

# Android™

## for Programmers

### An App-Driven Approach



Smartphone and Tablet Apps  
 ADT Plug-In for Eclipse  
 ADT Visual Layout Editor  
 Activities • GUI • Resources  
 Intents • Content Providers  
 Events • Touches • Gestures  
 ActionBar • Fragments  
 Audio • Video • Animation  
 Graphics • OpenGL ES  
 Gallery • Media Library  
 Files • Serialization • SQLite  
 Handlers • Multithreading  
 Camera • Maps • Sensors  
 Location Services • GPS  
 Speech • Web Services  
 Telephony • Bluetooth®  
 App Pricing • Monetization  
 Great App Design  
 App Publishing  
 AppWidgets



**ANDROID™ FOR PROGRAMMERS**  
**AN APP-DRIVEN APPROACH**  
**DEITEL® DEVELOPER SERIES**

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U. S. Corporate and Government Sales  
(800) 382-3419  
corpsales@pearsontechgroup.com

For sales outside the U. S., please contact:

International Sales  
international@pearsoned.com

Visit us on the Web: [informit.com/ph](http://informit.com/ph)

### *Library of Congress Cataloging-in-Publication Data*

On file

© 2012 Pearson Education, Inc.

Portions of the cover are modifications based on work created and shared by Google (<http://code.google.com/policies.html>) and used according to terms described in the Creative Commons 3.0 Attribution License (<http://creativecommons.org/licenses/by/3.0/>).

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13282154-4

ISBN-10: 0-13-282154-0

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.

Second printing, January 2012

---

# ANDROID™ FOR PROGRAMMERS

## AN APP-DRIVEN APPROACH

### DEITEL® DEVELOPER SERIES

Paul Deitel  
Harvey Deitel  
Abbey Deitel  
*Deitel & Associates, Inc.*

Michael Morgano  
*Imerj*



PRENTICE  
HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco  
New York • Toronto • Montreal • London • Munich • Paris • Madrid  
Capetown • Sydney • Tokyo • Singapore • Mexico City



## **Trademarks**

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

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

Microsoft, Internet Explorer and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Google is a trademark of Google, Inc.

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.

*In memory of Daniel McCracken.  
Computer science has lost one of its  
greatest educators.*

*Paul, Harvey, Abbey and Michael*

*This page intentionally left blank*

# Contents

<b>Preface</b>	<b>xiv</b>
<b>Before You Begin</b>	<b>xxii</b>
<b>I Introduction to Android</b>	<b>I</b>
1.1 Introduction	2
1.2 Android Overview	4
1.3 Android 2.2 (Froyo)	7
1.4 Android 2.3 (Gingerbread)	10
1.5 Android 3.0 (Honeycomb)	12
1.6 Android Ice Cream Sandwich	15
1.7 Downloading Apps from the Android Market	16
1.8 Packages	17
1.9 Android Software Development Kit (SDK)	18
1.10 Object Technology: A Quick Refresher	20
1.11 Test-Driving the <b>Doodlz</b> App in an Android Virtual Device (AVD)	23
1.12 Deitel Resources	32
1.13 Android Development Resources	33
1.14 Wrap-Up	34
<b>2 Android Market and App Business Issues</b>	<b>35</b>
2.1 Introduction	36
2.2 Building Great Android Apps	36
2.3 Android Best Practices	38
2.3.1 Compatibility	38
2.3.2 Supporting Multiple Screens	40
2.3.3 Android User Interface Guidelines	40
2.4 Registering at Android Market	44
2.5 Setting Up a Google Checkout Merchant Account	44
2.6 <code>AndroidManifest.xml</code> File	45
2.7 Preparing Your Apps for Publication	46
2.8 Uploading Your Apps to Android Market	51
2.9 Other Android App Marketplaces	54
2.10 Pricing Your App: Free or Fee	54
2.11 Monetizing Apps with In-App Advertising	56
2.12 Monetizing Apps: Using In-App Billing to Sell Virtual Goods in Your Apps	57

2.13	Launching the <b>Market</b> App from Within Your App	59
2.14	Managing Your Apps in Android Market	59
2.15	Marketing Your App	59
2.16	Other Popular App Platforms	64
2.17	Android Developer Documentation	65
2.18	Android Humor	66
2.19	Wrap-Up	67

### **3 Welcome App 68**

#### *Dive-Into® Eclipse and the ADT Plugin*

3.1	Introduction	69
3.2	Technologies Overview	69
3.3	Eclipse IDE	70
3.4	Creating a New Project	71
3.5	Building the Welcome App's GUI with the ADT's Visual Layout Editor	74
3.6	Examining the <code>main.xml</code> File	87
3.7	Running the <b>Welcome</b> App	89
3.8	Wrap-Up	89

### **4 Tip Calculator App 91**

#### *Building an Android App with Java*

4.1	Introduction	92
4.2	Test-Driving the <b>Tip Calculator</b> App	93
4.3	Technologies Overview	94
4.4	Building the App's GUI	94
4.4.1	<code>TableLayout</code> Introduction	94
4.4.2	Creating the Project and Adding the <code>TableLayout</code> and Components	95
4.4.3	Reviewing the Layout So Far	99
4.4.4	Customizing the Components to Complete the Design	99
4.4.5	Final XML Markup for the <b>Tip Calculator</b> GUI	102
4.4.6	<code>strings.xml</code>	106
4.5	Adding Functionality to the App	106
4.6	Wrap-Up	116

### **5 Favorite Twitter® Searches App 117**

#### *SharedPreferences, Buttons, Nested Layouts, Intents, AlertDialogs, Inflating XML Layouts and the Manifest File*

5.1	Introduction	118
5.2	Test-Driving the <b>Favorite Twitter Searches</b> App	119
5.3	Technologies Overview	121
5.4	Building the App's GUI and Resource Files	123
5.4.1	<code>main.xml</code> <code>TableLayout</code>	123
5.4.2	Creating the Project	124
5.4.3	Creating the Resource Files	124

5.4.4	Adding the <code>TableLayout</code> and Components	126
5.4.5	Creating a <code>TableRow</code> That Displays a Search and an <code>Edit</code> Button	130
5.5	Building the App	131
5.6	<code>AndroidManifest.xml</code>	142
5.7	Wrap-Up	143

## 6 Flag Quiz Game App 146

*Assets, AssetManager, Tweened Animations, Handler, Menus and Logging Error Messages*

6.1	Introduction	147
6.2	Test-Driving the <b>Flag Quiz Game</b> App	151
6.3	Technologies Overview	151
6.4	Building the App's GUI and Resource Files	153
6.4.1	<code>main.xml</code> <code>LinearLayout</code>	153
6.4.2	Creating the Project	153
6.4.3	Creating and Editing the Resource Files	154
6.4.4	Adding the Components to the <code>LinearLayout</code>	155
6.4.5	Creating a Button That Can Be Dynamically Inflated	158
6.4.6	Creating the Flag Shake Animation	158
6.5	Building the App	160
6.6	<code>AndroidManifest.xml</code>	174
6.7	Wrap-Up	175

## 7 Cannon Game App 176

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

7.1	Introduction	177
7.2	Test-Driving the <b>Cannon Game</b> App	178
7.3	Technologies Overview	179
7.4	Building the App's GUI and Resource Files	181
7.4.1	Creating the Project	181
7.4.2	<code>AndroidManifest.xml</code>	181
7.4.3	<code>strings.xml</code>	182
7.4.4	<code>main.xml</code>	182
7.4.5	Adding the Sounds to the App	183
7.5	Building the App	183
7.5.1	<code>Line</code> Class Maintains a Line's Endpoints	183
7.5.2	<code>CannonGame</code> Subclass of <code>Activity</code>	183
7.5.3	<code>CannonView</code> Subclass of <code>View</code>	186
7.6	Wrap-Up	203

## 8 SpotOn Game App 204

*Property Animation, ViewPropertyAnimator, AnimatorListener, Thread-Safe Collections, Default SharedPreferences for an Activity*

8.1	Introduction	205
8.2	Test-Driving the SpotOn Game App	206
8.3	Technologies Overview	207
8.4	Building the App's GUI and Resource Files	208
8.4.1	AndroidManifest.xml	208
8.4.2	main.xml RelativeLayout	208
8.4.3	untouched.xml ImageView for an Untouched Spot	209
8.4.4	life.xml ImageView for a Life	209
8.5	Building the App	210
8.5.1	SpotOn Subclass of Activity	210
8.5.2	SpotOnView Subclass of View	212
8.6	Wrap-Up	224

## 9 Doodlz App 225

*Two-Dimensional Graphics, SensorManager, Multitouch Events and Toasts*

9.1	Introduction	226
9.2	Test-Driving the Doodlz App	227
9.3	Technologies Overview	228
9.4	Building the App's GUI and Resource Files	229
9.4.1	Creating the Project	229
9.4.2	AndroidManifest.xml	230
9.4.3	strings.xml	230
9.4.4	main.xml	231
9.4.5	color_dialog.xml	231
9.4.6	width_dialog.xml	233
9.5	Building the App	234
9.5.1	Doodlz Subclass of Activity	234
9.5.2	DoodleView Subclass of View	247
9.6	Wrap-Up	256

## 10 Address Book App 258

*ListActivity, AdapterViews, Adapters, Multiple Activities, SQLite, GUI Styles, Menu Resources and MenuInflater*

10.1	Introduction	259
10.2	Test-Driving the Address Book App	261
10.3	Technologies Overview	262
10.4	Building the GUI and Resource Files	263
10.4.1	Creating the Project	264

10.4.2	AndroidManifest.xml	264
10.4.3	styles.xml	264
10.4.4	textview_border.xml	265
10.4.5	AddressBook Activity's Layout: contact_list_item.xml	266
10.4.6	ViewContact Activity's Layout: view_contact.xml	266
10.4.7	AddEditContact Activity's Layout: add_contact.xml	266
10.4.8	Defining the App's MenuItems with menu Resources in XML	268
10.5	Building the App	269
10.5.1	AddressBook Subclass of ListActivity	269
10.5.2	ViewContact Subclass of Activity	275
10.5.3	AddEditContact Subclass of Activity	281
10.5.4	DatabaseConnector Utility Class	284
10.6	Wrap-Up	290

## **11 Route Tracker App** **291**

*Google Maps API, GPS, LocationManager, MapActivity, MapView and Overlay*

11.1	Introduction	292
11.2	Test-Driving the Route Tracker App	294
11.3	Technologies Overview	296
11.4	Building the GUI and Resource Files	298
11.4.1	Creating the Project	298
11.4.2	AndroidManifest.xml	298
11.4.3	Route Tracker Layout: main.xml	299
11.5	Building the App	300
11.5.1	RouteTracker Subclass of MapActivity	300
11.5.2	BearingFrameLayout Subclass of FrameLayout	311
11.5.3	RouteOverlay Subclass of Overlay	314
11.6	Wrap-Up	318

## **12 Slideshow App** **319**

*Gallery and Media Library Access, Built-In Content Providers, MediaPlayer, Image Transitions, Custom ListActivity Layouts and the View-Holder Pattern*

12.1	Introduction	320
12.2	Test-Driving the Slideshow App	323
12.3	Technologies Overview	324
12.4	Building the GUI and Resource Files	327
12.4.1	Creating the Project	327
12.4.2	Using Standard Android Icons in the App's GUI	327
12.4.3	AndroidManifest.xml	327
12.4.4	Layout for ListView Items in the Slideshow ListActivity	328
12.4.5	Slideshow ListActivity's Menu	328
12.4.6	Layout for the EditText in the Set Slideshow Name Dialog	329

12.4.7	Layout for the SlideshowEditor ListActivity	329
12.4.8	Layout for ListView Items in the SlideshowEditor ListActivity	330
12.4.9	Layout for the SlideshowPlayer Activity	330
12.5	Building the App	331
12.5.1	SlideshowInfo Class	331
12.5.2	Slideshow Subclass of ListActivity	332
12.5.3	SlideshowEditor Subclass of ListActivity	343
12.5.4	SlideshowPlayer Subclass of ListActivity	351
12.6	Wrap-Up	358

## **13**    **Enhanced Slideshow App**    **360**

### *Serializing Data, Taking Pictures with the Camera and Playing Video in a VideoView*

13.1	Introduction	361
13.2	Test-Driving the Enhanced Slideshow App	362
13.3	Technologies Overview	363
13.4	Building the GUI and Resource Files	364
13.4.1	Creating the Project	365
13.4.2	AndroidManifest.xml	365
13.4.3	SlideshowEditor ListActivity's Modified Layout	366
13.4.4	PictureTaker Activity's Layout	366
13.4.5	SlideshowPlayer Activity's Modified Layout	366
13.5	Building the App	367
13.5.1	MediaItem Class	367
13.5.2	SlideshowInfo Class	368
13.5.3	Slideshow Class	370
13.5.4	SlideshowEditor Class	375
13.5.5	PictureTaker Subclass of Activity	378
13.5.6	SlideshowPlayer Class	384
13.6	Wrap-Up	389

## **14**    **Weather Viewer App**    **390**

### *Web Services, JSON, Fragment, ListFragment, DialogFragment, ActionBar, Tabbed Navigation, App Widgets, Broadcast Intents and BroadcastReceiver*

14.1	Introduction	391
14.2	Test-Driving the Weather App	393
14.3	Technologies Overview	394
14.4	Building the App's GUI and Resource Files	396
14.4.1	AndroidManifest.xml	396
14.4.2	WeatherViewerActivity's main.xml Layout	397
14.4.3	Default Cities and ZIP Codes in arrays.xml	398
14.4.4	WeatherViewerActivity's actionmenu.xml Menu Layout	398
14.4.5	WeatherProvider App Widget Configuration and Layout	399

14.5	Building the App	399
14.5.1	Class WeatherViewerActivity	400
14.5.2	Class CitiesFragment	415
14.5.3	Class AddCityDialogFragment	422
14.5.4	Class ForecastFragment	425
14.5.5	Class SingleForecastFragment	425
14.5.6	Class ReadLocationTask	432
14.5.7	Class ReadForecastTask	436
14.5.8	Class FiveDayForecastFragment	442
14.5.9	Class ReadFiveDayForecastTask	447
14.5.10	Class DailyForecast	452
14.5.11	Class WeatherProvider	453
14.6	Wrap-Up	459

## **Index** **460**

## **Chapters on the Web**

*See the Online Chapters section of the Preface for information on downloading these chapters.*

### **15 PHAB's Pizza App**

*Text-to-Speech, Speech-to-Text and Telephony*

### **16 Voice Recorder App**

*Audio Recording and Playback*

### **17 Enhanced Address Book App**

*Bluetooth*

### **18 3D Art App**

*OpenGL ES 3D Rendering*

### **19 HTML5 Favorite Twitter® Searches App**

*Bonus Chapter: HTML5, CSS3 and JavaScript for Experienced Web Developers*

# Preface

---

Welcome to the dynamic world of Android smartphone and tablet app development with the Android Software Development Kit (SDK) 2.3.x and 3.x, the Java™ programming language and the Eclipse™ integrated development environment (IDE).

This book presents leading-edge mobile computing technologies for professional software developers. At the heart of the book is our *app-driven approach*. We present concepts in the context of 17 *complete working Android apps*—16 developed in the native Android environment and one developed in HTML5 for the portable world of the web—rather than using code snippets. Chapters 3–19 each present one app. We begin each of these chapters with an introduction to the app, an app test-drive showing one or more sample executions and a technologies overview. Then we proceed with a detailed code walk-through of the app’s source code. The source code for all the apps is available at [www.deitel.com/books/AndroidFP/](http://www.deitel.com/books/AndroidFP/).

Sales of Android devices and app downloads have been growing exponentially. The first-generation Android phones were released in October 2008. A study by comScore® showed that by July 2011, Android had 41.8% of the U.S. smartphone market share, compared to 27% for Apple’s iPhone and 21.7% for BlackBerry.<sup>1</sup> Billions of apps have been downloaded from Android Market. More than 500,000 Android devices are being activated daily. The opportunities for Android app developers are enormous.

The demand for mobile devices is increasing as more people rely on smartphones and tablets to stay connected and be productive while away from their personal computers. According to comScore, 234 million Americans used mobile devices in a three-month period ending in July 2011. Of those subscribers, 40.6% used apps.<sup>2</sup>

Fierce competition among popular mobile platforms (Android, BlackBerry, iPhone, Palm, Symbian, Windows Phone 7 and others) and among mobile 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. There are now over 300 different Android devices.

*Android for Programmers: An App-Driven Approach* was fun to write! We got to know and love Android, many of its most popular apps and the diversity of Android-based devices. We developed lots of Android apps. The book’s apps were carefully designed to introduce you to a broad range of Android features and technologies, including audio, video, animation, telephony, Bluetooth®, speech recognition, the accelerometer, GPS, the compass, widgets, App Widgets, 3D graphics and more. You’ll quickly learn everything you’ll need to start building Android apps—beginning with a test-drive of the **Doodlz** app

- 
1. [www.comscore.com/Press\\_Events/Press\\_Releases/2011/8/comScore\\_Reports\\_July\\_2011\\_U.S.\\_Mobile\\_Subscriber\\_Market\\_Share](http://www.comscore.com/Press_Events/Press_Releases/2011/8/comScore_Reports_July_2011_U.S._Mobile_Subscriber_Market_Share).
  2. [www.comscore.com/Press\\_Events/Press\\_Releases/2011/8/comScore\\_Reports\\_July\\_2011\\_U.S.\\_Mobile\\_Subscriber\\_Market\\_Share](http://www.comscore.com/Press_Events/Press_Releases/2011/8/comScore_Reports_July_2011_U.S._Mobile_Subscriber_Market_Share).

in Chapter 1, then creating your first app in Chapter 3. Chapter 2, Android Market and App Business Issues walks you through designing great apps, uploading your apps to Google’s Android Market and other online app stores, what to expect in the process, deciding whether to sell your apps or offer them for free, and marketing them using the Internet and word-of-mouth, and more.

## Copyright Notice and Code License

*All of the code and Android apps in the book are copyrighted by Deitel & Associates, Inc. The sample programs in the book are licensed under a Creative Commons Attribution 3.0 Unported License ([creativecommons.org/licenses/by/3.0/](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. 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](mailto:deitel@deitel.com).*

## Intended Audience

We assume that you’re a Java programmer with object-oriented programming experience and that you’re familiar with XML. We use only complete, working apps, so if you don’t know Java and XML but have object-oriented programming experience in C#/.NET, Objective-C/Cocoa or C++ (with class libraries), you should be able to master the material quickly, learning a good amount of Java, Java-style object-oriented programming and XML along the way.

This book is *neither* a Java *nor* an XML tutorial, but it presents a significant amount of Java and XML technology in the context of Android app development. If you’re interested in learning Java, check out our publications:

- *Java for Programmers, 2/e* ([www.deitel.com/books/javafp2/](http://www.deitel.com/books/javafp2/))
- *Java Fundamentals: Parts I and II* LiveLessons videos ([www.deitel.com/books/LiveLessons/](http://www.deitel.com/books/LiveLessons/)).
- *Java How to Program, 9/e* ([www.deitel.com/books/jhttp9/](http://www.deitel.com/books/jhttp9/))

## Key Features

**App-Driven Approach.** Each of the apps chapters (3–19) presents one app—we discuss what the app does, show screen shots of the app in action, test-drive it and overview the technologies and architecture we’ll use to build it. Then we build the app, present the complete code and do a detailed code walkthrough. We discuss the programming concepts and demonstrate the functionality of the Android APIs used in the app. Figure 1 lists the book’s apps and the key technologies we used to build each.

Apps	Technologies
Chapter 3, <b>Welcome</b> App	Dive-Into® Eclipse and the ADT
Chapter 4, <b>Tip Calculator</b> App	Building an Android App with Java

**Fig. 1** | *Android for Programmers* apps and the technologies they introduce.

Apps	Technologies
Chapter 5, <b>Favorite Twitter® Searches</b> App	Collections, Widgets and Views
Chapter 6, <b>Flag Quiz</b> App	Intents and Menus
Chapter 7, <b>Cannon Game</b> App	Frame-By-Frame Animation and Handling User Events
Chapter 8, <b>Spot-On Game</b> App	Tweened Animation and Listening for Touches
Chapter 9, <b>Doodlz</b> App	Graphics and Accelerometer
Chapter 10, <b>Address Book</b> App	AdapterViews and Adapters
Chapter 11, <b>Route Tracker</b> App	Maps API and Compass
Chapter 12, <b>Slideshow</b> App	Photos and Audio Library Access
Chapter 13, <b>Enhanced Slideshow</b> App	Serializing Objects and Playing Video
Chapter 14, <b>Weather Viewer</b> App	Internet Enabled Applications, Web Services and App Widgets
Chapter 15, <b>Pizza Ordering</b> App	Android Telephony and Speech APIs
Chapter 16, <b>Voice Recorder</b> App	Audio Recording and Playback
Chapter 17, <b>Enhanced Address Book</b> App	Managing Persistent Data with SQLite 3 and Transferring Data Via Bluetooth
Chapter 18, <b>3D Art</b> App	3D Graphics and Animation with OpenGL ES
Chapter 19, <b>Favorite Twitter® Searches</b> App using HTML5 Technologies	Online Bonus Chapter: HTML5, CSS3 and JavaScript for Experienced Web Developers

**Fig. 1** | *Android for Programmers* apps and the technologies they introduce.

**Android SDK 2.x.** We cover many of the new features included in the Android Software Development Kit (SDK) 2.x, including Bluetooth, Google Maps, the Camera APIs, graphics APIs and support for multiple screen sizes and resolutions.

**Android SDK 3.x for Tablet Apps.** We cover many of the features of the new Android SDK 3.x for developing tablet apps, including property animation, action bar, fragments, status bar notifications and drag-and-drop.

**Android Maps APIs.** The **Route Tracker** App uses the Android Maps APIs which allow you to incorporate Google™ Maps in your app. Before developing any app using the Maps APIs, you *must* agree to the Android Maps APIs *Terms of Service* (including the related Legal Notices and Privacy Policy) at [code.google.com/android/maps-api-tos.pdf](http://code.google.com/android/maps-api-tos.pdf).

**Eclipse.** The free Eclipse integrated development environment (IDE) combined with the free Android SDK and the free Java Development Kit (JDK), provide everything you need to develop and test Android apps.

**Multimedia.** The apps use a broad range of Android multimedia capabilities, including graphics, images, frame-by-frame animation, property animation, audio, video, speech synthesis and speech recognition.

*Android Best Practices.* We adhere to accepted Android best practices, pointing them out in the detailed code walkthroughs. Check out our Android Best Practices Resource Center at [www.deitel.com/AndroidBestPractices/](http://www.deitel.com/AndroidBestPractices/).

*Web Services.* Web services allow you to use the web as a rich library of services—many of which are free. Chapter 11’s **Route Tracker** app uses the built-in Android Maps APIs to interact with the Google Maps web services. Chapter 14’s **Weather Viewer** app uses WeatherBug’s web services.<sup>3</sup>

## Features

*Syntax Shading.* For readability, we syntax shade the code, similar to Eclipse’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 place defining occurrences of key terms in *bold italic* text for easy reference. We identify on-screen components in the **bold Helvetica** font (e.g., the **File** menu) and Java and Android program text in the **Lucida** font (e.g., `int x = 5;`).

In this book you’ll create GUIs using a combination of visual programming (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 variable name and class name in a **Lucida** font—e.g., “**Button**” or “**myEditText**.”
- 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.deitel.com/books/AndroidFP/
www.informit.com/title/9780132121361

```

*Documentation.* All the Android and Java documentation you’ll need to develop Android apps is available free at [developer.android.com](http://developer.android.com). The documentation for Eclipse is available at [www.eclipse.org/documentation](http://www.eclipse.org/documentation).

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

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

---

3. [apireg.weatherbug.com/defaultAPI.aspx](http://apireg.weatherbug.com/defaultAPI.aspx).

*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 maroon**.

## Online Chapters

Chapter 1–14 are in the print book. Chapters 15–19 will be posted online as we complete them. We'll make draft versions of the chapters available first, and we'll update these drafts to the final versions once we incorporate all of the reviewers' comments. To access the online chapters, go to:

[www.informit.com/register](http://www.informit.com/register)

You must register for an InformIT account and then login. After you've logged into your account, you'll see the **Register a Product** box. Enter the book's ISBN to access the page with the online chapters.

## Slides for Instructors

PDF slides containing all of the code, tables and art in the text are available *to qualified instructors only* through Pearson Education's Instructor Resource Center at:

[www.pearsonhighered.com/irc](http://www.pearsonhighered.com/irc)

## The Deitel Online Android Resource Centers

Our Android Resource Centers include links to tutorials, documentation, software downloads, articles, blogs, podcasts, videos, code samples, books, e-books and more—most of these are free. Check out the growing list of Android-related Resource Centers, including:

- Android ([www.deitel.com/android/](http://www.deitel.com/android/))
- Android Best Practices ([www.deitel.com/androidbestpractices/](http://www.deitel.com/androidbestpractices/))
- Java ([www.deitel.com/java/](http://www.deitel.com/java/))
- Eclipse ([www.deitel.com/Eclipse/](http://www.deitel.com/Eclipse/))
- SQLite 3 ([www.deitel.com/SQLite3/](http://www.deitel.com/SQLite3/))

We announce our latest Resource Centers in our newsletter, the *Deitel<sup>®</sup> Buzz Online* and on Twitter<sup>®</sup> and Facebook<sup>®</sup>—see below.

## Follow Deitel & Associates, Inc. Online

To receive updates on this and other Deitel publications, new and updated apps, Resource Centers, instructor-led onsite training courses, partner offers and more, register for the free *Deitel<sup>®</sup> Buzz Online* e-mail newsletter at:

[www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)

follow us on Twitter

[@deitel](https://twitter.com/deitel)

and Facebook

[www.deitel.com/deitelfan/](http://www.deitel.com/deitelfan/)

## Contacting the Authors

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

`deitel@deitel.com`

We'll respond promptly, and post corrections and clarifications on:

`www.deitel.com/books/AndroidFP/`

and on Facebook and Twitter.

## Acknowledgments

We're fortunate to have worked on this project with the dedicated publishing professionals at Prentice Hall/Pearson. We appreciate the extraordinary efforts and 16-year mentorship of our friend and professional colleague Mark L. Taub, Editor-in-Chief of Pearson Technology Group. Olivia Basegio did a great job recruiting distinguished members of the Android community and managing the review process. Chuti Prasertsith designed the cover with creativity and precision—we gave him our vision for the cover and he made it happen. John Fuller does a superb job managing the production of all of our Deitel Developer Series books.

We'd like to thank our friend, Rich Wong (Partner, Accel Partners), who provided us with valuable contacts in the Android and mobile app development communities.

We'd like to thank AWS Convergence Technologies, Inc., owners of WeatherBug (`weather.weatherbug.com/`), for giving us permission to use their web services in Chapter 14's **Weather Viewer** app.

We'd also like to thank our colleague, Eric Kern, co-author of our related book, *iPhone for Programmers: An App-Driven Approach*, on which many of the apps in *Android for Programmers: An App-Driven Approach* are based.

### Reviewers

We wish to acknowledge the efforts of our reviewers. Adhering to a tight time schedule, the reviewers scrutinized the manuscript, providing constructive suggestions for improving the accuracy and completeness of the presentation:

- Paul Beusterien, Principal, Mobile Developer Solutions
- Eric J. Bowden, COO, Safe Driving Systems, LLC
- Ian G. Clifton, Independent Contractor and Android App Developer
- Daniel Galpin, Android Advocate and author of *Intro to Android Application Development*
- Douglas Jones, Senior Software Engineer, Fullpower Technologies
- Sebastian Nykopp, Chief Architect, Reaktor
- Ronan “Zero” Schwarz, CIO, OpenIntents

Well, there you have it! *Android for Programmers: An App-Driven Approach* will quickly get you developing Android apps. We hope you enjoy reading the book as much as we enjoyed writing it!

*Paul, Harvey and Abbey Deitel, and Michael Morgano, October 2011*

## About the Authors

**Paul J. Deitel**, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT, where he studied Information Technology. Through Deitel & Associates, Inc., he has delivered hundreds of Java, C++, C, C#, Visual Basic and Internet programming courses to industry clients, including Cisco, IBM, Siemens, Sun Microsystems, Dell, Lucent Technologies, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Stratus, Cambridge Technology Partners, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems, 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 and professional book authors.

**Dr. Harvey M. 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 from MIT and a Ph.D. 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., with his son, Paul J. Deitel. He and Paul are the co-authors of dozens of books and LiveLessons video packages and they are writing many more. The Deitels' texts have earned international recognition, with translations published in Japanese, German, Russian, Chinese, Spanish, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of professional programming seminars to major corporations, academic institutions, government organizations and the military.

**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 14 years. She has contributed to numerous Deitel & Associates publications and, together with Paul and Harvey, is the co-author of *iPhone for Programmers: An App-Driven Approach* and *Internet & World Wide Web How to Program, 5/e*.

**Michael Morgano**, Android Developer at Imerj™, is a graduate of Northeastern University where he received a B.S. and M.S. degrees in Computer Science. Michael is the co-author of *iPhone for Programmers: An App-Driven Approach*.

## Corporate Training from Deitel & Associates, Inc.

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

Through its 36-year publishing partnership with Prentice Hall/Pearson, Deitel & Associates, Inc., publishes leading-edge programming professional books, college text-

books, and *LiveLessons* DVD- and web-based video courses. Deitel & Associates, Inc. and the authors can be reached at:

[deitel@deitel.com](mailto:deitel@deitel.com)

To learn more about Deitel's *Dive Into*<sup>®</sup> *Series* Corporate Training curriculum, visit:

[www.deitel.com/training/](http://www.deitel.com/training/)

To request a proposal for on-site, instructor-led training at your company or organization, e-mail [deitel@deitel.com](mailto:deitel@deitel.com).

Individuals wishing to purchase Deitel books and *LiveLessons* DVD- and web-based training courses can do so through [www.deitel.com](http://www.deitel.com). Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, visit [www.pearsoned.com/professional/index.htm](http://www.pearsoned.com/professional/index.htm).

# Before You Begin

This section contains information and instructions you should review to ensure that your computer is set up properly for use with this book. We'll post updates (if any) to the Before You Begin section on the book's website:

[www.deitel.com/books/AndroidFP/](http://www.deitel.com/books/AndroidFP/)

## 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`).

## Software and Hardware System Requirements

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

[developer.android.com/sdk/requirements.html](http://developer.android.com/sdk/requirements.html)

We developed the apps in this book using the following software:

- Java SE 6 Software Development Kit
- Eclipse 3.6.2 (Helios) IDE for Java Developers
- Android SDK versions 2.2, 2.3.3 and 3.x
- ADT (Android Development Tools) Plugin for Eclipse

We tell you where to get each of these in the next section.

## Installing the Java Development Kit (JDK)

Android requires the *Java Development Kit (JDK)* version 5 or 6 (JDK 5 or JDK 6). *We used JDK 6.* To download the JDK for Linux or Windows, go to

[www.oracle.com/technetwork/java/javase/downloads/index.html](http://www.oracle.com/technetwork/java/javase/downloads/index.html)

You need only the JDK. Be sure to follow the installation instructions at

[www.oracle.com/technetwork/java/javase/index-137561.html](http://www.oracle.com/technetwork/java/javase/index-137561.html)

Recent versions of Mac OS X come with Java SE 6. Be sure to get the latest version by using the Apple menu feature to check for software updates.

## Installing the Eclipse IDE

Eclipse is the recommended integrated development environment (IDE) for Android development, though it's possible to use other IDEs, text editors and command-line tools. To download the *Eclipse IDE for Java Developers*, go to

[www.eclipse.org/downloads/](http://www.eclipse.org/downloads/)

This page will allow you to download the latest version of Eclipse—3.7.1 at the time of this writing. To use the same version we used when developing this book (3.6.2), click the **Older Versions** link above the list of downloads. Select the appropriate version for your operating system (Windows, Mac or Linux). To install Eclipse, you simply extract the archive's contents to your hard drive. On our Windows 7 system, we extracted the contents to C:\Eclipse. For more Eclipse installation information, see

[bit.ly/InstallingEclipse](http://bit.ly/InstallingEclipse)

*Important:* To ensure that the book's examples compile correctly, configure Eclipse to use JDK 6 by performing the following steps:

1. Locate the Eclipse folder on your system and double click the Eclipse (🌐) icon to open Eclipse.
2. When the **Workspace Launcher** window appears, click **OK**.
3. Select **Window > Preferences** to display the **Preferences** window.
4. Expand the **Java** node and select the **Compiler** node. Under **JDK Compliance**, set **Compiler compliance level** to 1.6.
5. Close Eclipse.

## Installing the Android SDK

The *Android Software Development Kit (SDK)* provides the tools you need to develop, test and debug Android apps. You can download the Android SDK from

[developer.android.com/sdk/index.html](http://developer.android.com/sdk/index.html)

Click the link for your platform—Windows, Mac OS X or Linux—to download the SDK's archive file. Once you've downloaded the archive, simply extract its contents to a directory of your choice on your computer. The SDK *does not* include the Android platform—you'll download this separately using the tools in the Android SDK.

## Installing the ADT Plugin for Eclipse

The *Android Development Tools (ADT) Plugin* for Eclipse enables you to use the Android SDK tools to develop Android applications in the Eclipse IDE. To install the ADT Plugin, go to

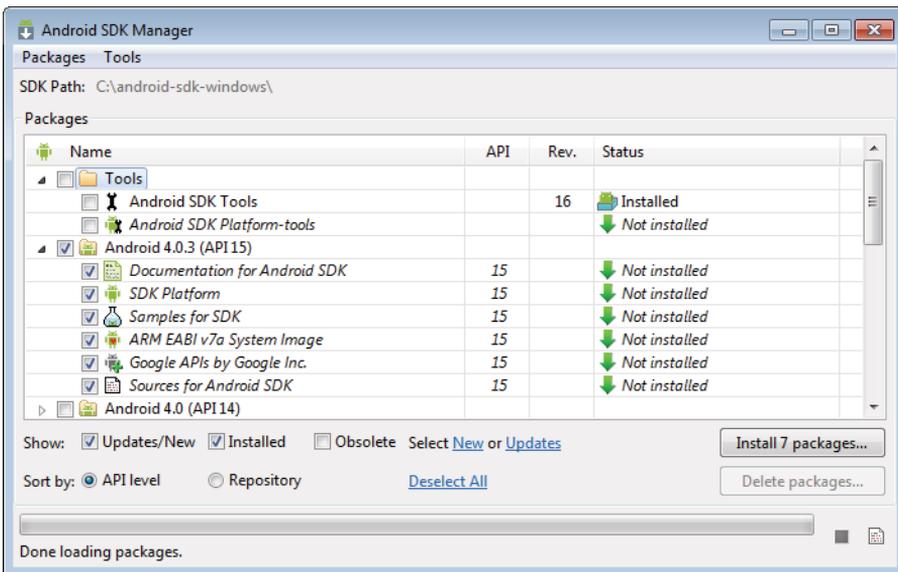
[developer.android.com/sdk/eclipse-adt.html](http://developer.android.com/sdk/eclipse-adt.html)

and *carefully* follow the instructions for downloading and installing the ADT Plugin. If you have any trouble with the installation, be sure to read the troubleshooting tips further down the web page.

## Installing the Android Platform(s)

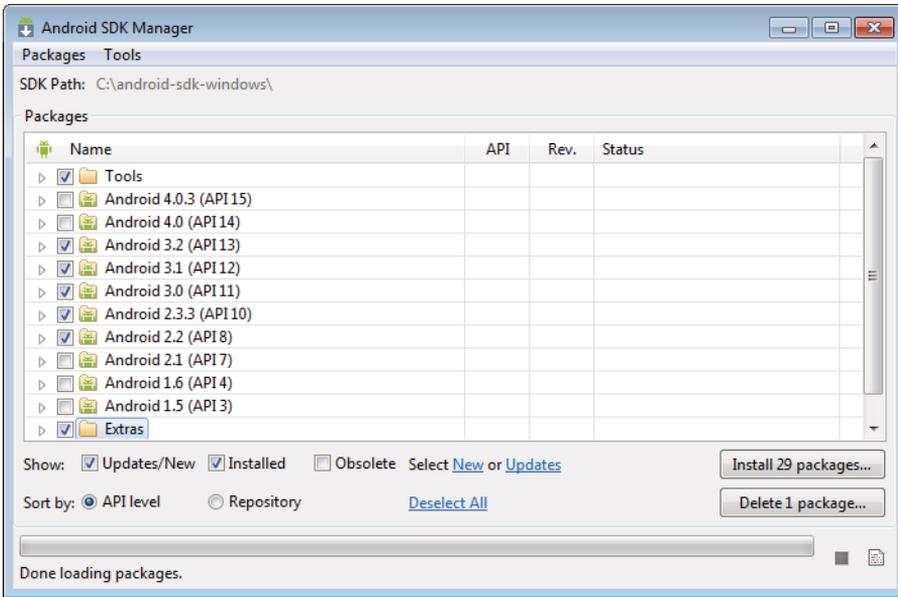
You must now install the Android platform(s) that you wish to use for app development. In this book, we used Android 2.2, 2.3.3 and 3.x. Perform the following steps to install the Android platform(s) and additional SDK tools:

1. Open Eclipse (🌐).
2. When the **Workspace Launcher** window appears, specify where you'd like your apps to be stored, then click **OK**.
3. Select **Window > Preferences** to display the **Preferences** window. In the window, select the **Android** node, then specify the location where you placed the Android SDK on your system in the **SDK Location** field. On our Windows system, we extracted it at `c:\android-sdk-windows`. Click **OK**.
4. Select **Window > Android SDK Manager** to display the **Android SDK Manager** window (Fig. 1).



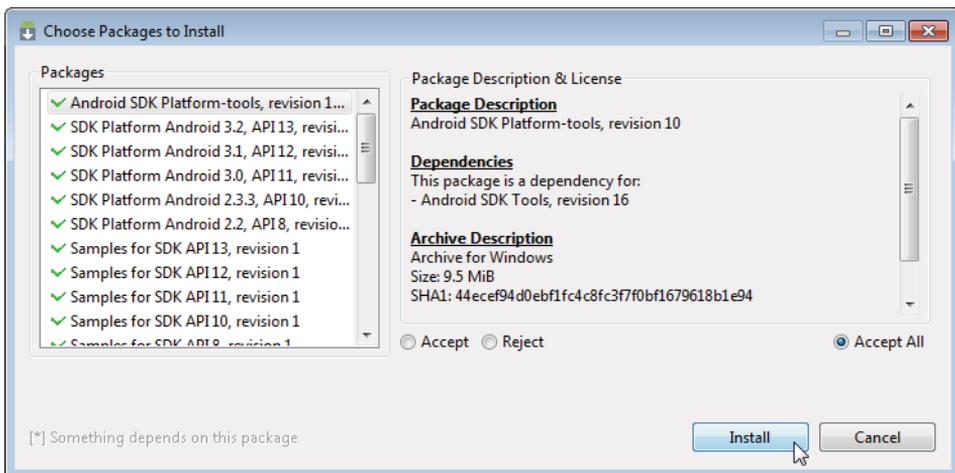
**Fig. 1** | Android SDK Manager window.

5. The **Name** column of the window shows all of the tools, Android platform versions and extras that you can install. For use with this book, you need the items that are checked in Fig. 2. [Note: Most items in the **Extras** node are optional. The **Google USB Driver package** is necessary only for testing Android apps on actual devices using Windows. The **Google Market Licensing package** is necessary only if you intend to develop apps that query the Android Market to determine if a user has a proper license for an app before allowing the app to be used. The **Google Market Billing package** is necessary only if you intend to sell digital content through your app.]



**Fig. 2** | Selecting items to install.

- Click the **Install** button to display the **Choose Packages to Install** window (Fig. 3). In this window, you can read the license agreements for each item. When you're done, click the **Accept All** radio button, then click the **Install** button. The status of the installation process will be displayed in the **Android SDK Manager** window. When the installation is complete, you should close and reopen Eclipse.

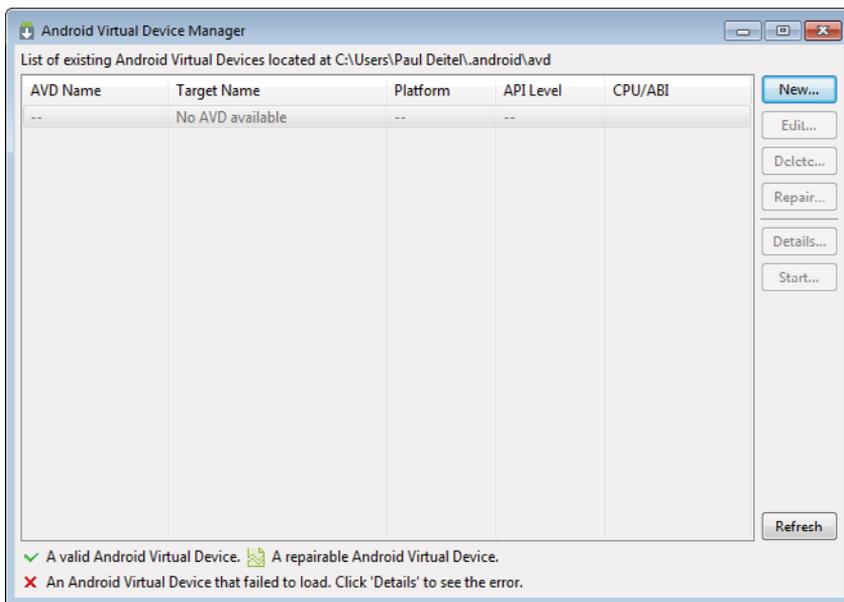


**Fig. 3** | Choose Packages to Install window.

## Creating Android Virtual Devices (AVDs) for Use in the Android Emulator

The *Android emulator*, included in the Android SDK, allows you to run Android apps in a simulated environment on your computer rather than on an actual Android device. Before running an app in the emulator, you must create an *Android Virtual Device (AVD)* which defines the characteristics of the device on which you want to test, 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. If you want to test your apps for multiple Android devices, you can create separate AVDs that emulate each unique device. To do so, perform the following steps:

1. Open Eclipse.
2. Select **Window > AVD Manager** to display the **Android Virtual Device Manager** window (Fig. 4).

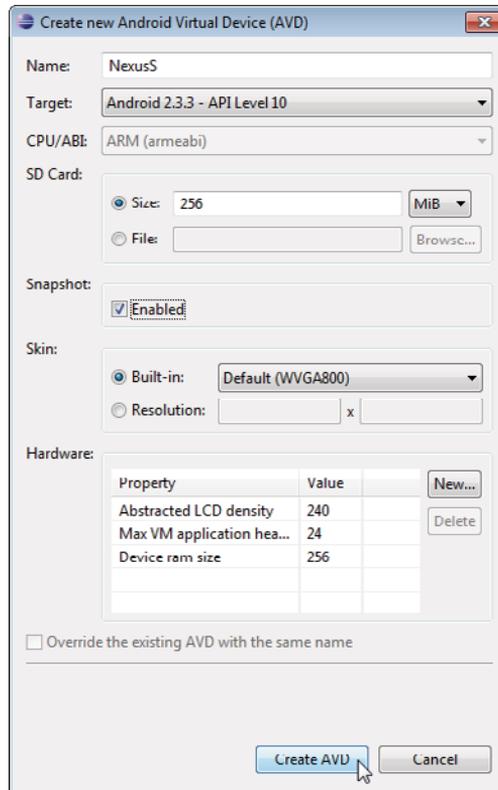


**Fig. 4** | Android AVD Manager window.

3. Click **New...** to display the **Create new Android Virtual Device (AVD)** window (Fig. 5), then configure the options as shown and click **Create AVD**. These settings simulate the primary Android phone that we used for testing—the original Samsung Nexus S, which was running Android 2.3.3 at the time of this writing. Each AVD you create has many other options specified in its `config.ini`. You can modify this file as described at

[developer.android.com/guide/developing/devices/  
managing-avds.html](http://developer.android.com/guide/developing/devices/managing-avds.html)

to more precisely match the hardware configuration of your device.



**Fig. 5** | Create new Android Virtual Device (AVD) window.

4. We also configured an AVD that represents the Motorola Xoom tablet running Android 3.1 so we could test our tablet apps. Its settings are shown in Fig. 6.

### *AVD Performance*

At the time of this writing, AVD performance was quite slow. To improve AVD load time, ensure that the **Enabled** checkbox in the Snapshot section is checked.

## (Optional) Setting Up an Android Device for Development

Eventually, you might want to execute your apps on actual Android devices. To do so, follow the instructions at

[developer.android.com/guide/developing/device.html](http://developer.android.com/guide/developing/device.html)

If you're developing on Microsoft Windows, you'll also need the Windows USB driver for Android devices, which we included as one of the checked items in Fig. 2. In some cases, you may also need device-specific USB drivers. For a list of USB driver sites for various device brands, visit:

[developer.android.com/sdk/oem-usb.html](http://developer.android.com/sdk/oem-usb.html)

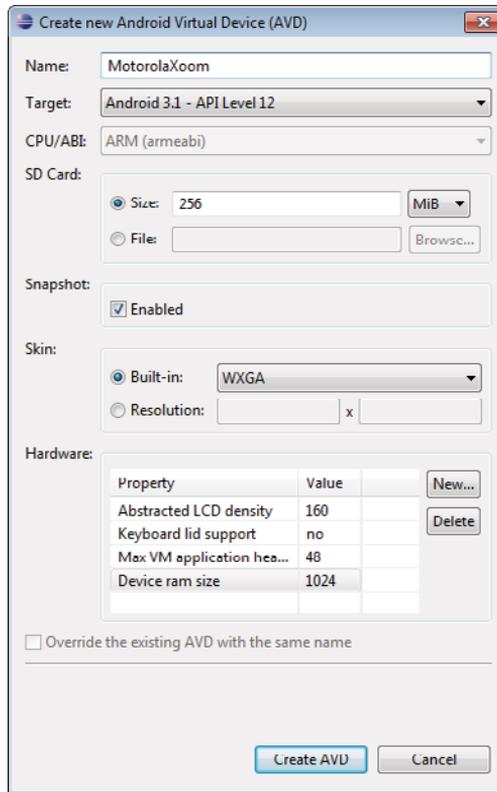


Fig. 6 | Create new Android Virtual Device (AVD) window.

### (Optional) Other IDEs for Developing Android Apps

We developed all the apps in this book using the Eclipse IDE. Though this is the most popular IDE for Android development, there are other IDEs and tools available. Many early Android developers preferred to work with the command-line tools and some phone vendors (such as Motorola) provide their own Android development tools. The site

[developer.android.com/guide/developing/projects/projects-cmdline.html](http://developer.android.com/guide/developing/projects/projects-cmdline.html)

includes information you'd need to develop Android apps using the command-line tools. Some of the tools for command-line development are summarized in (Fig. 7).

Tool	URL	Description
android	<a href="http://developer.android.com/guide/developing/tools/index.html">developer.android.com/guide/developing/tools/index.html</a>	Used to create, view and delete AVDs; create and update Android projects; and update your Android SDK.

Fig. 7 | Tools for developing Android apps in IDEs other than Eclipse.

Tool	URL	Description
Android Emulator	<a href="http://developer.android.com/guide/developing/tools/emulator.html">developer.android.com/guide/developing/tools/emulator.html</a>	Allows you to develop and test Android apps on a computer.
Android Debug Bridge (adb)	<a href="http://developer.android.com/guide/developing/tools/adb.html">developer.android.com/guide/developing/tools/adb.html</a>	Allows you to manage the state of a device or the emulator.
Apache Ant	<a href="http://ant.apache.org/">ant.apache.org/</a>	Application build tool.
Keytool and Jarsigner (or similar signing tool)	<a href="http://developer.android.com/guide/publishing/app-signing.html">developer.android.com/guide/publishing/app-signing.html</a>	Included in the JDK. Keytool generates a private key for digitally signing your Android apps. Jarsigner is used to sign the apps.

**Fig. 7** | Tools for developing Android apps in IDEs other than Eclipse.

## Obtaining the Code Examples

The examples for *Android for Programmers* are available for download at

[www.deitel.com/books/androidFP/](http://www.deitel.com/books/androidFP/)

If you're not already registered at our website, go to [www.deitel.com](http://www.deitel.com) and click the **Register** link below our logo in the upper-left corner of the page. Fill in your information. There's no charge to register, and we do not share your information with anyone. We send you only account-management e-mails unless you register separately for our free, double-opt-in *Deitel® Buzz Online* e-mail newsletter at

[www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)

After registering for our website, you'll receive a confirmation e-mail with your verification code—please verify that you entered your email address correctly. *You'll need to click the verification link in the email to 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.

Next, visit [www.deitel.com](http://www.deitel.com) and sign in using the **Login** link below our logo in the upper-left corner of the page. Go to [www.deitel.com/books/androidFP/](http://www.deitel.com/books/androidFP/). Click the **Examples** link to download the `Examples.zip` file to your computer. Double click `Examples.zip` to unzip the archive.

You're now ready to begin developing Android apps with *Android for Programmers: An App-Driven Approach*. Enjoy!

If you're not already registered at our website, go to [www.deitel.com](http://www.deitel.com) and click the **Register** link below our logo in the upper-left corner of the page. Fill in your information. There's no charge to register, and we do not share your information with anyone. We send you only account-management e-mails unless you register separately for our free, double-opt-in *Deitel® Buzz Online* e-mail newsletter at

[www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)

After registering for our website, you'll receive a confirmation e-mail with your verification code—please verify that you entered your email address correctly. *You'll need the verification code to sign in at www.deitel.com for the first time.* Configure your e-mail client to allow e-mails from [deitel.com](http://deitel.com) to ensure that the verification e-mail is not filtered as junk mail.

Next, visit [www.deitel.com](http://www.deitel.com) and sign in using the **Login** link below our logo in the upper-left corner of the page. Go to [www.deitel.com/books/androidFP/](http://www.deitel.com/books/androidFP/). Click the **Examples** link to download the `Examples.zip` file to your computer. Double click `Examples.zip` to unzip the archive.

You're now ready to begin developing Android apps with *Android for Programmers: An App-Driven Approach*. Enjoy!

# 3

## Welcome App

Dive-Into® Eclipse and the ADT Plugin

### Objectives

In this chapter you'll:

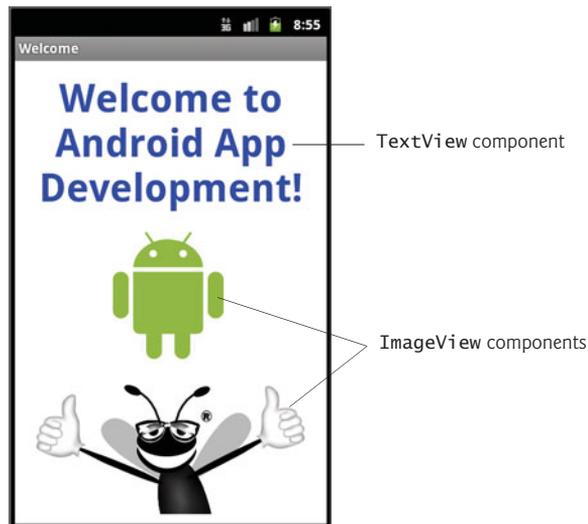
- Learn the basics of the Eclipse IDE for writing, running and debugging your Android apps.
- Create an Eclipse project to develop a new app.
- Design a GUI visually (without programming) using the ADT (Android Development Tools) visual layout editor.
- Edit the properties of GUI components.
- Build a simple Android app and execute it on an Android Virtual Device (AVD).



- 3.1 Introduction
- 3.2 Technologies Overview
- 3.3 Eclipse IDE
- 3.4 Creating a New Project
- 3.5 Building the Welcome App's GUI with the ADT's Visual Layout Editor
- 3.6 Examining the `main.xml` File
- 3.7 Running the **Welcome** App
- 3.8 Wrap-Up

## 3.1 Introduction

In this chapter, you'll build the **Welcome** app—a simple app that displays a welcome message and two images—*without writing any code*. You'll use the Eclipse IDE with the ADT (Android Development Tools) Plugin—the most popular tools for creating and testing Android apps. We'll overview Eclipse and show you how to create a simple Android app (Fig. 3.1) using the ADT's Visual Layout Editor, which allows you to build GUIs using drag-and-drop techniques. Finally, you'll execute your app on an Android Virtual Device (AVD).



**Fig. 3.1** | Welcome app.

## 3.2 Technologies Overview

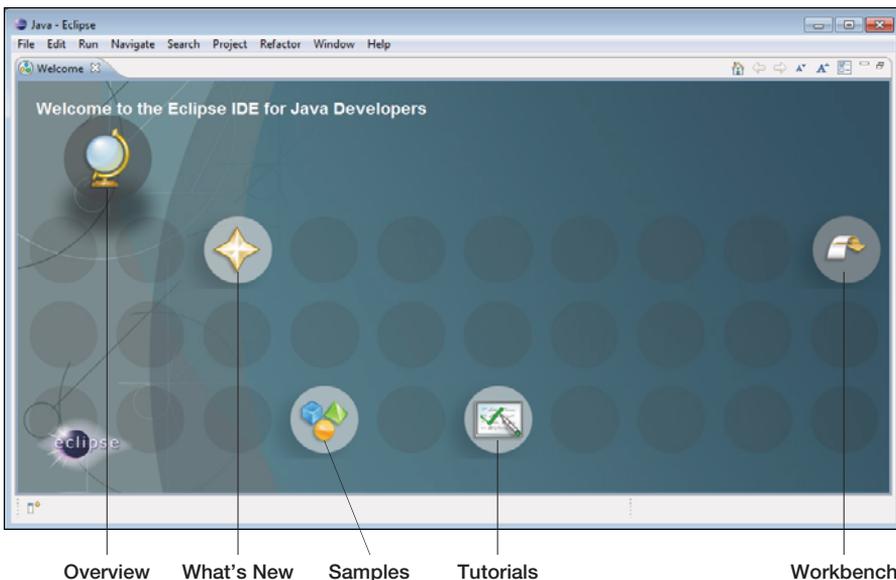
This chapter introduces the Eclipse IDE and ADT Plugin. You'll learn how to navigate Eclipse and create a new project. With the ADT Visual Layout Editor, you'll display pictures in **ImageViews** and display text in a **TextView**. You'll see how to edit GUI component properties (e.g., the `Text` property of a `TextView` and the `Src` property of an `ImageView`) in Eclipse's **Properties** tab and you'll run your app on an Android Virtual Device (AVD).

### 3.3 Eclipse IDE

This book's examples were developed using the versions of the Android SDK that were most current at the time of this writing (versions 2.3.3 and 3.0), and the Eclipse IDE with the ADT (Android Development Tools) Plugin. In this chapter, we assume that you've already set up the Java SE Development Kit (JDK), the Android SDK and the Eclipse IDE, as discussed in the Before You Begin section that follows the Preface.

#### *Introduction to Eclipse*

Eclipse enables you to manage, edit, compile, run and debug applications. The ADT Plugin for Eclipse gives you the additional tools you'll need to develop Android apps. You can also use the ADT Plugin to manage multiple Android platform versions, which is important if you're developing apps for many devices with different Android versions installed. When you start Eclipse for the first time, the **Welcome** tab (Fig. 3.2) is displayed. This contains several icon links, which are described in Fig. 3.3. Click the **Workbench** button to display the Java **development perspective**, in which you can begin developing Android apps. Eclipse supports development in many programming languages. Each set of Eclipse tools you install is represented by a separate development perspective. Changing perspectives reconfigures the IDE to use the tools for the corresponding language.



**Fig. 3.2** | Welcome to the Eclipse IDE for Java Developers tab in the Eclipse window.

Link	Description
Overview	Provides an overview of the IDE and its features.

**Fig. 3.3** | Links on the Eclipse IDE's **Welcome** tab. (Part 1 of 2.)

Link	Description
What's New	Provides information about what's new in the installed version of Eclipse as well as links to the online Eclipse community and updates for the IDE.
Samples	Provides links to samples for the Eclipse configuration you downloaded.
Tutorials	Provides tutorials to help you get started with Java development in Eclipse and to help you use various Eclipse capabilities.
Workbench	Takes you to the development perspective.

**Fig. 3.3** | Links on the Eclipse IDE's **Welcome** tab. (Part 2 of 2.)

### 3.4 Creating a New Project

To begin programming with Android in Eclipse, select **File > New > Project...** to display the **New Project** dialog. Expand the **Android** node, select **Android Project** and click **Next >** to display the **New Android Project dialog** (Fig. 3.4). You can also do this with the **New** () toolbar buttons's drop-down list. After you create your first project, the **Android Project** option will appear in the **File > New** menu and in the **New** () button's drop-down list.

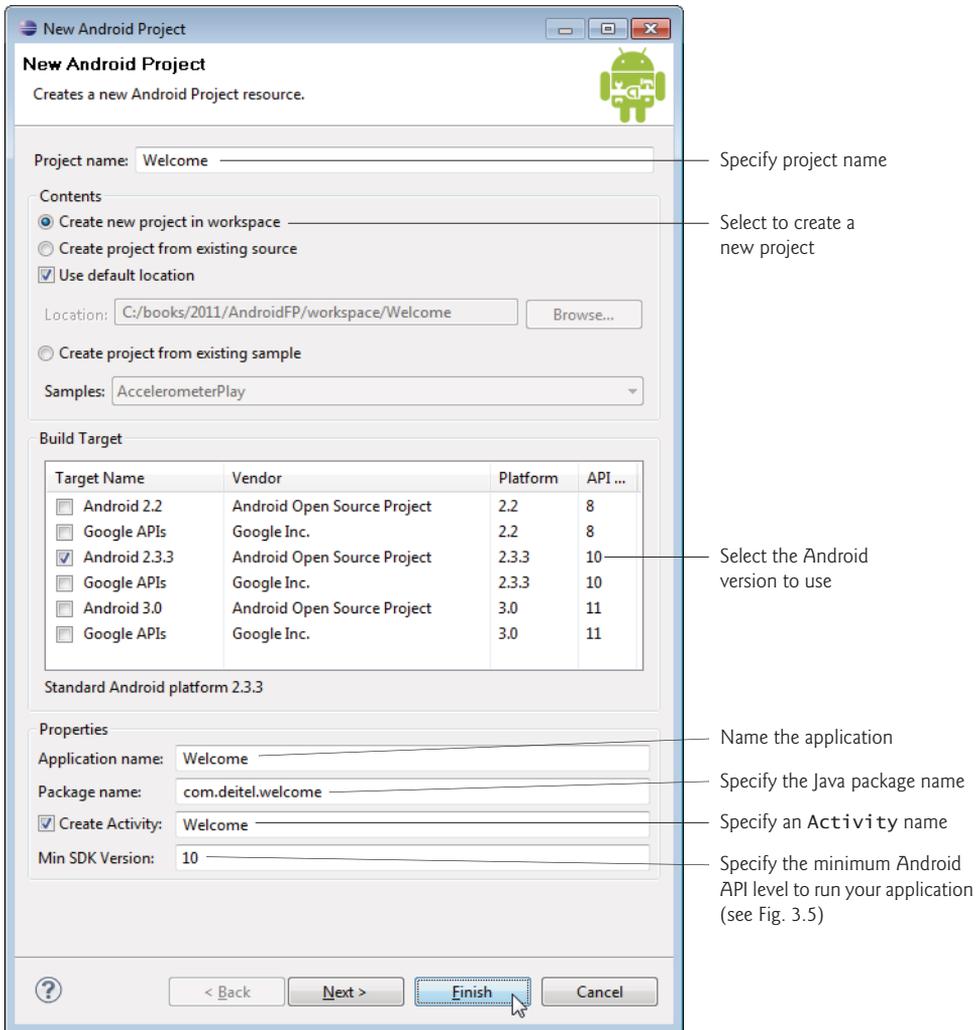
A **project** is a group of related files, such as the code files and any images that make up an app. Using the **New Android Project** dialog, you can create a project from scratch or you can use existing source code—such as the code examples from this book.

In this dialog, specify the following information:

1. In the **Project name:** field, enter `Welcome`. This will be the name of the project's root node in Eclipse's **Package Explorer** tab.
2. In the **Contents** section, ensure that **Create new project in workspace** is selected to create a new project from scratch. The **Create project from existing source** option allows you to create a new project and incorporate existing Java source-code files.
3. In the **Build Target** section, select the Android version you wish to use. For most of this book's examples, we use version 2.3.3; however, it's recommended that you select the minimum version that your app requires so that it can run on the widest variety of devices.

In the **Properties** section of the dialog, specify the following information:

1. In the **Application name:** field, enter `Welcome`. We typically give our applications the same name as their projects, but this is not required. This name appears in a bar at the top of the app, if that bar is not *explicitly* hidden by the app.
2. Android uses conventional Java package-naming conventions and requires a minimum of two parts in the package name (e.g., `com.deitel`). In the **Package name:** field, enter `com.deitel.welcome`. We use our domain `deitel.com` in reverse followed by the app's name. All the classes and interfaces that are created as part of your app will be placed in this Java package. Android and the Android Market use the package name as the app's unique identifier.
3. In the **Create Activity:** field, enter `Welcome`. This will become the name of a class that controls the app's execution. Starting in the next chapter, we'll modify this class to implement an app's functionality.



**Fig. 3.4** | New Android Project dialog.

- In the **Min SDK Version:** field, enter the minimum API level that's required to run your app. This allows your app to execute on devices at that API level and higher. In this book, we typically use the API level 10, which corresponds to Android 2.3.3, or API level 11, which corresponds to Android 3.0. To run your app on Android 2.2 and higher, select API level 8. *In this case, you must ensure that your app does not use features that are specific to more recent versions of Android.* Figure 3.5 shows the Android SDK versions and API levels. *Other versions of the SDK are now deprecated and should not be used.* The following webpage shows the current percentage of Android devices running each platform version:

[developer.android.com/resources/dashboard/platform-versions.html](http://developer.android.com/resources/dashboard/platform-versions.html)

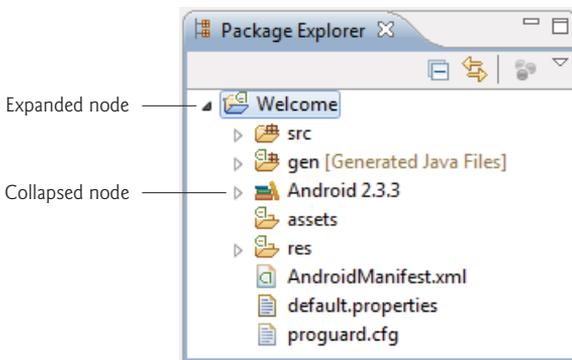
Android SDK version	API level
3.0	11
2.3.3	10
2.2	8
2.1	7
1.6	4
1.5	3

**Fig. 3.5** | Android SDK versions and API levels.  
([developer.android.com/sdk/index.html](http://developer.android.com/sdk/index.html))

5. Click **Finish** to create the project. [*Note:* You might see project errors while Eclipse loads the Android SDK.]

### *Package Explorer Window*

Once you create (or open) a project, the **Package Explorer** window at the left of the IDE provides access to all of the project's files. Figure 3.6 shows the project contents for the **Welcome** app. The **Welcome** node represents the project. You can have many projects open in the IDE at once—each will have its own top-level node.



**Fig. 3.6** | Package Explorer window.

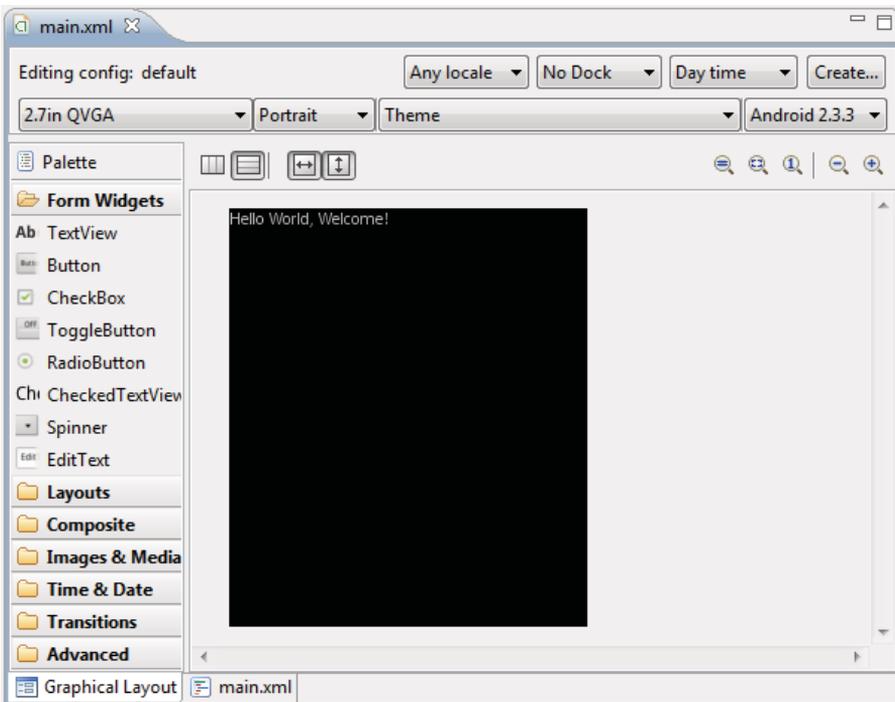
Within a project's node the project's contents are organized into various files and folders, including:

- **src**—A folder containing the project's Java source files.
- **gen**—A folder containing the Java files generated by the IDE.
- **Android 2.3.3**—A folder containing the Android framework version you selected when you created the app.
- **res**—A folder containing the **resource files** associated with your app, such as GUI layouts and images used in your app.

We discuss the other files and folders as necessary throughout the book.

### 3.5 Building the Welcome App's GUI with the ADT's Visual Layout Editor

Next, you'll create the GUI for the **Welcome** app. The ADT's **Visual Layout Editor** allows you to build your GUI by dragging and dropping GUI components, such as Buttons, TextViews, ImageViews and more, onto an app. For an Android app that you create with Eclipse, the *GUI layout is stored in an XML file called `main.xml`*, by default. Defining the GUI in XML allows you to easily separate your app's logic from its presentation. Layout files are considered app *resources* and are stored in the project's **res** folder. GUI layouts are placed within that folder's `layout` subfolder. When you double click the `main.xml` file in your app's `/res/layout` folder, the Visual Layout Editor view is displayed by default (Fig. 3.7). To view the XML contents of the file (Fig. 3.8), click the tab with the name of the layout file (`main.xml` in this case). You can switch back to the Visual Layout Editor by clicking the **Graphical Layout** tab. We'll present the layout's XML in Section 3.6.



**Fig. 3.7** | Visual Layout Editor view of the app's default GUI.

#### *The Default GUI*

The default GUI for a new Android app consists of a `LinearLayout` with a black background and contains a `TextView` with the text "Hello World, Welcome!" (Fig. 3.7). A **LinearLayout** arranges GUI components in a line horizontally or vertically. A **TextView** allows you to display text. If you were to execute this app in an AVD or on a device, you'd see the default black background and text.



**Fig. 3.8** | XML view of the app's default GUI.

Figure 3.9 lists some of the layouts from the **android.widget** package.<sup>1</sup> We'll cover many more GUI components that can be placed in layouts—for a complete list, visit:

[developer.android.com/reference/android/widget/package-summary.html](http://developer.android.com/reference/android/widget/package-summary.html)



### Look-and-Feel Observation 3.1

*To support devices of varying screen sizes and densities, it's recommended that you use `RelativeLayout` and `TableLayout` in your GUI designs.*

Layout	Description
FrameLayout	Allocates space for a single component. You can add more than one component to this layout, but each will be displayed from the layout's upper-left corner. The last component added will appear on top.
LinearLayout	Arranges components horizontally in one row or vertically in one column.
RelativeLayout	Arranges components relative to one another or relative to their parent container.
TableLayout	Arranges components into a table of rows. You can then use the <code>TableRow</code> layout (a subclass of <code>LinearLayout</code> ) to organize the columns.

**Fig. 3.9** | Android layouts (package `android.widget`).

### Configuring the Visual Layout Editor to use the Appropriate Android SDK

If you've installed multiple Android SDKs, the ADT Plugin selects the most recent one as the default for design purposes in the **Graphical Layout** tab—regardless of the SDK you selected when you created the project. In Fig. 3.7, we selected Android 2.3.3 from the

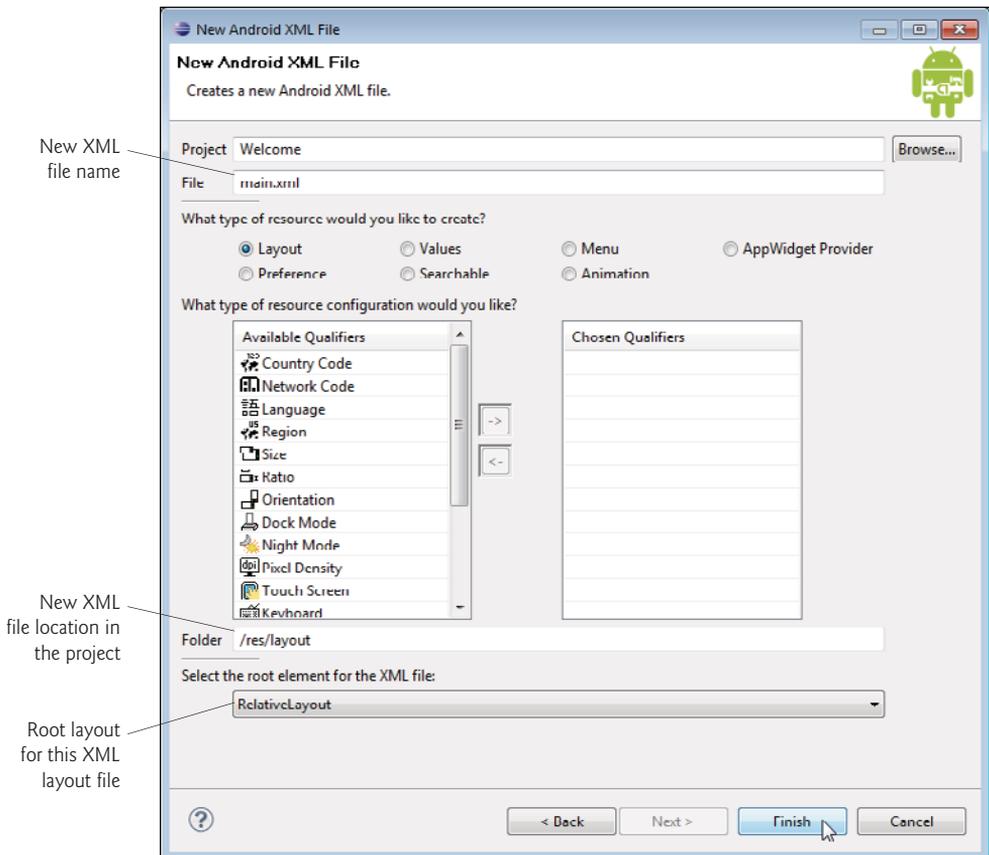
1. Earlier Android SDKs also have an `AbsoluteLayout` in which each component specifies its exact position. This layout is now deprecated. According to [developer.android.com/reference/android/widget/AbsoluteLayout.html](http://developer.android.com/reference/android/widget/AbsoluteLayout.html), you should use `FrameLayout`, `RelativeLayout` or a custom layout instead.

SDK selector drop-down list at the top-right side of the **Graphic Layout** tab to indicate that we're designing a GUI for an Android 2.3.3 device.

### *Deleting and Recreating the main.xml File*

For this application, you'll replace the default `main.xml` file with a new one that uses a `RelativeLayout`, in which components are arranged relative to one another. Perform the following steps to replace the default `main.xml` file:

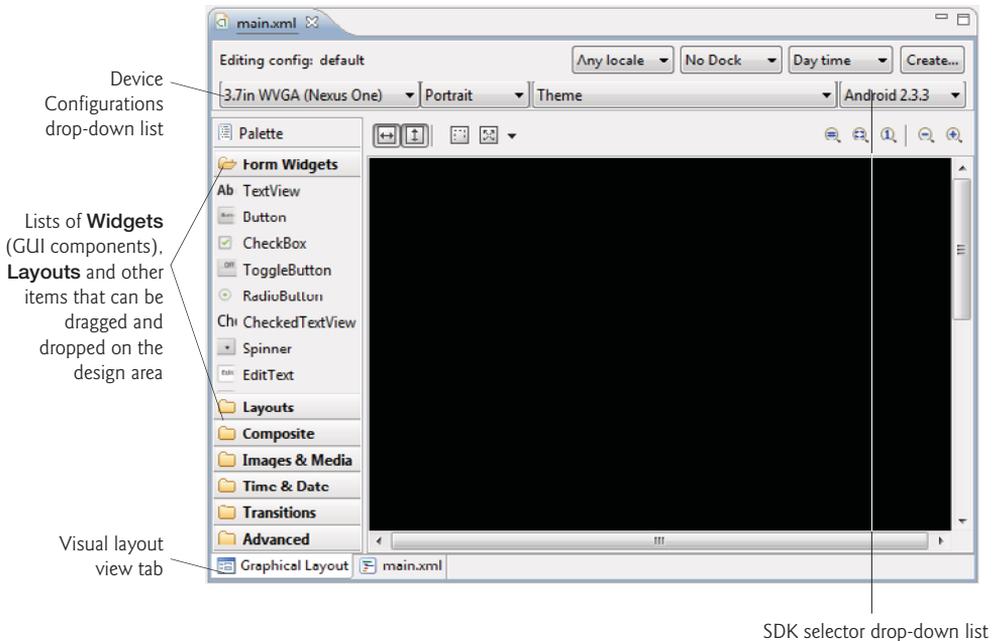
1. Make sure `main.xml` is closed, then right click it in the project's `/res/layout` folder and select **Delete** to delete the file.
2. Right click the layout folder and select **New > Other...** to display the **New** dialog.
3. In the **Android** node, select **Android XML File** and click **Next >** to display the **New Android XML File** dialog.
4. Configure the file name, location and root layout for the new `main.xml` file as shown in Fig. 3.10, then click **Finish**.



**Fig. 3.10** | Creating a new `main.xml` file in the **New Android XML File** dialog.

### Configuring the Visual Layout Editor's Size and Resolution

Figure 3.11 shows the new `main.xml` file in the Visual Layout Editor. Android runs on a wide variety of devices, so the Visual Layout Editor comes with several device configurations that represent various screen sizes and resolutions. These can be selected from the Device Configurations drop-down list at the top-left side of the **Graphic Layout** tab (Fig. 3.11). If these predefined configurations do not match the device you wish to target, you can create your own device configurations from scratch, or by copying and modifying the existing ones.



**Fig. 3.11** | Visual Layout Editor view of the app's default GUI.

Our primary testing device for this book was the Samsung Nexus S, which has a 4-inch screen with 480-by-800 (WVGA) resolution. When designing an Android GUI, you typically want it to be *scalable* so that it displays properly on various devices. For this reason, the Visual Layout Editor's design area does not need to precisely match your actual device's. Instead, you can choose a similar device configuration. In Fig. 3.11, we selected the **3.7in WVGA (Nexus One)** option—this device has the same WVGA resolution as the Nexus S, but a slightly smaller screen size. Many of today's smartphones have 480-by-800 or 480-by-854 resolution.

### Images and Screen Sizes/Resolutions

Because Android devices have various screen sizes, resolutions and pixel densities (that is, dots per inch or DPI), Android allows you to provide separate images (and other resources) that the operating system chooses based on the actual device's pixel density. For this reason your project's `res` folder contains three subfolders for images—`drawable-hdpi` (high den-

sity), `drawable-mdpi` (medium density) and `drawable-ldpi` (low density). These folders store images with different pixel densities (Fig. 3.12).

Density	Description
ldpi	Low density—approximately 120 dots-per-inch.
mdpi	Medium density—approximately 160 dots-per-inch.
hdpi	High density—approximately 240 dots-per-inch.
xhdpi	Extra high density—approximately 320 dots-per-inch.
nodpi	Indicates that a resource should not be scaled regardless of screen density.

**Fig. 3.12** | Android pixel densities.

Images for devices that are similar in pixel density to our testing device are placed in the folder `drawable-hdpi`. Images for medium- and low-density screens are placed in the folders `drawable-mdpi` and `drawable-ldpi`, respectively. As of Android 2.2, you can also add a `drawable-xhdpi` subfolder to the app's `res` folder to represent screens with extra high pixel densities. Android will scale images up and down to different densities as necessary.



### Look-and-Feel Observation 3.2

*For detailed information on supporting multiple screens and screen sizes in Android, visit [developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html).*



### Look-and-Feel Observation 3.3

*For images to render nicely, a high-pixel-density device needs higher-resolution images than a low-pixel-density device. Low-resolution images do not scale well.*

## Step 1: Adding Images to the Project

You'll now begin designing the **Welcome** app. In this chapter, we'll use the Visual Layout Editor and the **Outline** window to build the app, then we'll explain the generated XML in detail. In subsequent chapters, we'll also edit the XML directly.



### Look-and-Feel Observation 3.4

*Many Android professionals prefer to create their GUIs directly in XML and use the Visual Layout Editor to preview the results. As you type in the XML view, Eclipse provides auto-complete capabilities showing you component names, attribute names and values that match what you've typed so far. These help you write the XML quickly and correctly.*

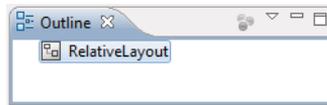
For this app, you'll need to add the Deitel bug image (`bug.png`) and the Android logo image (`android.png`) to the project—we've provided these in the `images` folder with the book's examples. Perform the following steps to add the images to this project:

1. In the **Package Explorer** window, expand the project's `res` folder.
2. Locate and open the `images` folder provided with the book's examples, then drag the images in the folder onto the `res` folder's `drawable-hdpi` subfolder.

These images can now be used in the app.

**Step 2: Changing the Id Property of the RelativeLayout**

You can use the **Properties** window to configure the properties of the selected layout or component without editing the XML directly. If the **Properties** window is not displayed, you can display it by double clicking the RelativeLayout in the **Outline** window. You can also select **Window > Show View > Other...**, then select **Properties** from the **General** node in the **Show View** dialog. To select a layout or component, you can either click it in the Visual Layout Editor or select its node in the **Outline** window (Fig. 3.13). The **Properties** window cannot be used when the layout is displayed in XML view.



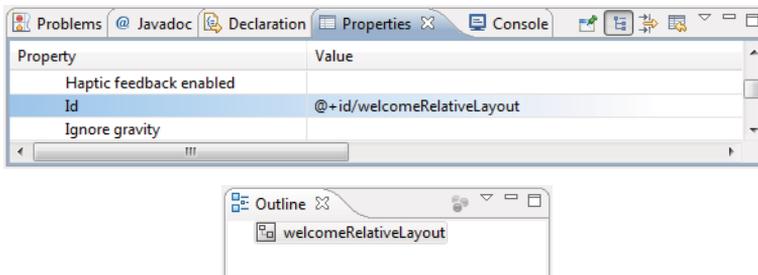
**Fig. 3.13** | Hierarchical GUI view in the **Outline** window.

You should rename each layout and component with a relevant name, especially if the the layout or component will be manipulated programmatically (as we'll do in later apps). Each object's name is specified via its **Id property**. The Id can be used to access and modify component without knowing its exact location in the XML. As you'll see shortly, the **id** can also be used to specify the relative positioning of components in a RelativeLayout.

Select the RelativeLayout, then scroll to the **Id property** in the **Properties** window and set its value to

```
@+id/welcomeRelativeLayout
```

The + in the syntax @+id indicates that a new id (that is, a variable name) should be created with the identifier to the right of the /. The **Properties** and **Outline** windows should now appear as in Fig. 3.14.



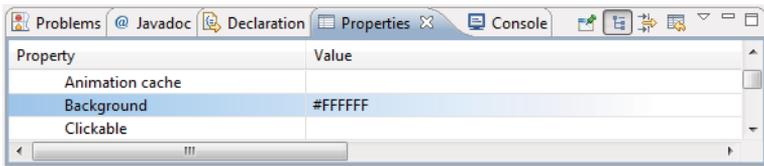
**Fig. 3.14** | **Properties** window after changing the RelativeLayout's Id property.

**Step 3: Changing the Background Property of the RelativeLayout**

The layout's default background color is black, but we'd like it to be white. Every color can be created from a combination of red, green and blue components called **RGB values**—each is an integer in the range 0–255. The first value defines the amount of red in the color, the second the amount of green and the third the amount of blue. When using

the IDE to specify a color you typically use hexadecimal format. In this case, the RGB components are represented as values in the range 00–FF.

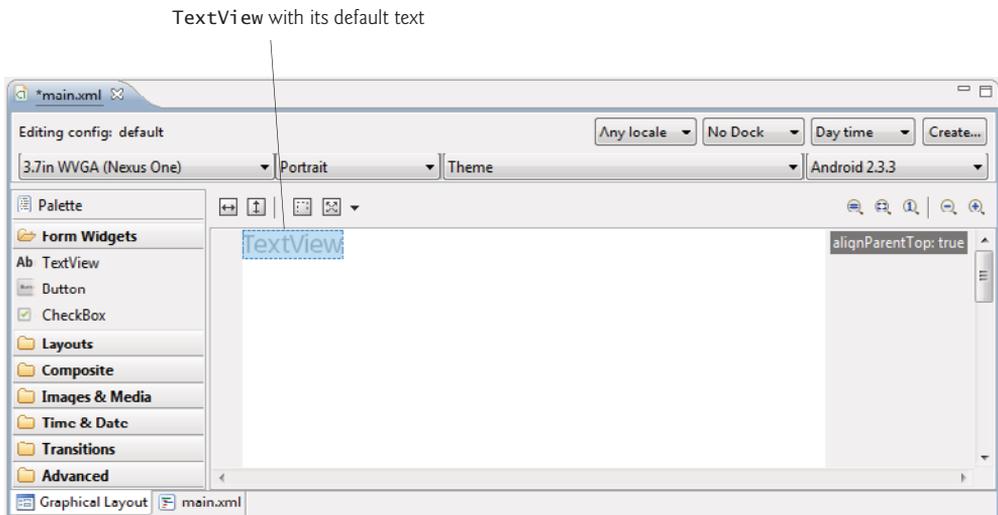
To change the background color, locate the **Background property** in the **Properties** window and set its value to #FFFFFF (Fig. 3.15). This represents white in the hexadecimal format #RRGGBB—the pairs of hexadecimal digits represent the red, green and blue color components, respectively. Android also supports alpha (transparency) values in the range 0–255, where 0 represents completely transparent and 255 represents completely opaque. If you wish to use alpha values, you can specify the color in the format #AARRGGBB, where the first two hexadecimal digits represent the alpha value. For cases in which both digits of each component of the color are the same, you can use the formats #RGB or #ARGB. For example, #FFF will be treated as #FFFFFF.



**Fig. 3.15** | Properties window after changing the RelativeLayout’s Background property.

#### Step 4: Adding a TextView

Next, we’ll add a TextView to the user interface. In the **Form Widgets** list at the left of the Visual Layout Editor window, locate TextView and drag it onto the design area (Fig. 3.16). When you add a new component to the user interface, it’s automatically selected and its properties are displayed in the **Properties** window.



**Fig. 3.16** | TextView with its default text.

**Step 5: Configuring the TextView's Text Property Using a String Resource**

According to the Android documentation for application resources

[developer.android.com/guide/topics/resources/index.html](http://developer.android.com/guide/topics/resources/index.html)

it's considered a good practice to “externalize” strings, string arrays, images, colors, font sizes, dimensions and other app resources so that you, or someone else on your team, can manage them separately from your application's code. For example, if you externalize color values, all components that use the same color can be updated to a new color simply by changing the color value in a central resource file.

If you wish to localize your app in several different languages, storing the strings separately from the app's code allows you to change them easily. In your project's `res` folder, the subfolder `values` contains a `strings.xml` file that's used to store strings. To provide localized strings for other languages, you can create separate `values` folders for each language. For example, the folder `values-fr` would contain a `strings.xml` file for French and `values-es` would contain a `strings.xml` file for Spanish. You can also name these folders with region information. For example, `values-en-rUS` would contain a `strings.xml` file for U.S. English and `values-en-rGB` would contain a `strings.xml` file for United Kingdom English. For more information on localization, see

[developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources](http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources)  
[developer.android.com/guide/topics/resources/localization.html](http://developer.android.com/guide/topics/resources/localization.html)

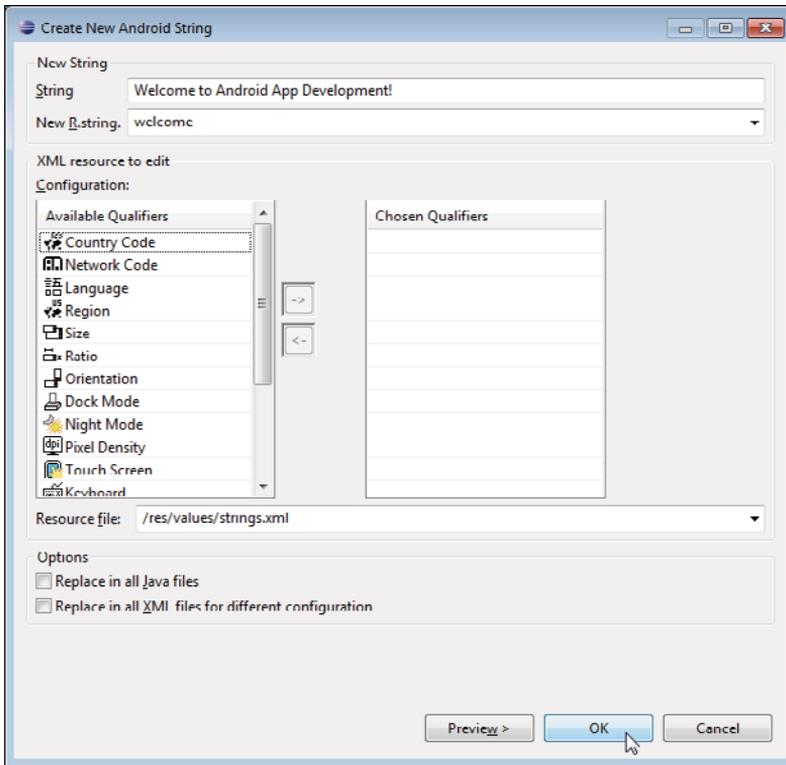
To set the `TextView`'s **Text** property, we'll create a new string resource in the `strings.xml` file.

1. Ensure that the `TextView` is selected.
2. Locate its **Text** property in the **Properties** window, click its default value, then click the ellipsis button () at the right side of the property's value field to display the **Resource Chooser** dialog.
3. In the **Resource Chooser** dialog, click the **New String...** button to display the **Create New Android String** dialog (Fig. 3.17).
4. Fill the **String** and **New R.string** fields as shown in Fig. 3.17, then click **OK** to dismiss the **Create New Android String** dialog and return to the **Resource Chooser** dialog.
5. The new string resource named `welcome` is automatically selected. Click **OK** to select this resource.

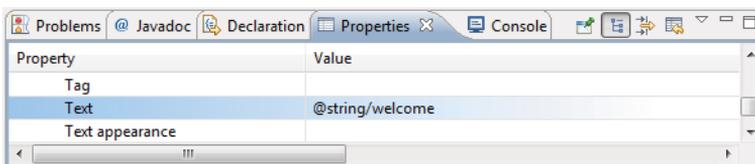
In the **Properties** window, the **Text** property should now appear as shown in Fig. 3.18. The syntax `@string` indicates that an existing string resource will be selected from the `strings.xml` file, and the name `welcome` indicates which string resource to select.

A key benefit of defining your string values this way is that you can easily *localize* your app by creating additional XML resource files for string resources in other languages. In each file, you use the same name in the **New R.string** field and provide the internationalized string in the **String** field. Android can then choose the appropriate resource file based on the device user's preferred language. For more information on localization, visit

[developer.android.com/guide/topics/resources/localization.html](http://developer.android.com/guide/topics/resources/localization.html)



**Fig. 3.17** | Create New Android String window.



**Fig. 3.18** | Properties window after changing the TextView's Text property.

### *Step 6: Configuring the TextView's Text size and Padding top Properties—Scaled Pixels and Density-Independent Pixels*

The sizes of GUI components and text in Android can be specified in several different units (Fig. 3.19). The documentation for supporting multiple screen sizes

[developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html)

recommends that you use density-independent pixels for the dimensions of GUI components and other screen elements and scale-independent pixels for font sizes.

Defining your GUIs with **density-independent pixels** enables the Android platform to automatically scale the GUI, based on the pixel density of the actual device's screen.

Unit	Description
px	pixel
dp or dip	density-independent pixel
sp	scale-independent pixel
in	inches
mm	millimeters

**Fig. 3.19** | Measurement units.

One density-independent pixel is equivalent to one pixel on a screen with 160 dpi (dots per inch). On a screen with 240 dpi, each density-independent pixel will be scaled by a factor of 240/160 (i.e., 1.5). So, a component that's 100 density-independent pixels wide will be scaled to 150 actual pixels wide. On a screen with 120 dpi, each density-independent pixel is scaled by a factor of 120/160 (i.e., .75). So, the same component that's 100 density-independent pixels wide will be 75 actual pixels wide. *Scale-independent pixels* are scaled like density-independent pixels, and they're also scaled by the user's preferred font size specified on the device. [Note: At the time of this writing, users cannot yet change the preferred font size on Android devices, but this feature is expected in the future.]

You'll now increase the size of the `TextView`'s font and add some padding above the `TextView` to separate the text from the edge of the device's screen.

1. To change the font size, ensure that the `TextView` is selected, then change its **Text size property** to 40sp.
2. To add some space between the top edge of the layout and the `TextView`, set the **Layout margin top property** in the **Misc** section of the **Properties** window to 10dp.

#### *Step 7: Configuring Additional TextView Properties*

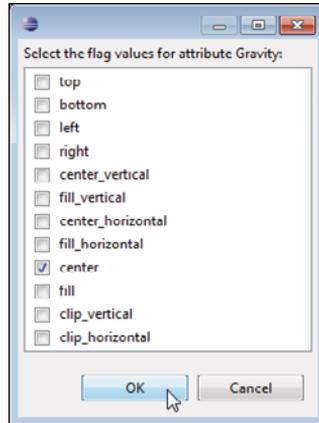
Configure the following additional `TextView`'s properties as well:

1. Set its **Id property** to `@+id/welcomeTextView`.
2. Set its **Text color property** to #00F (blue).
3. Set its **Text style property** to `bold`. To do so, click the **Value** field for this property, then click the ellipsis button (`...`) to display the dialog for selecting the font style. Click the **bold** checkbox, then click **OK** to set the text style.
4. To center the text in the `TextView` if it wraps to multiple lines, set its **Gravity property** to `center`. To do so, click the **Value** field for this property, then click the ellipsis button to display a dialog with the **Gravity** property's options (Fig. 3.20). Click the **center** checkbox, then click **OK** to set the value.

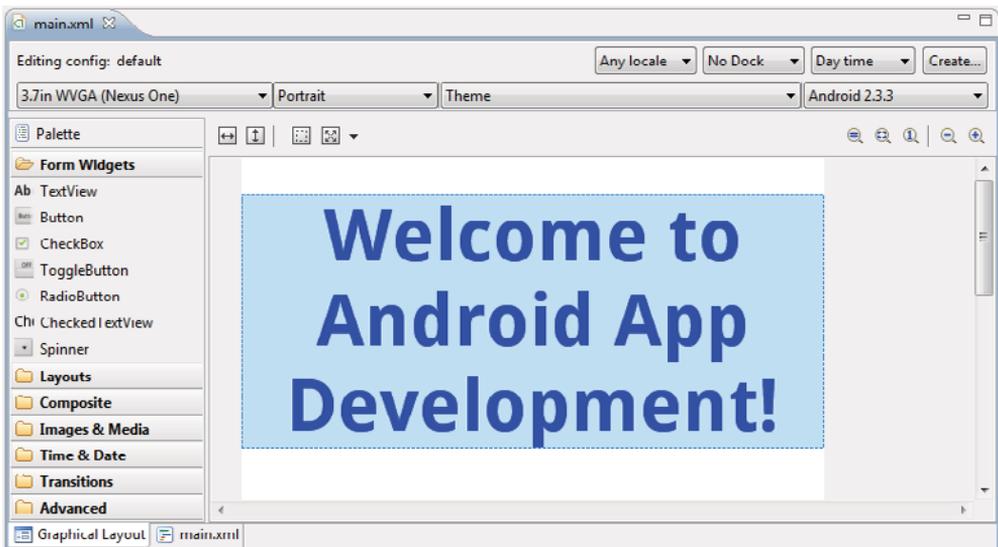
The Visual Layout Editor window should now appear as shown in Fig. 3.21.

#### *Step 8: Adding ImageViews to Display the Android Logo and the Deitel Bug Logo*

Next, you'll add two `ImageViews` to the GUI to display the images that you added to the project in *Step 1*. When you first drag an `ImageView` onto the Visual Layout Editor, nothing appears. For this reason, we'll use the **Outline** window to add the `ImageViews`. Perform the following steps:



**Fig. 3.20** | Options for the gravity attribute of an object.



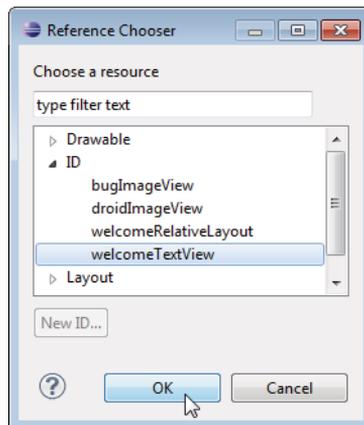
**Fig. 3.21** | Visual Layout Editor window after completing the TextView's configuration.

1. Drag an `ImageView` from the **Images & Media** category in the Visual Layout Editor's **Palette** and drop it onto the **Outline** window as shown in Fig. 3.22. The new `ImageView` appears below the `welcomeTextView` node. This *does not* indicate that this component will appear below the `TextView` in the GUI. This requires setting the **Layout below** property, which we'll do in a moment. [Note: If you drag the `ImageView` over the `welcomeTextView` and hover for a moment, a green rectangle with sections will appear around the `welcomeTextView`. If you then drag the `ImageView` over one of those sections and drop it, the Visual Layout Editor can set the relative positioning for you.]



**Fig. 3.22** | Dragging and dropping an `ImageView` onto the **Outline** window.

2. Set the `ImageView`'s **Id** property to `@+id/droidImageView`. The **Outline** window now shows the object's name as `droidImageView`.
3. Set the `droidImageView`'s **Layout below** property to `@id/welcomeTextView` to position the `ImageView` below the `welcomeTextView`. To do so, click the **Value** field for this property, then click the ellipsis button to display the **Reference Chooser** dialog (Fig. 3.23). The **ID** node contains the names of the objects in the GUI. Expand the **ID** node and select `welcomeTextView`.



**Fig. 3.23** | Selecting the value for the `droidImageView`'s **Layout below** property.

4. Set the `droidImageView`'s **Layout center horizontal** property to `true` to center the `ImageView` in the layout.
5. Set the `droidImageView`'s **Src property** to the image that should be displayed. To do so, click the **Value** field for this property, then click the ellipsis button to display the **Reference Chooser** dialog (Fig. 3.24). The **Drawable** node contains the resources in your app's `drawable` folders within the `res` folder. In the dialog, expand the **Drawable** node and select `android`, which represents the `android.png` image.
6. Repeat items 1–5 above to create the `bugImageView`. For this component, set its **Id** property to `@+id/bugImageView`, its **Src** property to `bug` and its **Layout below** property to `droidImageView`.

The Visual Layout Editor window should now appear as shown in Fig. 3.25.

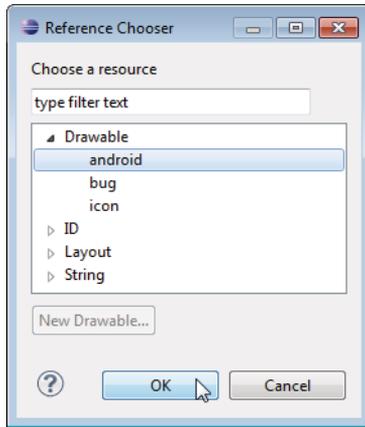


Fig. 3.24 | Selecting the value for the `droidImageView`'s `Src` property.

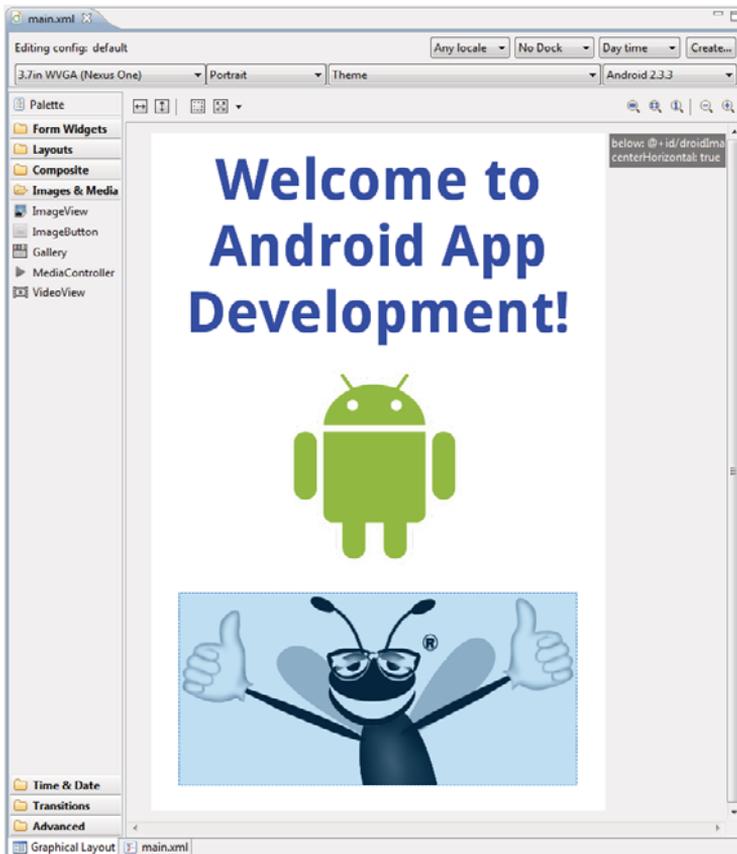


Fig. 3.25 | Visual Layout Editor window after completing the GUI configuration.

## 3.6 Examining the main.xml File

XML is a natural way to express a GUI's contents. It allows you, in a human- and computer-readable form, to say which layouts and components you wish to use, and to specify their attributes, such as size, position and color. The ADT Plugin can then parse the XML and generate the code that produces the actual GUI. Figure 3.26 shows the final main.xml file after you perform the steps in Section 3.5. We reformatted the XML and added some comments to make the XML more readable. (Eclipse's **Source > Format** command can help you with this.) As you read the XML, notice that each XML attribute name that contains multiple words does not contain spaces, whereas the corresponding properties in the **Properties** window do. For example, the XML attribute `android:paddingTop` corresponds to the property **Padding top** in the **Properties** window. When the IDE displays property names, it displays the multiword names as separate words for readability.

---

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- main.xml -->
3  <!-- Welcome App's XML layout. -->
4
5  <!-- RelativeLayout that contains the App's GUI components. -->
6  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      android:id="@+id/welcomeRelativeLayout" android:background="#FFFFFF">
10
11     <!-- TextView that displays "Welcome to Android App Development!" -->
12     <TextView android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="@string/welcome"
15         android:textSize="40sp" android:id="@+id/welcomeTextView"
16         android:textColor="#00F" android:textStyle="bold"
17         android:layout_centerHorizontal="true" android:gravity="center"
18         android:layout_marginTop="10dp"></TextView>
19
20     <!-- ImageView that displays the Android logo -->
21     <ImageView android:layout_height="wrap_content"
22         android:layout_width="wrap_content" android:id="@+id/droidImageView"
23         android:layout_centerHorizontal="true"
24         android:src="@drawable/android"
25         android:layout_below="@id/welcomeTextView"></ImageView>
26
27     <!-- ImageView that displays the Deitel bug logo -->
28     <ImageView android:layout_height="wrap_content"
29         android:layout_width="wrap_content" android:id="@+id/bugImageView"
30         android:src="@drawable/bug"
31         android:layout_below="@id/droidImageView"
32         android:layout_centerHorizontal="true"></ImageView>
33 </RelativeLayout>

```

---

**Fig. 3.26** | Welcome App's XML layout.

**welcomeRelativeLayout**

The `welcomeRelativeLayout` (lines 6–33) contains all of the app’s GUI components.

- Its opening XML tag (lines 6–9) sets various `RelativeLayout` attributes.
- Line 6 uses the `xmlns` attribute to indicate that the elements in the document are all part of the `android` XML namespace. This is required and auto-generated by the IDE when you create any layout XML file.
- Lines 7–8 specify the value `match_parent` for both the `android:layout_width` and `android:layout_height` attributes, so the layout occupies the entire width and height of layout’s parent element—that is, the one in which this layout is nested. In this case, the `RelativeLayout` is the *root node* of the XML document, so the layout occupies the *entire screen* (excluding the status bar).
- Line 9 specifies the values for the `welcomeRelativeLayout`’s `android:id` and `android:background` attributes.

**welcomeTextView**

The first element in the `welcomeRelativeLayout` is the `welcomeTextView` (lines 12–18).

- Lines 12 and 13 set the `android:layout_width` and `android:layout_height` attributes to `wrap_content`. This value indicates that the view should be just large enough to fit its content, including its padding values that specify the spacing around the content.
- Line 14 sets the `android:text` attribute to the string resource named `welcome` that you created in Section 3.5, Step 5.
- Line 15 sets the `android:textSize` attribute to `40sp` and the `android:id` attribute to `"@+id/welcomeTextView"`.
- Line 16 sets the `android:textColor` attribute to `"#00F"` (for blue text) and the `android:textStyle` attribute to `"bold"`.
- Line 17 sets the `android:layout_centerHorizontal` attribute to `"true"`, which centers the component horizontally in the layout, and sets the `android:gravity` attribute to `"center"` to center the text in the `TextView`. The `android:gravity` attribute specifies how the text should be positioned with respect to the width and height of the `TextView` if the text is smaller than the `TextView`.
- Line 18 sets the `android:marginTop` attribute to `10dp` so that there’s some space between the top of the `TextView` and the top of the screen.

**droidImageView**

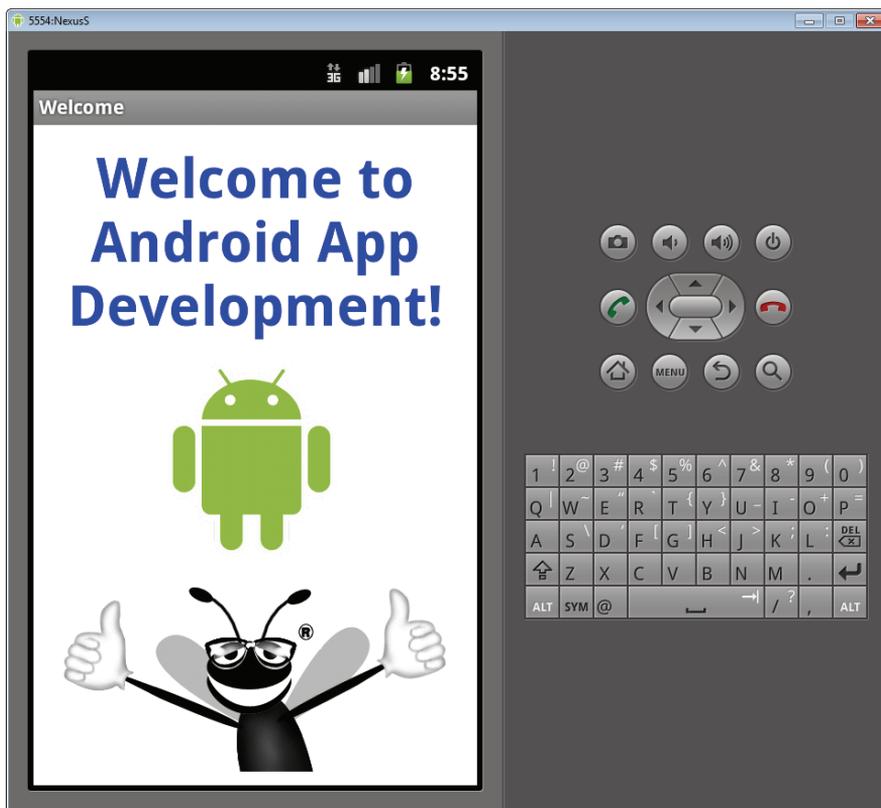
The last two elements nested in the `welcomeRelativeLayout` are the `droidImageView` (lines 21–25) and the `bugImageView` (lines 28–32). We set the same attributes for both `ImageView`s, so we discuss only the `droidImageView`’s attributes here.

- Lines 21 and 22 set the `android:layout_width` and `android:layout_height` attributes to `wrap_content`. Line 22 also sets the `android:id` attribute to `"@+id/droidImageView"`.
- Line 23 sets the `android:layout_centerHorizontal` attribute to `"true"` to centers the component in the layout.

- Line 24 sets the `android:src` attribute to the drawable resource named `android`, which represents the `android.png` image.
- Line 25 sets the `android:layout_below` attribute to `"@id/welcomeTextView"`. The `RelativeLayout` specifies each component's position relative to other components. In this case, the `ImageView` follows the `welcomeTextView`.

### 3.7 Running the Welcome App

To run the app in an Android Virtual Device (AVD), right click the app's root node in the **Package Explorer** window and select **Run As > Android Application**. Figure 3.27 shows the running app.



**Fig. 3.27** | Welcome app running in an AVD.

### 3.8 Wrap-Up

This chapter introduced key features of the Eclipse IDE and the ADT Visual Layout Editor. You used the Visual Layout Editor to create a working Android app without writing any code. You used the `TextView` and `ImageView` GUI components to display text and im-

ages, respectively, and you arranged these components in a `RelativeLayout`. You edited the properties of GUI components to customize them for your app. You then tested the app in an Android Virtual Device (AVD). Finally, we presented a detailed walkthrough of the XML markup that generates the GUI.

In the next chapter we introduce how to program Android apps using Java. Android development is a combination of GUI design, and Java and XML coding. Java allows you to specify the behavior of your apps. You'll develop the **Tip Calculator** app, which calculates a range of tip possibilities when given a restaurant bill amount. You'll design the GUI and add Java code to specify how the app should process user inputs and display the results of its calculations.

# Index

## Numerics

- 0-click NFC Peer-to-Peer Sharing **15**
- 3D Art** app xvi
- 3D graphics xiv, xvi, 14
- 3D motion processing 14
- 3D stack 14
- 3G network 13

## A

- accelerometer xiv, xvi, **19**
- accelerometer sensor **228**, 237
- access a color resource 127
- access a non-standard library 296
- access a predefined color resource 127
- access a resource value 129
  - @color/ 129
  - @dimen/ 130
  - @string/ 129
- access Android services 132
- accessibility 37, 43
- accessing Android content providers 18
- ACCURACY\_COARSE constant of class Criteria 303
- ACCURACY\_FINE constant of class Criteria 303
- ACCURACY\_HIGH constant of class Criteria 303
- ACCURACY\_LOW constant of class Criteria 303
- ACCURACY\_MEDIUM constant of class Criteria 303
- acquire method of class WakeLock **305**
- action bar 12, 13, 14, **228**, 230, 392, **395**
- action element of the manifest file **143**
- ACTION\_APPWIDGET\_UPDATE 399, 454
- ACTION\_GET\_CONTENT constant of class Intent 346

- ACTION\_VIEW constant of classIntent **141**
- ActionBar class 398, 401, 403
  - selectTab method **403**
- activity 14, **41**
  - is shut down 185
- Activity and Task Design Guidelines* 41
- Activity chooser 346
- Activity chooser dialog 322, 325
- Activity class 94, 106, **334**, 336, 341, 345, 346
  - finish method 276
  - getFragmentManager method **401**
  - getIntent method **276**, 282
  - getMenuInflater method **273**
  - getPreferences method **211**
  - getString method **140**
  - getSystemService method 138, 236
  - lifecycle 14
  - lifecycle methods 179
  - onActivityResult method 375
  - onCreate method **108**, 184, 212, 276, 401
  - onCreateOptionsMenu method **152**, 239
  - onDestroy method **179**, 184, 185
  - onOptionsItemSelected method **152**, 241
  - onPause method **179**, 184, 185
  - onRestoreInstanceState method 402
  - onResume method **212**, 271, 403
  - onSaveInstanceState method **108**, 113, 351, 355, 402
  - onStart method 212
  - onStop method **272**, 305
  - onTouchEvent method **179**, 185

- Activity class (cont.)
  - receiving results from another Activity 325
  - RESULT\_CANCELED constant 336
  - RESULT\_OK constant 336
  - runOnUiThread method **199**, **372**, 373
  - sendBroadcast method **405**
  - sent to background 185
  - setContentView method **110**
  - setResult method **382**
  - setVolumeControlStream method **180**
  - startActivity method 325
  - startActivityForResult method **325**, 334, 340, 345
  - startManagingCursor method **273**
- activity element of the manifest
  - android:label attribute **143**
  - android:name attribute **143**
  - android:screenOrientation attribute **174**, 208
  - android:theme attribute **296**, 299
  - android:windowSoftInputMode attribute **143**
  - AndroidManifest.xml
    - activity element 174
- activity fragment 14
- Activity Not Responding (ANR) dialog 263
- activity stack **41**
- Activity>default para font> class
  - onCreate method 270
- ActivityNotFoundException class 141, 334
- Adapter class xvi, 180, **263**
- AdapterView class xvi, **263**, 274
- AdapterView.OnItemClickedListener interface 270
- AdapterView.OnItemClickedListener interface 274
  - onItemClick method **274**

- AdapterView.OnItemClickListener interface **395**
- Adaptive Multi-Rate Wideband encoding (AMR-WB) 12
- adb (Android Debug Bridge) xxix
- add a class to a project 183
- addCallback method of class SurfaceHolder **190**
- adding components to a row 94
- Address Book app xvi, 20
- AdMob 55, 56
- ADT (Android Development Tools Plugin) xv, 6, 19
- ADT (Android Development Tools) 69, 70
  - Plugin for Eclipse xxii, **xxiii**
- ADT Plugin 69, 93
- ADT Plugin for Eclipse 45, 50
- ADT Plugin tools 94
- Advanced Audio Coding (AAC) 12
- advertising revenue 56
- AlarmManager class 399
- AlertDialog class **122**, 132, 137, 228
  - custom GUI for user input 325
- AlertDialog.Builder class
  - setMultiChoiceItems method **173**
- AlertDialog.Builder class **122**, 137
  - setItems method **173**
- align text
  - center\_vertical 101
- aligning text
  - center 100
- alpha 231
- alpha (transparency) values 80
- alpha animation 220
- alpha animation for a View **159**
- alpha method of class Color **241**
- alphabetical order 119
- altitude 297, 303
- Amazon Mobile app 55
- analysis **22**
- android xxix
- Android 2.2 (Froyo) **7**
- Android 2.3 303
  - overscroll 262
- Android 2.3 (Gingerbread) **10**
- Android 2.3.3 10
- Android 3.x 7, 208, 228, 230, 379, 396
  - action bar **228**, 392, **395**
  - android:resizeMode attribute of a menu item **399**
  - DialogFragment class 393, **394**
- Android 3.x (cont.)
  - fragment 392, **394**
  - Fragment class 393, **394**
  - FragmentManager class **395**
  - FragmentTransaction class **395**
  - holographic theme **228**, 230
  - Honeycomb **12**
  - JsonReader 391
  - ListFragment class 393, **394**
  - property animation 159
  - resizable app widget 393
  - SQLiteOpenHelper class's onDowngrade method 289
  - tabbed navigation 392
- Android APIs 5
- Android app marketplaces 54
  - Amazon Appstore 54
  - AndAppStore 54
  - Androidguys 54
  - Andspot Market 54
  - GetJar 54
  - Handango 54
  - Mplayit 54
  - PocketGear 54
  - Shop4Apps 54
  - SlideMe 54
  - Youpark 54
  - Zeewe 54
- Android Application Error Reports **9**, 59
- Android application package file (.apk file) 52
- Android Best Practices Resource Center ([www.deitel.com/AndroidBestPractices/](http://www.deitel.com/AndroidBestPractices/)) xvii, xviii, 32
- Android Cloud to Device Messaging (C2DM) **9**
- Android Content Guidelines* 53
- Android Debug Bridge (adb) xxix
- Android developer documentation ([developer.android.com](http://developer.android.com)) xvii
- Android Developer's Guide* **38**
- Android Development Tools (ADT) 69, 70
  - Plugin for Eclipse xxii, **xxiii**
- Android Development Tools (ADT) Plugin xv, 6, 19
- Android Device Chooser** window 294
- Android device manufacturers xiv
- Android emulator **xxvi**, xxix, 19, 292, 295
- Android for Programmers* page on InformIT xvii
- Android for Programmers* website xiv, xvii, xix, xxii, xxix, xxx
- Android Ice Cream Sandwich **15**
- Android Icon Templates Pack** 40
- Android Manifest Editor **45**
- Android Maps APIs xvi, xvii
- Android Market **3**, 4, 16, 32, 36, 44, 47, 55, 61
  - developer profile 3
  - filter 38, 39
  - language 52
  - location 53
  - locations 53
  - price 53
  - promotional graphic 52
  - publish 44, 51
  - publisher account 57
  - screenshots 52
  - Upload Application** 51
  - upload assets 52
- Android Market Content Policy for Developers* 44
- Android Market Developer Console 59
- Android Market Developer Distribution Agreement 3
- Android NDK 9
- Android Newsgroups
  - Android Discuss 3
- Android predefined color 127
- Android project
  - res folder **74**, **81**
  - value folder **81**
- Android Resource Center [www.deitel.com/android/](http://www.deitel.com/android/) xviii, 32
- Android SDK xxii, xxiii, xxvi, **2**, 19
- Android SDK 2.x xiv, xvi
- Android SDK 3.x 19
- Android SDK and AVD Manager** xxiv, xxvii, 295
- Android SDK versions and API levels 73
- Android service to inflate a layout 136
- Android services
  - access 132
- Android Software Development Kit (SDK) **xxiii**
- Android source code and documentation
  - FAQs 5
  - licenses 5
  - philosophy and goals 5
  - source code 4, 5
- Android User Interface Guidelines* 40

- Android versions
  - Android 1.6 (Donut) 6
  - Android 2.0–2.1 (Eclair) 7
  - Android 2.2 (Froyo) 7
  - Android 2.3 (Gingerbread) 7
  - Android 3.0 (Honeycomb) 7
- Android Virtual Device (AVD)
  - xxvi**, 19, 23, 26, 27, 69, 89
  - Setting hardware emulation options **xxvii**, 32
- `android:adjustViewBounds`
  - attribute of an `ImageView` **157**
- `android:background` attribute 88
  - of a `TextView` 265
- `android:drawableBottom`
  - attribute **328**
- `android:drawableLeft` attribute **328**
- `android:drawableRight` attribute **328**
- `android:drawableTop` attribute **328**
- `android:duration` attribute of a `translate animation` **159**
- `android:fromXDelta` attribute of a `translate animation` **159**
- `android:gravity` attribute 88
- `android:hardwareAccelerated` of the application element **208**
- `android:hint` attribute of an `EditText` **129**
- `android:icon` attribute of an item element **269**
- `android:icon` of the application element **142**
- `android:id` attribute 88
- `android:imeOptions` attribute of an `EditText` **129**, 268
- `android:inputType` attribute 268
- `android:inputType` attribute of an `EditText` 268
- `android:label` of the activity element **143**
- `android:label` of the application element **142**
- `android:layout_below` attribute 89
- `android:layout_centerHorizontal` attribute 88
- `android:layout_column` attribute of a component 100
- `android:layout_height` attribute 88
  - `fill_parent` value 88
  - `wrap_content` value 88
- `android:layout_span` attribute 129
- `android:layout_span` attribute of a component 100
- `android:layout_width` attribute 88
  - `match_parent` value 88
  - `wrap_content` value 88
- `android:marginTop` attribute 88
- `android:minSdkVersion` attribute 208, 396
- `android:name` of the activity element **143**
- `android:orderInCategory`
  - attribute of an item element **269**
- `android:resizeMode` attribute of a menu item **399**
- `android:screenOrientation`
  - attribute 181
  - of the activity element **174**, 208
- `android:showAsAction` attribute 395
- `android:showAsAction` attribute of a menu item **398**
- `android:singleLine` attribute of an `EditText` **329**
- `android:src` attribute 89
- `android:startOffset` attribute of a `translate animation` **159**
- `android:stretchColumns`
  - attribute of a `TableLayout` **97**, 123, 129
- `android:targetSdkVersion` attribute 230
- `android:text` attribute 88
- `android:textColor` attribute 88
- `android:textOff` attribute of class `ToggleButton`
- `android:textOn` attribute of class `ToggleButton`; `ToggleButton` class
  - `android:textOn` attribute **299**
- `android:textSize` attribute 88
- `android:textStyle` attribute 88
- `android:theme` attribute of the activity element **296**, 299
- `android:theme` of the application element **174**
- `android:thumb` attribute of a `SeekBar` 231
- `android:title` attribute of an item element **269**
- `android:titleCondensed`
  - attribute of an item element **269**
- `android:toXDelta` attribute of a `translate animation` **159**
- `android:windowSoftInputMode` of the activity element **143**
- `android.animation` package **207**
- `android.app` package 106, 132, 228, 263, 394, 395
- `android.appwidget` package **395**
- `android.content` package **122**, 132, 229, **395**, 396
- `android.content.res` package 152, **163**
- `android.database` package **263**
- `android.database.sqlite` package **263**
- `android.graphics` package 181, 229
- `android.graphics.drawable` package 152, **327**
- `android.hardware` package **228**, 364
- `android.intent.action.CHOOSE` R 346
- `android.location` package **297**
- `android.media` package 180, **326**
- `android.net` package 132
- `android.os` package 106, 152, **263**, 297
- `android.permission`
  - `ACCESS_FINE_LOCATION` **299**
  - `ACCESS_mock_LOCATION` **299**
  - `INTERNET` **299**
  - `WAKE_LOCK` **299**
- `android.provider` package **325**
- `android.text` package 107
- `android.util` package 152, 189, 396
- `android.view` package 132, 152, 179, 180, 263, 364
- `android.view.animation` package 152
- `android.view.inputmethod` package **132**
- `android.widget` package **75**, 107, 132, 229, 262, 263, 297, 299, 326
- Android@Home framework **15**
- AndroidLicenses 54
- `AndroidManifest.xml` 40, **45**, **122**, 142, 296, 298, 327, 365
  - action element **143**
  - application element **142**, **143**, 174
  - category element **143**

- AndroidManifest.xml (cont.)
  - describing multiple activities **262**
  - intent-filter element **143**
  - manifest element **142**
  - permissions 46
  - receiver element **396**
  - service element **396**
  - uses-sdk element **142**
- animate method of class View **207**, 220
- animateTo method of class MapController **306**
- animation xiv, xvi, 207
  - alpha animation for a View **159**, 220
  - AnimatorListener 220
  - cancel **207**
  - duration 207, 220
  - end **207**
  - framework 15
  - interpolator 207
  - lifecycle events 207
  - LinearInterpolator 221
  - location 220
  - manual 180
  - options in an XML file 152, 175
  - repeat **207**
  - rotate animation for a View **159**
  - rotation 220
  - scale 220
  - scale animation for a View **159**
  - set **159**
  - start **207**
  - thread 180
  - TimeInterpolator class 220
  - translate animation for a View **159**
  - translation 220
  - tween **159**
  - View based **159**
- Animation class **152**
  - setRepeatCount method **152**, 163
- AnimationListener interface 216
  - onAnimationCancel method 216
  - onAnimationEnd method 216
- AnimationUtils class **152**, 163
  - loadAnimation method **152**, 163
- Animator class **221**
  - cancel method **216**
- AnimatorListener interface **207**, 220
  - onAnimationEnd method **221**
  - onAnimationStart method **221**
- AnimatorListenerAdapter class **207**, 221
- AnimatorUpdateListener interface **207**
- anonymous inner class 94
- ANR (activity not responding) dialog 263
- ANR (Application Not Responding) dialog **109**, 134
- ANR dialog 180
- ant.apache.org/xxix
- anti-aliasing 248
- Apache Ant xxix
- API key 294
- APIs
  - Apps on external storage 9
  - Camera and Camcorder 9
  - Data backup 9
  - Device policy management 9
  - Graphics 9
  - Media framework 9
  - UI framework 9
- APIs (Application Programming Interfaces) 9
  - .apk file (Android application package file) **49**, 52
  - app xxii
  - app design resources 38
  - app development xxii
  - app permissions **296**
    - requesting 299
  - app platforms
    - Bing 65
    - BlackBerry 65
    - Chrome 65
    - Facebook 65
    - Foursquare 65
    - Google 65
    - Gowalla 65
    - iPhone 65
    - LinkedIn 65
    - Symbian 65
    - Twitter 65
    - webOS 65
    - Windows Mobile 65
    - Yahoo 65
- app review and recommendation sites
  - Android Tapp 17
  - AndroidLib 17
  - AndroidZoom 17
  - AppBrain 17
  - Applicious 17
  - doubleTwist 17
  - mplayit 17
- app review sites
  - Android and Me 62
  - Android App Review Source 62
  - Android Police 62
  - Android Tapp 62
  - AndroidGuys 62
  - AndroidLib 62
  - AndroidZoom 62
  - Androinica 62
  - AppBrain 62
  - Applicious 62
  - AppStoreHQ 62
  - Best Android Apps Review 62
  - Phandroid 62
- app store analytics 55
- App Widget xiv, xvi
- app widget 393
  - adding to home screen 394
  - layout 399
- app-driven approach xv, **2**
- application element of the manifest file **142**, **143**, 174
  - android:hardwareAccelerated attribute **208**
  - android:icon attribute **142**
  - android:label attribute **142**
  - android:theme attribute **174**
- Application Not Responding (ANR) dialog **109**, 134
- Application Programming Interfaces (APIs) 9
- application resource 17
- application resources
  - (developer.android.com/guide/topics/resources/index.html) **81**
- apply method of class SharedPreferences **405**
- apply method of class SharedPreferences.Editor **136**
- apps in the book xv
- appwidget
  - resizable 393
- AppWidgetProvider class **395**, 399, 453, 456
- argb method of class Color **242**
- ARGB\_8888 constant **249**
- arrange GUI components into rows and columns 94
- ArrayAdapter class **326**, 332, 336, 338, 341, 343, 349
  - getView method 339
  - remove method **348**
- ArrayList class **152**
- Arrays class 131
  - sort method **134**, **135**

arrays.xml resource file 398  
 asset 52  
 AssetManager class **152**  
   list method 163  
 assets folder of an Android app **151**  
 AsyncTask class **263**, 271, **272**, 272, **272**, 272, 273, 276, 277, 278, 372, 373, 374, 447  
   execute method **271**  
 AtomicBoolean class **228**, 235  
 attribute 14  
   in the UML 22  
   of a class 21  
   of an object 22  
 Attributes  
   android:background 88  
   android:drawableBottom **328**  
   android:drawableLeft **328**  
   android:drawableRight **328**  
   android:drawableTop **328**  
   android:gravity 88  
   android:id 88  
   android:layout\_below 89  
   android:layout\_centerHorizontal 88  
   android:layout\_height 88  
   android:layout\_width 88  
   android:marginTop 88  
   android:src 89  
   android:text 88  
   android:textColor 88  
   android:textSize 88  
   android:textStyle 88  
 AttributeSet class 189  
 audio xiv, xvi, 18  
 audio effects API 12  
 audio library access xvi  
 audio stream  
   music 190  
 audio streams 180  
   music 180  
 audio volume 180  
 AudioManager class **180**, 190  
 auto-discovery 7  
 auto-focus 13  
 AVD  
   sending GPS data to 295  
   transferring music and photos 323  
 AVD (Android Virtual Device) **xxvi**, **19**, 23, 26, 27, 69

**B**

background  
   activity sent to 185  
 background process 109  
**Background** property of a layout or component **80**  
 backward compatibility 13  
 Bank of America app 56  
 barometer sensor 14  
 battery  
   conserve 304  
 battery life 297  
 bearing 292, 301, 303, 311, 314  
 BearingRequired method of class Criteria **303**  
 bearingTo method of class Location **306**  
 Before You Begin xxii  
 beginArray method of class JsonReader 435, 440  
 beginObject method of class JsonReader 435, 438, 440  
 beginTransaction method **408**  
 behavior  
   of a class 21  
 best practices xvii, 38  
 bezier curve 253  
 Bitmap class **181**, **229**, 245  
   bitmap encoding **249**  
   compress method **256**  
   createBitmap method **249**  
   eraseColor method **245**  
 Bitmap.Config.ARGB\_8888  
   constant **249**  
 BitmapDrawable class **327**  
 BitmapFactory class **326**  
   decodeStream method **358**, 439  
 BitmapFactory.Options class **326**  
 blogging 60  
 blue method of class Color **241**  
 Bluetooth xvi, 8, 13  
   tethering 13  
 Bluetooth Advanced Audio Distribution Profile (A2DP) 15  
 brand awareness 55  
 branding apps  
   Amazon Mobile 55  
   Bank of America 56  
   Best Buy 56  
   Epicurious Recipe 56  
   ESPN ScoreCenter 56  
   ING Direct ATM Finder 56  
   Men's Health Workouts 56  
   NFL Mobile 56

branding apps (cont.)  
   NYTimes 56  
   Pocket Agent 56  
   Progressive Insurance 56  
   UPS Mobile 56  
   USA Today 56  
   Wells Fargo Mobile 56  
 broadcast Intent 396  
 BroadcastReceiver class **395**, **396**, 453  
**Browser** 13  
 built-in content providers 325  
 Bundle class 106, **110**, 113  
   for an Intent 275  
   getDouble method **111**  
   getLong method **276**  
   getString method **111**  
 Button class **122**, 132  
**C**  
 C2DM (Android Cloud to Device Messaging) **9**  
 CamcorderProfile classes 9  
**Camera** 13  
 camera 6, 11, 361  
   color effects 380  
   reconfigure 380  
 Camera API xvi, 12  
 Camera class **364**, 378  
   open method **381**  
   setParameters method **380**, 382  
   setPreviewDisplay method **382**  
   startPreview method **382**  
   stopPreview method **382**  
 camera controls 8  
 CAMERA permission 365  
 camera preview 382  
 Camera Preview API 9  
 camera settings 8  
 Camera.Parameters class **364**, 378, 380  
   setColorEffect method **380**, 382  
   setPreviewSize method **382**  
 Camera.PictureCallback class **364**, **382**, 383  
   onPictureTaken method **382**  
 Camera.ShutterCallback class **383**  
 Camera.Size class 378  
 cancel1 method of class Animator **216**  
**Cannon Game** app xvi, 20

- Canvas class **181**, 229, 316
  - drawBitmap method **250**
  - drawCircle method **198**
  - drawLine method **198**
  - drawPath method **250**, 254
  - drawRect method **198**
  - drawText method **198**
  - rotate method 314
- capturing high-quality audio 12
- carrier billing 55
- case-insensitive sort 134
- category element of the manifest file **143**
- cell in a TableLayout 94
- center text 100
- center\_vertical value of the Gravity property of a component 101
- chain method calls 207
- changeCursor method of class CursorAdapter **272**, 273
- characteristics of great apps 36
- check-in 60
- class 17, **21**, 94
  - instance variable **22**
- class library **5**
- Classes **208**
  - ActionBar 398, 401, 403
  - Activity 94, 106
  - ActivityNotFoundException 141, 334
  - Adapter **263**
  - AdapterView **263**
  - AlarmManager 399
  - AlertDialog **122**, 132
  - AlertDialog.Builder **122**
  - Animation **152**
  - AnimationUtils **152**, 163
  - Animator **221**
  - AnimatorListenerAdapter **207**, 221
  - AppWidgetProvider **395**, 399, 453, 456
  - ArrayAdapter **326**, 338, 343, 349
  - ArrayList **152**
  - Arrays 131
  - AssetManager **152**
  - AsyncTask **263**, 271, 276, 447
  - AtomicBoolean **228**, 235
  - AttributeSet 189
  - AudioManager **180**, 190
  - Bitmap **181**, **229**, 245
  - BitmapDrawable **327**
  - BitmapFactory **326**
  - BitmapFactory.Options **326**
- Classes (cont.)
  - BroadcastReceiver **395**, **396**, 453
  - Bundle 106, **110**, 113
  - Button **122**, 132
  - Camera **364**, 378
  - Camera.Parameter 378
  - Camera.Parameters **364**
  - Camera.PictureCallback **364**, **382**, 383
  - Camera.ShutterCallback **383**
  - Canvas **181**, 229
  - Collections **152**
  - Color 241
  - ComponentName **456**
  - ContentResolver **229**, 256
  - ContentValues **286**
  - ContentValues class **256**
  - Context 132, 371
  - Criteria **297**
  - Cursor **263**
  - CursorAdapter **263**, 271
  - CursorFactory 288
  - Dialog **228**
  - DialogFragment 393, **394**, 422, 424
  - DialogInterface 132
  - Display **298**
  - Drawable **152**, 167
  - EditText 94, **94**, 107
  - File 371
  - FileInputStream **364**
  - FileOutputStream **363**
  - Fragment 393, **394**, 401, 415, 422, 425, 446
  - FragmentManager **395**, 401
  - FragmentTransaction **395**
  - FragmentTransactions 397
  - FrameLayout 182, **299**, 397
  - GeoPoint **297**
  - GestureDetector.SimpleOnGestureListener **180**, 186
  - Handler **152**
  - HashMap **153**
  - ImageView **69**, 83
  - InputMethodManager 132
  - InputStream 152, 167, 175
  - InputStreamReader 434, 449
  - Intent **122**, 132, **141**
  - IntentService 455
  - JsonReader 391, **396**, 434, 438, 447
  - LayoutInflater **122**, 132
  - LayoutParams 311
  - LinearInterpolator 221
- Classes (cont.)
  - ListActivity **263**
  - ListFragment 393, **394**, 415
  - ListView **262**
  - Location **297**
  - LocationManager **297**
  - Log **152**, 167
  - MapActivity **297**
  - MapView **297**
  - MediaController **364**
  - MediaPlayer **326**, 353
  - MediaPlayer.OnCompletionListener **364**
  - Menu **152**, 170
  - LayoutInflater **263**, **273**
  - MotionEvent **179**, 185, **229**, 251
  - NotSerializableException **363**
  - ObjectAnimator **207**
  - ObjectInputStream **364**, 372
  - ObjectOutputStream 373
  - ObjectOutputStream **363**
  - OutputStream 363
  - OutputStream 229, 256
  - Paint **181**
  - Path **229**
  - PendingIntent **395**, 456
  - PowerManager **297**
  - Projection **317**
  - R **110**
  - R.color 127
  - R.drawable **110**
  - R.id **110**, **111**
  - R.layout **110**
  - R.string **110**
  - RelativeLayout.LayoutParams **219**
  - RemoteViews 438, 453, 456, 458
  - Resources **163**
  - ScrollView **121**
  - SeekBar 92, **94**, 107
  - Sensor **228**
  - SensorEvent **239**
  - SensorManager **228**, 235, 236
  - SharedPreferences **122**, 132, 133, 212
  - SharedPreferences.Editor **122**, 136
  - SimpleCursorAdapter **271**
  - SoundPool **180**, 190
  - SQLiteDatabase **263**
  - SQLiteOpenHelper **263**
  - SurfaceHolder **180**, 190
  - SurfaceView **180**, 190, 366

## Classes (cont.)

TableLayout **94**, 123  
 TableRow **94**  
 TextView **69**, 80, 94, 107  
 Thread 180, 201  
 Toast **229**, 256, 295  
 ToggleButton 292, **297**  
 TransitionDrawable **327**, 358  
 Uri 132, **141**  
 ValueAnimator **207**  
 VideoView 361, 363, **364**  
 View 180  
 ViewGroup **121**  
 ViewPropertyAnimator **207**, 220  
 WakeLock **297**

clear method of class  
   SharedPreferences.Editor 140

close method of class  
   SQLiteOpenHelper **285**

code file 52  
 code highlighting xvii, 2  
 code license xv  
 code walkthrough 2  
 code.google.com/android/maps-api-tos.pdf xvi

collection xvi  
   shuffle 167

Collections class **152**  
   shuffle method **152**

collision detection 192, 194

color 181, 231  
   transparent 130

Color class 241  
   alpha method **241**  
   argb method **242**  
   blue method **241**  
   green method **241**  
   red method **241**

color effects 380

color element representing a color resource **125**

color resource  
   predefined 127

colors.xml 124, 154

columns in a TableLayout 94  
   numbered from 0 100

com.google.android.maps package **297**

command-line xxix

commit method of class  
   FragmentManager **408**

Comparator<String> object  
   String.CASE\_INSENSITIVE\_ORDER 134

Compass xvi

compatibility 38

compatibility package 394

compatible device 38

compiling apps 46

component 20

ComponentName class **456**

CompoundButton.OnCheckedChangeListener interface **297**, 302, 309

compress method of class Bitmap **256**

concurrent garbage collector 14

ConcurrentLinkedQueue **208**

ConcurrentLinkedQueue interface **208**

connection state 18

conserve battery power 304

Constants  
   MODE\_PRIVATE **134**  
   MODE\_WORLD\_READABLE **134**  
   MODE\_WORLD\_WRITABLE **134**

Contacts 13

contain other Views **121**

content provider **324**

content providers  
   audio 325  
   built-in 325  
   images 325

ContentResolver class **229**, 256  
   insert method **256**

ContentValues class **256**, **286**  
   put method 256

Context class 132, 214, 371  
   Context.LAYOUT\_INFLATER\_SERVICE **136**  
   getSharedPreferences method **134**  
   getSystemService method **136**  
   startActivity method **122**, 141

Context menu 41

Context.LAYOUT\_INFLATER\_SERVICE ICE constant **136**

Context.SENSOR\_SERVICE constant 237

ContextWrapper class  
   getAssets method **163**, 167  
   getResources method **163**

control 19

corners element of a shape **265**

Craigslist 9

create a layout XML file 130, 158

create a resource file 124, 154

Create New Android String dialog 81

createBitmap method of class Bitmap **249**

createChooser method of class Intent **346**

createFromStream method of class Drawable **152**, 167

creating a database 285

Criteria class **297**, 303  
   ACCURACY\_COARSE 303  
   ACCURACY\_FINE 303  
   ACCURACY\_HIGH 303  
   ACCURACY\_LOW 303  
   ACCURACY\_MEDIUM 303  
   BearingRequired method **303**  
   NO\_REQUIREMENT 303  
   POWER\_HIGH 303  
   POWER\_LOW 303  
   POWER\_MEDIUM 303  
   setAccuracy method **303**  
   setAltitudeRequired method **303**  
   setCostAllowed method **303**  
   setPowerRequirement method **303**

cryptographic key 46

Cursor class **263**, 278, 288  
   deactivate method **272**, 273  
   deactivated Cursor **273**  
   getColumnIndex method **278**  
   getColumnIndexOrThrow method **278**  
   getString method **278**  
   life-cycle **273**  
   moveToFirst method **278**  
   query method **273**

Cursor visible property of an EditText 102

CursorAdapter class **263**, 271  
   changeCursor method **272**, 273  
   getCursor method **272**

CursorFactory class 288

custom subclass of View 186

custom view 179

customizable Home screens 12

## D

Dalvik Debug Monitor Server (DDMS) 152

Dalvik Debug Monitor Service (DDMS) 50

- Dalvik Virtual Machine 7
- data binding 263
- data cost 303
- database
  - creating 285
  - opening 285
  - upgrading 285
  - version number 288
- date/time picker 14
- DDMS (Dalvik Debug Monitor Server) 50, 152
  - sending GPS data to an AVD 295
- DDMS perspective 295
  - GPX tab 296
  - LogCat tab **152**
- deactivate method of class
  - Cursor **272**, 273
- deactivated Cursor **273**
- debug certificate 294
- debugging 47
  - logging exceptions **152**, 167
- decodeStream method of class
  - BitmapFactory **358**, 439
- @deitel (Deitel on Twitter) xviii
- Deitel Facebook Page
  - (www.deitel.com/deitelfan/) xviii
- Deitel Resource Centers
  - (www.deitel.com/ResourceCenters.html) xviii, 33
- Deitel Web site (www.deitel.com) xxx
- Deitel® Buzz Online Newsletter
  - (www.deitel.com/newsletter/subscribe.html) xviii, xxx, 33, 459
- Deitel® Training
  - (www.deitel.com/training) 459
- delete method of class
  - SQLiteDatabase **288**
- density 40
- density-independent pixel 125
- density-independent pixels (dp or dip) 82, **82**, 83
- deserialized **364**
- design process **22**
- designing for multiple devices 39
- designing for performance 38
- designing for responsiveness 38
- designing for seamlessness 38
- Dev Guide 38, 46
- developer documentation 65
  - Android Compatibility* 65
- developer documentation (cont.)
  - Android Emulator* 66
  - Android FAQs* 66
  - Android Market Content Policy for Developers* 66
  - Android User Interface Guidelines* 66
  - AndroidManifest.xml File Element* 65
  - Application Fundamentals* 65
  - Common Tasks and How to Do Them in Android* 66
  - Designing for Performance* 66
  - Designing for Responsiveness* 66
  - Designing for Seamlessness* 66
  - Icon Design Guidelines, Android 2.0* 66
  - In-app Billing* 66
  - Localization* 43, 66
  - Manifest.permission Summary* 65
  - Market Filters* 66
  - Preparing to Publish: A Checklist* 66
  - Publishing Your Applications: Using Intents to Launch the Market Application on a Device* 59
  - Sample Apps* 66
  - Signing Your Applications* 50
  - Speech Input* 66
  - Supporting Multiple Screens* 40, 66
  - Technical Articles* 66
  - Using Text-to-Speech* 66
  - Versioning Your Applications* 66
- developer.android.com/guide/developing/tools/adb.html xxxix
- developer.android.com/guide/developing/tools/emulator.html xxix
- developer.android.com/guide/developing/tools/othertools.html#android xxix
- developer.android.com/guide/publishing/app-signing.html xxix
- developer.android.com/sdk/android-2.2-highlights.html 9
- developer.android.com/sdk/eclipse-adt.html xxiii
- developer.android.com/sdk/index.html xxiii
- developer.android.com/sdk/requirements.html xxiii
- developer.motorola.com/docstools/motodevstudio/download/ xxix
- development perspective **70**
- device administration app policies 15
- device configuration 17
- Device Screen Capture window 50, 51
- .dex file (code file) 52
- Dialog class **228**
  - dismiss method **243**
  - setContentView method **241**, 244
  - show method **241**
- dialog icon 40
- DialogFragment class 393, **394**, 422, 424
- DialogInterface class 132
- DialogInterface.OnClickListener interface **132**, 173
- DialogInterface.OnMultiChoiceClickListener interface 173
- digital certificate **49**, 294
- Digital Rights Management (DRM) framework 15
- digital zoom 8
- digitally sign your app 49
- dimension element representing a dimension resource **125**
- dimen.xml 124, 154
- dimensions of the screen 312
- dip (density-independent pixels) 82, **82**, 83
- direction 292
- dismiss method of class Dialog **243**
- dispatchDraw method of class View **313**
- display a Drawable on a Button **328**
- Display class **298**, 312
  - getHeight method 312
  - getWidth method 312
- distanceTo method of class Location **305**
- documentation
  - Android Developer Guide* 2
  - Android Market Developer Distribution Agreement* 3
  - Android Market Getting Started* 3

- documentation (cont.)
    - application resources (developer.android.com/guide/topics/resources/index.html) **81**
    - Class Index* 2
    - Data Backup* 2
    - Debugging Tasks* 3
    - Managing Projects from Eclipse with ADT* 3
    - Package Index* 2
    - Publishing Your Applications* 3
    - Security and Permissions* 3
    - Tools Overview* 3
    - User Interface Guidelines* 2
    - Using the Android Emulator* 2
  - doInBackground method of class AsyncTask 271, **272**, 272, 277
  - Doodlz app xvi, **23**
  - double tap 20
  - double-tap event 180, 186, 195
  - downloading source code xvii
  - downsample an image 326
  - dp (density-independent pixels) 82, **82**, 83
  - drag 20
  - drag event 253
  - DragEvent framework 14
  - draw
    - circles 181
    - lines 181
    - text 181
  - draw method of class Overlay **316**
  - Drawable
    - display on a Button **328**
  - Drawable class **152**, 167
    - createFromStream method **152**, 167
  - Drawable resource
    - shape element **265**
  - drawBitmap method of class Canvas **250**
  - drawCircle method of class Canvas **198**
  - drawing characteristics 181
    - color 181
    - font size 181
    - line thickness 181
  - drawLine method of class Canvas **198**
  - drawPath method of class Canvas **250**, 254
  - drawRect method of class Canvas **198**
  - drawText method of class Canvas **198**
  - drive sales 55
  - duration of an animation 207, 220
  - dynamically load a resource 163
- ## E
- e method of class Log **167**
  - Eclipse xiv, xv, xxiii, 23, 69
    - download (www.eclipse.org/downloads/) xxiii
    - import project 93, 119, 151, 178, 206
    - Java perspective 23
    - Outline window 93, 94
    - Override/Implement Methods option **113**
  - Eclipse documentation (www.eclipse.org/documentation) xvii
  - Eclipse IDE **2**, 6, 19, 69, 70
    - Properties tab 69
  - Eclipse IDE for Java Developers xxii, xxiii
  - Eclipse Resource Center (www.deitel.com/Eclipse/) xviii, 32
  - edit method of class
    - SharedPreferences **136**
  - Editable interface 107
  - EditText class **94**, 107
    - android:hint attribute **129**
    - android:imeOptions attribute **129**, 268
    - android:inputType attribute 268
    - android:singleLine attribute **329**
    - Cursor visible property 102
    - Long clickable property 102
  - EditText component
    - Cursor visible property 102
    - Text size property 100
  - Emacs
    - setting up for Android xxix
  - emacs xxix
  - Email 13
  - emulator 19, 47
  - emulator functionality 47
  - emulator gestures and controls 20
  - emulator limitations 47
  - encapsulation **22**
  - End User License Agreement (EULA) 46, **47**
  - Enhanced Address Book app xvi
  - Enhanced Slideshow app xvi
  - Environment class
    - getExternalFilesDir method **371**
  - eraseColor method of class Bitmap **245**
  - event distribution 14
  - event handler
    - adding extra information to a View **326**
    - returning false **186**
    - returning true **186**
  - event handling 94
  - events **5**
  - Examples xxx
  - Examples.zip xxx
  - Exchange Calendar 7
  - execSQL method of class SQLiteDatabase **289**
  - execute method of class AsyncTask **271**
  - explicit Intent **141**, **263**, 275
  - export laws 53
  - Export Wizard 49
  - exposure data 9
  - external storage directory 371
  - extra information for use in event handlers **326**
  - extras for an Intent 275
- ## F
- Facebook xviii, **60**, 61
    - Deitel page (www.deitel.com/deitel/fan/) 33, 60, 61, 459
    - friend 61
  - factory settings 7
  - Favorite Twitter Searches app xvi, 118
  - featured apps 16
  - fee-based app 16
  - File class 371
    - getAbsolutePath method 371
  - file processing 361, 370
  - file system access 17
  - FileInputStream class **364**
  - FileOutputStream class **363**
  - fill\_parent value of the
    - android:layout\_height attribute 88
  - financial transaction 58
  - findFragmentById method of class FragmentManager **407**
  - finish method 346
  - finish method of class Activity 276, 346
  - Flag Quiz Game app xvi

- flash support 8
  - fling 20
  - fling touch event 180
  - Focusable** property of a component **102**, 102
  - Folders
    - res/menu 268
    - res/raw **179**, 183
  - folders
    - assets **151**
    - res/anim 159
    - res/drawable 265
  - font size 181
  - format method of class `String` 179
  - format specifier
    - multiple in a `String` resource 179, 182
    - numbering 179
    - numbering in a `String` resource 179, 182
  - format Strings 179
  - forums 3
    - Android Forums 4
    - Android Market Help Forum 4
    - Stack Overflow 3
  - fragment 392, **394**
  - `Fragment` class 393, **394**, 394, 401, 407, 415, 422, 425, 446
    - `getArguments` method 443
    - `isAdded` method 430
    - `onActivityCreated` method 416, 429
    - `onCreateView` method 424, 428
    - `setArguments` method 426, 442
  - fragment element in a layout XML file **397**
  - `FragmentManager` class **395**, 401, 407
    - `findFragmentById` method **407**
  - fragments compatibility package 394
  - `FragmentTransaction` class **395**, 397, 407, 408
    - `commit` method **408**
    - `replace` method **408**
  - frame-by-frame animation xvi
  - `FrameLayout` class 182, **299**, 397
  - fraudulent order 45
  - free app 16, 54
  - free apps xv
  - friend 61
  - friend in Facebook 61
  - Froyo (Android 2.2) 7
  - fully qualify a custom `View`'s class name in an XML layout 179
  - future proof 37
- ## G
- Gallery** 13
  - `Gallery` app 320
  - game loop 192, 201
  - games 37
  - gaming console 6
  - gen folder of an android project **110**
  - geographic coordinates 317
  - geographic location 297
  - geometric path 229
  - `GeoPoint` class **297**, 306, 317
  - gesture **6**
    - double tap 6
    - drag 6
    - fling 6
    - long press 6
    - pinch zoom 6
    - touch 6
  - `GestureDetector.OnDoubleTapListener` interface **180**, 186
  - `GestureDetector.OnGestureListener` interface **180**
  - `GestureDetector.SimpleGestureListener` class **180**, 186
  - `GestureDetector.SimpleGestureListener` class
    - `onDoubleTap` method **186**
  - Gestures
    - drag 6
    - flick 6
    - long press 6
    - pinch 6
    - tap 6
  - `getAbsolutePath` method of class `File` 371
  - `getActionBar` method 413
  - `getActionIndex` method of class `MotionEvent` **252**
  - `getActionMasked` method of class `MotionEvent` **251**
  - `getAll` method of class `SharedPreferences` **134**
  - `getArguments` method of class `Fragment` 443
  - `getAssets` method of class `ContextWrapper` **163**, 167
  - `getBearing` method of class `Location` **306**
  - `getBestProvider` method of class `LocationManager` **304**
  - `getColumnIndex` method of class `Cursor` **278**
  - `getColumnIndexOrThrow` method of class `Cursor` **278**
  - `getController` method of class `MapController` **301**
  - `getCursor` method of class `CursorAdapter` **272**
  - `getData` method of class `Intent` 345
  - `getDefaultSensor` method of class `SensorManager` **236**
  - `getDouble` method of class `Bundle` **111**
  - `getExternalFilesDir` method of class `Environment` **371**
  - `getExtras` method of class `Intent` **276**, 282
  - `getFragmentManager` method of class `Activity` **401**
  - `getHeight` method of class `Display` 312
  - `getHolder` method of class `SurfaceView` **190**
  - `getInt` method of class `SharedPreferences` 214
  - `getIntent` method of class `Activity` **276**, 282
  - `getLastKnownLocation` method of class `LocationManager` **304**
  - `getLatitude` method of class `Location` **306**
  - `getListView` method of class `ListActivity` **270**, 329
  - `getLong` method of class `Bundle` **276**
  - `getLongitude` method of class `Location` **306**
  - `getMenuInflater` method of class `Activity` **273**
  - `getPointerCount` method of class `MotionEvent` **253**
  - `getPreferences` method of class `Activity` **211**
  - `getProjection` method of class `MapView` **317**
  - `getResources` method of class `ContextWrapper` **163**
  - `getSharedPreferences` method of class `Context` **134**
  - `getString` method of class `Activity` **140**
  - `getString` method of class `Bundle` **111**
  - `getString` method of class `Cursor` **278**

- getString method of class
    - JsonReader 436
  - getString method of class
    - Resource 179
  - getString method of class
    - Resources 218
  - getString method of class
    - SharedPreferences **135**
  - getStringArray method of class
    - Resources **406**
  - getSystemService method of class
    - Activity 138
  - getSystemService method of class
    - Context **136**
  - getService method of class
    - clsdd Activity 236
  - getTag method 340, 341
  - getTag method of class View **326**, 340, 341
  - getView method of class
    - ArrayAdapter 339
  - getWidth method of class Display 312
  - getWritableDatabase method of class
    - SQLiteOpenHelper **285**
  - getX method of class MotionEvent 253
  - getY method of class MotionEvent 253
  - Global Address Lists look-up 7
  - Google APIs 5
  - Google Checkout 55
  - Google Maps xvi, **9**, 292
    - API 294
    - API key 294
    - obtain API key 294
    - satellite map 292
    - street map 292
  - Google Maps API 297, 299, 318
  - Google Maps web services xvi, xvii
  - GPS xiv
    - mock data for testing purposes 299
  - GPS data 297, 303
  - GPS Exchange Format **295**
  - GPS fix 297, 307
  - GPS satellite 295
  - GPS signal 295
    - acquired 295
  - GPS\_EVENT\_FIRST\_FIX constant of class
    - GpsStatus 307
  - GPSTLogger app for recording GPS data 295
  - GpsStatus class
    - GPS\_EVENT\_FIRST\_FIX 307
  - GpsStatus.Listener interface **297**, 304, 307
    - .gpx extension 295
  - GPX File
    - sending to an AVD 295
  - GPX file 295
  - GPX tab in the DDMS perspective 296
    - graphics xvi, 18
    - graphics APIs xvi
  - Gravity property of a component 101
  - gravity sensor 14, 228
  - green method of class Color **241**
  - gesture 19
  - GUI components
    - Button **122**
    - EditText **94**
    - ImageView **69**, 83
    - naming convention 94, 123, 153
    - programmatically create 122
    - ScrollView **121**
    - SeekBar 92, **94**
    - TextView **69**, **74**, 80, 94
    - ViewGroup **121**
  - GUI design 37
  - gyroscope sensor 14, 228
- ## H
- Handler class **152**
    - postDelayed method **152**
    - removeCallbacks method 354
  - handling user events xvi
  - hard buttons on an android device **25**
  - hardware accelerated graphics 208
  - hardware support 18
  - Hardware-accelerated 2D graphics 15
  - HashMap class **153**
  - hashtag **61**
  - hasNext method of class
    - JsonReader 450
  - head tracking **15**
  - Headset Profile (HSP) 15
  - height of a table row 94
  - hide the soft keyboard 137
  - hideSoftInputFromWindow
    - method of class
      - InputMethodManager 138
  - hiding an app's title bar 299
  - hint in an EditText 267
  - holographic theme **228**, 230
  - holographic UI 12, 14
  - Home screen widget 14
  - HTML5 8
  - HTTP Live Streaming (HLS) 14
  - HTTP progressive streaming 8
  - humor 66
- ## I
- i-Newswire 63
  - icon 42, 46, **47**
  - icon design firms
    - Aha-Soft 48
    - Androidicons 48
    - glyFX 48
    - Iconiza 48
  - Icon Design Guidelines* 40, 48
  - icon design services 48
  - id property of a layout or component **79**, 83
  - IDE (integrated development environment) xvi, xxiii, 19
  - image
    - downsample 326
  - image choosing Activity 346
  - images xvi
  - ImageView
    - android:adjustViewBounds attribute **157**
  - ImageView class **69**, 83
    - setImageResource method **219**
    - Src property **69**, **85**
  - implicit Intent **141**
  - import an existing project into
    - Eclipse 93, 119, 151, 178, 206
  - Import dialog 24, 93, 119, 151, 178, 206
  - in-app advertising **54**, **56**
  - In-app billing **57**
  - in-app billing 55, 57
    - security best practices 58
  - in-app purchase 58
  - incur data cost 303
  - inflate
    - XML layout dynamically 122
  - inflate a layout 136
  - inflate method of class
    - LayoutInflater **136**
  - inflate the GUI 190
  - inflating a GUI **111**
  - information hiding **22**
  - inheritance **22**, 94
  - InputMethodManager class 132
    - hideSoftInputFromWindow method 138

- InputStream class 152, 167, 175
    - setImageDrawable method 152, 167
  - InputStreamReader class 434, 449
  - insert method of class
    - ContentResolver 256
  - insert method of class
    - SQLiteDatabase 286
  - instance 21
  - instance variable 22
  - integrated development
    - environment (IDE) xvi, xxiii, 19
  - IntelliJ Idea xxix
    - enabling Android support xxix
  - intent xvi
  - Intent class 122, 132, 141
    - ACTION\_GET\_CONTENT constant 346
    - ACTION\_VIEW constant 141
    - broadcast 396
    - Bundle 275
    - createChooser method 346
    - explicit 141, 263, 275
    - extras 275
    - getData method 345
    - getExtras method 276, 282
    - implicit 141
    - MIME type 325
    - putExtra method 275
    - setType method 346
  - intent-filter element of the manifest file 143
  - IntentService class 455
  - interface 94
    - implementing methods in Java 114
    - tagging interface 363
  - Interfaces 208
    - AdapterView.OnItemClickListener 270
    - AdapterView.OnItemClickListener 274
    - AdapterView.OnItemLongClickListener 395
    - AnimatorListener 207
    - AnimatorUpdateListener 207
    - CompoundButton.OnCheckedChangeListener 297, 302, 309
    - DialogInterface.OnClick-Listener 132, 173
    - DialogInterface.OnMulti-ChoiceClickListener 173
    - Editable 107
  - Interfaces (cont.)
    - GestureDetector.OnDouble-TapListener 180, 186
    - GestureDetector.On-GestureListener 180
    - GpsStatus.Listener 297, 304, 307
    - List 153
    - LocationListener 297, 305, 306
    - Map 153
    - OnCompleteListener 385
    - OnItemLongClickListener 395
    - Runnable 152, 356
    - SeekBar.OnSeekBarChange-Listener 107, 241, 242
    - SensorEventListener 228, 237
    - Serializable 363, 367
    - SurfaceHolder.Callback 180, 190, 200
    - TabListener 413
    - TextWatcher 107
    - View.OnClickListener 132
  - internationalization 81
  - Internet calling 11
  - Internet enabled applications xvi
  - Internet Public Relations
    - InternetNewsBureau.com 63
    - PRX Builder 63
  - Internet public relations resources
    - ClickPress 63
    - Deitel Internet Public Relations Resource Center 62
    - i-Newswire 63
    - InternetNewsBureau.com 63
    - Marketwire 63
    - Mobility PR 63
    - openPR 63
    - PR Leap 63
    - Press Release Writing 63
    - PRLog 63
    - PRWeb 63
    - PRX Builder 63
  - Internet public relations resourcesPRWeb 63
  - Internet telephony 12
  - interpolator for an animation 207
  - invalidate method of class View 249
  - isAdded method of class Fragment 430
  - isRouteDisplayed method of class MapActivity 308
  - item element
    - android:icon attribute 269
    - android:orderInCategory attribute 269
    - android:title attribute 269
    - android:titleCondensed attribute 269
  - item element for defining a MenuItem in a resource file 268
  - item element in a string-array element 154
- ## J
- Jarsigner xxix, 49
  - Java xv, 5
  - Java code xxii
  - Java for Programmers, 2/e* xv
  - Java Fundamentals: Parts I and II* xv
  - Java How to Program* xv
  - Java perspective in Eclipse 23
  - Java SE 6 Software Development Kit xxii
  - java.io class 363
  - java.io package 152
  - java.util package 152
  - java.util.concurrent package 208
  - java.util.concurrent.atomic package 228
  - JavaScript Object Notation (JSON) 434
  - JSON (JavaScript Object Notation) 392, 434
  - JsonReader 396
  - JsonReader class 391, 396, 434, 438, 447
    - beginArray method 435, 440
    - beginObject method 435, 438, 440
    - getString method 436
    - hasNext method 450
    - nextName method 435, 438
    - nextString method 450
    - skipValue method 436, 440, 449, 450
- ## K
- kernel memory management boost 7
  - key/value pairs
    - persistent 132
  - key/value pairs associated with an app 122
  - keyboard 6
    - types 268

- keySet method of interface Map **134**
  - Keystore xxix, **49**
  - Khronos EGL APIs 18
  - Khronos EGL library 14
  - Khronos OpenGL ES interfaces 18
  - Khronos OpenGL ES API 14
- L**
- label 46
  - landscape mode 208
  - landscape orientation 395
  - large-screen device 12, 14
  - latitude 297, 306, 318
  - launcher icon 40
  - layer components 299
  - layout 17
  - Layout height** property of a component 101
  - Layout margin top** property of a component **83**
  - layout resource 262
  - Layout weight** property of a component 102
  - Layout width** property of a component 97, 127
    - match\_parent value 97, 127
  - layout XML file
    - create 130, 158
  - LayoutInflater class **122**, 132
    - inflate method **136**
  - LayoutParams class 311
  - Layouts
    - FrameLayout 182, **299**
  - layouts
    - LinearLayout **74**
      - main.xml **74**
    - TableLayout **94**, 123
  - license for Android 5
  - licensing policy 48
  - licensing service **48**
  - lifecycle methods 179
  - lifecycle methods of an app 106
  - light sensor 228
  - line thickness 181
  - linear acceleration sensor 228
  - LinearInterpolator class 221
  - LinearLayout **74**
  - LinearLayout class 266
  - Linux 19
  - List interface **153**
  - list item
    - touched 274
    - touched row ID 275
  - list method of class
    - AssetManager 163
  - list view icon 40
  - ListActivity class **263**, 266, 269, 332, 343, 344, 370
    - built-in ListView 270
    - custom GUI 266
    - custom layout 325, 329
    - getListView method **270**, 329
    - ListView android:id 329
    - setListAdapter method **271**
  - listen for location changes **304**
  - listening for touches xvi
  - ListFragment class 393, **394**, 415
  - ListView class **262**, 266, 269, 325, 332, 343
    - format of a list item 266
    - in a Fragment 394
    - item layout 328, 330
  - ListView performance 326
  - load a URL into a web browser 122
  - load method of class SoundPool **190**
  - loadAnimation method of class
    - AnimationUtils **152**, 163
  - localization 43, 81
    - numbering format specifiers in String resources 179
  - localize an app 95
  - location 292
    - bearing 292
    - direction 292
  - location animation 220
  - location changes
    - listen for **304**
  - Location class **297**, 305
    - bearingTo method **306**
    - distanceTo method **305**
    - getBearing method **306**
    - getLatitude method **306**
    - getLongitude method **306**
  - location provider 303
    - status **306**
  - location providers **297**
  - location-based services 18
  - LocationListener interface **297**, 305, 306
    - onLocationChanged method **306**
  - LocationManager class **297**, 304
    - getBestProvider method **304**
    - getLastKnownLocation method **304**
    - requestLocationUpdates method **304**
  - ToastCanvas method of class
    - SurfaceHolder **202**
  - Log class **152**, 167
    - e method **167**
  - LogCat** tab in the Android DDMS perspective **152**
  - Togcat tool **152**
  - logging exceptions **152**, 167
  - Long clickable** property of an EditText 102
  - long press 20
  - longitude 297, 306, 318
  - long-press touch event 180
- M**
- Mac OS X xxiii, 19
  - magnetic field sensor 228
  - main.xml **74**
    - contents explained 87
  - makeText method of class Toast **256**
  - Manage Applications** shortcut 11
  - managing persistent data xvi
  - manifest 298, 327
    - receiver element **396**
    - service element **396**
  - manifest editor 143
  - manifest element of the manifest file **142**
  - manifest file 14, 38, **45**, 46, 52, 142, 296
    - access a non-standard library 296
    - uses-library element **296**, 299
    - uses-permission element **296**, 299
  - manually add an attribute to the XML 100
  - manually perform an animation 180
  - map 292
    - geographic coordinates 317
    - move the center 306
    - orientation 292
    - pixel coordinates 317
    - zoom level 301
  - Map interface **153**
    - keySet method **134**
  - map tiles 294
  - MapActivity class **297**, 300
    - isRouteDisplayed method **308**
  - MapController class 301, 306
    - animateTo method **306**
    - getController method **301**
    - setZoom method **301**

- Maps API xvi
  - MapView class **297**, 300, 301, 311, 312
    - getProjection method **317**
    - setBuiltInZoomControls method **312**
    - setClickable method **312**
    - setEnabled method **312**
    - setLayoutParams method **312**
    - setSatellite method **312**
  - Market app 59
  - Market filters 39
  - marketing 16
  - Marketwire 63
  - mashup **9**
  - match\_parent value for the **Layout** width property of a component 97, 127
  - match\_parent value of the android:layout\_width attribute 88
  - match\_parent value of the **Layout** height property 127, 156
  - match\_parent value of the **Layout** width property 127, 156
  - MD5 fingerprint 294
  - media files 179
  - media framework 14
  - Media Transfer Protocol (MTP) 13
  - Media/Picture Transfer Protocol (MTP/PTP) 14
  - MediaController class **364**
  - MediaPlayer class **326**, **353**, **353**, **354**, **354**, 356
  - MediaPlayer.OnCompletionListener class **364**
  - MediaStore.Images.Thumbnails class **342**
  - menu xvi
    - Context** menu 41
    - Options** menu 41
  - Menu class **152**, 170
  - Menu Design Guidelines* 41
  - menu element **268**
  - menu icon 40
  - menu name xxii
  - menu resources **262**, 268
  - MenuInflater class **263**, **273**, 278
  - merchant account **44**
  - Messaging** 41
    - method **21**, 94
    - method call **22**
    - micro blogging 60, 61
    - microdegrees 306
    - MIME type 256
    - video/\* 376
  - MIME type of an Intent 325
  - Misc** section of the **Properties** window 100
  - MKMapView class 290
  - mobile advertising 54
  - mobile advertising network 56, 62
    - AdMob 56
    - AdWhirl 64
    - Pinch Media 56
  - mobile advertising networks
    - AdMob 63
    - AdMob (www.admob.com/) 63
    - AdWhirl 64
    - Decktrade 64
    - Flurry 64
    - Google AdSense for Mobile 64
    - Medialets 64
    - Nexage 64
    - Smaato 64
  - mobile app design resources
    - Android Developer Guide: Compatibility* 38
    - Android Developer Guide: Designing for Performance* 38
    - Android Developer Guide: Designing for Responsiveness* 38
    - Android Developer Guide: Designing for Seamlessness* 38
    - Android Developer Guide: Supporting Multiple Screens* 38
    - Android Developer Guide: User Interface Guidelines* 38
  - mobile payment provider 58
    - Boku 59
    - PayPal Mobile Libraries 58
    - Zong 59
  - mobile payment providers 58
  - mock GPS data for testing purposes 299
  - modal dialog **122**
  - MODE\_PRIVATE constant **134**
  - MODE\_WORLD\_READABLE constant **134**
  - MODE\_WORLD\_WRITABLE constant **134**
  - monetizing apps 36, 56, 63
  - MotionEvent class **179**, 185, **229**, 251
    - getActionIndex method **252**
    - getActionMasked method **251**
    - getPointerCount method **253**
    - getX method 253
    - getY method 253
  - MOTODEV Studio for Android xxix
    - move the center of the map 306
  - moveTo method of class Path **252**
  - moveToFirst method of class Cursor **278**, 278
  - MP3 12
  - MP3 player 6
  - Multicore processor architecture 14
  - multimedia xvi
  - multimedia playlist 14
  - multiple format specifiers 179, 182
  - multiselect 14
  - multitasking 13
  - multitouch screen **6**
  - music audio stream 180, 190
  - music library 320
- ## N
- naming convention
    - GUI components 94, 123, 153
  - Native Asset Manager API 14
  - native audio 14
  - native graphics management 14
  - native input 14
  - NAVIGATION\_MODE\_TABS **413**
  - near-field communication (NFC) 11
    - API 12
    - device 11, 12
    - tag 11, 12
  - nested structure of a layout 96
  - nested Views **121**
  - network access 17
  - networking 18
  - New Android Project** dialog **71**
  - New Android XML File** dialog 124, 154
  - newsgroups 3
    - Android Developers 4
  - newTab method **413**
  - newWakeLock method of class PowerManager **305**
  - nextName method of class JsonReader 435, 438
  - nextString method of class JsonReader 450
  - NO\_REQUIREMENT constant of class Criteria 303
  - non-standard library
    - access 296
  - notification 14
  - notifyDataSetChanged method **326**, 336, 341

- notifyDataSetChanged method of class ArrayAdapter **326**, 336, 341
  - NotSerializableException class **363**
  - number picker 14
  - numbering format specifiers 179, 182
- O**
- obfuscate 49
  - object 20, 94
  - object (or instance) 22
  - object graph **363**
  - object-oriented analysis and design (OOAD) **22**
  - object-oriented language **22**
  - object-oriented programming 94
  - object-oriented programming (OOP) **23**
  - object serialization xvi, 361, **363**, 370
  - ObjectAnimator class **207**
  - ObjectInputStream class **364**, 372
  - ObjectInputStream class
    - readObject method **364**, 372
    - writeObject method 373
  - Objective-C command xxii
  - ObjectOutputStream class 373
  - ObjectOutputStream class **363**
    - writeObject method **363**
  - obtaining a Google Maps API key 294
  - OEM original equipment manufacturer 5
  - onActivityCreated method of class Fragment 416
  - onActivityCreated method of class Fragment 429
  - onActivityResult method **334**, 336, 341, 345
  - onActivityResult method of class Activity **334**, 336, 341, 345, 375
  - onAnimationCancel method of interface AnimationListener 216
  - onAnimationEnd method of interface AnimatorListener 216, **221**
  - onAnimationStart method of interface AnimatorListener **221**
  - onCheckedChanged method of interface OnCheckedChangeListener **310**
  - OnCheckedChangeListener interface 310
  - OnCheckedChangeListener interface 310
    - onCheckedChanged method **310**
  - OnClickListener interface 208
  - onCompletion method of interface OnCompletionListener **385**
  - OnCompletionListener interface **385**
    - onCreate method of class Activity **108**, 184, 212, 270, 276, 333, 344, 352, 371, 401
  - onCreate method of class SQLiteOpenHelper **289**
  - onCreateOptionsMenu method of class Activity **152**, 239, 278, 334
  - onCreateView method of class Fragment 424, 428
  - onDestroy method of class Activity **179**, 184, 185, 354
  - onDoubleTap method of class GestureDetector.SimpleOnGestureListener **186**
  - onDowngrade method of class SQLiteOpenHelper **289**
  - onDraw method of class View **250**
  - onItemClick method of interface AdapterView.OnItemClickListener **274**
  - OnItemClickListener interface 270
  - OnItemClickListener interface 274
    - onItemClick method **274**
  - OnItemLongClickListener interface 395
  - onLocationChanged method of interface LocationListener **306**
  - on-off* state 297
  - on-off state button 292
  - onOptionsItemSelected method of class Activity **152**, 241, 279, 334
  - onPause method of class Activity **179**, 184, 185, 354
  - onPictureTaken method of class Camera.PictureCallback **382**
  - onPostExecute method **272**, 273, 277, 278
  - onPostExecute method of class AsyncTask **272**, 273, 277, 278
  - onProgressUpdate method **272**, 277
  - onProgressUpdate method of class AsyncTask **272**, 277
  - onRestoreInstanceState method of class Activity 402
  - onResume method of class Activity **212**, 271, 354, 403
  - onSaveInstanceState method of class Activity **108**, 113, 351, 355, 402
  - on-screen component xxii
  - onSensorChanged method of interface SensorEventListener **237**
  - onSizeChanged method of class View **190**, 215, 249
  - onStart method of class Activity 212, 354
  - onStop method of class Activity **272**, 305, 354
  - onTouchEvent method of class Activity **179**, 185
  - OnTouchEvent method of class View **229**, 250
  - onTouchEvent method of class View 221
  - onUpgrade method of class SQLiteOpenHelper **289**
  - OOAD (object-oriented analysis and design) 22
  - OOP (object-oriented programming) **23**
  - Open Handset Alliance 4
  - open method of class Camera **381**
  - open source 4
  - open source apps 5
  - Open Source Project discussion groups 4
  - OpenCORE 8
  - OpenGL 18
  - OpenGL ES xvi
  - OpenGL renderer 15
  - opening a database 285
  - openPR 63
  - operating system 4
  - operating system requirements xxii
  - operating systems services 17
  - Options menu 41
  - orientation
    - landscape 395
    - portrait 395

original equipment manufacturer (OEM) 5

Outline window 95, 96, 126

Outline window in Eclipse 93, 94

out-of-memory error 326

OutputStream class 363

OutputStream class 229, 256

Overlay class 302, 314, 316

draw method **316**

Override/Implement Methods

option in Eclipse **113**

overscroll **262**

## P

package 17

Package Explorer window 73, 78, 93, 119, 151, 178, 206

Packages

android.animation **207**

android.app 17, 106, 132, 228, 394, 395

android.appwidget 18, **395**

android.content 18, **122**, 132, 229, **395**, 396

android.content.res 17, 152, **163**

android.database 18, **263**

android.database.sqlite 18, **263**

android.graphics 18, 181, 229

android.graphics.drawable 18, 152, **327**

android.hardware 18, **228**

android.location 18, **297**

android.media 18, 180, **326**

android.net 17, 132

android.opengl 18

android.os 17, 106, 152, 297

android.provider 18, **325**

android.speech 18

android.speech.tts 18

android.telephony 18

android.text 17, 107

android.util 18, 152, 189, 396

android.view 17, 132, 152, 179, 180

android.view.animation 152

android.view.inputmethod **132**

android.widget 17, **75**, 107, 132, 229, 297, 299

com.google.android.maps 18, **297**

Packages (cont.)

java.io 17, 152, 363

java.net 18

java.nio 18

java.util 17, 152

java.util.concurrent **208**

java.util.concurrent.atomic **228**

javax.microedition.khronos.egl 18

javax.microedition.khronos.opengl 18

javax.xml.parsers 18

org.xml.sax 18

Padding bottom property of a

component 101, 102

padding element of a shape **265**

Padding left property of a

component 101, 102

Padding property of a layout **97**

Padding right property of a

component 101, 102

Paint class **181**, 315

filled shape with a border **248**  
filled shape without a border **248**

line **248**

setAntiAlias method **248**

setStrokeCap method **245**, 248

setStrokeWidth method **248**

styles **248**

parse method of class Uri **141**

PARTIAL\_WAKE\_LOCK constant of class PowerManager 305

password 7

password expiration 15

Path class **229**

moveTo method **252**

quadTo method **253**

reset method **252**

pause method **354**

pause method of class

MediaPlayer **354**

payment 45

payment processor 55

PendingIntent class **395**, 456

performance 38

permissions **296**

android.permission.ACCESS\_FINE\_LOCATION **299**

android.permission.ACCESS\_MOCK\_LOCATION **299**

android.permission.INTERNET **299**

permissions (cont.)

android.permission.WAKE\_LOCK **299**

requesting 299

persistent key/value pairs 132

photo xvi

photo preview 379

photo sharing 60

Photo Transfer Protocol (PTP) 13

photos 6

picture

take programmatically 364

pinch 20

Pinch Media 56

piracy 49

pixel coordinates 317

Pizza Ordering app xvi

play method of class SoundPool **195**

portrait mode 174, 190

portrait orientation 9, 395

postDelayed method of class Handler **152**

power 303

power level 305, **305**

power management 11

power state 297

POWER\_HIGH constant of class Criteria 303

POWER\_LOW constant of class Criteria 303

POWER\_MEDIUM constant of class Criteria 303

PowerManager class **297**, 305  
newWakelock method **305**

PARTIAL\_WAKE\_LOCK 305

PR Leap 63

predefined color 127

predefined color transparent 130

prepare method **353**

prepare method of class

MediaPlayer **353**

prepareAsync method **354**

prepareAsync method of class

MediaPlayer **354**

*Preparing to Publish: A Checklist* 46

press release writing 63

pressure sensor 228

prevent the soft keyboard from

displaying when app loads 123

price 16, 54, 55

privacy 7

private key **49**

PRLog 63

processing XML documents 18

programmatically create GUI components 122  
 progress bar 42, 43  
**Progress** property of a SeekBar **101**, 102  
 ProGuard **49**  
 project **71**  
 project, add a class 183  
 Projection class **317**  
   toPixels method **317**  
**Properties** tab in Eclipse 69  
**Properties** window 79, 80, 81  
   **Misc** section 100  
 property animation 159, **207**  
 property-animation lifecycle events 207  
 proximity sensor 228  
 public relations 62  
 publish a new version of an app 59  
 publishing data on an Android device 18  
 put method of class  
   ContentValues 256  
 putExtra method of class Intent **275**  
 putString method of class  
   SharedPreferences.Editor **136**

**Q**

quadratic bezier curve 253  
 quadTo method of class Path **253**  
 query method of class  
   SQLiteDatabase **288**  
**Queue** **208**  
**Queue** class **208**  
**Quick Contact** 7

**R**

R class **110**  
 R.color class 127  
 R.drawable class **110**  
 R.id class **110**, **111**  
 R.layout class **110**  
 R.layout.main constant **110**, 133  
 R.string class **110**  
 readObject method of class  
   ObjectInputStream **364**, 372  
 receiver element of the manifest **396**  
**Recent Apps** 13  
 reconfigure the camera 380  
 record GPS data 295  
 red method of class Color **241**  
 redraw a View 250

**Reference Chooser** dialog 85  
 registerListener method of class  
   SensorManager **236**  
 RelativeLayout.LayoutParams  
   class **219**  
 release method **354**  
 release method of class  
   MediaPlayer **354**  
 release method of class  
   SoundPool **200**  
 release method of class WakeLock **305**  
 release resources 278  
 Remote Wipe 7  
 RemoteViews class 438, 453, 456,  
   458  
 remove apps from Market 59  
 remove method of class  
   ArrayAdapter **348**  
 removeAllViews method of class  
   ViewGroup **137**, **216**  
 removeCallbacks method of class  
   Handler 354  
 rendering and tracking text 17  
 Renderscript 3D graphics engine 14  
 replace method of class  
   FragmentManager **408**  
 reporting bugs 4  
 requery method of class Cursor **273**  
 request code 334  
 requesting app permissions 299  
 requestLocationUpdates method  
   of class LocationManager **304**  
 requirements **22**  
 res folder of an Android project **74**,  
   **81**  
 res/anim folder 159  
 res/drawable folder 265  
 res/menu folder 268  
 res/raw folder of an Android  
   project **179**, 183  
 res/values folder 124  
 reset method **356**  
 reset method of class  
   MediaPlayer **356**  
 reset method of class Path **252**  
 resource 52  
**Resource Chooser** dialog 81  
**Resource** class 179  
   getString method 179  
 resource file  
   color element **125**  
   create 124, 154  
   dimen element **125**  
   resources element **125**

resource files **73**  
   arrays.xml 398  
   colors.xml 124, 154  
   dimen.xml 124, 154  
   item element for defining a  
     MenuItem **268**  
 resource value  
   access 129  
   access a color resource@color/  
     129  
   access a dimension  
     resource@dimen/ 130  
   access a string  
     resource@string/ 129  
 resources  
   android-developers.  
     blogspot.com 33  
   answers.oreilly.com/  
     topic/862-ten-tips-for-  
     android-application-  
     development/ 33  
   code.google.com/p/apps-  
     for-android/ 33  
   developer.htc.com/ 33  
   developer.motorola.com/ 33  
   developer.sprint.com/  
     site/global/develop/  
     mobile\_platforms/  
     android/android.jsp 33  
   developer.t-mobile.com/  
     site/global/resources/  
     partner\_hubs/android/  
     p\_android.jsp 33  
   dynamically load 163  
   layout 262  
   menu **262**, 268  
   style **262**, 264  
   www.androidtapp.com/10-  
     user-experience-tips-  
     for-successful-android-  
     apps/ 33  
   www.brighthub.com/mobile/  
     google-android.aspx 33  
   www.droidnova.com/ 33  
   www.ibm.com/  
     developerworks/  
     opensource/library/x-  
     android/index.html 33  
   www.youtube.com/user/  
     androiddevelopers 33  
**Resources** class **163**, 214  
   getString method 218  
   getStringArray method **406**  
 resources element of a resource file **125**  
 responsiveness 38

- RESULT\_CANCELED constant in class Activity 336
  - RESULT\_OK constant in class Activity 336
  - returning false from an event handler **186**
  - returning true from an event handler **186**
  - reusable software components 20
  - reuse 21
  - revenue share 16
  - reverse engineering 49
  - RGB color scheme 28
  - RGB values **79**
  - right align text 101
  - right value of the **Gravity** property of a component 101
  - rotate a Canvas 314
  - rotate animation for a View **159**
  - rotation animation 220
  - rotation vector sensor 228
  - Route Tracker** app xvi, 19, 20
  - row ID of a touched list item 275
  - Runnable interface 152, 199, 356
  - runOnUiThread method of class Activity **199, 372, 373**
- S**
- satellite map 292, 295, 302
  - save data on the iPhone 118
  - saved state 110
  - SAX API (Simple API for XML) 18
  - scale animation 220
  - scale animation for a View **159**
  - scale-independent pixel 125
  - scaled pixels (sp) 82
  - scale-independent pixels 82, **83**
  - scaleX method of class ViewPropertyAnimator **220**
  - scaleY method of class ViewPropertyAnimator **220**
  - screen capture 50
  - screen density 40
  - screen dimensions 312
  - screen orientation 42
    - portrait 181
  - screen size 40
  - screenshot specifications 50
  - scroll 6
  - scroll touch event 180
  - scrollable list of items 263
  - scrolling GUI components 119
  - ScrollView class **121, 266**
  - SD card 43, 371
  - seamlessness 38
  - security 7
  - SeekBar class 92, **94, 107**
    - android:thumb attribute 231
    - Progress** property **101, 102**
  - SeekBar.OnSeekBarChangeListener interface 107, 241, 242
  - seekTo method **354**
  - seekTo method of class MediaPlayer **354**
  - select multiple components 99
  - select videos 361
  - selectTab method of class ActionBar **403**
  - sell your app xv
  - send a message to an object 22
  - sendBroadcast method of class Activity **405**
  - sending GPS data to an AVD 295
  - Sensor class **228**
  - sensor events 14
  - SENSOR\_DELAY\_NORMAL constant of class SensorManager 236, 237
  - SENSOR\_SERVICE constant in class Context 237
  - Sensor.TYPE\_ACCELEROMETER constant 236, 237
  - SensorEvent class **239**
  - SensorEventListener interface **228, 237**
  - SensorEventListener listener **237**
  - SensorManager class **228, 235, 236**
    - getDefaultSensor method **236**
    - registerListener method **236**
  - SensorManager.SENSOR\_DELAY\_NORMAL constant 236, 237
  - sensors
    - accelerometer **228, 237**
    - gravity 228
    - gyroscope 228
    - light 228
    - linear acceleration 228
    - magnetic field 228
    - pressure 228
    - proximity 228
    - rotation vector 228
    - temperature 228
  - Serializable interface **363, 367**
  - serialization **363**
  - serialized object 363
  - service element of the manifest **396**
  - service to inflate a layout 136
  - Session Initiation Protocol (SIP) 11
    - SIP providers 11
  - set in an animation **159**
  - Set interface
    - toArray method **134**
  - setAccuracy method of class Criteria **303**
  - setAltitudeRequired method of class Criteria **303**
  - setAntiAlias method of class Paint **248**
  - setArguments method of class Fragment 426, 442
  - setBackgroundDrawable method **242**
  - setBackgroundColor method of class View **242**
  - setBuiltInZoomControls method of class MapView **312**
  - setClickable method of class MapView **312**
  - setColorEffect method of class Camera.Parameters **380, 382**
  - setCompoundDrawables method of class TextView 417
  - setContentView method of class Activity **110, 270**
  - setContentView method of class Dialog **241, 244**
  - setCostAllowed method of class Criteria **303**
  - setDataSource method **353**
  - setDataSource method of class MediaPlayer **353**
  - setDuration method of class ViewPropertyAnimator **220**
  - setEnabled method of class MapView **312**
  - setGravity method of class Toast **256**
  - setImageBitmap method of class View **245**
  - setImageDrawable method of class InputStream **152, 167**
  - setImageResource method of class ImageView **219**
  - setItems method of class AlertDialog.Builder **173**
  - setLayoutParams method of class MapView **312**
  - setLayoutParams method of class View **219**
  - setListAdapter method of class ListActivity **271**
  - setListener method of class ViewPropertyAnimator **220**

- setLooping method **354**
- setLooping method of class MediaPlayer **354**
- setMultiChoiceItems method of class AlertDialog, Builder **173**
- setParameters method of class Camera **380**, 382
- setPowerRequirement method of class Criteria **303**
- setPreviewDisplay method of class Camera **382**
- setPreviewSize method of class Camera.Parameters **382**
- setRepeatCount method of class Animation **152**, 163
- setResult method of class Activity **382**
- setSatellite method of class MapView **312**
- setStrokeCap method of class Paint **245**, 248
- setStrokeWidth method of class Paint 248
- setStyle method of class Paint **248**
- setTag method 339, 341
- setTag method of class View **326**, 339, 341
- Setting hardware emulation options xxvii, 32
- setType method of class Intent **346**
- setVolumeControlStream method of class Activity **180**
- setZoom method of class MapController **301**
- shape element **265**
- shared storage 324
- SharedPreferences class **122**, 132, 133, 403, 404, 405
  - apply method **405**
  - default file 212
  - edit method **136**
  - getAll method **134**
  - getInt method 214
  - getString method **135**
- SharedPreferences.Editor class **122**, 136
  - apply method **136**
  - clear method 140
  - putString method **136**
- Shop4Apps 54
- show method of class Dialog **241**
- shuffle a collection 167
- shuffle method of class Collections **152**
- shut down activity 185
- signing apps 46
- Simple API for XML (SAX API) 18
- simple collision detection 194
- simple touch events 179
- SimpleCursorAdapter class **271**, 325
- SimpleOnGestureListener interface 186
- skipValue method of class JsonReader 436, 440, 449, 450
- slider 94
- Slideshow app xvi
- social bookmarking 60
- social commerce 60
- social media 59
- social media sites
  - Bebo 60
  - Blogger 60
  - Delicious 60
  - Digg 60
  - Facebook 60
  - Flickr 60
  - Foursquare 60
  - Gowalla 60
  - Groupon 60
  - LinkedIn 60
  - Squidoo 60
  - StumbleUpon 60
  - Tip'd 60
  - Twitter 60
  - Wordpress 60
  - YouTube ([www.youtube.com](http://www.youtube.com)) 60
- social networking 60
- social news 60
- soft buttons on an Android device **25**
- soft keyboard
  - prevent from displaying when app loads 123
  - types 268
- sort (case insensitive) 134
- sort method of class Arrays **134**, **135**
- sound effects 180
- sound files 183
- sound quality 190
- Sound Recorder app 322
- SoundPool class **180**, 190
  - load method **190**
  - play method **195**
  - release method **200**
- sounds 179
- source code 2
- source-code listing 2
- sp (scaled pixels) 82
- spanning multiple columns in a TableLayout 99
- specify a component's column number in a TableLayout 100
- Speech APIs xvi
- speech recognition xvi, 18, 43
- speech synthesis xvi, 43
- speech-to-text input 43
- speed 297
- splash screen **43**
- Spot-On Game app xvi
- SQLite 18, 263
- SQLite 3 xvi
- SQLite 3 Resource Center ([www.deitel.com/SQLite3/](http://www.deitel.com/SQLite3/)) xviii, 32
- SQLiteDatabase class **263**
  - delete method **288**
  - execSQL method **289**
  - insert method **286**
  - query method **288**
  - update method **287**
- SQLiteOpenHelper class **263**, 285, 288
  - getWritableDatabase method **285**
  - onCreate method **289**
  - onDowngrade method **289**
  - onUpgrade method **289**
- SQLiteOpenHelper>default para font> class
  - close method **285**
- Src property of an ImageView 69, **85**
- stack components 299
- Stagefright **8**
- standard icons 269, 327
- star ratings for apps 59
- start method **354**
- start method of class MediaPlayer **354**
- startActivity method of class Activity 325
- startActivity method of class Context **122**, 141
- startActivityForResult method of class Activity **325**, 334, 340, 345
- startAnimation method of class View **152**
- startManagingCursor method of class Activity **273**

- startPreview method of class
    - Camera **382**
  - startTransition method of class
    - TransitionDrawable **358**
  - status bar icon 40
  - stopPreview method of class
    - Camera **382**
  - Storage Manager API 14
  - stream for playing music 190
  - streaming 17
  - streaming media 354
  - street map 292, 295, 302
  - Stretch columns** property of a
    - TableLayout **97**, 127
  - String arrays
    - defining in strings.xml 154
  - String class
    - format method 179
  - String resource
    - containing multiple format specifiers 179, 182
  - String resources 230
  - String.CASE\_INSENSITIVE\_ORDER Comparator<String> object 134
  - string-array element
    - item element **154**
  - string-array element in strings.xml **154**
  - strings.xml **81**, 95, 126, 182
    - defining String arrays 154
    - numbering format specifiers for localization 179
    - string-array element **154**
  - stroke element of a shape **265**
  - style attribute of a GUI component **262**, 266
  - style element 264
  - style resources **262**, 264
    - item element 264
    - name attribute 264
  - subclass 94
  - supporting multiple screens 38
  - supports-screens element 40
  - surfaceChanged method of interface
    - SurfaceHolder.Callback **200**
  - surfaceCreated method of class
    - SurfaceHolder.Callback **381**, **382**
  - surfaceCreated method of interface
    - SurfaceHolder.Callback **200**
  - surfaceDestroyed method of class
    - SurfaceHolder.Callback **200**
  - SurfaceHolder class **180**, 190, 378, 379
    - addCallback method **190**
    - lockCanvas method **202**
  - SurfaceHolder.Callback class
    - 379, 380
    - surfaceCreated method **381**, **382**
    - surfaceDestroyed method **381**
  - SurfaceHolder.Callback interface **180**, 190, 200
    - surfaceChanged method **200**
    - surfaceCreated method **200**
    - surfaceDestroyed method **200**
  - SurfaceView class **180**, 190, 364, 366, 378, 379
    - getHolder method **190**
  - swipe 6, 20
  - synchronized 202
  - syntax shading xvii, 2
  - System Bar 12, 13
- ## T
- Tab class **413**
  - tab icon 40
  - tabbed navigation 392
    - selected tab 402
  - TableLayout class **94**, 123, 266
    - android:stretchColumns attribute **97**, 123, 129
    - columns numbered from 0 100
    - documentation
      - (developer.android.com/reference/android/widget/TableLayout.html) 94
    - spanning multiple columns 99
    - specify a component's column number 100
  - TableRow class **94**
    - documentation
      - (developer.android.com/reference/android/widget/TableRow.html) 94
  - tablet 12
  - TabListener interface **413**
  - tagging interface **363**
  - take a picture 361, 364
  - tap 20
  - target SDK version 208, 230, 396
  - tasks **41**
  - telephony xiv, xvi
  - temperature sensor 228
  - testing 47
  - tethering 8
  - text alignment
    - right 101
  - text box 94
  - Text color** property of a component **83**
  - text field 94
  - Text** property of a component 69, **81**
  - Text size** property of a component **83**
  - Text size** property of an EditText 100
  - Text style** property of a component **83**
  - text-to-speech 18, 43
  - Text-to-Speech (TTS) **43**
  - TextView class **69**, 80, 94, 107
    - setCompoundDrawables method 417
    - Text property 69
  - TextView component **74**
  - TextWatcher interface 107
  - thread
    - UI 372
  - thread (for animation) 180
  - Thread class 201
  - ThreadR class 180
  - thread-safe list 208
  - thumbnail utility 9
  - TimeInterpolator class 220
  - time-lapse video recording 13
  - Tip Calculator** app xv, 20, 92
  - title bar
    - hiding 299
  - toArray method of interface Set **134**
  - Toast class **229**, 256, 295, 372
    - makeText method **256**
    - setGravity method **256**
  - ToggleButton class 292, **297**
    - android:textOff attribute **299**
  - Tools
    - logcat **152**
  - toPixels method of class
    - Projection **317**
  - touch event 229, **250**
  - touch events 379
    - double tap 180, 186
    - fling 180
    - long press 180
    - scroll 180
    - simple 179

touched list item 274  
 track app installs 59  
 transaction fee 45  
 transferring music and photos to an AVD 323  
 transferring data via Bluetooth xvi  
 TRANSIT\_FRAGMENT\_FADE **408**  
 TransitionDrawable class **327**, 358  
   startTransition method **358**  
 translate animation  
   android:duration attribute **159**  
   android:fromXDelta attribute **159**  
   android:startOffset attribute **159**  
   android:toXDelta attribute **159**  
 translate animation for a View **159**  
 translation animation 220  
 transparency 80, 228  
 transparent predefined Android color **130**  
 transparent predefined color 130  
 TTS (Text-to-Speech) **43**  
 turn off logging and debugging 46  
 tweened animation **159**  
 tweening 207  
 tweet **61**  
 Twitter xviii, **61**  
   @deitel 33, 61, 459  
   #AndroidFP 61  
   hashtag **61**  
   search 118  
   search operators 118  
   tweet **61**  
 TYPE\_ACCELEROMETER constant of class Sensor 236, 237

## U

UI thread 372  
 unique row ID of a touched list item 275  
 update method of class SQLiteDatabase **287**  
 upgrading a database 285  
 Uri class 132, **141**  
   parse method **141**  
**USB Debugging** 47  
**USB debugging** 31  
 User Interface Guidelines 36  
 uses-library element of manifest **296**, 299

uses-permission element of manifest **296**, 299  
 uses-sdk element of the manifest file **142**  
 using an external library 299  
 utilities 37

## V

ValueAnimator class **207**  
 values folder of an Android project **81**  
 version code 48  
 version name 48  
 versioning your app 46  
*Versioning Your Applications* 48  
 vertically center text 101  
 video xiv, xvi, 6, 18  
 video driver 14  
 video format 12  
 video playback 8  
 video sharing 60  
 VideoView class 361, 363, **364**, 384  
 view xvi, 107  
 View animation 207  
 View animations **159**  
 View class 180, **242**, 339, 340, 341  
   animate method **207**, 220  
   custom subclass 186  
   dispatchDraw method **313**  
   getTag method **326**  
   invalidate method **249**  
   onDraw method **250**  
   onSizeChanged method **190**, 215, 249  
   OnTouchEvent method **229**, 250  
   onTouchEvent method 221  
   redraw a View 250  
   setImageBitmap method **245**  
   setLayoutParams method **219**  
   setTag method **326**  
   size changes 190  
   startAnimation method **152**  
 View hierarchy 215  
 View.OnClickListener interface 132  
 ViewGroup class **121**  
   removeAllViews method **137**, 216  
 View-Holder Pattern 336  
 view-holder pattern **326**  
 ViewPropertyAnimator class **207**, 220  
   scaleX method **220**  
   scaleY method **220**

ViewPropertyAnimator class (cont.)  
   setDuration method **220**  
   setListener method **220**  
   x method **220**  
   y method **220**  
 viral marketing 59, 60  
 viral video 61  
 virtual camera operator **15**  
 virtual goods 57  
 Visual Layout Editor 69, **74**, 93, 94  
   select multiple components 99  
**Voice Recorder** app xvi  
 volume 180  
 VP8 open video compression 12

## W

WakeLock class **297**, 305  
   acquire method **305**  
   release method **305**  
**Weather AppWidget** 40  
**Weather Viewer** app xvi  
 WeatherBug web services xvii  
 web services xvi, xvii, **9**, 434  
   Amazon eCommerce 10  
   eBay 10  
   Facebook 10  
   Flickr 10  
   Foursquare 10  
   Google Maps 10  
   Groupon 10  
   Last.fm 10  
   LinkedIn 10  
   Microsoft Bing 10  
   Netflix 10  
   PayPal 10  
   Salesforce.com 10  
   Skype 10  
   Twitter 10  
   WeatherBug 10  
   Wikipedia 10  
   Yahoo Search 10  
   YouTube 10  
   Zillow 10  
 web services directories  
   APIfinder 10  
   Google Code API Directory 10  
   ProgrammableWeb 10  
   Webapi.org 10  
   Webmashup.com 10  
 WebM open container format 12  
**Welcome** app xv, 19  
**Welcome** tab **23**  
**Welcome** tab in Eclipse **70**

- widget xiv, xvi, 17, 107, 132
    - Android Agenda Widget 41
    - ESPN ScoreCenter 40
  - Widget Design Guidelines* 40
  - WIDGET\_UPDATE\_BROADCAST\_ACTION **405**
  - widgets 40
    - App Protector Pro 41
    - BatteryLife 41
    - Difficult Logic Riddles Pro 41
    - ecoTips 41
    - ESPN ScoreCenter 40
    - Favorite Quotes 41
    - Pandora Radio 40
    - Shazam Encore 41
    - Stock Alert 41
    - System Info Widget 41
    - The Coupons App 41
    - Twidroyd PRO 41
    - Weather & Toggle Widget 41
    - WeatherBug Elite 41
  - width of a column 94
  - Wi-Fi 8, 13
    - hotspot 8
  - Windows 19
  - word-of-mouth marketing 60
  - wrap\_content value of the
    - android:layout\_height attribute 88
    - android:layout\_width attribute 88
  - WRITE\_EXTERNAL\_STORAGE permission 365
  - writeObject method of class
    - ObjectInputStream 373
    - ObjectOutputStream **363**
  - www.deitel.com 32
  - www.deitel.com/training 459
  - www.housingmaps.com 9
- X**
- x method of class
    - ViewPropertyAnimator **220**
  - XML layout
    - inflate dynamically 122
  - XML markup of a GUI 94
  - XML utilities 18
- Y**
- y method of class
    - ViewPropertyAnimator **220**
- Z**
- zipalign **49**
  - zoom 6, 9
  - zoom level of a map 301