Modern Java

Course Summary

Description

Over twenty years after its inception, Java continues to be one of the most utilized programming languages in the world. But the software industry has changed dramatically during this time and so has Java. Java has evolved from its simple object-oriented and imperative roots to one that embraces reactive, modular, and functional programming paradigms. The outcome is Modern Java. This course will present you the most up to date features of Java, including the modular system, the REPL engine, type inference and the reactive programming. It will also take you on a deep dive in functional programming and offer a complete discussion on parallel and concurrent programming in Java – essential to all modern, core Java developers. This course will combine theoretical with practical examples and generous amounts of lab time to ensure the material presented is fully understood. It covers five learning modules over three days. Students will complete the course with a solid understanding of modern programming in Java 8, 9, 10 and 11.

Topics

- Understanding modularity, its objectives and how it relates to Java.
- Mastering the new modularity syntax in the Java Platform Module System (JPMS).
- Learning to apply modular thinking in real-world Java applications.
- Comparing the new module path concept to the classpath.
- Learning to use the new tools to build modular applications.
- Building modular Jars with versioning.
- Using Jlink to build light Java applications.
- Recognizing the goals of JPMS vis-à-vis microservices and OSGi.
- Understanding the history of functional programming and its principles tenets.
- Reconciling Java’s object oriented/imperative legacy with the functional paradigm.
- Understanding lambdas, lambda expressions, lambda blocks and how they differ from anonymous classes.
- Mastering the new Java 8 syntax for lambdas, functional interfaces, default methods and method references.
- Applying the standard functional interfaces in functional code.
- Using functional composition to create new functionality by combining disparate code segments.
- Utilizing the functionalized Collections library.
- Reasoning in functional streams.
- Writing efficient parallel code with parallel streams.
- Making appropriate use of mutable and non-mutable reduction with streams.
- Adapting functional concepts such as currying, higher-order functions, partial applications and recursion to Java 8.
- Comparing the differences between concurrent and parallel programming.
- Understanding threads, thread interruptions, synchronized blocks, Wait/notify, Executor service flavours, the fork-join framework, futures, atomics, Read/Write locks and conditions.
- Semaphores, barriers, countdown latches, exchangers, promises, and spliterators.
Modern Java

Course Summary (cont.)

- Knowing how to choose the right level of threading abstraction for the problem to solve.
- Understanding the goals of the reactive manifesto and how reactive programming is achieved in Java.
- Using the new JShell REPL engine for quick prototyping.
- Learning how to use the new HTTP 2 client libraries.
- Accessing processes natively in Java using the Process API.
- Understanding the new Stack Walker API.
- Recognizing the different garbage collectors in the platform and their strengths and weaknesses.
- Understanding the importance of type safety in Java.
- Grasping the concept of type erasure and the goals of generics.
- Understanding co-variance as it applies to regular Java objects vis-à-vis parametrized types.
- A peek into Java’s future with projects Valhalla, Coin and Amber

Audience

The course is modern take on Java and can be attended by beginner or advanced Java developers. It covers many of the new features that have been added in Java 8, 9 and 10 that reset the baseline for modern programming in Java. Students should take this advanced Java training if interested in learning more about the Java programming language in order to learn advanced programming techniques and concepts. Be prepared to participate and ask questions in class, as well as dive in and complete a variety of small labs (which can be done collaboratively with other class members).

Duration

Three days