

**Intermediate/Advanced Java Programming**

**Course Number:** JAV-104
**Duration:** 5 days

**Overview**

This Intermediate/Advanced Java Programming training course gives developers a review of Java foundational concepts before moving on to intermediate and advanced topics in Java and OO development. Participants learn functional programming with lambdas and streams, as well as the Java Platform Module System (JPMS). UML, Design Patterns, and the use of composition versus inheritance are taught, as is unit testing.

**Prerequisites**

All students must have a working knowledge of Java programming, including the use of inheritance, interfaces, and exceptions.

**Materials**

All Java training students receive courseware and a related textbook.

**Software Needed on Each Student PC**

* A recent version of Windows, macOS, or Linux with at least 8 GB RAM
* JDK 11 or later
* The Java tool the students are likely to use after the class (Eclipse or IntelliJ IDEA are recommended)
* Other free software - please contact us if you have purchased this class

**Objectives**

* Solidify their 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
* Leverage 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
* Understand 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
* Leverage the use cases for inner classes and refactor existing code to use them when appropriate
* Create and use custom annotations
* Use reflection and how to use it
* Understand the role of functional interfaces
* 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
* Create and use Java modules, understanding module descriptors, modular JARs, exports and dependencies, and the modulepath
* Understand the structure and behavior of the modular JDK, and how it supports modular applications as well as legacy classpath-based code
* Migrate classpath-based applications to Java 11, understanding the stages of migration and options available

**Outline**

* Introduction: Java State of the Union
	+ Java Release Cycle
	+ New Java Versions
* Review - Basics
	+ Java Environment
	+ Classes and Objects
		- Instance Variables, Methods, Constructors, Static Members
		- OO Principles: Data Encapsulation, Cohesion
		- Object Contracts: toString(), equals() and hashCode(), Comparable and Comparator
	+ Packages, Enums, Arrays
	+ Exceptions
	+ Date and Time API
	+ New Language Features
* Review (Inheritance and Interfaces)
	+ UML Overview
	+ Inheritance
		- Definition and IS-A Relationship
		- Method Overriding, @Override
		- OO Principles: Principle of Substitutability, Polymorphism and Encapsulation of Type, Coupling, Open-Closed Principle
		- Constructor Chaining
	+ Interfaces
		- Defining and Implementing, Interface Types
		- Interface Inheritance
	+ New Interface Features (Java 8+)
		- Default Methods, Static Methods
		- Functional Interfaces
	+ Guidelines
* JUnit
	+ Overview
	+ Tests, Assertions, and Fixtures
		- Writing and Running Tests
		- Assertions
		- Test Fixtures, @Before and @After, @BeforeClass and @AfterClass
		- Testing for Exceptions
	+ Best Practices and Test-Driven Development Overview (TDD)
* Collections and Generics
	+ Collections Overview
		- Generics and Type-Safe Collections
		- Diamond Operator
	+ Lists, Sets, and Maps
		- Interfaces and Contracts
		- Iteration and Autoboxing
		- Utility Classes - Collections and Arrays
	+ Writing Generic Classes
		- Inheritance with Generic Types
		- Wildcard Parameter Types
		- Type Erasure
* Techniques of Object Creation
	+ Design Patterns Overview
	+ Controlling Object Creation
		- Limitations of new Operator, Alternative Techniques
	+ Singleton Pattern
	+ Simple Factory
	+ Factory Method Pattern
	+ Other Techniques
		- Named Objects, JNDI
		- Dependency Injection Frameworks
* Using Composition and Inheritance Effectively
	+ Inheritance and Composition - Pros and Cons
		- Composition and Delegation
		- HAS-A, USES Relationships
	+ Strategy Pattern
	+ Decorator Pattern
	+ Façade and Other Patterns
		- Façade, Proxy, Template Method
* Inner Classes
	+ Overview and Motivation
		- Stronger Encapsulation, Rules and Caveats
	+ Defining and Using Inner Classes
		- Member-Level, Method-Local, Anonymous Classes
	+ Static Nested Classes
		- Nested Classes, Nested Interfaces, Nested Enums
* Annotations
	+ Overview
	+ Using Annotations
		- Target and Retention Policy
		- Annotation Parameters, Parameter Shortcuts
	+ Writing Custom Annotations
		- Syntax, Using the Meta-Annotations
		- Using a Custom Annotation
* Reflection
	+ Overview and API
		- The Class Called Class
		- Obtaining and Inspecting Class Objects
	+ Working with Objects Reflectively
		- Creating Instances, Invoking Methods, Setting Field Values
* Lambda Expressions
	+ Functional Interfaces and Lambdas
	+ Target Context
	+ Using Lambda Expressions
		- Syntax, Lambda Compatibility
		- Variable Capture
		- Type Inference
	+ Method References
		- Three Types of Method References
		- Refactoring Lambdas into Method References
* Streams
	+ Overview
		- Streams vs. Collections
		- Anatomy of a Stream
	+ Understanding the Stream API
		- Intermediate Operations and Stream Pipeline
		- Java 8 Functional Interfaces: Predicate, Comparator, Function
	+ Stream Processing
		- Filtering, Sorting, Mapping
		- Terminal Operations
	+ Collectors
		- Concepts
		- Partitioning and Grouping
* Introduction to Modules
	+ Motivation and Overview
	+ Types of Modules
	+ Modular JDK
	+ Our Approach
* Working with Modules
	+ Defining and Using Modules
	+ Services
	+ Compatibility and Migration
* Conclusion