Faysal Aziz

Foundational Java

Core programming with a guided introduction to Spring Boot

FA-0617 · 3 days or 6 half-day sessions

Software Development

Practise Java syntax, object-oriented programming and standard APIs, then extend a prepared Spring Boot example with persistence and tests.

Introduction

This course is for participants who already understand basic programming concepts and want a practical introduction to Java. It progresses from syntax and tooling to object-oriented design, collections, streams, exceptions and introductory concurrency.

The final day uses a prepared Spring Boot project to connect a small REST endpoint, persistence example and selected tests. Framework and packaging topics are introductory; completing the workshop does not establish production readiness or expertise in enterprise concurrency.

Learning outcomes

The course teaches participants to:

- Write and debug small Java console programs using types, control flow, methods and exceptions.
- Use an IDE, Git and one selected build tool to compile and test a project.
- Apply classes, interfaces, encapsulation, inheritance and composition in small examples.
- Use collections, generics, streams, lambdas and java.time for common tasks.
- Recognise basic thread, executor and synchronisation concepts and their limitations.
- Extend a prepared Spring Boot application with a simple endpoint and Jakarta Persistence example.
- Write selected JUnit tests, package the application and run it locally.

Prerequisites

- Basic understanding of programming concepts (variables, functions, control flow) in any language.

A computer with an IDE and a compatible JDK selected for the workshop, such as JDK 21 or 25. Build-tool, Spring Boot and testing-library versions must be aligned with that JDK before the course.

- Internet connectivity to download dependencies, libraries and frameworks.
- Optional but helpful: prior exposure to version control (Git) and a relational database (for persistence).

Training outline

1. Day 1 — Java Fundamentals and Tooling

Introduction to Java environment

- JDK installation, version selection and a consistent workshop environment. Distinguish stable language features from preview APIs.
- Setting up an IDE (IntelliJ, Eclipse, VS Code), configuring the project, compiling and running a simple “Hello World” program.
- Discuss folder/project structure, package naming, simple command-line compilation/IDE run.

Basic Java syntax

- Variables, data types (primitive types, wrappers, String), literals, constants.
- Operators: arithmetic, relational, logical, bitwise.
- Control flow: if-else, switch (including modern switch enhancements), loops (for, enhanced for, while, do-while).
- Methods (functions): declaration, arguments, return types, scope, overloading.
- Basic I/O and simple exception handling: try-catch, finally, throws.

Working in the IDE / simple program creation

- Create small console applications: e.g., compute factorial, iterate collections, simple input/output from console.
- Using IDE features: debugging, breakpoints, stepping through code, inspecting variables.

Version control and build tools

- Git basics: repository init/cloning, commit, branch, merge, push/pull (remote repository concept).

- Build tools introduction: Maven vs Gradle – project structure, dependency declaration, build lifecycle.
- Hands-on: create project using Maven (pom.xml) or Gradle (build.gradle), add a dependency (e.g., JUnit) and build/compile/run tests.
- Best practices: dependencies, plugin use, project modularity.

2. Day 2 — Object-Oriented Programming and APIs

Object-Oriented Programming Essentials

- Classes and objects: fields, constructors, methods.
- Encapsulation (private/protected/public), getters/setters.
- Inheritance: extends, super, method overriding.
- Polymorphism, abstract classes, interfaces, default/static methods in interfaces.
- Composition vs inheritance; design-thinking for simpler, maintainable code.

Java standard APIs and modern language features

- Collections framework: List, Set, Map; working with generics; iteration, traversal.
- Java 8+ features: Lambda expressions, functional interfaces, streams, method references.
- Date/Time API (java.time), optional, typesafe handling.
- Exception handling best practices: checked vs unchecked exceptions, custom exceptions, resource-management (try-with-resources).
- Introductory concurrency: threads, Runnable/Callable and simple synchronisation. Virtual threads are a stable feature from JDK 21; structured concurrency is discussed as a version-dependent preview API, not a foundation required for the lab.

Hands-on coding and small project

- Build a small application that uses collections + streams + lambdas (e.g., filter/process a list of objects).
- Add exception handling and examine one small thread/synchronisation example; identify shared-state risks.

Build and run with the tools

- Integrate the application into the Maven/Gradle build.
- Use Git to track code changes, revert, branch for feature work, merge back.

3. Day 3 – Guided Framework, Persistence and Testing Project

Introduction to Spring Framework and Spring Boot

- What is Spring, what problems does it solve (dependency injection, inversion of control).
- Setting up a Spring Boot project (e.g., with Spring Initializr), key annotations (@SpringBootApplication, @Component, @Service, @Repository, @RestController).
- Building REST endpoints, wiring components, basic configuration.

Persistence with Jakarta Persistence (JPA)

- Jakarta Persistence concepts and integration with Spring Data JPA; use the namespaces and versions appropriate to the selected framework.
- Entities: @Entity, @Id, fields, relationships (one-to-many, many-to-one).
- Spring Data JPA repositories, CRUD operations, custom queries.
- Database setup: in-memory (H2) or light SQL for demo purposes.

Unit testing and integration testing with JUnit

- Writing tests with JUnit (setup, teardown, assertions).
- Selected component and integration-test examples appropriate to the chosen Spring/JUnit versions.
- Test-first thinking, useful assertions and a high-level introduction to coverage and CI.

Threads, concurrency in framework context

- Demonstration of asynchronous methods and task executors in a prepared Spring context; do not equate asynchronous execution with thread safety.
- Exception handling in multi-threaded applications.

Packaging and Build Configuration

- Package the example application as an executable JAR using one selected build tool; compare WAR packaging conceptually.
- Introduction to configuration profiles, reproducible builds and dependency review.
- Git best practices in team/enterprise context (branching models, pull requests, code reviews).

Final hands-on: small end-to-end application

- Extend a prepared small application combining Spring Boot, a simple JPA entity/repository, REST endpoints, selected unit tests, one build tool and Git.
- Run locally and test endpoints; review a prepared concurrent-request example rather than conducting production load testing.

Review, next steps, branching into advanced topics

- Review release-specific modern Java features, stable virtual threads and the preview status of structured concurrency; use matching official documentation.
- Further-learning paths: microservices, reactive programming, deployment and scaling. These are not full workshop implementation topics.
- Q&A, wrap-up, resources for further learning.

© 2026 Faysal Aziz · Kuala Lumpur