Non-OO, Procedural (C, COBOL, 4GL) Programmers

Lesson: Collections

- The Collections Framework
- Collections Feature Types
- Collections Interface Hierarchy
- Map Interfaces
- Optional Methods
- The Collection Interface
- Iterators
- The Set Interface
- SortedSet
- Set and SortedSet Example
- Comparable and Comparator
- The List Interface
- List Example
- ListIterator
- Queue Interface; QueueExample
- BlockingQueue
- BlockingQueue Implementations
- Collections Utility Methods
- Features of the Implementation Classes
- Synchronization Wrappers
- Feature Comparison
- Using the Right Collection
- Use of Collections vs. Vector/Hashtable
- Optimizing Collection Constructors
- Copying Arrays
- Creating and Using an ArrayList
- Creating and Using a HashMap

Lesson: Generics

- Generics and Parametric Polymorphism
- Simple Generics
- The Mechanics of Generics
- Generics and Subtyping
- Compiler Restrictions on Generics and Subtyping
- Generics as Arguments in Methods

- Rationale Behind Wildcards
- Wildcards In Use
- Regular Wildcards in Method Parameters
- Bounded Wildcards
- Standard Rules Apply
- Generic Methods
- Interoperability with LegacyCode
- Raw Types
- Legacy Calls To Generics
- When Generics Should Be Used
- Build Paths in Your IDE

Lesson: Overview of Java GUIs

- JFC – Java Foundation Classes
- Categories of Classes in JFC
- Creating the Frame
- Adding Content to a Frame
- A Closer Look at Layout Managers
- BorderLayout
- JFC Provides an Event Handling Structure

Lesson: JDBC

- What is JDBC?
- Structured Query Language (SQL)
- Connecting to the Database
- Statements
- Statement and PreparedStatement
- ResultSet
- JDBC Diagram
- Executing Inserts, Updates, and Deletes
- Controlling Transactions and Concurrency
- Mapping SQL Types to Java Types
- Database Connection Via JDBC Calls
- Rationale for Connection Pooling
- Connection Pooling in JDBC
- Database Connection Using a DataSource

Java 6 Programming Fundamentals for Non-OO, Procedural (C, COBOL, 4GL) Programmers

- Stored Procedures Defined
- Callable Statement Syntax
- Stored Procedure Parameters
- RowSet Implementations
- JDBCRowSet
- JDBCRowSet Approach
- JDBCRowSet – Retrieving Data
- JDBCRowSet Example
- CachedRowSet
- CachedRowSet Approach
- CachedRowSet Example

Lesson: Java Logging

- Why Logging?
- Logging Framework
- Logging in Java
- Java Logging Framework
- The Logger Class
- Global Configuration
- Logging Levels
- Programmatically Setting Logging Properties
- Programmatic Handlers
- Formatters
- Logging Security & Performance