

GLOBAL
EDITION



Android™

How to Program

SECOND EDITION



Paul Deitel • Harvey Deitel • Abbey Deitel

ALWAYS LEARNING

PEARSON

ONLINE ACCESS

Thank you for purchasing a new copy of ***Android™ How to Program, Second Edition***. Your textbook includes 12 months of prepaid access to the book's Companion Website. This prepaid subscription provides you with full access to the following student support areas:

- Source code
- Premium web chapters

**Use a coin to scratch off the coating and reveal your student access code.
Do not use a knife or other sharp object as it may damage the code.**

To access the ***Android How to Program, Second Edition***, Companion Website for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

- 1.** Go to **www.pearsonglobaleditions.com/deitel**
- 2.** Click on **Companion Website**.
- 3.** Click on the **Register** button.
- 4.** On the registration page, enter your student access code* found beneath the scratch-off panel. Do not type the dashes. You can use lower- or uppercase.
- 5.** Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
- 6.** Once your personal Login Name and Password are confirmed, you can begin using the *Android How to Program* Companion Website!

To log in after you have registered:

You only need to register for this Companion Website once. After that, you can log in any time at **www.pearsonglobaleditions.com/deitel** by providing your Login Name and Password when prompted.

*Important: The access code can only be used once. This subscription is valid for 12 months upon activation and is not transferable. If this access code has already been revealed, it may no longer be valid. If this is the case, you can purchase a subscription by going to **www.pearsonglobaleditions.com/deitel** going to the Android book and following the on-screen instructions.

Android™

HOW TO PROGRAM

SECOND EDITION
Global Edition

Deitel® Series Page

How To Program Series

Android How to Program, 2/e
C++ How to Program, 9/E
C How to Program, 7/E
Java™ How to Program, 10/E
Java™ How to Program, Late Objects Version, 10/E
Internet & World Wide Web How to Program, 5/E
Visual C++® 2008 How to Program, 2/E
Visual Basic® 2012 How to Program, 6/E
Visual C#® 2012 How to Program, 5/E

Simply Series

Simply C++: An App-Driven Tutorial Approach
Simply Java™ Programming: An App-Driven Tutorial Approach
Simply C#: An App-Driven Tutorial Approach
Simply Visual Basic® 2010: An App-Driven Approach, 4/E

CourseSmart Web Books

www.deitel.com/books/CourseSmart/
C++ How to Program, 8/E and 9/E
Simply C++: An App-Driven Tutorial Approach
Java™ How to Program, 9/E and 10/E
Simply Visual Basic 2010: An App-Driven Approach, 4/E

(continued from previous column)

Visual Basic® 2012 How to Program, 6/E
Visual Basic® 2010 How to Program, 5/E
Visual C#® 2012 How to Program, 5/E
Visual C#® 2010 How to Program, 4/E

Deitel® Developer Series

Android for Programmers: An App-Driven Approach, 2/e, Volume 1
C for Programmers with an Introduction to C11
C++11 for Programmers
C# 2012 for Programmers
Dive Into® iOS 6 for Programmers: An App-Driven Approach
Java™ for Programmers, 2/e
JavaScript for Programmers

LiveLessons Video Learning Products

www.deitel.com/books/LiveLessons/
Android App Development Fundamentals,
C++ Fundamentals
Java™ Fundamentals
C# 2012 Fundamentals
C# 2010 Fundamentals
iOS® 6 App Development Fundamentals
JavaScript Fundamentals
Visual Basic Fundamentals

To receive updates on Deitel publications, Resource Centers, training courses, partner offers and more, please join the Deitel communities on

- Facebook®—[facebook.com/DeitelFan](https://www.facebook.com/DeitelFan)
- Twitter®—[@deitel](https://twitter.com/deitel)
- Google+™—[google.com/+DeitelFan](https://plus.google.com/+DeitelFan)
- YouTube™—[google.com/+DeitelFan](https://www.youtube.com/+DeitelFan)
- LinkedIn®—[linkedin.com/company/deitel-&-associates](https://www.linkedin.com/company/deitel-&-associates)

and register for the free *Deitel® Buzz Online* e-mail newsletter at:

www.deitel.com/newsletter/subscribe.html

To communicate with the authors, send e-mail to:

deitel@deitel.com

For information on *Dive-Into® Series* on-site seminars offered by Deitel & Associates, Inc. worldwide, write to us at deitel@deitel.com or visit:

www.deitel.com/training/

For continuing updates on Pearson/Deitel publications visit:

www.deitel.com
www.pearsonglobaleditions.com/Deitel

Visit the Deitel Resource Centers that will help you master programming languages, software development, Android and iOS app development, and Internet- and web-related topics:

www.deitel.com/ResourceCenters.html

Android™

HOW TO PROGRAM

SECOND EDITION
Global Edition

Paul Deitel • Harvey Deitel • Abbey Deitel

Deitel & Associates, Inc.

Global Edition contributions by Muthuraj M.



PEARSON

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editorial Director, ECS: *Marcia Horton*
Head of Learning Asset Acquisition, Global Edition:
Laura Dent
Executive Editor: *Tracy Johnson (Dunkelberger)*
Director of Marketing: *Christy Lesko*
Marketing Manager: *Yez Alayan*
Marketing Assistant: *Jon Bryant*
Director of Program Management: *Erin Gregg*
Program Management-Team Lead: *Scott Disanno*
Program Manager: *Carole Snyder*
Project Management-Team Lead: *Laura Burgess*
Project Manager: *Robert Engelhardt*
Publishing Administrator and Business Analyst,
Global Edition: *Shokhi Shah Khandelwal*
Acquisitions Editor, Global Edition: *Karthik Subramanian*

Assistant Project Editor, Global Edition: *Sinjita Basu*
Media Producer, Global Edition: *M. Vikram Kumar*
Senior Manufacturing Controller, Production, Global
Edition: *Trudy Kimber*
Procurement Specialist: *Linda Sager*
Permissions Supervisor: *Michael Joyce*
Permissions Administrator: *Jenell Forschler*
Director, Image Asset Services: *Annie Atherton*
Manager, Visual Research: *Karen Sanatar*
Media Project Manager: *Renata Butera*
Cover Designer: *Shree Inbakumar*
Cover Photo: *Kirill_M/ Shutterstock*
Cover Printer: *Courier Westford*

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

© Pearson Education Limited 2015

The rights of Paul Deitel, Harvey Deitel, and Abbey Deitel to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Android: How to Program, 2nd edition, ISBN 978-0-13-376403-1, by Paul Deitel, Harvey Deitel, and Abbey Deitel, published by Pearson Education © 2015.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6-10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

ISBN 10: 0-273-79339-X

ISBN 13: 978-0-273-79339-7

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1
14 13 12 11 10

Typeset in Adobe Garamond by GEX Publishing Services.

Printed and bound by Courier Westford in the United States of America.

*In Memory of Amar G. Bose, MIT Professor and
Founder and Chairman of the Bose Corporation:*

*It was a privilege being your student—and members
of the next generation of Deitels, who heard our dad
say how your classes inspired him to do his best work.
You taught us that if we go after the really hard prob-
lems, then great things can happen.*

*Harvey Deitel
Paul and Abbey Deitel*

Trademarks

DEITEL, the double-thumbs-up bug and DIVE-INTO are registered trademarks of Deitel & Associates, Inc.

Java is a registered trademark of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Google, Android, Google Play, Google Maps, Google Wallet, Nexus, YouTube, AdSense and AdMob are trademarks of Google, Inc.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Throughout this book, trademarks are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names in an editorial fashion only and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Preface	19
Before You Begin	31
I Introduction to Android	39
1.1 Introduction	40
1.2 Android—The World’s Leading Mobile Operating System	41
1.3 Android Features	41
1.4 Android Operating System	45
1.4.1 Android 2.2 (Froyo)	45
1.4.2 Android 2.3 (Gingerbread)	46
1.4.3 Android 3.0 through 3.2 (Honeycomb)	46
1.4.4 Android 4.0 through 4.0.4 (Ice Cream Sandwich)	46
1.4.5 Android 4.1–4.3 (Jelly Bean)	47
1.4.6 Android 4.4 (KitKat)	48
1.5 Downloading Apps from Google Play	49
1.6 Packages	50
1.7 Android Software Development Kit (SDK)	51
1.8 Object-Oriented Programming: A Quick Refresher	54
1.8.1 The Automobile as an Object	55
1.8.2 Methods and Classes	55
1.8.3 Instantiation	55
1.8.4 Reuse	55
1.8.5 Messages and Method Calls	55
1.8.6 Attributes and Instance Variables	56
1.8.7 Encapsulation	56
1.8.8 Inheritance	56
1.8.9 Object-Oriented Analysis and Design (OOAD)	56
1.9 Test-Driving the Doodlz App in an Android Virtual Device (AVD)	57
1.9.1 Running the Doodlz App in the Nexus 4 Smartphone AVD	57
1.9.2 Running the Doodlz App in a Tablet AVD	66
1.9.3 Running the Doodlz App on an Android Device	68
1.10 Building Great Android Apps	68
1.11 Android Development Resources	70
1.12 Wrap-Up	73

2 Welcome App 76

Dive-Into® the Android Developer Tools: Introducing Visual GUI Design, Layouts, Accessibility and Internationalization

2.1	Introduction	77
2.2	Technologies Overview	78
2.2.1	Android Developer Tools IDE	78
2.2.2	TextViews and ImageViews	78
2.2.3	App Resources	78
2.2.4	Accessibility	78
2.2.5	Internationalization	78
2.3	Creating an App	79
2.3.1	Launching the Android Developer Tools IDE	79
2.3.2	Creating a New Project	79
2.3.3	New Android Application Dialog	80
2.3.4	Configure Project Step	81
2.3.5	Configure Launcher Icon Step	81
2.3.6	Create Activity Step	83
2.3.7	Blank Activity Step	84
2.4	Android Developer Tools Window	85
2.4.1	Package Explorer Window	86
2.4.2	Editor Windows	86
2.4.3	Outline Window	86
2.4.4	App Resource Files	86
2.4.5	Graphical Layout Editor	87
2.4.6	The Default GUI	87
2.5	Building the App's GUI with the Graphical Layout Editor	89
2.5.1	Adding Images to the Project	89
2.5.2	Changing the Id Property of the RelativeLayout and the TextView	90
2.5.3	Configuring the TextView	91
2.5.4	Adding ImageViews to Display the Images	95
2.6	Running the Welcome App	97
2.7	Making Your App Accessible	98
2.8	Internationalizing Your App	100
2.9	Wrap-Up	104

3 Tip Calculator App 107

Introducing GridLayout, LinearLayout, EditText, SeekBar, Event Handling, NumberFormat and Defining App Functionality with Java

3.1	Introduction	108
3.2	Test-Driving the Tip Calculator App	109
3.3	Technologies Overview	110
3.3.1	Class Activity	110
3.3.2	Activity Lifecycle Methods	110
3.3.3	Arranging Views with LinearLayout and GridLayout	111

3.3.4	Creating and Customizing the GUI with the Graphical Layout Editor and the Outline and Properties Windows	111
3.3.5	Formatting Numbers as Locale-Specific Currency and Percentage Strings	112
3.3.6	Implementing Interface <code>TextWatcher</code> for Handling <code>EditText</code> Text Changes	112
3.3.7	Implementing Interface <code>OnSeekBarChangeListener</code> for Handling <code>SeekBar</code> Thumb Position Changes	112
3.3.8	<code>AndroidManifest.xml</code>	113
3.4	Building the App's GUI	113
3.4.1	<code>GridLayout</code> Introduction	113
3.4.2	Creating the <code>TipCalculator</code> Project	115
3.4.3	Changing to a <code>GridLayout</code>	115
3.4.4	Adding the <code>TextViews</code> , <code>EditText</code> , <code>SeekBar</code> and <code>LinearLayouts</code>	116
3.4.5	Customizing the Views to Complete the Design	118
3.5	Adding Functionality to the App	122
3.6	<code>AndroidManifest.xml</code>	130
3.7	<code>Wrap-Up</code>	131

4 Twitter® Searches App 135

SharedPreferences, Collections, ImageButton, ListView, ListActivity, ArrayAdapter, Implicit Intents and AlertDialogs

4.1	Introduction	136
4.2	Test-Driving the App	137
4.2.1	Importing the App and Running It	137
4.2.2	Adding a Favorite Search	138
4.2.3	Viewing Twitter Search Results	139
4.2.4	Editing a Search	140
4.2.5	Sharing a Search	142
4.2.6	Deleting a Search	142
4.2.7	Scrolling Through Saved Searches	143
4.3	Technologies Overview	143
4.3.1	<code>ListView</code>	143
4.3.2	<code>ListActivity</code>	144
4.3.3	Customizing a <code>ListActivity</code> 's Layout	144
4.3.4	<code>ImageButton</code>	144
4.3.5	<code>SharedPreferences</code>	144
4.3.6	Intents for Launching Other Activities	145
4.3.7	<code>AlertDialog</code>	145
4.3.8	<code>AndroidManifest.xml</code>	146
4.4	Building the App's GUI	146
4.4.1	Creating the Project	146
4.4.2	<code>activity_main.xml</code> Overview	147
4.4.3	Adding the <code>GridLayout</code> and Components	148

4.4.4	Graphical Layout Editor Toolbar	153
4.4.5	ListView Item's Layout: list_item.xml	154
4.5	Building the MainActivity Class	155
4.5.1	package and import Statements	155
4.5.2	Extending ListActivity	157
4.5.3	Fields of Class MainActivity	157
4.5.4	Overriding Activity Method onCreate	158
4.5.5	Anonymous Inner Class That Implements the saveButton's onClickListener to Save a New or Updated Search	160
4.5.6	addTaggedSearch Method	162
4.5.7	Anonymous Inner Class That Implements the ListView's onItemClickListener to Display Search Results	163
4.5.8	Anonymous Inner Class That Implements the ListView's onItemLongClickListener to Share, Edit or Delete a Search	165
4.5.9	shareSearch Method	167
4.5.10	deleteSearch Method	168
4.6	AndroidManifest.xml	170
4.7	Wrap-Up	170

5 Flag Quiz App

174

Fragments, Menus, Preferences, AssetManager, Tweened Animations, Handler, Toasts, Explicit Intents, Layouts for Multiple Device Orientations

5.1	Introduction	175
5.2	Test-Driving the Flag Quiz App	177
5.2.1	Importing the App and Running It	177
5.2.2	Configuring the Quiz	177
5.2.3	Taking the Quiz	179
5.3	Technologies Overview	181
5.3.1	Menus	181
5.3.2	Fragments	181
5.3.3	Fragment Lifecycle Methods	182
5.3.4	Managing Fragments	182
5.3.5	Preferences	182
5.3.6	assets Folder	182
5.3.7	Resource Folders	183
5.3.8	Supporting Different Screen Sizes and Resolutions	183
5.3.9	Determining the Screen Size	184
5.3.10	Toasts for Displaying Messages	184
5.3.11	Using a Handler to Execute a Runnable in the Future	184
5.3.12	Applying an Animation to a View	184
5.3.13	Logging Exception Messages	185
5.3.14	Using an Explicit Intent to Launch Another Activity in the Same App	185
5.3.15	Java Data Structures	185
5.4	Building the GUI and Resource Files	185

5.4.1	Creating the Project	185
5.4.2	strings.xml and Formatted String Resources	186
5.4.3	arrays.xml	187
5.4.4	colors.xml	188
5.4.5	dimens.xml	188
5.4.6	activity_settings.xml Layout	189
5.4.7	activity_main.xml Layout for Phone and Tablet Portrait Orientation	189
5.4.8	fragment_quiz.xml Layout	189
5.4.9	activity_main.xml Layout for Tablet Landscape Orientation	192
5.4.10	preferences.xml for Specifying the App's Settings	193
5.4.11	Creating the Flag Shake Animation	194
5.5	MainActivity Class	196
5.5.1	package Statement, import Statements and Fields	196
5.5.2	Overridden Activity Method onCreate	197
5.5.3	Overridden Activity Method onStart	199
5.5.4	Overridden Activity Method onCreateOptionsMenu	199
5.5.5	Overridden Activity Method onOptionsItemSelected	200
5.5.6	Anonymous Inner Class That Implements OnSharedPreferencesChangeListener	201
5.6	QuizFragment Class	202
5.6.1	package Statement and import Statements	202
5.6.2	Fields	203
5.6.3	Overridden Fragment Method onCreateView	204
5.6.4	Method updateGuessRows	206
5.6.5	Method updateRegions	207
5.6.6	Method resetQuiz	207
5.6.7	Method loadNextFlag	209
5.6.8	Method getCountryName	211
5.6.9	Anonymous Inner Class That Implements OnClickListener	211
5.6.10	Method disableButtons	214
5.7	SettingsFragment Class	214
5.8	SettingsActivity Class	215
5.9	AndroidManifest.xml	215
5.10	Wrap-Up	216

6 Cannon Game App 220

Listening for Touches, Manual Frame-By-Frame Animation, Graphics, Sound, Threading, SurfaceView and SurfaceHolder

6.1	Introduction	221
6.2	Test-Driving the Cannon Game App	223
6.3	Technologies Overview	223
6.3.1	Attaching a Custom View to a Layout	223
6.3.2	Using the Resource Folder raw	223
6.3.3	Activity and Fragment Lifecycle Methods	223

6.3.4	Overriding View Method onTouchEvent	224
6.3.5	Adding Sound with SoundPool and AudioManager	224
6.3.6	Frame-by-Frame Animation with Threads, SurfaceView and SurfaceHolder	224
6.3.7	Simple Collision Detection	225
6.3.8	Drawing Graphics Using Paint and Canvas	225
6.4	Building the App's GUI and Resource Files	225
6.4.1	Creating the Project	225
6.4.2	strings.xml	226
6.4.3	fragment_game.xml	226
6.4.4	activity_main.xml	227
6.4.5	Adding the Sounds to the App	227
6.5	Class Line Maintains a Line's Endpoints	227
6.6	MainActivity Subclass of Activity	228
6.7	CannonGameFragment Subclass of Fragment	228
6.8	CannonView Subclass of View	230
6.8.1	package and import Statements	230
6.8.2	Instance Variables and Constants	231
6.8.3	Constructor	232
6.8.4	Overriding View Method onSizeChanged	234
6.8.5	Method newGame	235
6.8.6	Method updatePositions	236
6.8.7	Method fireCannonball	239
6.8.8	Method alignCannon	240
6.8.9	Method drawGameElements	241
6.8.10	Method showGameOverDialog	243
6.8.11	Methods stopGame and releaseResources	244
6.8.12	Implementing the SurfaceHolder.Callback Methods	245
6.8.13	Overriding View Method onTouchEvent	246
6.8.14	CannonThread: Using a Thread to Create a Game Loop	247
6.9	Wrap-Up	248

7 Doodlz App 253

Two-Dimensional Graphics, Canvas, Bitmap, Accelerometer, SensorManager, Multitouch Events, MediaStore, Printing, Immersive Mode

7.1	Introduction	254
7.2	Technologies Overview	256
7.2.1	Using SensorManager to Listen for Accelerometer Events	256
7.2.2	Custom DialogFragments	256
7.2.3	Drawing with Canvas and Bitmap	257
7.2.4	Processing Multiple Touch Events and Storing Lines in Paths	257
7.2.5	Android 4.4 Immersive Mode	257
7.2.6	GestureDetector and SimpleOnGestureListener	257
7.2.7	Saving the Drawing to the Device's Gallery	257

7.2.8	Android 4.4 Printing and the Android Support Library's PrintHelper Class	258
7.3	Building the App's GUI and Resource Files	258
7.3.1	Creating the Project	258
7.3.2	strings.xml	258
7.3.3	dimens.xml	259
7.3.4	Menu for the DoodleFragment	260
7.3.5	activity_main.xml Layout for MainActivity	261
7.3.6	fragment_doodle.xml Layout for DoodleFragment	261
7.3.7	fragment_color.xml Layout for ColorDialogFragment	262
7.3.8	fragment_line_width.xml Layout for LineWidthDialogFragment	264
7.3.9	Adding Class EraseImageDialogFragment	265
7.4	MainActivity Class	266
7.5	DoodleFragment Class	267
7.6	DoodleView Class	274
7.7	ColorDialogFragment Class	286
7.8	LineWidthDialogFragment Class	289
7.9	EraseImageDialogFragment Class	293
7.10	Wrap-Up	294

8 Address Book App **298**

ListFragment, FragmentTransactions and the Fragment Back Stack, Threading and AsyncTasks, CursorAdapter, SQLite and GUI Styles

8.1	Introduction	299
8.2	Test-Driving the Address Book App	302
8.3	Technologies Overview	302
8.3.1	Displaying Fragments with FragmentTransactions	303
8.3.2	Communicating Data Between a Fragment and a Host Activity	303
8.3.3	Method onSaveInstanceState	303
8.3.4	Defining Styles and Applying Them to GUI Components	303
8.3.5	Specifying a Background for a TextView	303
8.3.6	Extending Class ListFragment to Create a Fragment That Contains a ListView	304
8.3.7	Manipulating a SQLite Database	304
8.3.8	Performing Database Operations Outside the GUI Thread with AsyncTasks	304
8.4	Building the GUI and Resource Files	304
8.4.1	Creating the Project	304
8.4.2	Creating the App's Classes	305
8.4.3	strings.xml	305
8.4.4	styles.xml	306
8.4.5	textview_border.xml	307
8.4.6	MainActivity's Layout: activity_main.xml	308
8.4.7	DetailsFragment's Layout: fragment_details.xml	308
8.4.8	AddEditFragment's Layout: fragment_add_edit.xml	310
8.4.9	Defining the Fragments' Menus	311

14 Contents

8.5	MainActivity Class	312
8.6	ContactListFragment Class	318
8.7	AddEditFragment Class	325
8.8	DetailsFragment Class	331
8.9	DatabaseConnector Utility Class	339
8.10	Wrap-Up	344

9 Google Play and App Business Issues **348**

9.1	Introduction	349
9.2	Preparing Your Apps for Publication	349
9.2.1	Testing Your App	350
9.2.2	End User License Agreement	350
9.2.3	Icons and Labels	350
9.2.4	Versioning Your App	351
9.2.5	Licensing to Control Access to Paid Apps	351
9.2.6	Obfuscating Your Code	351
9.2.7	Getting a Private Key for Digitally Signing Your App	352
9.2.8	Screenshots	352
9.2.9	Promotional App Video	353
9.3	Pricing Your App: Free or Fee	354
9.3.1	Paid Apps	355
9.3.2	Free Apps	355
9.4	Monetizing Apps with In-App Advertising	356
9.5	Monetizing Apps: Using In-App Billing to Sell Virtual Goods	357
9.6	Registering at Google Play	358
9.7	Setting Up a Google Wallet Merchant Account	359
9.8	Uploading Your Apps to Google Play	360
9.9	Launching the Play Store from Within Your App	361
9.10	Managing Your Apps in Google Play	362
9.11	Other Android App Marketplaces	362
9.12	Other Popular Mobile App Platforms	362
9.13	Marketing Your Apps	363
9.14	Wrap-Up	367

A Introduction to Java Applications **370**

A.1	Introduction	371
A.2	Your First Program in Java: Printing a Line of Text	371
A.3	Modifying Your First Java Program	375
A.4	Displaying Text with <code>printf</code>	377
A.5	Another Application: Adding Integers	377
A.6	Memory Concepts	381
A.7	Arithmetic	382
A.8	Decision Making: Equality and Relational Operators	385
A.9	Wrap-Up	389

B Introduction to Classes, Objects, Methods and Strings **394**

B.1	Introduction	395
B.2	Declaring a Class with a Method and Instantiating an Object of a Class	395
B.3	Declaring a Method with a Parameter	398
B.4	Instance Variables, <i>set</i> Methods and <i>get</i> Methods	401
B.5	Primitive Types vs. Reference Types	405
B.6	Initializing Objects with Constructors	406
B.7	Floating-Point Numbers and Type <code>double</code>	408
B.8	Wrap-Up	412

C Control Statements **416**

C.1	Introduction	417
C.2	Algorithms	417
C.3	Pseudocode	418
C.4	Control Structures	418
C.5	<code>if</code> Single-Selection Statement	419
C.6	<code>if...else</code> Double-Selection Statement	419
C.7	<code>while</code> Repetition Statement	422
C.8	Case Study: Counter-Controlled Repetition	422
C.9	Case Study: Sentinel-Controlled Repetition	426
C.10	Case Study: Nested Control Statements	431
C.11	Compound Assignment Operators	434
C.12	Increment and Decrement Operators	434
C.13	Primitive Types	436
C.14	Essentials of Counter-Controlled Repetition	437
C.15	<code>for</code> Repetition Statement	438
C.16	Examples Using the <code>for</code> Statement	440
C.17	<code>do...while</code> Repetition Statement	442
C.18	<code>switch</code> Multiple-Selection Statement	443
C.19	<code>break</code> and <code>continue</code> Statements	450
C.20	Logical Operators	450
C.21	Wrap-Up	453

D Methods: A Deeper Look **461**

D.1	Introduction	462
D.2	Program Modules in Java	462
D.3	<code>static</code> Methods, <code>static</code> Fields and Class <code>Math</code>	463
D.4	Declaring Methods with Multiple Parameters	465
D.5	Notes on Declaring and Using Methods	468
D.6	Method-Call Stack and Activation Records	469
D.7	Argument Promotion and Casting	469
D.8	Java API Packages	470

D.9	Introduction to Random-Number Generation	471
D.9.1	Scaling and Shifting of Random Numbers	472
D.9.2	Random-Number Repeatability for Testing and Debugging	473
D.10	Case Study: A Game of Chance; Introducing Enumerations	474
D.11	Scope of Declarations	478
D.12	Method Overloading	480
D.13	Wrap-Up	483

E Arrays and ArrayLists **490**

E.1	Introduction	491
E.2	Arrays	491
E.3	Declaring and Creating Arrays	492
E.4	Examples Using Arrays	493
E.5	Case Study: Card Shuffling and Dealing Simulation	502
E.6	Enhanced for Statement	506
E.7	Passing Arrays to Methods	507
E.8	Case Study: Class GradeBook Using an Array to Store Grades	511
E.9	Multidimensional Arrays	516
E.10	Case Study: Class GradeBook Using a Two-Dimensional Array	520
E.11	Class Arrays	526
E.12	Introduction to Collections and Class ArrayList	528
E.13	Wrap-Up	531

F Classes and Objects: A Deeper Look **536**

F.1	Introduction	537
F.2	Time Class Case Study	537
F.3	Controlling Access to Members	541
F.4	Referring to the Current Object's Members with the <code>this</code> Reference	542
F.5	Time Class Case Study: Overloaded Constructors	544
F.6	Default and No-Argument Constructors	550
F.7	Composition	551
F.8	Enumerations	554
F.9	Garbage Collection	556
F.10	<code>static</code> Class Members	557
F.11	<code>final</code> Instance Variables	561
F.12	Packages	561
F.13	Package Access	562
F.14	Wrap-Up	562

G Object-Oriented Programming: Inheritance and Polymorphism **565**

G.1	Introduction to Inheritance	566
G.2	Superclasses and Subclasses	567
G.3	<code>protected</code> Members	568

G.4	Relationship between Superclasses and Subclasses	569
G.4.1	Creating and Using a <code>CommissionEmployee</code> Class	569
G.4.2	Creating and Using a <code>BasePlusCommissionEmployee</code> Class	574
G.4.3	Creating a <code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy	579
G.4.4	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>protected</code> Instance Variables	582
G.4.5	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>private</code> Instance Variables	585
G.5	Class Object	590
G.6	Introduction to Polymorphism	591
G.7	Polymorphism: An Example	592
G.8	Demonstrating Polymorphic Behavior	593
G.9	Abstract Classes and Methods	596
G.10	Case Study: Payroll System Using Polymorphism	597
G.10.1	Abstract Superclass <code>Employee</code>	598
G.10.2	Concrete Subclass <code>SalariedEmployee</code>	601
G.10.3	Concrete Subclass <code>HourlyEmployee</code>	603
G.10.4	Concrete Subclass <code>CommissionEmployee</code>	604
G.10.5	Indirect Concrete Subclass <code>BasePlusCommissionEmployee</code>	606
G.10.6	Polymorphic Processing, <code>Operator instanceof</code> and Downcasting	607
G.10.7	Summary of the Allowed Assignments Between Superclass and Subclass Variables	612
G.11	<code>final</code> Methods and Classes	613
G.12	Case Study: Creating and Using Interfaces	614
G.12.1	Developing a Payable Hierarchy	615
G.12.2	Interface <code>Payable</code>	616
G.12.3	Class <code>Invoice</code>	617
G.12.4	Modifying Class <code>Employee</code> to Implement Interface <code>Payable</code>	619
G.12.5	Modifying Class <code>SalariedEmployee</code> for Use in the Payable Hierarchy	621
G.12.6	Using Interface <code>Payable</code> to Process Invoices and Employees Polymorphically	623
G.13	Common Interfaces of the Java API	624
G.14	Wrap-Up	625

H Exception Handling: A Deeper Look **629**

H.1	Introduction	630
H.2	Example: Divide by Zero without Exception Handling	630
H.3	Example: Handling <code>ArithmeticExceptions</code> and <code>InputMismatchExceptions</code>	632
H.4	When to Use Exception Handling	637
H.5	Java Exception Hierarchy	637
H.6	<code>finally</code> Block	640
H.7	Stack Unwinding and Obtaining Information from an Exception Object	644
H.8	Wrap-Up	647

I	GUI Components and Event Handling	650
I.1	Introduction	651
I.2	Nimbus Look-and-Feel	651
I.3	Text Fields and an Introduction to Event Handling with Nested Classes	652
I.4	Common GUI Event Types and Listener Interfaces	658
I.5	How Event Handling Works	659
I.6	JButton	660
I.7	JComboBox; Using an Anonymous Inner Class for Event Handling	665
I.8	Adapter Classes	668
I.9	Wrap-Up	669
J	Other Java Topics	670
J.1	Introduction	671
J.2	Collections Overview	671
J.3	Type-Wrapper Classes for Primitive Types	672
J.4	Interface Collection and Class Collections	672
J.5	Lists	673
	J.5.1 ArrayList and Iterator	673
	J.5.2 LinkedList	675
	J.5.3 Views into Collections and Arrays Method asList	678
J.6	Collections Methods	680
	J.6.1 Method sort	680
	J.6.2 Method shuffle	682
J.7	Interface Queue	683
J.8	Sets	683
J.9	Maps	684
J.10	Introduction to Files and Streams	687
J.11	Class File	688
J.12	Introduction to Object Serialization	689
J.13	Introduction to Multithreading	690
J.14	Creating and Executing Threads with the Executor Framework	691
J.15	Overview of Thread Synchronization	695
J.16	Concurrent Collections Overview	696
J.17	Multithreading with GUI	696
J.18	Wrap-Up	703
K	Operator Precedence Chart	706
L	Primitive Types	708
	Index	709

Preface

Build a better mousetrap, and the world will beat a path to your door.

—Ralph Waldo Emerson

Science and technology and the various forms of art,
all unite humanity in a single and interconnected system.

—Zhores Aleksandrovich Medvede

Welcome to the dynamic world of Android smartphone and tablet app development with the Android Software Development Kit (SDK), the Java™ programming language, the Android Development Tools IDE, and the new and rapidly evolving Android Studio. We present leading-edge mobile computing technologies for students, instructors and professional software developers.

Android How to Program, 2/e

With this unique book—the second edition of the world’s first Android computer science textbook—you can learn Android even if you don’t know Java and even if you’re a programming novice. This book includes a complete, 300-page introduction to the Java core programming concepts that you’ll need when developing Android apps. The Java content is appropriate for programming novices.

Android How to Program, 2/e was formed by merging

- our professional book *Android for Programmers: An App-Driven Approach, 2/e, Volume 1*
- additional online chapters selected from *Android for Programmers: An App-Driven Approach, 2/e, Volume 2*
- condensed, introductory core content on object-oriented Java programming from our college textbook *Java How to Program, 9/e*
- hundreds of Android short-answer questions and app-development exercises we created for this book—most are in the book and many of the short-answer questions are in the test-item file for instructors.

We scoured the Android material, especially the fully coded Android apps, and enumerated the Java features that you’ll need to build these and similar apps. Then we extracted the corresponding Java content from *Java How to Program, 9/e*. That’s a 1500-page book, so it was challenging to whittle down that much content and keep it friendly, even for programming novices.

When you study the Android content, you’ll be thinking like a developer from the start. You’re going to study and build lots of real stuff and you’ll face the kinds of challenges professional developers must deal with. We’ll point you to the online documenta-

tion and forums where you can find additional information and get answers to your questions. We'll also encourage you to read, modify and enhance open-source code as part of your learning process.

Intended Audiences

There are several audiences for this book. Most commonly, it will be used in upper-level elective college courses and industry professional courses for people familiar with object-oriented programming but who may or may not know Java and want to learn Android app development.

Uniquely, the book can also be used in introductory courses like CS1, intended for programming novices. We recommend that schools typically offering many sections of CS1 in Java consider designating one or two sections for ambitious students who have at least some prior programming experience and who want to work hard to learn a good amount of Java and Android in an aggressively paced one-semester course. The schools may want to list the courses with “honors” or “accelerated” designations. The book works especially well in two-semester introductory programming sequences where the introduction to Java is covered first.

App-Development Courses

In 2007, Stanford offered a new course called Creating Engaging Facebook Apps. Students worked in teams developing apps, some of which landed in Facebook's top 10, earning some of the student developers millions of dollars.¹ This course gained wide recognition for encouraging student creativity and teamwork. Scores of colleges now offer app-development courses across many social networking and mobile platforms such as Android and iOS. We encourage you to read the online mobile app development syllabi and check out the YouTube™ videos created by instructors and students for many of these courses.

Android Ecosystem: Competition, Innovation, Explosive Growth and Opportunities

Sales of Android devices and app downloads have been growing exponentially. The first-generation Android phones were released in October 2008. A study by Strategy Analytics showed that by October 2013, Android had 81.3% of the global smartphone market share, compared to 13.4% for Apple, 4.1% for Microsoft and 1% for Blackberry.² According to an IDC report, by the end of the first quarter of 2013 Android had 56.5% of the global tablet market share, compared to 39.6% for Apple's iPad and 3.7% for Microsoft Windows tablets.³

There are now over one billion Android smartphones and tablets in use,⁴ and more than 1.5 million Android devices are being activated daily.⁵ According to IDC, Samsung

-
1. <http://www.businessinsider.com/these-stanford-students-made-millions-taking-a-class-on-facebook-2011-5>.
 2. <http://blogs.strategyanalytics.com/WSS/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipments-in-Q3-2013.aspx>.
 3. <http://www.idc.com/getdoc.jsp?containerId=prUS24093213>.
 4. <http://www.android.com/kitkat>.
 5. <http://www.technobuffalo.com/2013/04/16/google-daily-android-activations-1-5-million>.

is the leading Android manufacturer, accounting for nearly 40% of Android device shipments in the third quarter of 2013.

Billions of apps have been downloaded from Google Play™—Google’s marketplace for Android Apps. The opportunities for Android app developers are enormous.

Fierce competition among popular mobile platforms and carriers is leading to rapid innovation and falling prices. Competition among the dozens of Android device manufacturers is driving hardware and software innovation within the Android community.

App-Driven Approach

At the heart of the book is our *app-driven approach*—we present concepts in the context of *seven complete working Android apps* in the print book and more online. We begin each of the app chapters with an *introduction* to the app, an app *test-drive* showing one or more *sample executions*, and a *technologies overview*. We build the app’s GUI and resource files. Then we proceed with a detailed *code walkthrough* of the app’s source code in which we discuss the programming concepts and demonstrate the functionality of the Android APIs used in the app. All the source code is available at the book’s Companion Website www.pearsonglobaleditions.com/Deitel. We recommend that you have the source code open in the IDE as you read the book. Figure 1 lists the book’s apps and the key technologies we used to build each.

App	Technologies
Chapter 2, Welcome App	The Android Developer Tools (the Eclipse IDE and the ADT Plugin), visual GUI design, layouts, TextViews, ImageViews, accessibility and internationalization.
Chapter 3, Tip Calculator App	GridLayout, LinearLayout, EditText, SeekBar, event handling, NumberFormat and defining app functionality with Java.
Chapter 4, Twitter® Searches App	SharedPreferences, collections, ImageButton, ListView, ListActivity, ArrayAdapter, implicit intents and AlertDialogs.
Chapter 5, Flag Quiz App	Fragments, menus, preferences, AssetManager, tweened animations, Handler, Toasts, Explicit Intents, layouts for multiple device orientations.
Chapter 6, Cannon Game App	Listening for touches, frame-by-frame animation, graphics, sound, threading, SurfaceView and SurfaceHolder.
Chapter 7, Doodlz App	Two-dimensional graphics, Canvas, Bitmap, accelerometer, SensorManager, multitouch events, MediaStore, printing and Immersive Mode.
Chapter 8, Address Book App	AdapterViews and Adapters

Fig. 1 | *Android How to Program apps in the print book.*

Online Chapters and Book Updates

The Companion Website contains additional app-development chapters that introduce property animation, Google Play game services, video, speech synthesis and recognition, GPS, the Maps API, the compass, object serialization, Internet-enabled apps, audio recording and playback, Bluetooth®, HTML5 mobile apps and more. **Most of these chapters will be available for fall 2014 courses. For the status of the online chapters and for continuing book updates, visit**

www.pearsonglobal editions.com/Deitel

Join the Deitel communities on Facebook® (<http://www.deitel.com/deitelfan>), Twitter® (@deitel), LinkedIn® (<http://bit.ly/DeitelLinkedIn>) Google+™ (<http://google.com/+DeitelFan>), and YouTube™ (<http://youtube.com/user/DeitelTV>) and subscribe to the *Deitel® Buzz Online* newsletter (<http://www.deitel.com/newsletter/subscribe.html>).

Copyright Notice and Code License

All of the Android code and Android apps in the book are copyrighted by Deitel & Associates, Inc. The sample Android apps in the book are licensed under a Creative Commons Attribution 3.0 Unported License (<http://creativecommons.org/licenses/by/3.0>), with the exception that they may not be reused in any way in educational tutorials and textbooks, whether in print or digital format. Additionally, the authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs or to the documentation contained in this book. The authors and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs. You're welcome to use the apps in the book as shells for your own apps, building on their existing functionality. If you have any questions, contact us at deitel@deitel.com.

Getting up to Speed in Java and XML

The Android portion of this book assumes that you already know Java and object-oriented programming. If you're not familiar with these, the appendices provide a condensed, friendly introduction to Java and the object-oriented programming techniques you'll need to develop Android apps. If you're interested in learning Java in more depth, you may want to check out the comprehensive treatment in our textbook *Java How to Program, 10/e* www.pearsonglobal editions.com/Deitel.

Because of the improved Android development tools, we were able to eliminate almost all XML markup in this edition. There are still two small, easy-to-understand XML files you'll need to manipulate. If you're not familiar with XML, see these online tutorials:

- http://www.deitel.com/articles/xml_tutorials/20060401/XMLBasics/
- http://www.deitel.com/articles/xml_tutorials/20060401/XMLStructuringData/
- <http://www.ibm.com/developerworks/xml/newto/>
- http://www.w3schools.com/xml/xml_what_is.asp

Key Features of *Android How to Program, 2/e*

- ***Android SDK 4.3 and 4.4.*** We cover various new Android Software Development Kit (SDK) 4.3 and 4.4 features. [Note: The apps in this book are configured to run on Android devices with Android 4.3 and higher; however, most apps will work in 4.0 and higher by changing their minimum required SDK.]
- ***Fragments.*** Starting with Chapter 5, we use Fragments to create and manage portions of each app's GUI. You can combine several fragments to create user interfaces that take advantage of tablet screen sizes. You also can easily interchange fragments to make your GUIs more dynamic, as you'll do in Chapter 8.
- ***Support for multiple screen sizes and resolutions.*** Throughout the app chapters we demonstrate how to use Android's mechanisms for automatically choosing resources (layouts, images, etc.) based on a device's size and orientation.
- ***Eclipse-Based Android Development Tools (ADT) coverage in the print book.*** The free Android Development Tools (ADT) integrated development environment (IDE)—which includes Eclipse and the ADT plugin—combined with the free Java Development Kit (JDK) provide all the software you'll need to create, run and debug Android apps, export them for distribution (e.g., upload them to Google Play™) and more.
- ***Android Studio.*** This is the preferred IDE for the future of Android app development. Because this IDE is evolving quickly, we put our discussions of it online at:

www.pearsonglobaleditions.com/Deitel

- ***Immersive Mode.*** The status bar at the top of the screen and the menu buttons at the bottom can be hidden, allowing your apps to fill more of the screen. Users can access the status bar by swiping down from the top of the screen, and the system bar (with the back button, home button and recent apps button) by swiping up from the bottom.
- ***Printing Framework.*** Android 4.4 KitKat allows you to add printing functionality to your apps, such as locating available printers over Wi-Fi or the cloud, selecting the paper size and specifying which pages to print.
- ***Testing on Android Smartphones, Tablets and the Android Emulator.*** For the best app-development experience, you should test your apps on actual Android smartphones and tablets. You can still have a meaningful experience using the Android emulator (see the Before You Begin section), however it's processor-intensive and can be slow—particularly with games that have a lot of moving parts. In Chapter 1, we mention some Android features that are not supported on the emulator.
- ***Multimedia.*** The apps in the print book use a broad range of Android multimedia capabilities, including graphics, images, frame-by-frame animation and audio. The apps in the online chapters use property animation, video, speech synthesis and speech recognition.
- ***Android Best Practices.*** We adhere to accepted Android best practices, pointing them out in the detailed code walkthroughs. For more information, visit <http://developer.android.com/guide/practices/index.html>.
- ***Java Content in the Appendices Can Be Used With Java SE 6 or Higher.***

- *Java Exception Handling.* We integrate basic exception handling early in the Java content then present a richer treatment in Appendix H; we use exception handling throughout the Android chapters.
- *Classes **Arrays** and **ArrayList**; Collections.* Appendix E covers class **Arrays**—which contains methods for performing common array manipulations—and generic class **ArrayList**—which implements a dynamically resizable array-like data structure. Appendix J introduces Java’s generic collections that are used frequently in our Android treatment.
- *Java Multithreading.* Maintaining app responsiveness is a key to building robust Android apps and requires extensive use of Android multithreading. Appendix J introduces multithreading fundamentals so that you can understand our use of the Android **AsyncTask** class in Chapter 8.
- *GUI Presentation.* Appendix I introduces Java GUI development. Android provides its own GUI components, so this appendix presents a few Java GUI components and focuses on nested classes and anonymous inner classes, which are used extensively for event-handling in Android GUIs.

Working with Open-Source Apps

There are numerous free, open-source Android apps available online which are excellent resources for learning Android app development. We encourage you to download open-source apps and read their source code to understand how they work. Throughout the book you’ll find programming exercises that ask you to modify or enhance existing open-source apps. Our goal is to give you handles on interesting problems that may also inspire you to create new apps using the same technologies. **Caution: The terms of open source licenses vary considerably.** Some allow you to use the app’s source code freely for any purpose, while others stipulate that the code is available for personal use only—not for creating for-sale or publicly available apps. **Be sure to read the licensing agreements carefully.** If you wish to create a commercial app based on an open-source app, you should consider having an intellectual property attorney read the license; be aware that these attorneys charge significant fees.

Pedagogic Features

Syntax Shading. For readability, we syntax shade the code, similar to Eclipse’s and Android Studio’s use of syntax coloring. Our syntax-shading conventions are as follows:

```
comments appear in gray
constants and literal values appear in bold darker gray
keywords appear in bold black
all other code appears in non-bold black
```

Code Highlighting. We emphasize the key code segments in each program by enclosing them in light gray rectangles.

Using Fonts for Emphasis. We use various font conventions:

- The defining occurrences of key terms appear in **bold** for easy reference.
- On-screen IDE components appear in **bold Helvetica** (e.g., the **File** menu).

- Program source code appears in Lucida (e.g., `int x = 5;`).

In this book you'll create GUIs using a combination of visual programming (point and click, drag and drop) and writing code.

We use different fonts when we refer to GUI elements in program code versus GUI elements displayed in the IDE:

- When we refer to a GUI component that we create in a program, we place its class name and object name in a Lucida font—e.g., “Button saveContactButton.”
- When we refer to a GUI component that's part of the IDE, we place the component's text in a **bold Helvetica** font and use a plain text font for the component's type—e.g., “the **File** menu” or “the **Run** button.”

Using the > Character. We use the > character to indicate selecting a menu item from a menu. For example, we use the notation **File > New** to indicate that you should select the **New** menu item from the **File** menu.

Source Code. All of the book's source code is available for download from:

www.pearsonglobaleditions.com/Deitel

Chapter Objectives. Each chapter begins with a list of learning objectives.

Figures. Hundreds of tables, source code listings and screen shots are included.

Software Engineering. We stress program clarity and performance, and concentrate on building well-engineered, object-oriented software.

Self-Review Exercises and Answers. Extensive self-review exercises *and* answers are included for self study.

Exercises with a Current Flair. We've worked hard to create topical Android app-development exercises. You'll develop apps using a broad array of current technologies. All of the Android programming exercises require the implementation of complete apps. You'll be asked to enhance the existing chapter apps, develop similar apps, use your creativity to develop your own apps that use the chapter technologies and build new apps based on open-source apps available on the Internet (and again, **be sure to read and comply with the open-source code-license terms for each app**). The Android exercises also include short-answer fill-in and true/false questions.

In the Java exercises, you'll be asked to recall important terms and concepts; indicate what code segments do; indicate what's wrong with a portion of code; write Java statements, methods and classes; and write complete Java programs.

Index. We include an extensive index for reference. The page number of the defining occurrence of each key term in the book is highlighted in the index in **bold**.

Software Used in *Android How to Program, 2/e*

All the software you'll need for this book is available free for download from the Internet. See the Before You Begin section for the download links.

Documentation. All the Android and Java documentation you'll need to develop Android apps is available free at <http://developer.android.com> and <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. The documentation for Eclipse is available at www.eclipse.org/documentation. The documentation for Android Studio is available at <http://developer.android.com/sdk/installing/studio.html>.

Instructor Resources

The following supplements are available to **qualified college instructors only** through Pearson Education's Instructor Resource Center www.pearsonglobaleditions.com/Deitel:

- **PowerPoint® slides** containing all the code and figures in the text.
- **Test Item File** of short-answer questions.
- **Solutions Manual** with solutions to the end-of-chapter **short-answer exercises** for *both* the Java and Android content. For the Java content, solutions are provided for *most* of the programming exercises.

The suggested Android app-development project exercises are *not* typical homework problems. These tend to be *substantial* projects—many of which could require weeks of effort, possibly with students working in teams. **Selected solutions only** are provided for these project exercises—these will be available on the Pearson Instructor's Resource Center (IRC) for fall semester 2014 classes. Contact us at deitel@deitel.com if you have any questions.

Please do not write to us requesting access to the Pearson Instructor's Resource Center. Access is restricted to qualified college instructors teaching from the book. Instructors may obtain access *only* through their Pearson representatives. If you're not a registered faculty member, contact your Pearson representative.

Before You Begin

For information configuring your computer so that you can develop apps with Java and Android, see the Before You Begin section that follows this Preface.

Acknowledgments

Thanks to Barbara Deitel for long hours devoted to this project—she created all of our Java and Android Resource Centers, and patiently researched hundreds of technical details.

This book was a cooperative effort between the academic and professional divisions of Pearson. We appreciate the guidance, wisdom and energy of Tracy Johnson, Executive Editor, Computer Science. Tracy and her team handle all of our academic textbooks. Carole Snyder recruited the book's academic reviewers and managed the review process. Bob Engelhardt managed the book's publication. We selected the cover art and Marta Samsel designed the cover.

We also appreciate the efforts and 18-year mentorship of our friend and professional colleague Mark L. Taub, Editor-in-Chief of the Pearson Technology Group. Mark and his team handle all of our professional books and LiveLessons video products. Kim Boe-

digheimer recruited and managed the professional reviewers for the Android content. John Fuller manages the production of all of our Deitel Developer Series books.

We'd like to thank Michael Morgano, a former colleague of ours at Deitel & Associates, Inc., now an Android developer at Imerj™, who co-authored the first editions of this book and our book, *iPhone for Programmers: An App-Driven Approach*. Michael is an extraordinarily talented software developer.

Reviewers of the Content from Android How to Program and Android for Programmers: An App-Driven Approach Recent Editions

We wish to acknowledge the efforts of our first and second edition reviewers. They scrutinized the text and the code and provided countless suggestions for improving the presentation: Paul Beusterien (Principal, Mobile Developer Solutions), Eric J. Bowden, COO (Safe Driving Systems, LLC), Tony Cantrell (Georgia Northwestern Technical College), Ian G. Clifton (Independent Contractor and Android App Developer, Daniel Galpin (Android Advocate and author of *Intro to Android Application Development*), Jim Hathaway (Application Developer, Kellogg Company), Douglas Jones (Senior Software Engineer, Fullpower Technologies), Charles Lasky (Nagautuck Community College), Enrique Lopez-Manas (Lead Android Architect, Sixt, and Computer Science Teacher at the University of Alcalá in Madrid), Sebastian Nykopp (Chief Architect, Reaktor), Michael Pardo (Android Developer, Mobiata), Ronan “Zero” Schwarz (CIO, OpenIntents), Arijit Sen Gupta (Wright State University), Donald Smith (Columbia College), Jesus Ubaldo Quevedo-Torrero (University of Wisconsin, Parkside), Dawn Wick (Southwestern Community College) and Frank Xu (Gannon University).

Reviewers of the Content from Java How to Program Recent Editions

Lance Andersen (Oracle), Soundararajan Angusamy (Sun Microsystems), Joseph Bowbeer (Consultant), William E. Duncan (Louisiana State University), Diana Franklin (University of California, Santa Barbara), Edward F. Gehringer (North Carolina State University), Huiwei Guan (Northshore Community College), Ric Heishman (George Mason University), Dr. Heinz Kabutz (JavaSpecialists.eu), Patty Kraft (San Diego State University), Lawrence Premkumar (Sun Microsystems), Tim Margush (University of Akron), Sue McFarland Metzger (Villanova University), Shyamal Mitra (The University of Texas at Austin), Peter Pilgrim (Consultant), Manjeet Rege, Ph.D. (Rochester Institute of Technology), Manfred Riem (Java Champion, Consultant, Robert Half), Simon Ritter (Oracle), Susan Rodger (Duke University), Amr Sabry (Indiana University), José Antonio González Seco (Parliament of Andalusia), Sang Shin (Sun Microsystems), S. Sivakumar (Astra Infotech Private Limited), Raghavan “Rags” Srinivas (Intuit), Monica Sweat (Georgia Tech), Vinod Varma (Astra Infotech Private Limited) and Alexander Zuev (Sun Microsystems).

As you read the book, we'd sincerely appreciate your comments, criticisms and suggestions for improving the text. Please address all correspondence to:

`deitel@deitel.com`

We'll respond promptly. We really enjoyed writing this book—we hope you enjoy reading it!

Paul Deitel
Harvey Deitel
Abbey Deitel

About the Authors

Paul Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT, where he studied Information Technology. He holds the Java Certified Programmer and Java Certified Developer certifications, and is an Oracle Java Champion. Through Deitel & Associates, Inc., he has delivered hundreds of programming courses worldwide to clients, including Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Nortel Networks, Puma, iRobot, Invensys and many more. He and his co-author, Dr. Harvey M. Deitel, are the world's best-selling programming-language textbook/professional book/video authors.

Dr. Harvey Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 50 years of experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University. He has extensive college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., in 1991 with his son, Paul Deitel. The Deitels' publications have earned international recognition, with translations published in Simplified Chinese, Traditional Chinese, Korean, Japanese, German, Russian, Spanish, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to corporate, academic, government and military clients.

Abbey Deitel, President of Deitel & Associates, Inc., is a graduate of Carnegie Mellon University's Tepper School of Management where she received a B.S. in Industrial Management. Abbey has been managing the business operations of Deitel & Associates, Inc. for 16 years. She has contributed to numerous Deitel & Associates publications and, together with Paul and Harvey, is the co-author of *Android for Programmers: An App-Driven Approach*, 2/e, *iPhone for Programmers: An App-Driven Approach*, *Internet & World Wide Web How to Program*, 5/e, *Visual Basic 2012 How to Program*, 6/e and *Simply Visual Basic 2010*, 5/e.

Deitel® Dive-Into® Series Programming Languages Training

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate training organization, specializing in computer programming languages, object technology, mobile app development and Internet and web software technology. The company's training clients include many of the world's largest companies, government agencies, branches of the military, and academic institutions. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms, including Android app development, Objective-C and iOS app development, Java™, XML®, C++, C, Visual C#®, Visual Basic®, Visual C++®, Python®, object technology, Internet and web programming and a growing list of additional programming and software development courses.

Through its 37-year publishing partnership with Prentice Hall/Pearson, Deitel & Associates, Inc., publishes leading-edge programming college textbooks and professional books in print and a wide range of electronic formats and *LiveLessons* video courses. Deitel & Associates, Inc. and the authors can be reached at:

To learn more about Deitel's *Dive-Into® Series* Corporate Training curriculum, visit:

<http://www.deitel.com/training>

To request a proposal for worldwide on-site, instructor-led training at your organization, e-mail deitel@deitel.com.

Individuals wishing to purchase Deitel books and *LiveLessons* video training can do so through www.deitel.com. Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, please contact your Pearson representative.

Pearson wishes to thank and acknowledge the following people for their work on the Global Edition:

Contributor:

Muthuraj M., Android Developer

Reviewers:

SC Raghavendra SSE, Intuit, India

Manasa S., NMAM Institute of Technology, Nitte, India

Before You Begin

In this section, you'll set up your computer for use with this book. The Android development tools are frequently updated. Before reading this section, check the book's website

www.pearsonglobal editions.com/Deitel

to see if we've posted an updated version.

Font and Naming Conventions

We use fonts to distinguish between on-screen components (such as menu names and menu items) and Java code or commands. Our convention is to show on-screen components in a sans-serif bold **Helvetica** font (for example, **Project** menu) and to show file names, Java code and commands in a sans-serif **Lucida** font (for example, the keyword `public` or class `Activity`). When specifying commands to select in menus, we use the `>` notation to indicate a menu item to select. For example, **Window > Preferences** indicates that you should select the **Preferences** menu item from the **Window** menu.

Software and Hardware System Requirements

To develop Android apps you need a Windows®, Linux or Mac OS X system. To view the latest operating-system requirements visit:

<http://developer.android.com/sdk/index.html>

and scroll down to the **SYSTEM REQUIREMENTS** heading. We developed the apps in this book using the following software:

- Java SE 7 Software Development Kit
- Android SDK/ADT Bundle based on the Eclipse IDE
- Android SDK versions 4.3 and 4.4

You'll see how to obtain each of these in the next sections.

Installing the Java Development Kit (JDK)

Android requires the *Java Development Kit (JDK)* version 7 (JDK 7) or 6 (JDK 6). *We used JDK 7.* To download the JDK for Windows, OS X or Linux, go to

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

You need only the JDK. Choose the 32-bit or 64-bit version based on your computer hardware and operating system. Most recent computers have 64-bit hardware—check your system's specifications. If you have a 32-bit operating system, you must use the 32-bit JDK. Be sure to follow the installation instructions at

<http://docs.oracle.com/javase/7/docs/webnotes/install/index.html>

Android Integrated Development Environment (IDE) Options

Google now provides two Android IDE options:

- Android SDK/ADT bundle—a version of the *Eclipse IDE* that comes preconfigured with the latest Android Software Development Kit (SDK) and the latest Android Development Tools (ADT) plugin. At the time of this writing, these were Android SDK version 4.4 and ADT version 22.3.
- Android Studio—Google’s new Android IDE based on IntelliJ® IDEA and their preferred future IDE.

The Android SDK/ADT bundle has been widely used in Android app development for several years. Android Studio, introduced in May 2013, is an *early access version* and will be evolving rapidly. For this reason, we’ll stay with the widely used Android SDK/ADT bundle in the book, and as online supplements at

www.pearsonglobaleditions.com/Deitel

we’ll provide Android Studio versions of the Chapter 1 Test-Drive section and the Building the GUI section for each app, as appropriate.

Installing the Android SDK/ADT Bundle

To download the Android SDK/ADT bundle, go to

<http://developer.android.com/sdk/index.html>

and click the **Download the SDK ADT Bundle** button. When the download completes, extract the ZIP file’s contents to your system. The resulting folder has an `eclipse` subfolder containing the Eclipse IDE and an `sdk` subfolder containing the Android SDK. As with the JDK, you can choose a 32-bit or 64-bit version. The Android SDK/ADT bundle 32-bit version should be used with the 32-bit JDK, and the 64-bit version with the 64-bit JDK.

Installing Android Studio



The IDE instructions in the printed book use the Android SDK/ADT bundle. You can also optionally install and use Android Studio. To download Android Studio, go to

<http://developer.android.com/sdk/installing/studio.html>

and click the **Download Android Studio** button. When the download completes, run the installer and follow the on-screen instructions to complete the installation. [*Note:* For Android 4.4 development in Android Studio, Android now supports Java SE 7 language features, including the diamond operator, multi-catch, Strings in `switch` and `try-with-resources`.]

Set the Java Compiler Compliance Level and Show Line Numbers



Android does not fully support Java SE 7. To ensure that the book’s examples compile correctly, configure Eclipse to produce files that are compatible with Java SE 6 by performing the following steps:

1. Open Eclipse ( or ) , which is located in the `eclipse` subfolder of the Android SDK/ADT bundle’s installation folder.
2. When the **Workspace Launcher** window appears, click **OK**.

3. Select **Window > Preferences** to display the **Preferences** window. On Mac OS X, select **ADT > Preferences....**
4. Expand the **Java** node and select the **Compiler** node. Under **JDK Compliance**, set the **Compiler compliance level** to 1.6 (to indicate that Eclipse should produce compiled code that's compatible with Java SE 6).
5. Expand the **General > Editors** node and select **TextEditors**, then ensure that **Show line numbers** is selected and click **OK**.
6. Close Eclipse.

Android 4.3 SDK

This book's examples were written using the Android 4.3 and 4.4 SDKs. At the time of this writing, 4.4 was the version included with the Android SDK/ADT bundle and Android Studio. You should also install Android 4.3 (and any other versions you might want to support in your apps). To install other Android platform versions, perform the following steps (skipping Steps 1 and 2 if Eclipse is already open):

1. Open Eclipse. Depending on your platform, the icon will appear as  or .
2. When the **Workspace Launcher** window appears, click **OK**.
3. On Mac OS X, if you see a window indicating “**Could not find SDK folder '/Users/YourAccount/android-sdk-macosx/'**,” click **Open Preferences** then **Browse...** and select the sdk folder located where you extracted the Android SDK/ADT bundle.
4. Select **Window > Android SDK Manager** to display the **Android SDK Manager** (Fig. 1).

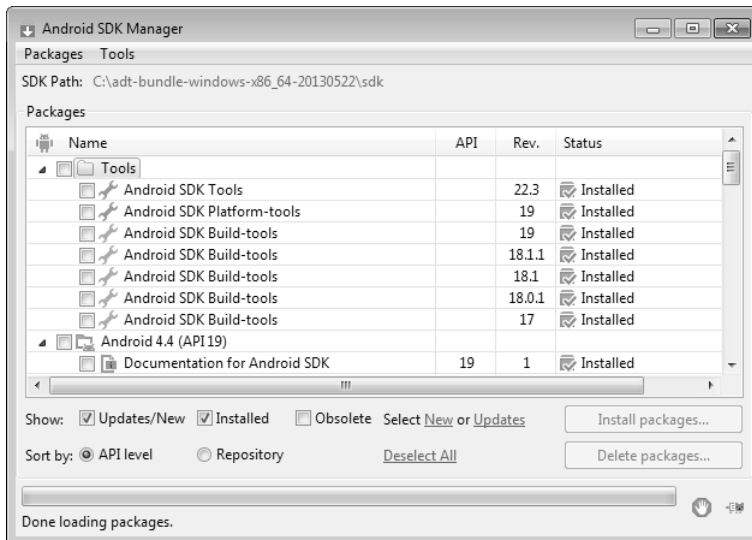


Fig. 1 | Android SDK Manager window.

5. The **Android SDK Manager**'s **Name** column shows all of the tools, platform versions and extras (such as APIs for interacting with Google services, like Maps) that you

can install. Uncheck the **Installed** checkbox. Then, if any of **Tools**, **Android 4.4 (API19)**, **Android 4.3 (API18)** and **Extras** appear in the **Packages** list, ensure that they're checked and click **Install # packages...** (# is the number of items to be installed) to display the **Choose Packages to Install** window. Most items in the **Extras** node are optional. For this book, you'll need the **Android Support Library** and **Google Play services**. The **Google USB Driver** is necessary for Windows users who wish to test apps on Android devices.]

6. In the **Choose Packages to Install** window, read the license agreements for each item. When you're done, click the **Accept License** radio button, then click the **Install** button. The status of the installation process will be displayed in the **Android SDK Manager** window.

Creating Android Virtual Devices (AVDs)

The **Android emulator**, included in the Android SDK, allows you to test apps on your computer rather than on an actual Android device. This is useful if you're learning Android and don't have access to Android devices, but can be *very* slow, so a real device is preferred if you have one. There are some hardware acceleration features that can improve emulator performance (developer.android.com/tools/devices/emulator.html#acceleration). Before running an app in the emulator, you must create an **Android Virtual Device (AVD)** which defines the characteristics of the device you want to test on, including the screen size in pixels, the pixel density, the physical size of the screen, size of the SD card for data storage and more. To test your apps for multiple Android devices, you can create AVDs that emulate each unique device. For this book, we use AVDs for Google's Android reference devices—the Nexus 4 phone, the Nexus 7 small tablet and Nexus 10 large tablet—which run unmodified versions of Android. To do so, perform the following steps:

1. Open Eclipse.
2. Select **Window > Android Virtual Device Manager** to display the **Android Virtual Device Manager** window, then select the **Device Definitions** tab (Fig. 2).

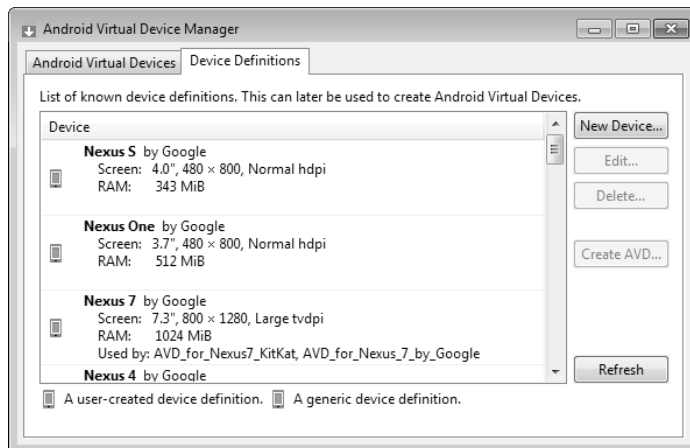


Fig. 2 | Android Virtual Device Manager window.

3. Google provides preconfigured devices that you can use to create AVDs. Select **Nexus 4 by Google**, then click **Create AVD...** to display the **Create new Android Virtual Device (AVD)** window (Fig. 3), then configure the options as shown and click **OK** to create the AVD. If you check **Hardware keyboard present**, you'll be able to use your computer's keyboard to type data into apps that are running in the AVD, but this may prevent the soft keyboard from displaying on the screen. If your computer does not have a camera, you can select **Emulated** for the **Front Camera** and **Back Camera** options. Each AVD you create has many other options specified in its `config.ini`. You can modify this file as described at

<http://developer.android.com/tools/devices/managing-avds.html>

to more precisely match the hardware configuration of your device.

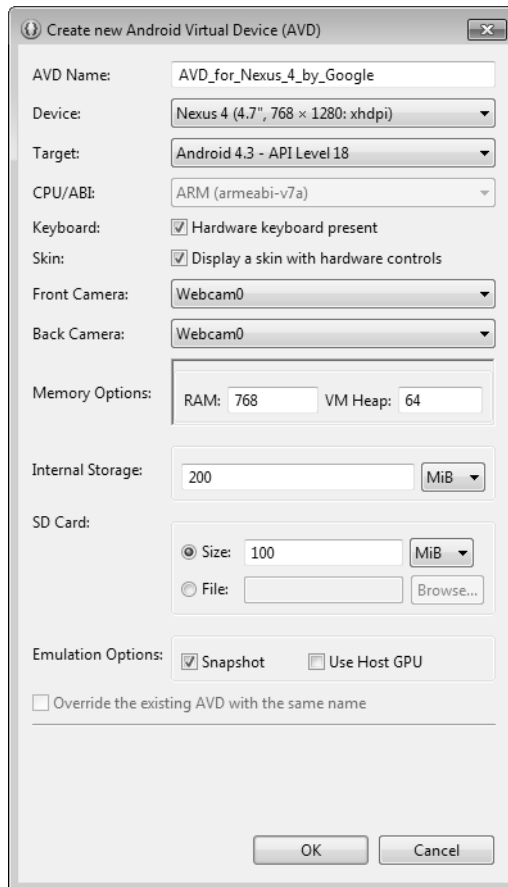


Fig. 3 | Configuring a Nexus 4 smartphone AVD for Android 4.3.

4. We also configured Android 4.3 AVDs that represent Nexus 7 by Google and Nexus 10 by Google for testing our tablet apps. Their settings are shown in Fig. 4. In

addition, we configured Android 4.4 AVDs for the Nexus 4, Nexus 7 and Nexus 10 with the names: AVD_for_Nexus_4_KitKat, AVD_for_Nexus_7_KitKat, and AVD_for_Nexus_10_KitKat,

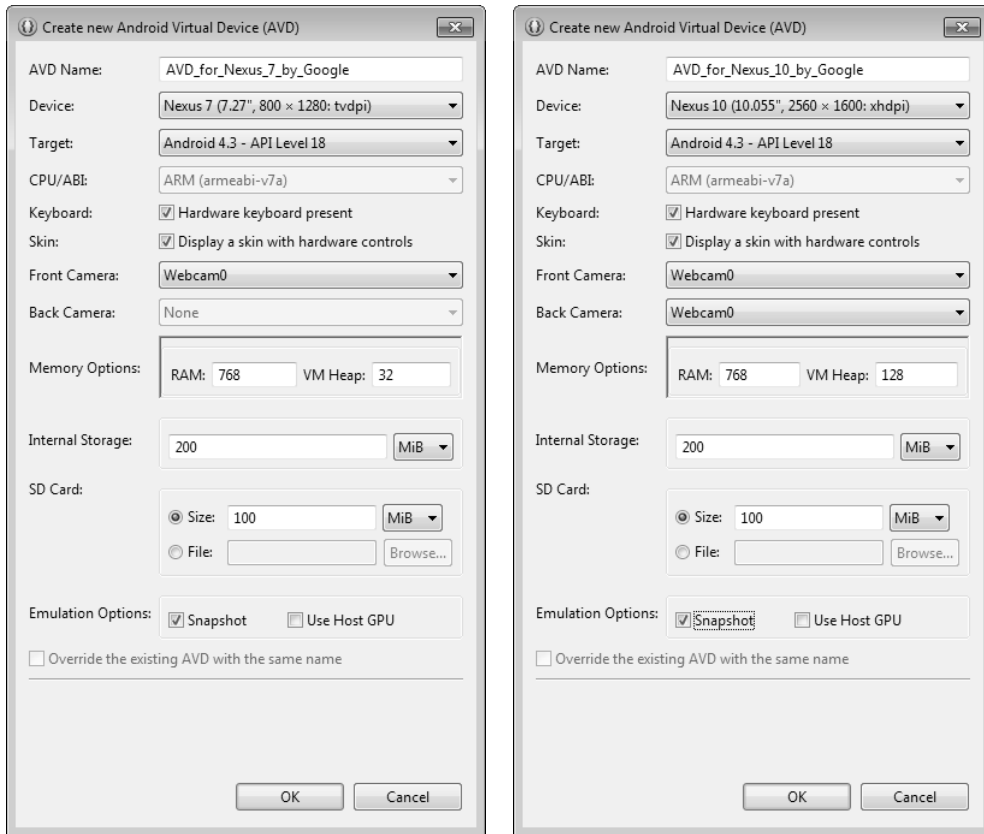


Fig. 4 | Configuring Nexus 7 and Nexus 10 tablet AVDs.

(Optional) Setting Up an Android Device for Development

As we mentioned, testing apps on AVDs can be slow due to AVD performance. If you have an Android device available to you, you should test the apps on that device. In addition, there are some features that you can test only on actual devices. To execute your apps on Android devices, follow the instructions at

<http://developer.android.com/tools/device.html>

If you're developing on Microsoft Windows, you'll also need the Windows USB driver for Android devices. In some cases on Windows, you may also need device-specific USB drivers. For a list of USB driver sites for various device brands, visit:

<http://developer.android.com/tools/extras/oem-usb.html>

Obtaining the Book's Code Examples

The examples for *Android How to Program, 2/e* are available for download at

www.pearsonglobal editions.com/Deitel

If you're not already registered at our website, go to www.deitel.com and click the **Register** link. Fill in your information. Registration is free, and we do not share your information with anyone. Please verify that you entered your registration e-mail address correctly—you'll receive a confirmation e-mail with your verification code. *You must click the verification link in the e-mail before you can sign in at www.deitel.com for the first time.* Configure your e-mail client to allow e-mails from [deitel.com](http://www.deitel.com) to ensure that the verification e-mail is not filtered as junk mail. We send only occasional account-management e-mails unless you register separately for our free *Deitel® Buzz Online* e-mail newsletter at

<http://www.deitel.com/newsletter/subscribe.html>

Next, visit www.deitel.com and sign in using the **Login** link below our logo in the upper-left corner of the page. Go to <http://www.deitel.com/books/AndroidHTP2/>. Click the **Examples** link to download a ZIP archive file containing the examples to your computer. Double click the ZIP file to unzip the archive, and make note of where you extract the file's contents on your system.

A Note Regarding the Android Development Tools

Google *frequently* updates the Android development tools. This often leads to problems compiling our apps when, in fact, the apps do not contain any errors. If you import one of our apps into Eclipse or Android Studio and it does not compile, there is probably a minor configuration issue. Please contact us by e-mail at deitel@deitel.com or by posting a question to:

- Facebook®—facebook.com/DeitelFan
- Google+™—google.com/+DeitelFan

and we'll help you resolve the issue.

You've now installed all the software and downloaded the code examples you'll need to study Android app development with *Android How to Program, 2/e* and to begin developing your own apps. Enjoy!

Introduction to Android

1

Objectives

In this chapter you'll be introduced to:

- The history of Android and the Android SDK.
- Google Play Store for downloading apps.
- The Android packages used in this book to help you create Android apps.
- Basic object-technology concepts.
- Key software for Android app development, including the Android SDK, the Java SDK, the Eclipse integrated development environment (IDE) and Android Studio.
- Important Android documentation.
- Test-driving an Android drawing app in Eclipse (in the print book) and in Android Studio (online).
- Characteristics of great Android apps.

-
- 1.1 Introduction
 - 1.2 Android—The World’s Leading Mobile Operating System
 - 1.3 Android Features
 - 1.4 Android Operating System
 - 1.4.1 Android 2.2 (Froyo)
 - 1.4.2 Android 2.3 (Gingerbread)
 - 1.4.3 Android 3.0 through 3.2 (Honeycomb)
 - 1.4.4 Android 4.0 through 4.0.4 (Ice Cream Sandwich)
 - 1.4.5 Android 4.1–4.3 (Jelly Bean)
 - 1.4.6 Android 4.4 (KitKat)
 - 1.5 Downloading Apps from Google Play
 - 1.6 Packages
 - 1.7 Android Software Development Kit (SDK)
 - 1.8 Object-Oriented Programming: A Quick Refresher
 - 1.8.1 The Automobile as an Object
 - 1.8.2 Methods and Classes
 - 1.8.3 Instantiation
 - 1.8.4 Reuse
 - 1.8.5 Messages and Method Calls
 - 1.8.6 Attributes and Instance Variables
 - 1.8.7 Encapsulation
 - 1.8.8 Inheritance
 - 1.8.9 Object-Oriented Analysis and Design (OOAD)
 - 1.9 Test-Driving the **Doodlz** App in an Android Virtual Device (AVD)
 - 1.9.1 Running the **Doodlz** App in the Nexus 4 Smartphone AVD
 - 1.9.2 Running the **Doodlz** App in a Tablet AVD
 - 1.9.3 Running the **Doodlz** App on an Android Device
 - 1.10 Building Great Android Apps
 - 1.11 Android Development Resources
 - 1.12 Wrap-Up
-

Self-Review Exercises | Answers to Self-Review Exercises | Exercises

1.1 Introduction

Welcome to Android app development! We hope that working with *Android How to Program, 2/e* will be an informative, challenging, entertaining and rewarding experience for you.

This portion of the book is geared toward students with *Java programming experience*. We use only complete working apps, so if you don’t know Java but have object-oriented programming experience in another language, such as C#, Objective-C/Cocoa or C++ (with class libraries), you should be able to master the material quickly, learning Java and Java-style object-oriented programming as you learn Android app development. If you do not know Java, we also provide a friendly, rich introduction to it in the book’s appendices.

App-Driven Approach

We use an **app-driven approach**—new features are discussed in the context of complete working Android apps, with one app per chapter. For each app, we first describe it, then have you *test-drive* it. Next, we briefly overview the key **Eclipse IDE** (integrated development environment), Java and **Android SDK** (Software Development Kit) technologies we use to implement the app. For apps that require it, we walk through designing the GUI *visually* using Eclipse. Then we provide the complete source-code listing, using line numbers, *syntax shading* and *code highlighting* to emphasize the key portions of the code. We also show one or more screen shots of the running app. Then we do a detailed code walk-through, emphasizing the new programming concepts introduced in the app. You can download the source code for all of the book’s apps from <http://www.deitel.com/books/AndroidHTP2/>.

For each chapter, we also provide **Android Studio** IDE versions of any Eclipse-specific instructions. Because Android Studio is an early access version and will be evolving rapidly, we provide the Android Studio instructions on the book’s website

<http://www.deitel.com/books/AndroidHTP2>

This will enable us to keep the instructions up to date.

1.2 Android—The World’s Leading Mobile Operating System

Android device sales are growing quickly, creating enormous opportunities for Android app developers.

- The first-generation Android phones were released in October 2008. By October 2013, a Strategy Analytics report showed that Android had 81.3% of the global *smartphone* market share, compared to 13.4% for Apple, 4.1% for Microsoft and 1% for Blackberry.¹
- According to an IDC report, by the end of the first quarter of 2013 Android had 56.5% of the global *tablet* market share, compared to 39.6% for Apple’s iPad and 3.7% for Microsoft Windows tablets.²
- As of April 2013, more than 1.5 million Android devices (including smartphones, tablets, etc.) were being activated daily.³
- At the time of this writing, there were over *one billion* activated Android devices.⁴
- Android devices now include smartphones, tablets, e-readers, robots, jet engines, NASA satellites, game consoles, refrigerators, televisions, cameras, health-care devices, smartwatches, automobile in-vehicle “infotainment” systems (for controlling the radio, GPS, phone calls, thermostat, etc.) and more.⁵

1.3 Android Features

Openness and Open Source

One benefit of developing Android apps is the openness of the platform. The operating system is *open source* and free. This allows you to view Android’s source code and see how its features are implemented. You can also contribute to Android by reporting bugs (see <http://source.android.com/source/report-bugs.html>) or by participating in the Open Source Project discussion groups (<http://source.android.com/community/index.html>). Numerous open-source Android apps from Google and others are available on the Internet

-
1. <http://blogs.strategyanalytics.com/WSS/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipments-in-Q3-2013.aspx>.
 2. <http://www.idc.com/getdoc.jsp?containerId=prUS24093213>.
 3. <http://www.technobuffalo.com/2013/04/16/google-daily-android-activations-1-5-million>.
 4. <http://venturebeat.com/2013/09/03/android-hits-1b-activations-and-will-be-called-kitkat-in-next-version>.
 5. <http://www.businessweek.com/articles/2013-05-29/behind-the-internet-of-things-is-android-and-its-everywhere>.

(Fig. 1.1). Figure 1.2 shows you where you can get the Android source code, learn about the philosophy behind the open-source operating system and get licensing information.

URL	Description
http://en.wikipedia.org/wiki/List_of_open_source_Android_applications	Extensive list of open-source apps, organized by category (e.g., games, communication, emulators, multimedia, security).
http://developer.android.com/tools/samples/index.html	Google's sample apps for the Android platform; includes over 60 apps and games such as Lunar Lander, Snake and Tic Tac Toe.
http://github.com/	GitHub allows you to share your apps and source code and contribute to others' open-source projects.
http://sourceforge.net	SourceForge also allows you to share apps and source code and contribute to others' open-source projects.
http://f-droid.org/	Hundreds of free and open-source Android apps including the Adblock Plus advertisement blocker, aMetro public transportation navigation, AnySoftKeyboard (available in several languages), Apollo music player, Chinese Checkers game, DroidWeight weight tracker, Earth Live Wallpaper and more.
http://blog.interstellr.com/post/39321551640/14-great-android-apps-that-are-also-open-source	Lists 14 open-source Android apps with links to the code.
http://www.openintents.org/en/libraries	Provides nearly 100 open-source libraries that can be used to enhance app capabilities.
http://www.androidviews.net	Customized GUI controls for enhancing your app's appearance.
http://www.stackoverflow.com	Stack Overflow is a question-and-answer website for programmers. Users can vote on each answer, and the best responses rise to the top.

Fig. 1.1 | Open-source Android app and library resource sites.

Title	URL
Get Android Source Code	http://source.android.com/source/downloading.html
Governance Philosophy	http://source.android.com/about/philosophy.html
Licenses	http://source.android.com/source/licenses.html
FAQs	http://source.android.com/source/faqs.html

Fig. 1.2 | Resources and source code for the open-source Android operating system.

The openness of the platform spurs rapid innovation. Unlike Apple's *proprietary* iOS, which is available only on Apple devices, Android is available on devices from dozens of orig-

inal equipment manufacturers (OEMs) and through numerous telecommunications carriers worldwide. The intense competition among OEMs and carriers benefits customers.

Java

Android apps are developed with Java—one of the world’s most widely used programming languages. Java was a logical choice for the Android platform, because it’s powerful, free, open source and millions of developers already know it. Experienced Java programmers can quickly dive into Android development, using Google’s Android APIs (Application Programming Interfaces) and others available from third parties.

Java is object oriented and has access to extensive class libraries that help you develop powerful apps quickly. GUI programming in Java is *event driven*—in this book, you’ll write apps that respond to various user-initiated *events* such as *screen touches*. In addition to directly programming portions of your apps, you’ll also use the Eclipse and Android Studio IDEs to conveniently drag and drop predefined objects such as buttons and text-boxes into place on your screen, and label and resize them. Using these IDEs, you can create, run, test and debug Android apps quickly and conveniently.

Multi-touch Screen

Android smartphones wrap the functionality of a mobile phone, Internet client, MP3 player, gaming console, digital camera and more into a handheld device with full-color *multi-touch screens*. With the touch of your fingers, you can navigate easily between using your phone, running apps, playing music, web browsing and more. The screen can display a keyboard for typing e-mails and text messages and entering data in apps (some Android devices also have physical keyboards).

Gestures

The multi-touch screens allow you to control the device with *gestures* involving one touch or multiple simultaneous touches (Fig. 1.3).

Gesture name	Physical action	Used to
Touch	Tap the screen once.	Open an app, “press” a button or a menu item.
Double touch	Tap the screen twice.	Zoom in on pictures, Google Maps and web pages. Tap the screen twice again to zoom back out.
Long press	Touch the screen and hold your finger in position.	Select items in a view—for example, checking an item in a list.
Swipe	Touch the screen, then move your finger in the swipe direction and release.	Flip item-by-item through a series, such as photos. A swipe automatically stops at the next item.
Drag	Touch and drag your finger across the screen.	Move objects or icons, or scroll through a web page or list.
Pinch zoom	Pinch two fingers together, or spread them apart.	Zoom in and out on the screen (e.g., resizing text and pictures).

Fig. 1.3 | Some common android gestures.

Built-in Apps

Android devices come with several default apps, which may vary, depending on the device, the manufacturer or the mobile service carrier. These typically include **Phone**, **People**, **Email**, **Browser**, **Camera**, **Photos**, **Messaging**, **Calendar**, **Play Store**, **Calculator** and more.

Web Services

Web services are software components stored on one computer that can be accessed by an app (or other software component) on another computer over the Internet. With web services, you can create **mashups**, which enable you to rapidly develop apps by quickly *combining* complementary web services, often from different organizations and possibly other forms of information feeds. For example, 100 Destinations (www.100destinations.co.uk) combines the photos and tweets from Twitter with the mapping capabilities of Google Maps to allow you to explore countries around the world through the photos of others.

Programmableweb (<http://www.programmableweb.com/>) provides a directory of over 9,400 APIs and 7,000 mashups, plus how-to guides and sample code for creating your own mashups. Figure 1.4 lists some popular web services. According to Programmableweb, the three most widely used APIs for mashups are Google Maps, Twitter and YouTube.

Web services source	How it's used
Google Maps	Mapping services
Twitter	Microblogging
YouTube	Video search
Facebook	Social networking
Instagram	Photo sharing
Foursquare	Mobile check-in
LinkedIn	Social networking for business
Groupon	Social commerce
Netflix	Movie rentals
eBay	Internet auctions
Wikipedia	Collaborative encyclopedia
PayPal	Payments
Last.fm	Internet radio
Amazon eCommerce	Shopping for books and lots of other products
Salesforce.com	Customer Relationship Management (CRM)
Skype	Internet telephony
Microsoft Bing	Search
Flickr	Photo sharing
Zillow	Real-estate pricing
Yahoo Search	Search
WeatherBug	Weather

Fig. 1.4 | Some popular web services (<http://www.programmableweb.com/apis/directory/1?sort=mashups>).

1.4 Android Operating System

The Android operating system was developed by Android, Inc., which was acquired by Google in 2005. In 2007, the Open Handset Alliance™—which now has 84 company members (http://www.openhandsetalliance.com/oha_members.html)—was formed to develop, maintain and evolve Android, driving innovation in mobile technology and improving the user experience while reducing costs.

Android Version Naming Convention

Each new version of Android is named after a dessert, going in alphabetical order (Fig. 1.5).

Android version	Name
Android 1.5	Cupcake
Android 1.6	Donut
Android 2.0–2.1	Eclair
Android 2.2	Froyo
Android 2.3	Gingerbread
Android 3.0–3.2	Honeycomb
Android 4.0	Ice Cream Sandwich
Android 4.1–4.3	Jelly Bean
Android 4.4	KitKat

Fig. 1.5 | Android version numbers and the corresponding names.

1.4.1 Android 2.2 (Froyo)

Android 2.2 (also called **Froyo**, released in May 2010) introduced external storage, allowing you to store apps on an external memory device rather than just in the Android device's internal memory. It also introduced the **Android Cloud to Device Messaging (C2DM)** service. **Cloud computing** allows you to use software and data stored in the “cloud”—i.e., accessed on remote computers (or servers) via the Internet and available on demand—rather than having it stored on your desktop, notebook computer or mobile device. Cloud computing gives you the flexibility to increase or decrease computing resources to meet your resource needs at any given time, making it more cost effective than purchasing expensive hardware to ensure that you have enough storage and processing power for occasional peak levels. Android C2DM allows app developers to send data from their servers to their apps installed on Android devices, even when the apps are *not* currently running. The server notifies the apps to contact it directly to receive updated app or user data.⁶ C2DM is now deprecated in favor of Google Cloud Messaging.

For information about additional Android 2.2 features—OpenGL ES 2.0 graphics capabilities, the media framework and more—visit <http://developer.android.com/about/versions/android-2.2-highlights.html>.

6. <http://code.google.com/android/c2dm/>.

1.4.2 Android 2.3 (Gingerbread)

Android 2.3 (Gingerbread), released later in 2010, added more user refinements, such as a redesigned keyboard, improved navigation capabilities, increased power efficiency and more. It also added several developer features for communications (e.g., technologies that make it easier to make and receive calls from within an app), multimedia (e.g., new audio and graphics APIs) and gaming (e.g., improved performance and new sensors, such as a gyroscope for better motion processing).

One of the most significant new features in Android 2.3 was support for **near-field communication (NFC)**—a short-range wireless connectivity standard that enables communication between two devices within a few centimeters. NFC support and features vary by Android device. NFC can be used for payments (for example, touching your NFC-enabled Android device to a payment device on a soda machine), exchanging data such as contacts and pictures, pairing devices and accessories and more.

For a more Android 2.3 developer features, see <http://developer.android.com/about/versions/android-2.3-highlights.html>.

1.4.3 Android 3.0 through 3.2 (Honeycomb)

Android 3.0 (Honeycomb) included user-interface improvements specifically for large-screen devices (e.g., tablets), such as a redesigned keyboard for more efficient typing, a visually appealing 3D user interface, easier navigation between screens within an app and more. New Android 3.0 developer features included:

- fragments, which describe portions of an app's user interface and can be combined into one screen or used across multiple screens
- a persistent Action Bar at the top of the screen providing users with options for interacting with apps
- the ability to add large-screen layouts to existing apps designed for small screens to optimize your app for use on different screen sizes
- a visually attractive and more functional user interface, known as “Holo” for its holographic look and feel
- a new animation framework
- improved graphics and multimedia capabilities
- ability to use multicore processor architectures for enhanced performance
- increased Bluetooth support (e.g., enabling an app to determine if there are any connected devices such as headphones or a keyboard)
- and an animation framework for animating user-interface or graphics objects.

For a list of Android 3.0 user and developer features and platform technologies, go to <http://developer.android.com/about/versions/android-3.0-highlights.html>.

1.4.4 Android 4.0 through 4.0.4 (Ice Cream Sandwich)

Android 4.0 (Ice Cream Sandwich), released in 2011, merged Android 2.3 (Gingerbread) and Android 3.0 (Honeycomb) into one operating system for use on all Android devices. This allowed you to incorporate into your smartphone apps Honeycomb's features that

previously were available only on tablets—the “Holo” user interface, a new launcher (used to customize the device’s home screen and launch apps) and more—and easily scale your apps to work on different devices. Ice Cream Sandwich also added several APIs for improved communication between devices, accessibility for users with disabilities (e.g., vision impairments), social networking and more (Fig. 1.6). For a complete list of Android 4.0 APIs, see <http://developer.android.com/about/versions/android-4.0.html>.

Feature	Description
Face detection	Using the camera, compatible devices can determine the positioning of the user’s eyes, nose and mouth. The camera can also track the user’s eye movement, allowing you to create apps that change perspective, based on where the user is looking.
Virtual camera operator	When filming video of multiple people, the camera will automatically focus on the person who is speaking.
Android Beam	Using NFC, Android Beam allows you to touch two Android devices to share content (e.g., contacts, pictures, videos).
Wi-Fi Direct	Wi-Fi P2P (peer-to-peer) APIs allow you to connect multiple Android devices using Wi-Fi. The devices can communicate wirelessly at a greater distance than when using Bluetooth.
Social API	Access and share contact information across social networks and apps (with the user’s permission).
Calendar API	Add and share events across multiple apps, manage alerts and attendees and more.
Accessibility APIs	Use the new Accessibility Text-to-Speech APIs to enhance the user experience of your apps for people with disabilities such as vision impairments and more. The explore-by-touch mode allows users with vision impairments to touch anywhere on the screen and hear a voice description of the touched content.
Android@Home framework	Use the Android@Home framework to create apps that control appliances in users’ homes, such as, thermostats, irrigation systems, networked light bulbs and more.
Bluetooth Health Devices	Create apps that communicate with Bluetooth health devices such as scales, heart-rate monitors and more.

Fig. 1.6 | Some Android Ice Cream Sandwich developer features (<http://developer.android.com/about/versions/android-4.0.html>).

1.4.5 Android 4.1–4.3 (Jelly Bean)

Android Jelly Bean, released in 2012, includes support for external displays, improved security, appearance enhancements (e.g., resizable app widgets and larger app notifications) and performance improvements that make switching between apps and screens more seamless (Fig. 1.7). For the Jelly Bean features list, see <http://developer.android.com/about/versions/jelly-bean.html>.

Feature	Description
Android Beam	You can use Android Beam to easily pair your smartphone or tablet with wireless Bluetooth® speakers or special headphones.
Lock screen widgets	Create widgets that appear on the user's screen when the device is locked, or modify your existing home-screen widgets so that they're also visible when the device is locked.
Photo Sphere	APIs for working with the new panoramic photo features that enable users to take 360-degree photos, similar to those used for Google Maps Street View.
Daydreams	Daydreams are interactive screensavers that are activated when a device is docked or charging. Daydreams can play audio and video and respond to user interactions.
Language support	New features help your apps reach international users, such as bidirectional text (left-to-right or right-to-left), international keyboards, additional keyboard layouts and more.
Developer options	Several new tracking and debugging features help you improve your apps, such as bug reports that include a screen shot and device state information.

Fig. 1.7 | Some Android Jelly Bean features (<http://developer.android.com/about/versions/jelly-bean.html>).

1.4.6 Android 4.4 (KitKat)

Android 4.4 KitKat, released in October 2013, includes several performance improvements that make it possible to run the operating system on all Android devices, including older, memory-constrained devices, which are particularly popular in developing countries.⁷

Enabling more users to update to KitKat will reduce the “fragmentation” of Android versions in the market, which has been a challenge for developers who previously had to design apps to run across multiple versions of the operating system, or limit their potential market by targeting their apps to a specific version of the operating system.

Android KitKat also includes security and accessibility enhancements, improved graphics and multimedia capabilities, memory-use analysis tools and more. Figure 1.8 lists some of the key new KitKat features. For a complete list, see

<http://developer.android.com/about/versions/kitkat.html>

Feature	Description
Immersive mode	The status bar at the top of the screen and the menu buttons at the bottom can be hidden, allowing your apps to fill more of the screen. Users can access the status bar by swiping down from the top of the screen, and the system bar (with the back button, home button and recent apps button) by swiping up from the bottom.

Fig. 1.8 | Some Android KitKat features (<http://developer.android.com/about/versions/kitkat.html>). (Part 1 of 2.)

7. <http://techcrunch.com/2013/10/31/android-4-4-kitkat-google/>.

Feature	Description
Printing framework	Build printing functionality into your apps, including locating available printers over Wi-Fi or the cloud, selecting the paper size and specifying which pages to print.
Storage access framework	Create document storage providers that allow users to browse, create and edit files (e.g., documents and images) across multiple apps.
SMS provider	Create SMS (Short Message Service) or MMS (Multimedia Messaging Service) apps using the new SMS provider and APIs. Users can now select their default messaging app.
Transitions framework	The new framework makes it easier to create transition animations.
Screen recording	Record video of your app in action to create tutorials and marketing materials.
Enhanced accessibility	The captioning manager API allows apps to check the user's captioning preferences (e.g., language, text styles and more).
Chromium WebView	Supports the latest standards for displaying web content including HTML5, CSS3 and a faster version of JavaScript.
Step detector and step counter	Create apps that detect whether the user is running, walking or climbing stairs and count the number of steps.
Host Card Emulator (HCE)	HCE enables any app to perform secure NFC transactions (e.g., mobile payments) without the need for a secure element on the SIM card controlled by the wireless carrier.

Fig. 1.8 | Some Android KitKat features (<http://developer.android.com/about/versions/kitkat.html>). (Part 2 of 2.)

1.5 Downloading Apps from Google Play

At the time of this writing, there were over one million apps in **Google Play**, and the number is growing quickly.⁸ Figure 1.9 lists some popular free and fee-based apps. You can download apps through the **Play Store** app installed on the device. You can also log into your Google Play account at <http://play.google.com> through your web browser, then specify the Android device on which to install the app. It will then download via the device's WiFi or 3G/4G connection. In Chapter 9, Google Play and App Business Issues, we discuss additional app stores, offering your apps for free or charging a fee, app pricing and more.

Google Play category	Some popular apps in the category
Books and Reference	Kindle, Wikipedia, Audible for Android, Google Play Books
Business	Office Suite Pro 7, Job Search, Square Register, GoToMeeting

Fig. 1.9 | Some popular Android apps in Google Play. (Part 1 of 2.)

8. en.wikipedia.org/wiki/Google_Play.

Google Play category	Some popular apps in the category
Comics	ComicRack, Memedroid Pro, Marvel Comics, Comic Strips
Communication	Facebook Messenger, Skype™, Groove IP
Education	Duolingo: Learn Languages Free, TED, Mobile Observatory
Entertainment	SketchBook Mobile, Netflix, Fandango® Movies, iFunny :)
Finance	Mint.com Personal Finance, Google Wallet, PayPal
Games: Arcade & Action	Minecraft—Pocket Edition, Fruit Ninja, Angry Birds
Games: Brain & Puzzle	Where's My Water?, Draw Something, Can You Escape
Games: Cards & Casino	Solitaire, Slots Delux, UNO™ & Friends, DH Texas Poker
Games: Casual	Candy Crush Saga, Hardest Game Ever 2, Game Dev Story
Health & Fitness	RunKeeper, Calorie Counter, Workout Trainer, WebMD®
Lifestyle	Zillow Real Estate, Epicurious Recipe App, Family Locator
Live Wallpaper	PicsArt, GO Launcher EX, Beautiful Widgets Pro
Media & Video	MX Player, YouTube, KeepSafe Vault, RealPlayer®
Medical	Epocrates, ICE: In Case of Emergency, Medscape®
Music & Audio	Pandora®, Shazam, Spotify, Ultimate Guitar Tabs & Chords
News & Magazines	Flipboard, Pulse News, CNN, Engadget, Dripler
Personalization	Beautiful Widgets Pro, Zedge™, GO Launcher EX
Photography	Camera ZOOM FX, Photo Grid, InstaPicFrame for Instagram
Productivity	Adobe® Reader®, Dropbox, Google Keep, SwiftKey Keyboard
Shopping	eBay, Amazon Mobile, Groupon, The Coupons App
Social	Facebook®, Instagram, Vine, Twitter, Snapchat, Pinterest
Sports	SportsCenter for Android, NFL '13, Team Stream™
Tools	Titanium Backup PRO, Google Translate, Tiny Flashlight®
Transportation	Uber, Trapster, Lyft, Hailo™, Ulysse Speedometer
Travel & Local	Waze, GasBuddy, KAYAK, TripAdvisor, OpenTable®
Weather	WeatherBug, AccuWeather, The Weather Channel
Widgets	Zillow, DailyHoroscope, Starbucks, Family Locator

Fig. 1.9 | Some popular Android apps in Google Play. (Part 2 of 2.)

1.6 Packages

Android uses a collection of *packages*, which are named groups of related, predefined classes. Some of the packages are Android specific, some are Java specific and some are Google specific. These packages allow you to conveniently access Android OS features and incorporate them into your apps. The Android packages help you create apps that adhere to Android's unique look-and-feel conventions and style guidelines (<http://developer.android.com/design/index.html>). Figure 1.10 lists the packages we discuss in this book. For a complete list of Android packages, see developer.android.com/reference/packages.html.

Package	Description
<code>android.app</code>	Includes high-level classes in the Android app model. (Chapter 3's Tip Calculator app.)
<code>android.content</code>	Access and publish data on a device. (Chapter 6's Cannon Game app.)
<code>android.content.res</code>	Classes for accessing app resources (e.g., media, colors, drawables, etc.), and device-configuration information affecting app behavior. (Chapter 5's Flag Quiz app.)
<code>android.database</code>	Handling data returned by the content provider. (Chapter 8's Address Book app.)
<code>android.database.sqlite</code>	SQLite database management for private databases. (Chapter 8's Address Book app.)
<code>android.graphics</code>	Graphics tools used for drawing to the screen. (Chapter 5's Flag Quiz app and Chapter 7's Doodlz app.)
<code>android.hardware</code>	Device hardware support. (Chapter 7's Doodlz app.)
<code>android.media</code>	Classes for handling audio and video media interfaces. (Chapter 6's Cannon Game app.)
<code>android.net</code>	Network access classes. (Chapter 4's Twitter® Searches app.)
<code>android.os</code>	Operating-systems services. (Chapter 3's Tip Calculator app.)
<code>android.preference</code>	Working with an app's user preferences. (Chapter 5's Flag Quiz app.)
<code>android.provider</code>	Access to Android content providers. (Chapter 7's Doodlz app.)
<code>android.support.v4.print</code>	Android Support Library features for using the Android 4.4 printing framework. (Chapter 7's Doodlz app.)
<code>android.text</code>	Rendering and tracking text on a device. (Chapter 3's Tip Calculator app.)
<code>android.util</code>	Utility methods and XML utilities. (Chapter 6's Cannon Game app.)
<code>android.widget</code>	User-interface classes for widgets. (Chapter 3's Tip Calculator app.)
<code>android.view</code>	User interface classes for layout and user interactions. (Chapter 4's Twitter® Searches app.)
<code>java.io</code>	Streaming, serialization and file-system access of input and output facilities. (Chapter 5's Flag Quiz app.)
<code>java.text</code>	Text formatting classes. (Chapter 4's Twitter® Searches app.)
<code>java.util</code>	Utility classes. (Chapter 4's Twitter® Searches app.)
<code>android.graphics.drawable</code>	Classes for display-only elements (e.g., gradients, etc.). (Chapter 5's Flag Quiz app.)

Fig. 1.10 | Android and Java packages used in this book, listed with the chapter in which they first appear.

1.7 Android Software Development Kit (SDK)

The Android SDK provides the tools you'll need to build Android apps. It's available at no charge through the Android Developers' site. See the Before You Begin section for details on downloading the Android app-development tools you'll need to develop Android apps, including the Java SE, the Android SDK/ADT Bundle (which includes the Eclipse IDE) and the Android Studio IDE.

Android SDK/ADT Bundle

The Android SDK/ADT Bundle—which includes the Eclipse IDE—is the most widely integrated development environment for Android development. Some developers use only a text editor and command-line tools to create Android apps. The Eclipse IDE includes:

- Code editor with support for syntax coloring and line numbering
- Auto-indenting and auto-complete (i.e., type hinting)
- Debugger
- Version control system
- Refactoring support

You'll use Eclipse in Section 1.9 to test-drive the **Doodlz** app. Starting in Chapter 2, **Welcome App**, you'll use Eclipse to build apps.

Android Studio

Android Studio, a new Android IDE based on the JetBrains IntelliJ IDEA Java IDE (<http://www.jetbrains.com/idea/>), was announced in 2013 and is Google's preferred Android IDE of the future. At the time of this writing, Android Studio was available only as an *early access preview*—many of its features were still under development. For each chapter, we also provide Android Studio versions of any Eclipse-specific instructions on the book's website

<http://www.deitel.com/books/AndroidHTP2>

To learn more about Android Studio, installing it and migrating from Eclipse, visit <http://developer.android.com/sdk/installing/studio.html>.

Android Development Tools (ADT) Plugin for Eclipse

The **Android Development Tools (ADT) Plugin for Eclipse** (part of the Android SDK/ADT Bundle) allows you to create, run and debug Android apps, export them for distribution (e.g., upload them to Google Play), and more. ADT also includes a visual GUI design tool. GUI components can be dragged and dropped into place to form GUIs without any coding. You'll learn more about ADT in Chapter 2.

The Android Emulator

The **Android emulator**, included in the Android SDK, allows you to run Android apps in a simulated environment within Windows, Mac OS X or Linux, without using an actual Android device. The emulator displays a realistic Android user-interface window. It's particularly useful if you do not have access to Android devices for testing. You should certainly test your apps on a variety of Android devices before uploading them to Google Play.

Before running an app in the emulator, you'll need to create an **Android Virtual Device (AVD)**, which defines the characteristics of the device on which you want to test, including the hardware, system image, screen size, data storage and more. If you want to test your apps for multiple Android devices, you'll need to create separate AVDs to emulate each unique device, or use a service (like testdroid.com or appthwack.com) that enables you to test on many different devices.

We used the emulator (not an actual Android device) to take most but not all of the Android screen shots for this book. You can reproduce on the emulator most of the

Android gestures (Fig. 1.11) and controls (Fig. 1.12) using your computer's keyboard and mouse. The gestures on the emulator are a bit limited, since your computer probably cannot simulate all the Android hardware features. For example, to test GPS apps in the emulator, you'll need to create files that simulate GPS readings. Also, although you can simulate orientation changes (to *portrait* or *landscape* mode), simulating particular **accelerometer** readings (the accelerometer allows the device to respond to up/down, left/right and forward/backward acceleration) requires features that are not built into the emulator. There is a *Sensor Simulator* available at

<https://code.google.com/p/openintents/wiki/SensorSimulator>

that you can use to send simulated sensor information into an AVD to test other sensor features in your apps. Figure 1.13 lists Android functionality that's *not* available on the emulator. You can, however, upload your app to an Android device to test these features. You'll start creating AVDs and using the emulator to develop Android apps in Chapter 2's **Welcome** app.

Gesture	Emulator action
Touch	Click the mouse once. Introduced in Chapter 3's Tip Calculator app.
Double touch	Double click the mouse. Introduced in Chapter 6's Cannon Game app.
Long press	Click and hold the mouse.
Drag	Click, hold and drag the mouse. Introduced in Chapter 6's Cannon Game app.
Swipe	Click and hold the mouse, move the pointer in the swipe direction and release the mouse. Introduced in Chapter 8's Address Book app.
Pinch zoom	Press and hold the <i>Ctrl</i> (<i>Control</i>) key. Two circles that simulate the two touches will appear. Move the circles to the start position, click and hold the mouse and drag the circles to the end position.

Fig. 1.11 | Android gestures on the emulator.

Control	Emulator action
Back	<i>Esc</i>
Call/dial button	<i>F3</i>
Camera	<i>Ctrl-KEYPAD_5</i> , <i>Ctrl-F3</i>
End call button	<i>F4</i>
Home	<i>Home</i> button
Menu (left softkey)	<i>F2</i> or <i>Page Up</i> button
Power button	<i>F7</i>

Fig. 1.12 | Android hardware controls on the emulator (for additional controls, go to <http://developer.android.com/tools/help/emulator.html>). (Part 1 of 2.)

Control	Emulator action
Search	<i>F5</i>
* (right softkey)	<i>Shift-F2</i> or <i>Page Down</i> button
Rotate to previous orientation	<i>KEYPAD_7, Ctrl-F11</i>
Rotate to next orientation	<i>KEYPAD_9, Ctrl-F12</i>
Toggle cell networking on/off	<i>F8</i>
Volume up button	<i>KEYPAD_PLUS, Ctrl-F5</i>
Volume down button	<i>KEYPAD_MINUS, Ctrl-F6</i>

Fig. 1.12 | Android hardware controls on the emulator (for additional controls, go to <http://developer.android.com/tools/help/emulator.html>). (Part 2 of 2.)

Android functionality not available on the emulator
<ul style="list-style-type: none">• Making or receiving real phone calls (the emulator allows simulated calls only)• Bluetooth• USB connections• Device-attached headphones• Determining connected state of the phone• Determining battery charge or power charging state• Determining SD card insert/eject• Sensors (accelerometer, barometer, compass, light sensor, proximity sensor)

Fig. 1.13 | Android functionality not available on the emulator (<http://developer.android.com/tools/devices/emulator.html>).

1.8 Object-Oriented Programming: A Quick Refresher

Android uses object-oriented programming techniques, so in this section we review the basics of object technology. We use all of these concepts in this book.

Building software quickly, correctly and economically remains an elusive goal at a time when demands for new and more powerful software are soaring. *Objects*, or more precisely the *classes* objects come from, are essentially *reusable* software components. There are date objects, time objects, audio objects, video objects, automobile objects, people objects, etc. Almost any *noun* can be reasonably represented as a software object in terms of *attributes* (e.g., name, color and size) and *behaviors* (e.g., calculating, moving and communicating). Software developers are discovering that using a modular, object-oriented design-and-implementation approach can make software development groups much more productive than they could be with earlier popular techniques like “structured programming”—object-oriented programs are often easier to understand, correct and modify.

1.8.1 The Automobile as an Object

To help you understand objects and their contents, let's begin with a simple analogy. Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal*. What must happen before you can do this? Well, before you can drive a car, someone has to *design* it. A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house. These drawings include the design for an accelerator pedal. The pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal *hides* the mechanisms that slow the car, and the steering wheel *hides* the mechanisms that turn the car. This enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.

Just as you cannot cook meals in the kitchen of a blueprint, you cannot drive a car's engineering drawings. Before you can drive a car, it must be *built* from the engineering drawings that describe it. A completed car has an *actual* accelerator pedal to make it go faster, but even that's not enough—the car won't accelerate on its own (hopefully!), so the driver must *press* the pedal to accelerate the car.

1.8.2 Methods and Classes

Let's use our car example to introduce some key object-oriented programming concepts. Performing a task in a program requires a **method**. The method houses the program statements that actually perform its tasks. The method hides these statements from its user, just as the accelerator pedal of a car hides from the driver the mechanisms of making the car go faster. A program unit called a **class** houses the methods that perform the class's tasks. For example, a class that represents a bank account might contain one method to *deposit* money to an account, another to *withdraw* money from an account and a third to *inquire* what the account's current balance is. A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.

1.8.3 Instantiation

Just as someone has to *build a car* from its engineering drawings before you can actually drive a car, you must *build an object* of a class before a program can perform the tasks that the class's methods define. The process of doing this is called *instantiation*. An object is then referred to as an **instance** of its class.

1.8.4 Reuse

Just as a car's engineering drawings can be *reused* many times to build many cars, you can *reuse* a class many times to build many objects. **Reuse** of existing classes when building new classes and programs saves time and effort. Reuse also helps you build more reliable and effective systems, because existing classes and components often have gone through extensive *testing*, *debugging* and *performance* tuning. Just as the notion of *interchangeable parts* was crucial to the Industrial Revolution, reusable classes are crucial to the software revolution that has been spurred by object technology.

1.8.5 Messages and Method Calls

When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster. Similarly, you *send messages to an object*. Each message is a **method call**

that tells a method of the object to perform its task. For example, a program might call a particular bank-account object's *deposit* method to increase the account's balance.

1.8.6 Attributes and Instance Variables

A car, besides having capabilities to accomplish tasks, also has *attributes*, such as its color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading). Like its capabilities, the car's attributes are represented as part of its design in its engineering diagrams (which, for example, include an odometer and a fuel gauge). As you drive an actual car, these attributes are carried along with the car. Every car maintains its *own* attributes. For example, each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.

An object, similarly, has attributes that it carries along as it's used in a program. These attributes are specified as part of the object's class. For example, a bank-account object has a *balance attribute* that represents the amount of money in the account. Each bank-account object knows the balance in the account it represents, but *not* the balances of the *other* accounts in the bank. Attributes are specified by the class's **instance variables**.

1.8.7 Encapsulation

Classes **encapsulate** (i.e., wrap) attributes and methods into objects—an object's attributes and methods are intimately related. Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves. This **information hiding** is crucial to good software engineering.

1.8.8 Inheritance

A new class of objects can be created quickly and conveniently by **inheritance**—the new class absorbs the characteristics of an existing one, possibly customizing them and adding unique characteristics of its own. In our car analogy, a “convertible” certainly *is an* object of the more *general* class “automobile,” but more *specifically*, the roof can be raised or lowered.

1.8.9 Object-Oriented Analysis and Design (OOAD)

How will you create the code for your programs? Perhaps, like many programmers, you'll simply turn on your computer and start typing. This approach may work for small programs, but what if you were asked to create a software system to control thousands of automated teller machines for a major bank? Or suppose you were asked to work on a team of 1,000 software developers building the next U.S. air traffic control system? For projects so large and complex, you should not simply sit down and start writing programs.

To create the best solutions, you should follow a detailed **analysis** process for determining your project's **requirements** (i.e., defining *what* the system is supposed to do) and developing a **design** that satisfies them (i.e., deciding *how* the system should do it). Ideally, you'd go through this process and carefully review the design (and have your design reviewed by other software professionals) before writing any code. If this process involves analyzing and designing your system from an object-oriented point of view, it's called an **object-oriented analysis and design (OOAD) process**. Languages like Java are object oriented. Programming in such a language, called **object-oriented programming (OOP)**, allows you to implement an object-oriented design as a working system.

1.9 Test-Driving the Doodlz App in an Android Virtual Device (AVD)

In this section, you'll run and interact with your first Android app. The **Doodlz** app allows you to drag your fingers on the screen to “paint.” You can control the brush sizes and colors using options provided in the app's *options menu*. There is no need to look at the app's code—you'll build the app and study its code in Chapter 7.

The following steps show how to import the app's project into Eclipse and how to test-drive the app in the Nexus 4 Android Virtual Device (AVD) that you set up in the Before You Begin section following the Preface. Later in this section, we'll also discuss how to run the app on a tablet AVD and on an Android device. When the app is running in an AVD, you can create a new painting by “dragging your finger” anywhere on the canvas. You “touch” the screen by using the mouse.

Android SDK/ADT Bundle and Android Studio IDEs



The IDE screen captures in the following steps (and throughout this book) were taken on a computer running Windows 7, the Java SE 7 JDK and the Android SDK/ADT Bundle that you installed in the Before You Begin section. Because Android Studio is an early access version and will be evolving rapidly, we provide the Android Studio instructions for this test-drive on the book's website

www.deitel.com/books/AndroidHTP2

This will enable us to update the instructions in response to Google's changes. Both the Android SDK/ADT Bundle and Android Studio use the *same* Android emulator, so once an app is running in an AVD, the steps are identical.

1.9.1 Running the Doodlz App in the Nexus 4 Smartphone AVD

To test-drive the **Doodlz** app, perform the following steps:

1. **Checking your setup.** If you have not done so already, perform the steps specified in the Before You Begin section located after the Preface.
2. **Opening Eclipse.** Open the `eclipse` subfolder of the Android SDK/ADT bundle's installation folder, then double click the Eclipse icon ( or , depending on your platform).
3. **Specifying your workspace location.** When the **Workspace Launcher** window appears, specify where you'd like the apps that you create to be stored, then click **OK**. We used the default location—a folder named `workspace` in your user directory. A **workspace** is a collection of projects, and each project is typically an app or a library that can be shared among apps. Each workspace also has its own settings, such as where various Eclipse subwindows are displayed. You can have many workspaces and switch between them for different development tasks—for example, you could have separate workspaces for Android app development, Java app development and web app development, each with its own custom settings. If this is your first time opening Eclipse, the **Welcome** page (Fig. 1.14) is displayed.

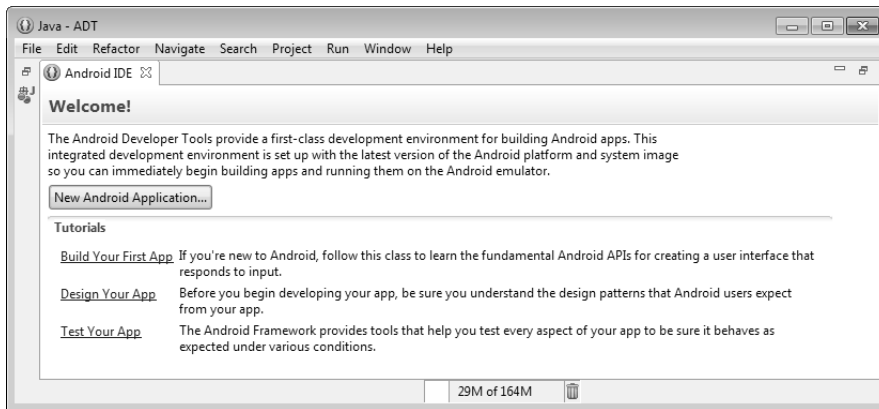


Fig. 1.14 | Welcome page in Eclipse.

4. *Launching the Nexus 4 AVD.* For this test-drive, we'll use the Nexus 4 smartphone AVD that you configured for Android 4.4 (KitKat) in the Before You Begin section—in Section 1.9.2, we'll show the app running in a tablet AVD. An AVD can take several minutes to load, so you should launch it in advance of when you intend to use it and keep it running in the background while you're building and testing your apps. To launch the Nexus 4 AVD, select **Window > Android Virtual Device Manager** to display the **Android Virtual Device Manager** dialog (Fig. 1.15). Select the Nexus 4 AVD for Android KitKat and click **Start...**, then click the **Launch** button in the **Launch Options** dialog that appears. You should not attempt to execute the app until the AVD finishes loading. Once the AVD appears as shown in Fig. 1.16, unlock the AVD by dragging the mouse pointer from the lock icon to the edge of the screen.

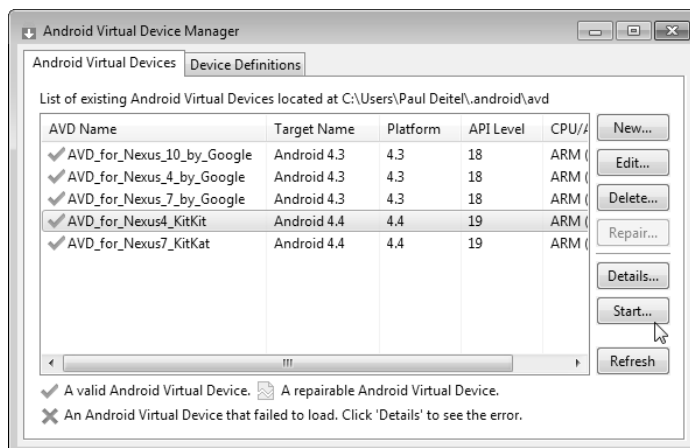


Fig. 1.15 | Android Virtual Device Manager dialog.

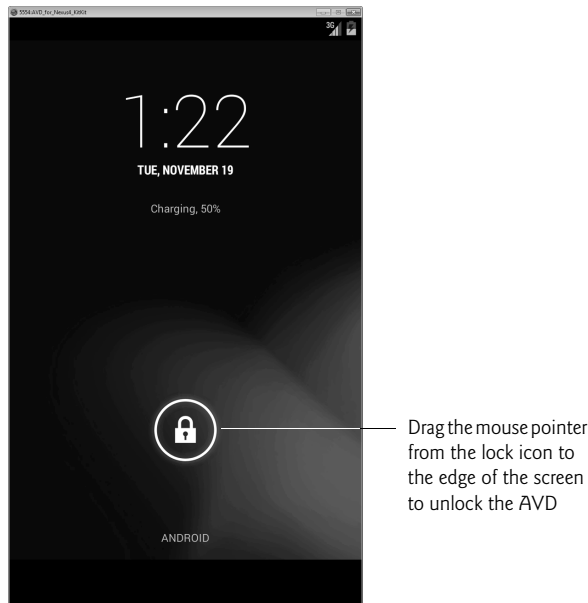


Fig. 1.16 | Nexus 4 AVD home screen (for Android 4.4) when the AVD finishes loading.

5. *Importing the Doodlz app's project.* Select **File > Import...** to open the **Import** dialog (Fig. 1.17(a)). Expand the **General** node and select **Existing Projects into Workspace**, then click **Next >** to proceed to the **Import Projects** step (Fig. 1.17(b)). Click the **Browse...** button to the right of the **Select root directory** textbox. In the

a) **Import** dialog

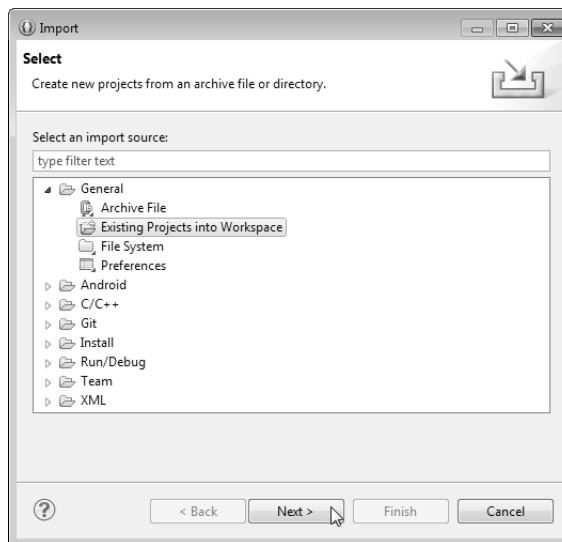


Fig. 1.17 | Importing an existing project. (Part 1 of 2.)

b) Import dialog's
Import Projects step

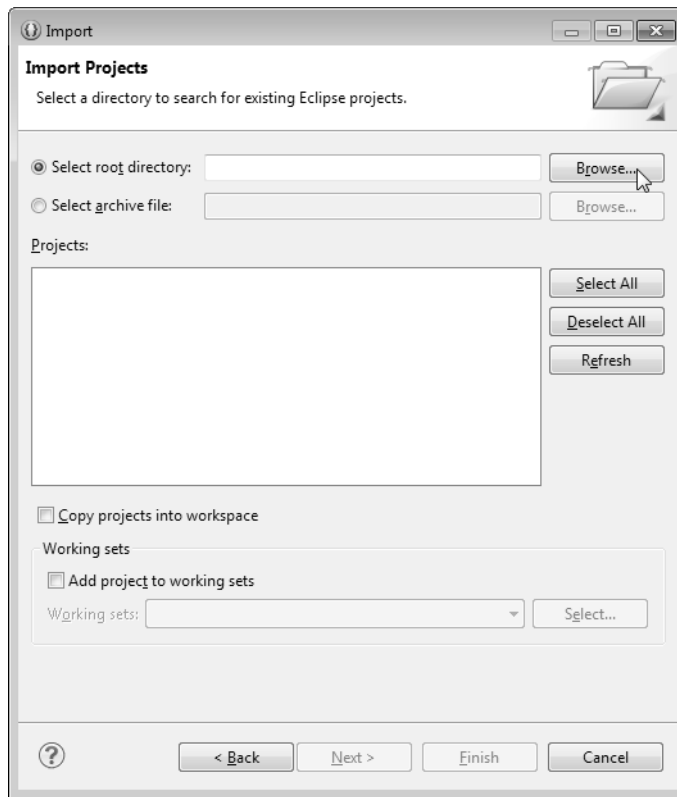


Fig. 1.17 | Importing an existing project. (Part 2 of 2.)

Browse For Folder dialog, locate the **Doodlz** folder in the book's examples folder, select it and click **Open**. Click **Finish** to import the project into Eclipse. The project now appears in the **Package Explorer** window (Fig. 1.18) at the left side of Eclipse. If the **Package Explorer** window is not visible, you can view it by selecting **Window > Show View > Package Explorer**.

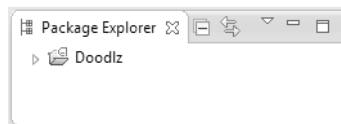


Fig. 1.18 | Package Explorer window.

6. *Launching the Doodlz app.* In Eclipse, right click the **Doodlz** project in the **Package Explorer** window, then select **Run As > Android Application** (Fig. 1.19). This will execute **Doodlz** in the AVD that you launched in Step 4 (Fig. 1.20).

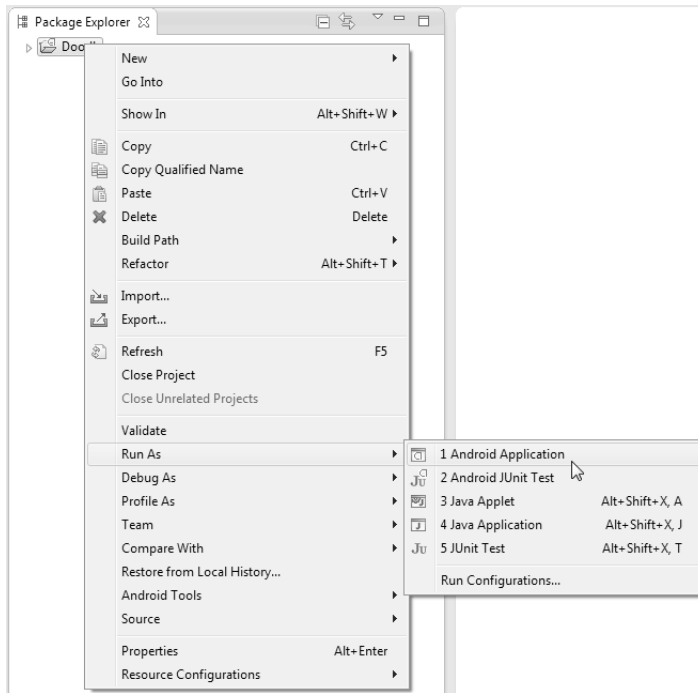


Fig. 1.19 | Launching the **Doodlz** app.

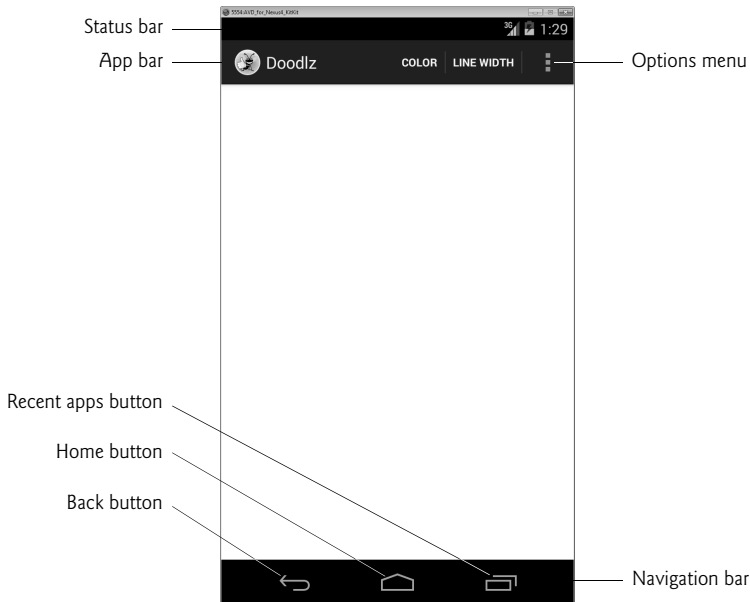


Fig. 1.20 | **Doodlz** app running in the Android Virtual Device (AVD).

7. *Exploring the AVD and immersive mode.* At the AVD screen's bottom are various **soft buttons** that appear on the device's touch screen. You touch these (by using the mouse in an AVD) to interact with apps and the Android OS. The *back button* goes back to the app's prior screen, or back to a prior app if you're in the current app's initial screen. The *home button* returns you to the device's home screen. The *recent apps button* allows you to view the recently used apps list, so that you can switch back to them quickly. At the screen's top is the app's *app bar*, which displays the app's icon and name and may contain other app-specific soft buttons—some appear on the app bar (**COLOR** and **LINE WIDTH** in Fig. 1.20) and the rest appear in the app's *options menu* (☰). The number of options on the app bar depends on the size of the device—we discuss this in Chapter 7. Android 4.4 supports a new *immersive mode* that enables apps to use the entire screen. In this app, you can tap once in the white drawing area to hide the device's status and navigation bars as well as the app's action bar. You can redisplay these by tapping the drawing area again or by swiping from the top edge of the screen.
8. *Understanding the app's options.* To display the options that do not appear on the app bar, touch (i.e., click) the options menu (☰) icon. Figure 1.21(a) shows the action bar and options menu on the Nexus 4 AVD and Fig. 1.21(b) shows them on a Nexus 7 AVD—options shown on the action bar appear in small capital letters. Touching **COLOR** displays a GUI for changing the line color. Touching **LINE WIDTH** displays a GUI for changing the thickness of the line that will be drawn. Touching **Eraser** sets the drawing color to white so that as you draw over colored areas, the color is erased. Touching **Clear** first confirms whether you wish to erase the entire image, then clears the drawing area if you do not cancel the action. Touching **Save Image** saves the image into the device's **Gallery** of images. On Android 4.4, touching **Print** displays a GUI for selecting an available printer so can print your image or save it as a PDF document (the default). You'll explore each of these options momentarily.

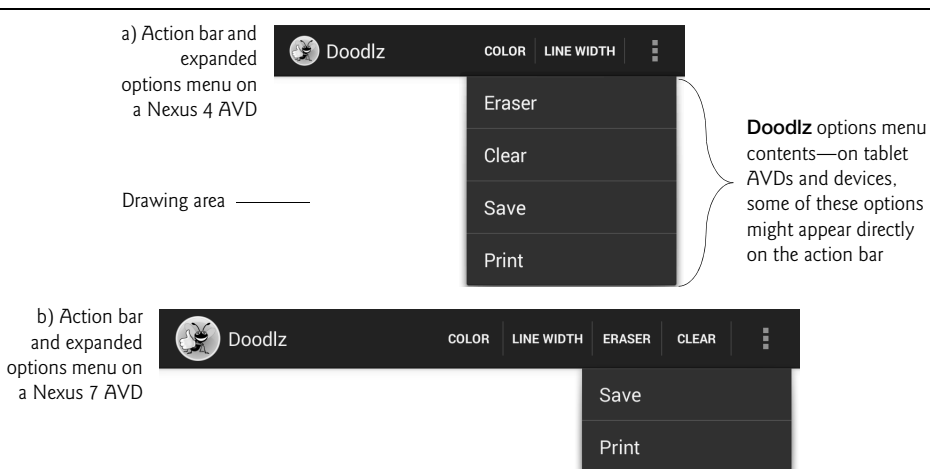


Fig. 1.21 | Doodlz options menu expanded.

9. *Changing the brush color to red.* To change the brush color, first touch **COLOR** on the action bar to display the **Choose Color** dialog (Fig. 1.22). Colors are defined using the *ARGB color scheme* in which the *alpha* (i.e., *transparency*), red, green and blue components are specified by integers in the range 0–255. For alpha, 0 means *completely transparent* and 255 means *completely opaque*. For red, green and blue, 0 means *none* of that color and 255 means the *maximum amount* of that color. The GUI consists of **Alpha**, **Red**, **Green** and **Blue** SeekBars that allow you to select the amount of alpha, red, green and blue in the drawing color. You drag the SeekBars to change the color. As you do, the app displays the new color below the SeekBars. Select a red color now by dragging the **Red** SeekBar to the right as in Fig. 1.22. Touch the **Set Color** button to return to the drawing area.

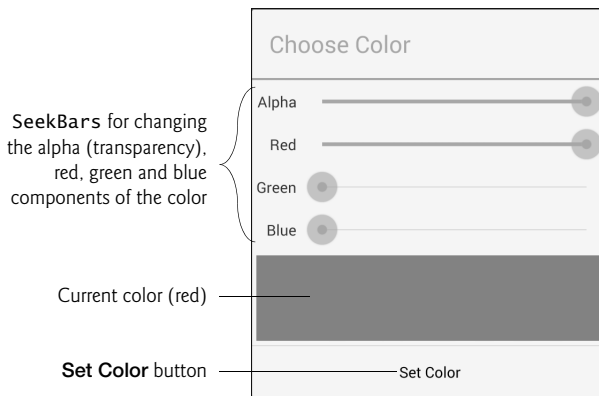


Fig. 1.22 | Changing the drawing color to red.

10. *Changing the line width.* To change the line width, touch **LINE WIDTH** on the action bar to display the **Choose Line Width** dialog. Drag the SeekBar for the line width to the right to thicken the line (Fig. 1.23). Touch the **Set Line Width** button to return to the drawing area.

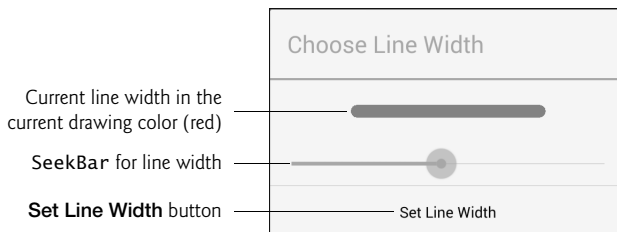


Fig. 1.23 | Changing the line thickness.

11. *Drawing the flower petals.* Tap the screen to enter immersive mode, then drag your “finger”—the mouse when using the emulator—on the drawing area to draw flower petals (Fig. 1.24).

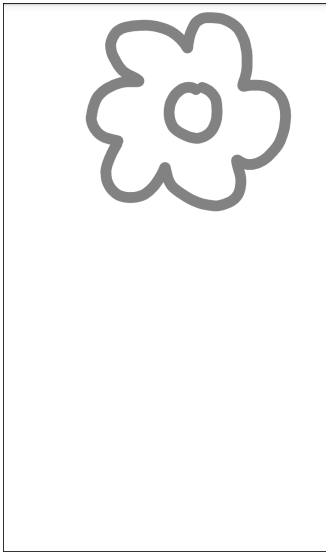
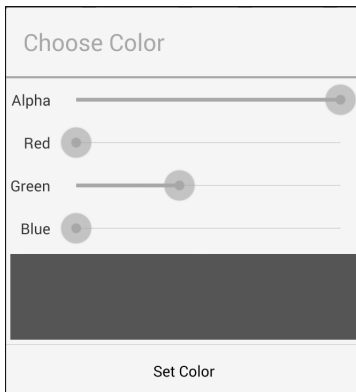


Fig. 1.24 | Drawing flower petals.

12. *Changing the brush color to dark green.* Tap the screen to leave immersive mode then touch **COLOR** to display the **Choose Color** dialog. Select a dark green color by dragging the **Green** SeekBar to the right and ensuring that the **Red** and **Blue** SeekBars are at the far left (Fig. 1.25(a)).

a) Selecting dark green as the drawing color



b) Selecting a thicker line

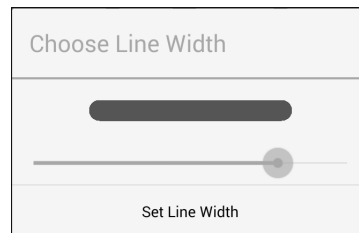


Fig. 1.25 | Changing the color to dark green and making the line thicker.

13. *Changing the line width and drawing the stem and leaves.* Touch **LINE WIDTH** to display the **Choose Line Width** dialog. Drag the SeekBar for the line width to the right to thicken the line (Fig. 1.25(b)). Tap the screen to re-enter immersive

mode, then draw the flower stem and leaves. Repeat Steps 12 and 13 for a lighter green color and thinner line, then draw the grass (Fig. 1.26).

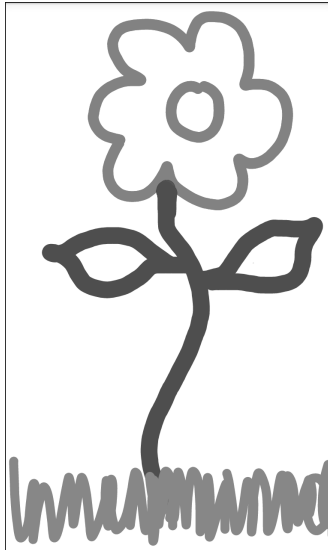
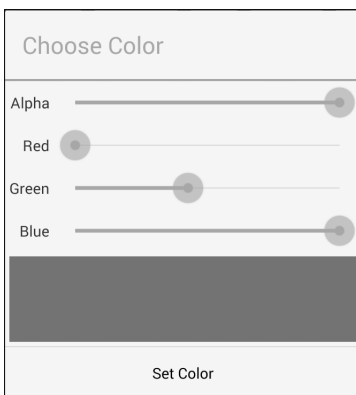


Fig. 1.26 | Drawing the stem and grass.

- 14. *Finishing the drawing.*** Tap the screen to exit immersive mode. Next, change the drawing color to blue (Fig. 1.27(a)) and select a narrower line (Fig. 1.27(b)). Then tap the screen to enter immersive mode and draw the raindrops (Fig. 1.28).

a) Selecting blue as the drawing color



b) Selecting a thinner line

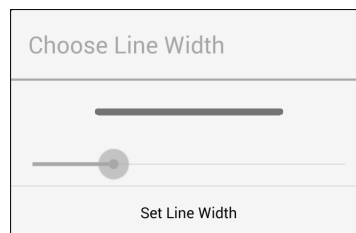





Fig. 1.27 | Changing the line color to blue and narrowing the line.



Fig. 1.28 | Drawing the rain in the new line color and line width.

15. ***Saving the image.*** You can save your image to the device's **Gallery** app by selecting **Save** from the options menu . You can then view this image and others stored on the device by opening the **Gallery** app.
16. ***Printing the image.*** To print the image, select **Print** from the options menu. This displays the print dialog, which allows you to save the image as a PDF document by default. To select a printer, tap **Save as PDF** and select from the available printers. If no printers appear in the list, you may need to configure Google Cloud Print for your printer. For information on this, visit

<http://www.google.com/cloudprint/learn/>

17. ***Returning to the home screen.*** You can return to the AVD's home screen by tapping the home () button on the AVD. To view the drawing in the **Gallery** app touch  to display the list of apps installed on the AVD. You can then open the **Gallery** app to view the drawing.

1.9.2 Running the Doodlz App in a Tablet AVD

To test the app in a tablet AVD, first launch the AVD by performing Step 4 in Section 1.9.1, but select the Nexus 7 AVD rather than the Nexus 4 AVD. Next, right click the **Doodlz** project in Eclipse's **Package Explorer** window and select **Run As > Android Application**. If multiple AVDs are running when you launch an app, the **Android Device Chooser** dialog (Fig. 1.29) appears so that you can choose the AVD on which to install and execute the app. In this case, both the Nexus 4 and Nexus 7 AVDs were running on our system, so there were two Android virtual devices on which we could possibly run the

app. Select the Nexus 7 AVD and click **OK**. This app runs in portrait orientation (the width is less than the height) on phone and small tablet devices. If you run the app on a large tablet AVD (or large tablet device) the app runs in landscape orientation (the width is greater than the height). Figure 1.30 shows the app running in the Nexus 7 AVD. If the AVD is too tall to display on your screen, you can change the AVD's orientation by typing *Ctrl* + *F12* (on a Mac use *fn* + *control* + *F12*). On some keyboards the *Ctrl* key is labeled *Control*.

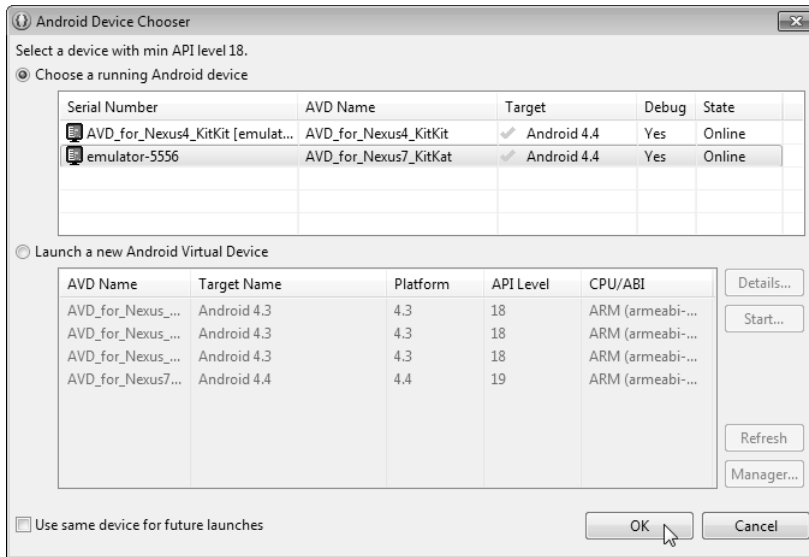


Fig. 1.29 | Android Device Chooser dialog.



Fig. 1.30 | Drawing in the Nexus 7 AVD.

1.9.3 Running the Doodlz App on an Android Device

If you have an Android device, you can easily execute an app on it for testing purposes.

1. *Enabling the developer options on the device.* First, you must enable debugging on the device. To do so, go to the device's **Settings** app, then select **About phone**, (or **About tablet**) locate the **Build number** (at the bottom of the list) and tap it repeatedly until you see the message **You are now a developer** on the screen. This will enable an entry named **Developer options** to the **Settings** app.
2. *Enabling debugging on the device.* Return to the **Settings** app, select **Developer options** and ensure that **USB debugging** is checked—this is the default when you first enable the developer options on the device.
3. *Connecting your device.* Next, connect the device to your computer via the USB cable that came with your device. If you're a Windows user, recall from the Before You Begin section that you might need to install a USB driver for your device. See the following two web pages for details:

```
developer.android.com/tools/device.html
developer.android.com/tools/extras/oem-usb.html
```

4. *Running Doodlz on the Android device.* In Eclipse, right click the **Doodlz** project in the **Package Explorer** window, then select **Run As > Android Application**. If you do not have any AVDs open, but do have an Android device connected, the IDE will automatically install the app on your device and execute it. If you have one or more AVDs open and/or devices connected, the **Android Device Chooser** dialog (Fig. 1.29) is displayed so that you can select the device or AVD on which to install and execute the app.

Preparing to Distribute Apps

When you build apps for distribution via app stores like Google Play, you should test the apps on as many actual devices as you can. Remember that some features can be tested *only* on actual devices. If you don't have many devices available to you, create AVDs that simulate the various devices on which you'd like your app to execute. When you configure each AVD to simulate a particular device, look up the device's specifications online and configure the AVD accordingly. In addition, you can modify the AVD's `config.ini` file as described in the section **Setting hardware emulation options** at

```
developer.android.com/tools/devices/
managing-avds-cmdline.html#hardwareopts
```

This file contains options that are not configurable via the **Android Virtual Device Manager**. Modifying these options allows you to more precisely match the hardware configuration of an actual device.

1.10 Building Great Android Apps

With over 800,000 apps in Google Play,⁹ how do you create an Android app that people will find, download, use and recommend to others? Consider what makes an app fun, useful, in-

9. <http://www.pureoxygenmobile.com/how-many-apps-in-each-app-store/>.

teresting, appealing and enduring. A clever app name, an attractive icon and an engaging description might lure people to your app on Google Play or one of the many other Android app marketplaces. But once users download the app, what will make them use it regularly and recommend it to others? Figure 1.31 shows some characteristics of great apps.



Fig. 1.31 | Characteristics of great apps. (Part 1 of 2.)

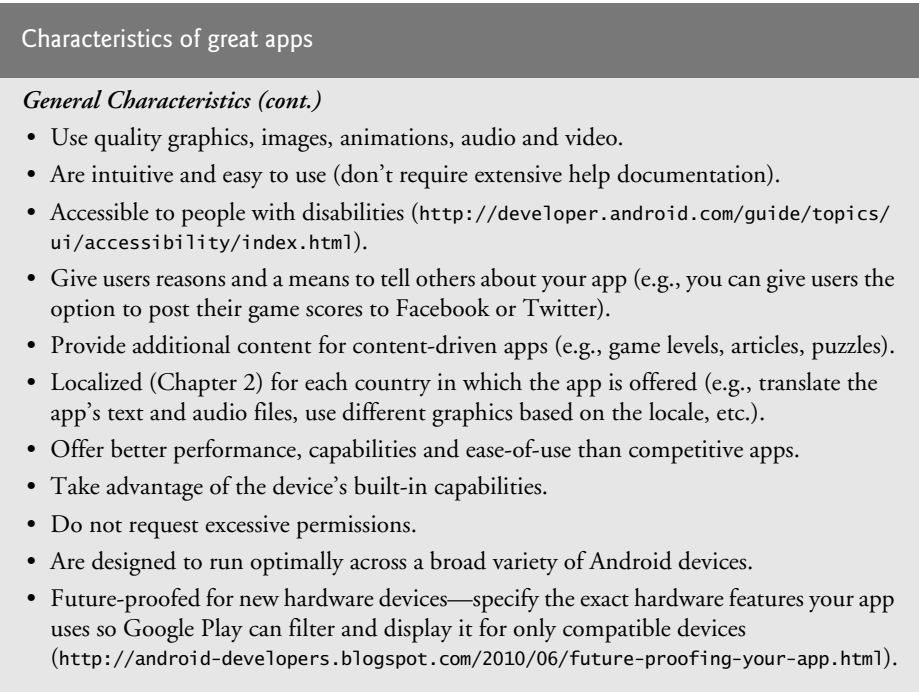


Fig. 1.31 | Characteristics of great apps. (Part 2 of 2.)

1.1.1 Android Development Resources

Figure 1.32 lists some of the key documentation from the Android Developer site. As you dive into Android app development, you may have questions about the tools, design issues, security and more. There are several Android developer newsgroups and forums where you can get the latest announcements or ask questions (Fig. 1.33). Figure 1.34 lists several websites where you'll find Android development tips, videos and resources.

Title	URL
<i>App Components</i>	http://developer.android.com/guide/components/index.html
<i>Using the Android Emulator</i>	http://developer.android.com/tools/devices/emulator.html
<i>Package Index</i>	http://developer.android.com/reference/packages.html
<i>Class Index</i>	http://developer.android.com/reference/classes.html
<i>Android Design</i>	http://developer.android.com/design/index.html

Fig. 1.32 | Key online documentation for Android developers. (Part 1 of 2.)

Title	URL
<i>Data Backup</i>	http://developer.android.com/guide/topics/data/backup.html
<i>Security Tips</i>	http://developer.android.com/training/articles/security-tips.html
<i>Managing Projects from Eclipse with ADT</i>	http://developer.android.com/guide/developing/projects/projects-eclipse.html
<i>Getting Started with Android Studio</i>	http://developer.android.com/sdk/installing/studio.html
<i>Debugging</i>	http://developer.android.com/tools/debugging/index.html
<i>Tools Help</i>	http://developer.android.com/tools/help/index.html
<i>Performance Tips</i>	http://developer.android.com/training/articles/perf-tips.html
<i>Keeping Your App Responsive</i>	http://developer.android.com/training/articles/perf-anr.html
<i>Launch Checklist (for Google Play)</i>	http://developer.android.com/distribute/googleplay/publish/preparing.html
<i>Get Started with Publishing</i>	http://developer.android.com/distribute/googleplay/publish/register.html
<i>Managing Your App's Memory</i>	http://developer.android.com/training/articles/memory.html
<i>Google Play Developer Distribution Agreement</i>	http://play.google.com/about/developer-distribution-agreement.html

Fig. 1.32 | Key online documentation for Android developers. (Part 2 of 2.)

Title	Subscribe	Description
Android Discuss	<i>Subscribe using Google Groups:</i> android-discuss <i>Subscribe via e-mail:</i> android-discuss-subscribe@googlegroups.com	A general Android discussion group where you can get answers to your app-development questions.
Stack Overflow	http://stackoverflow.com/questions/tagged/android	Use this list for beginner-level Android app-development questions, including getting started with Java and Eclipse, and questions about best practices.
Android Developers	http://groups.google.com/forum/?fromgroups#!forum/android-developers	Experienced Android developers use this list for troubleshooting apps, GUI design issues, performance issues and more.

Fig. 1.33 | Android newsgroups and forums. (Part 1 of 2.)

Title	Subscribe	Description
Android Forums	http://www.androidforums.com	Ask questions, share tips with other developers and find forums targeting specific Android devices.

Fig. 1.33 | Android newsgroups and forums. (Part 2 of 2.)

Android development tips, videos and resources	URL
Sample Android apps from Google	http://code.google.com/p/apps-for-android/
O'Reilly article, "Ten Tips for Android Application Development"	http://answers.oreilly.com/topic/862-ten-tips-for-android-application-development/
Bright Hub™ website for Android programming tips and how-to guides	http://www.brighthub.com/mobile/google-android.aspx
The Android Developers Blog	http://android-developers.blogspot.com/
The Sprint® Application Developers Program	http://developer.sprint.com/site/global/develop/mobile_platforms/android/android.jsp
HTC's Developer Center for Android	http://www.htcdev.com/
The Motorola Android development site	http://developer.motorola.com/
Top Android Users on Stack Overflow	http://stackoverflow.com/tags/android/topusers
AndroidDev Weekly Newsletter	http://androiddevweekly.com/
Chet Haase's Codependent blog	http://graphics-geek.blogspot.com/
Cyril Mottier's Android blog	http://cyrilmottier.com/
Romain Guy's Android blog	http://www.curious-creature.org/category/android/
Android Developers Channel on YouTube®	http://www.youtube.com/user/androiddevelopers
Android Video Playlists	http://developer.android.com/develop/index.html
What's New in Android Developer Tools	http://www.youtube.com/watch?v=lmv1dTnhLH4
Google I/O 2013 Developer Conference session videos	http://developers.google.com/events/io/sessions

Fig. 1.34 | Android development tips, videos and resources.