Java SE 11: Programming Complete

JAVA

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

Course Overview

The course is based on the current long-term support version of Java SE 11

What You Will Learn

Introduction to Java

- Course Goals
- Audience
- Course Schedule
- Course Practices
- Lesson Objectives
- 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

Primitive Types, Operators, and Flow Control Statements

- Objectives
- 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
- Bitwise Operators
- Equality, Relational, and Conditional Operators
- Short-Circuit Evaluation
- Flow Control Using if/else Construct
- Ternary Operator
- Flow Control Using switch Construct
- JShell
- Summary
- Practices

Text, Date, Time, and Numeric Objects

- Objectives
- String Initialization
- String Operations
- String Indexing
- StringBuilder: Introduction
- 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
- Formatting and Localization: Example
- Summary
- Practices

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
- NetBeans IDE: Introduction
- Summary
- Practices

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

Inheritance

- 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
- 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
- Override Object Class Operations: toString
- Override Object Class Operations: equals
- Override Object Class Operations: hashCode
- Compare String Objects
- Factory Method Pattern
- Summary
- Practices

Interfaces

- Objectives
- Java Interfaces
- Multiple Inheritance Problem
- Implement Interfaces
- Default, Private, and Static Methods in Interfaces
- Interface Hierarchy
- 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

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

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

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

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
- Summary
- Practices

Handle Exceptions and Fix Bugs

- 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
- 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

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

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
- Non-Blocking Atomic Actions
- Ensure Exclusive Object Access Using Intrinsic Locks
- Intrinsic Lock Automation
- Non-Blocking Concurrency Automation
- Alternative Locking Mechanisms
- Summary
- Practices

Java Modules

- Objectives
- Compile, Package, and Execute Non-Modular Java Applications
- What Is a Module?
- Java Platform Module System (JPMS)
- JPMS Module Categories
- Define Module Dependencies
- Export Module Content
- Modules 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 Modules
- Create Custom Runtime Image
- Execute Runtime Image
- Optimize a Custom Runtime Image
- Summary
- Practices
- A Annotations
- Objectives A-2
- Introduction to Annotations A-3
- Design Annotations A-4
- Apply Annotations A-5
- Dynamically Discover Annotations A-7
- Document the Use of Annotations A-9