Course Outline

Java 12 Programming with Spring

Course Summary

Description

Java is one of the preferred languages among developers, used in everything right from smartphones, and game consoles to even supercomputers, and its new features simply add to the richness of the language. This course on Java programming begins by helping you learn how to install the Java Development Kit. You will then focus on understanding object-oriented programming (OOP), with exclusive insights into concepts like abstraction, encapsulation, inheritance, and polymorphism, which will help you when programming for real-world apps. Next, you'll cover fundamental programming structures of Java such as data structures and algorithms that will serve as the building blocks for your apps. You will also delve into core programming topics that will assist you with error handling, debugging, and testing your apps. As you progress, you'll move on to advanced topics such as Java libraries, database management, and network programming, which will hone your skills in building professional-grade apps. Further on, you'll understand how to create a graphic user interface using JavaFX and learn to build scalable apps by taking advantage of reactive and functional programming. You'll learn how to efficiently build and deploy microservices using Spring Boot. This spring/springboot with java microservices course will take you through tried and tested approaches to building distributed systems and implementing microservices architecture in your organization.

Objectives

By the end of this course, students will be able to:

- Strengthen your knowledge of important programming concepts and the latest features in Java
- Explore core programming topics including GUI programming, concurrency, and error handling
- Learn the idioms and best practices for writing high-quality Java code.
- Learn how to create SpringBoot microservices

Topics

- Getting Started with Java 12
- Java Object-Oriented Programming (OOP)
- Java Fundamentals
- Exception Handling
- Strings, Input/Output, and Files
- Data Structures, Generics, and Popular Utilities
- Java Standard and External Libraries
- Multithreading and Concurrent Processing
- JVM Structure and Garbage Collection
- Managing Data in a Database
- Functional Programming
- Microservices
- Introduction to Spring Boot
- Creating a Set of Cooperating Microservices
- Deploying Our Microservices Using Docker
- Adding an API Description Using OpenAPI/Swagger
- Adding Persistence
- Developing Reactive Microservices
- Introduction to Kubernetes

Course Outline

Java 12 Programming with Spring

Course Summary (cont.)

Audience

This course is geared for attendees with basic programming fundamentals knowledge who wish to be well versed with Java 10, 11, and 12, but also gain a perspective into the future of this language and software development in general.

Prerequisite

Students should have:

- Basic to Intermediate IT Skills. Attendees have basic programming fundamentals knowledge.
- Good foundational mathematics or logic skills

Duration

Five Days

Course Outline

Java 12 Programming with Spring

Course Outline

I. Getting Started with Java 12
A. How to install and run Java
B. How to install and run an IDE
C. Java primitive types and operators
D. String types and literals
E. Identifiers and variables
F. Java statements

II. Java Object-Oriented Programming (OOP)
A. OOP concepts
B. Class
C. Interface
D. Overloading, overriding, and hiding
E. Final variable, method, and classes
F. Polymorphism in action

III. Java Fundamentals
A. Packages, importing, and access
B. Java reference types
C. Reserved and restricted keywords
D. Usage of the this and super keywords
E. Converting between primitive types
F. Converting between primitive and reference types

IV. Exception Handling
A. Java exceptions framework
B. Checked and unchecked exceptions
C. The try, catch, and finally blocks
D. The throws statement
E. The throw statement
F. The assert statement
G. Best practices of exceptions handling

V. Strings, Input/Output, and Files
A. Strings processing
B. I/O streams
C. File management
D. Apache Commons utilities FileUtils and IOUtils

VI. Data Structures, Generics, and Popular Utilities
A. List, Set, and Map interfaces
B. Collections utilities
C. Arrays utilities
D. Object utilities
E. java.time package

VII. Java Standard and External Libraries
A. Java Class Library
B. External libraries

VIII. Multithreading and Concurrent Processing
A. Thread versus process
B. User thread versus daemon
C. Extending class thread
D. Implementing interface Runnable
E. Extending thread vs implementing Runnable
F. Using pool of threads
G. Getting results from thread
H. Parallel vs concurrent processing
I. Concurrent modification of the same resource

IX. JVM Structure and Garbage Collection
A. Java application execution
B. Java processes
C. JVM structure
D. Garbage collection

X. Managing Data in a Database
A. Creating a database
B. Creating a database structure
C. Connecting to a database
D. Releasing the connection
E. CRUD data

XI. Functional Programming
A. What is functional programming?
B. Standard functional interfaces
C. Lambda expression limitations
D. Method references

XII. Microservices
A. What is a microservice?
B. The size of a microservice
C. How microservices talk to each other
D. The reactive system of microservices

Course Outline

Java 12 Programming with Spring

Course Outline (cont.)

XIII. Introduction to Spring Boot
A. Technical requirements
B. Learning about Spring Boot
C. Beginning with Spring WebFlux
D. Exploring SpringFox
E. Understanding Spring Data
F. Understanding Spring Cloud Stream
G. Learning about Docker

XIV. Creating a Set of Cooperating Microservices
A. Technical requirements
B. Introducing the microservice landscape
C. Generating skeleton microservices
D. Adding RESTful APIs
E. Adding a composite microservice
F. Adding error handling
G. Testing APIs manually
H. Adding automated microservice tests in isolation
I. Adding semi-automated tests of a microservice landscape

XV. Deploying Our Microservices Using Docker
A. Technical requirements
B. Introduction to Docker
C. Challenges with running Java in Docker
D. Using Docker with one microservice
E. Managing a landscape of microservices using Docker Compose
F. Testing them all together automatically

XVI. Adding an API Description Using OpenAPI/Swagger
A. Technical requirements
B. Introduction to using SpringFox
C. Changes in the source code
D. Building and starting the microservice landscape
E. Trying out the Swagger documentation

XVII. Adding Persistence
A. Technical requirements
B. But first, let's see where we are heading
C. Adding a persistence layer to the core microservices
D. Writing automated tests that focus on persistence
E. Using the persistence layer in the service layer
F. Extending the composite service API
G. Adding databases to the Docker Compose landscape
H. Manual tests of the new APIs and the persistence layer
I. Updating the automated tests of the microservice landscape

XVIII. Developing Reactive Microservices
A. Technical requirements
B. Choosing between non-blocking synchronous APIs and event-driven asynchronous services
C. Developing non-blocking synchronous REST APIs using Spring
D. Developing event-driven asynchronous services
E. Running manual tests of the reactive microservice landscape
F. Running automated tests of the reactive microservice landscape

XIX. Introduction to Kubernetes
A. Technical requirements
B. Introducing Kubernetes concepts
C. Introducing Kubernetes API objects
D. Introducing Kubernetes runtime components
E. Creating a Kubernetes cluster using Minikube
F. Trying out a sample deployment
G. Managing a Kubernetes cluster