Extracted from:

# Programming Kotlin

## Creating Elegant, Expressive, and Performant JVM and Android Applications



The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

# Programming Kotlin 1.3

Create Elegant, Expressive, and Performant JVM and Android Applications

Venkat Subramaniam

*edited by Jacquelyn Carter*

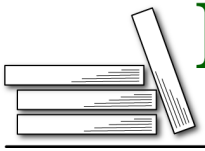
# Programming Kotlin

## Creating Elegant, Expressive, and Performant JVM and Android Applications

Venkat Subramaniam

The Pragmatic Bookshelf

Raleigh, North Carolina





Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Susan Conant
Development Editor: Jacquelyn Carter
Copy Editor: Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.



ISBN-13: 978-1-68050-635-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—September 2019

# Introduction

You can tell that programming languages fascinate me. As a language enthusiast, I code in about fifteen different languages and have written books on a few of them. I love languages, all of them.

Learning a language is like visiting a city. You come across new things but you also see things that are common. What's familiar brings comfort and the differences intrigue. To succeed in learning a language we need a good balance of the two.

Several of us aspire to be polyglot programmers. Kotlin is in itself a polyglot language. It brings together the powerful capabilities from many different languages. The creators of Kotlin took the good parts from various languages and combined them into one highly approachable and pragmatic language.

When I started learning Kotlin my mind was on fire, seeing the features I've enjoyed in different languages such as Java, Groovy, Scala, C++, C#, Ruby, Python, JavaScript, Erlang, Haskell... At the same time, there are so many nuances in Kotlin that made me so much more productive in programming than any one of those languages.

Some languages tell you how you should write code. Kotlin isn't one of those. With Kotlin, you decide which paradigm works best for the application at hand. For example, you may find that the imperative style is better in a large application because you have to deal with inherent side effects and exceptions, and Kotlin makes it easy to program in that style. But, in another part of the application, you may need to deal with big data transformations, so you decide the functional style is better for that, and Kotlin will transform into a charming functional programming language instantaneously. When you write object-oriented code, you'll see that the compiler works for you instead of you working for the compiler.

You'll see my excitement for the language throughout this book. It's been fun learning and applying Kotlin. I hope this book will make your journey to learn the language an enjoyable one. Thank you for picking it up.

## Who Is This Book For?

This book is for programmers, lead developers, architects, and technical managers. The book assumes familiarity with the basics of programming and also assumes moderate knowledge of Java and the JDK. The book doesn't assume any knowledge of Kotlin. If you're an Android programmer, this book lays the good foundation you'll need to program those devices with Kotlin, though this book doesn't focus exclusively on the Android platform.

If you're new to Kotlin, this book will help you get started and quickly move forward with the application of the language for your projects. If you're already using Kotlin, you may use this book to gain deeper understanding of some of the advanced features of the language.

You may also use this book to train your developers to get proficient with Kotlin, to use it to create highly fluent and expressive code, and to solve intricate problems.

## What's in This Book?

Kotlin is a multi-paradigm programming language. You may write plain scripts in Kotlin, write object-oriented code, functional style code, program asynchronously, and more. To provide reasonable coverage to this broad spectrum of topics, this book is divided into multiple parts.

Part I focuses on scripting with Kotlin. Part II is all about object-oriented programming. In Part III, you'll learn to use the functional-style capabilities of the language. Part IV will tie what you've learned so far together to make your code more fluent and teach you how to create internal domain-specific languages (DSLs). In Part V, you'll learn about the new coroutines and programming asynchronously. The final Part VI deals with Java interop, testing, using Kotlin with Spring, and programming Android applications with Kotlin.

Here's what's covered in each chapter:

In Chapter 1, Hello Kotlin, on page ?, we look at the reasons to use Kotlin, download the necessary tools, and get started with writing code.

Programmers coming to Kotlin from Java have a few practices and syntax to unlearn before they can jump into learning what's new and different in Kotlin. We'll cover those in Chapter 2, Kotlin Essentials for the Java Eyes, on page ?.

Functions are first-class citizens in Kotlin, and the language has so much to offer, like default and named arguments, and varargs. Explore these function-related capabilities in Chapter 3, Working with Functions, on page ?.

When programming in the imperative style, we often use external iterators. You'll see in Chapter 4, External Iteration and Argument Matching, on page ?, how Kotlin's iterators make that task bearable and how the argument matching syntax removes a lot of noise from conditional statements.

We work with collections extensively when programming. Chapter 5, Using Collections, on page ?, will show you how to use view interfaces to work with the JDK collections from Kotlin.

Kotlin has a sound type system, and its compile-time type checking goes beyond what we have come to expect from statically typed languages. In Chapter 6, Type Safety to Save the Day, on page ?, we'll look at Kotlin's fundamental types, nullable and non-nullable references, smart casts, generics variance, and more.

Though semantically equivalent, creating classes in Kotlin is quite different than in Java. In Chapter 7, Objects and Classes, on page ?, you'll learn to create singletons, classes, companion objects, and the reasons to use data classes.

Kotlin's treatment of inheritance is a lot different from the way it's used in Java. Classes are final by default, and the language places some rules to improve type safety and compile-time checks. We'll explore this topic in depth in Chapter 8, Class Hierarchies and Inheritance, on page ?.

As one of the languages that has direct support for delegation, Kotlin provides a few built-in delegates and also makes it easier to create custom delegates. We'll start with a discussion of when and why to use delegation and then dive into using delegates in Chapter 9, Extension Through Delegation, on page ?.

In Chapter 10, Functional Programming with Lambdas, on page ?, you'll learn how to create lambda expressions and how to write higher-order functions. We'll also cover the facilities offered in Kotlin to eliminate function call overhead and improve performance.

The internal iterators offer fluency, and sequences give us efficiency. We'll apply the functional style to iterate and process collections of objects in Chapter 11, Internal Iteration and Lazy Evaluation, on page ?.

Chapter 12, Fluency in Kotlin, on page ?, will show many of the capabilities of Kotlin to create concise, fluent, elegant, and expressive code.

Chapter 13, Creating Internal DSLs, on page ?, builds on the topic of fluency to create internal DSLs, to define your own syntax for your specialized language, but with full compile-time type safety.

Kotlin is one of the few languages on the JVM that provides tail call optimization. We'll see that in action in Chapter 14, Programming Recursion and Memoization, on page ?, along with using memoization to reduce computational complexity.

Coroutines are a stable feature in Kotlin 1.3 and, along with continuations, provide the fundamental infrastructure for programming asynchronously. The basics of coroutines and continuations are covered in Chapter 15, Exploring Coroutines, on page ?.

In Chapter 16, Asynchronous Programming, on page ?, you'll apply coroutines to create practical applications that can benefit from asynchronous program execution.

Kotlin can run on different platforms, including the Java Virtual Machine. In Chapter 17, Intermixing Java and Kotlin, on page ?, you'll learn how to intermix Kotlin with Java; how to use Kotlin in modern versions of Java—that is, with Java modules; how to use it with Maven and Gradle; and also how to smoothly work with both Java and Kotlin within the same application.

Even though the Kotlin compiler will catch several errors, automated testing is an essential practice for sustainable agile development. You'll learn about creating unit tests and measuring code coverage in Chapter 18, Unit Testing with Kotlin, on page ?.

In Chapter 19, Programming Spring Applications with Kotlin, on page ?, we'll explore the Spring libraries geared toward Kotlin programmers and the unique capabilities these offer.

Finally, in Chapter 20, Writing Android Applications with Kotlin, on page ?, we'll use Kotlin to create an Android application that talks to a back-end service.

## Kotlin and Java Versions Used in This Book

To run the examples in this book you should have Kotlin 1.3 and Java 1.6 or higher. Although most examples will also work in earlier versions of Kotlin, some examples will require Kotlin 1.3. The examples in the Java Interop

chapter will require Java 9 or later. Instructions for downloading the necessary tools are provided in the next chapter.

## How to Read the Code Examples

Most examples in this book are written as Kotlin script so you can easily run them as a single file without explicit compilation. Where compilation and other steps are necessary, instructions are provided alongside the code.

To save space, the output from a piece of code is shown as a comment line on the same line as the `println()` command where possible or on the next line. Occasionally, a line of comment says something about the code instead of showing the expected output.

## Online Resources

This book has an official page[1] at the Pragmatic Bookshelf website. From there you can download all the example source code for the book. You can also provide feedback by submitting errata entries.

If you're reading the book in PDF form, you can click on the link above a code listing to view or download the specific examples.

Now for some fun with Kotlin…

1. http://www.pragprog.com/titles/vskotlin