SYNTAX COLLEGE

6-DAY ADVANCED KOTLIN COURSE

Kotlin Mastery for Java Developers

Not "Kotlin syntax over Java syntax" – but how Kotlin enables more idiomatic, safe, expressive and coroutine-friendly code on Android and the JVM.

DURATION	FORMAT	LEVEL	HOURS
6 days	On-site, lecture + labs	Experienced Java developers	09:00–16:00

01 About the course

An intensive program for developers who want to go beyond syntax and truly master Kotlin. Participants unlearn Java habits and rebuild their thinking around Kotlin's conciseness, null-safety and expressiveness – then go deep into functional programming, Kotlin-style object orientation and DSLs, structured concurrency with Coroutines and Flow, production testing and dependency injection, and finally what happens under the hood: JNI, reflection, JVM internals and Java interop.

Every day combines lectures and code examples with hands-on practice and "code transformation" exercises that refactor real Java code into idiomatic Kotlin.

Who should attend

- Experienced Java developers (Android or backend)
- Teams migrating Java codebases to Kotlin
- Android developers who want deep Coroutines/Flow and testing skills

What participants will be able to do

- Refactor Java into idiomatic, null-safe Kotlin
- Design clean APIs with higher-order and inline functions, generics and DSLs
- Build structured-concurrency features with Coroutines, Flow and channels
- Test coroutine code with Kotest, MockK and Turbine; apply Hilt
- Interoperate safely with Java and native code

02 Day-by-day syllabus

DAY 1 Kotlin foundations for Java minds

Unlearn Java habits and rebuild thinking with Kotlin's syntax and control flow.

- Kotlin philosophy vs Java: conciseness, safety, expressiveness
- Type inference, val vs var, null-safety (?., ?:, !!) and smart casts
- if/when as expressions; ranges and progressions
- Read-only vs mutable collections; scope functions
- Default and named arguments; expression bodies; infix and extension functions
- Concise classes, init blocks, visibility, interfaces with defaults
- Data classes and destructuring
- Error handling without checked exceptions; @Throws; use { }

HANDS-ON LABS

- Null-safety refactoring
- Overloads → default args with @JvmOverloads
- POJO + Builder → data class and DSL
- when + smart casts refactoring

DAY 2 Idiomatic Kotlin – functions, lambdas and collections

Develop functional fluency and Kotlin-style APIs.

- Lambda syntax and memory model; capturing vs non-capturing
- Higher-order functions and function references; local vs non-local returns
- inline, noinline, crossinline; reified generics
- Value (inline) classes and type aliases
- Generics deep-dive: variance (in/out), star-projection
- Functional collections: map, flatMap, fold, groupBy, zip…
- Lazy sequences and performance

HANDS-ON LABS

- Lambda performance measurement
- Inline retry with noinline/crossinline
- Reified JSON decoder
- Units as value classes
- E-commerce analytics pipeline

DAY 3 Object orientation done the Kotlin way

Break away from Java-style class hierarchies.

- lazy, lateinit and property delegates
- object, companion object and @JvmStatic; class delegation with by
- SAM conversion, fun interfaces and operator invoke
- Operator overloading
- Nested vs inner classes; delegation vs inheritance
- Algebraic data types with sealed hierarchies
- DSL construction with lambdas with receiver
- Context receivers

HANDS-ON LABS

- Encrypted SharedPreferences via class delegation
- Form-validation DSL
- Text-to-speech app with ViewBinding and operator invoke

DAY 4 Coroutines and asynchronous thinking

Structured concurrency and callback refactoring.

- CoroutineScope, launch/async, withContext; cooperative cancellation
- Coroutines vs threads; dispatchers and custom contexts
- Callback → suspend with suspendCancellableCoroutine
- Cold Flows: operators, error handling, flatMapLatest
- Hot streams: StateFlow, SharedFlow, stateIn/shareIn; buffering and channels
- callbackFlow for event streams
- Lifecycle-aware collection and cleanup

HANDS-ON LABS

- Callback API → suspend function
- Thread loop → Flow with debounce and retry
- Permission and speech-recognition wrappers
- Reactive search UI pipeline with StateFlow

DAY 5 Production foundations – testing, DI and architecture

Must-have tools for production Kotlin on Android and the JVM.

- Kotest spec styles on the JUnit platform
- Testing coroutines and Flow: runTest, virtual time, Turbine
- Mocking with MockK; property-based testing
- Dependency injection with Hilt: modules, qualifiers, scopes, test replacement
- Clean layering with ViewModel + StateFlow; Compose interop (collectAsState, LaunchedEffect)

HANDS-ON LABS

- End-to-end coroutine/Flow test suite
- Refactor a Room MVVM app to Hilt
- Capstone: migrate a Java feature to idiomatic Kotlin

DAY 6 Under the hood – native bridges, JVM and interop

- JNI/NDK from Kotlin: external functions, JNI_OnLoad, native callbacks as suspend functions
- Kotlin reflection vs Java reflection
- JVM basics: class loaders, MethodHandles, invokedynamic, allocation tips
- Java interop annotations: @JvmStatic, @JvmOverloads, @JvmName, @JvmField
- Kotlin Multiplatform intro: commonMain and expect/actual

HANDS-ON LABS

- JNI bridge to zlib with coroutine-friendly APIs
- Reflection playground and mini DI container
- Custom class loader; reflection vs MethodHandle
- Exposing a Kotlin API to Java

03 Format & logistics

09:00–09:15	Kickoff and review
09:15–10:45	Lecture and code examples
11:00–12:30	Hands-on practice
13:30–15:00	Advanced concepts and labs
15:00–16:00	Code transformation and group exercises

Requirements

- A training room with a projector or large screen
- A laptop per participant with Android Studio installed (setup guide sent in advance)
- Android devices or emulators; USB debugging enabled
- Up to 15 participants per group

Days can be selected or reordered – e.g. a 3-day Coroutines & Flow track or a Java-to-Kotlin migration track.

04 Your instructor

All Syntax College courses are designed and taught personally by **Eran Katsav**, founder of Syntax College. Eran holds a B.Sc. in Computer Science from the Hebrew University of Jerusalem and an M.Sc. with honors from Tel Aviv University, and has more than 20 years of experience training engineers in industry, the defense sector and academia.

He lectures on native Android development in Kotlin and Java at Reichman University and at the Holon Institute of Technology (HIT), where he also heads the mobile lab. As an independent developer he created **Bip it**, the first Hebrew voice assistant, which reached about one million downloads in Israel; white-label versions of its engine ship in all Suzuki vehicles sold in Israel.

Selected clients: **Samsung, General Motors, Cisco, ironSource, CWT, Ophir Optronics**. Authorized supplier of the Israeli Ministry of Defense.

Bring this course to your team
Tailored syllabus, on-site at your offices, in Hebrew or English.

Eran Katsav
+972-50-6944922
info@syntax.org.il
www.syntax.org.il