THE COMPLETE GUIDE TO ANDROID 17 DEVELOPMENT

From Fundamentals to Production:
Kotlin, Compose, Architecture, and Beyond

- Modern Kotlin Development
- Jetpack Compose UI Toolkit
- Architecture & Best Practices
- Privacy, Security & Performance
- Testing, Publishing & Play Store

SCOTT LIONEL

The Complete Guide to Android 17 Development

From Fundamentals to Production: Kotlin, Compose, Architecture, and Beyond

Steve T. Publications

This book is available at
https://leanpub.com/thecompleteguidetoandroid17development

This version was published on 2026-07-13

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Steve T. Publications

Contents

From Fundamentals to Production: Kotlin, Compose, Architecture, and Beyond 1

Introduction: Why Android 17 Matters **2**
Who This Book Is For 2
What You Will Learn 2
How to Use This Book 3
Getting the Most Out of This Book 3

Chapter 1: The Android Platform — History, Architecture, and Where We Are Today **5**
From Android 1 to Android 17 — A Brief History 5
The Android Platform Architecture: HAL, Kernel, Runtime, and Framework 7
What Makes Android 17 Different: Key Themes and Design Philosophy 10
The Android Open Source Project and the Release Cadence 12
Device Fragmentation in 2026 and API Level Targeting 12
Chapter Summary 14
Exercises 14
Interview Questions 15

Chapter 2: Setting Up Your Development Environment **16**
Installing Android Studio Ladybug/Next Generation 16
Configuring the Android 17 SDK and System Images 17
Emulator Setup: Hardware Acceleration, Virtual Devices, and Cloud Testing 19
Physical Device Configuration: USB Debugging, ADB, and Fastboot . . 21
Project Templates and Your First "Hello World" 24
Chapter Summary 28
Exercises 28
Interview Questions 28

Chapter 3: Kotlin-First Development on Android **30**
Why Kotlin Won: History, Adoption, and Language Design 30
Kotlin Fundamentals for Android: Types, Null Safety, and Extension Functions 31
Coroutines and Structured Concurrency on Android 34
Kotlin Flow for Reactive Data Streams 38
Kotlin Multiplatform: Shared Business Logic 40
Chapter Summary 42
Exercises 42
Interview Questions 42

Chapter 4: Modern App Architecture **44**
The Evolution of Android Architecture: MVC to MVVM to MVI 44
ViewModel, Repository, and Use Case Patterns 44
Dependency Injection: Hilt and Koin 45
Navigation Component and Deep Linking 46
Clean Architecture on Android 17 46
Chapter Summary 47
Exercises 47
Interview Questions 47

Chapter 5: Jetpack Compose – Declarative UI for Android **48**
From XML Layouts to Compose: The Declarative Paradigm Shift . . . 48
Compose Fundamentals: Composables, State, and Recomposition . . . 48
Layouts, Modifiers, and Custom Components 49
Navigation in Compose: Single-Activity Architecture 50
Animations, Theming, and Material Design 3 50
Chapter Summary 51
Exercises 51
Interview Questions 51

Chapter 6: Android Views and Compose Interoperability **52**
The View System: Layouts, Adapters, and RecyclerView 52
Compose in Views: ComposeView and AndroidView 52
Views in Compose: AndroidViewBridge 53
Incremental Migration Strategies 53
When to Use Views vs Compose 54
Chapter Summary 54
Exercises 54

Interview Questions 54

Chapter 7: Lifecycle Management and Process Death 55
Activity Lifecycle: Creation, Resumption, Pausing, and Destruction . . 55
Service Lifecycles: Foreground, Background, and WorkManager 55
Configuration Changes and SavedStateHandle 56
Process Death and State Restoration 56
Lifecycle-Aware Components and LifecycleOwner 57
Chapter Summary 57
Exercises 57
Interview Questions 57

Chapter 8: Permissions, Security, and Privacy in Android 17 58
The Permission Model: Normal, Dangerous, and Special Permissions . 58
Runtime Permissions and Scoped Storage 58
Android 17 Privacy Enhancements: Photo Picker, Notification Permissions, and More 59
Security Best Practices: Keystore, Encryption, and Network Security Config 59
App Sandbox and Inter-Process Communication 60
Chapter Summary 60
Exercises 61
Interview Questions 61

Chapter 9: Background Processing and Battery Optimization 62
Background Execution Limits and Android 17 Restrictions 62
WorkManager: Deferrable, Guaranteed Background Work 62
Foreground Services and Foreground Service Types 63
JobScheduler and AlarmManager for Scheduled Tasks 63
Battery Optimization: Doze Mode, App Standby, and Power Profiles . 64
Chapter Summary 64
Exercises 64
Interview Questions 65

Chapter 10: Networking and Data Storage 66
HTTP Client Libraries: Retrofit, Ktor, and OkHttp 66
JSON Serialization: Kotlinx Serialization and Moshi 66
Room Database: Entities, DAOs, and Migrations 67
DataStore: Preferences and Proto-based Storage 67
Caching Strategies and Offline-First Architecture 68

Chapter Summary 68
Exercises 68
Interview Questions 68

Chapter 11: Media, Graphics, and Sensors 70
CameraX and the Modern Camera API 70
Audio Playback: ExoPlayer, Media3, and AudioFocus 70
Video Recording and Processing 70
Skia, Canvas, and Custom Drawing 71
Sensor APIs: Accelerometer, Gyroscope, Location, and Geofencing . . 71
Chapter Summary 72
Exercises 72
Interview Questions 72

Chapter 12: AI, On-Device ML, and Android 17 Intelligent Features . . . 73
TensorFlow Lite on Android: Model Loading and Inference 73
MediaPipe Tasks: Vision, Audio, and NLP 73
Neural Networks API (NNAPI) for Hardware Acceleration 74
On-Device LLMs and Gemini SDK Integration 74
Android 17 AI Features: Live Captions, Translation, and Smart Compose 74
Chapter Summary 75
Exercises 75
Interview Questions 75

Chapter 13: Performance Optimization and Profiling 76
Android Studio Profiler: CPU, Memory, and Network 76
Frame Rendering and Jank Detection with Choreographer 76
Memory Management: Leak Canary, Heap Analysis, and Bitmap Optimization 77
Startup Time Optimization and Splash Screen API 77
Benchmark Library and Macro/Micro Benchmarks 78
Chapter Summary 78
Exercises 78
Interview Questions 79

Chapter 14: Accessibility and Internationalization 80
Accessibility Fundamentals: Content Descriptions and Touch Targets 80
TalkBack, Switch Access, and Screen Magnification 80
Dynamic Type and Scalable UI 81
Internationalization: Locale, Number Formatting, and Date/Time APIs 81

Right-to-Left Layouts and Bidirectional Text 82
Chapter Summary 82
Exercises 82
Interview Questions 82

Chapter 15: Testing on Android 83
Unit Testing with JUnit 5 and MockK 83
Compose Testing: Semantics, Actions, and Assertions 83
Espresso and UI Automation 84
Continuous Integration and Cloud Testing (Firebase Test Lab) 84
Chapter Summary 84
Exercises 85
Interview Questions 85

Chapter 16: Debugging, Crashes, and Production Monitoring 86
Android Studio Debugger: Breakpoints, Stepping, and Evaluate Expression 86
Logcat Mastery: Filtering, Tags, and Remote Logging 86
Crash Reporting: Firebase Crashlytics and ACRA 87
ANR Analysis and Main Thread Violations 87
Remote Configuration and Feature Flags 87
Chapter Summary 88
Exercises 88
Interview Questions 88

Chapter 17: Publishing to the Play Store 89
App Signing: Upload Keys, Play App Signing, and Bundletool 89
Android App Bundles vs APKs 89
Play Console: Store Listing, Screenshots, and Metadata 90
Release Tracks: Internal, Alpha, Beta, and Production 90
Play Store Policies, Rating Requirements, and Content Guidelines . . . 90
Chapter Summary 91
Exercises 91
Interview Questions 91

Chapter 18: Migration Strategies and Backward Compatibility 92
API Level Targeting: minSdk, targetSdk, and compileSdk 92
Deprecated APIs in Android 17 and Their Replacements 92
Build Configuration: Flavors, Dimensions, and Dynamic Features . . . 93
Backward Compatibility Libraries: AndroidX and Jetpack 93

Migration Case Studies: From Android 12 to Android 17 93
Chapter Summary 94

Conclusion: Building for the Future **95**

References **96**

From Fundamentals to Production: Kotlin, Compose, Architecture, and Beyond

About this book: Android 17 "Cinnamon Bun" (API level 37) represents the latest evolution of Google's mobile platform, bringing significant changes to app architecture, privacy, performance, and developer tooling. This comprehensive guide walks you through everything from setting up your development environment to publishing production-quality apps on the Play Store. Whether you are a beginner learning Android for the first time or an experienced developer adapting to the latest APIs and behavior changes, this book provides the knowledge, code examples, and best practices you need to build modern, high-quality Android applications. Every chapter includes practical code samples, architectural patterns, troubleshooting guidance, and exercises designed to solidify your understanding.

Introduction: Why Android 17 Matters

In June 2026, Google released Android 17, codenamed "Cinnamon Bun," marking the seventeenth major release of the world's most widely deployed mobile operating system. For developers, this is not merely an incremental update. It is a platform that enforces stricter privacy boundaries, introduces new windowing capabilities, hardens security by default, and pushes the entire ecosystem toward more adaptive, performant, and intelligent applications.

If you have built Android apps before, you know the rhythm of the platform. Each year brings new SDK versions, deprecations, and behavioral shifts that require adaptation. Android 17 continues this tradition with particular emphasis on three themes: user privacy through permission refinement, developer productivity through improved tooling and APIs, and cross-device continuity through features like the Handoff API and mandatory large-screen adaptivity.

Who This Book Is For

This book is written for three audiences. First, if you are new to Android development, the early chapters will guide you from installing Android Studio to building your first Compose-based application. Second, if you are an intermediate developer comfortable with Kotlin and basic Android components, the middle chapters will deepen your understanding of architecture, testing, performance optimization, and modern patterns. Third, if you are a senior developer preparing your existing codebase for Android 17, the later chapters provide detailed migration strategies, behavior change analysis, and new API adoption guidance.

What You Will Learn

By the end of this book, you will be able to:

- Set up a complete Android 17 development environment with Android Studio Quail, the latest SDK, and testing infrastructure

- Write idiomatic Kotlin code using coroutines, Flow, and modern language features
- Build adaptive user interfaces with Jetpack Compose following Material Design 3 guidelines
- Architect applications using recommended patterns: MVVM, MVI, Clean Architecture, and dependency injection with Hilt
- Manage the Android component lifecycle correctly and handle process death gracefully
- Implement privacy-first permissions, scoped storage, and security best practices
- Schedule reliable background work with WorkManager and respect battery optimization constraints
- Connect to networks, persist data locally, and build offline-first applications
- Integrate on-device machine learning using LiteRT and MediaPipe
- Profile, optimize, and benchmark your application for performance and battery efficiency
- Make your app accessible to all users through TalkBack, dynamic type, and inclusive design
- Write comprehensive unit, instrumentation, and UI tests
- Publish to the Google Play Store with proper signing, bundling, and compliance

How to Use This Book

The chapters are organized to build on each other progressively. Chapters 1 through 3 establish the foundation: platform architecture, development environment setup, and Kotlin fundamentals. Chapters 4 through 7 cover application structure and UI development. Chapters 8 through 12 address system integration, data, and media. Chapters 13 through 16 focus on quality assurance, performance, and production readiness. The final chapters cover publishing and migration strategies.

Each chapter contains section headings that organize the material into digestible topics. Code examples are provided throughout, marked clearly with language labels. Where a concept benefits from visual explanation, ASCII diagrams or structured tables are included. End-of-chapter exercises give you hands-on practice, and interview questions help you prepare for technical discussions.

Getting the Most Out of This Book

For the best results, follow along with the code examples using Android Studio Quail (2026.1.x) or later, targeting Android 17 (API level 37). The code samples use Kotlin as the primary language, following Google's current recommendations. Where Java equivalents are relevant, they are noted.

You can find all official documentation referenced in this book at the Android Developers site (developer.android.com), and source code for examples is available through the companion repository noted in the references section.

Now let us begin.

Chapter 1: The Android Platform — History, Architecture, and Where We Are Today

Android is not simply an operating system. It is a complete software stack built on Linux, designed for diversity across form factors from smartphones to tablets, foldables, watches, automotive displays, and XR headsets. Understanding the platform's architecture and evolution is essential context for every Android developer. This chapter traces Android's history, explains its layered architecture, examines what makes Android 17 significant, and discusses the realities of device fragmentation.

From Android 1 to Android 17 — A Brief History

Android was conceived in 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White. Google acquired the company in 2005, and the first Android SDK was released in November 2007. The T-Mobile G1, also known as the HTC Dream, became the first commercial Android device in October 2008, running Android 1.0 with API level 1.

The early years moved fast. Android 1.5 "Cupcake" (API level 3, 2009) introduced on-screen soft keyboards and widgets. Android 2.0/2.1 "Donut" (API level 7, 2009) added the Searchable widget and Android Market (later Google Play). Android 2.2 "Froyo" (API level 8, 2010) brought JIT compilation, Bluetooth sharing, and VoIP support. Android 2.3 "Gingerbread" (API level 9-10, 2010-2011) improved touch responsiveness and added copy-and-paste.

The platform matured dramatically with Android 4.x "Ice Cream Sandwich" through "KitKat" (API levels 14-19, 2011-2013). Holo design language arrived, multi-window support was introduced, and the ART runtime was added as an experimental alternative to Dalvik. Android 5.0 "Lollipop" (API level 21, 2014) made ART the default runtime, introduced Material Design, and brought significant performance improvements through AOT compilation.

Android 6.0 “Marshmallow” (API level 23, 2015) revolutionized permissions with runtime permission requests, replacing the all-or-nothing install-time model. Android 7.0 “Nougat” (API level 24-25, 2016) introduced multi-window support and ART improvements. Android 8.0 “Oreo” (API level 26, 2017) brought background execution limits, notification channels, and Picture-in-Picture mode.

The pace accelerated. Android 9 “Pie” (API level 28, 2018) introduced neural networks API and adaptive battery. Android 10 (API level 29, 2019) brought scoped storage, dark theme, and on-device intelligence. Android 11 (API level 30, 2020) added bubble conversations, device pairing flows, and one-handed mode. Android 12 (API level 31, 2021) introduced Material You with dynamic color, a new motion system, and precise background location access. Android 13 (API level 33, 2022) refined permissions further with granular media permissions and notification permission. Android 14 (API level 34, 2023) brought foreground service types, health connect, and battery-saving standby buckets. Android 15 (API level 35, 2024) introduced per-app language settings, predictive back animation, and additional privacy controls. Android 16 (API level 36, 2025) added the permission controller API, eye dropper enhancements, and large screen adaptivity improvements.

Now we arrive at Android 17 “Cinnamon Bun” (API level 37, released June 16, 2026), which continues this trajectory of platform evolution with mandatory large-screen adaptivity, enhanced privacy protections, new cross-device features, and significant performance improvements.

The following table summarizes the version history:

Version	API Level	Codename	Year	Key Innovation
1.0	1	—	2008	First release
1.5	3	Cupcake	2009	Soft keyboard, widgets
4.0	14-15	Ice Cream Sandwich	2011	Holo design, multi-touch
5.0	21	Lollipop	2014	ART runtime, Material Design
6.0	23	Marshmallow	2015	Runtime permissions

Version	API Level	Codename	Year	Key Innovation
7.0	24-25	Nougat	2016	Multi-window, ART improvements
8.0	26	Oreo	2017	Background limits, notification channels
9	28	Pie	2018	NNAPI, adaptive battery
10	29	—	2019	Scoped storage, dark theme
11	30	—	2020	Bubbles, device pairing
12	31	—	2021	Material You, precise location
13	33	—	2022	Granular media permissions
14	34	—	2023	Foreground service types
15	35	—	2024	Per-app language, predictive back
16	36	Baklava	2025	Permission controller, large screen adaptivity
17	37	Cinnamon Bun	2026	Mandatory adaptivity, Handoff API, memory limits

Source: Android version history documentation and release notes [1, 2].

The Android Platform Architecture: HAL, Kernel, Runtime, and Framework

Android's software stack is organized into five layers, each building on the one below it. Understanding these layers helps you understand where your application code fits in the broader system.

Linux Kernel

At the foundation sits the Linux kernel, which provides core operating system services. The kernel handles process scheduling, memory management, networking, device drivers, and power management. Android uses a subset of Linux kernel features and adds its own components like the wakelock mechanism for power management and the Binder inter-process communication (IPC) mechanism for efficient cross-process calls.

The kernel layer also includes hardware-specific drivers that communicate directly with device components: display, camera, touchscreen, audio, Bluetooth, Wi-Fi, GPS, and more. These drivers are essential because they translate between the abstract interfaces expected by the upper layers and the concrete electrical signals required by physical hardware.

Hardware Abstraction Layer (HAL)

The HAL sits between the kernel's device drivers and the Android framework. It provides standard interfaces that the Java/Kotlin API framework can use to access hardware capabilities without knowing the specifics of any particular device. The HAL consists of modular library files, typically implementing interfaces for camera, audio, Bluetooth, sensors, and graphics.

When your application calls a framework API like `CameraManager.openCamera()`, the request flows through the Java framework, crosses into native code via JNI, reaches the appropriate HAL module, and eventually communicates with the kernel driver. This layered approach means that the same Android application can run on devices from completely different manufacturers, as long as each manufacturer provides proper HAL implementations for their hardware.

Android Runtime (ART)

The Android Runtime is one of the most critical components for application developers. ART executes your compiled application code and provides the runtime environment in which your app runs. Key aspects include:

- **Process isolation**: Each Android app runs in its own Linux process with its own ART instance, providing security sandboxing.
- **DEX format**: Build tools compile Java and Kotlin source code into Dalvik Executable (DEX) bytecode, a format optimized for minimal memory footprint on mobile devices.
- **AOT and JIT compilation**: ART uses ahead-of-time (AOT) compilation during app installation to convert DEX to native machine code, and just-in-time (JIT) compilation at runtime to profile and optimize frequently executed code paths.
- **Generational garbage collection**: Android 17 introduces generational GC in ART's Concurrent Mark-Compact collector, prioritizing frequent, low-cost "young generation" collections for improved performance [3].
- **Core libraries**: ART bundles core Java libraries that provide most standard functionality, including select Java 8 features used by the Android framework.

Native C/C++ Libraries

Beneath the Java/Kotlin framework layer, Android relies on native libraries written in C and C++. These include:

- **SurfaceFlinger**: Composes and manages all visual surfaces on the display
- **Media Framework**: Handles audio and video encoding/decoding
- **SQLite**: Embedded relational database engine
- **OpenGL ES / Vulkan**: 2D and 3D graphics rendering
- **WebGPU**: New in Android 17, providing Kotlin and Java graphics APIs backed by Vulkan [4]
- **libc, libm, libz**: Standard C library, math library, and compression library

Developers who need native code performance can use the Android NDK to write C/C++ libraries that are loaded into their applications via `System.loadLibrary()`. Android 17 extends Safer Dynamic Code Loading (DCL) protection to native libraries, requiring all files loaded via `System.load()` to be marked read-only [5].

Java API Framework

The framework layer provides the APIs that developers interact with most directly. Key components include:

- **Activity Manager**: Manages application lifecycles, navigation, and the back stack
- **Window Manager**: Handles window layout, input events, and display composition
- **Content Providers**: Enable data sharing between applications
- **Resource Manager**: Provides access to non-code resources like strings, layouts, drawables, and animations
- **Notification Manager**: Enables apps to post status bar notifications
- **View System**: The traditional UI toolkit for building interfaces with XML layouts and View classes
- **Jetpack Libraries**: Modern, backward-compatible components that extend the framework with features like Lifecycle, ViewModel, Room, WorkManager, Navigation, and Compose

System Applications

At the top of the stack are the system applications: Launcher, Settings, Phone, Contacts, Messaging, and others. These apps have no special privileges over third-party applications (with a few exceptions like Settings). Users can replace most system defaults with third-party alternatives.

What Makes Android 17 Different: Key Themes and Design Philosophy

Android 17 follows three major design themes that distinguish it from previous releases:

Theme 1: Adaptive by Default

Android 16 introduced platform API changes to ignore orientation, resizability, and aspect ratio constraints on large screens (smallest width >= 600dp).

Android 17 removes the developer opt-out entirely. Apps targeting API level 37 can no longer use `screenOrientation`, `resizeableActivity`, `minAspectRatio`, or `maxAspectRatio` to restrict their behavior on tablets, foldables, and desktop-mode devices [6]. This forces developers to build truly adaptive layouts that respond to available screen space.

Additionally, Android 17 optimizes configuration changes. The system no longer restarts Activities by default for keyboard availability, navigation method, touchscreen type, color mode, and desk UI mode transitions. This reduces jank and preserves application state during common hardware events [7].

Theme 2: Privacy Hardening

Android 17 introduces several privacy-first changes:

- **ACCESS_LOCAL_NETWORK permission**: Apps targeting API 37 must request this runtime permission (part of the NEARBY_DEVICES group) to access local network devices. This was opt-in in Android 16 and is now mandatory [8].
- **Contacts Picker**: A new system-level picker allows apps to request specific contact fields without needing the broad READ_CONTACTS permission. Access is session-based and user-controlled [9].
- **OTP Protection**: Standard SMS messages containing one-time passwords are delayed by three hours for most apps, preventing OTP hijacking. Developers should migrate to SMS Retriever or SMS User Consent APIs [10].
- **ECH (Encrypted Client Hello)**: TLS connections now use ECH by default, encrypting the Server Name Indication in the handshake to prevent network observers from identifying target domains [11].

Theme 3: Performance and Reliability

Android 17 brings significant runtime improvements:

- **Lock-free MessageQueue**: A new implementation reduces contention and missed frames for apps targeting API 37 [12].

- **Generational GC**: ART's garbage collector now distinguishes between young and old objects, performing cheaper collections more frequently [13].
- **App Memory Limits**: The system enforces RAM-based memory limits per app, with `ApplicationExitInfo.getDescription()` returning "MemoryLimiter" when an app is killed [14].
- **New Profiling Triggers**: ProfilingManager adds triggers for cold start, OOM, excessive CPU usage, and anomaly detection, making it easier to diagnose performance issues in production [15].

The Android Open Source Project and the Release Cadence

Android is primarily developed through the Android Open Source Project (AOSP), which publishes its source code under Apache 2.0 and other permissive licenses. Google leads development, but manufacturers, carriers, and independent contributors all participate.

The release cadence has accelerated in recent years. Android 17 followed this schedule:

- **November 2025**: Developer Preview (Canary) builds available for early testing
- **February 2026**: Beta 1 released, suitable for development and general use
- **February–March 2026**: Beta 2 with Platform Stability milestone (final APIs and behaviors)
- **March 2026**: Beta 3 with API surface locked
- **April 2026**: Beta 4, the final scheduled beta
- **May 12, 2026**: Official announcement at The Android Show
- **June 16, 2026**: Stable release on supported Pixel devices
- **Q4 2026 (planned)**: Minor SDK release with additional API extensions

Quarterly Platform Releases (QPRs) provide incremental updates between major releases, delivering bug fixes, security patches, and minor feature additions to devices that have already received the latest major Android version.

Device Fragmentation in 2026 and API Level Targeting

Despite Android's growth, fragmentation remains a reality. As of mid-2026, the installed base spans roughly twelve major Android versions, from Android 10 (API 29) through Android 17 (API 37). This diversity means that developers must make deliberate choices about which API levels to support.

Understanding the Three API Level Attributes

Every Android application declares three key API level attributes in its manifest:

- **minSdk**: The minimum API level your app supports. Devices running older versions cannot install your app.
- **targetSdk**: The highest API level you have tested against. This determines which behavioral changes apply to your app.
- **compileSdk**: The API level used by the compiler to check your code. This should generally match or exceed targetSdk.

Google Play Requirements

As of August 31, 2026, all new apps and app updates submitted to Google Play must target Android 16 (API level 36) or higher [16]. This means that for any app submitted to the Play Store in the second half of 2026, targetSdk must be at least 36. Targeting API level 37 (Android 17) is strongly recommended to take advantage of new features and ensure your app behaves correctly on the latest devices.

Choosing Your minSdk

The choice of minimum SDK level involves trade-offs. A lower minSdk reaches more users but requires more compatibility code and testing. A higher minSdk simplifies development but excludes older devices. Current guidance suggests:

- For new apps in 2026, a minSdk of 24 (Android 7.0) or higher is reasonable, as Android 7.0+ covers the vast majority of active devices

- For apps with heavy use of modern APIs like Compose, Room, or WorkManager, the effective minimum may be higher due to library dependencies
- Always check your target audience's device distribution using the Google Play Console dashboard

The Fragmentation Strategy

The recommended approach is to set your minSdk as low as your business requirements allow, while setting targetSdk to the latest stable release. Use AndroidX libraries for backward-compatible features, runtime API checks for newer capabilities, and feature flags to gracefully degrade functionality on older devices.

Chapter Summary

Android 17 represents a platform that is more adaptive, more private, and more performant than ever before. Its mandatory large-screen support forces developers to think about responsive layouts. Its privacy enhancements require careful permission management. Its performance improvements provide new tools for optimization and debugging.

Understanding the Android platform architecture helps you write better code because you know where your application fits in the broader system. Knowing the release cadence helps you plan your development timeline. And understanding fragmentation helps you make informed decisions about API level targeting.

In the next chapter, we will set up your complete development environment, install the latest tools, and create your first Android 17 project.

Exercises

1. Research the Android version installed on three different devices you have access to (or can find information about). Note their API levels and screen sizes. What patterns do you observe?

2. Read the official Android 17 behavior changes documentation at developer.android.com/about/versions/17/behavior-changes-17. Identify three changes that would affect a typical application and describe how you would adapt your code.
3. Draw the Android platform architecture diagram from memory, labeling each layer and describing one key component in each layer.

Interview Questions

1. What are the five layers of the Android platform architecture, and what is the role of each?
2. How does ART differ from the original Dalvik runtime, and what improvements did Android 17 bring to garbage collection?
3. What is the difference between minSdk, targetSdk, and compileSdk, and why does each matter?
4. Why does Google mandate a minimum targetSdk for Play Store submissions, and how does this affect app behavior?
5. Describe the Android 17 release schedule. What is the significance of the Platform Stability milestone?

Chapter 2: Setting Up Your Development Environment

Before you can write a single line of Android code, you need a properly configured development environment. This chapter walks through installing Android Studio, configuring the SDK, setting up emulators and physical devices, and creating your first project. A solid foundation here saves hours of debugging later.

Installing Android Studio Ladybug/Next Generation

Android Studio is Google's official integrated development environment (IDE) for Android development. Built on JetBrains IntelliJ IDEA Community Edition, it provides code editing, debugging, profiling, testing, and deployment tools all in one package.

Current Version: Android Studio Quail

As of mid-2026, the stable release is Android Studio Quail version 2026.1.x. The preview channel offers Android Studio Quail 3 (2026.1.3 Canary 3) with experimental features for early adopters. You can install both channels side by side on the same machine, as they use separate configuration directories.

System Requirements

Android Studio has substantial system requirements due to the Gradle build system, emulator, and profiling tools:

- **Operating System**: Windows 10 or later (64-bit), macOS 13 Ventura or later, or Linux (64-bit, glibc 2.17+)
- **RAM**: Minimum 8 GB recommended, 16 GB or more for comfortable development

- **Disk Space**: At least 8 GB for Android Studio itself, plus 20 GB or more for SDK packages and system images
- **SSD**: Strongly recommended for faster builds and emulator performance
- **Display Resolution**: Minimum 1280 x 800 pixels

Installation Steps

1. Download Android Studio from developer.android.com/studio
2. Run the installer and follow the setup wizard
3. During installation, choose the "Standard" installation type to include the Android SDK, emulator, and performance profiling tools
4. The Android SDK Manager will launch after installation. Accept the licenses for all required components
5. Restart Android Studio when prompted

First Launch Configuration

On first launch, Android Studio may prompt you to:

- Import settings from a previous version (skip if this is your first installation)
- Choose a UI theme (Dark or Light)
- Install additional plugins
- Configure proxy settings if behind a corporate firewall

Configuring the Android 17 SDK and System Images

The Android Software Development Kit (SDK) provides the APIs, tools, and libraries you need to build Android applications. The SDK Manager within Android Studio handles installation and updates.

Installing the Android 17 SDK Platform

Open the SDK Manager (Tools > SDK Manager) and navigate to the SDK Platforms tab. Check "Android 17.0 (Cinnamon Bun)" with API level 37, then click Apply. This installs:

- **Android SDK Platform 37**: The core Android framework APIs
- **Google APIs Intel x86_64 Atom System Image**: For emulator use
- **Google Play Intel x86_64 Atom System Image**: Includes Google Play Services
- **Android Emulator**: The virtual device runner

SDK Tools

In the SDK Tools tab, ensure the following are installed:

- **Android SDK Build-Tools** (latest version): Contains `aapt`, `dx`, `d8`, `apksigner`, and other build tools
- **Android SDK Platform-Tools**: Contains `adb` (Android Debug Bridge) and `fastboot`
- **Android Emulator**: Latest version for virtual device support
- **Android SDK Command-line Tools**: For command-line builds and CI/CD pipelines
- **NDK (Side by side)**: If you plan to use native C/C++ code
- **Performance (Intel HAXM)**: Hardware-accelerated execution manager for emulators on Windows

Configuring SDK Paths

The default SDK installation path varies by operating system:

OS	Default Path
Windows	`C:\Users\<username>\AppData\Local\Android\Sdk`
macOS	`~/Library/Android/sdk`

OS	Default Path
Linux	`~/Android/Sdk`

You can customize this path in Android Studio settings (Appearance & Behavior > System Settings > Android SDK). The `ANDROID_HOME` environment variable should point to your SDK directory for command-line tools and CI systems.

Verifying Your Installation

Open a terminal and run:

```
adb version
```

This should return the ADB version information. You can also verify the SDK platform by listing available platforms:

```
sdkmanager --list
```

Emulator Setup: Hardware Acceleration, Virtual Devices, and Cloud Testing

The Android Emulator provides a virtual device that runs on your development machine. It is essential for testing across different screen sizes, Android versions, and hardware configurations without physical devices.

Creating a Virtual Device

Open the Device Manager (Tools > Device Manager) and click "Create Device." The wizard guides you through:

1. **Hardware Selection**: Choose from predefined device definitions (Pixel 9, Pixel Fold, Galaxy S series, etc.) or create a custom profile

2. **System Image Selection**: Choose an Android version with Google APIs or Play Store support. For Android 17 development, select the API level 37 image
3. **Verification**: The wizard checks for hardware acceleration support and provides recommendations
4. **Naming**: Give your virtual device a descriptive name

Hardware Acceleration

Emulator performance depends heavily on hardware acceleration. The following technologies provide GPU and CPU acceleration:

- **Intel HAXM** (Hardware Accelerated Execution Manager): For Intel processors on Windows and macOS
- **Windows Hypervisor Platform**: For Windows 10/11 with Hyper-V enabled
- **Apple Silicon Native Emulation**: On M1/M2/M3 Macs, the emulator runs natively in ARM mode
- **KVM** (Kernel-based Virtual Machine): For Linux systems

To verify hardware acceleration is working, check the emulator's performance monitor. A well-accelerated emulator should achieve 60 frames per second for standard UI rendering.

Emulator Performance Tips

- Allocate at least 2 GB of RAM and 2 CPU cores to your virtual device
- Use the x86_64 system images rather than ARM images (unless testing specific ARM behavior)
- Enable "Host GPU" acceleration in the advanced settings
- Use cold boot for initial testing, then switch to quick boot for faster restarts
- Snapshot the emulator state after setup to avoid reconfiguring on each launch

Essential Emulator Commands

The emulator supports several useful command-line options:

```
# List running emulators
emulator -list-avds

# Start a specific virtual device
emulator -avd Pixel_9_API_37

# Take a screenshot
adb shell screencap -p /sdcard/screenshot.png
adb pull /sdcard/screenshot.png

# Simulate phone call
adb emu gsm call 5556

# Simulate GPS location
adb emu geo fix -122.39 37.70
```

Cloud Testing Alternatives

For comprehensive device testing, consider cloud-based solutions:

- **Firebase Test Lab**: Google's cloud testing service that runs your tests on hundreds of physical devices
- **BrowserStack App Automate**: Cross-device testing with real and virtual devices
- **Sauce Labs**: Automated mobile testing across platforms

These services are particularly valuable for testing on device models you do not own physically.

Physical Device Configuration: USB Debugging, ADB, and Fastboot

While emulators are convenient, physical devices provide the most realistic testing environment. They reveal real-world performance characteristics, hardware-specific bugs, and network conditions that emulation cannot fully replicate.

Enabling Developer Options

On your Android device:

1. Open Settings > About phone
2. Tap “Build number” seven times to enable Developer options
3. Return to Settings > System > Developer options
4. Enable “USB debugging”
5. Optionally enable “USB debugging (Security settings)” for automated permission grants

Connecting via USB

1. Connect your device to your development machine using a USB cable
2. On the device, you may see a prompt asking to allow USB debugging from this computer. Check “Always allow” and tap OK
3. Verify the connection by running `adb devices` in a terminal

```
$ adb devices
List of devices attached
ABC123DEF456    device
```

If your device shows as “unauthorized,” check the USB debugging dialog on the device. If it does not appear at all, try a different USB cable or port.

ADB (Android Debug Bridge) Essentials

ADB is the Swiss Army knife of Android development. Here are the most commonly used commands:

```
# Install an APK
adb install app-debug.apk

# Uninstall an app
adb uninstall com.example.myapp

# Clear app data
adb shell pm clear com.example.myapp

# View logcat output
adb logcat

# Filter logcat by tag
adb logcat -s ActivityManager

# Forward a port (useful for debugging web servers)
adb forward tcp:8080 tcp:8080

# Reboot the device
adb reboot

# Reboot into recovery mode
adb reboot recovery

# Pull files from the device
adb pull /sdcard/Download/file.txt ./

# Push files to the device
adb push myfile.txt /sdcard/Download/

# Screenshot
adb shell screencap -p /sdcard/screen.png
adb pull /sdcard/screen.png

# Screen recording
adb shell screenrecord /sdcard/recording.mp4
adb pull /sdcard/recording.mp4
```

Fastboot for Bootloader Operations

Fastboot communicates with devices in bootloader mode. Common uses include:

```
# Enter fastboot mode from adb
adb reboot bootloader

# Flash a custom recovery
fastboot flash recovery recovery.img

# Flash a system image
fastboot flash boot boot.img

# Reboot from fastboot
fastboot reboot
```

Fastboot is primarily used for flashing factory images, installing custom ROMs, and unlocking bootloaders. Use it cautiously, as incorrect commands can brick your device.

Project Templates and Your First "Hello World"

With your environment configured, it is time to create your first Android 17 project.

Creating a New Project

1. Open Android Studio and select "New Project"
2. Choose the "Empty Activity" template (for Compose-based projects) or "Empty Views Activity" (for traditional View-based projects)
3. Configure the project:

    - **Name**: MyFirstApp
    - **Package name**: com.example.myfirstapp
    - **Save location**: Your preferred project directory
    - **Language**: Kotlin
    - **Minimum SDK**: API 24 (Android 7.0) or higher
    - **Build Configuration Language**: Prefer Kotlin DSL (`build.gradle.kts`) over Groovy

4. Click "Finish" and wait for Gradle to sync

Understanding the Project Structure

A new Android project follows this structure:

```
MyFirstApp/
├── app/
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/com/example/myfirstapp/
│   │   │   │   ├── MainActivity.kt
│   │   │   │   └── MyApplication.kt
│   │   │   ├── res/
│   │   │   │   ├── drawable/
│   │   │   │   ├── mipmap-*/
│   │   │   │   ├── values/
│   │   │   │   │   ├── strings.xml
│   │   │   │   │   ├── themes.xml
│   │   │   │   │   └── colors.xml
│   │   │   │   └── xml/
│   │   │   └── AndroidManifest.xml
│   │   ├── androidTest/
│   │   └── test/
│   ├── build.gradle.kts
│   └── proguard-rules.pro
├── build.gradle.kts          (project-level)
├── settings.gradle.kts
├── gradle.properties
└── local.properties
```

Key files explained:

- **MainActivity.kt**: Your main entry point Activity, containing the `onCreate()` method
- **AndroidManifest.xml**: Declares app components, permissions, and configuration
- **build.gradle.kts** (app-level): App-specific dependencies and build configuration
- **build.gradle.kts** (project-level): Project-wide plugins and repositories
- **settings.gradle.kts**: Module inclusion and plugin management
- **local.properties**: Local SDK path (not committed to version control)

The Empty Compose Activity Template

The default "Empty Activity" template creates a Compose-based application. Here is what the generated `MainActivity.kt` looks like:

```
package com.example.myfirstapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.myfirstapp.ui.theme.MyFirstAppTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            MyFirstAppTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android 17",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello, $name!",
        modifier = modifier
    )
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    MyFirstAppTheme {
        Greeting("Android")
    }
}
```

This template demonstrates several key concepts:

- `ComponentActivity`: The base class for activities using Jetpack components
- `setContent {}`: Sets the Compose UI hierarchy for the activity
- `enableEdgeToEdge()`: Enables edge-to-edge layout with system bar transparency
- `Scaffold`: A layout component that implements the basic Material Design visual layout structure
- `@Preview`: Annotation that enables instant preview in Android Studio's layout editor

Building and Running

1. Select a target device (emulator or physical) from the device selector toolbar
2. Click the green "Run" button (Shift + F10) or select Run > Run 'app'
3. Gradle builds the application, packages it as an APK, installs it on the selected device, and launches the main activity

The first build may take a minute or two as Gradle downloads dependencies and compiles the project. Subsequent builds are significantly faster.

Troubleshooting Common Setup Issues

Problem	Solution
"SDK not found" error	Verify ANDROID_HOME points to your SDK directory in File > Project Structure > SDK Location
Emulator won't start	Check hardware acceleration is enabled; try creating a new virtual device with x86_64 image
Device not detected by ADB	Enable USB debugging, try different USB cable/port, install OEM USB drivers (Windows)

Problem	Solution
Gradle sync fails	Check internet connection; try File > Invalidate Caches / Restart
"Build tools revision too old"	Update Build-Tools in SDK Manager to the latest version
Compose preview not rendering	Enable "Enable Jetpack Compose Preview" in Settings > Languages & Frameworks > Kotlin > Compiler

Chapter Summary

Setting up your development environment is an investment that pays dividends throughout your Android development journey. Android Studio Quail provides a comprehensive IDE with the Android 17 SDK, emulators for testing across configurations, and powerful debugging tools. Physical device testing through ADB ensures your app works correctly on real hardware. The project template gives you a solid starting point with modern Compose-based architecture.

In the next chapter, we dive into Kotlin, the primary language for Android development, covering the features and patterns that make it ideal for building Android applications.

Exercises

1. Install Android Studio Quail (or the latest stable version available) on your machine. Verify the installation by checking the "About" dialog.
2. Create a virtual device running Android 17 (API level 37) with Google APIs. Launch it and verify hardware acceleration is working by checking the performance monitor.
3. Connect a physical Android device via USB and verify the connection using `adb devices`. Take a screenshot using ADB commands.
4. Create a new "Empty Activity" project targeting API level 37. Modify the greeting text to display your name, build the project, and run it on both an emulator and a physical device.

Interview Questions

1. What is the difference between the SDK Platform, SDK Build-Tools, and SDK Platform-Tools?
2. How do you enable hardware acceleration for the Android Emulator on different operating systems?
3. What is the purpose of the `adb` tool, and name five commonly used ADB commands.
4. Why is the AndroidManifest.xml file essential, and what are three things it declares?
5. What is the difference between the “Empty Activity” template and the “Empty Views Activity” template?

Chapter 3: Kotlin-First Development on Android

Kotlin is the official preferred language for Android development. Google announced Kotlin-first status at Google I/O 2019, and today every new template, library, and documentation example is written in Kotlin first. This chapter covers the Kotlin features that matter most for Android development, from basic syntax to advanced concurrency patterns.

Why Kotlin Won: History, Adoption, and Language Design

Kotlin was created by JetBrains and first released in 2011. It was designed from the start to be fully interoperable with Java, run on the JVM, and address the pain points developers experienced when writing Java code. Key design goals included null safety, conciseness without sacrificing readability, and functional programming support.

The Android Adoption Story

Google's endorsement at I/O 2019 was the catalyst. Within two years, the majority of new Android projects were Kotlin-first. By 2024, over 90 percent of top Google Play apps used Kotlin extensively. The language became so deeply integrated into the Android ecosystem that many Jetpack libraries provide Kotlin-specific APIs with features like extension functions, coroutines support, and type-safe builders that have no Java equivalent.

Why Kotlin Over Java for Android

- **Null Safety**: Kotlin's type system distinguishes between nullable and non-nullable types at compile time, eliminating the dreaded NullPointerException as a category of bugs

- **Conciseness**: Data classes, extension functions, string templates, and smart casts reduce boilerplate significantly
- **Coroutines**: Structured concurrency provides an elegant solution to asynchronous programming without callback hell
- **Interoperability**: Kotlin calls Java code seamlessly and vice versa, making migration incremental
- **Modern Language Features**: Pattern matching (when expressions), sealed classes, inline functions with reified type parameters, and delegation

Kotlin Fundamentals for Android: Types, Null Safety, and Extension Functions

Type System and Null Safety

Kotlin's type system is the language's most important safety feature. Every type in Kotlin is non-nullable by default:

```
// This variable cannot hold null
var name: String = "Android"
name = null  // Compile error: null can not be a value of a non-null type String

// To allow null, explicitly mark the type as nullable
var description: String? = null
description = "A mobile OS"
```

When working with nullable types, Kotlin forces you to handle the null case:

```
fun greet(name: String?) {
    // Option 1: Safe call operator
    val length = name?.length  // Returns Int? (nullable)

    // Option 2: Elvis operator for default values
    val safeLength = name?.length ?: 0

    // Option 3: Not-null assertion (use with caution)
    val asserted = name!!  // Throws NullPointerException if name is null

    // Option 4: Safe cast
    val str = name as? String  // Returns null instead of throwing on failed cast
}
```

Smart Casts

Kotlin's compiler tracks type checks and automatically casts variables when it can prove the type is safe:

```
fun formatValue(value: Any): String {
    if (value is String) {
        // value is automatically cast to String here
        return value.uppercase()
    }
    if (value is Int) {
        return "Number: $value"
    }
    return "Unknown"
}
```

Data Classes

Data classes reduce boilerplate for classes that primarily hold data:

```
data class User(
    val id: Long,
    val name: String,
    val email: String?,
    val isActive: Boolean = true
)

val user = User(1, "Alice", "alice@example.com")
val copy = user.copy(email = "newalice@example.com")  // Easy immutable updates
println(user.component1())  // Destructuring: 1
val (id, name, email, active) = user  // Destructuring declaration
```

Data classes automatically generate `equals()`, `hashCode()`, `toString()`, `copy()`, and `componentN()` functions.

Extension Functions

Extension functions let you add methods to existing classes without modifying their source code:

```kotlin
// Extend Context with a convenience function
fun Context.toast(message: String, duration: Int = Toast.LENGTH_SHORT) {
    Toast.makeText(this, message, duration).show()
}

// Use it in an Activity
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        toast("Hello Android 17!")  // No need for Toast.makeText()
    }
}

// Extension properties
val Activity.screenWidth: Int
    get() = windowManager.defaultDisplay.width

val Activity.screenHeight: Int
    get() = windowManager.defaultDisplay.height
```

When Expressions

Kotlin's when expression replaces Java's switch statement with a more powerful construct:

```kotlin
fun describe(apiLevel: Int): String = when (apiLevel) {
    37 -> "Android 17 - Cinnamon Bun"
    36 -> "Android 16 - Baklava"
    in 30..35 -> "Android 11-15"
    in 21..29 -> "Android 5.0-9"
    else -> "Legacy Android"
}

// When without argument (like if-else chain)
fun parseInput(input: Any): String = when {
    input is String -> "String: ${input.length} chars"
    input is Int -> "Integer: $input"
    input.isEmpty() -> "Empty collection"
    else -> "Unknown type"
}
```

Scope Functions

Kotlin provides five scope functions that control how you reference the receiver object and the return value:

Function	Receiver	Return Value	Use Case
`let`	`it`	Lambda result	Null-checking and transformations
`run`	`this`	Lambda result	Configuration without null check
`with`	`this`	Lambda result	Non-member function, configuration
`apply`	`this`	Object itself	Object initialization and setup
`also`	`it`	Object itself	Side effects and logging

```kotlin
// let: null-safe transformation
val length = userName?.let { name ->
    name.length
}

// apply: object configuration
val textView = TextView(this).apply {
    text = "Hello"
    textSize = 16f
    setPadding(16, 16, 16, 16)
}

// also: side effects
imageView.also { view ->
    Log.d("ImageDebug", "Setting image on $view")
}.setImageBitmap(bitmap)
```

Coroutines and Structured Concurrency on Android

Asynchronous programming is fundamental to Android development. Network requests, database queries, file I/O, and image loading all happen off the main thread. Kotlin coroutines provide an elegant solution.

Understanding Coroutines

A coroutine is a lightweight thread of execution that can be suspended and resumed without blocking the underlying OS thread. This makes them ideal

for Android, where you must keep the main (UI) thread responsive while performing background work.

```
import kotlinx.coroutines.*
import kotlinx.coroutines.flow.*

class MainActivity : AppCompatActivity() {
    private val viewModel: MainViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // ...
    }
}

class MainViewModel : ViewModel() {
    private val _uiState = MutableStateFlow<UiState>(UiState.Loading)
    val uiState: StateFlow<UiState> = _uiState

    init {
        loadData()
    }

    fun loadData() {
        viewModelScope.launch {
            _uiState.value = UiState.Loading
            try {
                val data = withContext(Dispatchers.IO) {
                    repository.fetchData()  // Runs on IO dispatcher
                }
                _uiState.value = UiState.Success(data)
            } catch (e: Exception) {
                _uiState.value = UiState.Error(e.message ?: "Unknown error")
            }
        }
    }
}

sealed interface UiState {
    object Loading : UiState
    data class Success(val data: List<Item>) : UiState
    data class Error(val message: String) : UiState
}
```

Coroutine Scope and Lifecycle Awareness

Android provides lifecycle-aware coroutine scopes that automatically cancel coroutines when the appropriate lifecycle event occurs:

- **viewModelScope**: Tied to ViewModel lifecycle; cancels when ViewModel is cleared
- **lifecycleScope**: Tied to LifecycleOwner; respects lifecycle states
- **repeatOnLifecycle**: Runs a coroutine block whenever the lifecycle reaches a specified state

```
class MainActivity : AppCompatActivity() {
    private val viewModel: MainViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        lifecycleScope.launch {
            repeatOnLifecycle(Lifecycle.State.STARTED) {
                // This block runs when STARTED, cancels when STOPPED,
                // and re-runs when STARTED again
                viewModel.uiState.collect { state ->
                    when (state) {
                        is UiState.Loading -> showLoading()
                        is UiState.Success -> displayData(state.data)
                        is UiState.Error -> showError(state.message)
                    }
                }
            }
        }
    }
}
```

Dispatchers

Kotlin coroutines provide several dispatchers for controlling where code executes:

```
// Main thread (UI operations)
withContext(Dispatchers.Main) {
    textView.text = "Updated"
}

// Background thread pool (CPU-intensive work)
withContext(Dispatchers.Default) {
    val sorted = largeList.sorted()
}

// IO-optimized thread pool (network, disk, database)
withContext(Dispatchers.IO) {
    val data = database.query()
    val response = apiService.getData()
}

// Unconfined (not recommended for production; runs in caller's thread)
// Dispatchers.Unconfined
```

Structured Concurrency

Structured concurrency ensures that all child coroutines are properly managed and cancelled:

```
suspend fun loadMultipleData() {
    // All three coroutines run concurrently
    // If any fails, all are cancelled automatically
    val results = coroutineScope {
        val deferredUsers = async(Dispatchers.IO) { fetchUsers() }
        val deferredPosts = async(Dispatchers.IO) { fetchPosts() }
        val deferredComments = async(Dispatchers.IO) { fetchComments() }

        CombinedData(
            users = deferredUsers.await(),
            posts = deferredPosts.await(),
            comments = deferredComments.await()
        )
    }
}
```

Coroutine Best Practices for Android

1. Always use lifecycle-aware scopes (`viewModelScope`, `lifecycleScope`) rather than creating custom scopes in Activities or Fragments

2. Use `Dispatchers.IO` for I/O operations and `Dispatchers.Default` for CPU-intensive work
3. Avoid `GlobalScope` in application code; it creates orphaned coroutines
4. Use `repeatOnLifecycle` when collecting flows in UI components
5. Handle exceptions with try/catch or coroutine supervisors
6. Prefer `StateFlow` and `SharedFlow` over `LiveData` for new development

Kotlin Flow for Reactive Data Streams

Flow is Kotlin's reactive streams API, providing a cold asynchronous stream that emits values over time. It is the recommended approach for handling streaming data in Android applications.

Creating and Collecting Flows

```
// Creating a flow
val numbersFlow = flow {
    for (i in 1..5) {
        Thread.sleep(100)  // Simulate work
        emit(i)
    }
}

// Collecting a flow (must be done in a coroutine)
lifecycleScope.launch {
    numbersFlow.collect { number ->
        Log.d("FlowExample", "Received: $number")
    }
}
```

Flow Operators

Flows support a rich set of operators for transforming, filtering, and combining data:

```
val transformedFlow = repository.userFlow
    .filter { user -> user.isActive }           // Filter inactive users
    .map { user -> user.name.uppercase() }      // Transform to uppercase names
    .distinctUntilChanged()                      // Skip duplicates
    .debounce(300)                              // Wait 300ms between emissions
    .catch { e -> emit(emptyList()) }           // Handle errors gracefully

// Combining flows
val combinedFlow = combine(userFlow, settingsFlow) { user, settings ->
    UserWithSettings(user, settings)
}

// Zipping two flows
val zippedFlow = flow1.zip(flow2) { a, b ->
    Pair(a, b)
}
```

StateFlow and SharedFlow

For Android development, `StateFlow` and `SharedFlow` are the most commonly used Flow types:

```
// StateFlow: always has a current value, observers see the latest value
class UserRepository @Inject constructor(
    private val dao: UserDao
) {
    private val _currentUser = MutableStateFlow<User?>(null)
    val currentUser: StateFlow<User?> = _currentUser

    fun loadUser(userId: Long) {
        viewModelScope.launch {
            _currentUser.value = dao.getUserById(userId)
        }
    }
}

// SharedFlow: hot stream, no initial value required, configurable replay
class EventManager {
    private val _events = MutableSharedFlow<AppEvent>(replay = 0)
    val events: SharedFlow<AppEvent> = _events

    suspend fun sendEvent(event: AppEvent) {
        _events.emit(event)
    }
}

sealed class AppEvent {
```

```
    object UserLoggedIn : AppEvent()
    object UserLoggedOut : AppEvent()
    data class NavigationRequested(val route: String) : AppEvent()
}
```

Flow vs LiveData

Feature	StateFlow	LiveData
Null safety	Type-safe nullable types	Requires explicit null checks
Initial value	Required for StateFlow	Optional
Backpressure	Built-in operators	Limited
Testing	Easy with TestDispatcher	Requires main dispatcher override
Lifecycle awareness	Manual (repeatOnLifecycle)	Automatic
Compose integration	Direct collectAsStateWith-Lifecycle	Requires asStateFlow() conversion

For new projects, StateFlow combined with `collectAsStateWithLifecycle()` in Compose is the recommended approach.

Kotlin Multiplatform: Shared Business Logic

Kotlin Multiplatform (KMP) allows you to share business logic across Android, iOS, desktop, and web platforms while keeping platform-specific UI code native.

What Can Be Shared

- **Network layer**: API clients, data models, serialization
- **Business logic**: Use cases, validation rules, domain models
- **Data storage**: Room-compatible databases (SQLite), preferences
- **Utilities**: Date formatting, string manipulation, encryption

What Remains Platform-Specific

- **UI**: Compose for Android, SwiftUI for iOS
- **Platform APIs**: Sensors, camera, notifications
- **Native libraries**: Platform-specific SDKs

Basic KMP Project Structure

```
shared/
├── src/
│   ├── commonMain/kotlin/
│   │   ├── data/
│   │   │   ├── repository/
│   │   │   └── model/
│   │   ├── domain/
│   │   │   ├── usecase/
│   │   │   └── model/
│   │   └── di/
│   ├── androidMain/kotlin/
│   │   └── Platform.android.kt
│   └── iosMain/kotlin/
│       └── Platform.ios.kt
```

```kotlin
// Expect/Actual pattern for platform-specific code
// In commonMain:
expect object Platform {
    val name: String
}

// In androidMain:
actual object Platform {
    actual val name: String = "Android ${android.os.Build.VERSION.SDK_INT}"
}

// In iosMain:
actual object Platform {
    actual val name: String = "iOS"
}
```

When to Use KMP

Kotlin Multiplatform makes sense when:

- You are building applications for both Android and iOS
- Significant business logic can be shared between platforms
- Your team has Kotlin expertise but limited iOS development resources
- You want to maintain native UI quality on each platform

For Android-only projects, the complexity of KMP may not justify the benefits. Focus on mastering Android-specific patterns first.

Chapter Summary

Kotlin is the foundation of modern Android development. Its null safety eliminates an entire category of runtime errors. Its conciseness reduces boilerplate without sacrificing clarity. Coroutines and Flow provide elegant solutions for asynchronous programming that respect Android's lifecycle model. Extension functions, data classes, scope functions, and when expressions make everyday coding more pleasant.

In the next chapter, we explore modern app architecture patterns: how to structure your codebase for maintainability, testability, and scalability using MVVM, MVI, Clean Architecture, and dependency injection with Hilt.

Exercises

1. Create a Kotlin data class representing a blog post with fields for title, author, content, publish date, and a list of tags. Write extension functions to format the post as a preview string and to check if it contains a search term.
2. Write a coroutine that fetches data from a simulated API (use `delay()` instead of actual network calls), transforms the result, and emits it through a StateFlow. Collect the flow in a lifecycle-aware manner.
3. Convert a Java class with nullable fields, getter/setter methods, and a switch statement to idiomatic Kotlin using data classes, properties, and when expressions.
4. Research Kotlin Multiplatform and identify three libraries that support KMP for networking, serialization, and dependency injection.

Interview Questions

1. What is the difference between a nullable type (`String?`) and a non-nullable type (`String`) in Kotlin, and how does the compiler enforce this distinction?
2. Explain the difference between `let`, `run`, `with`, `apply`, and `also`. When would you use each one?
3. What is structured concurrency in Kotlin coroutines, and why is it important for Android development?
4. How do `StateFlow` and `SharedFlow` differ, and when would you choose one over the other?
5. What are the benefits of using Kotlin Multiplatform, and what types of code should remain platform-specific?

Chapter 4: Modern App Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Evolution of Android Architecture: MVC to MVVM to MVI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Early Days: MVC and Code Behind

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

MVP: The First Separation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

MVVM: Google's Long-Standing Recommendation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

MVI: The Compose Era Contender

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

ViewModel, Repository, and Use Case Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

ViewModel: The State Holder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Repository: The Data Mediator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Use Cases: Single-Responsibility Actions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Dependency Injection: Hilt and Koin

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Hilt: Google's Recommended DI Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Setting Up Hilt

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Hilt Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Hilt Component Hierarchy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Koin: The Lightweight Alternative

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Hilt vs Koin Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Navigation Component and Deep Linking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Compose Navigation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Deep Linking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Clean Architecture on Android 17

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Layer Responsibilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Module Boundaries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 5: Jetpack Compose — Declarative UI for Android

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

From XML Layouts to Compose: The Declarative Paradigm Shift

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Imperative View System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Declarative Compose Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Why Declarative?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Compose Fundamentals: Composables, State, and Recomposition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Writing Your First Composable

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

State in Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

State Hoisting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Recomposition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Side Effects in Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Layouts, Modifiers, and Custom Components

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Core Layout Composables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Modifiers: Composable Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Lazy Lists: Efficient Scrolling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Building Custom Components

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Navigation in Compose: Single-Activity Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Bottom Navigation Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Animations, Theming, and Material Design 3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Compose Animations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Material Design 3 Theming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Adaptive Layouts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 6: Android Views and Compose Interoperability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The View System: Layouts, Adapters, and RecyclerView

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Core Layout Containers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

RecyclerView for Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Compose in Views: ComposeView and AndroidView

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Using Compose Inside an Activity or Fragment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

ComposeFragmentActivity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

AndroidView: Embedding Views in Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Views in Compose: AndroidViewBridge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Incremental Migration Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Strategy 1: Compose Islands

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Strategy 2: Feature-by-Feature Migration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Strategy 3: Bottom-Up Component Replacement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Migration Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

When to Use Views vs Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 7: Lifecycle Management and Process Death

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Activity Lifecycle: Creation, Resumption, Pausing, and Destruction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Lifecycle States

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Critical Rule

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Service Lifecycles: Foreground, Background, and WorkManager

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Started Services

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Foreground Services

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Configuration Changes and SavedStateHandle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Surviving Configuration Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17's Optimized Configuration Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Process Death and State Restoration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Understanding Process Death

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Surviving Process Death

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

State Restoration Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Lifecycle-Aware Components and LifecycleOwner

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Lifecycle Observer Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

collectAsStateWithLifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 8: Permissions, Security, and Privacy in Android 17

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Permission Model: Normal, Dangerous, and Special Permissions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Permission Categories

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Declaring Permissions in the Manifest

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Runtime Permissions and Scoped Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Requesting Dangerous Permissions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Permission Groups

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Scoped Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 Privacy Enhancements: Photo Picker, Notification Permissions, and More

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

ACCESS_LOCAL_NETWORK Permission

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android Contacts Picker

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

OTP Protection for SMS Messages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Encrypted Client Hello (ECH)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Security Best Practices: Keystore, Encryption, and Network Security Config

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android Keystore System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Network Security Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

App Sandbox and Inter-Process Communication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Android Sandbox

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Content Providers for Safe Data Sharing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 Security Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 9: Background Processing and Battery Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Background Execution Limits and Android 17 Restrictions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17's App Memory Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

WorkManager: Deferrable, Guaranteed Background Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Basic Work Request

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Periodic Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Work Chains and Continuations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Observing Work Status

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Foreground Services and Foreground Service Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Declaring Foreground Service Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Foreground Service Types Available

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 Background Audio Hardening

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

JobScheduler and AlarmManager for Scheduled Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

JobScheduler for Constraint-Based Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

AlarmManager for Precise Timing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Battery Optimization: Doze Mode, App Standby, and Power Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Doze Mode

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

App Standby Buckets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Power Optimization Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 10: Networking and Data Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

HTTP Client Libraries: Retrofit, Ktor, and OkHttp

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Retrofit: The Industry Standard

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Ktor Client: The Multiplatform Alternative

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

OkHttp Interceptors for Cross-Cutting Concerns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

JSON Serialization: Kotlinx Serialization and Moshi

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Kotlinx Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Moshi with Kotlin Codegen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Room Database: Entities, DAOs, and Migrations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Defining Entities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

DAO (Data Access Object)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Database with Migrations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

DataStore: Preferences and Proto-based Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Preferences DataStore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Proto DataStore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Caching Strategies and Offline-First Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Repository with Cache-Then-Network Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Offline-First Principles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 11: Media, Graphics, and Sensors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

CameraX and the Modern Camera API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 Camera Enhancements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Audio Playback: ExoPlayer, Media3, and AudioFocus

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Media3 ExoPlayer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

AudioFocus Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Video Recording and Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

MediaRecorder for Video Capture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Skia, Canvas, and Custom Drawing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Custom Compose Drawings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 WebGPU Support

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Sensor APIs: Accelerometer, Gyroscope, Location, and Geofencing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

SensorManager for Device Sensors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Location and Geofencing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 12: AI, On-Device ML, and Android 17 Intelligent Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

TensorFlow Lite on Android: Model Loading and Inference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Setting Up LiteRT

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Running Inference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

MediaPipe Tasks: Vision, Audio, and NLP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Image Classification with MediaPipe

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Object Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Neural Networks API (NNAPI) for Hardware Acceleration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

On-Device LLMs and Gemini SDK Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

MediaPipe LLM Inference API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Gemini SDK for Android

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 AI Features: Live Captions, Translation, and Smart Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Live Captions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

On-Device Translation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Smart Compose and Predictive Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 13: Performance Optimization and Profiling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android Studio Profiler: CPU, Memory, and Network

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

CPU Profiler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Memory Profiler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Network Profiler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Frame Rendering and Jank Detection with Choreographer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Understanding the Frame Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Detecting Jank

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Reducing Jank in Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Memory Management: Leak Canary, Heap Analysis, and Bitmap Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Common Memory Leak Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Heap Analysis in Android Studio

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Bitmap Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Startup Time Optimization and Splash Screen API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Measuring Startup Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Splash Screen API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Benchmark Library and Macro/Micro Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Macrobenchmarks: Measuring Real-World Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Microbenchmarks: Measuring Individual Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 14: Accessibility and Internationalization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Accessibility Fundamentals: Content Descriptions and Touch Targets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Content Descriptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Touch Target Sizes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

TalkBack, Switch Access, and Screen Magnification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Testing with TalkBack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Announcing Dynamic Content

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Switch Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Dynamic Type and Scalable UI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Material 3 Adaptive Typography

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Internationalization: Locale, Number Formatting, and Date/Time APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

String Resources with Plurals and Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Number and Date Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Per-App Language Settings (Android 13+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Right-to-Left Layouts and Bidirectional Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

RTL Support in Compose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Testing RTL

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 15: Testing on Android

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Unit Testing with JUnit 5 and MockK

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Setting Up the Test Environment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Testing ViewModels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Testing Repositories

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Compose Testing: Semantics, Actions, and Assertions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Basic Compose Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Semantics for Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Espresso and UI Automation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Testing View-Based Screens

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Idling Resources for Async Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Continuous Integration and Cloud Testing (Firebase Test Lab)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

GitHub Actions CI Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Firebase Test Lab

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 16: Debugging, Crashes, and Production Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android Studio Debugger: Breakpoints, Stepping, and Evaluate Expression

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Advanced Breakpoint Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Evaluate Expression

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Logcat Mastery: Filtering, Tags, and Remote Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Structured Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Logcat Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Crash Reporting: Firebase Crashlytics and ACRA

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Firebase Crashlytics Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

ANR Analysis and Main Thread Violations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Understanding ANRs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Detecting Main Thread Violations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 New ProfilingManager Triggers for Production Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Remote Configuration and Feature Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Firebase Remote Config

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 17: Publishing to the Play Store

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

App Signing: Upload Keys, Play App Signing, and Bundletool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Play App Signing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android 17 PQC APK Signing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Android App Bundles vs APKs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Why App Bundles?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Building an AAB

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Play Console: Store Listing, Screenshots, and Metadata

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Store Listing Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Content Rating and Data Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Release Tracks: Internal, Alpha, Beta, and Production

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Phased Rollouts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Play Store Policies, Rating Requirements, and Content Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Key Policy Areas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Common Rejection Reasons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter 18: Migration Strategies and Backward Compatibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

API Level Targeting: minSdk, targetSdk, and compileSdk

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

The Three SDK Attributes Revisited

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Runtime API Checks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Deprecated APIs in Android 17 and Their Replacements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Migration: Cleartext Traffic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Migration: SMS OTP Reading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Build Configuration: Flavors, Dimensions, and Dynamic Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Build Flavors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Dynamic Feature Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Backward Compatibility Libraries: AndroidX and Jetpack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Migration Case Studies: From Android 12 to Android 17

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Case Study 1: Background Service Migration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Case Study 2: Large Screen Adaptivity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Case Study 3: Permission Migration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Conclusion: Building for the Future

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

Key Takeaways

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

What Comes Next

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/thecompleteguidetoandroid17development.