KT. ACADEMY / COURSE CALENDAR

Kotlin Mastery

Six weeks with Marcin Moskała · February 2027

Plan your learning

For developers who already write Kotlin. Work through one lesson per teaching day, run its examples, and apply what you learn in quizzes and coding challenges.

Before Week 1: read the introduction and take the optional Kotlin refresher. The exact start date and live session times have not been announced.

Week	Focus	What you'll practise
1	The Kotlin you use every day	Make a function easier to call and harder to misuse.
2	Model the domain	Represent states explicitly instead of making callers interpret flags and nullable fields.
3	Design APIs in functional Kotlin	Build a small DSL that reads like the problem it solves.
4	Work with collections	Turn a data question into a pipeline you can explain and measure.
5	Put the compiler to work	Write reusable abstractions without asking callers for casts.
6	Kotlin beyond the language basics	Evaluate a Kotlin design at runtime, from Java and across platforms.

Course questions: contact@kt.academy

KOTLIN MASTERY / FEBRUARY 2027

Week 1 · The Kotlin you use every day

Make a function easier to call and harder to misuse.

Day 1	**Functions** Replace a family of overloads with named and default arguments.
Day 2	**Classes and composition** Decide what belongs in a constructor and when one class should delegate work to another.
Day 3	**Result, null or an exception?** Pick a failure signal that tells the caller what to do next.
Day 4	**The type system** Trace how Kotlin narrows a type and what the compiler can prove about nullability.
Day 5	**Extensions** Predict which extension a call resolves to before you run the code.

Week 2 · Model the domain

Represent states explicitly instead of making callers interpret flags and nullable fields.

Day 1	**Data classes** See which properties participate in equality, copying and destructuring.
Day 2	**Sealed types, enums and objects** Model a fixed set of values or a family of cases with different data.
Day 3	**Operators** Make operator overloads behave the way readers expect.
Day 4	**Mutability** Spot the difference between a read-only reference and a value that cannot change.
Day 5	**Function types and lambdas** Pass behaviour as a value and read function signatures with receivers.

Week 3 · Design APIs in functional Kotlin

Build a small DSL that reads like the problem it solves.

Day 1	**Function references** Choose between a lambda, a bound reference and an unbound reference.
Day 2	**Scope functions** Choose let, run, with, apply or also by receiver and return value.
Day 3	**DSL builders** Use lambdas with receivers to build a nested, type-safe API.
Day 4	**Context parameters** Express the dependencies an operation needs without threading them through every call.
Day 5	**Factory functions** Give construction a meaningful name and control which implementation callers receive.

Week 4 · Work with collections

Turn a data question into a pipeline you can explain and measure.

Day 1	**Collection types** Choose a list, set or map based on the operations your code actually needs.
Day 2	**Processing pipelines** Combine map, filter and aggregation without obscuring intermediate results.
Day 3	**Grouping and associating** Decide whether duplicate keys should overwrite values or collect them into groups.
Day 4	**Sorting and windowing** Rank items, compare neighbours and calculate results over a moving window.
Day 5	**Sequences and Flow** Contrast eager, lazy and asynchronous processing, including when laziness is not a win.

Week 5 · Put the compiler to work

Write reusable abstractions without asking callers for casts.

Day 1	**Generics**	Use bounds and type parameters to express relationships between inputs and results.
Day 2	**Inline functions**	Recognize where inlining helps and what reified type parameters make possible.
Day 3	**Variance**	Work out when a producer or consumer of one type can stand in for another.
Day 4	**Contracts**	Tell the compiler what a successful call implies for the code that follows.
Day 5	**Property delegation**	Move repeated property behaviour into a delegate and implement a lazy property.

Week 6 · Kotlin beyond the language basics

Evaluate a Kotlin design at runtime, from Java and across platforms.

Day 1	**Reflection**	Inspect properties at runtime and apply that knowledge to JSON serialization.
Day 2	**Performance**	Investigate a memory leak and identify unnecessary work in a data pipeline.
Day 3	**Java interoperability**	Inspect a Kotlin API from its Java caller’s perspective and improve its surface.
Day 4	**Kotlin Multiplatform**	Separate shareable logic from platform-specific dependencies.
Day 5	**Best practices**	Revisit earlier design choices and decide which idioms make the code easier to maintain.

Optional extra: KSP and compiler plugins.