Prerequisites

This course was developed for experienced Java Developers that want to catch up with the new capabilities that were added to the Java programming language during the last years.

Public Course

You can either attend the public version of this course (https://lifemichael.com/courses/javax), arrange having this course delivered at your company or having it delivered in person (one on one).

Duration

The public version of this course includes 4 meetings. Each meeting includes 5 academic hours delivered in 240 minutes (first 90 minutes include a lecture only, then we have a break of 15 minutes, and right after we have 135 minutes that include, mostly, coding exercises). When this course is delivered at your company or when delivered in person (one on one) we will adjust its delivery to your preferences. We can add or remove topics, and we can deliver it in whatever time frame you would like.

Premium Training

This course can be delivered either in Hebrew or in English. It can be delivered either online, in person, or in a hybrid way that allows the participants to choose whether to attend the meeting room where the training takes place or to join online.

The Lecturer

This seminar is delivered by Haim Michael, an experienced, well-known software development trainer more than 25 years of experience in software development training and 30 years of experience in Java programming. You can find more information about Haim Michael at https://blog.lifemichael.com.

The Topics

This course focuses on the new capabilities that were added to the Java programming language during the last decade. When this course is delivered in your company or delivered in person, we can add or remove any topic you choose. The following table lists the topics delivered when taking the public version of this course.

Meeting	Topic
1	**Inner Classes** We start with a short overview of nested classes and the special private case of instance class. We find that most developers need to familiarize themselves with the difference, which eventually damages their understanding of the lambda expressions topic. **Interface Default Methods** The ability to include in our interface a definition of non-abstract methods, also known as default methods, allows us to use interfaces as if they were traits. We explain what a trait can be useful and go over common scenarios in which the use of default methods might be useful. **Functional Interfaces** We explain when an interface becomes a functional interface, explain the purpose of functional interfaces, and explain special cases in which an interface becomes a functional interface even though it has more than one abstract method. **Lambda Expressions** We start with a short overview of the parts that most developers are already familiar with and quickly move forward to cover the more special cases, such as the use of the :: operator and in-depth understanding (in the context of inner and nested classes). **Java 8 Streams** We start by explaining how streams (Java's reply to generators in other programming languages) work, show the use of functional interfaces in this context, and explain in-depth what exactly happens. **Stream Gatherers** The Stream Gatherers API allows us to add more sophisticated operations to those already supported by the available functional interfaces.
2	**The Date/Time API** The Date/Time API was added in Java 8. It overcomes many of the drawbacks of the old Date/Time API. We will go over the main classes and interfaces this new API includes, and overview various practical examples for using this new API. **The Modules System** We will overview the module system introduced in Java 9 and explain various common cases we should be familiar with. **The HTTP Client** We will explore Java 11's standardization of the HTTP client API that implements

	HTTP/2 and Web Socket. This new HTTP client aims to replace the legacy HttpUrlConnection class that has been present in the JDK since the very early years of Java. **Local Variables Type Inference** We will overview the use of the var keyword that allows the compiler to infer the accurate type. We will overview the cases in which we can use the var keyword and learn how to reduce complex boilerplate code using this new keyword. **Switch Expressions** We will overview this new addition to the Java programming language, explain the syntax and the usability and introduce the yield statement in this context.
3	**Record Classes** We will introduce the possibility of defining record classes, explain the motivation for having that possibility, and provide an overview of various examples in the context of pattern matching. **Sealed Classes** We will explain what a sealed class is and overview various cases of using sealed classes in our code. In addition, we will also overview the constraints we have on the permitted subclasses. We will explore this topic further by introducing the possibility of defining a sealed interface and showing how to use record classes as permitted subclasses. **Text Blocks** We will overview the possibility of creating text blocks, include values of variables in those text blocks, and specify the format in which we want to display those values.
4	**Pattern Matching** We will cover the support of the Java programming language for Pattern Matching and demonstrate the use of Guarded Patterns and Records Patterns. **The Visitor Design Pattern *** Showing how to implement the Visitor Design Pattern can be easily implemented using pattern matching, records, switch expressions, and sealed types. **Virtual Threads** We will explain what a virtual thread is, understand how the use of virtual threads should lead to improvement in the performance of our code, and provide an overview of the cases in which the virtual threads are pinned to the carrier thread that handles them.

* if time allows