Intermediate/Advanced Java 8

Course Summary

Description

This is a fast-paced course suitable for developers with some previous experience in working with Java and Object-Oriented (OO) programming. The course can also be delivered to developers experienced in other OO languages (Python, C++, C#) but with limited Java exposure, as well as used as an advanced Java course for more experienced developers. There are numerous optional parts allowing you to tailor the coverage to your group.

It includes an accelerated, yet thorough hands-on review of Java foundational concepts, with attention given to OO design and implementation principles. It then moves on to comprehensive coverage of more advanced topics in Java and OO development to provide participants with a strong grounding to use Java in a sophisticated and productive manner.

This course covers far more than an introductory course, including important topics such as UML and Design Patterns, and using composition vs. inheritance, which are all key to creating well-structured OO systems. After these important areas, it moves on to advanced Java topics such as inner classes, reflection, writing generic classes, and functional programming with lambdas and streams. It teaches a number of useful techniques that enhance productivity and good system design - which may otherwise take Java developers years to absorb on their own. The course concludes with build tools and logging. Unit testing is stressed throughout the course, with most labs implemented as JUnit tests.

Although the course is designed to be delivered in its entirety, it can be customized to fit your needs. Be prepared to work hard and learn a great deal!

All labs are done with the Eclipse IDE Java EE version, and the lab instructions include detailed directions for using it.

Objectives

After taking this course, students will be able to:

- Solidify Java foundational knowledge, including the important contracts of class Object
- Understand the uses and consequences of inheritance and composition, and reinforce the role of interfaces
- Reinforce fundamental OO principles such as cohesion, coupling, and polymorphism
- Use the JUnit testing framework and become fluent in writing assertions to verify correct program behavior
- Familiarity with UML modeling in class diagrams and sequence diagrams
- Use advanced techniques for object creation, including factories and singletons
- Use established design patterns for object composition, including Strategy, Decorator, and Facade
- Write and use generic classes and methods
- Learn the use cases for inner classes and refactor existing code to use them when appropriate
- Create and use custom annotations
- Be familiar with reflection and how to use it
- Understand the role of functional interfaces introduced in Java 8
- Understand lambda expressions and method references, and use them to pass behavior (methods)
- Use the Stream API to perform complex processing of collections and other input sources
- Learn the new Date/Time API and use it to solve date-oriented business problems
- Use Javadoc to write professional-grade API comments for your classes
- Understand build tools such as Ant and Maven, and how to manage application dependencies
- Write appropriate log statements in your code and configure logging with Log4j
Intermediate/Advanced Java 8

Course Summary (cont’d)

Topics
- Review - Basics
- Review (Inheritance and Interfaces)
- JUnit
- Collections and Generics
- Techniques of Object Creation
- Using Composition and Inheritance Effectively
- Inner Classes
- Annotations
- Reflection
- Lambda Expressions
- Streams
- Date/Time API
- Overview of Build Tools (optional)
- Logging and Log4j (optional)

Audience

This is a fast-paced course suitable for developers with some previous experience in working with Java and Object-Oriented (OO) programming. The course can also be delivered to developers experienced in other OO languages (Python, C++, C#) but with limited Java exposure, as well as used as an advanced Java course for more experienced developers.

Prerequisites

Before taking this course, you should have some previous experience in working with Java and Object-Oriented (OO) programming or be experienced in other OO languages (Python, C++, C#) but with limited Java exposure.

Duration

Five days
Course Outline

I. Review - Basics
   A. Java Environment
   B. Classes and Objects
      1. Instance Variables, Methods, Constructors, Static Members
      2. OO Principles: Data Encapsulation, Cohesion
      3. Object Contracts: toString(), equals() and hashCode(), Comparable and Comparator
   C. Packages, Enums, Arrays
   D. Exceptions

II. Review (Inheritance and Interfaces)
   A. UML Overview
   B. Inheritance
      1. Definition and IS-A Relationship
      2. Method Overriding, @Override
      3. OO Principles: Principle of Substitutability, Polymorphism and Encapsulation of Type, Coupling, Open-Closed Principle
      4. Constructor Chaining
   C. Interfaces
      1. Defining and Implementing, Interface Types
      2. Interface Inheritance
   D. New Interface Features in Java 8
      1. Default Methods, Static Methods
      2. Functional Interfaces
   E. Guidelines

III. JUnit
   A. Overview
   B. Tests, Assertions, and Fixtures
      1. Writing and Running Tests
      2. Assertions
      3. Test Fixtures, @Before and @After, @BeforeClass and @AfterClass
      4. Testing for Exceptions
   C. Best Practices and Test-Driven Development Overview (TDD)

IV. Collections and Generics
   A. Collections Overview Generics and Type-Safe Collections
      1. Diamond Operator
   B. Lists, Sets, and Maps
      1. Interfaces and Contracts
      2. Iteration and Autoboxing
      3. Utility Classes - Collections and Arrays
   C. Writing Generic Classes
      1. Inheritance with Generic Types
      2. Wildcard Parameter Types
      3. Type Erasure

V. Techniques of Object Creation
   A. Design Patterns Overview
   B. Controlling Object Creation
      1. Limitations of new Operator, Alternative Techniques
   C. Singleton Pattern
   D. Simple Factory
   E. Factory Method Pattern
   F. Other Techniques Named Objects, JNDI
      1. Dependency Injection Frameworks

VI. Using Composition and Inheritance Effectively
   A. Inheritance and Composition - Pros and Cons
      1. Composition and Delegation
      2. HAS-A, USES Relationships
   B. Strategy Pattern
   C. Decorator Pattern
   D. Façade and Other Patterns
      1. Facade, Proxy, Template Method

VII. Inner Classes
   A. Overview and Motivation
      1. Stronger Encapsulation, Rules and Caveats
   B. Defining and Using Inner Classes
      1. Member-Level, Method-Local, Anonymous Classes
   C. Static Nested Classes
      1. Nested Classes, Nested Interfaces, Nested Enums

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically.
Intermediate/Advanced Java 8

Course Outline (cont’d)

VIII. Annotations
A. Overview
B. Using Annotations
   1. Target and Retention Policy
   2. Annotation Parameters, Parameter Shortcuts
C. Writing Custom Annotations
   1. Syntax, Using the Meta-Annotations
   2. Using a Custom Annotation

IX. Reflection
A. Overview and API
   1. The Class Called Class
   2. Obtaining and Inspecting Class Objects
B. Working with Objects Reflectively
   1. Creating Instances, Invoking Methods, Setting Field Values

X. Lambda Expressions
A. Overview
   1. Functional Interfaces and Lambdas
   2. Target Context
B. Using Lambda Expressions
   1. Syntax, Lambda Compatibility
   2. Variable Capture
   3. Type Inference
C. Method References
   1. Three Types of Method References
   2. Refactoring Lambdas into Method References

XI. Streams
A. Overview
   1. Streams vs. Collections
   2. Anatomy of a Stream
B. Understanding the Stream API
   1. Intermediate Operations and Stream Pipeline
   2. Java 8 Functional Interfaces: Predicate, Comparator, Function, Consumer, Supplier
C. Stream Processing
   1. Filtering, Sorting, Mapping
   2. Terminal Operations
D. Collectors
   1. Partitioning and Grouping
   2. Reducing and Summarizing
   3. Downstream Reductions

XII. Date/Time API
A. Overview
B. Dates, Times, and Instants
   1. Creating, Parsing, and Formatting
   2. Accessing Date and Time Fields
   3. Deriving New Values
   4. Time Zones
C. Periods and Durations
   1. Intervals of Time: Date-Based and Time-Based
   2. Adding and Subtracting Intervals

XIII. Overview of Build Tools (optional)
A. Javadoc
   1. Generating API Documentation
   2. Doc Comments and Doc Tags
B. Ant
   1. History and Overview
   2. Buildfiles, Targets, and Tasks
C. Maven
   1. Maven Concepts: Project, POM, Artifact, Naming, Dependency, Plugin, Repository
   2. Convention over Configuration
   3. Lifecycles and Phases
   4. Plugins and Goals

XIV. Logging and Log4j (optional)
A. Overview
   1. Popular Logging Frameworks
B. Writing Log Messages
   1. Creating Loggers and Writing Log Messages
   2. Log Levels
C. Configuration
   1. Appenders and Layouts
   2. Logger Configuration and Logger Hierarchy