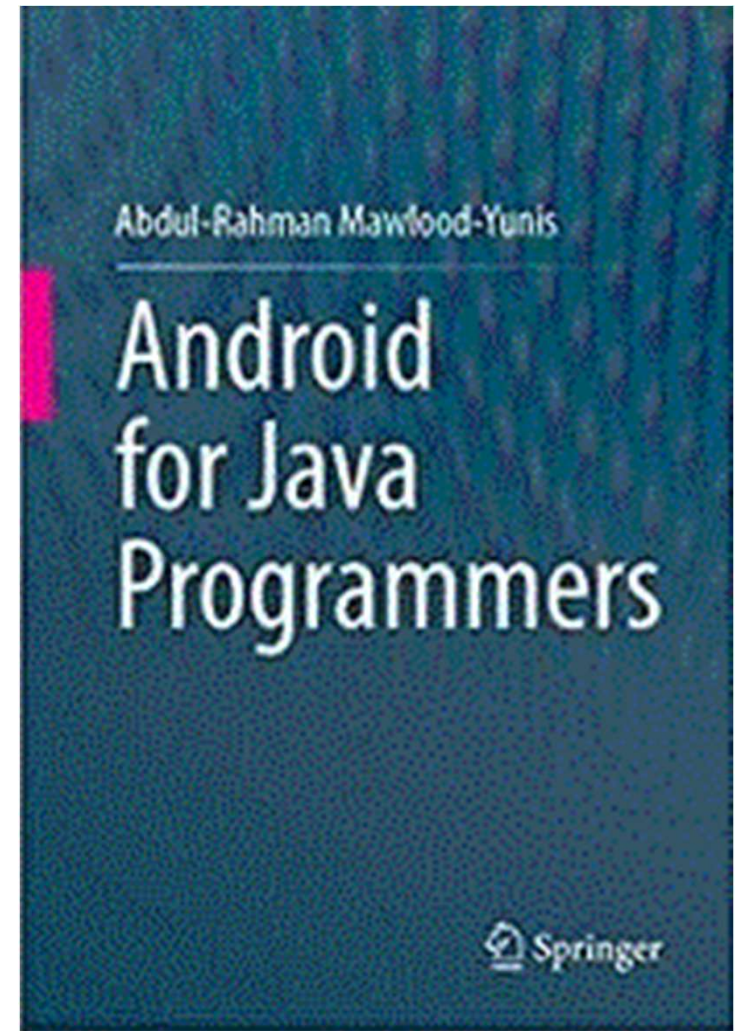


Chapter 02: Getting Started with Android



What You Will Learn in This Chapter

Download and
install Android
Studio

Create Android
Project

Compile, build
and run android
apps

Differentiate
between the
module and top-
level Gradle builds

Mange code
versioning

Describe Android
stack and
framework



Part 1: Starting with Android



Let's start this chapter by learning about Android as a mobile operating system and platform and as a framework for developing apps



2.1 A Brief Android History

The story of the Android mobile platform begins in October 2003 when Andy Rubin, Rich Miner, Nick Sears, and Chris White started a company called Android, Inc

The company started with the intention to create an advanced operating system for digital cameras, i.e., use the Linux kernel to create an embedded operating system for digital cameras

Google acquired Android, Inc., and with it acquired the Android operating system in August 2005. People consider this date as the date that Google entered the mobile operating system market.

In November 2007, several tech companies including Google, Sony, HTC, Samsung, and others created a consortium called the Open Handset Alliance to develop open standards for mobile devices

One year later, in October 2008, the first Android phone, the HTC Dream, was launched marking the start of the mobile phones we know today

2.2 Android Is Open Source

Android is a mobile operating system based on the Linux kernel

It is an open-source operating system for mobile devices that has been built upon other open-source projects

As a developer, this means you have access to the source code of the platform, which in turn helps you to better understand how the interface and the various other pieces of the platform work

2.3 Android Libraries

Android provides users with the interface for touch screens

Users can interact with the Android devices by swiping, tapping, pinching, or using the virtual keyboard and voice

The voice access app for Android lets you control your device with spoken commands, i.e., use your voice to open apps, navigate, and edit text hands-free

2.4 Android Popularity

[Mobile Operating System Market Share Worldwide | Statcounter Global Stats](#)

[Mobile App Download Statistics & Usage Statistics \(2023\) - BuildFire](#)

Android is a very popular operating system

Based on Google Play Store's statistics, there are billions of active Android devices, and there are millions of Android apps in the Google Play Store, and these numbers grow daily

Android supports the developer community through documentation and sample codes

The fact that Android is an open-source operating system based on Linux and uses the Java programming language gives Android a unique advantage to becoming a popular platform

The Linux operating system made it easier for device manufacturers to develop drivers for their products



2.5 Android Development Environment

To develop Android mobile apps, in addition to your knowledge of the Java data types, operations, statements, syntax, and program structures, you need to become familiar with a new set of libraries, classes, and interfaces, i.e., you need to learn to use the Android Software Development Kit and Android API

Unlike Java and C++, which have very stable and well-established SDKs and APIs, the Android SDK and API evolve quickly which puts an extra challenge on your learning path

You not only need to learn a new API, but also watch out for the latest updates to avoid your code from becoming obsolete

2.5 Android Development Environment

Mobile devices come with built-in sensors, GPS , WI-FI, Bluetooth, cameras, USBs, screens, and other components and features used for text messaging, finding device orientation, tracking the user's movements, etc

Writing programs to use these new components and features are relatively novel to Java programmers and is something that they need to learn and get used to it

2.6 Android Developer's Skills




If you know the Java programming language, can run applications on the Linux operating system, and have some basic XML and Linux administration knowledge, you are pretty much set and ready to be an Android developer



Knowledge of Java, XML, and Linux are helpful and set you up on the path to becoming an Android developer



The objective of this course is to teach you the skills needed to become a successful Android programmer



2.7 Model View Controller and App development

Android programming involves not only a clear separation between the visual components and the app's computational logic, or business objects, but also using XML elements to represent the visual objects

You can create all the GUI components of your app using XML files

At run-time, the XML files are converted to Java code and linked to the main source code to be executed

The separation between views, logic, and data is essential in the Android app development philosophy and you must master this concept to be able to code for Android devices

2.8 Android's Main Program



For any program to start, there is a need for an entry point into the program



For Java programs, the main method in a class is the entry point



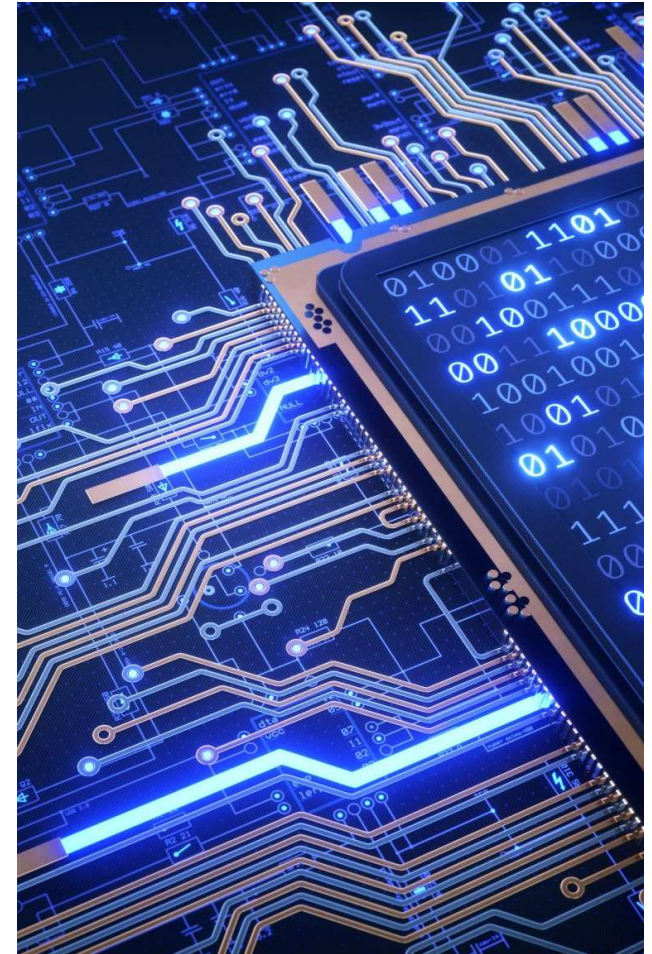
For Web applications, you type the address of the website in the browser which looks for the index.html file to load the page and display it

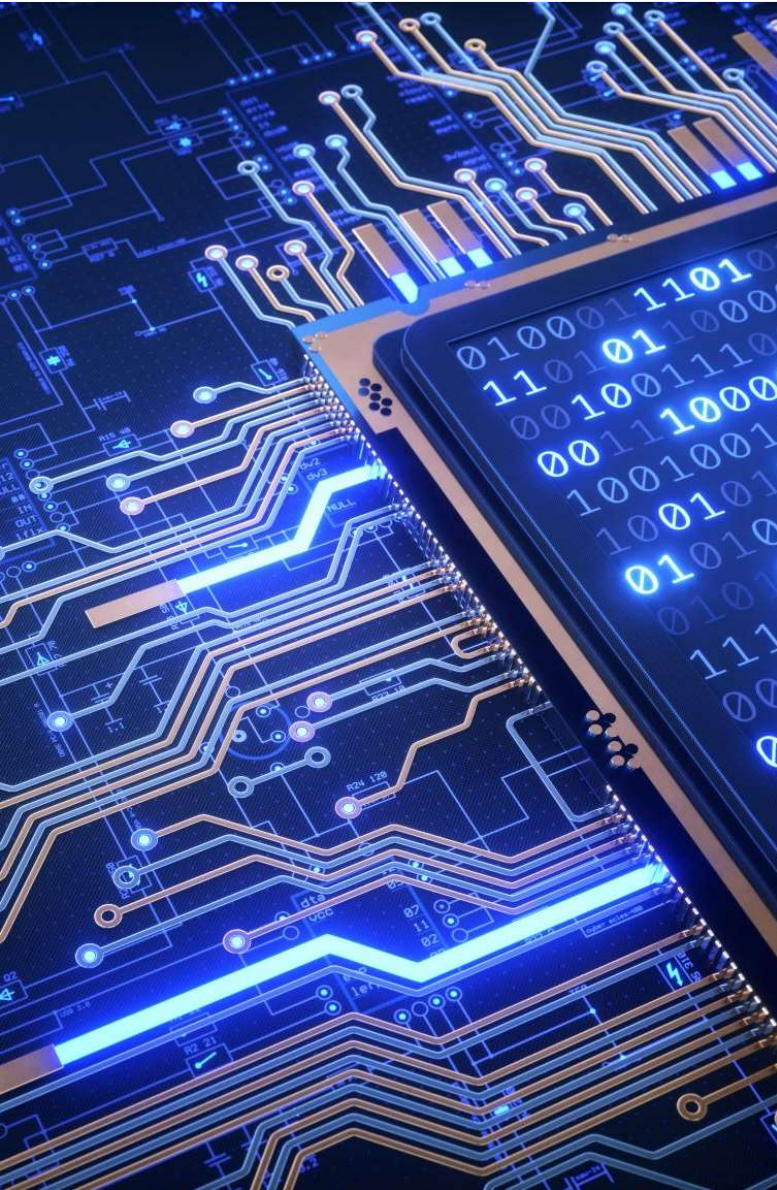


All Android application programs start with the AndroidManifest.xml file

2.9 Java and Android

- Android is an adapted Linux operating system that can run on low-energy mobile touch-screen devices
- At first, Android development was done in Eclipse with some plugins to enable Android app development
- Coding was done in Java and then installed on Android-enabled devices or emulators powered with the Java Virtual Machine for running
- Java is one of the two official languages for Android development, the other language is Kotlin
- Kotlin depends on JVM and can coexist with Java in the same application. Kotlin aims to solve several limitations of Java.





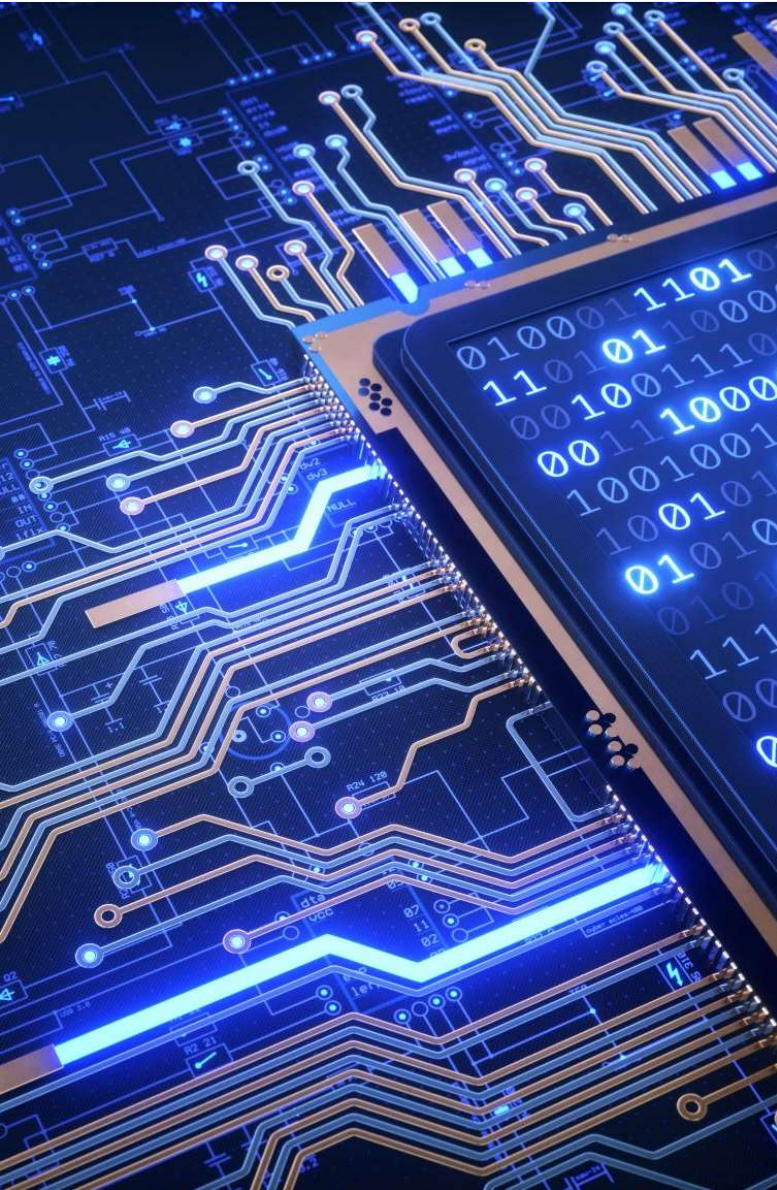
2.9 Java and Android

- Android uses the Java API (Application Programming Interface) and supports most Java features
- Not only does Android use the Java programming language, but app development for Android devices follows the JavaFx application design as well
- Similar to JavaFx, developing Android apps involves using both XML files and code as well as extending an already defined class to render graphical components of the app



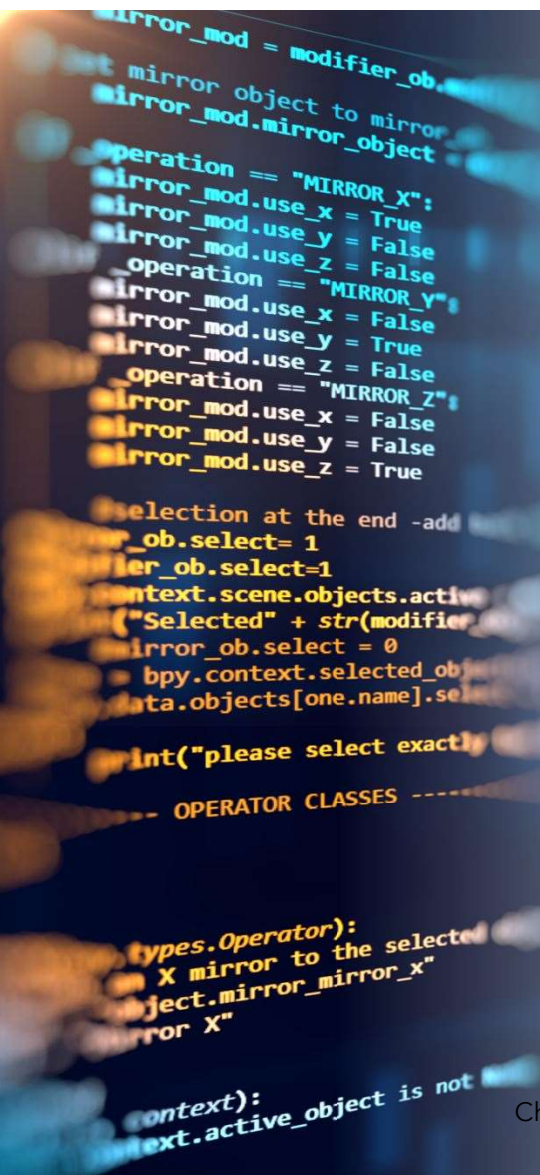
2.10 Why Use Java for Android?

- From a technical perspective, Java is selected as a programming language for Android for three main reasons
- First, Java is a popular programming language. This enables Java developers to create Android apps without having to learn a new programming language
- Second, Java runs inside the Java Virtual Machine which helps Android apps to run on any Android-powered device regardless of the device's manufacturer
- Third, Java is a comprehensive programming language with a large number of libraries, classes, and interfaces that are open source, or mostly open-source, and readily available for use



2.11 Android and Linux

- Android smartphones and tablets contain the Linux kernel to manage active processes and communication between hardware and software components
- Instead of writing the kernel from scratch, Android developers from Google modified the Linux kernel for their devices
- This is possible because Linux is an open-source operating system
- While the Linux kernel is an important component of Android and has helped to build Android, the Android team made many changes to the Linux kernel



Part 2: Download and Install Android Studio and Android SDK

2.12

Download the Android Studio

Android Studio is the official Integrated Development Environment for Android development and is based on IntelliJ IDEA. It is not the only IDE that you can use to develop apps

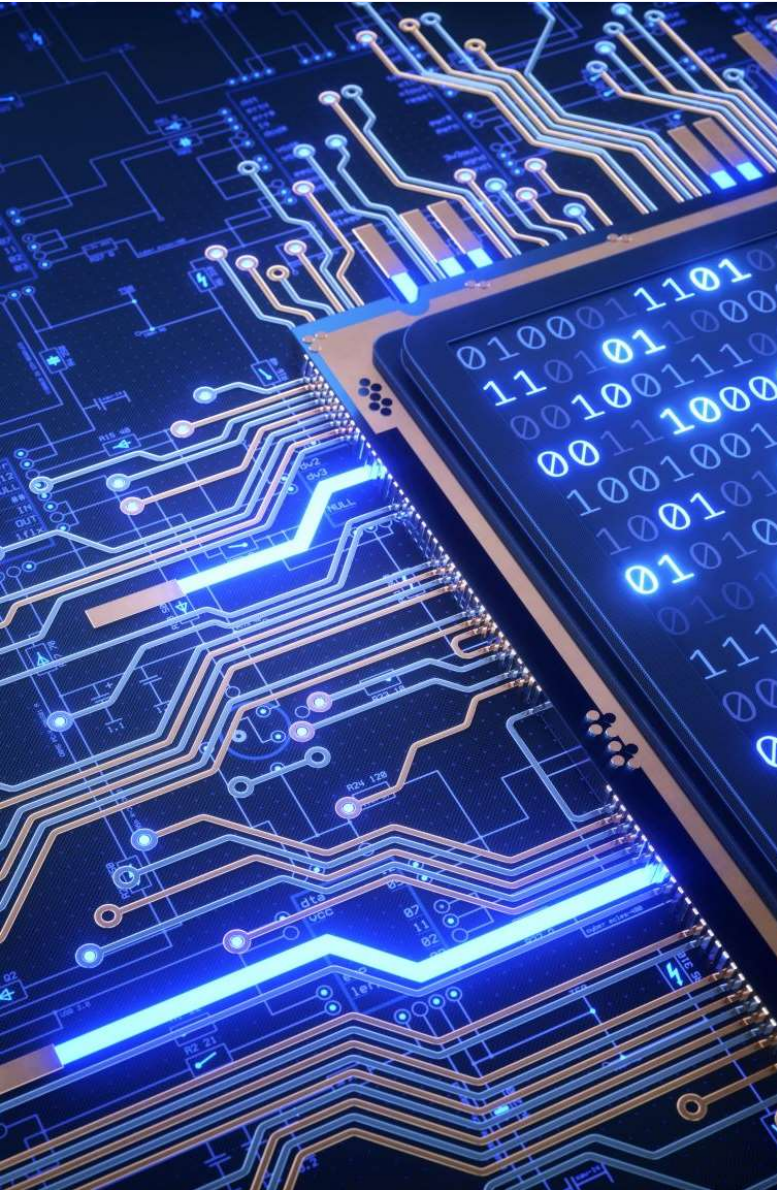
However, Android Studio includes everything you need to start developing Android apps

These include project and activity templates, a layout editor, testing tools, the Gradle build tool, a log console, a debugger, and virtual devices to emulate phones, tablets, watches, and many more plugins

Click or put this download link ([Download Android Studio and SDK tools | Android Developers](#)) into your browser to get started

While native iOS developers must develop on a Mac, with Android, you can develop on Windows, Mac, or Linux

To get started with Android on a Mac, however, you need to download Android Studio for Mac



2.13 Install Android Studio

- Installing Android is straightforward
- After downloading Android Studio, go to your download directory and double click on the executable file you downloaded to open Android Studio
- To complete the installation press “Yes” for the installation process to start as shown in Fig
- The location folder should have enough space to install and run Android projects
- On Windows, a shortcut will be created on the start menu which points to Android Studio executable in the bin folder



2.14 Update Android Files

- If Android Studio is missing updates, a message gets displayed to indicate the updates needed. Click on the displayed link and download all of the updates
- You can push updates to run in the *background* and continue working. You can check for updates yourself by clicking on the Help tab on the Android Studio toolbar and clicking check for updates

2.15 Release Note



Release notes describe new features added, issues resolved, and improvements made to the Android



It is important to read the release notes



Sometimes, Android makes substantive changes and improvements that have an impact on your existing code or the code you are about to write



In cases where the new updates are related to plugins that you are not using, you don't need to update



You can find what is new in a release from this link: (<https://developer.android.com/jetpack/androidx/versions/all-channel>)



2.16 Android SDK

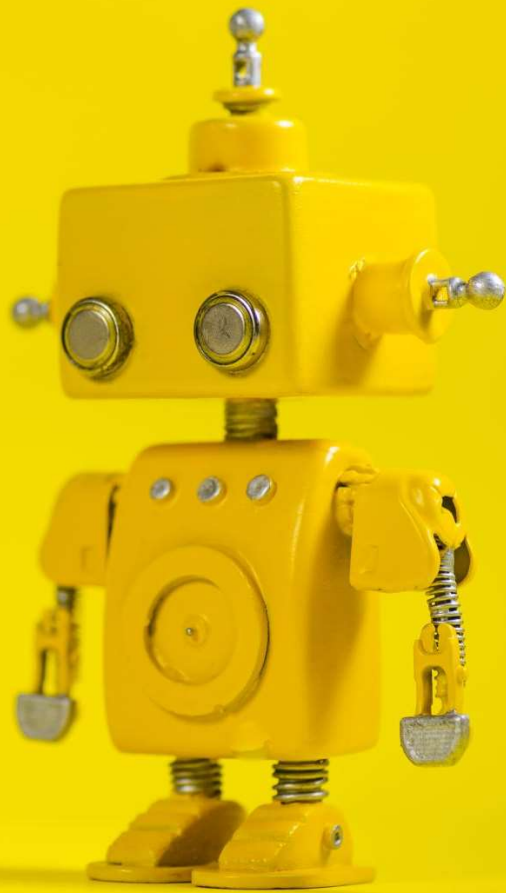
The Android Standard Development Kit (SDK) consists of a large number of Java classes, interfaces, and packages that are essential to developing apps

These are classes and interfaces that you import to your code during development

The SDK is independent of the Android Studio

However, it is more suitable to work with when used with Android Studio or other IDEs such as NetBeans or Eclipse

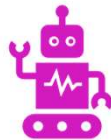
On your machine, the SDK is located at `*~/Android/sdk*`, e.g., `C:\Users\username\AppData\Local\Android\Sdk`



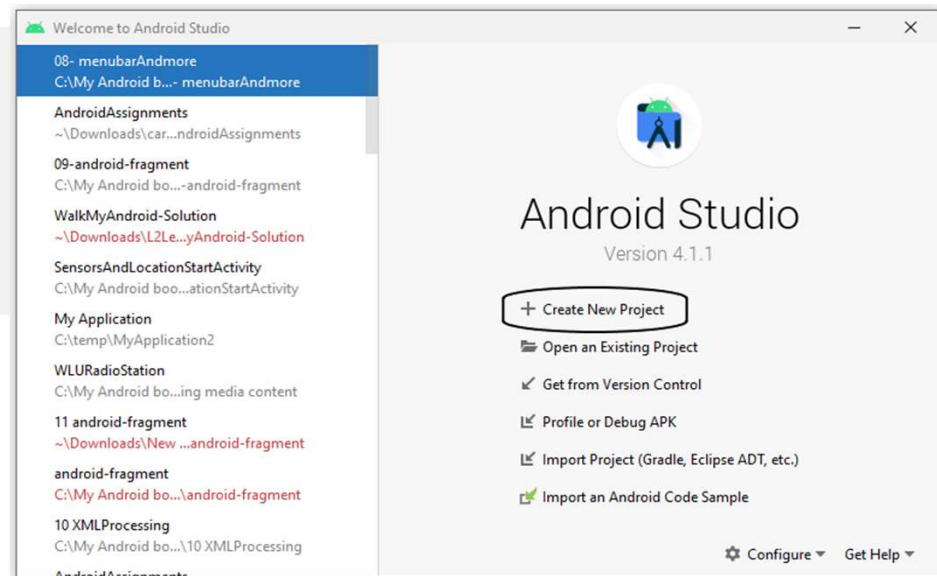
Part 3: Create a New Android Project

- To start a new Android app project for HelloWorld, follow the wizard steps

2.17 Start New Project




Click "Create a new Android Studio project" as shown below



2.18 Select an Activity Template

You need to create one or more Activity (class) for your project



To create an activity, you select an activity template for your app



Activity templates help fast development and best design practices



Android studios come with multiple code templates



To make yourself familiar with the Android app templates, I encourage you to read about Android templates that come with Android studio (<https://developer.android.com/studio/projects/templates>)

2.19 Fill in Application Requirement


Fill in the application name, package name, and project location

The project location is where your project files reside


You also have a choice to select Java or Kotlin as a programming language for your coding

2.20 Define SDK Requirements

For Minimum Required SDK, you can accept the default and click *next* or select the desired API for your app, i.e., define the target device for your app



It is important to choose the correct version of the API for your app



This a design decision, you decide what is best for your app, or you build more than one version of your app to work with different APIs

2.21 Finish the Project Creation



Open the app/java folder and look at the MainActivity.java code



It is made of a package name, some import statements, and a class definition



The file is saved as MainActivity.java which has one overwritten method called onCreate



The meaning of the code components such as AppCompatActivity, setContentView, Bundle, etc will be explained in the coming chapters

Part 4: Compiling and Running Android Apps



How you run your app depends on two things: whether you have an Android device or are using an Android emulator



This chapter shows you how to install and run your app on Android devices as well as on an Android Emulator

2.22 Running HelloWorld on Your Phone



Let's run our code directly on the phone



The biggest advantage of using android devices for development is that it is fast to load and run programs



In contrast, the emulator, which we will discuss next, runs slowly; it's not recommended to use the emulator for the whole book

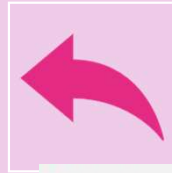


If you are going to be an Android developer, it is time to think about getting an Android device and using Emulator as a backup

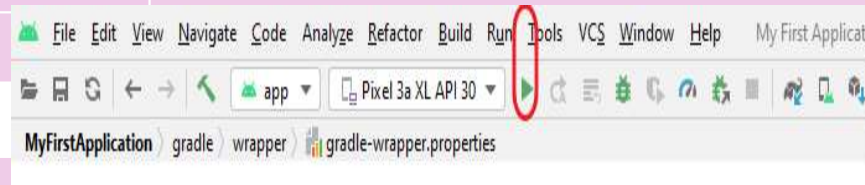


Before you start, plug your phone into your laptop using the USB port. Android Studio installs the Hello World app on your connected phone and starts it

2.23 Running the Android app in Android Studio



Click on the *Run* button on the Android Studio toolbar as shown



The app is downloaded and installed and runs on the phone

2.24 Issues Starting First App



If for some reason your app did not run, i.e., you do not see the app running on your phone, make sure that USB has been enabled



If USB debugging is enabled, unplug the USB cable and plug it in again



Now, after re-launching the app, you should see HelloWorld on the phone

2.25 Running HelloWorld on Emulator



Now, let's run the HelloWorld program on the Android Emulator which emulates the functionality of the phone as best as possible and allows developers to run, test and debug code



The code that runs on the emulator runs unchanged on the real device as well



To run the app, disconnect your phone first, then click the run button in Android Studio



Android Studio will install the app on your AVD and start it

2.26 Setting up the Emulator

In the previous step, we assumed that you have already set up the emulator

There are so many Android devices available that it would be difficult to test your apps on all or many real devices

Android provides a way to create as many emulated devices as you want using the virtual device manager

The AVD Manager allows you to emulate both the hardware and software part of the devices

2.26 Setting up the Emulator

The AVD Manager allows you to emulate both the hardware and software part of the devices

Click on the AVD menu or click on tools | AVD to start the Android Virtual Device Manager

You can also use third-party emulators such as Genymotion that can be faster than the native emulators or work better with your device especially if you have Mac

2.27 Do it yourself



Install Android and AVD

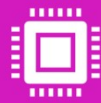


Create your first project



Replace the Hello World text with some other text and familiarize yourself with Android Studio

Part 5: Compiling, Building, and Packaging Technologies



It is important to understand how your Java code is turned into a .dex (Dalvik Executable) file that will end up on your Android devices



Similarly, it is essential to learn about project building and code packaging for publishing



We introduce these topics here to help you understand the Android application structure and project structure that you will create



We will discuss these topics more as the course progresses and wherever we find it necessary

2.28

Compiling Android Code and Using R8 Compiler



You need a Java compiler, *javac*, to compile the Java code of your application.



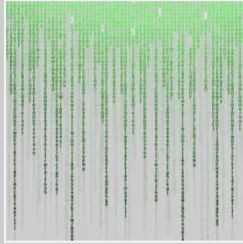
The *javac* is included in the Java development kit (JDK) that you need to download separately from downloading Android Studio



Your code and imported codes from the Android API and custom libraries are compiled.

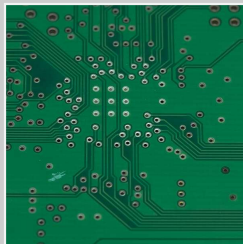
The output of this step is Java bytecode files. This is a normal Java compiling process. You are turning *.java classes into *.class bytecode.

2.28 Compiling Android Code



To make it hard to decompile packaged Java code, currently, the Android Gradle plugin uses the R8 compiler to

1. minimize the size of the code and
2. obscure code by changing method names and applying code encryption



The described steps can be summarized as follows:

Source code (*.java) → using javac → Java Bytecode (*.class) → using Proguard or R8 → optimized Java bytecode (*.classs) → using D8 or Dex → Dalvik Optimized bytecode (*.dex)

2.28 Turning .class into .dex bytecode



bytecodes is converted to Dalvik bytecode and stored in a .dex (Dalvik Executable) file.



The conversion is done by a command-line tool called D8 (DEX compiler).

D8 is a replacement for the DX dexter



Android Studio and the Android Gradle Plugin use D8 to compile Java bytecode into DEX bytecode.



The .dex bytecode is an optimized bytecode format for Android that can be executed by DALVIK or Android runtime (ART).



2.29 packaging dex file

The APK Packager combines the DEX files and compiled resources into a single APK file

The APK is a file format designed for distributing Android applications

It is simply just a ZIP archive file similar to the Java JAR package designed for the distribution of Java applications

In addition to the .dex file, APK contains other files, for example, AndroidManifest.xml

2.29 Android App Bundle

In addition to the APK, Android supports another build format, the **Android App Bundle (AAB) build format**.

The Android App Bundle is Android's new format to build and release apps

The generated bundle includes all your app's compiled code and resources

However, Google Play defers generating the APK to optimize downloading for each device configuration

2.30 Do it Yourself



Create a new project and try to generate a .apk using the build APK steps



Familiarize yourself with creating APK, signed APK, and AAB files

2.31 Install Android Apps

An Android Package Kit (APK)) is like a .exe file for windows PC

The Android operating system uses .apk for the distribution and installation of apps

When you download, or when you get an APK, you're getting an app

We will describe how to install APK files posted online and from your computer



2.32 Install APK From Online

The best way to download apps is to get them from trusted sources such as the Google Play Store

To install APK posted online to your Android device, you need to find the APK file and download it

The app should begin installing on your device

2.33 Install APK From Files

If you happen to have an APK file on your computer, you can install the APK file on your Android device

To do so, you need to connect your computer to your Android device and copy the APK file to it

Find the APK file on your device, tap it, then hit 'install' when you are prompted by the installation process



2.34 From Dalvik to ART Runtime

Android runtime (ART) is a replacement for the Dalvik virtual machine

Before Lollipop, Dalvik used to translate bytecodes to the native code, or machine-level code, files

With ART, the translation from bytecode to machine level code is done during the app installation which makes apps run faster but takes more memory and time when installing them for the first time

2.35 Gradle Build

Android uses the **Gradle** build system to build, compile, and package apps

Gradle does the same job as the Apache Ant and Maven toolkits that you are probably familiar with

Compared to Ant and Maven tools, Gradle is a faster and more flexible building tool

It allows customized builds enabling you to create APKs that can be uploaded to the emulator for demoing

There are other build tools such as **Bazel**, which is used by Google to build all its software, and **Buck**, which is developed and used by Facebook

2.35 Gradle Build

- Gradle is a part of the project structure displayed on Android Studio.
- Gradle creates two build files,
 - the ***module-level build.gradle*** file and
 - ***top-level build.gradle*** file
- Each of these files is used for different purposes. If your app is big or complex, for example, it requires more than one team to develop it, then it makes sense for each team to be specialized and work on one part of the app. In this case, each team will create its own *module-level build.gradle* file for the app.
- *top-level build.gradle* holds configuration information for the entire project. It is stored in the root directory of your project. There is only one top-level build file for each project.

2.36 Software Versioning Using Local or Remote Repositories

Controlling software versions, also known as Source Management Control, is an essential part of the software development process

You need to manage your code versions whether you are writing a simple app for your interest or are a team member working on a large software development project

An open-source version control tool called **Git** is integrated into the Android Studio and can be used to manage code versioning

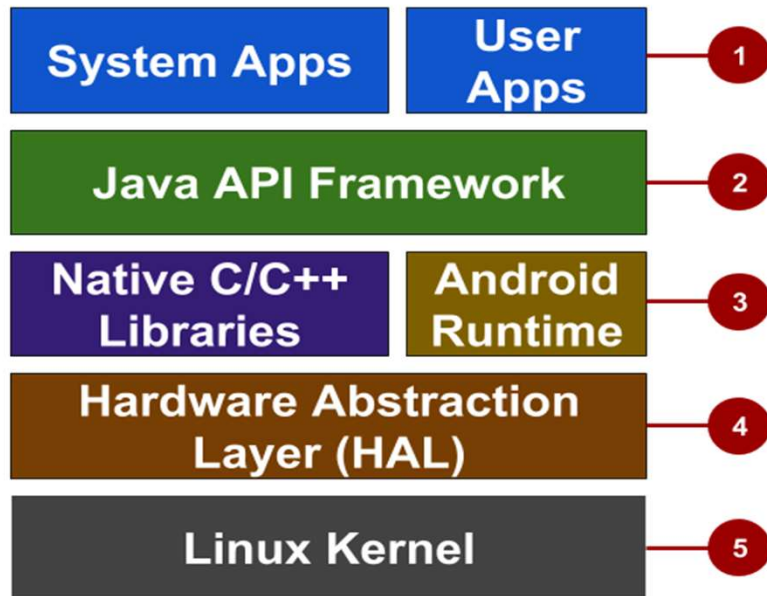
Using Git, you can trace the history of your code changes, coordinate work among several programmers working on the same source code, and if needed, return to previous versions of your code

Git enables you to develop your code separately from the rest of your team

Part 6: Android Stack and Framework



2.37 Android Architecture

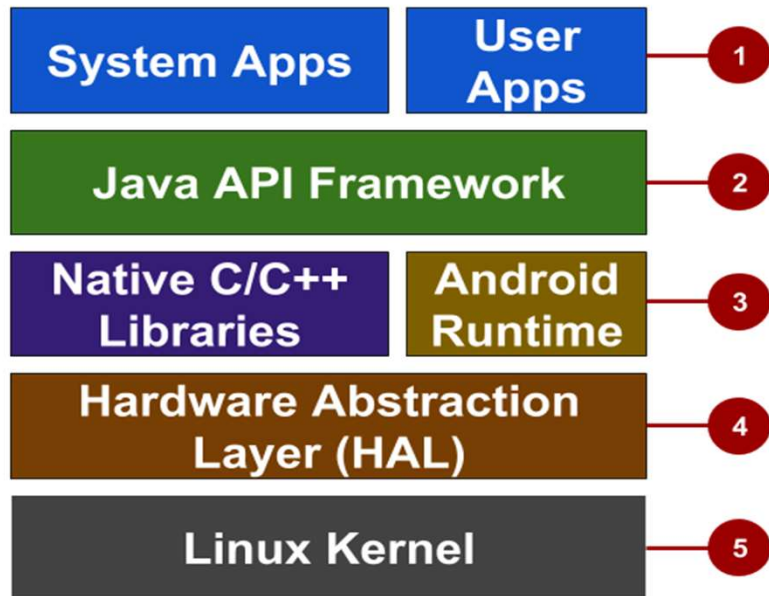


For developing Android apps, you need to have some high-level understanding of Android Architecture. Understanding Android Architecture is an asset that can help you develop high-quality apps

The figure shows the major components of the Android development architecture and operating system, also known as the Android stack

The stack is made of the typical layers that you will find in any system platform

2.37 Android Architecture



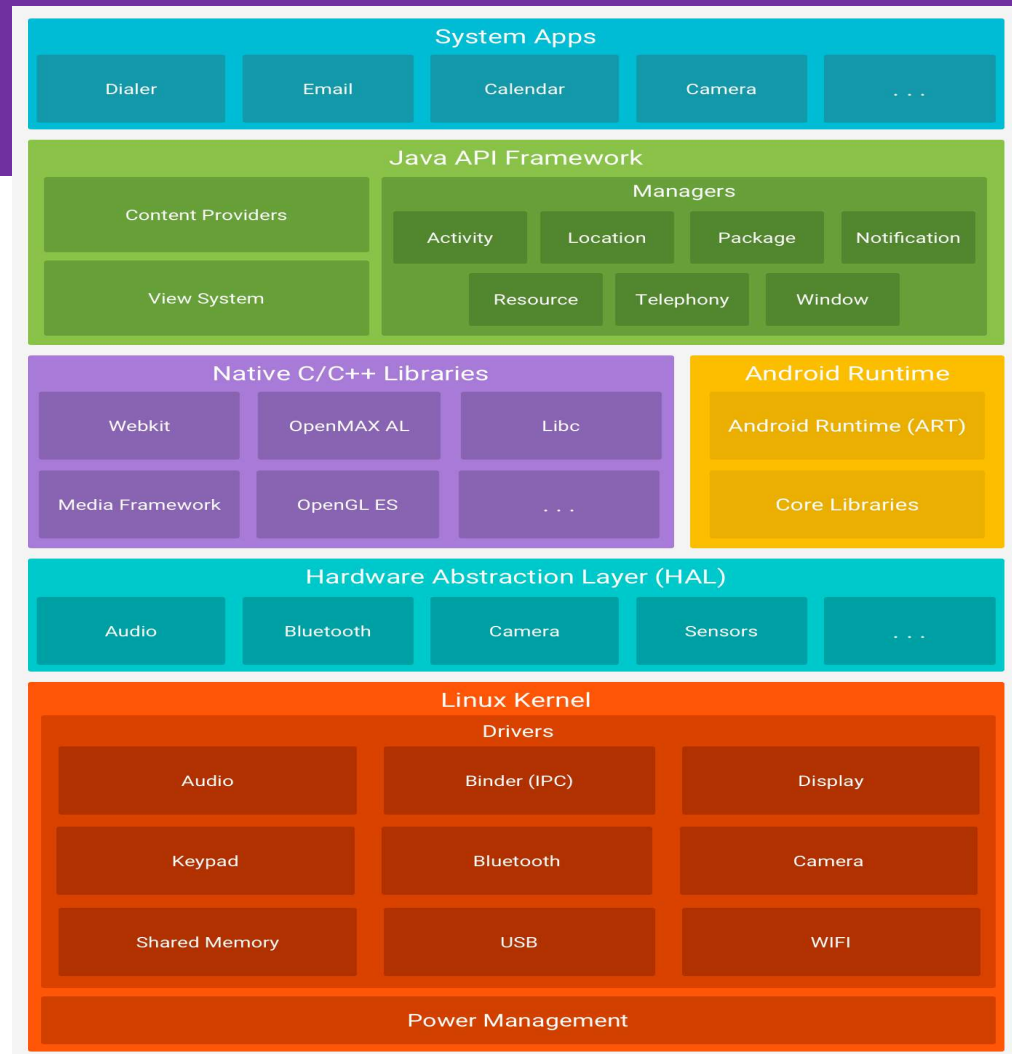
The layers are:

1. the application layer,
2. the class and libraries ,
3. the compiler and run time environment layer,
4. a layer that connects software to hardware interfaces , and finally,
5. the physical hardware layer such as the motherboard, and other network and hardware components

Android stack

- More information on each layer and detailed Android architecture components and modules are listed in the figure shown here
- A short description of some of the components/layers will be described

Chapter 2 Getting Started with Android



2.38 User and System Apps

The apps you will develop will live in Layer1 along with core system apps

This is the top layer of the Android stack

System apps, such as Clock and Calculator, are preinstalled on Android devices

Many system apps can't be uninstalled

2.38 User and System Apps

- Some system apps are critical for device functionality.
- You don't want to disable these critical apps, otherwise, your device will not work
- For example, you can't disallow:
 - Bluetooth, contacts, keychain, keyguard,
 - com.android.launcher, com.android.nfc,
 - com.android.phone, com.android.settings,
 - etc.

2.39 Java API Framework

All the classes, interfaces, and packages you import to your code during app development form this layer.

In addition to Java API, currently, Android supports Kotlin API as well.

The more you know about the Android API, the easier it is to develop Android apps.

2.39 Java API Framework

At the starting point, learning these systems is necessary to be an Android developer

- **View System** is used to build apps' user interfaces , including lists, buttons, menus, etc
- **Resource Manager** is used to accessing non-code resources such as localized strings written in XML, graphics, and layout files
- **Notification Manager** is used to displaying custom alerts to the user
- **Activity Manager** is used for managing the app's lifecycle
- **Content Provider** is used to enable apps to access data from other apps
- **Location Manager** is used for providing location information using data from GPS sensors, cell towers, and Wi-Fi networks

2.40 Native Libraries and Android Runtime

- Each app runs in its process and with its instance of the Android Runtime (ART).
- Before Android Lollipop, Dalvik used to translate bytecodes to the native code, or machine-level code, files
- With ART, the translation from bytecode to machine level code is done during app installation which makes apps run faster but takes more memory and time when installing for the first time

2.40 Native Libraries and Android Runtime

- The core libraries help developers create apps using the C/C++ programming language and using NDK libraries
- This means you'll be writing code that runs directly on the devices' hardware and your app can access physical components such as sensors and cameras directly
- Many components of the Android system such as ART and HAL are built using native libraries
- Using C/C++ libraries mean don't need to use the Java Virtual Machine will give you some advantages. For example, you will have better control over memory allocation, memory cleanup, and the app's performance can be improved
- Using C/C++ libraries are important for performance-intensive apps like games.

2.41 Hardware Abstraction Layer



It is an abstraction layer between Android's physical hardware and its software



It provides ways for data passing from/to hardware devices to the higher-level Java API framework



The HAL consists of multiple libraries, each library implements access for a specific type of hardware components such as the camera or Bluetooth



Standard interfaces that expose device hardware capabilities as libraries

2.42 Linux Kernel

- The basis of the Android platform is the Linux kernel
- The four layers above rely on the Linux kernel for underlying functionalities such as threading, memory management, process management, security, networking, etc
- These include drivers for the camera, keyboard, display, USB, audio, etc
- Device manufacturers can develop hardware drivers for Android easily which in turn has helped the Android devices' popularity
- This is because Linux is an open-source software that can easily be modified to meet hardware needs

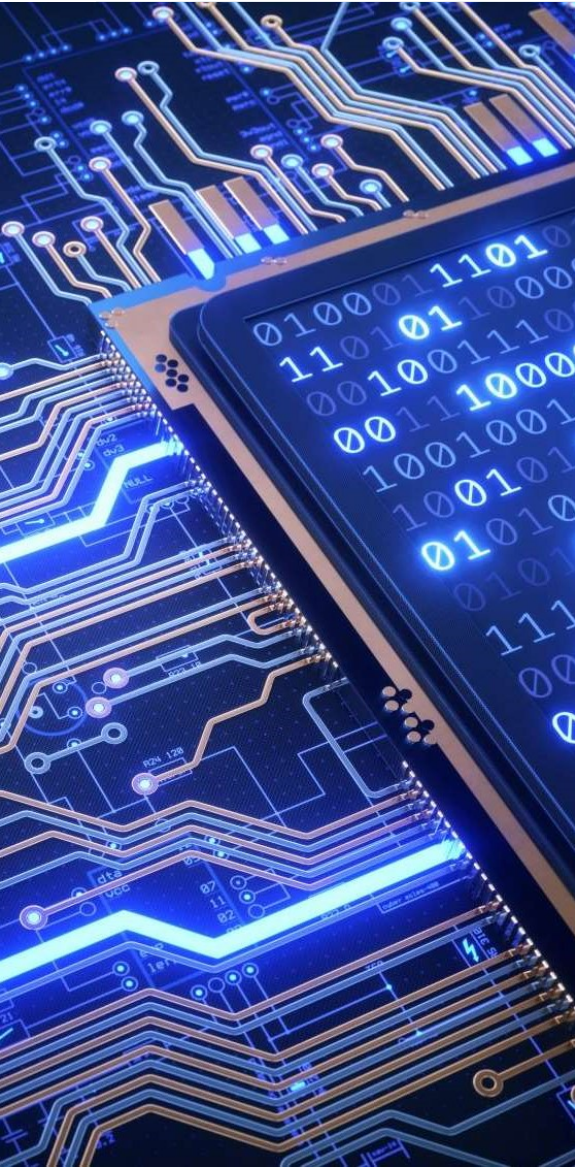


2.7 Chapter Summary

In this chapter, we put together all the nuts and bolts that set you up for developing Android applications

We covered getting and setting up Android IDE, how to run apps on the Android Emulator and the phone, Android SDK components and packages, android architecture and API framework, and other topics

Once you complete reading this chapter, you will be well on your way to start the journey of learning Android application development



Check Your Knowledge

- Some of the fundamental concepts and vocabulary have been covered in this lesson.
- To test your knowledge and your understanding of this chapter you should be able to describe each of these concepts in one or two sentences.

Hardware Abstraction Layer

obscuring code

ProGuard

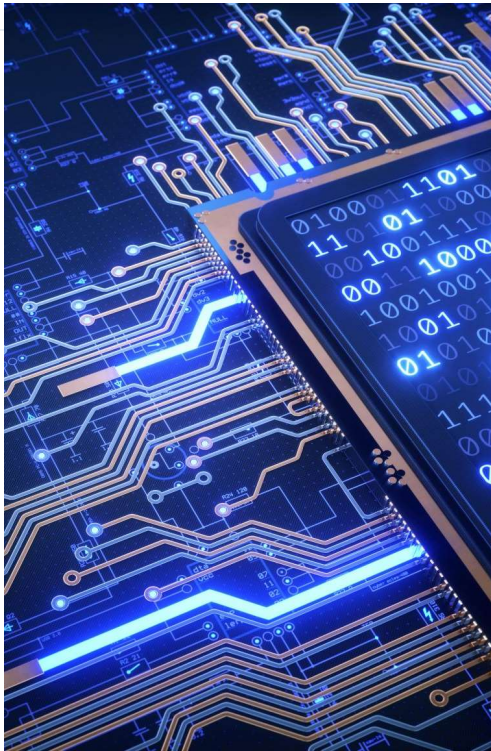
R8

Release note

SDK Manager

SDK Platforms

Check Your Knowledge



.dex file

AndroidX

APK

App Bundle

ART

AVD

Dalvik

Git

GitHub

Gradle



2.8 Further Reading

Android Developers, Docs, Guides, Build a simple user interface, Available:

<https://developer.android.com/guide/components/fundamentals.html#Components>

Configure Your Build with Gradle, Available:

<https://sodocumentation.net/android-gradle/topic/2161/configure-your-build-with-gradle>