Java SE 21 Programming Complete Ed 1 LVC

JAVA

DURATION	MODULES	COURSE CODE
5 Days	**3 Lectures**	**—**

Course Overview

It provides an in-depth coverage for all core Java topics and most frequently used APIs. It also offers a set of practical exercises teaching how to build a fully functioning Java application from scratch.

What You Will Learn

- I Course Introduction
- Course Goals I-2
- Audience I-3
- Course Structure I-4

Introduction to Java

- Objectives
- What Is Java?
- How Java Works
- Object-Oriented Principles
- Classes and Objects
- Classes
- Objects
- Inheritance
- Java APIs
- Java Keywords, Reserved Words, and Special Identifiers
- Java Naming Conventions
- Java Basic Syntax Rules
- Defining a Java Class
- Accessing Classes Across Packages
- Implementing Encapsulation with Access Modifiers
- Creating a Main Application Class
- Compiling a Java Program
- Executing a Java Program

- Comments and Documentation
- Code Snippets in Javadoc
- External Snippets
- Summary
- Practices for Lesson 1: Overview

Primitive Types, Operators, and Flow Control Statements

- Objectives
- Java Primitives
- Declaring and Initializing Primitive Variables
- Primitive Declarations and Initializations: Restrictions
- Java Operators
- Assignment and Arithmetic Operators
- Arithmetic Operations and Type Casting
- More Mathematical Operations
- Binary Number Representation
- Bitwise Logical Operators
- Equality, Relational, and Conditional Operators
- Short-Circuit Evaluation
- Flow Control Using if/else Construct
- Ternary Operator
- Flow Control Using switch Construct
- Switch -> No Fall-Through Syntax
- Switch Expressions yield a Value
- Switch Statements and Expressions Summary
- Using JShell (REPL Tool)
- JShell Commands
- Code Snippets
- Summary
- Practices for Lesson 2: Overview

Text, Date, Time, and Numeric Objects

- Objectives
- String Initialization
- String Operations
- String Indexing
- Mutable Text Objects
- Text Blocks
- Spaces, Lines, and Quotes
- Wrapper Classes for Primitives
- Representing Numbers Using BigDecimal Class
- Method Chaining
- Local Date and Time API
- More Local Date and Time Operations
- Instants, Durations, and Periods

- Zoned Date and Time
- Representing Languages and Countries
- Formatting and Parsing Numeric Values
- Formatting and Parsing Date and Time Values
- Localizable Resources
- Formatting Message Patterns
- Formatting and Localization: Summary
- Summary
- Practices for Lesson 3: Overview

Classes and Objects

- Objectives
- UML: Introduction
- Modeling Classes
- Modeling Interactions and Activities
- Designing Classes
- Creating Objects
- Defining Instance Variables
- Defining Instance Methods
- Object Creation and Access: Example
- Local Variables and Recursive Object Reference
- Local Variable Type Inference
- Defining Constants
- Static Context
- Accessing Static Context
- Combining Static and Final
- Other Static Context Use Cases
- IntelliJ IDE: Introduction
- Summary
- Practices for Lesson 4: Overview

Improved Class Design

- Objectives
- Overload Methods
- Variable Number of Arguments
- Defining Constructors
- Reusing Constructors
- Access Modifiers: Summary
- Defining Encapsulation
- Defining Immutability
- Constants and Immutability
- Enumerations
- Complex Enumerations
- Java Memory Allocation
- Parameter Passing

- Java Memory Cleanup
- Summary
- Practices for Lesson 5: Overview

Implement Inheritance and Use Records

- Objectives
- Extending Classes
- Object Class
- Reusing Parent Class Code Through Inheritance
- Instantiating Classes and Accessing Objects
- Rules of Reference Type Casting
- Verifying Object Type Before Casting the Reference
- Pattern Matching for instanceof
- Reference Code Within the Current or Parent Object
- Defining Subclass Constructors
- Class and Object Initialization: Summary
- Overriding Methods and Using Polymorphism
- Reusing Parent Class Logic in Overwritten Method
- Defining Abstract Classes and Methods
- Defining Final Classes and Methods
- Sealed Classes and Interfaces
- Overriding Object Class Operations: toString
- Overriding Object Class Operations: equals
- Override Object Class Operations: hashCode
- Comparing String Objects
- Java Records
- Custom Record Constructors
- Record Patterns
- Pattern Matching for switch
- Factory Methods
- Summary
- Practices for Lesson 6: Overview

Interfaces and Generics

- Objectives
- Java Interfaces
- Multiple Inheritance Problem
- Implement Interfaces
- Default, Private, and Static Methods in Interfaces
- Interface Hierarchy
- Default Methods Inheritance
- Interface Is a Type
- Functional Interfaces
- Generics
- Use Generics

- Examples of Java Interfaces: java.lang.Comparable
- Examples of Java Interfaces: java.util.Comparator
- Examples of Java Interfaces: java.lang.Cloneable
- Composition Pattern
- Summary
- Practices for Lesson 7: Overview

Arrays and Loops

- Objectives
- Arrays
- Combined Declaration, Creation, and Initialization of Arrays
- Multidimensional Arrays
- Copying Array Content
- Arrays Class
- Loops
- Processing Arrays by Using Loops
- Complex for Loops
- Embedded Loops
- Break and Continue
- Summary
- Practices for Lesson 8: Overview

Collections

- Objectives
- Introduction to Java Collection API
- Java Collection API Interfaces
- Java Collection API Implementation Classes
- Java Collection API Interfaces and Implementation Classes
- Create List Object
- Manage List
- Create Set Object
- Manage Set
- Create Deque Object
- Manage Deque
- Create HashMap Object
- Manage HashMap
- Iterate Through Collections
- Sequenced Collections
- Other Collection Behaviors
- Use java.util.Collections Class
- Access Collections Concurrently
- Prevent Collections Corruption
- Legacy Collection Classes
- Summary
- Practices for Lesson 9: Overview

Nested Classes and Lambda Expressions

- Objectives
- Types of Nested Classes
- Static Nested Classes
- Member Inner Classes
- Local Inner Classes
- Anonymous Inner Classes
- Anonymous Inner Classes and Functional Interfaces
- Understand Lambda Expressions
- Define Lambda Expression Parameters and Body
- Use Method References
- Default and Static Methods in Functional Interfaces
- Use Default and Static Methods of the Comparator Interface
- Use Default and Static Methods of the Predicate Interface
- Summary
- Practices for Lesson 10: Overview

Java Streams API

- Objectives
- Characteristics of Streams
- Create Streams Using Stream API
- Stream Pipeline Processing Operations
- Using Functional Interfaces
- Primitive Variants of Functional Interfaces
- Bi-argument Variants of Functional Interfaces
- Perform Actions with Stream Pipeline Elements
- Perform Filtering of Stream Pipeline Elements
- Perform Mapping of Stream Pipeline Elements
- Join Streams Using flatMap Operation
- Other Intermediate Stream Operations
- Short-Circuit Terminal Operations
- Process Stream Using count, min, max, sum, average Operations
- Aggregate Stream Data using reduce Operation
- General Logic of the collect Operation
- Using Basic Collectors
- Perform a Conversion of a Collector Result
- Perform Grouping or Partitioning of the Stream Content
- Mapping and Filtering with Respect to Groups or Partitions
- Parallel Stream Processing
- Parallel Stream Processing Guidelines
- Restrictions on Parallel Stream Processing
- Spliterator
- Spliterator Characteristics
- Summary

- Practices for Lesson 11: Overview

Exception Handling, Logging, and Debugging

- Objectives
- Using Java Logging API
- Logging Method Categories
- Guarded Logging
- Log Writing Handling
- Logging Configuration
- Describe Java Exceptions
- Create Custom Exceptions
- Throwing Exceptions
- Catching Exceptions
- Exceptions and the Execution Flow
- Helpful NullPointerExceptions
- Example Throwing an Unchecked Exception
- Example Throwing a Checked Exception
- Handling Exceptions
- Resource Auto-Closure
- Suppressed Exceptions
- Handle Exception Cause
- Java Debugger
- Debugger Actions
- Manipulate Program Data in Debug Mode
- Validate Program Logic Using Assertions
- Normal Program Flow with No Exceptions
- Program Flow Producing a Runtime Exception
- Program Flow Catching Specific Checked Exception
- Program Flow Catching Any Exceptions
- Summary
- Practices for Lesson 12: Overview

Java IO API

- Objectives
- Java Input-Output Principals
- Java Input-Output API
- Reading and Writing Binary Data
- Basic Binary Data Reading and Writing
- Reading and Writing Character Data
- Basic Character Data Reading and Writing
- Connecting Streams
- Standard Input and Output
- Using Console
- Understand Serialization
- Serializable Object Graph

- Object Serialization
- Serialization of Sensitive Information
- Customize Serialization Process
- Serialization and Versioning
- Working with Filesystems
- Constructing Filesystem Paths
- Navigating the Filesystem
- Analyze Path Properties
- Set Path Properties
- Create Paths
- Create Temporary Files and Folders
- Copy and Move Paths
- Delete Paths
- Handle Zip Archives
- Represent Zip Archive as a FileSystem
- Access HTTP Resources
- Summary
- Practices for Lesson 13: Overview

Java Concurrency and Multithreading

- Objectives
- Java Concurrency Concepts
- Implement Threads
- Thread Life Cycle
- Interrupt Thread
- Block Thread
- Make Thread Wait Until Notified
- Common Thread Properties
- Create Executor Service Objects
- Manage Executor Service Life Cycle
- Implementing Executor Service Tasks
- Locking Problems
- Writing Thread-Safe Code
- Ensure Consistent Access to Shared Data
- Nonblocking Atomic Actions
- Ensure Exclusive Object Access Using Intrinsic Locks
- Intrinsic Lock Automation
- Nonblocking Concurrency Automation
- Alternative Locking Mechanisms
- CPU Versus IO Bound Concurrent Tasks
- Virtual Threads API
- Virtual Thread Operations
- Summary
- Practices for Lesson 14: Overview

Modules and Deployment

- Objectives
- Compile, Package, and Execute Nonmodular Java Applications
- Nonmodular Java Characteristics
- What Is a Module?
- Java Modules
- Java Module Categories
- Define Module Dependencies
- Export Module Content
- Module Example
- Open Module Content
- Open an Entire Module
- Produce and Consume Services
- Services Example
- Multi-Release Module Archives
- Compile and Package a Module
- Execute a Modularized Application
- Migrating Legacy Java Applications Using Automatic module
- Create Custom Runtime Image
- Execute Runtime Image
- Optimize a Custom Runtime Image
- Check Dependencies
- Summary
- Practices for Lesson 15: Overview
- A Annotations
- Objectives A-2
- Annotations: Introduction A-3
- Design Annotations A-4
- Apply Annotations A-5
- Dynamically Discover Annotations A-6
- Document the Use of Annotations A-7
- Annotations that Validate Design A-8
- Deprecated Annotation A-9
- Suppress Compiler Warnings A-10
- Varargs and Heap Pollution A-11
- Summary A-12
- B Java Database Connectivity
- Objectives B-2
- Java Database Connectivity (JDBC) B-3
- JDBC API Structure B-4
- Manage Database Connections B-5
- Create and Execute Basic SQL Statements B-6
- Create and Execute Prepared SQL Statements B-7
- Create and Execute Callable SQL Statements B-8
- Process Query Results B-9