*Making Everything Easier!*™

# Java® Programming for Android™ Developers

FOR DUMMIES®

A Wiley Brand

## Learn to:

- Create an Android program from start to finish
- Master basic Java development concepts and techniques
- Handle programming challenges
- Assemble the pieces and debug your app


**Barry Burd**
*Author of* Java For Dummies

# Get More and Do More at Dummies.com®


Start with **FREE** Cheat Sheets

Cheat Sheets include

- Checklists
- Charts
- Common Instructions
- And Other Good Stuff!

**To access the Cheat Sheet created specifically for this book, go to**
***www.dummies.com/cheatsheet/javaprogrammingforandroiddevelopers***

## Get Smart at Dummies.com

Dummies.com makes your life easier with 1,000s of answers on everything from removing wallpaper to using the latest version of Windows.

Check out our

- Videos
- Illustrated Articles
- Step-by-Step Instructions

Plus, each month you can win valuable prizes by entering our Dummies.com sweepstakes. *

Want a weekly dose of Dummies? Sign up for Newsletters on

- Digital Photography
- Microsoft Windows & Office
- Personal Finance & Investing
- Health & Wellness
- Computing, iPods & Cell Phones
- eBay
- Internet
- Food, Home & Garden

## Find out "HOW" at Dummies.com

**Sweepstakes not currently available in all countries; visit Dummies.com for official rules.*


Java® Programming for Android™ Developers
FOR DUMMIES®
A Wiley Brand

# Java® Programming for Android™ Developers


**by Barry Burd**

FOR DUMMIES®
A Wiley Brand

**Java® Programming for Android™ Developers For Dummies®**

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



Library of Congress Control Number: 2013948033

ISBN 978-1-118-50438-3 (pbk); ISBN 978-1-118-61212-5 (ebk); ISBN 978-1-118-61214-9 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance

# Table of Contents

# Introduction

Android is everywhere. In mid-2013, Android ran on 53 percent of all smartphones in the United States and on 80 percent of all smartphones worldwide.[1] In a study that spans the Americas, Europe, Asia, and the Middle East, GlobalWebIndex reports that Android tablets outnumber iPads by 34 million.[2] More than a million apps are available for download at the Google Play store (double the number of apps that were available in May 2012).[3] And more than 9 million developers write code using Java, the language that powers Android devices.[4]

If you read this book in a public place (on a commuter train, at the beach, or on the dance floor at the Coyote Ugly saloon, for example), you can read proudly, with a chip on your shoulder and with your head held high. Android is hot stuff, and you're cool because you're reading about it.

## How to Use This Book

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

[1]See `www.kantarworldpanel.com/global/News/news-articles/Samsung-nears-50-share-across-Europe-as-Apple-powers-back-in-the-US` and `http://www.idc.com/getdoc.jsp?containerId=prUS24257413`.

[2]See `www.globalwebindex.net/android-tablets-dominate-q1-mobile-market`.

[3]See `www.androidguys.com/2013/07/24/sundar-pichai-there-are-now-more-than-1-million-android-apps`.

[4]See `www.java.com/en/about`.

In general, my advice is this:

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

# Conventions Used in This Book

Almost every technically themed book starts with a little typeface legend, and *Java Programming For Android Developers For Dummies* is no exception. What follows is a brief explanation of the typefaces used in this book:

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

  ```
  public void Anyname
  ```

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

# What You Don't Have to Read

Pick the first chapter or section that has material you don't already know and start reading there. Of course, you may hate making decisions as much as I do. If so, here are some guidelines you can follow:

- **If you already know what kind of an animal Java is and you don't care what happens behind the scenes when an Android app runs:** Skip Chapter 1 and go straight to Chapter 2. Believe me — I won't mind.
- **If you already know how to get an Android app running:** Skip Part I and start with Part II.

- **If you have experience writing computer programs in languages other than C and C++:** Start with Part II. You'll probably find Part II to be easy reading. When you get to Part III, it'll be time to dive in.
- **If you have experience writing computer programs in C or C++:** Skim Part II and start reading seriously in Part III. (Java is a bit different from C++ in the way it handles classes and objects.)
- **If you have experience writing Java programs:** Come to my house and help me write *Java Programming For Android Developers For Dummies,* 2nd Edition.

If you want to skip the sidebars and the paragraphs with Technical Stuff icons, please do. In fact, if you want to skip anything at all, feel free.

# Foolish Assumptions

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

- **I assume that you have access to a computer.** Access to an Android device is helpful but not absolutely necessary! All the software you need in order to test Android apps on a laptop or desktop computer is freely available. You simply download, install, and get going.
- **I assume that you can navigate your computer's common menus and dialog boxes.** You don't have to be a Windows, Linux, or Macintosh power user, but you should be able to start a program, find a file, put a file into a certain directory — that sort of thing. Much of the time, when you follow the instructions in this book, you're typing code on the keyboard, not pointing and clicking the mouse.

  On those occasions when you need to drag and drop, cut and paste, or plug and play, I guide you carefully through the steps. But your computer may be configured in any of several billion ways, and my instructions may not quite fit your special situation. When you reach one of these platform-specific tasks, try following the steps in this book. If the steps don't quite fit, consult a book with instructions tailored to your system. If you can't find such a book, send me an e-mail. (My address appears later in the Introduction.)
- **I assume that you can think logically.** That's all there is to application development — thinking logically. If you can think logically, you've got it made. If you don't believe that you can think logically, read on. You may be pleasantly surprised.

- **I make very few assumptions about your computer programming experience (or your lack of such experience).** In writing this book, I've tried to do the impossible: make the book interesting for experienced programmers yet accessible to people with little or no programming experience. This means that I don't assume any particular programming background on your part. If you've never created a loop or indexed an array, that's okay.

  On the other hand, if you've done these things (maybe in Visual Basic, COBOL, or C++), you'll discover some interesting plot twists in Java. The creators of Java took the best ideas from object-oriented programming, streamlined them, reworked them, and reorganized them into a sleek, powerful way of thinking about problems. You'll find many new, thought-provoking features in Java. As you find out about these features, many of them will seem quite natural to you. One way or another, you'll feel good about using Java.

# How This Book Is Organized

This book is divided into subsections, which are grouped into sections, which come together to make chapters, which are lumped, finally, into five parts (like one of those Russian *matryoshka* dolls). The parts of the book are described here.

## Part 1: Getting Started with Java Programming for Android Developers

Part I covers all the nuts and bolts. It introduces you to the major ideas behind Java and Android software development and walks you through the installation of the necessary software products. You also run a few simple Java and Android programs.

The instructions in these chapters cover both Windows and Macintosh computers. They cover many computer configurations, including some not-so-new operating system versions, the differences between 32-bit systems and 64-bit systems, and situations in which you already have some form of Java on your computer. But installing software is always tricky, and you might have a few hurdles to overcome. If you do, check the end of this chapter for ways to reach me (the author) and get some quick advice. (Yes, I answer e-mails, tweets, Facebook posts, and notes sent by carrier pigeons.)

## Part II: Writing Your Own Java Programs

Chapters 5 through 8 cover Java's basic building blocks. These chapters describe the things you need to know so that you can get your computer humming along.

If you've written programs in Visual Basic, C++, or in any another language, some of the material in Part II may be familiar to you. If so, you can skip some sections or read this stuff quickly. But don't read *too* quickly. Java is a little different from some other programming languages, especially in the features I describe in Chapter 6.

## Part III: Working with the Big Picture: Object-Oriented Programming

Part III has some of my favorite chapters. This part covers the all-important topic of object-oriented programming. In these chapters, you find out how to map solutions to big problems. (Sure, the examples in these chapters aren't big, but the examples involve big ideas.) You discover, in bite-worthy increments, how to design classes, reuse existing classes, and construct objects.

Have you read any of those books that explain object-oriented programming in vague, general terms? I'm very proud to say that *Java Programming for Android Developers For Dummies* isn't like that. In this book, I illustrate each concept with a simple-yet-concrete program example.

## Part IV: Powering Android with Java Code

If you've tasted some Java and want more, you can find what you need in this part of the book. This part's chapters are devoted to details — the things you don't see when you first glance at the material. This part includes some fully functional Android apps. So, after you read the earlier parts and write some programs on your own, you can dive in a little deeper by reading Part IV.

## Part V: The Part of Tens

In The Part of Tens, which is a little Java candy store, you can find lists — lists of tips for avoiding mistakes, tracking down resources, and finding all kinds of interesting goodies.

## More on the web!

You've read the *Java Programming For Android Developers* book, seen the *Java Programming For Android Developers* movie, worn the *Java Programming for Android Developers* T-shirt, and eaten the *Java Programming for Android Developers* candy. What more is there to do?

That's easy. Just visit this book's website: `www.allmycode.com/Java4Android`. There you can find updates, comments, additional information, and answers to commonly asked questions from readers. You can also find a small chat application for sending me quick questions when I'm online. (When I'm not online, you can contact me in other ways. See the end of this chapter for more info.)

# Icons Used in This Book

If you could watch me write this book, you'd see me sitting at my computer, talking to myself. I say each sentence in my head. Most of the sentences I mutter several times. When I have an extra thought, a side comment, or something else that doesn't belong in the regular stream, I twist my head a little bit. That way, whoever's listening to me (usually nobody) knows that I'm off on a momentary tangent.

Of course, in print, you can't see me twisting my head. I need some other way to set a side thought in a corner by itself. I do it with icons. When you see a Tip icon or a Remember icon, you know that I'm taking a quick detour.

Here's a list of icons that I use in this book:


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


Everyone makes mistakes. Heaven knows that I've made a few in my time. Anyway, when I think people are especially prone to make a mistake, I mark the text with a Warning icon.


*Question:* What's stronger than a tip but not as strong as a warning?

*Answer:* A Remember icon.


"If you don't remember what *such-and-such* means, see *blah-blah-blah,*" or "For more information, read *blahbity-blah-blah.*"


This icon calls attention to useful material that you can find online. (You don't have to wait long to see one of these icons. I use one at the end of this introduction!)


Occasionally, I run across a technical tidbit. The tidbit may help you understand what the people behind the scenes (the people who created Java) were thinking. You don't have to read it, but you may find it useful. You may also find the tidbit helpful if you plan to read other (geekier) books about Java and Android.

# Beyond the Book

We have written a lot of extra content that you won't find in this book. Go online to find the following:

- **Dummies.com online articles:** Be sure to check out `www.dummies.com/extras/javaprogrammingforandroiddevelopers` for additional online content dealing with Java and Android app development. Here you'll find examples of delightfully weird code, a disquisition on classes and objects, a quick look at using Android Asset Studio, an additional Parts of Ten chapter, and much more. And, if we have to post any updates to this edition of *Java Programming for Android Developers For Dummies*, here's where you'd find them.
- **The Cheat Sheet for this book is at** `www.dummies.com/cheatsheet/javaprogrammingforandroiddevelopers`

# Where to Go from Here

If you've gotten this far, you're ready to start reading about Java and Android application development. Think of me (the author) as your guide, your host, your personal assistant. I do everything I can to keep things interesting and, most importantly, to help you understand.


If you like what you read, send me a note. My e-mail address, which I created just for comments and questions about this book, is `java4android@allmycode.com`. If e-mail and chat aren't your favorites, you can reach me instead on Twitter (`@allmycode`) and on Facebook (`/allmycode`). And don't forget — for the latest updates, visit this book's website. The site's address is `www.allmycode.com/java4android`.

# Part I

# Getting Started with Java Programming for Android Developers



Visit www.dummies.com for great *For Dummies* content online.

## In this part . . .

- Downloading the software
- Installing Java and Android
- Testing Android apps on your computer

# Chapter 1

# All about Java and Android

### *In This Chapter*

- The consumer's view of the Android ecosystem
- The ten-cent tour of Java and Android technologies

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

In 2007, a group of 34 companies formed the Open Handset Alliance. Its task is "to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience"; its primary project is *Android,* an open, free operating system based on the Linux operating system kernel.

Though HTC released the first commercially available Android phone near the end of 2008, in the United States the public's awareness of Android and its potential didn't surface until early 2010.

As I sit and write in mid-2013, Mobile Marketing Watch reports more than 50 billion downloads from the Google Play app store.[1] Android developers earned more from their apps in the first half of 2013 than in all of 2012. And according to *Forbes*, Google paid approximately $900 million to Android developers during the 12-month period starting in mid-2012.[2] The pace is accelerating.

[1]See `www.mobilemarketingwatch.com/google-play-tops-50-billion-app-downloads-34516/`.

[2]See `www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/`.

## The Consumer Perspective

A consumer considers the alternatives:

- **Possibility #1: No mobile phone.**

  *Advantages:* Inexpensive; no interruptions from callers.

  *Disadvantages:* No instant contact with friends and family; no calls to services in case of emergencies.

- **Possibility #2: A feature phone.**

  This type of mobile phone isn't a smartphone. Though no official rule defines the boundary between feature phone and smartphone, a feature phone generally has an inflexible menu of Home screen options compared with a smartphone's "desktop" of downloaded apps.

  *Advantage:* Less expensive than a smartphone.

  *Disadvantages:* Less versatile than a smartphone, not nearly as cool as a smartphone, and nowhere near as much fun as a smartphone.

- **Possibility #3: An iPhone.**

  *Advantages:* Great-looking graphics.

  *Disadvantages:* Little or no flexibility with the single-vendor iOS operating system; only a handful of models to choose from.

- **Possibility #4: A Windows phone, a BlackBerry, or another non-Android, non-Apple smartphone**

  *Advantage:* Having a smartphone without having to belong to a crowd.

  *Disadvantage:* The possibility of owning an orphan product when the smartphone wars come to a climax.

- **Possibility #5: An Android phone**

  *Advantages:* Using a popular, open platform with lots of industry support and powerful market momentum; writing your own software and installing it on your own phone (without having to post the software on a company's website); publishing software without having to face a challenging approval process.

  *Disadvantages:* Security concerns when using an open platform; dismay when iPhone users make fun of your phone.

For me, Android's advantages far outweigh its possible disadvantages. And you're reading a paragraph from *Java Programming For Android Developers For Dummies*, so you're likely to agree with me.

# The Many Faces of Android

Version numbers can be tricky. My PC's model number is T420s. When I download the users' guide, I download one guide for any laptop in the T400 series. (No guide specifically addresses the T420, let alone the T420s.) But when I have driver problems, knowing that I have a T420s isn't good enough. I need drivers that are specific to my laptop's seven-digit model number. The moral to this story: What constitutes a "version number" depends on who's asking for the number.

With that in mind, you can see a history of Android versions in Figure 1-1.

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

- **The platform number is of interest to the consumer and to the company that sells the hardware.**

  If you're buying a phone with Android 4.2.2, for example, you might want to know whether the vendor will upgrade your phone to Android 4.3.

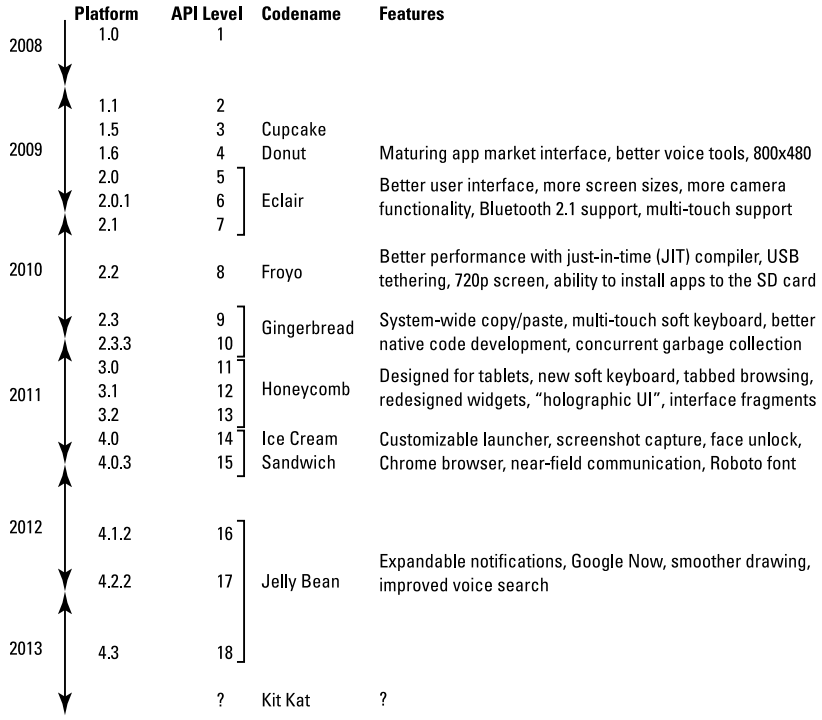


**Figure 1-1:** Versions of Android.

- **The API level (also known as the SDK version) is of interest to the Android app developer.**

  For example, the word `MATCH_PARENT` has a specific meaning in Android API Levels 8 and higher. You might type `MATCH_PARENT` in code that uses API Level 7. If you do (and if you expect `MATCH_PARENT` to have that specific meaning), you'll get a nasty-looking error message.

  
  

  You can read more about the Application Programming Interface (API) in Chapter 2. For more information about the use of Android's API levels (SDK versions) in your code, see Chapter 4.

- **The code name is of interest to the creators of Android.**

  A *code name* refers to the work done by the creators of Android to bring Android to the next level. Picture Google's engineers working for months behind closed doors on Project Cupcake, and you'll be on the right track.


An Android version may have variations. For example, plain-old Android 2.2 has an established set of features. To plain-old Android 2.2 you can add the Google APIs (thus adding Google Maps functionality) and still be using platform 2.2. You can also add a special set of features tailored for the Samsung Galaxy Tab.

As a developer, your job is to balance portability with feature-richness. When you create an app, you specify a target Android version and a minimum Android version. (You can read more about this topic in Chapter 4.) The higher the version, the more features your app can have. But on the flip side, the higher the version, the fewer devices that can run your app.

# The Developer Perspective

Android is a multifaceted beast. When you develop for the Android platform, you use many toolsets. This section gives you a brief rundown.

## Java

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