Core Java 25 Programming Developer's Workshop - TT2100

Get hands-on with modern Java to build scalable apps, boost performance, streamline workflows, and create enterprise-ready solutions

Duration: 5 Days
Skill Level: Introductory
Available Format: Instructor-Led Online; Instructor-Led, Onsite In Person ; Blended; On Public Schedule

What You'll Learn

Overview

Throughout this five-day, hands-on Java training course, students learn the best practices for writing great object-oriented programs in Java 25, using sound development techniques, modern features for improved performance, and new capabilities that accelerate rapid application development.

This course explores key enhancements introduced since Java 11, including Records, Sealed Classes, Text Blocks, Pattern Matching, Virtual Threads, and numerous API updates. In addition, Java 25 continues the trend of simplifying the language for newcomers by providing streamlined syntax, enhanced pattern matching, and a more approachable set of defaults that make it easier for developers new to Java to quickly write correct, readable, and efficient code without getting bogged down in boilerplate.

With Java 25, learning the language as a beginner has become more approachable than ever. New features such as Compact Source Files and Instance Main Methods allow students to create and run small programs with minimal setup, eliminating much of the boilerplate code that used to overwhelm newcomers. Flexible Constructor Bodies simplify class creation by reducing strict syntax requirements, making it easier to focus on concepts instead of technical hurdles. Additionally, Pattern Matching for Primitives streamlines common coding tasks by replacing verbose casting with more natural and readable code. These improvements reflect Javaâ€™s ongoing effort to lower the learning curve, helping new developers get started faster while still building a foundation that is compatible with professional software development practices.

Developers leaving this course will be well-prepared to work on Java 8, Java 11, Java 17, and Java 21 projects, while also being ready to contribute effectively to modern projects using Java 25.

Previous versions, including Java 11, and 17, 21 are also available upon request. Contact us for details.

Objectives

This **skills-centric** course is about **50% hands-on lab and 50% lecture** , designed to train attendees in basic OO coding and Java development skills, coupling the most current, effective techniques with the soundest industry practices. Throughout the course students will be led through a series of progressively advanced topics, where each topic consists of lecture, group discussion, comprehensive hands-on lab exercises, and lab review.

Our engaging instructors and mentors are highly experienced practitioners who bring years of current " on-the-job " experience into every classroom. Working in a hands-on learning environment, guided by our expert team, attendees will learn to:

- Understand what object-oriented (OO) programming is and recognize the advantages it provides in today's software development world.
- Gain a solid grasp of the fundamentals of the Java language, including its importance, uses, strengths, and limitations.
- Connect the basics of the Java language to OO programming and the Java Object Model.
- Learn to use Java's exception handling features to build more reliable applications.

- Work with the Java Modular System (Project Jigsaw) to create organized, maintainable applications.
- Design and implement classes that demonstrate inheritance and polymorphism.
- Use collections, generics, autoboxing, and enumerations to efficiently manage data.
- Process large volumes of data using lambda expressions and the Stream API.
- Define and implement abstract, static, and private methods in interfaces.
- Take advantage of Java development tooling available in modern programming environments.
- Write modern Java code using switch expressions for more concise and expressive branching logic.
- Use text blocks to create clean, multi-line string literals.
- Apply pattern matching for instanceof to write safer and more readable type checks.
- Introduce records as immutable data carriers to simplify domain models.
- Use pattern matching in switch statements to simplify conditional logic.
- Apply record patterns to deconstruct and access data directly within records.

Specific Java 25 features that are covered in the course include:

- Learn how to use **primitive types in patterns**, allowing pattern matching to work directly with primitive values for safer and more concise code.
- Understand **module import declarations**, which simplify modular programming by making module dependencies clearer and easier to manage.
- Explore **flexible constructor bodies**, which give developers greater control over how constructors are structured and how initialization logic is handled.
- Practice writing **compact source files and instance main methods**, which reduce boilerplate and make it easier for beginners to create simple Java applications.

Audience

Participants should be familiar with basic programming concepts such as variables, control structures, functions/methods, and data structures.

Pre-Requisites

This is a foundational Java programming course designed for attendees who already have prior development experience in another programming language. Participants should be familiar with basic programming concepts such as variables, control structures, functions/methods, and data structures.

Agenda

1) The Java Platform

Introduce the Java platform and its architecture, including the Java Standard Edition, JVM responsibilities, and the lifecycle of a Java program. Students will also explore garbage collection and learn about **Compact Source Files & Instance Main Methods** for simplified program entry points.

- Introduce the Java Platform
- Explore the Java Standard Edition
- Discuss the lifecycle of a Java Program
- Explain the responsibilities of the JVM
- Executing Java programs
- Compact Source Files & Instance Main Methods
- Garbage Collection

2) Using the JDK

Walk through the JDK file structure and how to compile and run Java programs from the command line. This lesson emphasizes documentation, code reuse.

- Explain the JDK's file structure
- Use the command line compiler to compile a Java class
- Use the command line Java interpreter to run a Java application class
- Documentation and Code Reuse
- Lab: Exploring MemoryViewer
- Lab: The SwingSet demo

3) Using the IntelliJ IDE

Learn the basics of IntelliJ IDEA, including projects, modules, and running applications. Students practice with IntelliJ IDEA and gain experience navigating modern Java IDE workflows.

- Introduce the IntelliJ IDE
- The Basics of the IntelliJ interface
- IntelliJ Projects and Modules
- Creating and running Java applications

- Tutorial: Working with IntelliJ (Community Edition)

4) Writing a Simple Class

Understand the structure of a simple Java class, defining variables, creating instances, and implementing a main method. Students also learn about Java keywords, reserved words, and object references.

- Write a Java class that does not explicitly extend another class
- Define instance variables for a Java class
- Create object instances
- Primitives vs Object References
- Implement a main method to create an instance of the defined class
- Java keywords and reserved words
- Lab: Create a Simple Class

5) Adding Methods to the Class

Dive deeper into classes by writing accessor methods, constructors, and using this for clarity. This lesson also introduces annotations, deprecation, and reinforces best practices through a hands-on lab.

- Write a class with accessor methods to read and write instance variables
- Write a constructor to initialize an instance with data
- Write a constructor that calls other constructors of the class to benefit from code reuse
- Use the this keyword to distinguish local variables from instance variables
- Introducing annotations
- Deprecating classes and methods
- Lab: Create a Class with Methods

6) Object-Oriented Programming

Explore the principles of OO programming through real-world examples. Students learn how objects, classes, methods, and messages interact, building a strong conceptual foundation.

- Real-World Objects
- Classes and Objects
- Object Behavior
- Methods and Messages
- Lab: Define and use a New Java class
- Lab: Define and use Another Java Class (optional)

7) Language Statements

Practice Java language constructs including operators, loops, and conditional logic. Special focus is placed on **Switch Expressions**, the yield keyword, and **Pattern Matching with Primitive Types** introduced in Java 25.

- Arithmetic operators
- Operators to increment and decrement numbers
- Comparison operators
- Logical operators
- Return type of comparison and logical operators
- Use for loops
- Switch Expressions
- Switch Expressions and yield
- Primitive Types in Pattern Matching (instanceof and switch)
- Lab: Looping (optional)
- Lab: Language Statements
- Lab: Switch Expressions

8) Using Strings and Text Blocks

Work with the String class and related utilities, compare String, StringBuffer, and StringBuilder, and practice text handling with **Text Blocks** and Unicode support.

- Create an instance of the String class
- Test if two strings are equal
- Perform a case-insensitive equality test
- Contrast String, StringBuffer, and StringBuilder
- Compact Strings
- Text Blocks

- Unicode support
- Lab: Fun with Strings
- Lab: Using StringBuffers and StringBuilders

9) Fields and Variables

Clarify variable scope and distinguish between instance, method, and block variables. Students also learn about static and final fields, as well as default values.

- Discuss Block Scoping Rules
- Distinguish between instance variables and method variables within a method
- Explain the difference between the terms field and variable
- List the default values for instance variables
- Final and Static fields and methods
- Lab: Field Test

10) Specializing in a Subclass

Examine inheritance by extending classes, overriding methods, and using instanceof. Students learn **Pattern Matching for instanceof** and explore **Flexible Constructor Bodies** added in Java 25 for improved subclass initialization.

- Constructing a class that extends another class
- Implementing equals and toString
- Writing constructors that pass initialization data to parent constructor
- Using instanceof to verify type of an object reference
- Overriding subclass methods
- Pattern matching for instanceof
- Safely casting references to a more refined type
- Flexible Constructor Bodies
- Lab: Creating Subclasses

11) Using Arrays

Introduce arrays, covering declaration, allocation, initialization, and variable argument methods.

- Declaring an array reference
- Allocating an array
- Initializing the entries in an array
- Writing methods with a variable number of arguments
- Lab: Creating an Array

12) Records

Learn how **Records** serve as immutable data carriers. Students define records, write canonical and compact constructors, and build lightweight domain objects.

- Data objects in Java
- Introduce records as carrier of immutable data
- Defining records
- The Canonical constructor
- Compact constructors
- Lab: Records

13) Java Packages and Visibility

Understand packages, imports, accessibility levels, and modularity. This module highlights **Module Import Declarations** from Java 25, which simplify modular code.

- Use the package keyword to define a class within a specific package
- Discuss levels of accessibility/visibility
- Using the import keyword to declare references to classes in a specific package
- Using the standard type naming conventions
- Introduce the Java Modular System
- Visibility in the Java Modular System
- Module Import Declarations (simplifying module dependencies)

14) Utility Classes

Explore wrapper classes, autoboxing, enums, static imports, and the Date/Time API. Students practice formatting, enumerations, and text handling.

- Introduce the wrapper classes
- Explain Autoboxing and Unboxing
- Converting String representations of primitive numbers into their primitive types
- Defining Enumerations
- Using static imports
- Introduce the Date/Time API
- LocalDate / LocalDateTime etc.
- Apply text formatting
- Using System.out.printf
- Lab: Enumerations
- Lab: TextBlocks
- Lab: Working with Dates (optional)

15) Inheritance and Polymorphism

Delve into polymorphism, type casting, and overriding rules. Students see how inheritance supports reusable, flexible designs.

- Write a subclass with a method that overrides a method in the superclass
- Group objects by their common supertype
- Utilize polymorphism
- Cast a supertype reference to a valid subtype reference
- Use the final keyword on methods and classes to prevent overriding
- Lab: Salaries - Polymorphism

16) Interfaces and Abstract Classes

Contrast abstract classes and interfaces, and learn when to use each. Students implement both, gaining clarity on contracts and polymorphic design.

- Define supertype contracts using abstract classes
- Implement concrete classes based on abstract classes
- Define supertype contracts using interfaces
- Implement concrete classes based on interfaces
- Explain advantage of interfaces over abstract classes

- Explain advantage of abstract classes over interfaces
- Lab: Interfaces

17) Sealed Classes

Learn how **Sealed Classes and Interfaces** restrict and control inheritance hierarchies. Students also see how sealed types integrate with pattern matching.

- Introduce sealed classes
- The sealed and permits modifier
- Sealed interfaces
- Sealed classes and pattern matching
- Lab: Sealed Classes

18) Pattern Matching

Explore modern **Pattern Matching** in Java, including switch expressions, sealed classes and record patterns.

- Pattern Matching in switch statements
- Pattern Matching and sealed classes
- Record Patterns
- Lab: Pattern Matching

19) Introduction to Exception Handling

Cover the basics of Java's exception hierarchy, and practice writing try/catch blocks to handle runtime errors.

- Introduce the Exception architecture
- Defining a try/catch blocks
- Checked vs Unchecked exceptions
- Lab: Exceptions

20) Exceptions

Deepen exception handling with custom exceptions, try-with-resources, suppressed exceptions, and enhanced null-safety features.

- Defining your own application exceptions
- Automatic closure of resources
- Suppressed exceptions
- Handling multiple exceptions in one catch
- Enhanced try-with-resources
- Helpful NullPointerException(s)
- Lab: Exceptional
- Lab: Helpful Nullpointers (optional)

21) Building Java Applications

Learn the build process, standard project layout, and dependency management. Students practice with **Maven.**

- Explain the steps involved in building applications
- Define the build process
- Introduce build scripts
- Explain the standard folder layout
- Resolving project dependencies
- Tutorial: Importing code Using Maven

22) Introduction to Generics

Understand the purpose of generics, parameterized types, bounded wildcards, and best practices for type-safe collections.

- Explain the purpose of generics in Java.
- Identify the risks of using non-generic collections.
- Describe how to define and use generic classes.
- Implement generic methods in Java.
- Distinguish between raw types and parameterized types.
- Apply type parameter naming conventions in generic code.
- Demonstrate the use of bounded types and bounded wildcards.

- Lab: Working with Generics

23) Introducing Lambda Expressions and Functional Interfaces

Introduce functional programming concepts in Java, focusing on lambda expressions, functional interfaces, and utility methods.

- Identify the core differences between functional and object-oriented programming.
- Explain the purpose and structure of functional interfaces in Java.
- Demonstrate how to implement functional interfaces with lambda expressions.
- Utilize utility methods that accept functional interfaces as parameters.
- Optimize lambda expressions for clarity and conciseness.
- Use generic functional interfaces for flexible code reuse.
- Lab: Lambdas

24) Collections

Review the Collection API, collection types, and iteration. This module includes newer collection features like **Sequenced Collections**.

- Provide an overview of the Collection API
- Review the different collection implementations (Set, List and Queue)
- Explore how generics are used with collections
- Examine iterators for working with collections
- Sequenced Collections
- Lab: Create a simple Game using Collections

25) Using Collections

Apply sorting, comparators, and lambdas to collections. Students work with **Sequenced Sets** and practice collection manipulation.

- Collection Sorting
- Comparators
- Using the Right Collection
- Lambda expressions in Collections

- Sequenced Sets
- Lab: Using Collections

26) Streams

Shift from imperative to declarative programming using the Stream API. Students learn filtering, finding, and collecting elements effectively.

- Understanding the problem with collections in Java
- Thinking of program solutions in a declarative way
- Use the Stream API to process collections of data
- Understand the difference between intermediate and terminal stream operations
- Filtering elements from a Stream
- Finding element(s) within a Stream
- Collecting the elements from a Stream into a List
- Lab: Working with Streams

27) Collectors

Practice advanced collection operations, grouping, and statistical aggregation using the Collectors utility class.

- Using different ways to collect the items from a Stream
- Grouping elements within a stream
- Gathering statistics about numeric property of elements in a stream
- Lab: Collecting

Additional Topic: Time Permitting

These topics will be included in your course materials but may or may not be presented during the live class depending on the pace of the course and attendee skill level and participation.

1) Introduction to Annotations

Learn the basics of annotations, how they are defined and used, and where they appear in modern frameworks.

- Discuss how annotations work in Java
- Understand what is required to work with Java's annotations
- Use annotations
- Other technologies that are using annotations

2) Java Data Access JDBC API

Understand database connectivity with JDBC, including queries, prepared statements, and data manipulation.

- Connecting to a database using JDBC
- Executing a statement against a database that returns a ResultSet
- Setting up and working with PreparedStatements
- Extracting multiple rows of data from a ResultSet
- Inserting, updating and deleting rows in a table
- Lab: Intro to JDBC

Follow On Courses

TT3335	Mastering Spring and Spring Boot

This course can be taught using a **local installation model**, where students install the Java Development Kit (JDK 25) and IntelliJ IDEA on their own systems. This approach allows participants to work directly on their personal development environment, gaining familiarity with tools they are likely to use in real-world projects. Installation instructions and guidance are provided before the course to ensure everyone is prepared to begin hands-on exercises right away.

Alternatively, the course can be delivered using our **Remote Desktop solution**, which provides students with access to a fully pre-configured environment. This setup includes JDK 25, IntelliJ IDEA, and all supporting tools installed and ready to use. The remote desktop option eliminates the need for local configuration and ensures a consistent, reliable experience for all participants, accessible through a standard Remote Desktop client or web browser. **This approach also enables the instructor to access each student's environment directly when needed, with full keyboard and mouse control, without ever accessing the student's personal computer. In addition, it reduces the risk of accidental sharing of company or personal information during screen sharing, since all work takes place within the controlled remote environment.**

For More Information

Please contact us or call 844-475-4559 toll free for more information about our training services (instructor-led, self-paced or blended), coaching and mentoring services, public course enrollment or questions, partner programs, courseware licensing options and more.