

# Java SE 17: Programming Complete LVC

JAVA

DURATION

**5 Days**

MODULES

**3 Lectures**

COURSE CODE

—

## Course Overview

This course is intended for students with some programming experience and is a comprehensive training for the Java programming language.

## What You Will Learn

### Introduction to Java

- Course Goals
- Audience
- Course Schedule
- Course Practices
- Objectives
- Object Oriented Principals
- What Is Java?
- How Java Works?
- Classes
- Objects
- Inheritance
- Java APIs
- Java Keywords, Reserved Words, and a Special Identifier
- Java Naming Conventions
- Java Basic Syntax Rules
- Define Java Class
- Access Classes Across Packages
- Use Access Modifiers
- Create Main Application Class
- Compile Java Program
- Execute Java Program
- Comments and Documentation

- Summary
- Practices for Lesson 1: Overview

## Primitive Types, Operators, and Flow Control Statements

- Objectives
- Java Primitives
- Declare and Initialize Primitive Variables
- Restrictions on Primitive Declarations and Initializations
- Java Operators
- Assignment and Arithmetic Operators
- Arithmetic Operations and Type Casting
- More Mathematical Operations
- Binary Number Representation
- Equality, Relational, and Conditional Operators
- Short-Circuit Evaluation
- Flow Control Using if/else Construct
- Ternary Operator
- Flow Control Using switch Construct
- Switch Expressions
- Yield a Value, Set a Variable
- Scope of Local Variables
- JShell
- Getting Started with JShell and REPL
- Code Snippets
- Summary
- Practices for Lesson 2: Overview

## Text, Date, Time, and Numeric Objects

- Objectives
- String Initialization
- String Operations
- String Indexing
- StringBuilder: Introduction
- Text Blocks
- String Methods Related to Text Blocks
- Escape Sequences
- Wrapper Classes for Primitives
- Representing Numbers Using BigDecimal Class
- Method Chaining
- Local Date and Time
- More Local Date and Time Operations
- Instants, Durations, and Periods
- Zoned Date and Time
- Represent Languages and Countries
- Format and Parse Numeric Values

- Format and Parse Date and Time Values
- Localizable Resources
- Format Message Patterns
- Using format() of Java.text.MessageFormat
- Formatting and Localization: Example
- Summary
- Practices for Lesson 3: Overview

## Classes and Objects

- Objectives
- UML: Introduction
- Modeling Classes
- Modeling Interactions and Activities
- Designing Classes
- Creating Objects
- Define Instance Variables
- Define Instance Methods
- Object Creation and Access: Example
- Local Variables and Recursive Object Reference
- Local Variable Type Inference
- Define 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
- Define Constructors
- Reuse Constructors
- Access Modifiers Summary
- Define Encapsulation
- Define 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
- Extend Classes
- Object Class
- Reuse Parent Class Code Through Inheritance
- Instantiating Classes and Accessing Objects
- Rules of Reference Type Casting
- Verify Object Type Before Casting the Reference
- Pattern Matching for instanceof
- Reference Code Within the Current or Parent Object
- Define Subclass Constructors
- Class and Object Initialization Summary
- Override Methods and Use Polymorphism
- Reuse Parent Class Logic in Overwritten Method
- Define Abstract Classes and Methods
- Define Final Classes and Methods
- Sealed Classes, Inheritance Under Control
- Sealed Classes
- Sealed Class Example
- Override Object Class Operations: toString
- Override Object Class Operations: equals
- Override Object Class Operations: hashCode
- Compare String Objects
- Factory Method Pattern
- Records
- Constructors for Record Classes
- Features of Record Classes
- Local Records
- Summary
- Practices for Lesson 6: Overview

## Interfaces and Generics

- Objectives
- Java Interfaces
- Multiple Inheritance Problem
- Implement Interfaces
- Default, Private, and Static Methods in Interfaces
- Inheritance Rules of default Methods, Rule 1
- Inheritance Rules of default Methods, Rule 2
- Inheritance Rules of default Methods, Rule 3
- Interface Hierarchy
- Interface is a Type
- Functional Interfaces

- Generics Improve Code Reusability
- 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
- The 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 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
- 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
- Java Streams Processing
- Spliterator
- Decomposition with trySplit()

- Integration with Streams
- Modifying LongStream Splitter
- Splitter TestSplitter
- Using TestSplitter
- 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
- Example Throwing an Unchecked Exception
- Example Throwing a Checked Exception
- Handling Exceptions
- Helpful NullPointerExceptions
- 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
- Analyse 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
- Summary
- Practices for Lesson 14: Overview

- Objectives
- Compile, Package, and Execute Nonmodular Java Applications
- Nonmodular Java Characteristics
- What Is a Module?
- Java Platform Module System (JPMS)
- JPMS 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
- Introduction to Annotations 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
- Var-args and Heap Pollution A-11
- Warnings for Value-Based Classes A-12
- Summary A-13
- 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
- Control Transactions B-11
- Discover Metadata B-12
- Customize ResultSet B-13
- Set Up ResultSet Type B-14
- Set Up ResultSet Concurrency and Holdability B-16
- Summary B-17
- C Java Security
- Objectives C-2
- Security as Nonfunctional Requirement C-3
- Security Threats C-4
- Denial of Service (DoS) Attack C-5
- Define Security Policies C-6
- Control Access Using Permissions C-8
- Execute Privileged Code C-9
- Secure Filesystem and IO Operations C-11
- Best Practices for Protecting Your Code C-12
- Erroneous Value Guards C-13
- Protect Sensitive Data (Part 1) C-14
- Protect Sensitive Data (Part 2) C-15
- Prevent SQL Injections C-16
- Prevent JavaScript Injections C-17
- Prevent XML Injections C-18
- Discover and Document Security Issues C-19
- Summary C-20
- D Advanced Generics
- Objectives D-2
- Compiler Erases Information About Generics D-3
- Generic and Raw Type Compatibility D-4
- Generics and Type Hierarchy D-5
- Wildcard Generics D-6
- Upper Bound Wildcard D-7
- Lower Bound Wildcard D-8
- Collections and Generics Best Practices D-9
- Summary D-10
- E Oracle Cloud Deployment
- Objectives E-2
- Cloud Application Requirements E-3
- Cloud Application Runtime Infrastructure E-4
- Cloud Java Application Servers E-5
- Package and Deploy Cloud Application E-6
- Example of the Microservices Architecture for an Online Mobile Application E-7
  - HTTP Protocol Basics E-8
  - REST Service Conventions and Resources E-10
  - Configure and Launch REST Service Application Using Helidon SE E-11
  - Summary E-12

- Practices for Appendix E: Overview E-13
- F Miscellaneous Java Topics
- Objectives F-2
- Builder Design Pattern F-3
- Factory Design Pattern F-6
- Singleton Design Pattern F-7
- Bitwise Logical Operators F-8
- Java: Regular Expressions F-11
- Character Classes F-12
- Predefined Character Classes F-13
- Quantifiers F-14
- Greediness F-15
- Boundary Matchers F-16
- Matching and Groups F-17
- Using the replaceAll Method F-18
- Threads Deadlock, Livelock, and Starvation F-19
- Stream Examples F-23
- Java IO, File Watch Service F-29
- Summary F-34
- Demos F-35