Java SE: Exploiting Modularity and Other New Features Ed 1.1

JAVA

DURATION	MODULES	COURSE CODE
2 Days	**8 Lectures**	**—**

Course Overview

The Java SE: Exploiting Modularity and Other New Features course introduces the Java module system and other recent features that are carried forward to Java SE 11, including JShell, convenience methods, new techniques for working with streams, and managing deprecated APIs.

What You Will Learn

Introduction

- Course Goals
- Course Objectives
- Introductions
- Audience
- Prerequisites
- Course Map
- Day 1 Schedule (Jigsaw)
- Day 2 Schedule (Non-Jigsaw)
- Lesson Format
- Practice Environment
- JDK Enhancement Proposal (JEP)
- What's Not Covered in This Course
- Benefits Provided Automatically
- Additional Resources
- Practices

Why Modules?

- Objectives
- Topics
- What is a Module?
- Module System

- Modular Development in JDK 9
- Topics
- Reusing Code in Java: Applications as Classes
- Reusing Code in Java: Applications as Packages
- Reusing Code in Java
- Releasing Code in Java: JARs
- JAR Files and Distribution Issues
- Class Path Problems
- JAR Dependency Problems
- Example JAR Duplicate Class Problem 1
- Example JAR Duplicate Class Problem 2
- Topics
- Accessibility
- Module System: Advantages
- Dependencies Across Modules
- A Modular Java Application
- Summary
- Practice: Overview

Working with the Module System

- Objectives
- Topics
- What is a Module
- Module Dependencies with requires
- Module Package Availability with exports
- Module Graph 1
- Module Graph 2
- Transitive Dependencies
- Access to Types via Reflection
- Topics
- Example Hello World Modular Application Code
- Example Hello World Modular File Structure
- Topics
- Compiling a Modular Application
- Single Module Compilation Example
- Multi Module Compilation Example
- Creating a Modular JAR
- Run a Modular Application
- Topics
- TeamGameManager Application Example
- requires and requires transitive in TeamGameManager 1
- requires and requires transitive in TeamGameManager 2
- Topics
- Creating a Modular Project in NetBeans
- Quiz 3-1

- Quiz 3-2
- Quiz 3-3
- Quiz 3-4
- Summary of Keywords
- Summary of Accessibility Between Classes
- Summary
- Practices

The Modular JDK

- Objectives
- Topics
- Modular Development in JDK 9
- The JDK
- The Modular JDK
- Listing the JDK’s Modules
- Topics
- Java SE Modules
- The Modular Graph of Java SE
- The Base Module
- Finding the Right Platform Module
- Modules: Location of Some Tools in JDK 9
- Topics
- Java EE Modules in JDK 9
- Java EE Modules in JDK
- Resolving Java EE Modules in JDK 9
- Example: Resolving Java EE Modules in JDK 9
- Topics
- Changed JDK and JRE Layout
- Topics
- Using JDK Internal APIs
- JDK 9 Encapsulation Policy for JDK Internals
- Illegal Access to JDK Internals in JDK 9
- Quiz 4-1
- Quiz 4-2
- Additional Resources
- Summary
- Practices Overview

Creating Custom Runtime Images

- Objectives
- Topics
- What Is a Custom Runtime Image?
- Link Time
- Topics
- Using jlink to Create a Custom Runtime Image

- Using jlink to Create a Runtime Image
- Example: Using jlink to Create a Runtime Image
- Examining the Generated Image
- Modules Resolved in a Custom Runtime Image
- Advantages of a Custom Runtime Image
- JIMAGE Format
- Footprint of a Custom Runtime Image
- Topics
- Examining a Custom Runtime Image
- Running the Application
- jlink Resolves Transitive Dependencies
- Topics
- Using Plug-ins with the jlink Tool
- Optimizing a Custom Runtime Image
- Example: Optimizing a Custom Runtime Image
- Quiz 5-1
- Quiz 5-2
- Additional Resources
- Summary
- Practices Overview

Migration

- Objectives
- Topics
- The Non-Modular League Application
- Run the Application
- The Unnamed Module
- Topics
- Top down Migration and Automatic Modules
- Automatic Modules
- Top-Down Migration
- Creating module-info.java—Determining Dependencies
- Check Dependencies
- Soccer Module
- League Module
- Library JAR Automatic Module
- Typical Application Modularized
- Topics
- Bottom up Migration
- Bottom-Up Migration
- display.asci Module Migration
- Modularized Library
- Run Bottom-Up Migrated Application
- Fully Modularized Application
- Module Resolution

- Topics
- More About Libraries
- Run Application with Jackson Libraries
- Open Soccer to Reflection from Jackson Libraries
- Topics
- Split Packages
- Splitting a Java 8 Application into Modules
- Java SE 8 Application Designed with Split Packages
- Migration of Split Package JARs to Java SE 9
- Addressing Split Packages
- Topics
- Cyclic Dependencies
- Addressing Cyclic Dependency 1
- Addressing Cyclic Dependency 2
- Top down or Bottom up Migration Summary
- Summary
- Practice: Overview

Services

- Objectives
- Topics
- Modules and Services
- Components of a Service
- Produce and Consume Services
- Module Dependencies without Services
- Service Relationships
- Expressing Service Relationships
- Topics
- Using the Service Type in competition
- Choosing a Provider Class
- Module Dependencies and Services 1
- Module Dependencies and Services 2
- Module Dependencies and Services 3
- Designing a Service Type
- Topics
- TeamGameManager Application with Additional Services
- module-info.java for competition module
- module-info.java for league and knockout modules
- module-info.java for soccer and basketball modules
- Summary
- Practice: Overview

Multi-Release JAR Files

- Objectives
- Topics

- Packaging an Application for Different JDKs
- Packaging an Application for Different JDK Versions
- The Solution: A Multi-Release JAR file
- What Is a Multi-Release JAR File?
- Topics
- Structure of a JAR File
- Structure of a Multi-Release JAR File
- Example: Structure of a Multi-Release JAR File
- Search Process in an MRJAR
- Topics
- Creating a Multi-Release JAR File
- Example: Creating a Multi-Release JAR File
- Running the Main Class on Java SE 7
- Running the Main Class on Java SE 9
- Topics
- Creating a Modular Multi-Release JAR File
- Example: Creating a Modular Multi-Release JAR File
- Running the Main Class on Java SE 7
- Running the Main Class on Java SE 9 on the Class Path
- Running the Main Class on Java SE 9 on the Module Path
- Summary
- Practice: Overview

Private Methods in Interfaces

- Objectives
- Topics
- Private Methods in Interfaces
- Java SE 7 Interfaces
- Implementing Java SE 7 Interface Methods
- Example: Implementing abstract Methods
- Example: Duplicating Logic
- Quiz 9-1
- Topics
- Implementing Methods in Interfaces
- Example: Implementing default Methods
- Example: Inheriting default Methods
- Example: Overriding a default Method
- What About the Problems of Multiple Inheritance?
- Inheritance Rules of default Methods
- Interfaces Don't Replace Abstract Classes
- Quiz 9-2
- Topics
- What If default Methods Duplicate Logic?
- Duplication Between default Methods
- The Problem with This Approach

- Introducing private Methods in Interfaces
- Example: Using private Methods to Reduce Duplication Between
- default Methods
- Types of Methods in Java SE 9 Interfaces
- Quiz 9-3
- Summary
- Practices

Enhancements to the Stream API

- Objectives
- Topics
- One More Benefit of Default Methods
- Quiz 10-1
- Changing a Java Interface
- Topics
- Why Enhance the Stream API?
- Scenario: A Savings Account
- An Ordered List
- takeWhile Provides a Solution
- The Effects and Benefits of takeWhile
- An Unordered List
- filter vs takeWhile
- Scenario: A Scientific Experiment
- What About the Remaining Elements?
- dropWhile Provides a Solution
- The Effects and Benefits of dropWhile
- Other Notes on takeWhile and dropWhile
- Is There an Alternative to Generating This Stream Manually?
- iterate Is Overloaded in Java SE 9
- Similarities with the for Loop
- Quiz 10-2
- Topics
- Remember Optionals
- Converting an Optional into a Stream
- An Optional Method in the SavingsAccount Class
- The or Method
- The ifPresent and orElse Methods in Java SE 8
- The ifPresentOrElse Method
- Quiz 10-3
- Summary of New Stream Interface Methods
- Summary of New Optional Class Methods
- Summary of Lambda Expressions Types
- Summary
- Practices

JShell

- Objectives
- Topics
- Has This Happened to You?
- A Million Test Classes and Main Methods
- JShell Provides a Solution
- Topics
- Comparing Normal Execution with REPL
- Getting Started with JShell and REPL
- Scratch Variables
- Declaring Traditional Variables
- Code Snippets
- Completing a Code Snippet
- Tab Completion and Tab Tips
- Semicolons
- JShell Commands
- Importing Packages
- Quiz 11-1
- Topics
- JShell API
- Why Incorporate JShell in an IDE?
- Use Cases
- Two Ways to Open JShell in NetBeans
- Summary
- Practices

Convenience Methods for Collections

- Objectives
- Topics
- What Are Convenience Methods?
- Many Convenience Methods in Java SE 9
- Key Collections Interfaces
- Topics
- of Method for Lists
- of Method for Sets
- of Method for Maps
- Quiz 12-1
- Overloading the of Convenience Method
- Why Overload the of Method?
- Growing a Collection
- ofEntries Method for Maps
- Topics
- Immutability
- No Null Values

- No Duplicates
- Randomized Iteration Order
- Quiz 12-2
- Summary
- Practices

Convenience Methods for Arrays

- Objectives
- Topics
- Arrays
- Modeling DNA Strands
- Working with DNA Strands
- Working with DNA Strands by Using a for Loop
- Convenience Methods in the Arrays Class
- Equating DNA Strands
- DNA Subsequences
- Equating Subsequences of DNA
- Many Array Types
- Quiz 13-1
- Topics
- Returning to the DNA Scenario
- Measuring Differences in DNA
- Distances Between Elements
- Absolute Values with compareUnsigned
- Variants of the compare Method
- Comparing Subsequences of DNA
- Only Compare the First Mismatch
- Quiz 13-2
- Topics
- One Last Return to the DNA Scenario
- Pinpointing the Mismatch in DNA Strands
- Variants of the mismatch Method
- Pinpointing the Mismatch Between Subsequences of DNA
- Quiz 13-3
- Summary
- Practices

Enhanced Deprecation

- Objectives
- Topics
- What Is Deprecation?
- What Is Enhanced Deprecation?
- How Do You Deprecate an API?
- Using @deprecated
- Topics

- Enhancements to the @Deprecated Annotation in JDK 9
- Using the @Deprecated Annotation
- Example: Using the @Deprecated Annotation
- Examples of Deprecated APIs in JDK 9
- Topics
- Notifications and Warnings
- Compiler Deprecation Warnings
- Example: Compiler Deprecation Warnings
- Terminal Deprecation Warnings (forRemoval=true)
- Suppressing Deprecation Warnings
- Topics
- Static Code Analysis
- Benefits of jdeprscan
- Running jdeprscan
- Static Analysis Using jdeprscan
- jdeprscan Analysis Can Be Version-Specific
- Quiz 14-1
- Quiz 14-2
- Additional Resources
- Summary
- Practices Overview