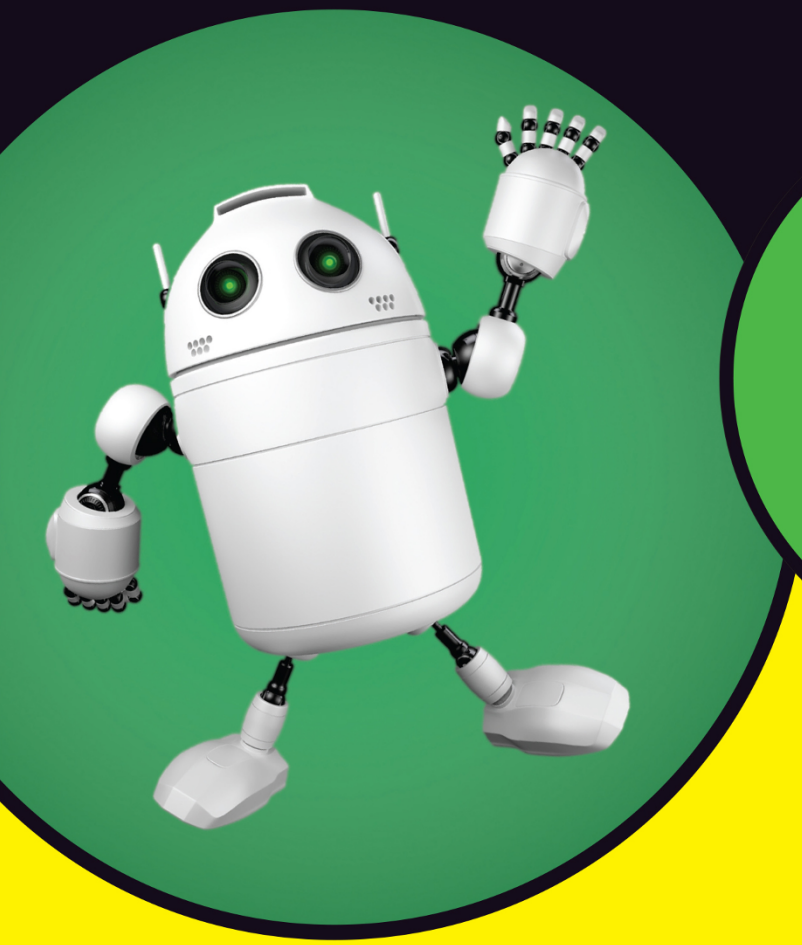
LEARNING MADE EASY

3rd Edition

Android® Application Development

ALL-IN-ONE

for dummies®
A Wiley Brand



6 Books in one!

Barry Burd
Author of *Java For Dummies*

John Paul Mueller

Android® Application Development

ALL-IN-ONE

3rd Edition

by Barry Burd and John Paul Mueller

for dummies®
A Wiley Brand

Android® Application Development All-in-One For Dummies®, 3rd Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

Library of Congress Control Number: 2020939707

ISBN: 978-1-119-66045-3

ISBN 978-1-119-66043-9 (ebk); ISBN 978-1-119-66046-0 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Introduction 1

Book 1: Getting Started with Android Application Development 5

CHAPTER 1: All about Android 7
CHAPTER 2: Installing the Software Tools 19
CHAPTER 3: Creating an Android App 43
CHAPTER 4: Examining a Basic Android App 67
CHAPTER 5: Conjuring and Embellishing an Android App 101
CHAPTER 6: Improving Your App 131

Book 2: Android Background Material 167

CHAPTER 1: Using Android Studio 169
CHAPTER 2: Kotlin for Java Programmers 203
CHAPTER 3: Kotlin for Everyone 227
CHAPTER 4: What Kotlin Does (and When) 261
CHAPTER 5: Object-Oriented Programming in Kotlin 291
CHAPTER 6: Functional Programming in Kotlin 315
CHAPTER 7: A <brief> Look at XML 339

Book 3: The Building Blocks 357

CHAPTER 1: Getting an Overview of Jetpack 359
CHAPTER 2: Building a Foundation for Your App 377
CHAPTER 3: Creating an Architecture 405
CHAPTER 4: Defining an App's Behavior 451
CHAPTER 5: Interacting with the Users 491

Book 4: Programming Cool Phone Features 521

CHAPTER 1: Hungry Burds: A Simple Android Game 523
CHAPTER 2: An Android Social Media App 547
CHAPTER 3: Going Native 567

Book 5: Apps for Tablets, Watches, TV Sets, and Cars 585

CHAPTER 1: Apps for Tablets 587
CHAPTER 2: Developing for Android Wear 615
CHAPTER 3: Developing for Android TV 637
CHAPTER 4: Developing for Android Auto 653

Book 6: The Job Isn't Done Until 679
CHAPTER 1: Publishing Your App to the Google Play Store 681
CHAPTER 2: Monetizing and Marketing Your App 717
CHAPTER 3: Creating Public Support for Your App. 739

Index 759

Table of Contents

INTRODUCTION 1
How to Use This Book 1
Conventions Used in This Book 2
Foolish Assumptions 2
Icons Used in This Book 3
Beyond the Book 4
Where to Go from Here 4

BOOK 1: GETTING STARTED WITH ANDROID APPLICATION DEVELOPMENT 5

CHAPTER 1: All about Android 7
The Consumer Perspective 8
The Versions of Android 9
The Developer Perspective 12
Java and Kotlin 12
XML 14
Linux 16
The Business Perspective 18

CHAPTER 2: Installing the Software Tools 19
Setting Up the Software 20
Considering the requirements 20
Downloading the software 21
Installing Android Studio 23
Installing offline tools 25
Launching the Android Studio IDE 28
In Windows 29
On a Mac 29
In Linux 30
In Chrome OS 30
Using the Android Studio Setup Wizard 30
Fattening Up the Android SDK 32
The more things stay the same, the more they change 32
Installing new versions (and older versions) of Android 33
Creating an Android virtual device 35
A third-party emulator 39

CHAPTER 3: Creating an Android App 43
Creating Your First App 43
Starting the IDE and creating your first app 45
Launching your first app 50

Running Your App 52
You Can Download All the Code 55
Troubleshooting Common IDE Errors 58
Error message: Failed to find target 58
Error running 'app': No target device found 58
Error message: Android Virtual Device may be incompatible with your configuration 58
You lose contact with the Android Debug Bridge (ADB) 59
You don't like whatever AVD opens automatically 59
The emulator stalls during startup 60
Error message: The user data image is used by another emulator 61
Error message: Unknown virtual device name 63
The emulator displays a "process isn't responding" dialog box . . . 63
Changes to your app don't appear in the emulator 64
Testing Apps on a Real Device 64

CHAPTER 4: **Examining a Basic Android App** 67
A Project's Files 68
The MainActivity.kt file 71
The onCreate() method 72
Using other templates 73
The res Branch 76
The res/drawable branch 77
The res/layout branch 77
The res/menu branch 78
The res/mipmap branch 79
The res/values branch 80
Other Files in an Android Project 82
The build.gradle file 82
The AndroidManifest.xml file 85
The R.java file 87
The assets folder 89
The android.jar archive 90
The APK file 91
What Did I Agree To? 93
What's in a name? 93
Choosing a language 95
Your app's API levels 95

CHAPTER 5: **Conjuring and Embellishing an Android App** 101
Dragging, Dropping, and Otherwise Tweaking an App 102
Creating the "look" 102
Coding the behavior 112

A Bit of Debugging 118
Try it! 118
Discovering the secrets of Logcat 123
Using the debugger 126

CHAPTER 6: **Improving Your App** 131
Improving the Layout 131
Changing the layout 132
Creating a reusable layout 139
Reusing a layout 142
Starting Another Activity 145
Localizing Your App 151
Responding to Check Box Events 155
Displaying Images 157
Sending in Your Order 162

BOOK 2: ANDROID BACKGROUND MATERIAL 167

CHAPTER 1: **Using Android Studio** 169
Good to Know versus Need to Know 170
Getting a Feel for the Big Picture 171
The main window 173
Viewing modes 179
The Designer tool 181
Discovering What You Can Do 184
Finding things 185
Fixing code 190
Refactoring 199

CHAPTER 2: **Kotlin for Java Programmers** 203
Using Kotlin or Java for Development 204
Defining the Java Issues That Kotlin Fixes 207
Improving control over null references 207
Removing raw data types 210
Using invariant arrays 210
Working with proper function types 212
Getting rid of the checked exceptions 213
Nothing's Perfect: Kotlin Is Missing Features, Too 214
Considering primitive types that are not classes 214
Losing static members 214
Eliminating non-private fields 215
Reducing confusion by eliminating wildcard-types 216
Abandoning the ternary-operator a ? b : c 217

Looking at What Kotlin Adds to the Picture 218
Considering higher order functions and lambdas 218
Refining object orientation using extension functions 218
Relying on smart casts 219
Employing string templates 220
Understanding primary constructors 221
Implementing first-class delegation 221
Using ranges of values 223
Creating data classes 224
Overloading operators 224
Developing asynchronous code using coroutines 225

CHAPTER 3: **Kotlin for Everyone** 227
Moving from Development to Execution with Kotlin 228
What is a compiler? 228
Understanding native code compiler or interpreter issues 230
Considering the Android Runtime (ART) 231
Grasping Kotlin Code 235
Nearly everything begins with an expression 236
The Kotlin class 238
Classes and objects 239
Kotlin types 240
Performing casts 245
The Kotlin function 249
Objects and their constructors 252
Classes grow on trees 254
The Kotlin package 255
Considering Kotlin visibility rules 257
Defying your parent 258
Kotlin comments 259

CHAPTER 4: **What Kotlin Does (and When)** 261
Making Decisions (Kotlin if Statements) 261
Testing for equality 264
Choosing among many alternatives (Kotlin when statements) 266
Repeating Instructions Over and Over Again 269
Kotlin while statements 269
Kotlin do statements 271
Arrays in Kotlin 273
Kotlin's for statements 277
Looping using Kotlin recursion 281
Working with break and continue 283

Jumping Away from Trouble 284
Working with Kotlin Collections. 286
Considering the collection types 287
Differentiating between read-only and mutable collections. 289

CHAPTER 5: **Object-Oriented Programming in Kotlin** 291
Static Fields and Methods 291
Interfaces and Callbacks. 294
Event handling and callbacks. 299
An object remembers who created it. 302
A less wordy way to implement an interface 303
Classes That Must (and Must Not) Be Extended. 305
The need to override. 306
Java's final classes 306
Kotlin's open classes 307
Kotlin extensions 307
Abstract classes 308
Inner Classes 310
Named inner classes. 310
Anonymous inner classes. 312

CHAPTER 6: **Functional Programming in Kotlin** 315
Defining Functional Programming 316
Differences from other programming paradigms 316
Understanding its goals 317
Understanding Pure and Impure Languages 318
Using the pure approach 318
Using the impure approach 320
Comparing the Functional Paradigm 320
Using Kotlin for Functional Programming Needs 322
Defining the Role of State. 323
Using Recursion to Perform Calculations 324
Relying on standard recursion. 324
Relying on tail recursion 326
Using Function Types 327
Understanding Function Literals. 329
Lambda expressions 329
Anonymous functions. 330
Defining the Function Types 331
Comprehensions 331
Receivers. 332
Inline 334
Utility 335
Using Functional Programming for Android Apps 336

CHAPTER 7: A <brief> Look at XML 339
XML Isn't Ordinary Text 340
Of tags and elements 340
Other things you find in an XML document. 348
What's in a Namespace? 350
The package attribute 353
The style attribute 354

BOOK 3: THE BUILDING BLOCKS 357

CHAPTER 1: Getting an Overview of Jetpack 359
Understanding the Benefits of Jetpack 360
Eliminating boilerplate code 360
Managing background tasks 361
Navigating between activities and fragments 362
Managing memory 364
Performing configuration changes 365
Considering the Jetpack Components 366
Foundation 367
Architecture 368
Behavior 370
UI 372
Getting an Overview of the AndroidX Package 373
Working with Lifecycle-Aware Components 374
Focusing on activities 375
Understanding events and states 376

CHAPTER 2: Building a Foundation for Your App 377
Working with Android KTX 378
Getting a feel for KTX features. 378
Using KTX in your project 381
Considering the modules 382
Addressing Security Issues. 389
Benchmarking Your Application 392
Removing barriers to correct results 393
Creating a test app 394
Profiling your app 397
Tracing your app 398
Checking for benchmarking module support 400
Benchmarking the app 401
Testing Application Functionality 403

CHAPTER 3: Creating an Architecture 405
Managing Application Activities. 405
Defining an activity 406
Getting an overview of intent filters 407

Considering the activity lifecycle 407
Understanding the backstack 409
Working with fragments 412
Considering the fragment lifecycle 416
Seeing activities and fragments in action 417
Providing for Navigational Needs 433
Creating the navigational graph 434
Adding a NavHostFragment to your activity 437
Adding destinations 438
Creating links between destinations 440
Creating the required linkages 442
Performing Background Tasks Using WorkManager 446

CHAPTER 4: **Defining an App's Behavior** 451
Working with Notifications 452
Understanding what notifications do 452
Anatomy of a notification 454
Assigning a channel to your notification 456
Setting the notification importance 457
Considering the notification types 458
Relying on notification updates 459
Do Not Disturb mode 460
Creating a notification 460
Getting Permission 466
Considering permission use 467
Configuring permissions in AndroidManifest.xml 468
Complying with User Preferences 469
Deciding on a preference set 470
Setting preferences using the Preference Library 472
Working with MediaPlayer 481
Adding Camera Support Using CameraX 484
Sharing with Others 487
Performing simple share actions with other apps 487
Using Slices 488

CHAPTER 5: **Interacting with the Users** 491
Creating a Great Layout 492
Defining the View and ViewGroup elements 492
Creating a layout using XML 493
Modifying a layout at runtime 497
Considering the common layouts 498
Working with adapters 499
Debugging your layout 500

Employing Color and Texture 502
Working with styles and themes 503
Creating a palette 509
Using swatches to create color schemes 510
Using Animations and Transitions 510
Understanding the need for animations 511
Animating graphics 511
Communicating with Emoji 514
Keyboard emoji support 515
Using the cut-and-paste method on standard controls 516
Using the AndroidX approach 517

BOOK 4: PROGRAMMING COOL PHONE FEATURES 521

CHAPTER 1: Hungry Burds: A Simple Android Game 523
Introducing the Hungry Burds Game 523
The Hungry Burds Project's Files 526
The Main Activity 528
The Code, All the Code, and Nothing But the Code 530
Setting Up the Game 535
Declaring properties 535
The onCreate Method 537
Displaying a Burd 538
Creating random values 538
Creating a Burd 539
Placing a Burd on the constraint layout 540
Animating a Burd 542
Handling a Touch Event 544
Finishing Up 546

CHAPTER 2: An Android Social Media App 547
Setting Things Up on Facebook's Developer Site 548
A Minimal Facebook App 549
The build.gradle file 550
The manifest file 550
A Bare-Bones Main Activity 551
Enriching the Minimal App 555
Working with a radio group 559
Controlling the web view 562
Who tests your Facebook app? 563

CHAPTER 3: Going Native 567
The Native Development Kit 567
Understanding why you need the NDK 568
Knowing what you get 569
Getting the NDK 570

Creating an Application 573
Starting with the template 573
Seeing the essential project differences 575
Considering the build.gradle (Module: app) differences 577
Understanding the default template differences 580
Getting an overview of the C++ file 582
Seeing the result 583

BOOK 5: APPS FOR TABLETS, WATCHES, TV SETS, AND CARS 585

CHAPTER 1: Apps for Tablets 587
Gaining Perspective 588
Creating the right devices 589
Running code on multiple devices 593
Copying the project 594
Seeing presentation differences 596
Developing a Nested Navigational Graph 603
Understanding the uses for nested navigational graphs 603
Developing an app design 604
Considering the content needs 608
Creating a Responsive App 612

CHAPTER 2: Developing for Android Wear 615
Seeing Where Wearables Are Used 615
Setting Up Your Testing Environment 617
Creating the project 617
Configuring a wearable device emulator 620
Other testing configurations 624
Wearable Apps: What's the Big Deal? 625
Case Study: A Watch Face 626
Defining the watch face project 627
Testing the watch face app 628
Dissecting the skeletal watch face project 631
Enhancing the skeletal watch face project 634

CHAPTER 3: Developing for Android TV 637
Getting Started 638
Running the Skeletal App 641
Dissecting the TV App 644
Adding to the standard AndroidManifest.xml 644
Looking into build.gradle (Module: app) 645
Defining a layout 646
The adapter and the presenter 647
Using the Adapter class 648
Using the Presenter class 650

CHAPTER 4: **Developing for Android Auto** 653
Checking Auto Compatibility 654
Choosing the Google Play Services 656
Considering Notification Limits 658
Creating an Emulator 660
Configuring your car for development 661
Defining an emulator 662
Developing an Android Auto App 670
Creating the project 670
Viewing the project configuration 672
Performing required configuration tasks 674
Touring the Media Service app 675

BOOK 6: THE JOB ISN'T DONE UNTIL 679

CHAPTER 1: **Publishing Your App to the Google Play Store** 681
Creating a Google Play Developer Account 681
Preparing Your Code 682
Un-testing the app 683
Choosing Android versions 683
Setting your app's own version code and version name 684
Choosing a package name 685
Preparing Graphic Assets for the Play Store 685
Creating an icon 686
Creating screenshots 688
Providing other visual assets 690
Creating a Publishable File 691
Differences among builds 692
Creating the release build 697
Running a new APK file 702
Running the app in a new AAB file 703
Another way to build and run an AAB file 705
Publishing Your App 708
The App Releases page 708
The Store Listing page 710
The App Signing page 711
Other pages 711
Leave No Stone Unturned 714
Publishing Elsewhere 714
The Amazon Appstore 714
Other venues 715

CHAPTER 2: **Monetizing and Marketing Your App** 717
Choosing a Revenue Model 718
Charging for your app 719
Offering an extended free trial 723
Freemium apps 724
Selling things with your app 726
Subscription pricing 729
Earning revenue from advertising 729
Variations on in-app advertising 731
Donationware 732
Offering your app for free 732
Getting paid to develop apps for others 732
Marketing Your Application 733
Brick Breaker Master: An App Marketing Case Study 734

CHAPTER 3: **Creating Public Support for Your App** 739
Obtaining Support through Patreon 740
Discovering that patronage isn't new 740
Considering crowdfunding 741
Defining why you should use crowdfunding 741
Understanding the development angle 742
Determining the trade-offs 744
Developing Your Own Distribution Stream 744
Creating podcasts 744
Developing YouTube videos 746
Employing social media 748
Answering questions 750
Taking the Personal Approach 750
Creating a blog 751
Answering your email 752
Considering App Store Alternatives 754
Getting Awards 756
Looking for awards in all the right places 757
Strutting your stuff 757

INDEX 759

Introduction

As the year 2019 drew to a close, Android ran on 51.8 percent of all smartphones in the United States and on 85 percent of all smartphones worldwide.[1,2] The Google Play Store had 2.57 million apps compared with only 1.84 million in Apple's App Store.[3]

Today, Android is everywhere, and experts predict that Android will dominate the global smartphone market for years to come.[4] So, if you read this book in a public place (on a commuter train, at the beach, on the dance floor at the Coyote Ugly saloon), you can read proudly, with a chip on your shoulder and with your chest held high. Android is hot stuff, and you're cool because you're reading about it.

How to Use This Book

You can attack this book in either of two ways. You can go cover to cover, or you can poke around from one chapter to another. You can even do both (start at the beginning and then jump to a section that particularly interests you). In this book, the basic topics come first, and the more involved topics follow the basics. You may already be comfortable with some basics, or you may have specific goals that don't require you to know about certain topics.

The best advice is as follows:

- If you already know something, don't bother reading about it.
- If you're curious, don't be afraid to skip ahead. You can always sneak a peek at an earlier chapter if you really need to do so.

[1] See `www.statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states/`.

[2] See `https://hostingtribunal.com/blog/operating-systems-market-share/#gref`.

[3] See `statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`.

[4] See `www.idc.com/promo/smartphone-market-share/os`.

Conventions Used in This Book

Almost every technical book starts with a little typeface legend, and this book is no exception. What follows is a brief explanation of the typefaces used in this book:

- New terms are set in *italics*.
- If you need to type something that's mixed in with the regular text, the characters you type appear in bold. For example: "Type **MyNewProject** in the text field."
- You see this `computerese` font for Kotlin code, filenames, web page addresses (URLs), onscreen messages, and other such things. Also, if something you need to type is really long, it appears in `computerese` font on its own line (or lines).
- You need to change certain things when you type them on your own computer keyboard. For instance, the instructions may ask you to type

  ```
  public void Anyname
  ```

  which means that you type **public void** and then some name that you make up on your own. Words that you need to replace with your own words are set in *`italicized computerese`*.

Foolish Assumptions

This book makes a few assumptions about you, the reader. If one of these assumptions is incorrect, you're probably okay. If all these assumptions are incorrect . . . well, buy the book anyway.

The assumptions are as follows:

- You can navigate through your computer's common menus and dialog boxes. You don't have to be a Windows, Macintosh, or Linux power user, but you should be able to start a program, find a file, put a file into a certain folder . . . that sort of thing. Much of the time, when you practice the stuff in this book, you're typing code on your keyboard, not pointing and clicking your mouse.
- You can think logically. That's all there is to application development — thinking logically. If you can think logically, you have it made. If you don't believe that you can think logically, read on. You may be pleasantly surprised.

» You have some programming experience (maybe not a lot). This book should be interesting for experienced programmers, yet accessible to people who don't write code for a living. If you're a programming guru, that's great. If you're a certified Linux geek, that's great, too. But no one expects you to be able to recite the names of Kotlin's concurrency primitives in your sleep, or pipe together a chain of 14 Linux commands without reading the documentation.

By the way, if you have no experience with an object-oriented language, you can get some. Your favorite bookstore has a terrific book titled *Java For Dummies,* 7th Edition, by Barry Burd (John Wiley & Sons, Inc.). The book comes highly recommended.

Icons Used in This Book

Throughout this book, an icon in the margin marks the beginning of a little detour — a fact or tidbit that stands out for one reason or another. The paragraphs marked by icons differ in their degree of importance. Some are valuable reading, others are silly trivia, and many are somewhere in the middle.

What kinds of icons do you find in this book? Here's a list:


TIP

A Tip is an extra piece of information — something helpful that the other books may forget to tell you.


WARNING

A Warning icon describes a mistake that many people make. Don't interpret the icon to mean, "Never make this mistake." That would be unreasonable. Instead, think of the icon as a word of comfort. It says, "Like everyone else, you'll probably make this mistake. When you do, remember that you once read about it here. Return to this section if you feel so inclined."


REMEMBER

Question: What's stronger than a Tip, but not as strong as a Warning?

Answer: A Remember icon.


TECHNICAL STUFF

Each Technical Stuff icon introduces an interesting fact. Some of these facts help you understand the reasoning behind the design of Android. You don't have to read all the Technical Stuff icons, but you may find them useful. They're especially helpful if you plan to read other (geekier) books about Android app development.

Beyond the Book

You've read the *Android All-in-One* book, seen the *Android All-in-One* movie, worn the *Android All-in-One* T-shirt, and eaten the *Android All-in-One* candy. What more is there to do?

That's easy. Just visit this book's website — `www.allmycode.com/Android`. At the website, you can find updates, comments, additional information, and lots of downloadable code. (You can also get there by visiting `www.dummies.com` and searching for *Android Application Development All-in-One For Dummies,* 3rd Edition.

Also on this book's page at `www.dummies.com` is the Cheat Sheet, which provides you with hints you need for nearly every Android app, such as parts of an Android app and the contents of that all important `.apk` file. You also get quick reminders about navigation classes, parts of a notification, and user interface elements.

Where to Go from Here

If you've gotten this far, you're ready to start reading about Android application development. Think of us (this book's authors) as your guides, your hosts, your personal assistants. We do everything we can to keep things interesting and, most important, help you understand.

If you experience any problems at all with this book, please contact either or both of us: Barry (`android@allmycode.com`) or John (`John@JohnMuellerBooks.com`) for assistance. We want you to be truly happy with your purchase and will help in any way we can with book-specific questions. You can also contact Barry on Twitter (`@allmycode`) and Facebook (`www.facebook.com/allmycode`).

Occasionally, we have updates to our technology books. If this book does have technical updates, they will be posted at this book's page at `www.dummies.com` and at `http://allmycode.com/android`.

1

Getting Started with Android Application Development

Contents at a Glance

CHAPTER 1: **All about Android** 7
The Consumer Perspective 8
The Versions of Android 9
The Developer Perspective 12
The Business Perspective 18

CHAPTER 2: **Installing the Software Tools** 19
Setting Up the Software 20
Launching the Android Studio IDE 28
Fattening Up the Android SDK 32

CHAPTER 3: **Creating an Android App** 43
Creating Your First App 43
Running Your App 52
You Can Download All the Code 55
Troubleshooting Common IDE Errors 58
Testing Apps on a Real Device 64

CHAPTER 4: **Examining a Basic Android App** 67
A Project's Files 68
The MainActivity.kt file 71
The res Branch 76
Other Files in an Android Project 82
What Did I Agree To? 93

CHAPTER 5: **Conjuring and Embellishing an Android App** . . . 101
Dragging, Dropping, and Otherwise Tweaking an App 102
A Bit of Debugging 118

CHAPTER 6: **Improving Your App** 131
Improving the Layout 131
Starting Another Activity 145
Localizing Your App 151
Responding to Check Box Events 155
Displaying Images 157
Sending in Your Order 162

IN THIS CHAPTER

- Your take on Android (depending on who you are)
- A tour of Android technologies

Chapter 1
All about Android

Until the mid-2000s, the word "Android" stood for a mechanical humanlike creature — a rootin' tootin' officer of the law with built-in machine guns, or a hyperlogical space traveler who can do everything except speak using contractions. But in 2005, Google purchased Android, Inc. — a 22-month-old company creating software for mobile phones. That move changed everything.

In 2007, a group of 34 companies formed the Open Handset Alliance. The Alliance's task was (and still is) "to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience." The Alliance's primary project is *Android* — an open, free operating system based on the Linux operating system kernel.

HTC released the first commercially available Android phone near the end of 2008, but the public's awareness of Android and its potential didn't surface until early 2010. By the mid-2010s, the world had more than 400 Android device manufacturers with 500 mobile carriers using Android and 1.5 million Android activations each day (`https://expandedramblings.com/index.php/android-statistics/`). By mid-2019, more than 2.5 billion active devices ran the Android operating system (`https://venturebeat.com/2019/05/07/android-passes-2-5-billion-monthly-active-devices/`). (We know. By the time you read this book, the year 2019 is old news. That's okay.)

This chapter introduces Android. The chapter examines Android from a few different angles.

The Consumer Perspective

A consumer considers the mobile phone alternatives.

Possibility #1: No mobile phone.

- **Advantages:** Inexpensive. No junk calls. No interruptions. No GPS tracking or snooping by businesses or other agencies.
- **Disadvantages:** No instant contact with friends and family. No calls to services in case of an emergency. No hand-held games, no tweeting, tooting, hooting, homing, roaming, or booping. And worst of all, to break up with your boyfriend or girlfriend, you can't simply send a text message.

Possibility #2: A feature phone — a mobile phone that's not a smartphone.

- **Advantages:** Cheaper than a smartphone.
- **Disadvantages:** Not as versatile as a smartphone. Not nearly as cool as a smartphone. Nowhere near as much fun as a smartphone.

There's no official rule defining the boundary between feature phones and smartphones. But generally, a feature phone is one with an inflexible menu of home-screen options. A feature phone's menu items relate mostly to traditional mobile phone functions, such as dialing, texting, and maybe some limited web surfing and gaming. In contrast, a smartphone's home screen provides access to the underlying file system and has icons, customizable skins, and many other features that used to be available only to general-purpose computer operating systems.

Don't write off feature phones. As late as March 2019, Counterpoint Research predicted that people will buy a billion feature phones between 2019 and 2022 (`https://www.counterpointresearch.com/more-than-a-billion-feature-phones-to-be-sold-over-next-three-years/`). This fact may shock you if you live in a country where feature phones are passé. But in 2019, the worldwide feature phone continued to grow.

Possibility #3: An iPhone.

- **Advantages:** Great graphics.
- **Disadvantages:** Little or no flexibility with the single-vendor iOS operating system. Only a handful of different models to choose from. No sanctioned "rooting," "modding," or "jailbreaking" the phone. And then there is the potential cost of an iPhone when compared to Android phones.

Possibility #4: An Ubuntu Touch phone, a Harmony OS phone, or some other non-Android, non-Apple smartphone.

- **Advantages:** Having a smartphone without belonging to a crowd.
- **Disadvantages:** Relatively difficult to get technical support. Not nearly as many apps as Android phones and Apple phones. Smaller selection of hardware to choose from.

Possibility #5: An Android phone.

- **Advantages:** Using an open platform. Using a popular platform with lots of industry support and with powerful market momentum. Writing your own software and installing the software on your own phone (without having to deal with Apple as an intermediary). Access to a broad range of hardware and price points. Publishing software without facing the challenging approval process used by Apple, plus you can choose not to use the Google Play Store (see `https://www.knowband.com/blog/mobile-app/alternatives-for-publishing-android-app-on-google-play-store/` for alternative ideas).
- **Disadvantages:** Security concerns when using an open platform. Confusion about the variety of manufacturers, each with different hardware and with some changes to the Android platform. Dismay when iPhone users make fun of your phone.

Android's advantages far outweigh the possible disadvantages. And you're reading a paragraph from *Android Application Development All-in-One For Dummies,* 3rd Edition, so you're likely to agree.

Having decided to go with an Android phone, the consumer asks, "Which phone?" And the salesperson says, "This phone comes with Android 10." (If you read between the lines, what the salesperson really means is "This phone comes with Android 9, which will eventually be upgraded to Android 10, or so claims the vendor.") So the consumer asks, "What are the differences among all the Android versions?"

The Versions of Android

Android comes with a few different notions of "version." Android has platform numbers, API levels, codenames, and probably some other versioning schemes. (The acronym *API* stands for *A*pplication *P*rogramming *I*nterface — a library full of prewritten programs available for use by a bunch of programmers. In this case, the "bunch" consists of all Android developers.)

To complicate matters, the versioning schemes don't increase in lockstep. For example, Android 8 (codenamed Oreo) has two API levels — levels 26 and 27. But Android 9 (codenamed Pie) has only one API level — level 28.

An Android version may have variations. For example, you can develop for plain old API Level 29 with an established set of features. To plain old API Level 29, you can add the Google APIs (thus adding Google Maps functionality) and still be using platform API Level 29. You can also add a special set with features tailored for a particular device manufacturer or a particular mobile service provider.

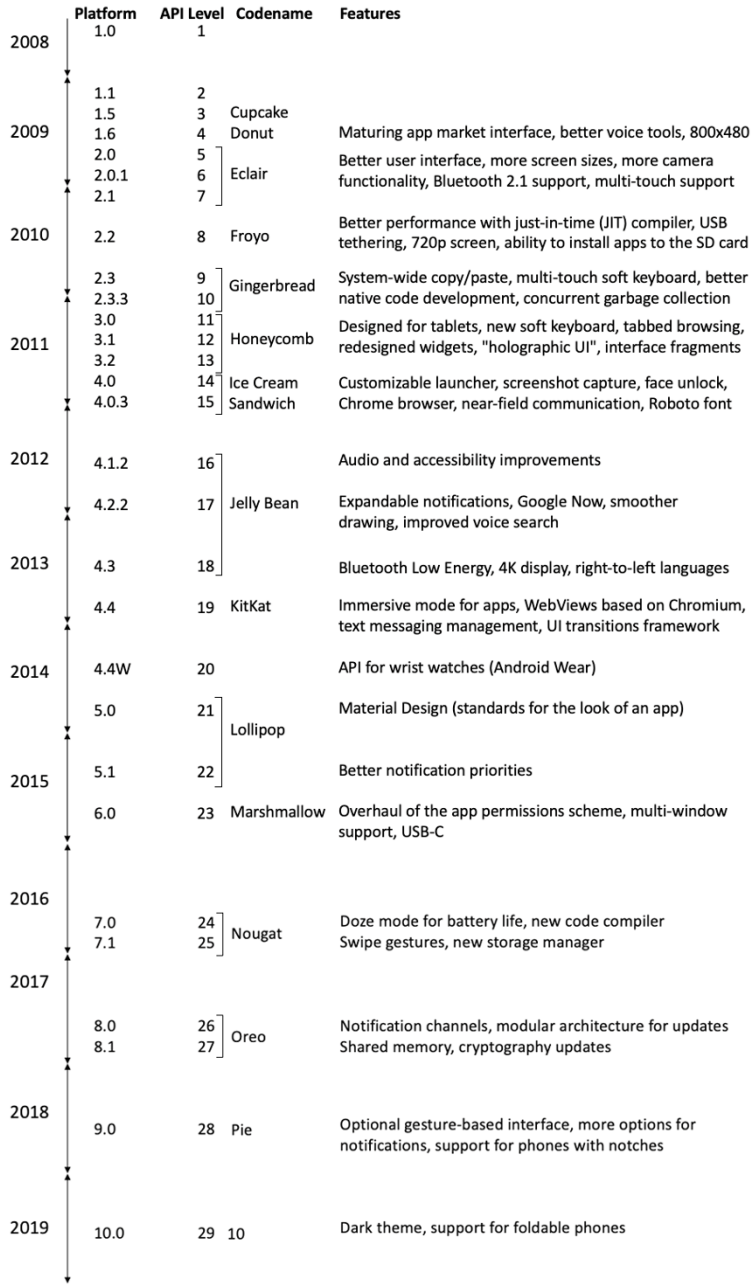
API levels 3 through 28 had tasty dessert codenames, and the names came in alphabetical order. For example, after Lollipop came Marshmallow; after Marshmallow came Nougat. Sad to say, the last-ever Android dessert codename was Pie, released in August 2018. About a year later, Google released a newer version simply named Android 10. The number 10 doesn't taste good the way lollipops and pies do.

Figure 1-1 has a summary of Android's API versions from 2008 to 2019.

A few notes on Figure 1-1 are in order:

- **The platform number is of interest to the consumer and to the company that sells the hardware.** If you're buying a phone with Android 9.0, for example, you might want to know whether the vendor will upgrade your phone to Android 10.0.
- **The API level (also known as the SDK version) is of interest to the Android app developer.** For example, in API level 8.1, the word `Build.SERIAL` stands for the phone's serial number. So, you might be tempted to type **Build.SERIAL** in code that uses API level 9.0. But in API level 9.0, `Build.SERIAL` doesn't help you get a phone's serial number. In API level 9.0, the value of `Build.SERIAL` is `"UNKNOWN"`.
- **The codename is of interest to the creators of Android.** A *codename* (also known as the *version code*) refers to the work done by the creators of Android to bring Android to the next level. Picture Google's engineers working for months behind closed doors on Project Oreo, and you'll be on the right track.

Since 2016, a new version of Android has come roughly once a year. Google released Nougat in 2016, Oreo in 2017, Pie in 2018, and the sugarless Android 10 in 2019. As a developer, your job is to balance portability with feature richness. When you create an app, you specify a minimum Android version. (You can read more about specifying a minimum version in Chapter 4 of this minibook.) The higher the version, the more features your app can have. But the higher the version, the fewer the devices that can run your app.



	Platform	API Level	Codename	Features
2008	1.0	1		
	1.1	2		
	1.5	3	Cupcake	
2009	1.6	4	Donut	Maturing app market interface, better voice tools, 800x480
	2.0	5	Eclair	Better user interface, more screen sizes, more camera functionality, Bluetooth 2.1 support, multi-touch support
	2.0.1	6		
	2.1	7		
2010	2.2	8	Froyo	Better performance with just-in-time (JIT) compiler, USB tethering, 720p screen, ability to install apps to the SD card
	2.3	9	Gingerbread	System-wide copy/paste, multi-touch soft keyboard, better native code development, concurrent garbage collection
	2.3.3	10		
	3.0	11	Honeycomb	Designed for tablets, new soft keyboard, tabbed browsing, redesigned widgets, "holographic UI", interface fragments
2011	3.1	12		
	3.2	13		
	4.0	14	Ice Cream Sandwich	Customizable launcher, screenshot capture, face unlock, Chrome browser, near-field communication, Roboto font
	4.0.3	15		
2012	4.1.2	16	Jelly Bean	Audio and accessibility improvements
	4.2.2	17		Expandable notifications, Google Now, smoother drawing, improved voice search
2013	4.3	18		Bluetooth Low Energy, 4K display, right-to-left languages
	4.4	19	KitKat	Immersive mode for apps, WebViews based on Chromium, text messaging management, UI transitions framework
2014	4.4W	20		API for wrist watches (Android Wear)
	5.0	21	Lollipop	Material Design (standards for the look of an app)
2015	5.1	22		Better notification priorities
	6.0	23	Marshmallow	Overhaul of the app permissions scheme, multi-window support, USB-C
2016	7.0	24	Nougat	Doze mode for battery life, new code compiler
	7.1	25		Swipe gestures, new storage manager
2017	8.0	26	Oreo	Notification channels, modular architecture for updates
	8.1	27		Shared memory, cryptography updates
2018	9.0	28	Pie	Optional gesture-based interface, more options for notifications, support for phones with notches
2019	10.0	29	10	Dark theme, support for foldable phones

FIGURE 1-1: Android version history.

This book contains tips and tricks for striking a happy medium between whiz-bang features and universal use, and Google has some nifty tools to help you sort out the differences among Android versions. In Chapter 3 of this minibook, you use Android Studio to create your first app. During the app setup, a drop-down

box gives you a choice of Android versions for your app. Beneath that drop-down box is an innocent-looking Help Me Choose link. If you click this link, you see a page with the title Android Platform/API Version Distribution. This interactive page lists several of the most recent Android versions along with the percentage of phones that can run each version. And, when you click one of the Android versions, the page provides information about that version's features.

Storks and fairies don't install updates on your Android devices. The updates come via Wi-Fi or phone service through your carrier or device manufacturer. But by downloading and installing an independently developed Android release, you can break free of the corporate giants. For information about these independently developed releases, visit `http://forum.xda-developers.com/custom-roms`.

The Developer Perspective

Android is a multifaceted beast. When you develop for Android, you use many tool sets. This section gives you a brief rundown of those tool sets.

Java and Kotlin

James Gosling from Sun Microsystems created the Java programming language in the mid-1990s. (Sun Microsystems has since been bought out by Oracle.) Java's meteoric rise in use came from the elegance of the language and the well-conceived platform architecture. After a brief blaze of glory with applets and the web, Java settled into being a solid, general-purpose language with special strength in servers and middleware.

In the meantime, Java was quietly seeping into embedded processors.

An *embedded processor* is a computer chip that's hidden from the user as part of some special-purpose device. The chips in today's cars are embedded processors, and the silicon that powers your photocopier at work is an embedded processor. Pretty soon, the flowerpots on your windowsill will probably have embedded processors. By 2002, Sun Microsystems was developing Java ME (*M*obile *E*dition) for creating *MIDlets* based on the Mobile Information Device Profile (MIDP) to run on mobile phones in a manner similar to applets on a web page (see `https://www.techopedia.com/definition/116/midlet` for details). Java became a major technology in Blu-ray disc players, parking meters, teller machines, and other devices. So, the decision to make Java the primary development language for Android apps was no big surprise.