Learn Android Studio 3

Efficient Android App Development

Ted Hagos

Apress®

Learn Android Studio 3

Efficient Android App Development

Ted Hagos

Apress®

Learn Android Studio 3: Efficient Android App Development

Ted Hagos
Manila, National Capital Region, Philippines

ISBN-13 (pbk): 978-1-4842-3155-5 ISBN-13 (electronic): 978-1-4842-3156-2
https://doi.org/10.1007/978-1-4842-3156-2

Library of Congress Control Number: 2018933042

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: Matthew Moodie
Coordinating Editor: Mark Powers

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484231555. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

For Adrianne and Stephanie.

Table of Contents

About the Author xi
About the Technical Reviewer xiii
Acknowledgments xv
Introduction xvii
■ Chapter 1: Introduction 1
History 1
Statistics 2
Operating System 3
■ Chapter 2: Android Studio 5
Android Studio Setup 6
macOS 6
Windows 6
Linux 7
Configuring Android Studio 8
Hardware Acceleration 14
The Android Studio IDE 15

■ **Chapter 3: Application Fundamentals** **19**

Creating a Project 19

Using Android Studio 24

Compilation and Runtime 34

Android Components 35

Components 36

Activities 38

Services 38

Content Providers 39

Broadcast Receivers 39

■ **Chapter 4: Activities and Layouts** **41**

Building the Hello Screen 41

The Layout File 42

Main Java Program 45

Views and Layout 48

■ **Chapter 5: Handling Events** **55**

Overview of Event Handling 55

Declarative Event Handling 57

Programmatic Event Handling 59

Working with Text and Buttons 64

More Event Handling Code 70

Using an Inner Class as a Listener 72

Using MainActivity as the Listener 76

■ **Chapter 6: Working with Multiple Activities** **83**

Component Activation 83

Launching a Specific Activity 85

Demo Project 85

Pass Data to Another Activity 95

About the GCF Algorithm 95

Returning Results from Other Activities 107

Implicit Intents 114
Demo Project 115
Opening an http Request 117
Activity Life Cycle 120
■Chapter 7: UI Elements 125
UI Elements 125
Themes and Colors 125
Colors 126
Themes 130
AppBar 132
Demo App 134
Fragments 142
Project Setup 144
Adding Fragments Programmatically 151
■Chapter 8: Running in the Background 157
Long Running Tasks 157
Demo Project 158
AsyncTask 162
■Chapter 9: Debugging 169
Syntax Errors 170
Runtime Exceptions 171
Logic Errors 173
■Chapter 10: Data Persistence 177
SharedPreferences 178
Demo Project 179
Verifying the File 185
Application Level SharedPreferences 187
Internal Storage 195
How to Work with Internal Storage 195
Demo Project 196

■ **Chapter 11: App Distribution** 203
Preparing the App for Release 204
Prepare Materials and Assets for Release 204
Configure the App for Release 204
Build a Release-Ready Application 205
Releasing the App 210
■ **Appendix A: Introduction to Java** 213
The Java Language 213
Virtual Machine 214
Editions 214
Setup 215
Hello World 216
Program Structure 218
Other Considerations 222
Variables and Data Types 223
Primitive Types 224
Reference Types 224
Operators 225
Program Flow 228
Decisions 228
Loops 232
Simple Application of Control Structures 234
The FizzBuzz Problem 234
How to Print a 5×5 Multiplication Table 235
Methods 236
Object Oriented Programming 239
More Details on Classes 242
Constructors 243
Overloading 244

Packages 245
Multiple Types in a Package 246
When We Don’t Need “import” 246
Inheritance 247
Object as the Root Class 248
Single Rooted Class Inheritance 249
Polymorphism 249
Interfaces 250
Multiple Inheritance 251
Exceptions 252
Index 255

About the Author

Ted Hagos is the CTO of RenditionDigital International, a software development company based out of Dublin. Before he joined RDI, he had various software development roles and also spent time as trainer at IBM Advanced Career Education, Ateneo ITI, and Asia Pacific College. He spent many years in software development dating back to Turbo C, Clipper, dBase IV, and Visual Basic. Eventually, he found Java and spent many years there. Nowadays, he’s busy with full-stack Javascript and Android.

About the Technical Reviewer

Wallace Jackson has been writing for leading multimedia publications about his work in new media content development since the advent of Multimedia Producer Magazine nearly two decades ago. He has authored a half-dozen Android book titles for Apress, including four titles in the popular Pro Android series. Wallace received his undergraduate degree in business economics from the University of California at Los Angeles and a graduate degree in MIS design and implementation from the University of Southern California. He is currently the CEO of Mind Taffy Design, a new media content production and digital campaign design and development agency.

Acknowledgments

I don’t think a lot of people read the acknowledgement section of any book, probably not even the people I’m going to thank. But just in case they do read this book (and this section), I’d like to extend my thanks to them.

Thanks to Mark Powers and Matthew Moodie for guiding me through the manuscript development process. I used to have a romantic notion of the writing life; now I know better. Thanks also to Wallace Jackson, who did the technical review, and to Massimo Nardone, who helped out in the author review of the last four chapters. Special thanks to Steve Anglin, who got me into Apress.

Thanks to Steph and Adrianne for understanding why I skipped some of my house chores while writing this book.

Covering a topic as vast as Android and a tool as rich as Android Studio requires the effort of many individuals whom I haven’t really met and know personally, but they do deserve gratitude. This is a tough section to make because I know I am bound to miss some names, so if I miss some, it’s not because of ingratitude, it’s because of ignorance.

Introduction

Welcome to *Learn Android Studio 3*. This book will help you get started in your programming journey with the little green robot. You already bought the book, so you don't need to be convinced that programming for the mobile platform offers a lot of opportunity for software developers.

While the book is aimed at beginning Android programmers, it isn't for people who are completely new to programming. The book assumes that you have prior coding experience with any of the CFOL (C family of languages, e.g., C, C++, Java, C#, JavaScript). Ideally, you are already a Java programmer trying to get your feet wet in Android; in case you're not, don't worry. Basic Java programming is covered in the Appendix, and you can refer to that as you try to feel your way into the language.

The book covers two fronts: the fundamentals of Android programming and the use of Android Studio 3. Android programming concepts and the use of the IDE are explained using a combination of graphics and code walkthroughs: there's plenty of those in the book.

Chapter Overviews

Chapter 1 - Introduces Android. It deals with a bit of Android's history and the technical makeup of its OS.

Chapter 2 - Walks you through to the setup of Android Studio and its requisite software. Whether you use macOS, Linux, or Windows, this chapter has you covered.

Chapter 3 - We start dipping our toes into Android programming. We'll start with creating a basic project and then run it on an emulator. This activity is something that you will do many times in the course of your Android programming career. Well finish up with a discussion of what makes up an Android application and how it is different from a desktop application.

Chapter 4 - This chapter deals with building user interfaces, one of the most basic and probably widely used components in Android.

Chapter 5 - Continuing from Chapter 4, after you've built some UIs, you might want it to actually do something. This chapter deals with how to respond to user-generated events.

Chapter 6 - This chapter deals with Intents. We've used Intents in the previous chapter, but this chapter digs in deeper. Intents are uniquely an Android thing; it truly embraces loose coupling. The chapter shows plenty of examples on how to use Intents for component activation on a multiactivity application and how to pass data between activities.

Chapter 7 - This chapter is shorter than the rest but it will help you put a bit of polish into your app. It deals with UI design, themes, styles, the AppBar, and Fragments.

Chapter 8 - Android is very protective of the user experience; it doesn't allow apps to freeze the UI leaving the user clueless as to what to do next. If you've seen ANR (Application Not Responding) errors before, this chapter shows you how to avoid these things.

Chapter 9 - Shows some of the things you can do to debug your apps in Android Studio 3. It goes through a list of the kinds of errors you might encounter while coding and what you can do in Android Studio to respond them.

Chapter 10 - At some point in time, you need to be able to save all the data you've created in the application. This chapter shows you the basics of saving data using a file, shared preferences, and the internal storage.

Chapter 11 - When you're ready to distribute your app, you'll need to sign it and list it in a marketplace like Google Play. This chapter walks you through the steps on how to do it.

Appendix - This chapter breezes through the Java language. It deals with some of the basic language concepts you will need to get started in Android programming.

Chapter 1

Introduction

Most people would think of Android as a phone or tablet; or at least, that is what end users would think. A developer would probably think of Android as an operating system (OS), and for the most part, it is. Android was designed originally to work as a mobile OS, but as it progressed, it found its way to some other places like TVs, car systems, watches, e-readers, netbooks, game consoles, and so forth.

Android includes quite a bit of stuff. It is a comprehensive platform. Apart from the OS and prebuilt applications, it includes a very capable software development kit, libraries, application frameworks, and reference design. We will explore some of them in considerable detail in the coming chapters. In the meantime, we'll look at Android's history, some statistics, and the Android platform architecture.

History

Android has an interesting and very colorful history. It first came to life in 2003 when a company called Android Inc. was founded by Andy Rubin. Android Inc. was backed by Google, but they did not own it yet. In 2005, Google bought Android Inc. to the tune of 50M+ dollars. Sometime in 2007, the Open Handset Alliance was born, and the Android OS has been officially open sourced. At this point, Android had not even reached version 1.0 and it was far from mainstream; it reached V1.0 sometime in 2008, but they had not thought about dessert names just yet.

The year 2009 up to 2010 saw a torrent of rapid releases. Android was picking up steam. Cupcake, Donut, Froyo, éclair, and Gingerbread were released during this two-year period. 2011 is a major milestone because up until this point, the Android OS remained confined to mobile phones. Honeycomb, the successor to Gingerbread, was the first Android version to be installed on tablets. There was a bit of controversy with Honeycomb because Google did not release its code to open source immediately. The following is a quick summary of Android's history.

T. Hagos, *Learn Android Studio 3*, https://doi.org/10.1007/978-1-4842-3156-2_1

2003	Android Inc., founded by Andy Rubin and backed by Google, was born
2005	Google bought Android Inc.
2007	Android was officially open sourced. Google turned over its ownership to the Open Handset Alliance (OHA)
2008	version 1.0 was released
2009	versions 1.1, 1.5 (Cupcake), 1.6 (Donut), and 2.0 (Eclair) were released
2010	versions 2.2 (Froyo) and 2.3 (Gingerbread) were released
2011	3.0 (Honeycomb) and 4.0 (Ice Cream Sandwich) were released
2012	version 4.1 (Jellybean)
2013	version 4.4 (KitKat)
2014	versions 5.0–5.1 (Lollipop); Android became 64-bit
2015	version 6.0 (Marshmallow)
2016	version 7.0-7.1.2 (Nougat)
2017	version 8 (Oreo)

One other thing that makes Android's history colorful is the lawsuits. Sometime in the past, Oracle sued Google, alleging that the latter infringed some copyrights of Java. But the Java implementation of Android isn't based on Oracle's Java language implementation; it is instead based on OpenJDK. Before Android Studio 2.2, installation of a separate Java SDK was a prerequisite for Android Studio; that is no longer the case because OpenJDK is now part of the installation. Then, there were the lawsuits between Apple and Samsung; the main part of all that was about Android. There were some bumps in the past but the little robot marched on.

Statistics

7.2 billion

Number of Android devices. It already has exceeded the total number of people in the planet

3

Number of decades it took for mobile devices to go from zero to 7.2 billion

1.5 million

Number of Android devices being activated daily

2,617

Number of times users touch their mobile devices in a day

2 billion

Number of active Android users monthly

87

Percentage of share of Android in the mobile OS market

I know you are already into Android development; you are reading this book after all. If you weren't aware of these statistics before, I hope this gives you extra motivation to continue your journey toward mobile development. Mobile computing usage is growing at a rapid pace, and Android has the lion's share of it.

Operating System

The most visible part of Android, at least for developers, is its OS. An OS is a complex thing, but for the most part, it is what stands between a user and the hardware. That is an oversimplification, but it will suffice for our purposes. By "user," I don't literally mean an end user or a person. What I mean by it is an application, a piece of code that a programmer creates, like a word processor or an e-mail client.

Take the e-mail app, for example; as you type each character on the keys, the app needs to communicate to the hardware for the message to make its way to your screen and hard drive and eventually send it to the cloud via your network. It is a more involved process than I describe it here, but that is the basic idea. At its simplest, an OS does three things:

- manages hardware on behalf of applications
- provides services to applications like networking, security and memory management, and so forth
- manages execution of applications; this is the part that allows us to run multiple applications (seemingly) almost at the same time

Figure 1-1 shows a logical diagram of Android's system architecture; it is far from complete, since it doesn't show all the apps, components, and libraries in the Android platform, but it should give you an idea on how things are organized.

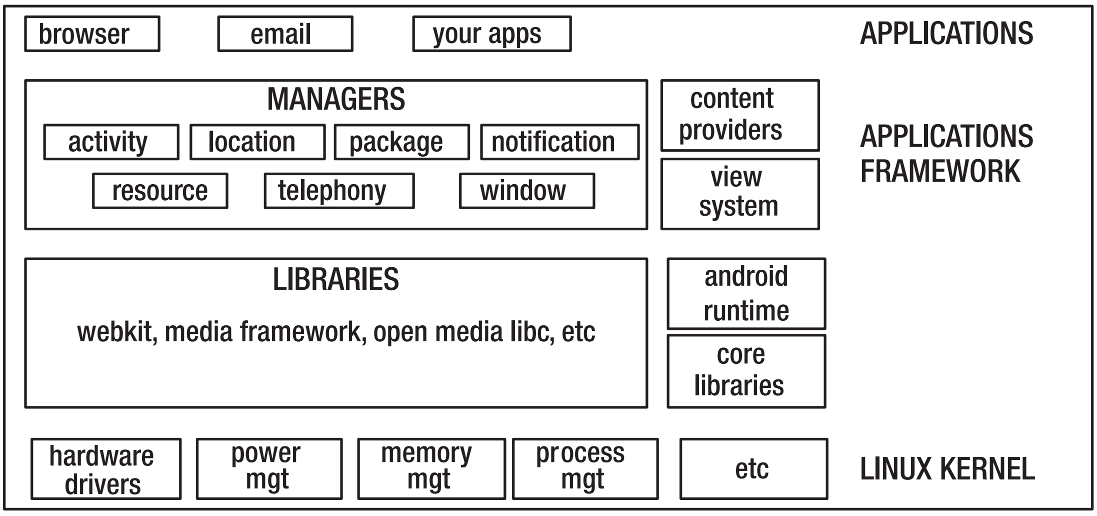


***Figure 1-1.** Platform architecture*

The lowest level in the diagram is the one responsible for interfacing with the hardware, various services like memory management, and executions of processes. It should sound familiar because these were the three things I said that OSes do. This part of the Android OS is Linux. Linux is a very stable OS and is quite ubiquitous itself. You can find it in many places like server hardware on data centers, appliances, medical devices, and so forth. Android has an embedded Linux inside it which handles hardware interfacing and some other kernel functions.

On top of the Linux kernel are low-level libraries like SQLite, OpenGL, and so on. These are not part of the Linux kernel but are still low level and as such, are written mostly in C/C++. On the same level, you will find the Android runtime (Android class libraries + Dalvik virtual machine), which is where Android applications are run. Unlike other Java programs, Android executables are not *.class* files; they are *.dex* files. Dex files are not run on a typical Java virtual machine (JVM) like the one installed on your desktop. The dex files are meant to run on a VM that is optimized for low-powered handheld devices. The compilation cycle could be summed to the following: .java files (source code) ➤ Java compiler (.class) ➤ dex compiler (.dex) ➤ packaging (.apk)

Note The Dalvik VM was written by Dan Borstein; the VM was named after a fishing village in Iceland.

Next up is the application framework layer. It sits on top of both the low-level libraries and the Android runtime because it needs both. This is the layer that we will interact with as an application developer because it contains all the libraries we need to write apps.

Finally, on top is the application layer. This is where all our apps reside, both the ones we write and the ones that comes prebuilt with the OS. It should be pointed out that prebuilt applications which come with the device do not have any special privileges over the ones we will write. If you don't like the e-mail app of the phone, you can write your own and replace it. Android is democratic like that.

Chapter 2

Android Studio

Developing applications for Android was not always as convenient as today. When Android 1.0 was released sometime in 2008, what developers got by way of a development kit was no more than a bunch of command-line tools and ant build scripts. Building apps with vim, ant, and some command-line tools, that wasn't so bad if you were used to that kind of thing, but many developers were not. The lack of integrated development environment (IDE) capabilities like code hinting, project setups, and integrated debugging was somewhat a barrier to entry.

Thankfully, the Android development tools (ADT) for the Eclipse IDE was released, also in 2008. Eclipse was, and still is, a favorite and dominant choice of IDE for many Java developers. It felt very natural that it would also be the go-to IDE for Android developers.

From 2009 up until 2012, Eclipse remained to be choice of IDE for development. The Android SDK has also undergone both major and incremental changes in structure and in scope. In 2009, the SDK manager was released; we use this to download tools, individual SDK versions, and Android images that we can use for the emulator. In 2010, additional images were released for the ARM processor and X86 CPUs. 2012 was a big year because Eclipse and ADT were finally bundled; this was a big deal because until that time, developers had to install Eclipse and the ADT separately, and the installation process wasn't always smooth. So, the bundling of the two together made it a whole lot easier to get started with Android development. 2012 was also a big year because it marked the last year of Eclipse being the dominant IDE for Android.

In 2013, Android Studio was released. To be sure, it was still on beta, but the writing on the wall was clear. It will be the official IDE for Android development. Android Studio is based on Jetbrain's IntelliJ. IntelliJ is a commercial Java IDE that also has a community (nonpaid) version. This is the version that would serve as the base for Android Studio. In this chapter, we will cover the following.

- Setup
- Configuration

T. Hagos, *Learn Android Studio 3*, https://doi.org/10.1007/978-1-4842-3156-2_2

- Hardware acceleration
- Some basic parts of Android Studio 3

Note Before you can install Android Studio, you need the Java 8 JDK. JDK installation instructions are in the appendix.

Android Studio Setup

The AS3 installer is available for macOS, Windows, and Linux. The download page detects the OS you are using, so you should be able to spot the download button fairly quickly. You will be asked to agree to some terms and conditions before you can proceed with the download. Read it, understand it, and agree to it so you can carry on. After that, the AS3 installed will be downloaded in a zipped file.

If you have an existing installation of Android Studio, you can keep using that version and still install the preview edition. AS3 can coexist with your existing version of Android Studio; its settings will be kept in a different directory.

macOS

You must have seen the installation instruction after the terms and conditions screen; if you haven't or you skipped through it, I suggest that you give it a once-over, because there is an installation warning in case you have an existing version of Android Studio. It says that if you downloaded Android Studio version 2.3 or earlier, the application name on macOS installer does not include the version number. So, you may want to rename your existing Android Studio prior to installing the preview version. You can rename your existing Android Studio installation by opening a finder window; then, select "Applications" from the sidebar, find Android Studio, activate the context menu (press Ctrl + mouse click), and choose rename. The installation notes for AS3 are at `https://developer.android.com/studio/preview/install-preview.html`

1. Unpack the zipped file
2. Drag the application file into the **Applications** folder
3. Launch AS3
4. AS3 will prompt you to import some settings; if you have a previous installation, you can import that (it is the default option)

Windows

1. Unzip the installer file
2. Move it to a folder location of your choice (e.g., `C:\AndroidStudio`). Drill down to this folder

3. Inside, you will find a `bin` folder; inside it, you will find `studio64.exe`. This file is what you need to launch. If you are on a 32-bit Windows, the launcher file is named `studio.exe`

> **Tip** If you right-click `studio64.exe` and choose **Pin to Start Menu**, you can make AS3 available from the Windows Start menu; alternatively, you can pin it to the Taskbar.

Linux

The Linux installation requires a bit more work than simply double-clicking and following the installer prompts. In future releases of Ubuntu and its derivatives, this might change and become as simple and frictionless as its Windows and macOS counterparts, but for now, we need to do some tweaking. The extra activities on Linux are mostly because AS3 needs some 32-bit libraries and hardware acceleration.

> **Note** The installation instructions in this section are meant for Ubuntu 64-bit and other Ubuntu derivatives: Linux Mint, Lubuntu, Xubuntu, Ubuntu MATE, and so forth. I chose this distribution because I assumed that it is a very common Linux flavor, hence, readers of this book will be using that distribution.
>
> If you are running a 64-bit version of Ubuntu, you will need to pull some 32-bit libraries in order for AS to function well.

To start pulling the 32 bit libraries for Linux, run the following commands on a terminal window.

```
sudo apt-get update && sudo apt-get upgrade -y
sudo dpkg --add-architecture i386
sudo apt-get install libncurses5:i386 libstdc++6:i386 zlib1g:i386
```

When all the prep work is done, the AS3 installation can be managed using the following steps.

1. Unpack the downloaded installer file. You can unpack the file using command-line tools or using the GUI tools; you can, for example, right-click the file and select the “Unpack here” option, if your file manager has that option
2. After unzipping the file, rename the folder to AndroidStudio
3. Move the folder to a location where you have read, write, and execute privileges. Alternatively, you can also move it to /usr/local/AndroidStudio

4. Open a terminal window and go to the AndroidStudio/bin and execute ./studio.sh

5. At first launch, AS3 will ask you if you want to import some settings; if you have installed a previous version of Android Studio, you may want to import those settings into AS3

Configuring Android Studio

Before we can dive into programming, we need to do a couple of things to complete the development setup. We need to

- Get some more software so we start creating programs that target a specific version of Android
- Make sure we have all the SDK tools we need, and optionally
- Change the way we get updates for AS3

Launch AS3 if you haven't done so yet. From the opening dialog, click "Configure" and choose "SDK Manager" from the drop-down list. This should take you to a window where you can select the SDK platforms to download (Figure 2-1).

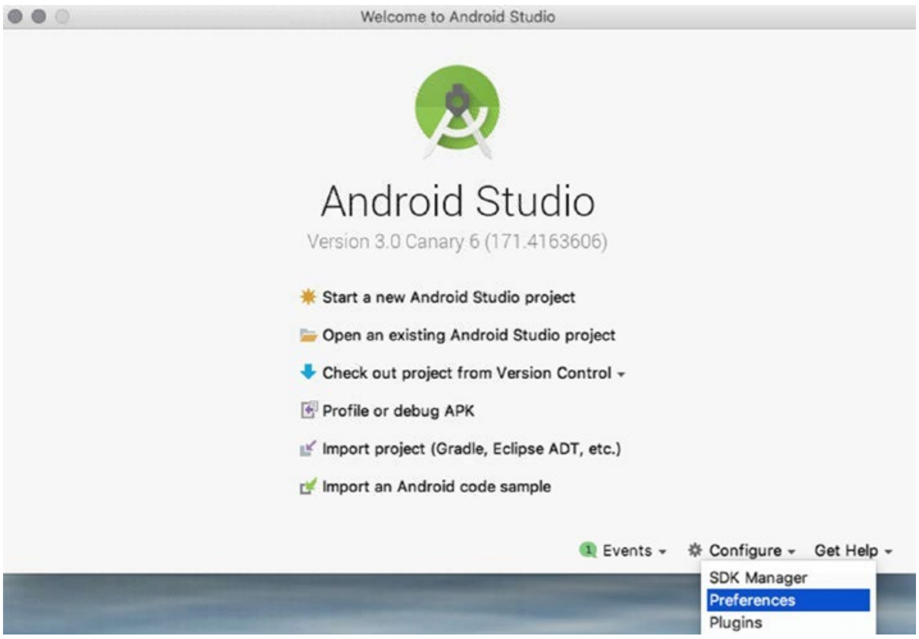

Welcome to Android Studio
Android Studio
Version 3.0 Canary 6 (171.4163606)
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Events
Configure
Get Help
SDK Manager
Preferences
Plugins

***Figure 2-1.** Preferences window*

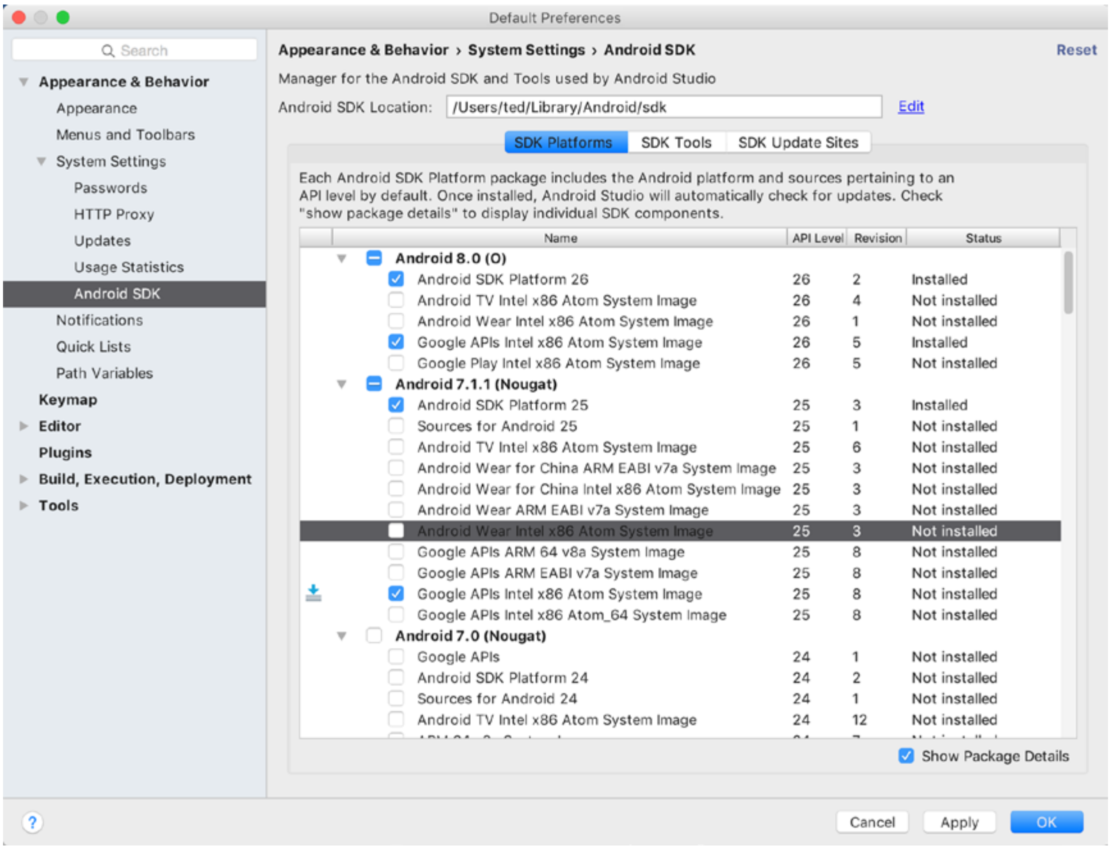
When you get to the SDK window, enable the "Show Package Details" option so you can see a more detailed view of each API level. We don't need to download everything in the SDK window. We will get only the items we need.

SDK levels or platform numbers are specific versions of Android. Android 8 or Android "O" is API level 26, and Nougat is API level 24 and 25. You don't need to memorize the platform numbers anymore, because AS3 shows the platform number with the corresponding Android nickname.

If you have a pretty fast Internet connection, you may choose to download everything. That way, you can create projects that target multiple versions of Android all the way down to Eclair. For our purposes, we will only download Nougat (platforms 24 and 25) and Oreo (platform 26). Make sure that together with the platforms, you will also download "Google APIs Intel x86 Atom_64 System Image". We will need those when we get to the part where we test run our applications.

Note Another way to build applications for earlier Android versions without having to download all the API levels is to use the Android Support Libraries; these libraries afford us backward compatibility.

Once you have completed the selection, click the "OK" button to start the download process (Figure 2-2).



Default Preferences

Search

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Updates
    - Usage Statistics
    - Android SDK
  - Notifications
  - Quick Lists
  - Path Variables
- Keymap
- Editor
- Plugins
- Build, Execution, Deployment
- Tools

Appearance & Behavior › System Settings › Android SDK Reset

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/ted/Library/Android/sdk Edit

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

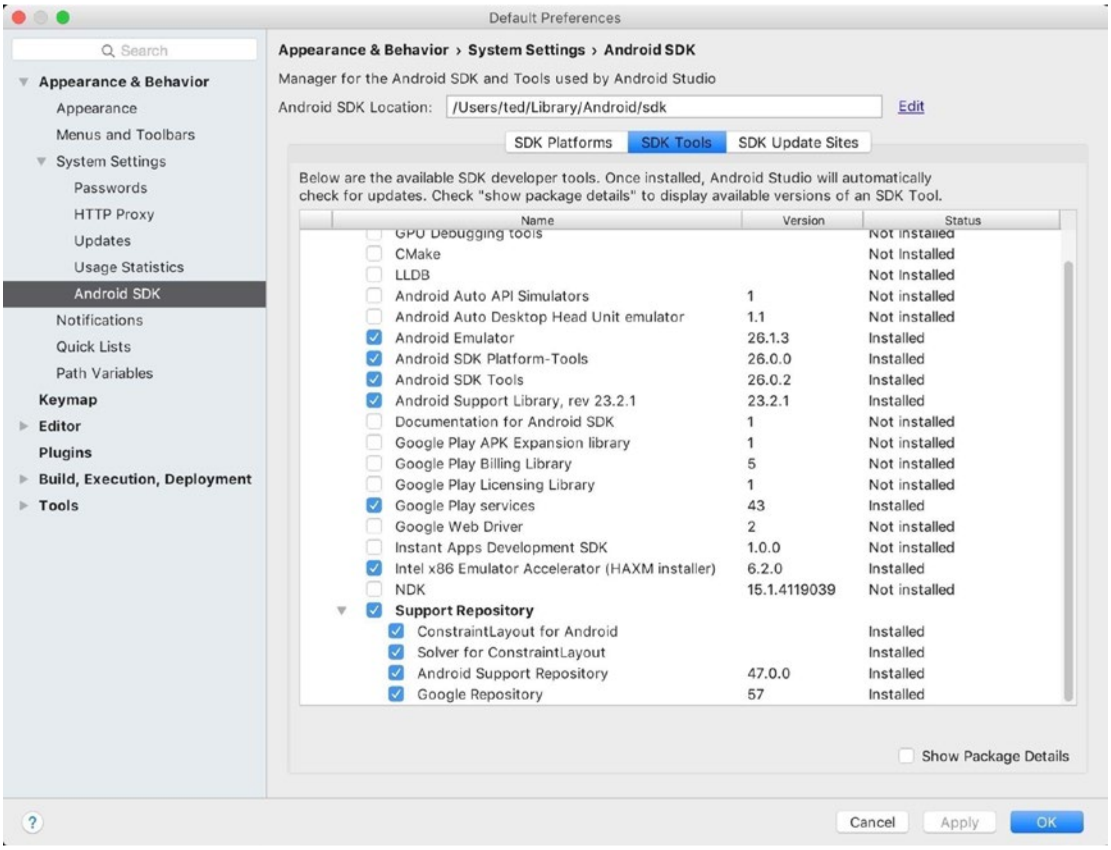
Name	API Level	Revision	Status
Android 8.0 (O)			
✓ Android SDK Platform 26	26	2	Installed
Android TV Intel x86 Atom System Image	26	4	Not installed
Android Wear Intel x86 Atom System Image	26	1	Not installed
✓ Google APIs Intel x86 Atom System Image	26	5	Installed
Google Play Intel x86 Atom System Image	26	5	Not installed
Android 7.1.1 (Nougat)			
✓ Android SDK Platform 25	25	3	Installed
Sources for Android 25	25	1	Not installed
Android TV Intel x86 Atom System Image	25	6	Not installed
Android Wear for China ARM EABI v7a System Image	25	3	Not installed
Android Wear for China Intel x86 Atom System Image	25	3	Not installed
Android Wear ARM EABI v7a System Image	25	3	Not installed
Android Wear Intel x86 Atom System Image	25	3	Not installed
Google APIs ARM 64 v8a System Image	25	8	Not installed
Google APIs ARM EABI v7a System Image	25	8	Not installed
✓ Google APIs Intel x86 Atom System Image	25	8	Not installed
Google APIs Intel x86 Atom_64 System Image	25	8	Not installed
Android 7.0 (Nougat)			
Google APIs	24	1	Not installed
Android SDK Platform 24	24	2	Not installed
Sources for Android 24	24	1	Not installed
Android TV Intel x86 Atom System Image	24	12	Not installed

✓ Show Package Details

Cancel Apply OK

Figure 2-2. SDK platforms

Next, we head to the “SDK Tools” tab. This is in the same section on the Preferences window; just click the tab in the middle to view the details of the tools (Figure 2-3).



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/ted/Library/Android/sdk Edit

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

	Name	Version	Status
☐	GPU Debugging tools		Not installed
☐	CMake		Not Installed
☐	LLDB		Not Installed
☐	Android Auto API Simulators	1	Not installed
☐	Android Auto Desktop Head Unit emulator	1.1	Not installed
✓	Android Emulator	26.1.3	Installed
✓	Android SDK Platform-Tools	26.0.0	Installed
✓	Android SDK Tools	26.0.2	Installed
✓	Android Support Library, rev 23.2.1	23.2.1	Installed
☐	Documentation for Android SDK	1	Not installed
☐	Google Play APK Expansion library	1	Not installed
☐	Google Play Billing Library	5	Not installed
☐	Google Play Licensing Library	1	Not installed
✓	Google Play services	43	Installed
☐	Google Web Driver	2	Not installed
☐	Instant Apps Development SDK	1.0.0	Not installed
✓	Intel x86 Emulator Accelerator (HAXM installer)	6.2.0	Installed
☐	NDK	15.1.4119039	Not installed
✓	**Support Repository**		
✓	ConstraintLayout for Android		Installed
✓	Solver for ConstraintLayout		Installed
✓	Android Support Repository	47.0.0	Installed
✓	Google Repository	57	Installed

☐ Show Package Details

Figure 2-3. *SDK tools*

You don't generally have to change anything on this window, but it wouldn't hurt to check that you have the following packages.

Android SDK Platform Tools	This contains important tools like `adb`, which will help us do diagnostics and debugging, and `sqlite3`, which we can use when we create applications that use databases, plus a couple of other tools.
Android SDK Tools	This includes essential Android tools like ProGuard. You don't need to deep dive into the details of these tools (for now). Just make sure this box is ticked and you're good to go.
Android Emulator	You will definitely use this. This is a device emulation tool. We will use this to test our applications in a virtual device.

Support Repository	If you want to write code that targets Android Wear, Android TV, or Google Cast, you want to download this. This also contains local Maven repository for support libraries. The support repository also allows you to use new features on older Android versions.
HAXM Installer	If you are using a macOS, or a PC with Intel processor, you can use this. It is an accelerator for the Android Emulator.

> **Note** If you are on the Linux platform, you cannot use HAXM even if you have an Intel processor. KVM will be used in Linux instead of HAXM.

After you've downloaded some target platforms and checked the SDK Tools, we can move on to the last configuration item, which is the "Update Channel". You can change this setting from the "Preferences" dialog window. From the opening dialog window, choose "Configure" and then "Preferences" (Figure 2-4).

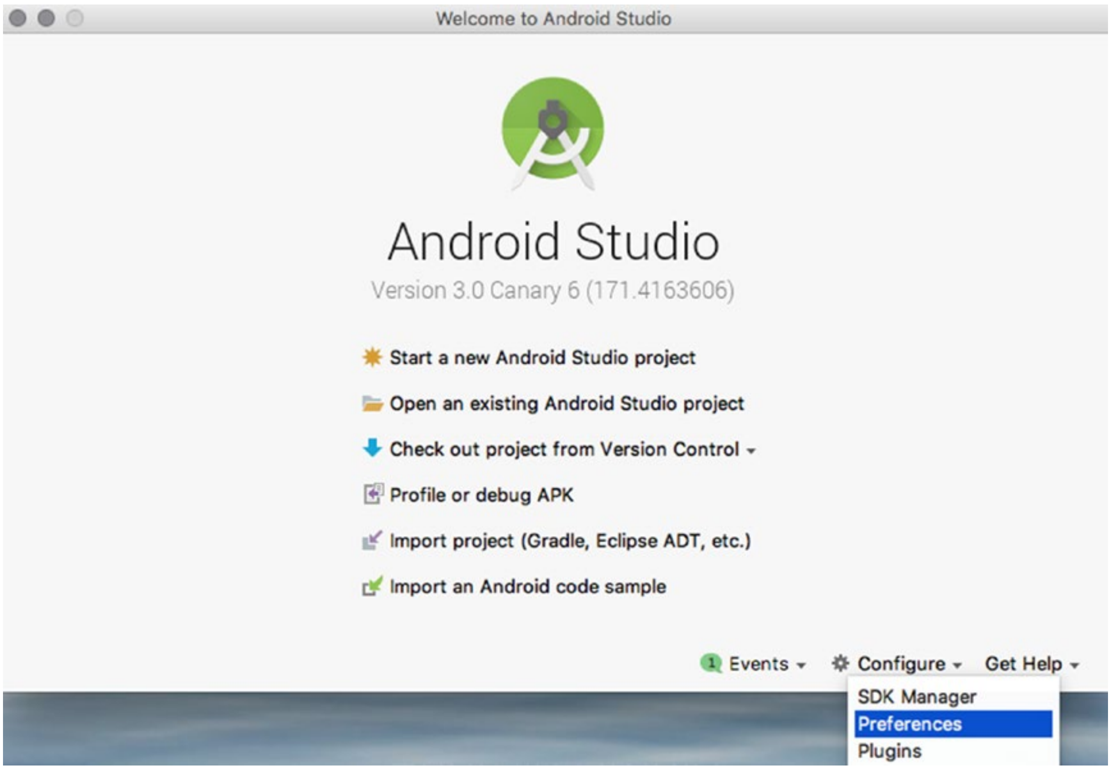
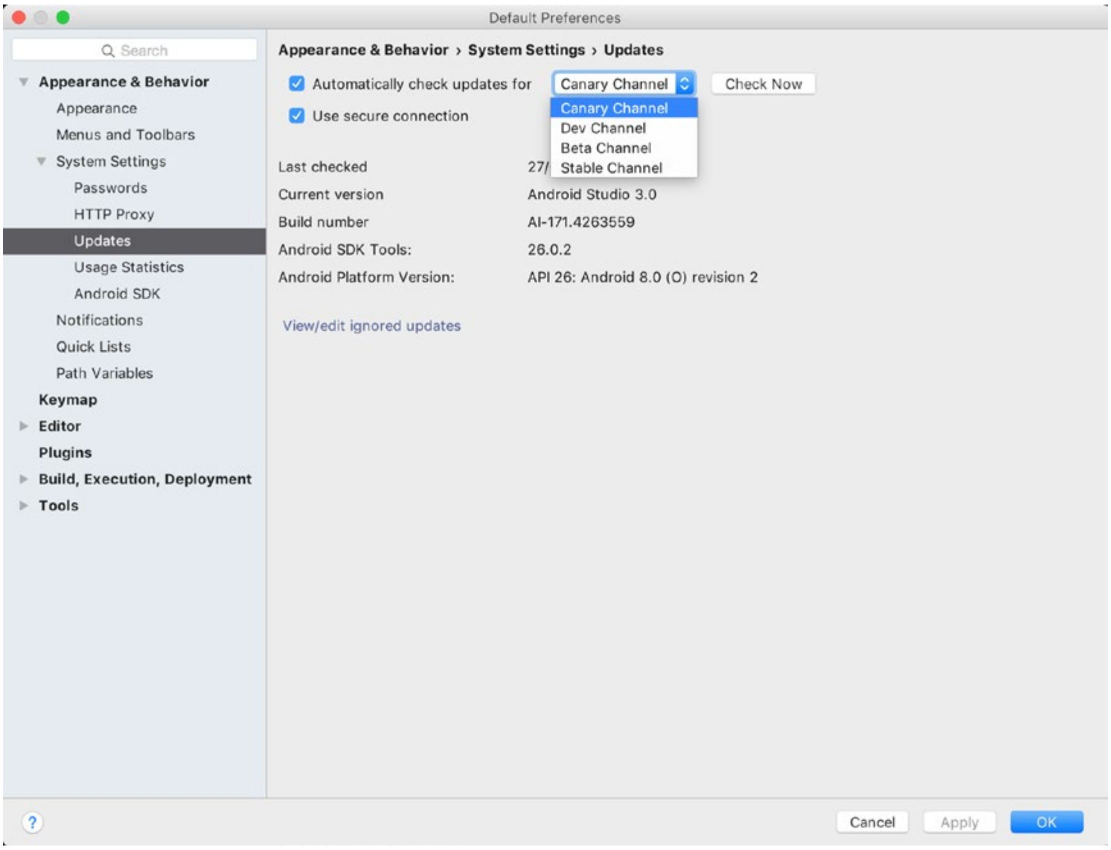

Welcome to Android Studio
Android Studio
Version 3.0 Canary 6 (171.4163606)
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Events
Configure
Get Help
SDK Manager
Preferences
Plugins

Figure 2-4. Preferences

On the left side of “Default Preferences”, choose “Updates” (Figure 2-5).



☑ Automatically check updates for Canary Channel

☑ Use secure connection

Last checked	27/
Current version	Android Studio 3.0
Build number	AI-171.4263559
Android SDK Tools:	26.0.2
Android Platform Version:	API 26: Android 8.0 (O) revision 2

***Figure 2-5.** Update Channel*

AS3, just like any Android Studio installation is configured by default to get updates from the channel where you originally downloaded the installer. At the time of this writing, AS3 was downloaded from the “Canary” Channel (also known as the Preview Channel), hence, it gets the update from the Canary Channel by default. You can change the channel to any one of these four:

- Canary Channel: This has bleeding-edge releases; it could be updated every week. You don’t want to use this for production codes
- Dev Channel: Just like the Canary Channel but a bit more stable. You still don’t want to use this for production
- Beta Channel: This contains release candidates. The devs are basically waiting for feedback before they get fed to the Stable Channel
- Stable Channel: This is the official stable release and is suited for production work