# Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11)

**Category:** Software Development & Programming | **Vendor:** Trivera Tech

**Duration:** 32.00 hours (4 days) **26.0 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

## Course Information

**Delivery Format:** Instructor Led - Online

## Course Overview

If you're an experienced OO developer (coming from a C# or C++ background, etc.) who needs to transition to programming in Java, this fast-paced, hands-on course will get you there quickly. Fast Track to Java Programming for OO Experienced Developers is a four-day, lab-intensive class where you'll quickly be immersed in working with the latest Java 11 programming techniques, using best practices for writing solid, robust (and well-written!) modern object-oriented applications. In addition to learning excellent, current coding skills in Java, you'll explore the new improved features for better performance and new capabilities for addressing rapid application development that Java 11 brings to the table.This course includes several key aspects that were introduced in Java 9, Java 10, and Java 11 including the Java Modular System, Local Variable Type Inference, and several API updates. This course also includes a Quick Look at what's next in Java Java 12, Java 13, Java 14 and beyond.

## About This Course

If you're an experienced OO developer (coming from a C# or C++ background, etc.) who needs to transition to programming in Java, this fast-paced, hands-on course will get you there quickly. Fast Track to Java Programming for OO Experienced Developers is a four-day, lab-intensive class where you'll quickly be immersed in working with the latest Java 11 programming techniques, using best practices for writing solid, robust (and well-written!) modern object-oriented applications. In addition to learning excellent, current coding skills in Java, you'll explore the new improved features for better performance and new capabilities for addressing rapid application development that Java 11 brings to the table.

This course includes several key aspects that were introduced in Java 9, Java 10, and Java 11 including the Java Modular System, Local Variable Type Inference, and several API updates. This course also includes a Quick Look at what's next in Java Java 12, Java 13, Java 14 and beyond.

## Who Should Attend

This is an introductory-level Java programming course, designed for experienced developers who wish to get up and running with Java, or who need to reinforce sound Java coding practices, immediately.

### Prerequisites & Entry Requirements

**General Prerequisites:**

To ensure a smooth learning experience and maximize the benefits of attending this course, you should have prior hands-on programming experience in another OO programming language such as C# or C++.

## Learning Outcomes

**Upon successful completion of this course, participants will be able to:**

Working in a hands-on learning environment, guided by our expert team, attendees will learn to:
Understand not only the fundamentals of the Java language, but also its importance, uses, strengths and weaknesses
Understand the basics of the Java language and how it relates to OO programming and the Object Model
Learn to use Java exception handling features
Work with the Modular system (Project Jigsaw)
Understand and use classes, inheritance and polymorphism
Understand and use collections, generics, autoboxing, and enumerations
Process large amount of data using Lambda expressions and the Stream API
Abstract, static and private methods in interfaces
Take advantage of the Java tooling that is available with the programming environment being used in the class
Specific Java 11 features covered: Using the Local Variable Type in Lambda expressions; Updates made to the String AP
Time Permitting: Quick look ahead – Java 12, Java 13, Java 14 and Beyond

## Detailed Course Outline

### The Java Platform

- Java Platforms
- Lifecycle of a Java Program
- Responsibilities of JVM
- Documentation and Code Reuse

### Using the JDK

- Setting Up Environment
- Locating Class Files
- Compiling Package Classes
- Source and Class Files
- Java Applications

### The Eclipse Paradigm

- Workbench and Workspace
- Views
- Editors
- Perspectives
- Projects

### Writing a Simple Class

- Classes in Java
- Class Modifiers and Types
- Class Instance Variables
- Primitives vs. Object References
- Creating Objects

### Adding Methods to the Class

- Passing Parameters into Methods
- Returning a Value from a Method
- Overloaded Methods
- Constructors
- Optimizing Constructor Usage

### Language Statements

- Operators
- Comparison and Logical Operators
- Looping
- Continue and Break Statements
- The switch Statement
- The for-each() Loop

## Using Strings

- Create an instance of the String class
- Test if two strings are equal
- Get the length of a string Parse a string for its token components
- Perform a case-insensitive equality test
- Build up a string using StringBuffer
- Contrast String, StringBuffer, and StringBuilder

## Specializing in a Subclass

- Extending a Class
- Casting
- The Object Class
- Default Constructor
- Implicit Constructor Chaining

## Fields and Variables

- Instance vs. Local Variables:
- Usage Differences
- Data Types
- Default Values
- Block Scoping Rules
- Final and Static Fields
- Static Methods

## Using Arrays

- Arrays
- Accessing the Array
- Multidimensional Arrays
- Copying Arrays
- Variable Arguments

## Local-Variable Type Inference

- Type inference
- Infering Types of Local Variables
- The var Reserved Type name
- Benefits of Using var
- Backward Compatibility

### Java Packages and Visibility

- Class Location of Packages
- The Package Keyword
- Importing Classes
- Executing Programs
- Visibility in the Modular System
- Java Naming Conventions

### Inheritance and Polymorphism

- Polymorphism: The Subclasses
- Upcasting vs. Downcasting
- Calling Superclass Methods from Subclass
- The final Keyword

### Interfaces and Abstract Classes

- Separating Capability from Implementation
- Abstract Classes
- Implementing an Interface
- Abstract Classes vs. Interfaces

### Introduction to Exception Handling

- Exception Architecture
- Throwing Exceptions
- Checked vs. Unchecked Exceptions

### Exceptions

- Handling Multiple Exceptions
- Automatic Closure of Resources
- Creating Your Own Exceptions

## Utility Classes

- Wrapper Classes
- Autoboxing/Unboxing
- Enumeration Syntax
- Using Static imports

## Introduction to Generics

- Generics and Subtyping
- Bounded Wildcards
- Generic Methods
- Legacy Calls to Generics
- When Generics Should Be Used

## Lambda Expressions and Functional Interface

- Lambda Expression Syntax
- Functional Interfaces
- Type Inference in Java 8
- Method references

## Collections

- Characterizing Collections
- Collection Interface Hierarchy
- The Set, List and Queue Interfaces
- Map Interfaces

## Using Collections

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

## Streams

- Processing Collections of data
- The Stream interface
- Reduction and Parallelism
- Filtering collection data
- Sorting Collection data
- Map collection data
- Find elements in Stream
- Numeric Streams
- Create infinite Streams
- Sources for using Streams

### Collectors

- Creating Collections from a Stream
- Group elements in the Stream
- Multi-level grouping of elements
- Partitioning Streams

### Introduction to the Module System

- Introduce Project Jigsaw
- Classpath and Encapsulation
- The JDK internal APIs
- Java 9 Platform modules
- Defining application modules
- Define module dependencies
- Implicit dependencies
- Implied Readability
- Exporting packages

### Java Date/Time

- The Date and Calendar classes
- Introduce the new Date/Time API
- LocalDate, LocalDateTime, etc.
- Formatting Dates
- Working with time zones
- Manipulate date/time values

### Java 12 and beyond

- Provide an overview of changes since Java 11
- Introduce Preview Features
- Records (Java 14)
- Switch Expressions (Java 12, Java 13, Java 14)
- Text Blocks (Java 13, Java 14)
- Helpful NullPointerExceptions (Java 14)
- Pattern Matching for instanceof (Java 14)

, 1. The Java Platform

¢ 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

¢ Garbage Collection

¢ Documentation and Code Reuse

2. Using the JDK

¢ 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

3. Using the IntelliJ IDE

¢ Introduce the IntelliJ IDE

¢ The Basics of the IntelliJ interface

¢ IntelliJ Projects and Modules

¢ Creating and running Java applications

¢ Tutorial: Exploring your IDE ( with IntelliJ 2023.2 (Community Edition) or Eclipse IDE

4. Writing a Simple Class

¢ 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

5. Adding Methods to the Class

¢ 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

6. Language Statements

¢ 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

7. Using Strings and Text Blocks

¢ Create an instance of the String class

¢ Test if two strings are equal

¢ Perform a case-insensitive equality test

¢ Contrast String, and StringBuilder

¢ Compact Strings

¢ Text Blocks

¢ Unicode support

¢ Lab: Fun with Strings

¢ Lab: Using StringBuffers and StringBuilders

8. Fields and Variables

¢ 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

9. Specializing in a Subclass

¢ 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

10. Using Arrays

¢ Declaring an array reference

¢ Allocating an array

¢ Initializing the entries in an array

¢ Writing methods with a variable number of arguments

11. Records

¢ Data objects in Java

¢ Introduce records as carrier of immutable data

¢ Defining records

¢ The Canonical constructor

¢ Compact constructors

12. Java Packages and Visibility

¢ 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

13. Utility Classes

¢ 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

14. Inheritance and Polymorphism

¢ 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

15. Interfaces and Abstract Classes

¢ 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

16. Sealed Classes

¢ Introduce sealed classes

¢ The sealed and permits modifier

¢ Sealed interfaces

¢ Sealed classes and pattern matching

17. Pattern Matching

¢ Pattern Matching in switch statements

¢ Pattern Matching and sealed classes

¢ Record Patterns

18. Introduction to Exception Handling

¢ Introduce the Exception architecture

¢ Defining a try/catch blocks

¢ Checked vs Unchecked exceptions

19. Exceptions

¢ Defining your own application exceptions

¢ Automatic closure of resources

¢ Suppressed exceptions

¢ Handling multiple exceptions in one catch

¢ Enhanced try-with-resources

¢ Helpful NullPointerException(s)

20. Building Java Applications

¢ 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

21. Introduction to Generics

¢ Generics and Subtyping

¢ Bounded Wildcards

¢ Generic Methods

22. Introducing Lambda Expressions and Functional Interfaces

¢ Understanding the concept of functional programming

¢ Understanding functional interfaces

¢ Lambda's and type inference

23. 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

24. Using Collections

¢ Collection Sorting

¢ Comparators

¢ Using the Right Collection

¢ Lambda expressions in Collections

¢ Sequenced Collections

Bonus Topics / Time Permitting

Streams

¢ 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

Collectors

¢ 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

# Additional Course Details

Nexus Humans Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11) training program is a workshop that presents an invigorating mix of sessions, lessons, and masterclasses meticulously crafted to propel your learning expedition forward.

This immersive bootcamp-style experience boasts interactive lectures, hands-on labs, and collaborative hackathons, all strategically designed to fortify fundamental concepts.

Guided by seasoned coaches, each session offers priceless insights and practical skills crucial for honing your expertise. Whether you're stepping into the realm of professional skills or a seasoned professional, this comprehensive course ensures you're equipped with the knowledge and prowess necessary for success.

While we feel this is the best course for the Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11) course and one of our Top 10 we encourage you to read the course outline to make sure it is the right content for you.

Additionally, private sessions, closed classes or dedicated events are available both live online and at our training centres in Dublin and London, as well as at your offices anywhere in the UK, Ireland or across EMEA.

# Frequently Asked Questions

### Q: What delivery options are available for Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11)?

We offer multiple delivery formats:

- Live Instructor-Led Classroom Online (Virtual/Live Online)
- Traditional Instructor-Led Classroom Training (ILT)
- On-site delivery at your offices anywhere in United Kingdom
- Private dedicated courses customized for your team

---

### Q: How many CPD hours does this course provide?

The 4-day Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11) course provides up to 26.0 CPD hours of structured learning. CPD certificates can be provided upon request.

---

### Q: What is the duration of the Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11) training?

The training takes place over 4 day(s), with each day lasting approximately 32.00 hours including breaks for lunch and refreshments.

---

### Q: Do you provide corporate training for Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11)?

Yes, we provide corporate training, dedicated training, and closed classes for Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11). Training can take place anywhere in United Kingdom including London, Manchester, Birmingham, Edinburgh, or live online allowing teams from across United Kingdom or internationally to attend.

---

### Q: What related terms do people search for?

Popular related searches include: Java

---

### Q: Why choose Nexus Human for Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11)?

Nexus Human is recognized as one of the leading training providers. Our trainers have won multiple awards including:

- Small Firms Best Trainer Award
- National Training Partner of the Year (Ireland) - Multiple Years
- Global Top 30 Instructor Awards (2012, 2019, 2021)
- Tech Excellence Award Nominations
- Learning Performance Institute (LPI) External Training Provider Sponsor 2024

---

### Q: Are there any discount codes available?

Yes! Use discount code **PENPAL5** when booking your Fast Track to Core Java Programming for Object Oriented Developers (TT2104-J11) training. Please note that only one discount code can be used per booking and cannot be combined with other special offers.


## Nexus Human

**Professional Training & Development**

Email: info@nexushuman.com

Website: www.nexushuman.com

Phone: +353 1 XXX XXXX (Ireland) | +44 20 XXXX XXXX (UK)