Faysal Aziz

Modern Java for Experienced Developers

Object-oriented business logic, recommendations and service-based applications

FA-0648 · 4 weeks

Software Development · Data & Analytics

Transfer existing programming experience to Java through object-oriented design, data processing, a small recommendation engine and a service-based project.

Introduction

This four-week programme is for experienced programmers, including developers working with legacy systems, who want to apply their existing business-logic and data skills in Java. It progresses through language/OOP foundations, data and concurrency, a small recommendation engine and introductory service-based architecture.

Projects use fictional business data and prepared modules to keep scope bounded. The source specifies four weeks but not daily contact hours; no additional hours or delivery commitment is invented. The course develops practical prototypes and review skills, not guaranteed scalable production systems, career transition or complete migration expertise.

Learning outcomes

The course teaches participants to:

- Express a bounded business requirement using Java classes, interfaces and collections.
- Apply exception handling, file processing and suitable data representations.
- Review serialization risks and use appropriate validation/constraints in selected examples.
- Use concurrent tasks and identify shared-state, transaction and timing limits.
- Implement and evaluate simple content-based and collaborative-filtering examples.

- Build a small recommendation prototype and assess its quality with prepared data.
- Compare service-based and monolithic designs, including operational trade-offs.
- Implement and test a simplified service-based business example with explicit data and communication boundaries.

Prerequisites

Programming and Development Experience

- Strong experience with a programming language, ideally with object-oriented or procedural knowledge. Familiarity with designing, debugging, and maintaining code is essential.
- Understanding of software development lifecycles (e.g., Agile, Waterfall) and practices like version control, code documentation, and testing.

Data and Environment Skills

- Relational-data/SQL knowledge, data access, integrity and basic performance considerations.
- Unix/Linux command-line familiarity, resource concepts and basic TCP/IP/HTTP/SSH knowledge.

Prepared Tools and Access

A compatible supported JDK and Java IDE, Maven and supplied project dependencies. VS Code/remote tooling is optional according to the training environment.

Access to an authorised disposable database and local/remote training environment; SSH settings follow the supplied environment rather than assuming a fixed port.

Training outline

1. Week 1 — Java Foundations and OOP

Language and Tools

- Java setup, IDEs, and essential tools.
- Key syntax and program structure, including classes, methods, and packages.
- Introduction to Java data types, variables, and basic operators.

Object-Oriented Design

- Core OOP concepts: Classes, Objects, and Interfaces.
- Detailed exploration of Inheritance, Polymorphism, Abstraction, and Encapsulation.
- Compare OOP and procedural approaches, choosing representations appropriate to the business requirement rather than claiming universal superiority.

Collections for Business Logic

- Using Java's Collections framework: lists, sets, maps, and queues.
- Choose data structures according to access, ordering, duplication and workload requirements.
- Collections for batch-processing examples; transaction integrity also depends on database and application boundaries.

Practice

- Exercises focused on creating and manipulating objects.
- Building simple OOP applications that replicate business logic.

2. Week 2 — Data-Centric Design and Concurrency

Exception Handling

- Basics of exception handling: try-catch-finally blocks, throw and throws.
- Designing error-handling strategies in Java to align with business requirements.
- Custom exceptions for specific business scenarios and logic enforcement.

Files and Data Representations

- Reading and writing data files for business processing.
- I/O streams and bounded/streamed data processing; do not assume arbitrary datasets fit memory.
- Compare serialization/data representations and validation. Native Java deserialization needs careful trust boundaries and filtering; do not treat arbitrary incoming objects as safe.

Business Algorithms

- Translating business logic into algorithms and designing workflows in Java.
- Compare algorithm/data-structure choices using measurement and representative data.
- Step-by-step exercises to convert business requirements into Java methods.

Concurrent Processing

- Multithreading for concurrent data processing; standard Java threading does not by itself guarantee hard real-time deadlines.
- Using the Executor framework, Runnable and Callable interfaces.
- Concurrent-task exercises with explicit shared-state and transaction boundaries.

3. Week 3 — Recommendation Engine

Recommendation Approaches

- Types of recommendation engines: content-based, collaborative filtering, and hybrid methods.
- Practical applications of recommendation engines in business contexts, such as customer suggestions or product ranking.

Data Structures

- Utilizing hash maps, sets, and lists to manage user data and preferences.
- Select data structures for memory/access efficiency; recommendation quality depends on data and algorithm evaluation, not container choice alone.

Selected Algorithms

- Designing core algorithms to calculate similarity, relevance, or preferences.
- Implementing basic collaborative filtering and content-based recommendation models in Java.
- Integrate prepared data/storage and inspect update behaviour without an unlimited real-time promise.

Guided Prototype and Evaluation

- Build one bounded recommendation prototype with selected content-based and collaborative-filtering examples; hybrid approaches are compared at overview level.
- Test representative cases and evaluate quality/performance using appropriate prepared data; no accuracy guarantee.

4. Week 4 — Introductory Service-Based Applications

Architecture Choices

- Service boundaries, modularity and independent deployment concepts; scalability is design- and workload-dependent.
- Compare microservices and monoliths, including complexity, network failure and operational costs rather than assuming microservices are always better.

Small Java Services

- Structuring Java applications into small, self-contained services.
- API contracts and communication choices, including a prepared REST example.
- Optional overview of a compatible Spring Boot setup; no full framework syllabus is added.

Persistence and Consistency

- Database integration with microservices.
- Data ownership, consistency and failure trade-offs across service boundaries.
- Approaches to data storage and retrieval in a microservices setup.

Business-Application Project

- Design and implement a simplified business application using microservices.
- Run/deploy a small prepared service-based transaction example in a disposable environment, not a production migration.
- Tests and fault diagnosis with selected reliability/performance measurements.

Learning Approach

Instructor-led explanations, code labs and team problem-solving use illustrative business scenarios and supplied sample modules. Project reviews provide feedback on code, assumptions, tests and remaining limitations.

© 2026 Faysal Aziz · Kuala Lumpur