Preface

JetBrains created Kotlin for two reasons: there was no language that filled all the gaps in Android development using (legacy) Java libraries, and a new language would allow Android development to set trends, rather than just follow them.

In February 2015, Kotlin 1.0 was officially announced. Kotlin is concise, safe, pragmatic, and focused on interoperability with Java code. It can be used everywhere Java is used today: for server-side development, Android apps, desktop or portable clients, IoT device programming, and much, much more. Kotlin gained popularity among Android developers quite rapidly, and Google's decision to adopt Kotlin as the official language of Android development resulted in skyrocketing interest in the language. According to the Android Developers website, more than 60% of professional Android developers currently use Kotlin.

The learning curve in Android is rather steep: admittedly, it's hard to learn and harder to master. Part of the Android developer "upbringing," for many, is to be exposed over time to unintended interactions between the Android operating system and the application. This book intends to bring those kinds of exposures to readers in depth and up close by examining such problems in Android. We'll talk not only about Kotlin and Java, but also about the concurrency problems that arise when using Android and how Kotlin is able to solve them.

We will sometimes compare Kotlin to Java when we believe doing so provides better insight (especially since most readers are expected to have a Java background). We can demonstrate, with working examples, how to bridge that gap, and how the underlying concepts of most Kotlin operations are more similar to the Java equivalent than not. The tasks will be organized by topic to provide software engineers with a structured decomposition of that mass of information, and they will show how to make an application robust and maintainable.

Additionally, users familiar with Java—including Android developers—will find their learning curve dramatically flatten when we present each of the common tasks in both Java and Kotlin. Where appropriate, we'll discuss the difference and the pitfalls of one or both, but we hope to provide bite-size and easily digestible examples of a task that will "just work," and enable the reader to consume and adapt to the modern paradigm, as well as become aware of the significance of the updated code immediately and instinctively.

While Kotlin is fully interoperable with Java, other Java application development (server-side programming, desktop clients, middleware, etc.) has not caught on to the extent that Android has. This is largely due to the maintainer of Android (Google) strongly "encouraging" its users to make the change. Users are regularly migrating to Kotlin, but even more still fall back to Java for mission-critical work. Our hope is that this book will serve as the lifeline an Android developer needs to feel safe in committing to the advantages and simplicity that Kotlin represents.

Who Should Read This Book

Any of the over six million Android engineers. We believe that virtually every Android engineer could benefit from this book. While a small percentage will be fluent in Kotlin, even they will likely learn something from the information we'll present. But realistically, we're targeting the very large majority who haven't made the transition to Kotlin. This book is also for those who have dipped a toe in but not gained the same level of familiarity with Kotlin that they may have accrued in Java-centric Android development:

Scenario 1

> A reader is proficient in Java, heard of this new Kotlin language, and wants to try it out. So they read some online tutorial and start using it and it works great. Soon they realize that this isn't just a new syntax. The idioms aren't the same (e.g., functional programming, coroutines) and a whole new way of developing is now possi-

ble. But they lack guidelines, structure. For them, this book is a perfect fit.

Scenario 2

A reader is part of a small team of Java developers. They have discussions about whether they should start including Kotlin in their project. Even if Kotlin is said to be 100% interoperable with Java, some colleagues argue that introducing another language will add complexity to the project. Others suggest it might limit the number of colleagues who will be able to work on the project because of the need to master two languages. The reader could use this book to convince their colleagues, if they can show that the benefits will outweigh the costs.

Scenario 3

An experienced Android developer may have played around with Kotlin or written a feature in it, but still falls back to the home base of Java when things need to get done. This was the scenario we found ourselves in when realizing the book we're pitching now would have made our lives much easier. This is also the state we see most commonly around us—many Android devs have touched Kotlin, and many feel like they understand enough to write it when necessary, but they are either unaware, or simply unconvinced, of the significance of data classes, immutable properties, and structured concurrency. We think this book will turn a curious person into a committed evangelist.

Why We Wrote This Book

There are plenty of books that show how Android works, how Kotlin works, or how concurrency works. Kotlin is becoming wildly popular with Android development for its easy adoption and cleaner syntax, but Kotlin offers Android much more than that: it offers new ways to solve concurrency problems in Android. We wrote this book to provide a unique and specific intersectionality of these topics in great depth. Both

Android and Kotlin are rapidly changing, separately and together. Trying to keep up with all the changes can be difficult.

We view this book as a valuable checkpoint in history: showing where Android came from, where it is now, and how it will continue to evolve with Kotlin as the language matures.

Navigating This Book

Sometimes we include code snippets as screenshots instead of regular atlas code formatting. This is particularly useful with coroutines and flows, as suspension points are clearly identifiable. We also get type hints from the IDE.

Chapter 1, “Kotlin Essentials” and Chapter 2, “The Kotlin Collections Framework” cover major notable transitions made with Android in Kotlin. While the information in these chapters is enough to give you a good grounding in Kotlin, further chapters will take a deeper dive into more complex/advanced features. Users familiar with Java or similar syntactic structures will find the translation surprisingly natural.

Chapter 3, “Android Fundamentals” and Chapter 4, “Concurrency in Android” will provide you with a foundation in the Android system in relation to memory and threading. As in any other operating system, concurrency is hard to achieve.

Chapter 5, “Thread Safety” through Chapter 11, “Performance Considerations with Android Profiling Tools” examine common issues surrounding memory and threading, while indicating how the Android framework has evolved over time to grant developers more control around them. In tandem, these chapters show how Kotlin’s extensions and language features can help developers write better applications faster.

Chapter 12, “Trimming Down Resource Consumption with Performance Optimizations” explores the use of powerful Android developer tools to examine performance and memory-related analytics under the hood—to

be able to see things you never really knew about. This book will provide engineers with professionally developed and curated implementations of the most common tasks seen in native Android development. Many tasks will consist of a real-world problem, followed by the corresponding solution in both Java and Kotlin. When further explanation is required, the solutions will follow a snappy compare-and-contrast model with a focus on brevity and natural language.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values or by values determined by context.

TIP

This element signifies a tip or suggestion.

NOTE

This element signifies a general note.

WARNING

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/ProgrammingAndroidWithKotlin*.

If you have a technical question or a problem using the code examples, please send email to *bookquestions@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming Android with Kotlin* by Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, G. Blake Meike, and Mike Dunn (O'Reilly). Copyright 2022 Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, and O'Reilly Media, Inc., 978-1-492-06300-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O’Reilly Online Learning

NOTE

For more than 40 years, *O’Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O’Reilly’s online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O’Reilly and 200+ other publishers. For more information, visit *http://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O’Reilly Media, Inc.
- 1005 Gravenstein Highway North
- Sebastopol, CA 95472
- 800-998-9938 (in the United States or Canada)
- 707-829-0515 (international or local)
- 707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/pak*.

Email *bookquestions@oreilly.com* to comment or ask technical questions about this book.

For news and information about our books and courses, visit *http://oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://youtube.com/oreillymedia*

Acknowledgments

This book has been greatly strengthened and improved thanks to our technical reviewers, Adnan Sozuan and Andrew Gibel. We give thanks to the folks at O'Reilly for helping to bring us together and giving us all the support we needed to bring this book to life, especially Jeff Bleiel and Zan McQuade.

We thank Roman Elizarov and Jake Wharton for taking the time to speak with us about direction on evolution in concurrency in Kotlin and low-level optics in Android.

We thank our friends, family, and colleagues for support. We thank the Kotlin community, and the individuals who have taken the time to read early drafts and give feedback.

Lastly, we dedicate this book to Mike Dunn: coauthor, colleague, friend, and father. We miss him dearly and hope this book is something he'd be proud of.