Learning Android

Marko Gargenta

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Learning Android
by Marko Gargenta

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editors: Andy Oram and Brian Jepson
Production Editor: Holly Bauer
Copyeditor: Genevieve d'Entremont
Proofreader: Jennifer Knight
Indexer: Jay Marchand
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Printing History:
March 2011: First Edition.

ISBN: 978-1-449-39050-1

[LSI]

1299702297

Table of Contents

Preface

This book sprang from years of delivering the Marakana Android Bootcamp training class to thousands of software developers at some of the largest mobile companies located on four continents around the world. Teaching this class, over time I saw what works and what doesn't. This book is a distilled version of the Android Bootcamp training course that I developed at Marakana and fine-tuned over numerous engagements.

My background is in Java from back before it was even called that. From the beginning, I was very interested in embedded development as a way to program various devices that surround us in everyday life. Because Java primarily took off in web application development, most of my experience in the previous decade has been in building large enterprise systems. Then Android arrived, and once again I became very excited about building software for nontraditional computers. My current interests lie in using Android on devices that may not even resemble a typical phone.

This book teaches anyone who knows Java (or a similar language) how to develop a reasonably complex Android application. I hope you find this book fairly comprehensive and that you find the example-based learning reasonably motivating. The goal of *Learning Android* is to get you to *think* in Android terms.

What's Inside

Chapter 1, Android Overview
: Is an introduction to Android and its history

Chapter 2, The Stack
: Is an overview of the Android operating system and all its parts from a very high level

Chapter 3, Quick Start
: Helps you set up your environment for Android application development

Chapter 4, Main Building Blocks
: Explains the Android components application developers use to put together an app

Chapter 5, Yamba Project Overview
: Explains the Yamba application that we'll build together through this book and use as an example to learn Android's various features

Chapter 6, Android User Interface
: Explains how to build the user interface for your application

Chapter 7, Preferences, the Filesystem, the Options Menu, and Intents
: Covers some of the operating system features that make an application developer's life easier

Chapter 8, Services
: Covers building an Android service to process background tasks

Chapter 9, The Database
: Explains the Android framework's support for the built-in SQLite database and how to use it to persist the data in your own application

Chapter 10, Lists and Adapters
: Covers an important feature of Android that allows large data sets to be linked efficiently to relatively small screens

Chapter 11, Broadcast Receivers
: Explains how to use the publish-subscribe mechanism in Android to respond to various system and user-defined messages

Chapter 12, Content Providers
: Shows how to design a content provider to share data between applications, in this case using it to enable our app widget to display data on the home screen

Chapter 13, System Services
: Introduces various system services that an app developer can tap into

Chapter 14, The Android Interface Definition Language
: Covers building an inter-process communication mechanism to allow for remote access to a service from another application

Chapter 15, The Native Development Kit (NDK)
: Introduces how to write native C code as part of your Android application

Conventions Used in This Book

The following typographical conventions are used in this book:

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

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, data types, and XML entities.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

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

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Using Code Examples

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

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Learning Android* by Marko Gargenta (O'Reilly). Copyright 2011 Marko Gargenta, 978-1-449-39050-1."

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

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, get exclusive access to manuscripts in development, and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

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

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

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

http://oreilly.com/catalog/9781449390501/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://oreilly.com*.

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

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

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

Acknowledgments

This book is truly a result of outstanding teamwork. First, I'd like to thank my editors at O'Reilly, Andy Oram and Brian Jepson. Andy, your comments were spot-on and constructive. Brian, thank you for persuading me to take on writing this book in the first place.

I would like to thank all my technical editors: Dan Bornstein, Hervé Guihot, Frank Maker III, and Bill Schrickel. Thank you for diligently reading my half-baked drafts and providing valuable comments.

This book wouldn't be what it is without field testing it on our numerous clients. You were the true pioneers on the cutting edge of Android, and your projects are all very inspiring. Thank you for your trust.

I'd like to thank my team at Marakana—Aleksandar (Saša) Gargenta, Ken Jones, and Laurent Tonon—for bringing back firsthand feedback from teaching Android Boot-camp courses using the draft of this book. Saša, special thanks to you for sending me back to the drawing board more times than I'd like to admit. This book is probably months past due because of your in-depth technical comments.

And finally, a huge thanks to my wife, Lisa, and daughter, Kylie. I know what a sacrifice it was for you while I was crisscrossing the world working on this material. Thank you for supporting me along the way.

CHAPTER 1

Android Overview

In this chapter, you will learn how Android came about. We'll take a look at its history to help us understand its future. As this mobile environment enters a make-or-break year, we look at the key players in this ecosystem, what motivates them, and what strengths and weaknesses they bring to the table.

By the end of this chapter, you will better understand the ecosystem from a business point of view, which should help clarify the technology choices and how they relate to long-term advantages for various platforms.

Android Overview

Android is a comprehensive open source platform designed for mobile devices. It is championed by Google and owned by Open Handset Alliance. The goal of the alliance is to "accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience." Android is the vehicle to do so.

As such, Android is revolutionizing the mobile space. For the first time, it is a truly open platform that separates the hardware from the software that runs on it. This allows for a much larger number of devices to run the same applications and creates a much richer ecosystem for developers and consumers.

Let's break down some of these buzz words and see what's behind them.

Comprehensive

Android is a comprehensive platform, which means it is a complete software stack for a mobile device.

For developers, Android provides all the tools and frameworks for developing mobile apps quickly and easily. The Android SDK is all you need to start developing for Android; you don't even need a physical phone.

For users, Android just works right out of the box. Additionally, users can customize their phone experience substantially.

For manufacturers, it is the complete solution for running their devices. Other than some hardware-specific drivers, Android provides everything else to make their devices work.

Open Source Platform

Android is an open source platform. The entire stack, from low-level Linux modules all the way to native libraries, and from the application framework to complete applications, is totally open.

More so, Android is licensed under business-friendly licenses (Apache/MIT) so that others can freely extend it and use it for variety of purposes. Even some third-party open source libraries that were brought into the Android stack were rewritten under new license terms.

So, as a developer, you have access to the entire platform source code. This allows you to see how the guts of the Android operating system work. As manufacturer, you can easily port Android OS to your specific hardware. You can also add your own proprietary secret sauce, and you do not have to push it back to the development community if you don't want to.

There's no need to license Android. You can start using it and modifying it today, and there are no strings attached. More so, Android has many hooks at various levels of the platform, allowing anyone to extend it in unforeseen ways.

There are couple of minor low-level pieces of code that are proprietary to each vendor, such as the software stack for the cellular, WiFi, and Bluetooth radios. Android tries hard to abstract those components with interfaces so that vendor-specific code can be managed easily.

Designed for Mobile Devices

Android is a purpose-built platform for mobile devices. When designing Android, the team looked at which mobile device constraints likely were not going to change for the foreseeable future. For one, mobile devices are battery powered, and battery performance likely is not going to get much better any time soon. Second, the small size of mobile devices means that they will always be limited in terms of memory and speed.

These constraints were taken into consideration from the get-go and were addressed throughout the platform. The result is an overall better user experience.

Android was designed to run on all sorts of physical devices. Android doesn't make any assumptions about a device's screen size, resolution, chipset, and so on. Its core is designed to be portable.

History

The history of Android is interesting and offers some perspective on what the future might hold.

These are the key events of the past few years:

- In 2005, Google buys Android, Inc. The world thinks a "gPhone" is about to come out.
- Everything goes quiet for a while.
- In 2007, the Open Handset Alliance is announced. Android is officially open sourced.
- In 2008, the Android SDK 1.0 is released. The G1 phone, manufactured by HTC and sold by the wireless carrier T-Mobile USA, follows shortly afterward.
- 2009 sees a proliferation of Android-based devices. New versions of the operating system are released: Cupcake (1.5), Donut (1.6), and Eclair (2.0 and 2.1). More than 20 devices run Android.
- In 2010, Android is second only to Blackberry as the best-selling smart phone platform. Froyo (Android 2.2) is released and so are more than 60 devices that run it.

In 2005, when Google purchased Android, Inc., the world thought Google was about to enter the smart phone market, and there were widespread speculations about a device called the gPhone.

Google's CEO, Eric Schmidt, made it clear right away that Android's ambitions were much larger than a single phone. Instead, they envisioned a platform that would enable many phones and other devices.

Google's Motivation

Google's motivation for supporting the Android project seems to be having Android everywhere and by doing that, creating a level playing field for mobile devices. Ultimately, Google is a media company, and its business model is based on selling advertising. If everyone is using Android, then Google can provide additional services on top of it and compete fairly. This is unlike the business models of other software vendors who depend on licensing fees.

Although Google does license some proprietary apps, such as Gmail and Maps, and makes some money off the Android market, its primary motivation is still the advertising revenue that those apps bring in.

Open Handset Alliance

For this to be bigger than just Google, Android is owned by the Open Handset Alliance, a nonprofit group formed by key mobile operators, manufacturers, carriers, and others. The alliance is committed to openness and innovation for the mobile user experience.

In practice, the alliance is still very young and many members are still learning to work with each other. Google happens to be putting the most muscle behind the Android project at the moment.

The first version of the Android SDK was released without an actual phone on the market. The point of this is that you don't really need a phone for Android development. There are some exceptions (hardware sensors, telephony, etc.), but for the most part the Android SDK contains everything you'll need for developing on this platform.

Android Versions

Like any software, Android is improved over time, which is reflected in its version numbers. However, the relationship between different version numbers can be confusing. Table 1-1 helps explain that.

Table 1-1. Android versions through Android 2.3

Android version	API level	Nickname
Android 1.0	1	
Android 1.1	2	
Android 1.5	3	Cupcake
Android 1.6	4	Donut
Android 2.0	5	Eclair
Android 2.01	6	Eclair
Android 2.1	7	Eclair
Android 2.2	8	Froyo (frozen yogurt)
Android 2.3	9	Gingerbread
Android 2.3.3	10	Gingerbread
Android 3.0	11	Honeycomb

The Android version number itself partly tells the story of the software platform's major and minor releases. What is most important is the API level. Version numbers change all the time, sometimes because the APIs have changed, and other times because of minor bug fixes or performance improvements.

As application developers, you will want to make sure you know which API level your application is targeting in order to run. That API level will determine which devices can and cannot run your application.

Typically your objective is to have your application run on as many devices as possible. So, with that in mind, try to shoot for an API level that is as low as possible. Keep in mind the distribution of Android versions on real devices out there. Figure 1-1 shows a snapshot of the Android Device Dashboard from mid-2010.

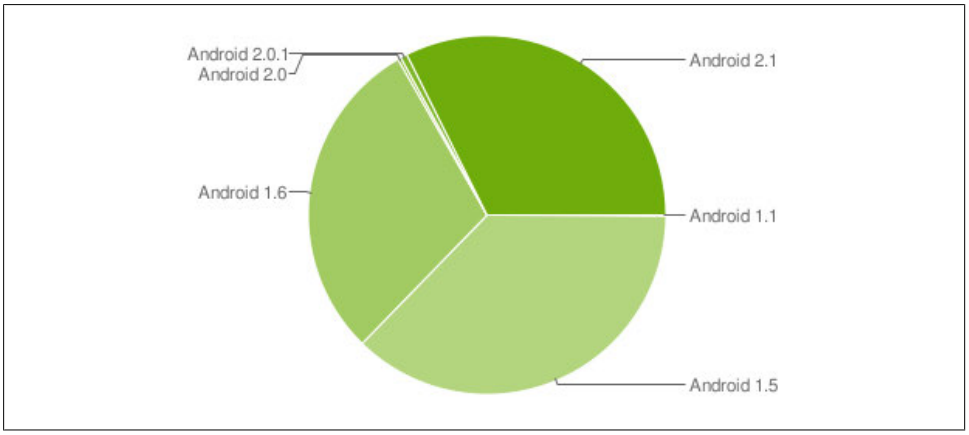

Figure 1-1. Historical Android version distribution through January 2011

You may notice that there are not a lot of users of Android 1.5 and 1.6. You may also notice that not a lot of users have the latest and greatest Android 2.3, but the number of 2.x users is growing. This is because everyone with 1.0 and 1.1 got upgraded over the air (OTA) automatically to 1.5. On the other hand, users who still have devices with Android 1.5 and 1.6 likely will never be able to upgrade to 2.x versions. Their older devices do not have the relevant firmware, and most manufacturers are not planning on releasing firmware upgrades as they are busy working on new models.

With that in mind, you will probably choose 1.6 or 2.0 as your minimum development target, unless you truly need the features of the latest version.

Summary

The Android operating system was designed from the ground up to be a comprehensive open source platform for mobile devices. It is a game-changer in the industry and has enjoyed great success.

In the next chapter, we'll take a look at the entire Android operating system at a high level to gain a technical understanding of how all the pieces fit together.

CHAPTER 2

The Stack

This is the 9,000-foot overview of the Android platform. Although you're concerned primarily with writing Android applications, understanding the layout of the system will help shape your understanding about what you can or cannot do easily with Android.

By the end of this chapter, you'll understand how the whole system works, at least from the high level.

Stack Overview

The Android operating system is like a cake consisting of various layers. Each layer has its own characteristics and purpose. The layers are not cleanly separated but often seep into each other.

When you read through this chapter, keep in mind that I am concerned only with the big picture of the entire system and will get into the nitty-gritty details later on. Figure 2-1 shows the parts of the Android stack.

Linux

Android is built on top of Linux. Linux is a great operating system and the poster child of open source. There are many good reasons for choosing Linux as the base of the Android stack. Some of the main ones are its portability, security, and features.

Portability

Linux is a portable platform that is relatively easy to compile on various hardware architectures. What Linux brings to Android is a level of hardware abstractions. By basing Android on Linux, we don't have to worry too much about underlying hardware features. Most low-level parts of Linux have been written in fairly portable C code, which allows for third parties to port Android to a variety of devices.

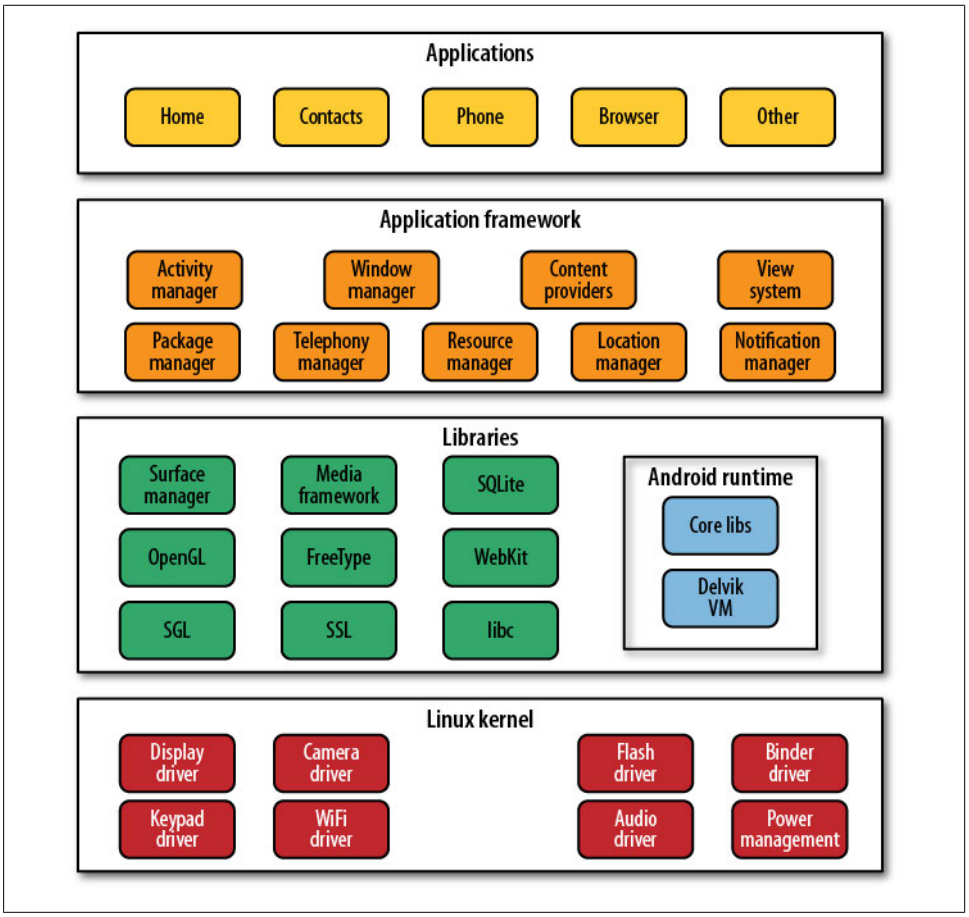

Figure 2-1. Android stack

Security

Linux is a highly secure system, having been tried and tested through some very harsh environments over the decades. Android heavily relies on Linux for security. All Android applications run as separate Linux processes with permissions set by the Linux system. As such, Android passes many security concerns to the underlying Linux system.

Features

Linux comes with a lot of very useful features. Android leverages many of them, such as support for memory management, power management, and networking.

Native Libraries

The native libraries are C/C++ libraries, often taken from the open source community in order to provide necessary services to the Android application layer. Among others, they include:

Webkit
: A fast web-rendering engine used by Safari, Chrome, and other browsers

SQLite
: A full-featured SQL database

Apache Harmony
: An open source implementation of Java

OpenGL
: 3D graphics libraries

OpenSSL
: The secure locket layer

Although many of these libraries are used as-is, one notable exception is Bionic, which is basically a rewritten version of the standard C library. Bionic is used for two reasons:

Technology
: To make it purpose-built for tiny, battery-powered devices

License
: To make it license-friendly for others who might want to adopt it and change it

GNU libc, the default C library for Linux, is licensed under a GPL license, which requires any changes that you release publicly to be pushed back to the open source community. As such, it might not be the most business-friendly open source license when a company wants to keep their derivative work proprietary. Bionic, on the other hand, is licensed under an Apache/MIT license, which doesn't require derivative works to be open sourced.

Dalvik

Dalvik is a purpose-built virtual machine designed specifically for Android, developed by Dan Bornstein and his team at Google.

The Java virtual machine (VM) was designed to be a one-size-fits-all solution, and the Dalvik team felt they could do a better job by focusing strictly on mobile devices. They looked at which constraints specific to a mobile environment are least likely to change in the near future. One of these is battery life, and the other is processing power. Dalvik was built from the ground up to address those constraints.

Another side effect of replacing the Java VM with the Dalvik VM is the licensing. Whereas the Java language, Java tools, and Java libraries are free, the Java virtual machine is not. This was more of an issue back in 2005 when the work on Dalvik started. Nowadays, there are open source alternatives to Sun's Java VM, namely the OpenJDK and Apache Harmony projects.

By developing a truly open source and license-friendly virtual machine, Android yet again provides a full-featured platform that others are encouraged to adopt for a variety of devices without having to worry about the license.

Android and Java

In Java, you write your Java source file, compile it into a Java byte code using the Java compiler, and then run this byte code on the Java VM. In Android, things are different. You still write the Java source file, and you still compile it to Java byte code using the same Java compiler. But at that point, you recompile it once again using the Dalvik compiler to Dalvik byte code. It is this Dalvik byte code that is then executed on the Dalvik VM. Figure 2-2 illustrates this comparison between standard Java (on the left) in Android using Dalvik (on the right).

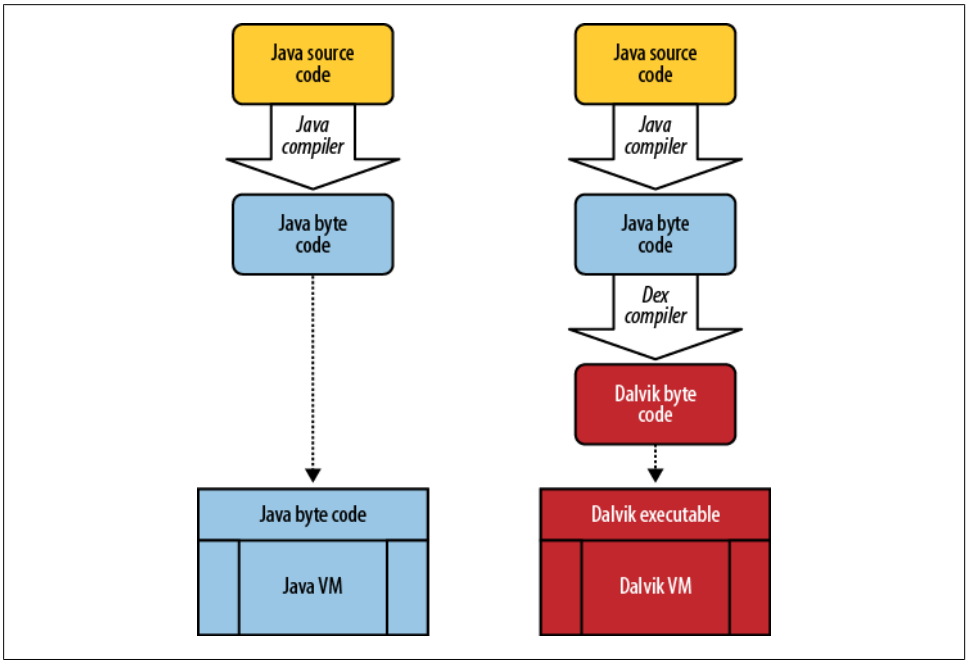

Figure 2-2. Java versus Dalvik

It might sound like you have to do a lot more work with Android when it comes to Java. However, all these compilation steps are automated by tools such as Eclipse or Ant, and you never notice the additional steps.

You may wonder, why not compile straight from Java into the Dalvik byte code? There are a couple of good reasons for the extra steps. Back in 2005, when work on Dalvik started, the Java language was going through frequent changes, but the Java byte code was more or less set in stone. So, the Android team chose to base Dalvik on Java byte code instead of Java source code.

A side effect of this is that in theory you could write Android applications in any other language that compiles down to Java byte code. For example, you could use Python or Ruby. I say "in theory" because in practice the appropriate libraries that are part of the SDK would need to be available. But it is likely that the open source community will come up with a solution to that in the future.

Another thing to keep in mind is that Android Java is a nonstandard collection of Java classes. Java typically ships in:

Java Standard Edition
: Used for development on basic desktop-type applications

Java Enterprise Edition (aka J2EE or JavaEE)
: Used for development of enterprise applications

Java Micro Edition (aka J2ME or JavaME)
: Java for mobile applications

Android's Java set of libraries is closest to Java Standard Edition. The major difference is that Java user interface libraries (AWT and Swing) have been taken out and replaced with Android-specific user interface libraries. Android also adds quite a few new features to standard Java while supporting most of Java's standard features. So, you have most of your favorite Java libraries at your disposal, plus many new ones.

Application Framework

The application framework is a rich environment that provides numerous services to help you, the app developer, get your job done. This is the best-documented and most extensively covered part of the platform because it is this layer that empowers developers to get creative and bring fantastic applications to the market.

In the application framework layer, you will find numerous Java libraries specifically built for Android. You will also find many services (or *managers*) that provide the ecosystem of capabilities your application can tap into, such as location, sensors, WiFi, telephony, and so on.