



Joseph Annuzzi, Jr.
Lauren Darcey
Shane Conder

Fourth Edition

Introduction to Android™ Application Development

Developer's Library



FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Praise for *Introduction to Android™ Application Development, Fourth Edition*

“Introduction to Android™ Application Development, Fourth Edition, is a phenomenal read and allows those interested in Android development to be brought up to speed and developing apps with minimal fuss. Everything from an introduction to the Android ecosystem and setting up a development environment to creating and publishing Android applications is covered in depth and with technical expertise. Those who crave even more from the book will be treated to a feast of useful links at the end of each chapter to help guide them on and expand their new-found knowledge base.”

—Philip Dutson, UX and mobile developer for ICON Health & Fitness

*“With this edition, you won’t find a more solid and comprehensive introduction to Android programming. Even if you already have another Android book, *Introduction to Android™ Application Development* makes a good second reference.”*

—Douglas Jones, senior software engineer, Fullpower Technologies

“Introduction to Android™ Application Development, Fourth Edition, is an important update to this invaluable reference for new and seasoned Android developers. It brings the latest up-to-date information about the newest releases of Android, showing you how to keep your application fresh on yesterday’s, today’s, and tomorrow’s Android devices.”

—Ray Rischpater, senior software engineer, Microsoft

This page intentionally left blank

Introduction to Android™ Application Development

Fourth Edition

Developer's Library Series



Visit developers-library.com for a complete list of available products

The **Developer's Library Series** from Addison-Wesley provides practicing programmers with unique, high-quality references and tutorials on the latest programming languages and technologies they use in their daily work. All books in the Developer's Library are written by expert technology practitioners who are exceptionally skilled at organizing and presenting information in a way that's useful for other programmers.

Developer's Library books cover a wide range of topics, from open-source programming languages and databases, Linux programming, Microsoft, and Java, to Web development, social networking platforms, Mac/iPhone programming, and Android programming.

PEARSON

Introduction to Android™ Application Development

Android Essentials

Fourth Edition

Joseph Anuzzi, Jr.
Lauren Darcey
Shane Conder

◆ Addison-Wesley

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

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.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Annuzzi, Joseph, Jr.

Introduction to Android application development : Android essentials / Joseph Annuzzi, Jr., Lauren Darcey, Shane Conder.—Fourth edition.

pages cm

Revised edition of first part of: Android wireless application development / Shane Conder, Lauren Darcey. c2010.

Includes bibliographical references and index.

ISBN-13: 978-0-321-94026-1 (alk. paper)

ISBN-10: 0-321-94026-1 (alk. paper)

1. Application software—Development. 2. Android (Electronic resource) 3. Mobile computing. 4. Wireless communication systems. I. Darcey, Lauren, 1977- II. Conder, Shane, 1975- III. Darcey, Lauren, 1977- Android wireless application development. IV. Title.

QA76.76.A65A56 2014

005.3—dc23

2013035917

Copyright © 2014 Joseph Annuzzi, Jr., Lauren Darcey, and Shane Conder

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.

Some figures that appear in this book have been reproduced from or are modifications based on work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License. See <https://developers.google.com/site-policies>.

Screenshots of Google products follow these guidelines:

<http://www.google.com/permissions/using-product-graphics.html>

The following are registered trademarks of Google:

Android™, Chromecast™, Google Play™, Google Wallet™, Glass™, Google+™, Nexus™, Google, and the Google logo are registered trademarks of Google Inc.

ISBN-13: 978-0-321-94026-1

ISBN-10: 0-321-94026-1

Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.

First printing: November, 2013

Editor-in-Chief

Mark L. Taub

Acquisitions Editor

Laura Lewin

Development Editor

Songlin Qiu

Managing Editor

John Fuller

Project Editor

Caroline Senay

Copy Editor

Barbara Wood

Indexer

Jack Lewis

Proofreader

Christine Clark

Technical Reviews

Douglas Jones

Ray Rischpater

Publishing Coordinator

Olivia Basegio

Compositor

Shepherd, Inc.



This book is dedicated to Cleopatra (Cleo).

—Joseph Annuzzi, Jr.

This book is dedicated to ESC.

—Lauren Darcey and Shane Conder



This page intentionally left blank

Contents at a Glance

Acknowledgments xxxi

About the Authors xxxiii

Introduction **1**

I: An Overview of the Android Platform

- 1** Introducing Android **11**
- 2** Setting Up Your Android Development Environment **37**
- 3** Writing Your First Android Application **55**

II: Android Application Basics

- 4** Understanding the Anatomy of an Android Application **97**
- 5** Defining Your Application Using the Android Manifest File **115**
- 6** Managing Application Resources **137**

III: Android User Interface Design Essentials

- 7** Exploring User Interface Building Blocks **177**
- 8** Designing with Layouts **209**
- 9** Partitioning the User Interface with Fragments **243**
- 10** Displaying Dialogs **265**

IV: Android Application Design Essentials

- 11** Using Android Preferences **281**
- 12** Working with Files and Directories **301**
- 13** Leveraging Content Providers **315**
- 14** Designing Compatible Applications **331**

V: Publishing and Distributing Android Applications

- 15** Learning the Android Software Development Process **355**
- 16** Designing and Developing Bulletproof Android Applications **379**
- 17** Planning the Android Application Experience **395**
- 18** Testing Android Applications **423**
- 19** Publishing Your Android Application **459**

VI: Appendixes

- A** Mastering the Android Development Tools **485**
- B** Quick-Start Guide: The Android Emulator **505**
- C** Quick-Start Guide: Android DDMS **529**
- D** Android IDE and Eclipse Tips and Tricks **547**
- E** Answers to Quiz Questions **559**
- Index** **567**

Contents

Acknowledgments xxxi

About the Authors xxxiii

Introduction 1

Who Should Read This Book 1

Key Questions Answered in This Book 2

How This Book Is Structured 2

An Overview of Changes in This Edition 3

Development Environments Used in This Book 5

Supplementary Materials Available 6

Where to Find More Information 6

Conventions Used in This Book 7

Contacting the Authors 7

I: An Overview of the Android Platform

1 Introducing Android 11

A Brief History of Mobile Software Development 11

Way Back When 11

“The Brick” 13

Wireless Application Protocol (WAP) 15

Proprietary Mobile Platforms 17

The Open Handset Alliance 18

Google Goes Wireless 18

Forming the Open Handset Alliance 19

Manufacturers: Designing Android Devices 19

Mobile Operators: Delivering the Android Experience 20

Apps Drive Device Sales: Developing Android Applications 21

Taking Advantage of All Android Has to Offer 22

The Android Marketplace: Where We Are Now 22

Android Platform Differences 23

Android: A Next-Generation Platform 23

Free and Open Source 25

Familiar and Inexpensive Development Tools 25

Reasonable Learning Curve for Developers	26
Enabling Development of Powerful Applications	26
Rich, Secure Application Integration	26
No Costly Obstacles for Development	27
A “Free Market” for Applications	27
A Growing Platform	28
The Android Platform	29
Android’s Underlying Architecture	29
Security and Permissions	31
Exploring Android Applications	32
Summary	36
Quiz Questions	36
Exercises	36
References and More Information	36
2 Setting Up Your Android Development Environment	37
Configuring Your Development Environment	37
Configuring Your Operating System for Device Debugging	39
Configuring Your Android Hardware for Debugging	39
Upgrading the Android SDK	42
Problems with the Android Software Development Kit	42
Exploring the Android SDK	43
Understanding the Android SDK License Agreement	43
Reading the Android SDK Documentation	44
Exploring the Core Android Application Framework	45
Exploring the Core Android Tools	47
Exploring the Android Sample Applications	51
Summary	52
Quiz Questions	52
Exercises	53
References and More Information	53

3 Writing Your First Android Application 55

- Testing Your Development Environment 55
 - Adding the Android Samples Using the SDK Manager 56
 - Adding the Snake Project to Your Android IDE Workspace 57
 - Creating an AVD for Your Snake Project 61
 - Creating a Launch Configuration for Your Snake Project 62
 - Running the Snake Application in the Android Emulator 66
- Building Your First Android Application 68
 - Creating and Configuring a New Android Project 69
 - Core Files and Directories of the Android Application 73
 - Creating an AVD for Your Project 75
 - Creating a Launch Configuration for Your Project 75
 - Running Your Android Application in the Emulator 76
 - Debugging Your Android Application in the Emulator 80
 - Adding Logging Support to Your Android Application 83
 - Adding Some Media Support to Your Application 84
 - Adding Location-Based Services to Your Application 88
 - Debugging Your Application on Hardware 90
- Summary 92
- Quiz Questions 93
- Exercises 94
- References and More Information 94

II: Android Application Basics

4 Understanding the Anatomy of an Android Application 97

- Mastering Important Android Terminology 97
- The Application Context 98
 - Retrieving the Application Context 98
 - Using the Application Context 98
- Performing Application Tasks with Activities 99
 - The Lifecycle of an Android Activity 100

Organizing <code>Activity</code> Components with Fragments	105
Managing <code>Activity</code> Transitions with Intents	106
Transitioning between Activities with Intents	106
Organizing Application Navigation with Activities and Intents	110
Working with Services	110
Receiving and Broadcasting Intents	111
Summary	112
Quiz Questions	112
Exercises	112
References and More Information	113
5 Defining Your Application Using the Android Manifest File	115
Configuring Android Applications Using the Android Manifest File	115
Editing the Android Manifest File	116
Managing Your Application's Identity	122
Versioning Your Application	122
Setting the Application Name and Icon	122
Enforcing Application System Requirements	123
Targeting Specific SDK Versions	123
Enforcing Application Platform Requirements	126
Working with External Libraries	128
Other Application Configuration Settings and Filters	129
Registering Activities in the Android Manifest	129
Designating a Primary Entry Point <code>Activity</code> for Your Application Using an Intent Filter	130
Configuring Other Intent Filters	130
Registering Other Application Components	131
Working with Permissions	131
Registering Permissions Your Application Requires	131
Registering Permissions Your Application Enforces	132

Exploring Other Manifest File Settings	133
Summary	133
Quiz Questions	134
Exercises	134
References and More Information	135
6 Managing Application Resources	137
What Are Resources?	137
Storing Application Resources	137
Resource Value Types	138
Accessing Resources Programmatically	142
Setting Simple Resource Values	
Using the Android IDE	143
Working with Different Types of Resources	146
Working with String Resources	147
Using String Resources as Format Strings	148
Working with Quantity Strings	149
Working with String Arrays	150
Working with Boolean Resources	151
Working with Integer Resources	152
Working with Colors	152
Working with Dimensions	153
Drawable Resources	154
Working with Images	156
Working with Color State Lists	158
Working with Animation	159
Working with Menus	162
Working with XML Files	163
Working with Raw Files	164
References to Resources	165
Working with Layouts	166
Referencing System Resources	171
Summary	172
Quiz Questions	173
Exercises	173
References and More Information	173

III: Android User Interface Design Essentials

- 7 Exploring User Interface Building Blocks 177**
 - Introducing Android Views and Layouts 177
 - The Android View 177
 - The Android Controls 177
 - The Android Layout 178
 - Displaying Text to Users with `TextView` 179
 - Configuring Layout and Sizing 179
 - Creating Contextual Links in Text 180
 - Retrieving Data from Users with Text Fields 183
 - Retrieving Text Input Using `EditText` Controls 183
 - Constraining User Input with Input Filters 184
 - Helping the User with Autocompletion 186
 - Giving Users Choices Using `Spinner` Controls 188
 - Allowing Simple User Selections with Buttons and Switches 190
 - Using Basic Buttons 190
 - Using `CheckBox` and `ToggleButton` Controls 193
 - Using `RadioGroup` and `RadioButton` 194
 - Retrieving Dates, Times, and Numbers from Users with Pickers 197
 - Using Indicators to Display Progress and Activity to Users 199
 - Indicating Progress with `ProgressBar` 199
 - Indicating Activity with Activity Bars and Activity Circles 202
 - Adjusting Progress with Seek Bars 202
 - Other Valuable User Interface Controls 203
 - Displaying Rating Data with `RatingBar` 204
 - Showing Time Passage with the `Chronometer` 205
 - Displaying the Time 206
 - Summary 207
 - Quiz Questions 207
 - Exercises 207
 - References and More Information 208

8 Designing with Layouts 209

- Creating User Interfaces in Android 209
 - Creating Layouts Using XML Resources 209
 - Creating Layouts Programmatically 211
- Organizing Your User Interface 214
 - Using `ViewGroup` Subclasses for Layout Design 214
 - Using `ViewGroup` Subclasses as `View` Containers 214
- Using Built-in Layout Classes 215
 - Using `LinearLayout` 217
 - Using `RelativeLayout` 219
 - Using `FrameLayout` 222
 - Using `TableLayout` 224
 - Using `GridLayout` 228
 - Using Multiple Layouts on a Screen 230
- Using Container Control Classes 232
 - Using Data-Driven Containers 233
 - Adding Scrolling Support 238
 - Exploring Other `View` Containers 239
- Summary 239
- Quiz Questions 239
- Exercises 240
- References and More Information 240

9 Partitioning the User Interface with Fragments 243

- Understanding Fragments 243
 - Understanding the `Fragment` Lifecycle 244
 - Working with Special Types of Fragments 247
 - Designing `Fragment`-Based Applications 248
- Using the Android Support Package 258
 - Adding `Fragment` Support to Legacy Applications 259
 - Using `Fragments` in New Applications Targeting Older Platforms 259
 - Linking the Android Support Package to Your Project 260
- Exploring Nested `Fragments` 261

Summary	261
Quiz Questions	262
Exercises	262
References and More Information	263

10 Displaying Dialogs 265

Choosing Your Dialog Implementation	265
Exploring the Different Types of Dialogs	266
Working with Dialogs and Dialog Fragments	267
Tracing the Lifecycle of a Dialog and DialogFragment	268
Working with Custom Dialogs	270
Working with Support Package Dialog Fragments	271
Summary	275
Quiz Questions	276
Exercises	276
References and More Information	276

IV: Android Application Design Essentials

11 Using Android Preferences 281

Working with Application Preferences	281
Determining When Preferences Are Appropriate	281
Storing Different Types of Preference Values	282
Creating Private Preferences for Use by a Single Activity	282
Creating Shared Preferences for Use by Multiple Activities	282
Searching and Reading Preferences	283
Adding, Updating, and Deleting Preferences	284
Reacting to Preference Changes	285
Finding Preferences Data on the Android File System	285
Creating Manageable User Preferences	286
Creating a Preference Resource File	287
Using the PreferenceActivity Class	289
Organizing Preferences with Headers	291
Learning about Cloud Save for Android Applications	296

Summary	298
Quiz Questions	298
Exercises	299
References and More Information	299

12 Working with Files and Directories 301

Working with Application Data on a Device	301
Practicing Good File Management	302
Understanding Android File Permissions	303
Working with Files and Directories	303
Exploring with the Android Application Directories	304
Working with Other Directories and Files on the Android File System	309
Summary	312
Quiz Questions	312
Exercises	313
References and More Information	313

13 Leveraging Content Providers 315

Exploring Android's Content Providers	315
Using the MediaStore Content Provider	316
Using the CallLog Content Provider	318
Using the Browser Content Provider	319
Using the CalendarContract Content Provider	321
Using the UserDictionary Content Provider	321
Using the VoicemailContract Content Provider	322
Using the Settings Content Provider	322
Introducing the ContactsContract Content Providers	322
Modifying Content Provider Data	324
Adding Records	325
Updating Records	326
Deleting Records	327
Using Third-Party Content Providers	328
Summary	328
Quiz Questions	328

Exercises 329
References and More Information 329

14 Designing Compatible Applications 331

Maximizing Application Compatibility 331
Designing User Interfaces for Compatibility 333
 Working with Fragments 335
 Leveraging the Android Support Library 335
 Supporting Specific Screen Types 335
 Working with Nine-Patch Stretchable Graphics 336
 Using the “Working Square” Principle 336
Providing Alternative Application Resources 338
 Understanding How Resources Are Resolved 338
 Organizing Alternative Resources with Qualifiers 339
 Providing Resources for Different Orientations 345
 Using Alternative Resources Programmatically 345
 Organizing Application Resources Efficiently 345
Targeting Tablets, TVs, and Other New Devices 347
 Targeting Tablet Devices 347
 Targeting Google TV Devices 348
Targeting Google Chromecast Devices 350
Summary 350
Quiz Questions 350
Exercises 351
References and More Information 351

V: Publishing and Distributing Android Applications

15 Learning the Android Software Development Process 355

An Overview of the Mobile Development Process 355
Choosing a Software Methodology 356
 Understanding the Dangers of Waterfall Approaches 356
 Understanding the Value of Iteration 357
Gathering Application Requirements 357
 Determining Project Requirements 357
 Developing Use Cases for Mobile Applications 360

Incorporating Third-Party Requirements and Recommendations	360
Managing a Device Database	361
Assessing Project Risks	364
Identifying Target Devices	364
Acquiring Target Devices	366
Determining the Feasibility of Application Requirements	366
Understanding Quality Assurance Risks	367
Writing Essential Project Documentation	368
Developing Test Plans for Quality Assurance Purposes	368
Providing Documentation Required by Third Parties	369
Providing Documentation for Maintenance and Porting	369
Leveraging Configuration Management Systems	369
Choosing a Source Control System	369
Implementing an Application Version System That Works	370
Designing Mobile Applications	370
Understanding Mobile Device Limitations	370
Exploring Common Mobile Application Architectures	371
Designing for Extensibility and Maintenance	371
Designing for Application Interoperability	372
Developing Mobile Applications	373
Testing Mobile Applications	373
Controlling the Test Release	374
Deploying Mobile Applications	374
Determining Target Markets	375
Supporting and Maintaining Mobile Applications	375
Track and Address Crashes Reported by Users	376
Testing Firmware Upgrades	376
Maintaining Adequate Application Documentation	376
Managing Live Server Changes	376
Identifying Low-Risk Porting Opportunities	376
Application Feature Selection	377

Summary	377
Quiz Questions	377
Exercises	377
References and More Information	378
16 Designing and Developing Bulletproof Android Applications	379
Best Practices in Designing Bulletproof Mobile Applications	379
Meeting Mobile Users' Demands	380
Designing User Interfaces for Mobile Devices	380
Designing Stable and Responsive Mobile Applications	381
Designing Secure Mobile Applications	383
Designing Mobile Applications for Maximum Profit	383
Following the Android Application Quality Guidelines	384
Leveraging Third-Party Quality Standards	385
Designing Mobile Applications for Ease of Maintenance and Upgrades	385
Leveraging Android Tools for Application Design	387
Avoiding Silly Mistakes in Android Application Design	388
Best Practices in Developing Bulletproof Mobile Applications	388
Designing a Development Process That Works for Mobile Development	389
Testing the Feasibility of Your Application Early and Often	389
Using Coding Standards, Reviews, and Unit Tests to Improve Code Quality	390
Handling Defects Occurring on a Single Device	392
Leveraging Android Tools for Development	393
Avoiding Silly Mistakes in Android Application Development	393
Summary	393
Quiz Questions	394
Exercises	394
References and More Information	394

17 Planning the Android Application Experience 395

Thinking about Objectives 395

User Objectives 396

Team Objectives 396

Objectives of Other Stakeholders 396

Techniques for Focusing Your Product Efforts 397

Personas 397

Entity Discovery and Organization 398

Use Cases and Use Case Diagrams 398

Charting Your Application's Navigation 400

Android Application Navigation Scenarios 400

Launching Tasks and Navigating the Back Stack 404

Navigating with Fragments 404

Planning Application Navigation 404

Android Navigation Design Patterns 405

Encouraging Action 408

Menus 410

Action Bars 411

Dialogs 415

Actions Originating from Your Application's Content 416

Communicating Your Application's Identity 416

Designing Screen Layouts 417

Sketches 417

Wireframes 417

Design Comps 417

Reacting Properly with Visual Feedback 418

Observing Target Users for Usability 418

Mocking Up the Application 418

Testing the Release Build 419

Summary 419

Quiz Questions 420

Exercises 420

References and More Information 420

18 Testing Android Applications 423

- Best Practices in Testing Mobile Applications 423
 - Designing a Mobile Application Defect-Tracking System 423
 - Managing the Testing Environment 425
 - Maximizing Testing Coverage 427
 - Leveraging Android SDK Tools for Android Application Testing 434
 - Avoiding Silly Mistakes in Android Application Testing 435
- Android Application Testing Essentials 435
 - Unit Testing with JUnit 436
 - Introducing the PasswordMatcher Application 437
 - Determining What Our Tests Should Prove 441
 - Creating an Android Test Project 441
 - Writing the Tests 447
 - Running Your First Test Using the Android IDE 450
 - Analyzing the Test Results 450
 - Adding Additional Tests 453
- More Android Automated Testing Programs and APIs 455
- Summary 457
- Quiz Questions 457
- Exercises 457
- References and More Information 458

19 Publishing Your Android Application 459

- Choosing the Right Distribution Model 459
 - Protecting Your Intellectual Property 460
 - Following the Policies of Google Play 461
 - Billing the User 461
- Packaging Your Application for Publication 462
 - Preparing Your Code for Packaging 463
 - Packing and Signing Your Application 465
 - Testing the Release Version of Your Application Package 467
 - Including All Required Resources 467

Readying Your Servers or Services	467
Distributing Your Application	467
Publishing to Google Play	468
Signing Up for Publishing to Google Play	468
Uploading Your Application to Google Play	471
Uploading Application Marketing Assets	473
Configuring Pricing and Distribution Details	474
Configuring Additional Application Options	475
Managing Other Developer Console Options	476
Publishing Your Application to Google Play	476
Managing Your Application on Google Play	476
Google Play Staged Rollouts	478
Publishing to the Google Play Private Channel	478
Translating Your Application	478
Publishing Using Other Alternatives	479
Self-Publishing Your Application	479
Summary	480
Quiz Questions	481
Exercises	481
References and More Information	481

VI: Appendixes

A Mastering the Android Development Tools 485

Using the Android Documentation	485
Leveraging the Android Emulator	489
Viewing Application Log Data with LogCat	490
Debugging Applications with DDMS	490
Using Android Debug Bridge (ADB)	490
Using the Resource Editors and UI Designer	491
Using the Android Hierarchy Viewer	493
Launching the Hierarchy Viewer	494
Working in Layout View Mode	495
Optimizing Your User Interface	496
Working in Pixel Perfect Mode	497
Working with Nine-Patch Stretchable Graphics	497
Working with Other Android Tools	500

Summary	502
Quiz Questions	503
Exercises	503
References and More Information	503
B Quick-Start Guide: The Android Emulator	505
Simulating Reality: The Emulator's Purpose	505
Working with Android Virtual Devices	507
Using the Android Virtual Device Manager	508
Creating an AVD	509
Creating AVDs with Custom Hardware Settings	510
Launching the Emulator with a Specific AVD	512
Maintaining Emulator Performance	512
Configuring Emulator Startup Options	513
Launching an Emulator to Run an Application	513
Launching an Emulator from the Android Virtual Device Manager	515
Configuring the GPS Location of the Emulator	516
Calling between Two Emulator Instances	517
Messaging between Two Emulator Instances	518
Interacting with the Emulator through the Console	520
Using the Console to Simulate Incoming Calls	521
Using the Console to Simulate SMS Messages	522
Using the Console to Send GPS Coordinates	523
Using the Console to Monitor Network Status	523
Using the Console to Manipulate Power Settings	523
Using Other Console Commands	524
Enjoying the Emulator	524
Understanding Emulator Limitations	525
Summary	526
Quiz Questions	526
Exercises	527
References and More Information	527
C Quick-Start Guide: Android DDMS	529
Using DDMS with the Android IDE and as a Standalone Application	529

Getting Up to Speed Using Key Features of DDMS	530
Working with Processes, Threads, and the Heap	531
Attaching a Debugger to an Android Application	531
Stopping a Process	532
Monitoring Thread Activity of an Android Application	532
Monitoring Heap Activity	532
Prompting Garbage Collection	534
Creating and Using an HPROF File	534
Using the Allocation Tracker	534
Viewing Network Statistics	535
Working with the File Explorer	536
Browsing the File System of an Emulator or Device	536
Copying Files from the Emulator or Device	538
Copying Files to the Emulator or Device	539
Deleting Files on the Emulator or Device	539
Working with the Emulator Control	539
Change Telephony Status	540
Simulating Incoming Voice Calls	540
Simulating Incoming SMS Messages	540
Sending a Location Fix	541
Working with the System Information Pane	541
Taking Screen Captures of the Emulator and Device Screens	542
Working with Application Logging	543
Summary	544
Quiz Questions	545
Exercises	545
References and More Information	545
D Android IDE and Eclipse Tips and Tricks	547
Organizing Your Android IDE Workspace	547
Integrating with Source Control Services	547
Repositioning Tabs within Perspectives	548
Maximizing Windows	548
Minimizing Windows	548
Viewing Windows Side by Side	548

Viewing Two Sections of the Same File	550
Closing Unwanted Tabs	550
Keeping Windows under Control	550
Creating Custom Log Filters	551
Searching Your Project	551
Organizing Android IDE Tasks	551
Writing Code in Java	552
Using Autocomplete	552
Creating New Classes and Methods	552
Organizing Imports	553
Formatting Code	553
Renaming Almost Anything	553
Refactoring Code	554
Reorganizing Code	555
Using QuickFix	555
Providing Javadoc-Style Documentation	556
Resolving Mysterious Build Errors	556
Summary	557
Quiz Questions	557
Exercises	557
References and More Information	557
E Answers to Quiz Questions	559
Chapter 1: Introducing Android	559
Chapter 2: Setting Up Your Android Development Environment	559
Chapter 3: Writing Your First Android Application	559
Chapter 4: Android Application Basics	560
Chapter 5: Defining Your Application Using the Android Manifest File	560
Chapter 6: Managing Application Resources	560
Chapter 7: Exploring User Interface Building Blocks	561
Chapter 8: Designing with Layouts	561
Chapter 9: Partitioning the User Interface with Fragments	561
Chapter 10: Displaying Dialogs	562
Chapter 11: Using Android Preferences	562
Chapter 12: Working with Files and Directories	562

Chapter 13: Leveraging Content Providers	562
Chapter 14: Designing Compatible Applications	563
Chapter 15: Learning the Android Software Development Process	563
Chapter 16: Designing and Developing Bulletproof Android Applications	563
Chapter 17: Planning the Android Application Experience	564
Chapter 18: Testing Android Applications	564
Chapter 19: Publishing Your Android Application	564
Appendix A: Mastering the Android Development Tools	565
Appendix B: Quick-Start Guide: The Android Emulator	565
Appendix C: Quick-Start Guide: Android DDMS	565
Appendix D: Android IDE and Eclipse Tips and Tricks	566
Index	567

This page intentionally left blank

Acknowledgments

This book is the result of collaboration among a great group, from the efforts of the team at Pearson Education (Addison–Wesley Professional); from the suggestions made by the technical reviewers; and from the support of family, friends, coworkers, and acquaintances alike. We'd like to thank the Android developer community, Google, and the Open Handset Alliance for their vision and expertise. Special thanks go to Mark Taub for believing in the vision for this edition; Laura Lewin, who was the driving force behind the book and without whom it would not have become a reality; Olivia Basegio, who was instrumental in orchestrating the efforts of everyone involved; Songlin Qiu for performing countless iterations through the manuscript to make this book ready for production; and the technical reviewers: Ray Rischpater, who made many beneficial recommendations, and Doug Jones, who suggested improvements of the fine details (as well as Mike Wallace, Mark Gjoel, Dan Galpin, Tony Hillerson, Ronan Schwarz, and Charles Stearns, who reviewed previous editions). Dan Galpin also graciously provided the clever Android graphics used for Tips, Notes, and Warnings. Amy Badger must be commended for her wonderful waterfall illustration, and we also thank Hans Bodlaender for letting us use the nifty chess font he developed as a hobby project.

This page intentionally left blank

About the Authors

Joseph Annuzzi, Jr., is a freelance software architect, graphic artist, writer, and technical reviewer. He usually can be found mastering the Android platform, implementing cutting-edge HTML5 capabilities, leveraging various cloud technologies, speaking in different programming languages, working with diverse frameworks, integrating with various social APIs, tinkering with peer-to-peer, cryptography, and computer vision algorithms, or creating stunningly realistic 3D renders. He is always on the lookout for disruptive Internet and mobile technologies and has multiple patent applications in process. He graduated from the University of California, Davis, with a BS in managerial economics and a minor in computer science and lives where much of the action is, Silicon Valley.

When he is not working with technology, he has been known to lounge in the sun on the beaches of the Black Sea with international movie stars; he has trekked through the Bavarian forest in winter, has immersed himself in the culture of the Italian Mediterranean, and has narrowly escaped the wrath of an organized crime ring in Eastern Europe after his taxi dropped him off in front of the bank ATM they were liquidating. He also lives an active and healthy lifestyle, designs and performs custom fitness training routines to stay in shape, and adores his loyal beagle, Cleopatra.

Lauren Darcey is responsible for the technical leadership and direction of a small software company specializing in mobile technologies, including Android, iOS, BlackBerry, Palm Pre, BREW, and J2ME, and consulting services. With more than two decades of experience in professional software production, Lauren is a recognized authority in application architecture and the development of commercial-grade mobile applications. Lauren received a BS in computer science from the University of California, Santa Cruz.

She spends her copious free time traveling the world with her geeky mobile-minded husband and is an avid nature photographer. Her work has been published in books and newspapers around the world. In South Africa, she dove with 4-meter-long great white sharks and got stuck between a herd of rampaging hippopotami and an irritated bull elephant. She's been attacked by monkeys in Japan, gotten stuck in a ravine with two hungry lions in Kenya, gotten thirsty in Egypt, narrowly avoided a *coup d'état* in Thailand, geocached her way through the Swiss Alps, drank her way through the beer halls of Germany, slept in the crumbling castles of Europe, and gotten her tongue stuck to an iceberg in Iceland (while being watched by a herd of suspicious wild reindeer).

Shane Conder has extensive development experience and has focused his attention on mobile and embedded development for the past decade. He has designed and developed many commercial applications for Android, iOS, BREW, BlackBerry, J2ME, Palm, and

Windows Mobile—some of which have been installed on millions of phones worldwide. Shane has written extensively about the mobile industry and evaluated mobile development platforms on his tech blogs and is well known within the blogosphere. Shane received a BS in computer science from the University of California.

A self-admitted gadget freak, Shane always has the latest smartphone, tablet, or other mobile device. He can often be found fiddling with the latest technologies, such as cloud services and mobile platforms, and other exciting, state-of-the-art technologies that activate the creative part of his brain. He also enjoys traveling the world with his geeky wife, even if she did make him dive with 4-meter-long great white sharks and almost get eaten by a lion in Kenya. He admits that he has to take at least two phones with him when backpacking—even though there is no coverage—and that he snickered and whipped out his Android phone to take a picture when Laurie got her tongue stuck to that iceberg in Iceland, and that he is catching on that he should be writing his own bio.

Introduction

Android is a popular, free, open-source mobile platform that has taken the wireless world by storm. This book provides guidance for software development teams on designing, developing, testing, debugging, and distributing professional Android applications. If you're a veteran mobile developer, you can find tips and tricks to streamline the development process and take advantage of Android's unique features. If you're new to mobile development, this book provides everything you need to make a smooth transition from traditional software development to mobile development—specifically, its most promising platform: Android.

Who Should Read This Book

This book includes tips for successful mobile development based upon our years in the mobile industry and covers everything you need to know in order to run a successful Android project from concept to completion. We cover how the mobile software process differs from traditional software development, including tricks to save valuable time and pitfalls to avoid. Regardless of the size of your project, this book is for you.

This book was written for several audiences:

- **Software developers who want to learn to develop professional Android applications.** The bulk of this book is targeted at software developers with Java experience who do not necessarily have mobile development experience. More-seasoned developers of mobile applications can learn how to take advantage of Android and how it differs from the other technologies on the mobile development market today.
- **Quality assurance personnel tasked with testing Android applications.** Whether they are black-box or white-box testing, quality assurance engineers can find this book invaluable. We devote several chapters to mobile QA concerns, including topics such as developing solid test plans and defect-tracking systems for mobile applications, how to manage handsets, and how to test applications thoroughly using all the Android tools available.
- **Project managers planning and managing Android development teams.** Managers can use this book to help plan, hire for, and execute Android projects from start to finish. We cover project risk management and how to keep Android projects running smoothly.
- **Other audiences.** This book is useful not only to the software developer, but also to the corporation looking at potential vertical market applications, the entrepreneur thinking about a cool phone application, and the hobbyist looking for some

fun with his or her new phone. Businesses seeking to evaluate Android for their specific needs (including feasibility analysis) can also find the information provided valuable. Anyone with an Android handset and a good idea for a mobile application can put the information in this book to use for fun and profit.

Key Questions Answered in This Book

This book answers the following questions:

1. What is Android? How do the SDK versions differ?
2. How is Android different from other mobile technologies, and how can developers take advantage of these differences?
3. How do developers use the Android SDK and ADT Bundle to develop and debug Android applications on the emulator and handsets?
4. How are Android applications structured?
5. How do developers design robust user interfaces for mobile—specifically, for Android?
6. What capabilities does the Android SDK have and how can developers use them?
7. How does the mobile development process differ from traditional desktop development?
8. What strategies work best for Android development?
9. What do managers, developers, and testers need to look for when planning, developing, and testing a mobile application?
10. How do mobile teams design bulletproof Android applications for publication?
11. How do mobile teams package Android applications for deployment?
12. How do mobile teams make money from Android applications?
13. And, finally, what is new in this edition of the book?

How This Book Is Structured

Introduction to Android Application Development, Fourth Edition focuses on Android essentials, including setting up the development environment, understanding the application lifecycle, user interface design, developing for different types of devices, and the mobile software process from design and development to testing and publication of commercial-grade applications.

The book is divided into six parts. Here is an overview of the various parts:

■ Part I: An Overview of the Android Platform

Part I provides an introduction to Android, explaining how it differs from other mobile platforms. You become familiar with the Android SDK and tools, install the development tools, and write and run your first Android application—on the

emulator and on a handset. This section is of primary interest to developers and testers, especially white-box testers.

- **Part II: Android Application Basics**

Part II introduces the design principles necessary to write Android applications. You learn how Android applications are structured and how to include resources, such as strings, graphics, and user interface components, in your projects. This section is of primary interest to developers.

- **Part III: Android User Interface Design Essentials**

Part III dives deeper into how user interfaces are designed in Android. You learn about the core user interface element in Android: the `view`. You also learn about the most common user interface controls and layouts provided in the Android SDK. This section is of primary interest to developers.

- **Part IV: Android Application Design Essentials**

Part IV covers the features used by most Android applications, including storing persistent application data using preferences and working with files, directories, and content providers. You also learn how to design applications that will run smoothly on many different Android devices. This section is of primary interest to developers.

- **Part V: Publishing and Distributing Android Applications**

Part V covers the software development process for mobile, from start to finish, with tips and tricks for project management, software developers, user experience designers, and quality assurance personnel.

- **Part VI: Appendixes**

Part VI includes several helpful appendixes to help you get up and running with the most important Android tools. This section consists of an overview of the Android development tools, two helpful quick-start guides for the Android development tools—the emulator and `DDMS`—an appendix of Android IDE tips and tricks, as well as answers to the end-of-chapter quiz questions.

An Overview of Changes in This Edition

When we began writing the first edition of this book, there were no Android devices on the market. Today there are hundreds of devices shipping all over the world—smartphones, tablets, e-book readers, smart watches, and specialty devices such as gaming consoles, Google TV, and Google Glass. Other devices such as Google Chromecast provide screen sharing between Android devices and TVs.

The Android platform has gone through extensive changes since the first edition of this book was published. The Android SDK has many new features, and the development tools have received many much-needed upgrades. Android, as a technology, is now the leader within the mobile marketplace.

In this new edition, we took the opportunity to add a wealth of information about how to plan the Android application experience for users. In addition, we have included valuable and ready-to-use techniques for automating the testing of your Android

applications, to ensure that you deliver high-quality code. We have also updated many chapters and accompanying content for making use of `Fragment`-based implementation approaches. But don't worry, it's still the book readers loved the first, second, and third time around; it's just much bigger, better, and more comprehensive, following many best practices. In addition to adding new content, we've retested and upgraded all existing content (text and sample code) for use with the latest Android SDKs available while still remaining backward compatible. We created quiz questions to help readers ensure they understand each chapter's content, and we added end-of-chapter exercises for readers to perform to dig deeper into all that Android has to offer. The Android development community is diverse, and we aim to support all developers, regardless of which devices they are developing for. This includes developers who need to target nearly all platforms, so coverage in some key areas of older SDKs continues to be included because it's often the most reasonable option for compatibility.

Here are some of the highlights of the additions and enhancements we've made to this edition:

- Coverage of the latest and greatest Android tools and utilities is included.
- The topic of planning the Android application experience now has its own chapter, which includes a discussion of different navigation patterns with a new code sample and presents techniques that you can use to improve the quality of the user experience.
- The chapter on testing has brand-new content to include topics such as unit testing and provides a practical code sample showing how to leverage the automated testing techniques used by the experts for testing their Android applications.
- A new code sample and a discussion of how to add an `ActionBar` to your applications have been included.
- The chapter on dialogs has been completely updated to make use of `DialogFragments`.
- The chapter on Android preferences now includes an additional code sample with a brand-new discussion of how to add preference fragments that display accordingly within single-pane and multipane layouts.
- The publishing chapter has been completely redesigned to discuss using the new Google Play Developer Console for publishing your applications, in addition to outlining new features provided within the console.
- All chapters and appendixes now include quiz questions and exercises for readers to test their knowledge of the subject matter presented.
- All existing chapters have been updated, often with some entirely new sections.
- All sample code and accompanying applications have been updated to work with the latest SDK.

As you can see, we cover many of the hottest and most exciting features that Android has to offer. We didn't take this review lightly; we touched every existing chapter,

updated content, and added new chapters as well. Finally, we included many additions, clarifications, and, yes, even a few fixes based on the feedback from our fantastic (and meticulous) readers. Thank you!

Development Environments Used in This Book

The Android code in this book was written using the following development environments:

- Windows 7
- Android ADT Bundle (the `adt-bundle-windows-x86-20130729.zip` file was used)
- Android SDK Version 4.3, API Level 18 (Jelly Bean)
- Android SDK Tools Revision 22.0.5
- Android SDK Platform Tools 18.0.1
- Android SDK Build Tools 18.0.1
- Android Support Library Revision 18 (where applicable)
- Java SE Development Kit (JDK) 6 Update 45
- Android devices: Nexus 4 (phone), Nexus 7 (small tablet), and Nexus 10 (large tablet)

The Android platform continues to grow aggressively in market share against competing mobile platforms, such as Apple iOS and BlackBerry. New and exciting types of Android devices reach consumers' hands at a furious pace. Developers have embraced Android as a target platform to reach the device users of today and tomorrow.

Android's latest major platform update, Android 4.3—frequently called by its code name, Jelly Bean, or just JB—brings many new features that help differentiate Android from the competition. This book features the latest SDK and tools available, but it does not focus on them to the detriment of popular legacy versions of the platform. The book is meant to be an overall reference to help developers support all popular devices on the market today. As of the writing of this book, approximately 37.9% of users' devices are running a version of Android Jelly Bean, 4.1 or 4.2. Of course, some devices will receive upgrades, and users will purchase new Jelly Bean devices as they become available, but for now, developers need to straddle this gap and support numerous versions of Android to reach the majority of users in the field. In addition, the next version of the Android operating system is likely to be released in the near future.

So what does this mean for this book? It means we provide legacy API support and discuss some of the newer APIs available in later versions of the Android SDK. We discuss strategies for supporting all (or at least most) users in terms of compatibility. And we provide screenshots that highlight different versions of the Android SDK, because each major revision has brought with it a change in the look and feel of the overall platform. That said, we are assuming that you are downloading the latest Android tools, so we provide screenshots and steps that support the latest tools available at the time of writing, not

legacy tools. Those are the boundaries we set when trying to determine what to include and leave out of this book.

Supplementary Materials Available

The source code that accompanies this book is available for download from our book's website: <http://introductiontoandroid.blogspot.com/2013/05/book-code-samples.html>. You'll also find other Android topics discussed at our book website (<http://introductiontoandroid.blogspot.com>).

Where to Find More Information

There is a vibrant, helpful Android developer community on the Web. Here are a number of useful websites for Android developers and followers of the wireless industry:

- Android Developer website: the Android SDK and developer reference site:
<http://d.android.com/index.html> or <http://d.android.com>
- Google Plus: Android Developers Group
<https://plus.google.com/+AndroidDevelopers/posts>
- Stack Overflow: the Android website with great technical information (complete with tags) and an official support forum for developers:
<http://stackoverflow.com/questions/tagged/android>
- Open Handset Alliance: Android manufacturers, operators, and developers:
<http://openhandsalliance.com>
- Google Play: buy and sell Android applications:
<https://play.google.com/store>
- Mobiletuts+: mobile development tutorials, including Android:
<http://mobile.tutsplus.com/category/tutorials/android>
- anddev.org: an Android developer forum:
<http://anddev.org>
- Google Team Android Apps: open-source Android applications:
<http://apps-for-android.googlecode.com>
- Android Tools Project Site: the tools team discusses updates and changes:
<https://sites.google.com/a/android.com/tools/recent>
- FierceDeveloper: a weekly newsletter for wireless developers:
<http://fiercedev.com>
- Wireless Developer Network: daily news on the wireless industry:
<http://wirelessdevnet.com>

- XDA-Developers Android Forum: from general development to ROMs:
<http://forum.xda-developers.com/forumdisplay.php?f=564>
- Developer.com: a developer-oriented site with mobile articles:
<http://developer.com>

Conventions Used in This Book

This book uses the following conventions:

- Code and programming terms are set in monospace text.
- Java import statements, exception handling, and error checking are often removed from printed code examples for clarity and to keep the book a reasonable length.

This book also presents information in the following sidebars:



Tip

Tips provide useful information or hints related to the current text.



Note

Notes provide additional information that might be interesting or relevant.



Warning

Warnings provide hints or tips about pitfalls that may be encountered and how to avoid them.

Contacting the Authors

We welcome your comments, questions, and feedback. We invite you to visit our blog at

- <http://introductiontoandroid.blogspot.com>

Or email us at

- introtoandroid4e@gmail.com

Circle us on Google+:

- Joseph Anuzzi, Jr.: <http://goo.gl/FBQeL>
- Lauren Darcey: <http://goo.gl/P3RGo>
- Shane Conder: <http://goo.gl/BpVJh>

This page intentionally left blank

3

Writing Your First Android Application

You should now have a workable Android development environment set up on your computer. Ideally, you have an Android device as well. Now it's time for you to start writing some Android code. In this chapter, you learn how to install the Android sample applications and to add and create Android projects from within the Android IDE. You also learn how to verify that your Android development environment is set up correctly. You then write and debug your first Android application in the software emulator and on an Android device.



Note

The Android Development Tool Bundles are updated frequently. We have made every attempt to provide the latest steps for the latest tools. However, these steps and the user interfaces described in this chapter may change at any time. Please refer to the Android development website (<http://d.android.com/sdk/index.html>) and our book website (<http://introductiontoandroid.blogspot.com>) for the latest information.

Testing Your Development Environment

The best way to make sure you configured your development environment correctly is to run an existing Android application. You can do this easily by using one of the sample applications provided as part of the Android SDK in the `samples` subdirectory found where your Android SDK is installed.

Within the Android SDK sample applications, you will find a classic game called Snake ([http://en.wikipedia.org/wiki/Snake_\(video_game\)](http://en.wikipedia.org/wiki/Snake_(video_game))). To build and run the `Snake` application, you must create a new Android project in your Android IDE workspace based on the existing Android sample project, create an appropriate Android Virtual Device (AVD) profile, and configure a launch configuration for that project. After you have everything set up correctly, you can build the application and run it on the Android emulator and on an Android device. By testing your development environment with a sample application, you can rule out project configuration and coding issues and focus on determining

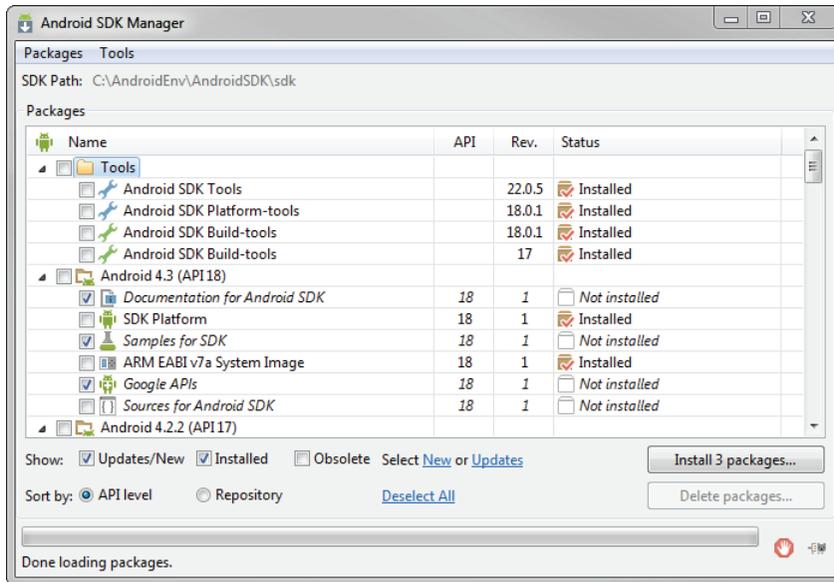


Figure 3.1 The Android SDK Manager.

whether the tools are set up properly for Android development. After this fact has been established, you can move on to writing and compiling your own applications.

Adding the Android Samples Using the SDK Manager

One quick way to learn how to develop Android applications is by reviewing an application that has already been created. There are many Android applications available for this purpose, but first we must download them. Here is how:

1. From within the Android IDE, click the Android SDK Manager icon (📁) to open the Android SDK Manager. You should see a dialog similar to that in Figure 3.1.
2. You now need to install the *Samples for SDK* listed under *Android 4.3 (API 18)*, so go ahead and select this item. You may also want to install a few additional items along with the samples, so select the following for installation (shown in Figure 3.1): *Documentation for Android SDK* and *Google APIs*. Then click *Install Packages*. Make sure that the proper *SDK Tools*, *Platform-tools*, *Build-tools*, *SDK Platform*, and *System Image* are installed as well; if they are not, you should select those for installation now, too.
3. A new dialog appears (see Figure 3.2) asking you to accept the license agreement for the packages that you will be installing. You may accept or reject each license

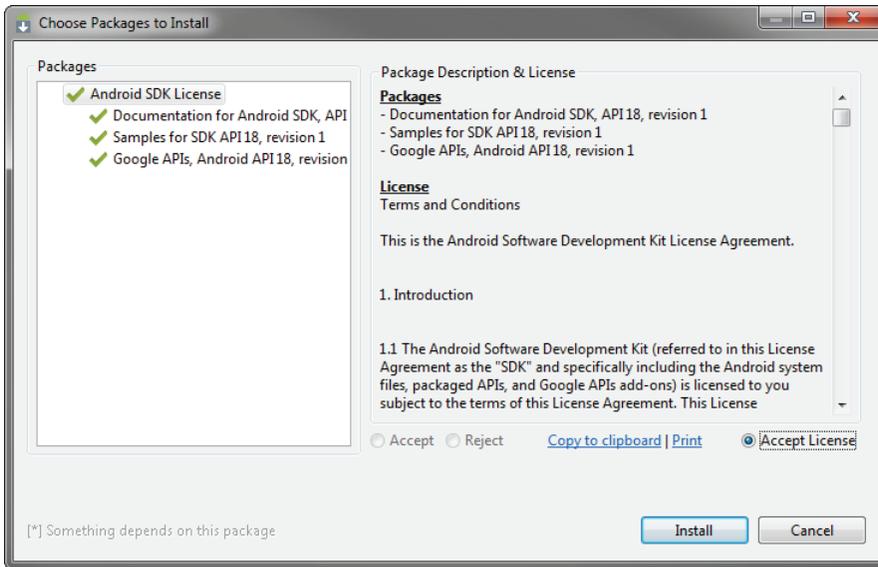


Figure 3.2 Accepting the license agreements.

individually by highlighting a particular package in the left pane and choosing Accept or Reject, or you can accept them all at once by highlighting Android SDK License in the left pane and choosing Accept License. Let's accept all the licenses together by selecting Android SDK License in the left pane, choosing Accept License, and then clicking Install. This will initiate the installation of the selected packages. Wait until the installation is complete.



Tip

To learn more about how to download the Android SDK sample applications for your particular development platform, see <http://d.android.com/tools/samples/index.html>.

Now that the installation is completed, you are ready to begin loading Android sample projects into your workspace.

Adding the Snake Project to Your Android IDE Workspace

To add the Snake project to your Android IDE workspace, follow these steps:

1. Choose File, New, Other...
2. Choose Android, Android Sample Project (see Figure 3.3). Click Next.

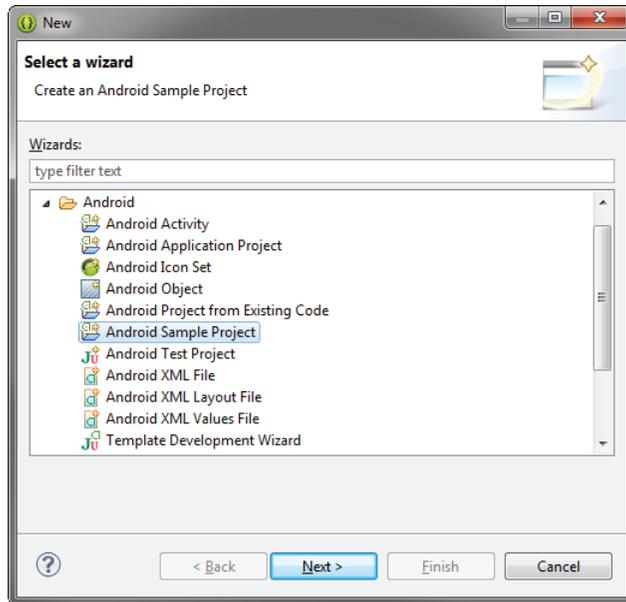


Figure 3.3 Creating a new Android sample project.

3. Choose your build target (see Figure 3.4). In this case, we've picked Android 4.3, API Level 18, from the Android Open Source Project. Click **Next**.
4. Select which sample you want to create (see Figure 3.5). Choose **Snake**.
5. Click **Finish**. You now see the **Snake** project files in your workspace (see Figure 3.6).



Warning

Occasionally the Android IDE shows an error like “Project ‘Snake’ is missing required source folder: gen” when you’re adding an existing project to the workspace. If this happens, navigate to the `/gen` directory and delete the files within. These files are automatically regenerated and the error should disappear. Performing a `Clean` operation followed by a `Build` operation does not always solve this problem.

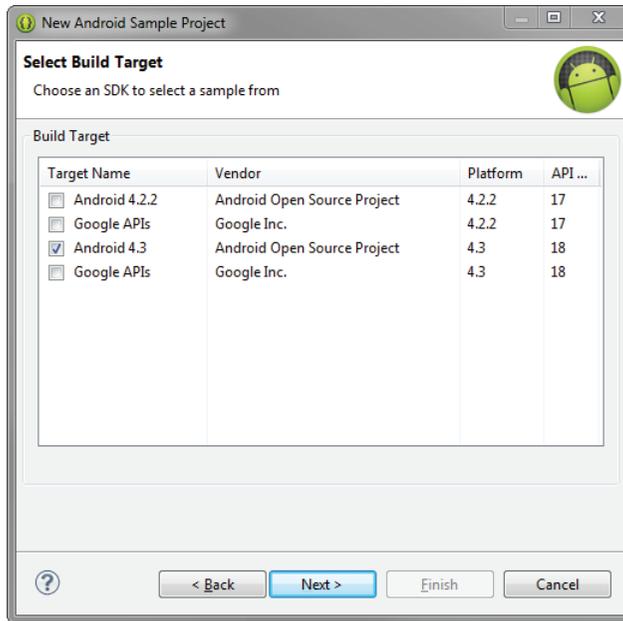


Figure 3.4 Choose an API level for the sample.

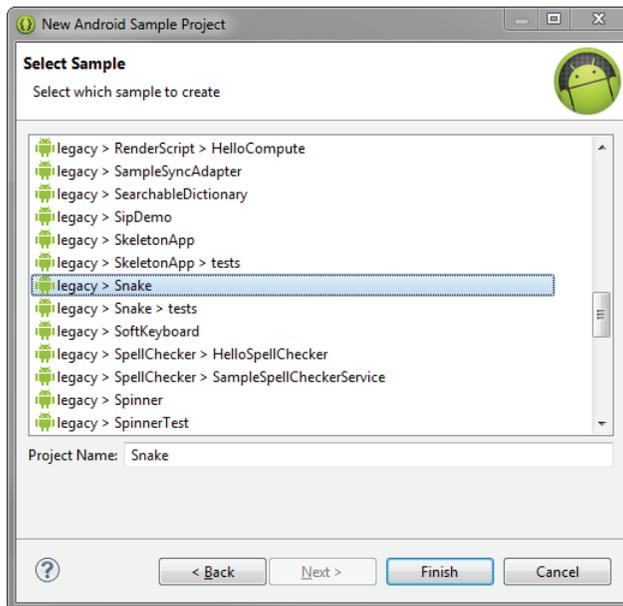


Figure 3.5 Picking the Snake sample project.

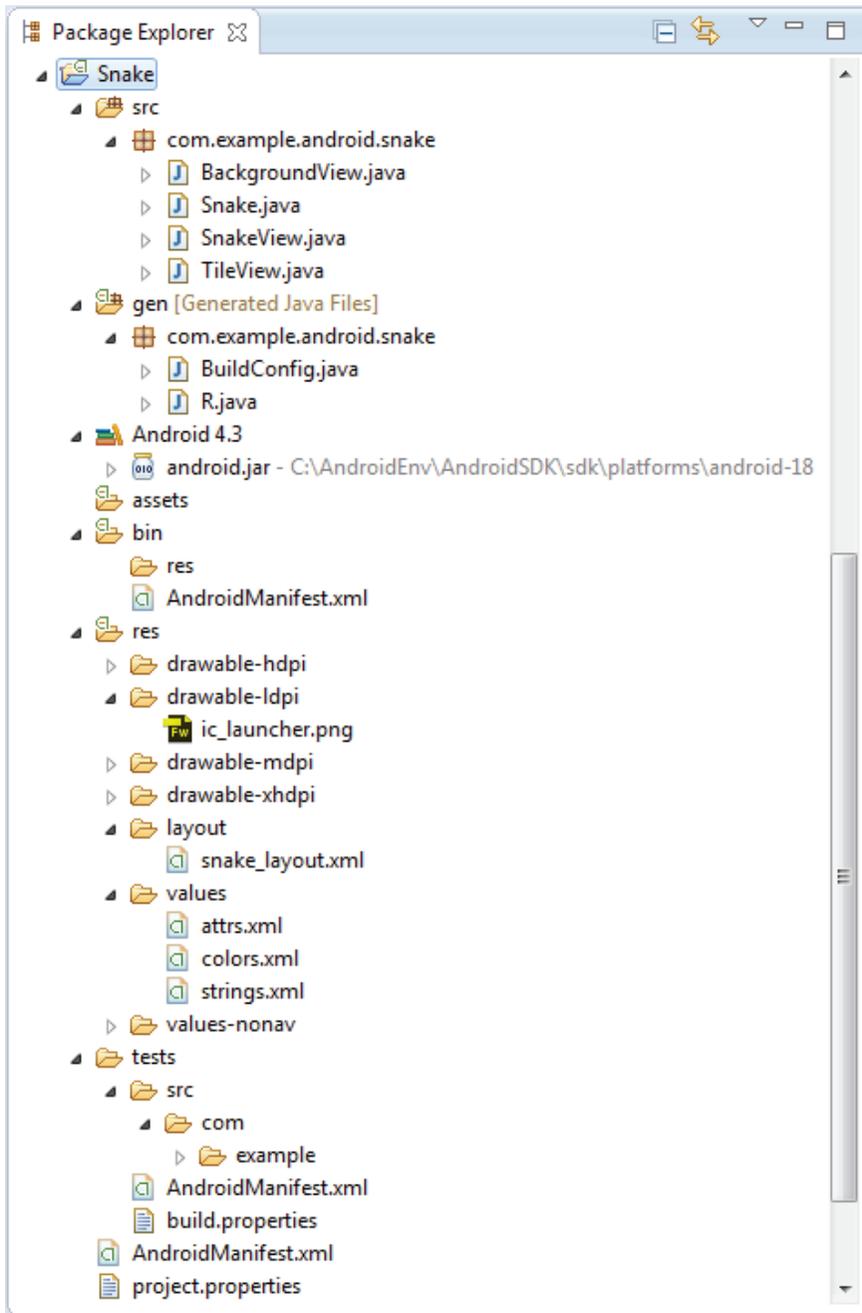


Figure 3.6 The Snake project files.

Creating an AVD for Your Snake Project

The next step is to create an AVD that describes what type of device you want to emulate when running the `Snake` application. This AVD profile describes what type of device you want the emulator to simulate, including which Android platform to support. You do not need to create new AVDs for each application, only for each device you want to emulate. You can specify different screen sizes and orientations, and you can specify whether the emulator has an SD card and, if it does, what capacity the card has.

For the purposes of this example, an AVD for the default installation of Android 4.3 suffices. Here are the steps to create a basic AVD:

1. Launch the Android Virtual Device Manager from within the Android IDE by clicking the little Android device icon on the toolbar (📱). If you cannot find the icon, you can also launch the manager through the Window menu of the Android IDE. You should now see the Android Virtual Device Manager window (see Figure 3.7).
2. Click the `New` button.
3. Choose a name for your AVD. Because we are going to take all the defaults, give this AVD a name of `AndroidVanilla`.

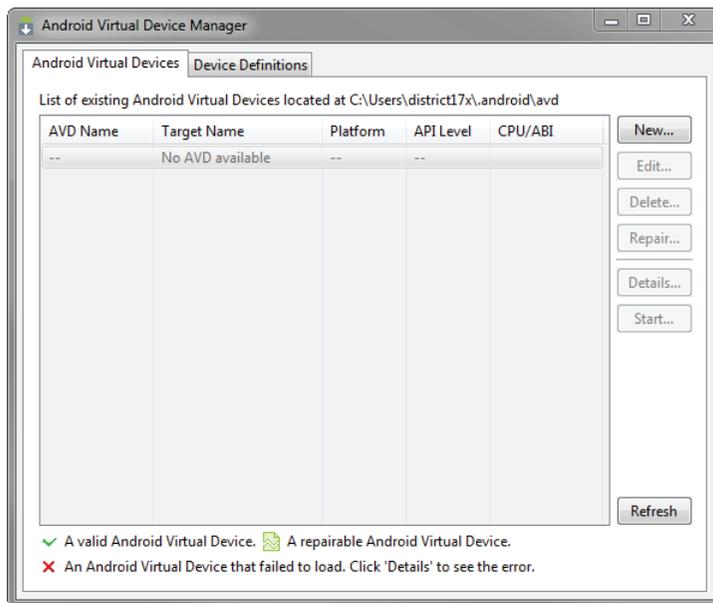


Figure 3.7 Android Virtual Device Manager.

4. Choose a device. This option controls the different resolutions of the emulator. We want to choose a typical device size, so in this case, select `Nexus 4 (4.7", 768 × 1280: xhdpi)`. This option most directly correlates to the popular Nexus 4 Google-branded device. Feel free to choose the most appropriate device to match the Android device on which you plan to run the application.
5. Choose a build target. We want a typical Android 4.3 device, so choose `Google APIs (Google Inc.) - API Level 18` from the drop-down menu. In addition to including the Android APIs, this option will also include the Google APIs and applications, such as the Maps application, as part of the platform image. Although we could choose the standard `Android 4.3 - APIs Level 18` for this project, it is important to be aware of the additional options the Google APIs provide.
6. For the `Memory Options` setting, you may have to try different values for optimal performance depending on the memory configuration of your development machine. The default `RAM` value for this virtual device is 1907 and the `VM Heap` is 64. If your machine is older and does not have a lot of memory, you may need to lower this value significantly to something like 512. The development machine used for this book has 8GB of RAM with a fairly powerful quad-core processor, and the `RAM` value we decided to use is 768 with the `VM Heap` set to 64.
7. Choose an SD card capacity, in either kibibytes or mibibytes. (Not familiar with kibibytes? See this Wikipedia entry: <http://en.wikipedia.org/wiki/Kibibyte>.) This SD card image will take up space on your hard drive and may also take a long time to allocate, so choose something reasonable, such as 1024MiB.
8. Seriously consider enabling the `Snapshot` feature listed under `Emulation Options`. This greatly improves emulator startup performance. See Appendix B, “Quick-Start Guide: The Android Emulator,” for details.
Your project settings will look like Figure 3.8.
9. Click the `OK` button to create the AVD, and then wait for the operation to complete.
10. You should now see the AVD that you just created listed within your Android Virtual Device Manager (see Figure 3.9).

For more information on creating different types of AVDs, check out Appendix B.

Creating a Launch Configuration for Your Snake Project

Next, you must create a launch configuration in the Android IDE to configure under what circumstances the `Snake` application builds and launches. The launch configuration is where you configure the emulator options to use and the entry point for your application.

You can create `Run` configurations and `Debug` configurations separately, each with different options. These configurations are created under the `Run` menu in the Android IDE

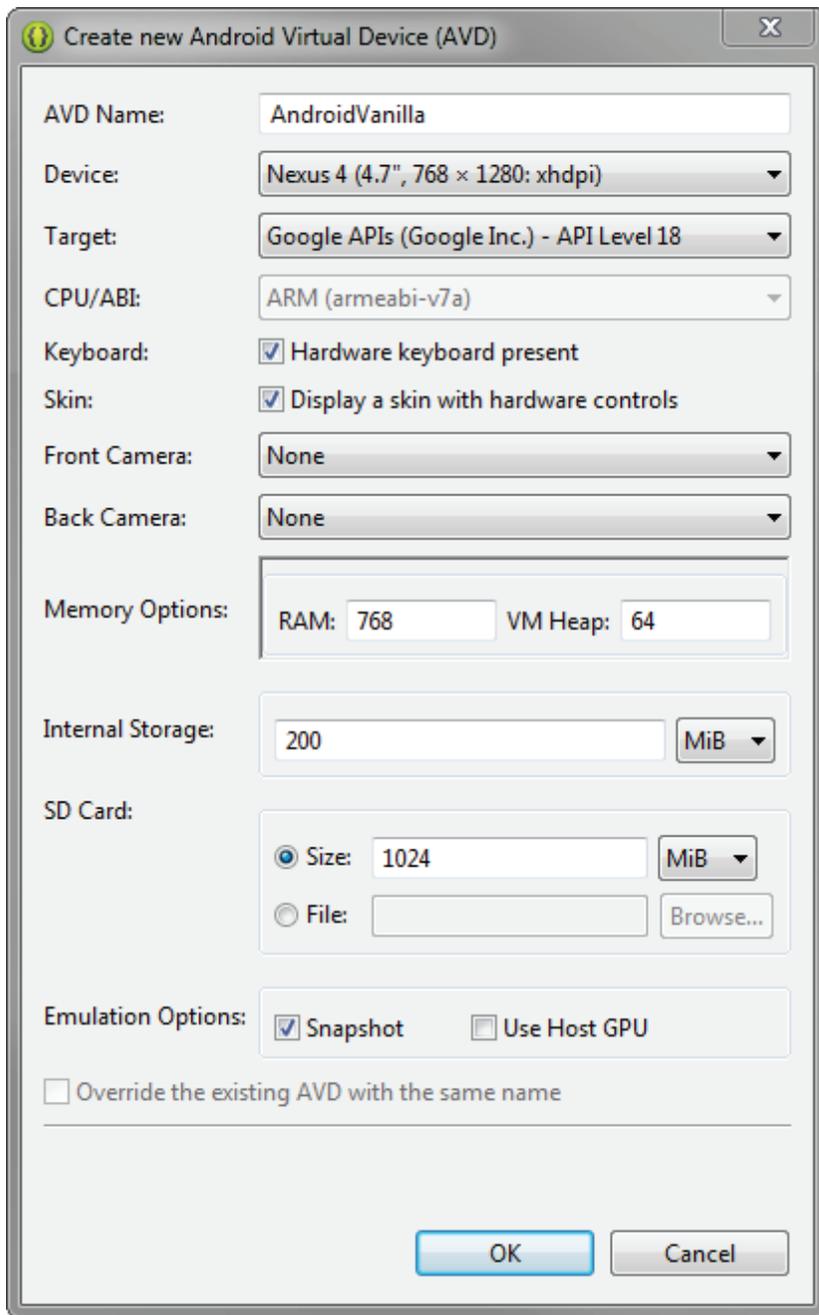


Figure 3.8 Creating a new AVD.

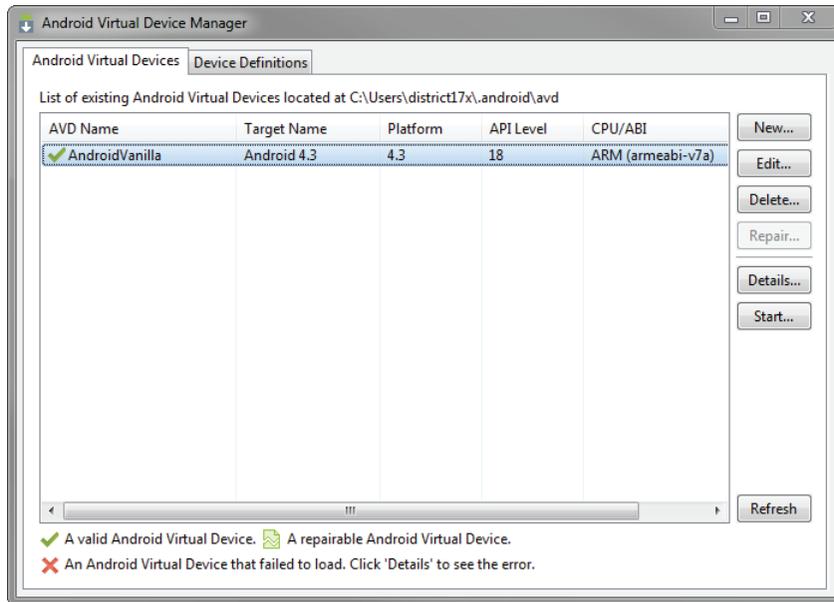


Figure 3.9 The new AVD is now listed.

(Run, Run Configurations... and Run, Debug Configurations...). Follow these steps to create a basic Debug configuration for the Snake application:

1. Choose Run, Debug Configurations...
2. Double-click `Android Application` to create a new configuration.
3. Name your Debug configuration `SnakeDebugConfig`.
4. Choose the project by clicking the `Browse` button and choosing the `Snake` project (see Figure 3.10).
5. Switch to the `Target` tab and, from the preferred AVD list, choose the `AndroidVanilla` AVD created earlier, as shown in Figure 3.11.
6. Choose `Apply` and then `Close`.

You can set other emulator and launch options on the `Target` and `Common` tabs, but for now we are leaving the defaults as they are.

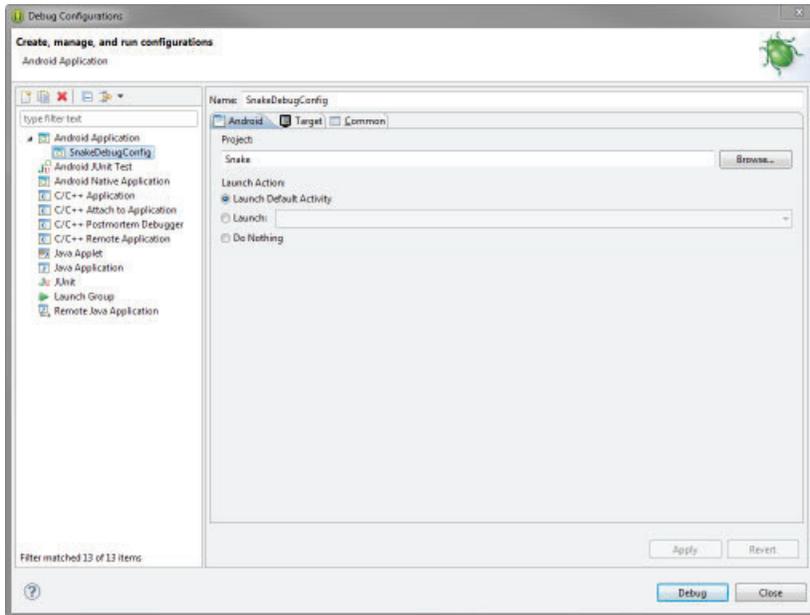


Figure 3.10 Naming the Debug configuration in the Android IDE.

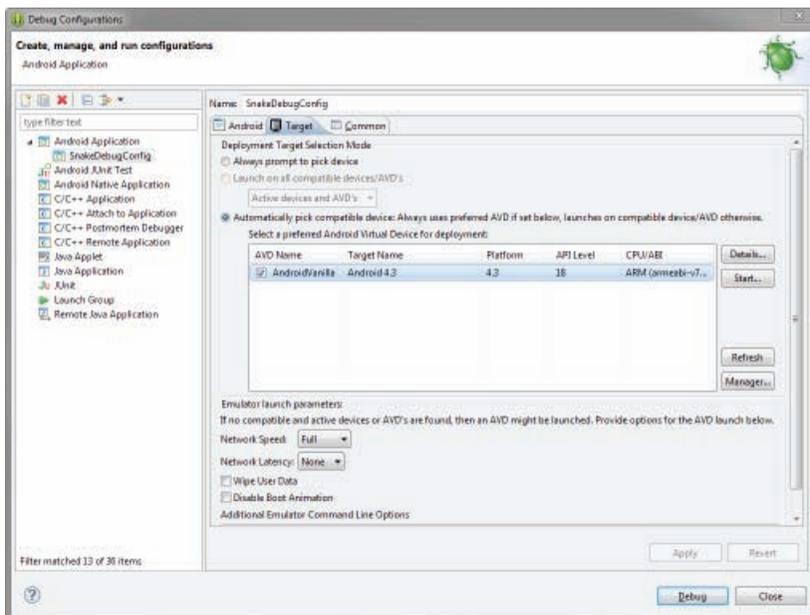


Figure 3.11 Target AVD for the Debug configuration in the Android IDE.

Running the Snake Application in the Android Emulator

Now you can run the *Snake* application using the following steps:

1. Choose the `Debug As` icon drop-down menu on the toolbar (🐛).
2. Pull the drop-down menu and choose the `SnakeDebugConfig` you created. If you do not see the `SnakeDebugConfig` listed, find it in the `Debug Configurations...` listing and click the `Debug` button. Subsequent launches can be initiated from the little bug drop-down.
3. The Android emulator starts up; this might take a few moments to initialize. Then the application will be installed or reinstalled onto the emulator.



Tip

It can take a long time for the emulator to start up, even on very fast computers. You might want to leave it around while you work and reattach to it as needed. The tools in the Android IDE handle reinstalling the application and relaunching it, so you can more easily keep the emulator loaded all the time. This is another reason to enable the `Snapshot` feature for each AVD. You can also use the `Start` button on the Android Virtual Device Manager to load an emulator before you need it. Launching the AVD this way also gives you some additional options such as screen scaling (see Figure 3.12), which can be used to either fit the AVD on your screen if it's very high resolution or more closely emulate the size it might be on real hardware.

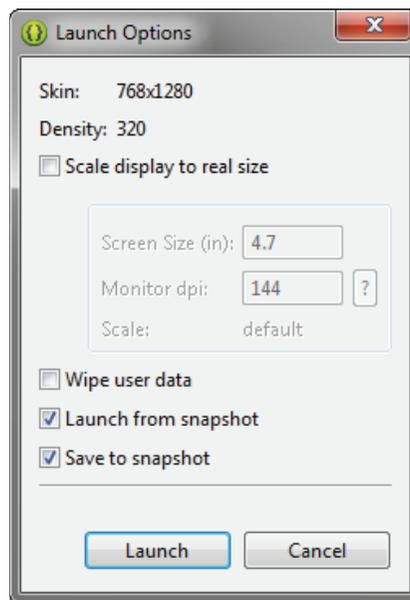


Figure 3.12 Configuring AVD launch options.

4. If necessary, swipe the screen from left to right to unlock the emulator, as shown in Figure 3.13.
5. The Snake application starts and you can play the game, as shown in Figure 3.14.

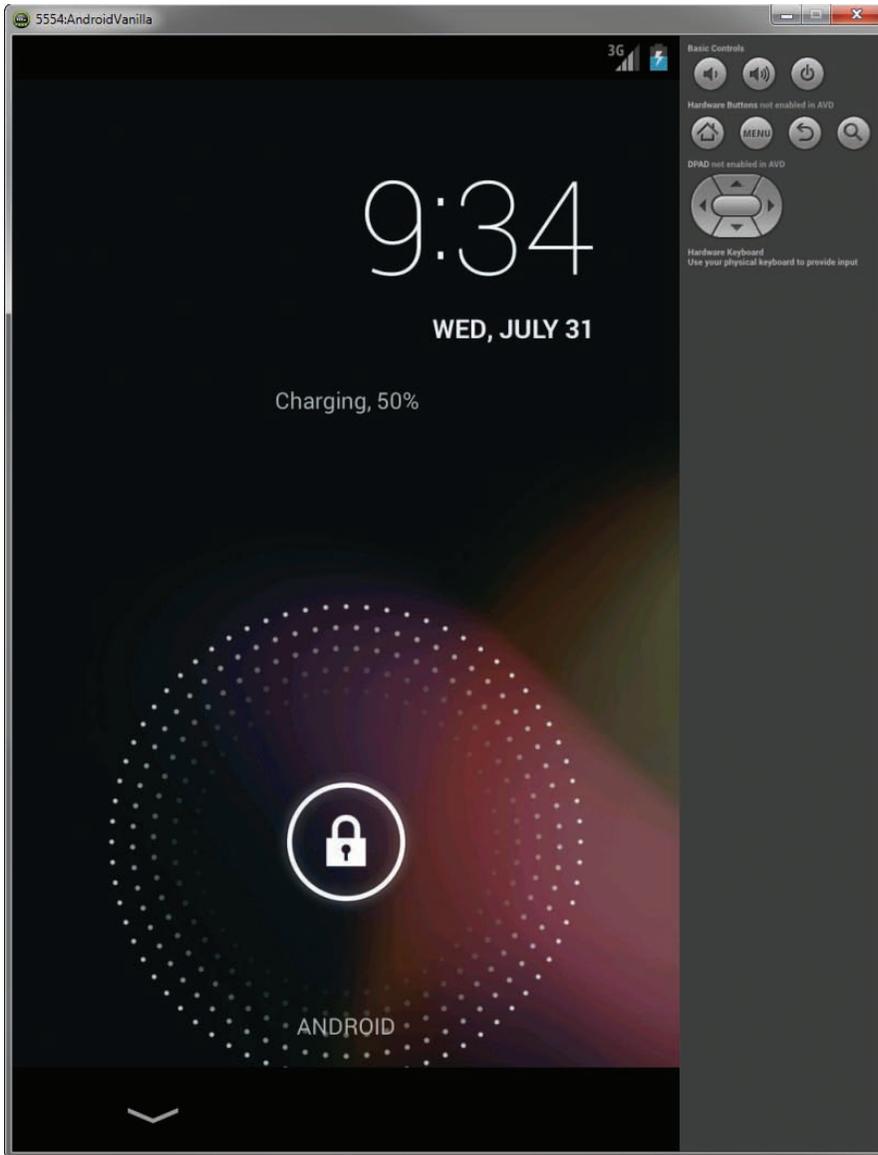


Figure 3.13 The Android emulator launching (locked).

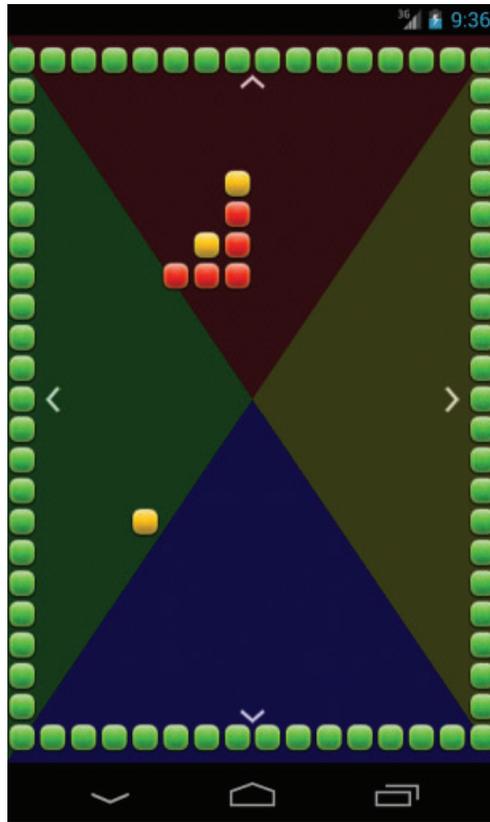


Figure 3.14 The Snake game in the Android emulator.

You can interact with the Snake application through the emulator and play the game. You can also launch the Snake application from the All Apps screen at any time by clicking its application icon. There is no need to shut down and restart the emulator every time you rebuild and reinstall your application for testing. Simply leave the emulator running on your computer in the background while you work in the Android IDE and then redeploy using the Debug configuration again.

Building Your First Android Application

Now it's time to write your first Android application from scratch. To get your feet wet, you will start with a simple "Hello World" application and build upon it to explore some of the features of the Android platform in more detail.

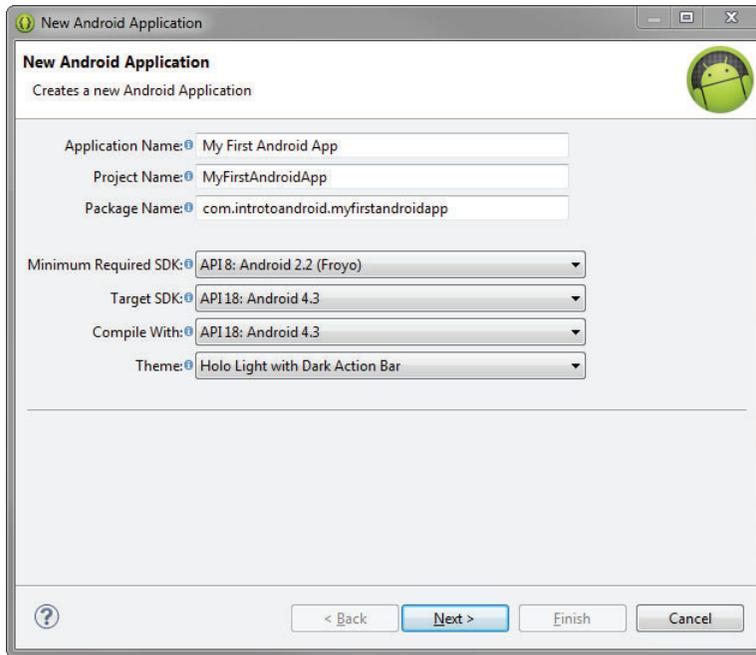


Figure 3.15 Configuring a new Android project.



Tip

The code examples provided in this chapter are taken from the `MyFirstAndroidApp` application. The source code for the `MyFirstAndroidApp` application is provided for download on the book's website.

Creating and Configuring a New Android Project

You can create a new Android application in much the same way that you added the `Snake` application to your Android IDE workspace.

The first thing you need to do is create a new project in your Android IDE workspace. The `Android Application Project` creation wizard creates all the required files for an Android application. Follow these steps within the Android IDE to create a new project:

1. Choose `File, New, Android Application Project` on the Android IDE toolbar.
2. Choose an application name as shown in Figure 3.15. The application name is the “friendly” name of the application and the name shown with the icon on the

application launcher. Name the application `My First Android App`. This will automatically create a project name of `MyFirstAndroidApp`, but you are free to change this to a name of your choosing.

3. We should also change the package name, using reverse domain name notation (http://en.wikipedia.org/wiki/Reverse_domain_name_notation), to `com.introtoandroid.myfirstandroidapp`. The Minimum Required SDK version should be the first SDK API level you plan to target. Because our application will be compatible with just about any Android device, you can set this number low (such as to 4 to represent Android 1.6) or at the target API level to avoid any warnings in the Android IDE. Make sure you set the minimum SDK version to encompass any test devices you have available so you can successfully install the application on them. The default options are just fine for our example. Click `Next`.
4. Keep the rest of the `New Android Application` settings at their defaults, unless you want to change the directory of where the source files will be stored. Click `Next` (see Figure 3.16).

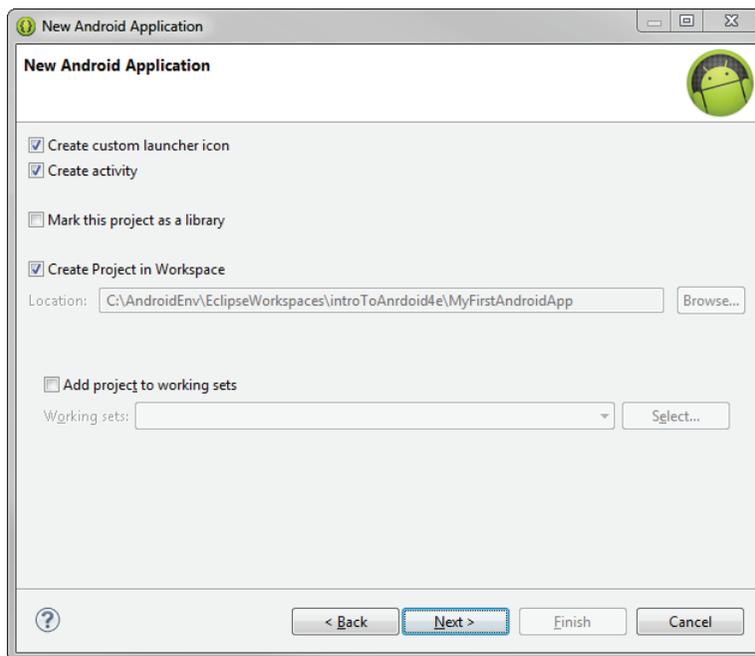


Figure 3.16 Configuring Android project options.

5. Leave the `Configure Launcher Icon` settings at their defaults. This option screen would allow us to define how our application launcher icon appears, but for this example, we will use the standard icon set included with the Android SDK. Choose `Next` (see Figure 3.17).
6. The `Create Activity` wizard allows us to include a default launch activity by type. We will leave the settings as is and choose `Next` (see Figure 3.18).
7. Choose an `Activity Name`. Call this `Activity` class `MyFirstAndroidApp` `Activity`. The `Layout Name` should automatically change to a name resembling what you just entered. Finally, click the `Finish` button (see Figure 3.19) to create the application.
8. The Android IDE should now display our first application created using the wizard with our layout file open and ready for editing (see Figure 3.20).

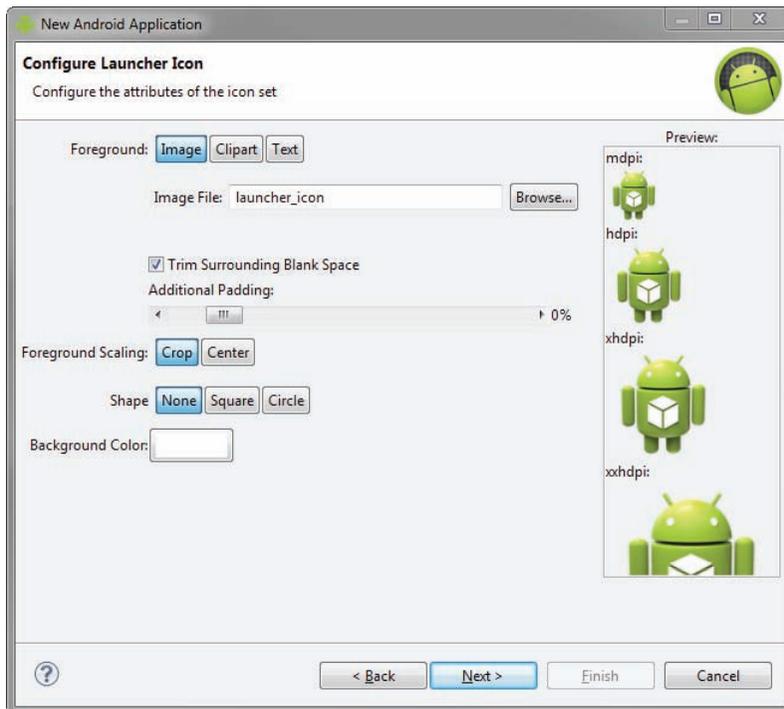


Figure 3.17 Configuring the launcher icon for our Android project.

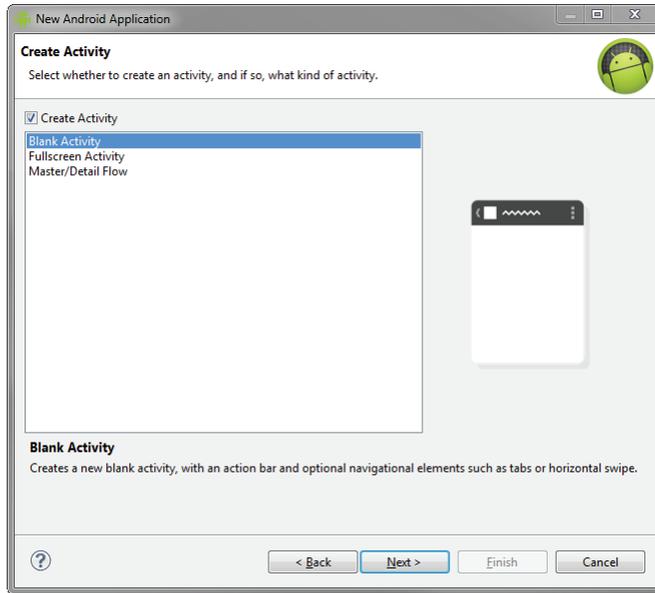


Figure 3.18 Creating an Activity for our Android project.

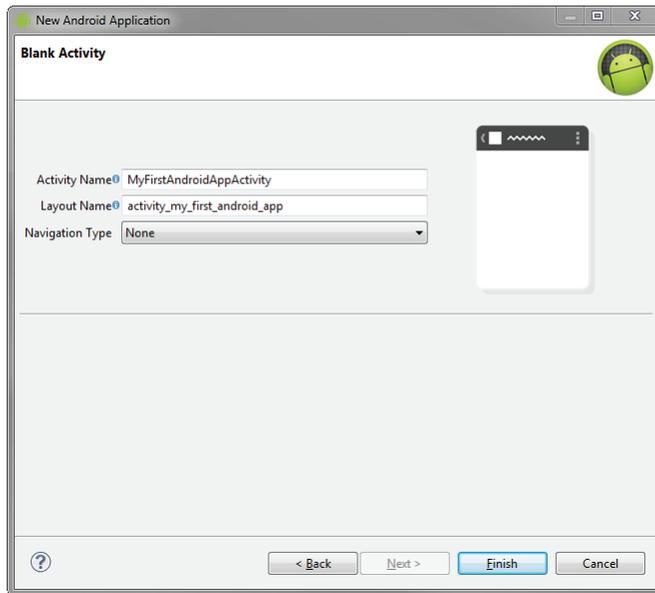


Figure 3.19 Choosing an Activity Name.

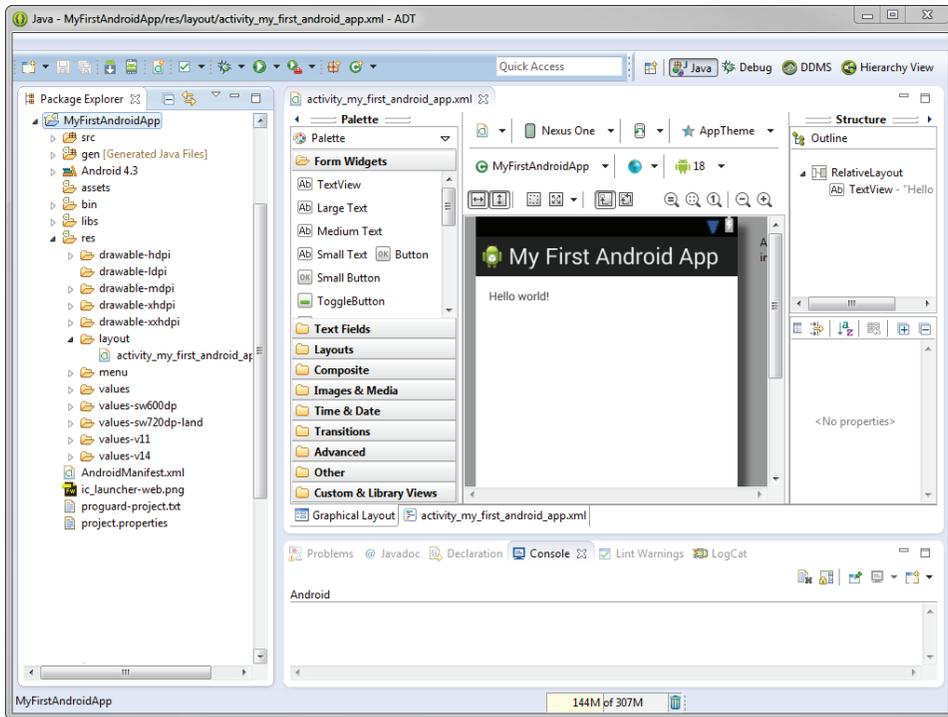


Figure 3.20 Our first application created with the wizard.

Core Files and Directories of the Android Application

Every Android application has a set of core files that are created and used to define the functionality of the application. The following files are created by default with a new Android application:

- **AndroidManifest.xml**—the central configuration file for the application. It defines your application’s capabilities and permissions as well as how it runs.
- **ic_launcher-web.png**—This is a high-resolution 32-bit 512 × 512 PNG application icon that is required and used for your application listing in the Google Play store. The size of this icon should not exceed 1024KB.
- **proguard-project.txt**—a generated build file used by the Android IDE and ProGuard. Edit this file to configure your code optimization and obfuscation settings for release builds.
- **project.properties**—a generated build file used by the Android IDE. It defines your application’s build target and other build system options, as required. Do not edit this file.

- **/src**—required folder for all source code.
- **/src/com/introtoandroid/myfirstandroidapp/MyFirstAndroidAppActivity.java**—main entry point to this application, named `MyFirstAndroidAppActivity`. This activity has been defined as the default launch activity in the Android manifest file.
- **/gen**—required folder for all autogenerated files.
- **/gen/com/introtoandroid/myfirstandroidapp/BuildConfig.java**—a generated source file used when debugging your applications. Do not edit this file.
- **/gen/com/introtoandroid/myfirstandroidapp/R.java**—a generated resource management source file. Do not edit this file.
- **/assets**—required folder where uncompiled file resources can be included in the project. Application assets are pieces of application data (files, directories) that you do not want managed as application resources.
- **/bin**—folder for creating autogenerated files for producing your application's APK file.
- **/libs**—folder for including any `.jar` library projects.
- **/libs/android-support-v4.jar**—This support library can be added to your projects to bring newer Android APIs to older devices running older versions of Android.
- **/res**—required folder where all application resources are managed. Application resources include animations, drawable graphics, layout files, datalike strings and numbers, and raw files.
- **/res/drawable-***—Application icon graphics resources are included in several sizes for different device screen resolutions.
- **/res/layout**—required folder that comprises one or more layout resource files, each file managing a different UI or App Widget layout for your application.
- **/res/layout/activity_my_first_android_app.xml**—layout resource file used by `MyFirstAndroidAppActivity` to organize controls on the main application screen.
- **/res/menu**—folder for including XML files for defining Android application menus.
- **/res/menu/my_first_android_app.xml**—menu resource file used by `MyFirstAndroidAppActivity` defining a menu item for Settings.
- **/res/values***—folders for including XML files for defining Android application dimensions, strings, and styles.
- **/res/values/dimens.xml**—dimension resource file used by `MyFirstAndroidAppActivity` defining default screen margins.
- **/res/values/strings.xml**—string resource file used by `MyFirstAndroidAppActivity` defining string variables that may be reused throughout the application.

- `/res/values/styles.xml`—style resource file used by `MyFirstAndroidAppActivity` to define the application theme.
- `/res/values-sw600dp/dimens.xml`—dimension resource file for overriding the `res/values/dimens.xml` for defining dimensions for 7-inch tablets.
- `/res/values-sw720dp-land/dimens.xml`—dimension resource file for overriding the `res/values/dimens.xml` for defining dimensions for 10-inch tablets in landscape mode.
- `/res/values-v11/styles.xml`—style resource file for overriding the `res/values/styles.xml` for devices running Android with an API greater than or equal to 11.
- `/res/values-v14/styles.xml`—style resource file for overriding the `res/values/styles.xml` for devices running Android with an API greater than or equal to 14.

A number of other files are saved on disk as part of the Android IDE project in the workspace. However, the files and resource directories included in the list here are the important project files you will use on a regular basis.

Creating an AVD for Your Project

The next step is to create an AVD that describes what type of device you want to emulate when running the application. For this example, we can use the AVD we created for the Snake application. An AVD describes a device, not an application. Therefore, you can use the same AVD for multiple applications. You can also create similar AVDs with the same configuration but different data (such as different applications installed and different SD card contents).

Creating a Launch Configuration for Your Project

Next, you must create a Run and Debug launch configuration in the Android IDE to configure the circumstances under which the `MyFirstAndroidApp` application builds and launches. The launch configuration is where you configure the emulator options to use and the entry point for your application.

You can create Run configurations and Debug configurations separately, with different options for each. Begin by creating a Run configuration for the application. Follow these steps to create a basic Run configuration for the `MyFirstAndroidApp` application:

1. Choose Run, Run Configurations... (or right-click the project and choose Run As).
2. Double-click Android Application.
3. Name your configuration `MyFirstAndroidAppRunConfig`.
4. Choose the project by clicking the Browse button and choosing the `MyFirstAndroidApp` project.

5. Switch to the `Target` tab and set the `Deployment Target Selection Mode` to `Always prompt to pick device`.
6. Click `Apply` and then click `Close`.



Tip

If you leave the `Deployment Target Selection Mode` set to `Automatic` when you choose `Run` or `Debug` in the Android IDE, your application is automatically installed and run on the device if the device is plugged in. Otherwise, the application starts in the emulator with the specified AVD. By choosing `Always prompt to pick device`, you are always prompted for whether (a) you want your application to be launched in an existing emulator; (b) you want your application to be launched in a new emulator instance and are allowed to specify an AVD; or (c) you want your application to be launched on the device (if it's plugged in). If any emulator is already running, the device is then plugged in, and the mode is set to `Automatic`, you see this same prompt, too.

Now create a `Debug` configuration for the application. This process is similar to creating a `Run` configuration. Follow these steps to create a basic `Debug` configuration for the `MyFirstAndroidApp` application:

1. Choose `Run, Debug Configurations...` (or right-click the project and choose `Debug As`).
2. Double-click `Android Application`.
3. Name your configuration `MyFirstAndroidAppDebugConfig`.
4. Choose the project by clicking the `Browse` button and choosing the `MyFirstAndroidApp` project.
5. Switch to the `Target` tab and set the `Deployment Target Selection Mode` to `Always prompt to pick device`.
6. Click `Apply` and then click `Close`.

You now have a `Debug` configuration for your application.

Running Your Android Application in the Emulator

Now you can run the `MyFirstAndroidApp` application using the following steps:

1. Choose the `Run As` icon drop-down menu on the toolbar (.
2. Pull the drop-down menu and choose the `Run` configuration you created. (If you do not see it listed, choose the `Run Configurations...` item and select the appropriate configuration. The `Run` configuration shows up on this drop-down list the next time you run the configuration.)

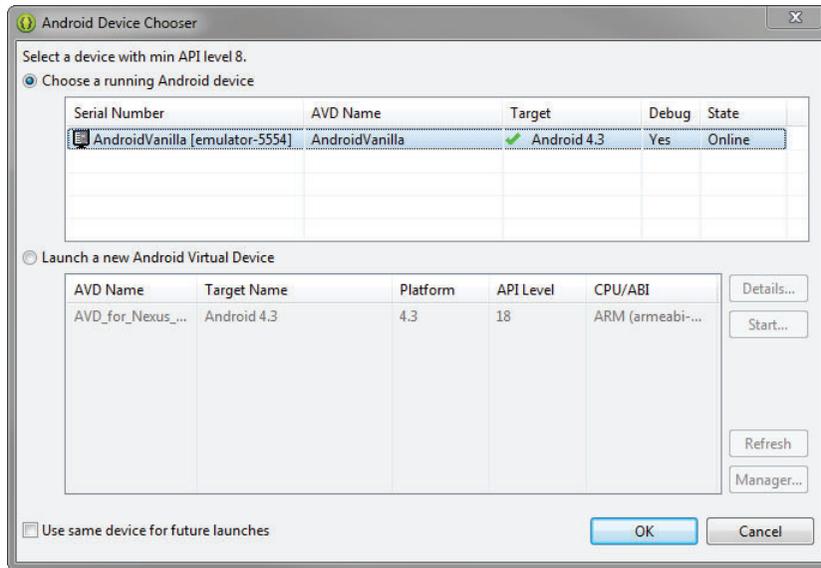


Figure 3.21 Manually choosing a deployment target selection mode.

3. Because you chose the `Always` prompt to pick device selection mode, you are now prompted for your emulator instance. Change the selection to `Launch a New Android Virtual Device` and then select the AVD you created. Here, you can choose from an already-running emulator or launch a new instance with an AVD that is compatible with the application settings, as shown in Figure 3.21.
4. The Android emulator starts up, which might take a moment.
5. Click the `Menu` button or push the slider to the right to unlock the emulator.
6. The application starts, as shown in Figure 3.22.
7. Click the `Back` button in the emulator to end the application, or click `Home` to suspend it.
8. Click the `All Apps` button (see Figure 3.23) found in the `Favorites` tray to browse all installed applications from the `All Apps` screen.
9. Your screen should now look something like Figure 3.24. Click the `My First Android App` icon to launch the application again.



Figure 3.22 My First Android App running in the emulator.



Figure 3.23 The All Apps button.

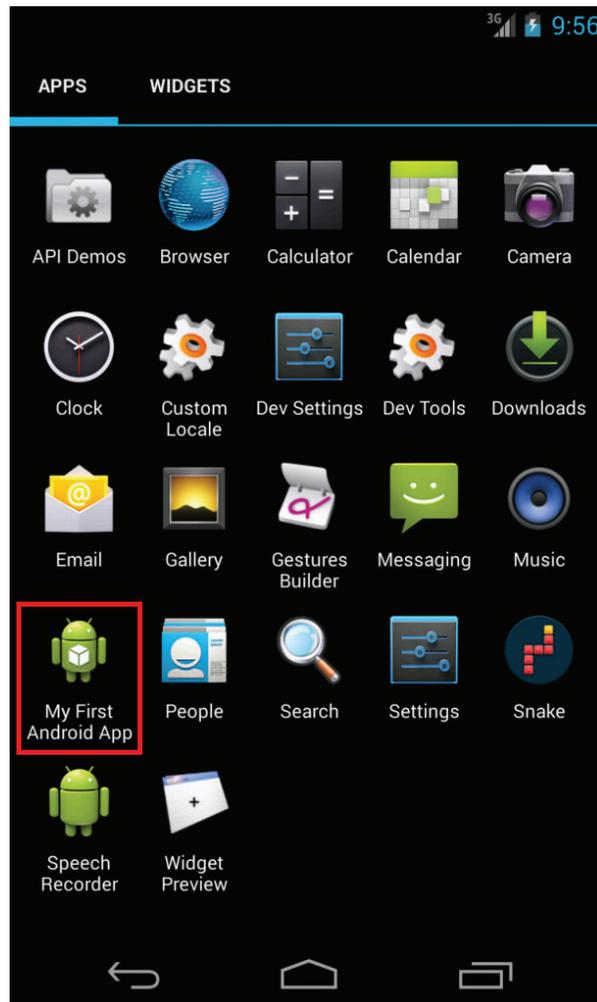


Figure 3.24 The My First Android App icon shown in the All Apps screen.

Debugging Your Android Application in the Emulator

Before going any further, you need to become familiar with debugging in the emulator. To illustrate some useful debugging tools, let's manufacture an error in the `My First Android App`.

In your project, edit the source file called `MyFirstAndroidAppActivity.java`. Create a new method called `forceError()` in your class and make a call to this method in your `Activity` class's `onCreate()` method. The `forceError()` method forces a new unhandled error in your application.

The `forceError()` method should look something like this:

```
public void forceError() {  
    if(true) {  
        throw new Error("Whoops");  
    }  
}
```

It's probably helpful at this point to run the application and watch what happens. Do this using the `Run` configuration first. In the emulator, you see that the application has stopped unexpectedly. You are prompted by a dialog that enables you to force the application to close, as shown in Figure 3.25.

Shut down the application but keep the emulator running. Now it's time to debug. You can debug the `MyFirstAndroidApp` application using the following steps:

1. Choose the `Debug As` icon drop-down menu on the toolbar.
2. Pull the drop-down menu and choose the `Debug` configuration you created. (If you do not see it listed, choose the `Debug Configurations...` item and select the appropriate configuration. The `Debug` configuration shows up on this drop-down list the next time you run the configuration.)
3. Continue as you did with the `Run` configuration and choose the appropriate AVD, and then launch the emulator again, unlocking it if needed.

It takes a moment for the debugger to attach. If this is the first time you've debugged an Android application, you may need to click through some dialogs, such as the one shown in Figure 3.26, the first time your application attaches to the debugger.

In the Android IDE, use the `Debug` perspective to set breakpoints, step through code, and watch the `LogCat` logging information about your application. This time, when the application fails, you can determine the cause using the debugger. You might need to click through several dialogs as you set up to debug within the Android IDE. If you allow the application to continue after throwing the exception, you can examine the results in the `Debug` perspective of the Android IDE. If you examine the `LogCat` logging pane, you see that your application was forced to exit due to an unhandled exception (see Figure 3.27).

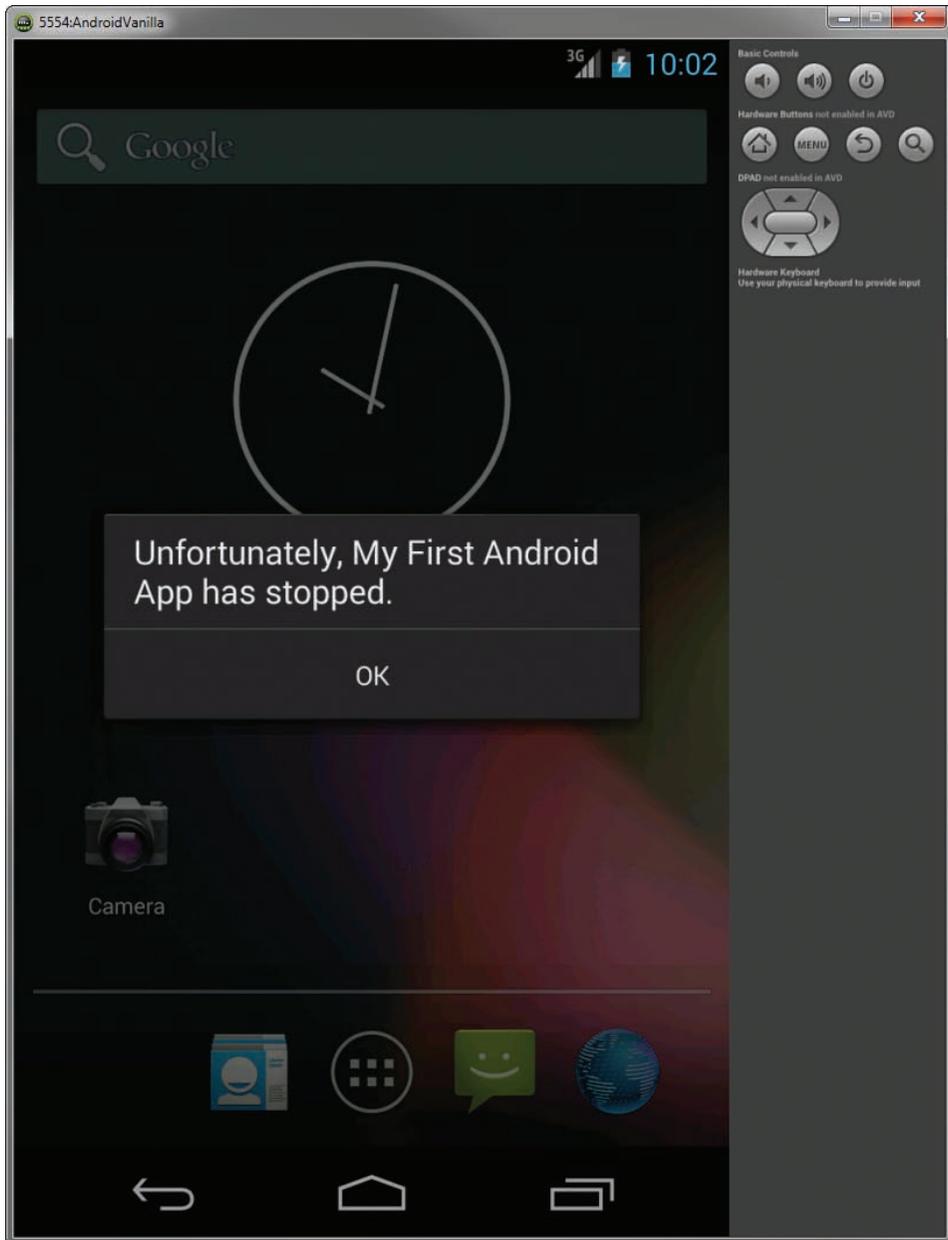


Figure 3.25 My First Android App crashing gracefully.

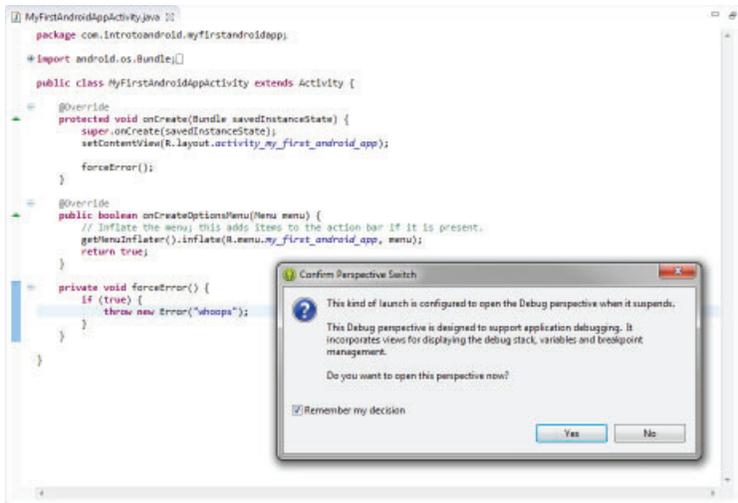


Figure 3.26 Switching to Debug perspective for Android emulator debugging.

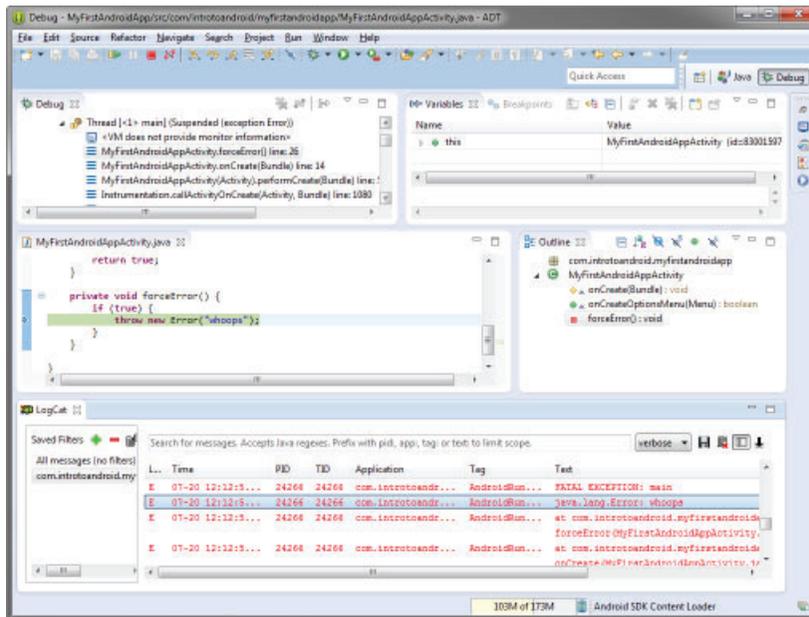


Figure 3.27 Debugging MyFirstAndroidApp in the Android IDE.

Specifically, there's a red `AndroidRuntime` error: `java.lang.Error: whoops`. Back in the emulator, click the `Force Close` button. Now set a breakpoint on the `forceError()` method by right-clicking the left side of the line of code and choosing `Toggle Breakpoint` (or double-clicking).



Tip

In the Android IDE, you can step through code using `Step Into` (F5), `Step Over` (F6), `Step Return` (F7), or `Resume` (F8). On Mac OS X, you might find that the F8 key is mapped globally. If you want to use the keyboard convenience command, you might want to change the keyboard mapping in the Android IDE by choosing `Window, Preferences, General, Keys` and then finding the entry for `Resume` and changing it to something else. Alternatively, you can change the Mac OS X global mapping by going to `System Preferences, Keyboard & Mouse, Keyboard Shortcuts` and then changing the mapping for F8 to something else.

In the emulator, restart your application and step through your code. You see that your application has thrown the exception, and then the exception shows up in the `Variable Browser` pane of the `Debug` perspective. Expanding its contents shows that it is the “Whoops” error.

This is a great time to crash your application repeatedly and get used to the controls. While you're at it, switch over to the `DDMS` perspective. Note that the emulator has a list of processes running on the device, such as `system_process` and `com.android.phone`. If you launch `MyFirstAndroidApp`, you see `com.introtoandroid.myfirstandroidapp` show up as a process on the emulator listing. Force the app to close because it crashes, and note that it disappears from the process list. You can use `DDMS` to kill processes, inspect threads and the heap, and access the phone file system.

Adding Logging Support to Your Android Application

Before you start diving into the various features of the Android SDK, you should familiarize yourself with logging, a valuable resource for debugging and learning Android. Android logging features are in the `Log` class of the `android.util` package. See Table 3.1 for some helpful methods in the `android.util.Log` class.

Table 3.1 Commonly Used Logging Methods

Method	Purpose
<code>Log.e()</code>	Log errors
<code>Log.w()</code>	Log warnings
<code>Log.i()</code>	Log informational messages
<code>Log.d()</code>	Log debug messages
<code>Log.v()</code>	Log verbose messages

To add logging support to `MyFirstAndroidApp`, edit the file `MyFirstAndroidApp.java`. First, you must add the appropriate import statement for the `Log` class:

```
import android.util.Log;
```



Tip

To save time in the Android IDE, you can use the imported classes in your code and add the imports needed by hovering over the imported class name and choosing the `Add Imported Class QuickFix` option.

You can also use the `Organize Imports` command (`Ctrl+Shift+O` in Windows or `Command+Shift+O` on a Mac) to have the Android IDE automatically organize your imports. This removes unused imports and adds new ones for packages used but not imported. If a naming conflict arises, as it often does with the `Log` class, you can choose the package you intended to use.

Next, within the `MyFirstAndroidApp` class, declare a constant string that you use to tag all logging messages from this class. You can use the `LogCat` utility within the Android IDE to filter your logging messages to this `DEBUG_TAG` tag string:

```
private static final String DEBUG_TAG= "MyFirstAppLogging";
```

Now, within the `onCreate()` method, you can log something informational:

```
Log.i(DEBUG_TAG,
    "In the onCreate() method of the MyFirstAndroidAppActivity Class");
```

While you're here, you must comment out your previous `forceError()` call so that your application doesn't fail. Now you're ready to run `MyFirstAndroidApp`. Save your work and debug it in the emulator. Notice that your logging messages appear in the `LogCat` listing, with the `Tag` field `MyFirstAppLogging` (see Figure 3.28).

Adding Some Media Support to Your Application

Next, let's add some pizzazz to `MyFirstAndroidApp` by having the application play an MP3 music file. Android media player features are found in the `MediaPlayer` class of the `android.media` package.

You can create `MediaPlayer` objects from existing application resources or by specifying a target file using a `URI`. For simplicity, we begin by accessing an MP3 using the `Uri` class from the `android.net` package.

Table 3.2 shows some methods used in the `android.media.MediaPlayer` and `android.net.Uri` classes.

To add MP3 playback support to `MyFirstAndroidApp`, edit the file `MyFirstAndroidApp.java`. First, you must add the appropriate import statements for the `MediaPlayer` class:

```
import android.media.MediaPlayer;
import android.net.Uri;
```

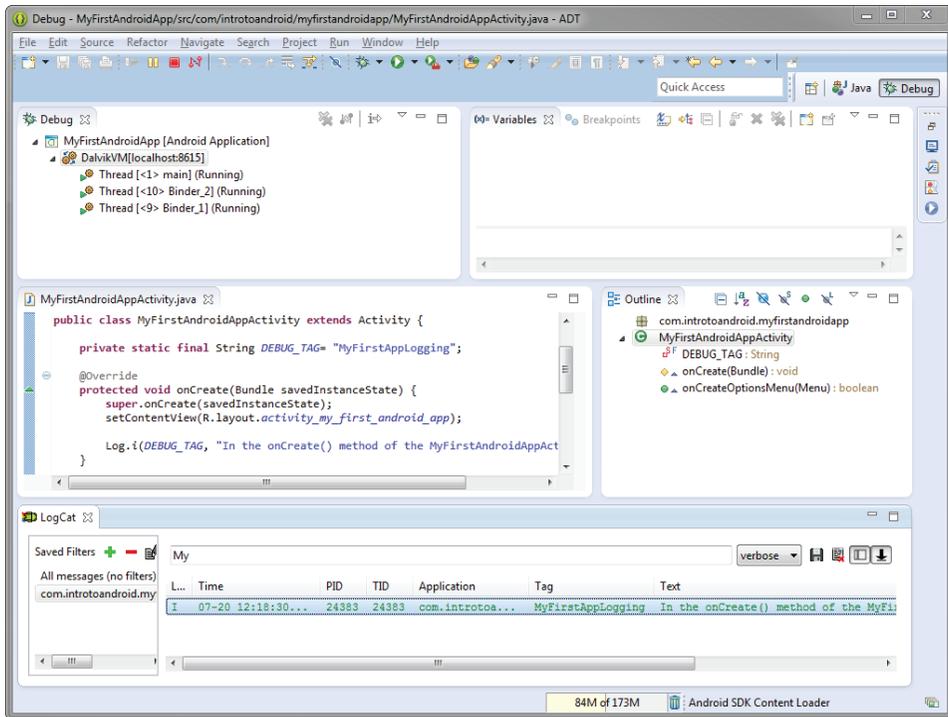


Figure 3.28 A LogCat log for MyFirstAndroidApp.

Table 3.2 Commonly Used MediaPlayer and Uri Parsing Methods

Method	Purpose
<code>MediaPlayer.create()</code>	Creates a new media player with a given target to play
<code>MediaPlayer.start()</code>	Starts media playback
<code>MediaPlayer.stop()</code>	Stops media playback
<code>MediaPlayer.release()</code>	Releases the media player resources
<code>Uri.parse()</code>	Instantiates a <code>Uri</code> object from an appropriately formatted URI address

Next, within the `MyFirstAndroidApp` class, declare a member variable for your `MediaPlayer` object:

```
private MediaPlayer mp;
```

Now, create a new method called `playMusicFromWeb()` in your class and make a call to this method in your `onCreate()` method. The `playMusicFromWeb()` method creates a

valid `Uri` object, creates a `MediaPlayer` object, and starts the MP3 playing. If the operation should fail for some reason, the method logs a custom error with your logging tag. The `playMusicFromWeb()` method should look something like this:

```
public void playMusicFromWeb() {
    try {
        Uri file = Uri.parse("http://www.perlgurl.org/podcast/archives"
            + "/podcasts/PerlgurlPromo.mp3");
        mp = MediaPlayer.create(this, file);
        mp.start();
    }
    catch (Exception e) {
        Log.e(DEBUG_TAG, "Player failed", e);
    }
}
```

As of Android 4.2.2 (API Level 17), using the `MediaPlayer` class to access media content on the Web requires the `INTERNET` permission to be registered in the application's Android manifest file. Finally, your application requires special permissions to access location-based functionality. You must register this permission in your `AndroidManifest.xml` file. To add permissions to your application, perform the following steps:

1. Double-click the `AndroidManifest.xml` file.
2. Switch to the `Permissions` tab.
3. Click the `Add` button and choose `Uses Permission`.
4. In the right pane, select `android.permission.INTERNET` (see Figure 3.29).
5. Save the file.

Later on, you'll learn all about the various `Activity` states and callbacks that could contain portions of the `playMusicFromWeb()` method. For now, know that the `onCreate()` method is called every time the user navigates to the `Activity` (forward or backward) and whenever he or she rotates the screen or causes other device configuration changes. This doesn't cover all cases but will work well enough for this example.

And finally, you want to cleanly exit when the application shuts down. To do this, you need to override the `onStop()` method of your `Activity` class and stop the `MediaPlayer` object and release its resources. The `onStop()` method should look something like this:

```
protected void onStop() {
    if (mp != null) {
        mp.stop();
    }
}
```

```
        mp.release();
    }
    super.onStop();
}
```



Tip

In the Android IDE, you can right-click within the class and choose `Source` (or press `Alt+Shift+S`). Choose the option `Override/Implement Methods` and select the `onStop()` method.

Now, if you run `MyFirstAndroidApp` in the emulator (and you have an Internet connection to grab the data found at the URI location), your application plays the MP3. When you shut down the application, the `MediaPlayer` is stopped and released appropriately.

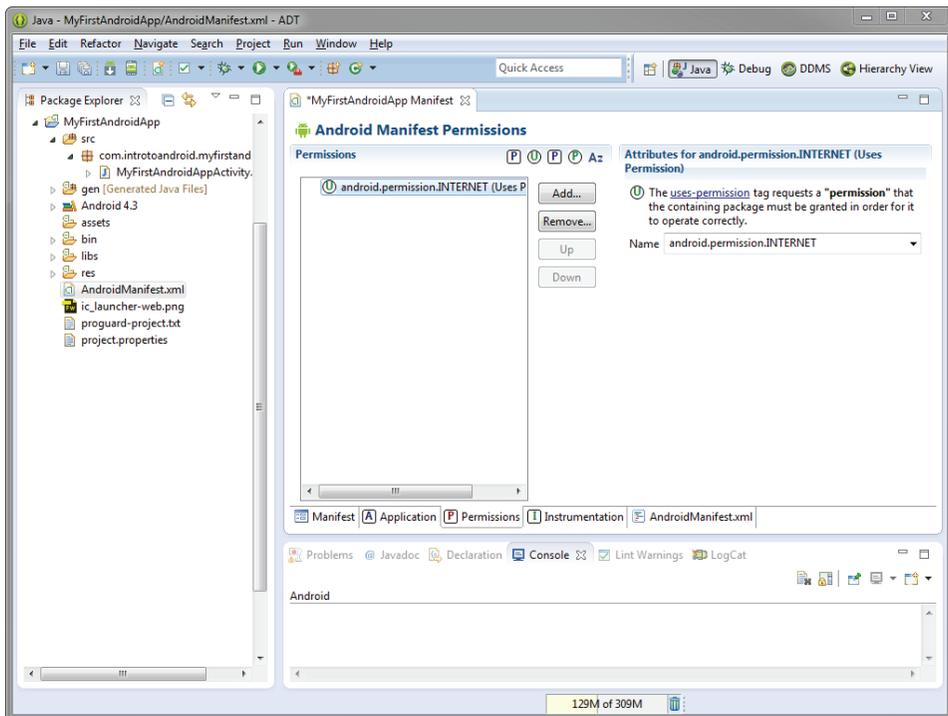


Figure 3.29 Adding the INTERNET permission in the manifest.

Adding Location-Based Services to Your Application

Your application knows how to say “Hello” and play some music, but it doesn’t know where it’s located. Now is a good time to become familiar with some simple location-based calls to get the GPS coordinates. To have some fun with location-based services and map integration, you will use some of the Google applications available on typical Android devices—specifically, the Maps application. You do not need to create another AVD, because you included the Google APIs as part of the target for the AVD you already created.

Configuring the Location of the Emulator

The emulator does not have location sensors, so the first thing you need to do is seed your emulator with some GPS coordinates. You can find the exact steps for how to do this in Appendix B, “Quick-Start Guide: The Android Emulator,” in the section “Configuring the GPS Location of the Emulator.” After you have configured the location of your emulator, the Maps application should display your simulated location, as shown in Figure 3.30. Make sure that the location icon (📍) is showing, which is indicative that the location settings have been enabled on the AVD.



Warning

If you do not see the location icon presented in the status bar, this means that the location is not yet activated and requires configuring within the AVD.

Your emulator now has a simulated location: Yosemite Valley!

Finding the Last Known Location

To add location support to `MyFirstAndroidApp`, edit the file `MyFirstAndroidApp.java`. First, you must add the appropriate import statements:

```
import android.location.Location;
import android.location.LocationManager;
```

Now, create a new method called `getLocation()` in your class and make a call to this method in your `onCreate()` method. The `getLocation()` method gets the last known location on the device and logs it as an informational message. If the operation fails for some reason, the method logs an error.

The `getLocation()` method should look something like this:

```
public void getLocation() {
    try {
        LocationManager locMgr = (LocationManager)
            this.getSystemService(LOCATION_SERVICE);
        Location recentLoc = locMgr.
            getLastKnownLocation(LocationManager.GPS_PROVIDER);
        Log.i(DEBUG_TAG, "loc: " + recentLoc.toString());
    }
}
```

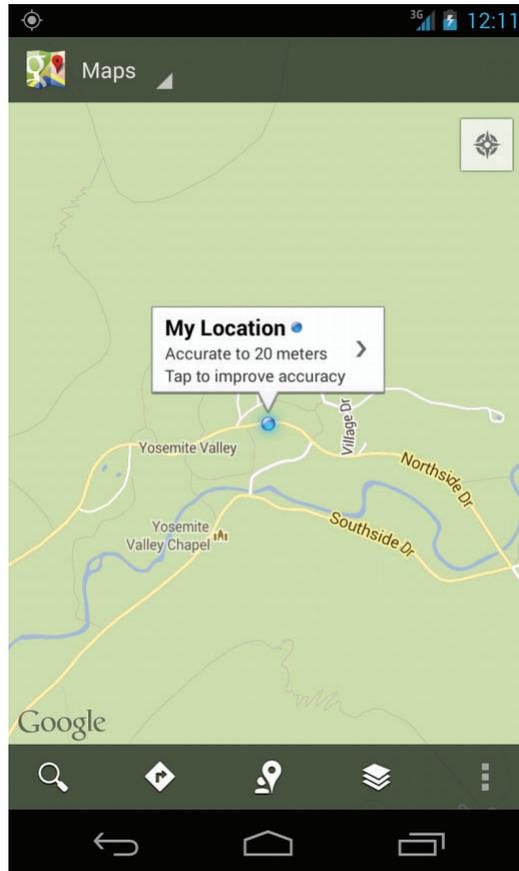


Figure 3.30 Setting the location of the emulator to Yosemite Valley.

```
}  
catch (Exception e) {  
    Log.e(DEBUG_TAG, "Location failed", e);  
}  
}  
}
```

Finally, your application requires special permission to access location-based functionality. You must register this permission in your `AndroidManifest.xml` file. To add location-based service permissions to your application, perform the following steps:

1. Double-click the `AndroidManifest.xml` file.
2. Switch to the `Permissions` tab.

3. Click the `Add` button and choose `Uses Permission`.
4. In the right pane, select `android.permission.ACCESS_FINE_LOCATION`.
5. Save the file.

Now, if you run `My First Android App` in the emulator, your application logs the GPS coordinates you provided to the emulator as an informational message, viewable in the `LogCat` pane of the Android IDE.

Debugging Your Application on Hardware

You have mastered running applications in the emulator. Now let's put the application on real hardware. This section discusses how to install the application on a Nexus 4 device with Android 4.3. To learn how to install on a different device or different Android version, read <http://d.android.com/tools/device.html>.

Connect an Android device to your computer via USB and relaunch the `Debug` configuration of the application. Because you chose the `Always prompt to pick device Deployment Target Selection Mode` for the configuration, you should now see a real Android device listed as an option in the `Android Device Chooser` (see Figure 3.31).

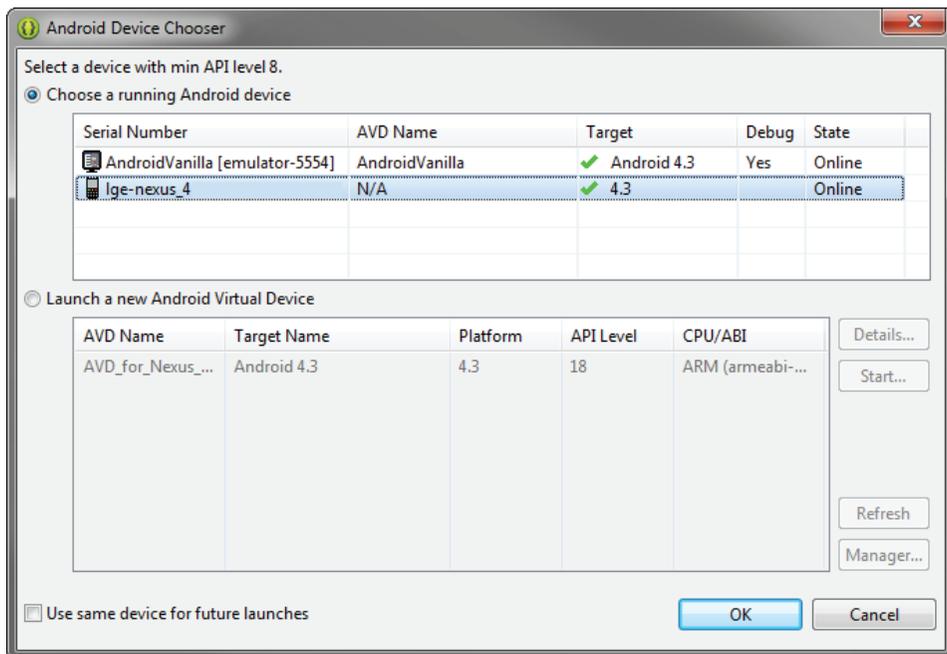


Figure 3.31 Android Device Chooser with USB-connected Android device.

Choose the Android device as your target, and you see that the `My First Android App` application gets loaded onto the Android device and launched, just as before. Provided you have enabled the development debugging options on the device, you can debug the application here as well. To allow USB debugging, go to `Settings, Developer options`, and under `Debugging`, choose `USB debugging`. A dialog prompt will appear (see Figure 3.32) requesting that USB debugging be allowed. Click `OK` to allow debugging.

Once the USB-connected Android device is recognized, you may be prompted with another dialog asking you to confirm the development computer's RSA key fingerprint. If so, select the option `Always allow from this computer` and click `OK` (see Figure 3.33).

Once enabled, you can tell that the device is actively using a USB debugging connection because a little Android bug-like icon appears in the status bar (🐞). Figure 3.34 shows a screenshot of the application running on a real device (in this case, a smartphone running Android 4.3).



Figure 3.32 Allowing USB debugging.



Figure 3.33 Remembering the computer's RSA key fingerprint.

Debugging on the device is much the same as debugging on the emulator, but with a couple of exceptions. You cannot use the emulator controls to do things such as send an SMS or configure the location to the device, but you can perform real actions (true SMS, actual location data) instead.

Summary

This chapter showed you how to add, build, run, and debug Android projects using the Android IDE. You started by installing the sample applications from within the Android IDE. You then began testing your development environment using a sample application from the Android SDK, and then you created a new Android application from scratch using the Android IDE. You also learned how to make some quick modifications to the application, demonstrating some exciting Android features you will learn about in future chapters.



Figure 3.34 My First Android App running on Android device hardware.

In the next few chapters, you will learn about the tools available for use in developing Android applications and then focus on the finer points about defining your Android application using the application manifest file. You will also learn how to organize your application resources, such as images and strings, for use within your application.

Quiz Questions

1. What are the benefits of choosing the Snapshot feature listed under the Emulation Options section of the AVD creation wizard?
2. What do the e, w, i, v, d letters stand for in relation to the `android.util.Log` class, for example, `Log.e()`?
3. What are the Debug breakpoint keyboard shortcuts for Step Into, Step Over, Step Return, and Resume?

4. What is the keyboard shortcut for organizing imports?
5. What is the keyboard shortcut for toggling a breakpoint in the Android IDE?
6. What is the keyboard shortcut for Override/Implement Methods in the Android IDE?

Exercises

1. Create a Nexus 7 AVD using the preconfigured device definitions.
2. Describe the purpose of the Minimum Required SDK, Target SDK, and Compile With options listed in the Android Application Project creation wizard.
3. Found in the Android Application Project creation wizard, describe the difference between a Blank Activity and a Fullscreen Activity.
4. Perform the steps for configuring the Run or Debug configurations of the `MyFirstAndroidApp` to launch the application on all compatible devices/AVDs when running or debugging. Write down each of the steps taken in order.
5. Create a new Android Application Project with a new Launcher Icon. For the Launcher Icon, increase the padding to 50%, change the Foreground Scaling to Center, give it a shape of Circle, and change the icon's background color to blue. Perform any other necessary steps required in order to create the application.
6. There are three Navigation Types available when creating a Blank Activity. Create a new Android Application Project for each Navigation Type available to see what each option provides. Perform any other necessary steps required in order to create the application.

References and More Information

Android SDK Reference regarding the application Activity class:

<http://d.android.com/reference/android/app/Activity.html>

Android SDK Reference regarding the application Log class:

<http://d.android.com/reference/android/util/Log.html>

Android SDK Reference regarding the application MediaPlayer class:

<http://d.android.com/reference/android/media/MediaPlayer.html>

Android SDK Reference regarding the application Uri class:

<http://d.android.com/reference/android/net/Uri.html>

Android SDK Reference regarding the application LocationManager class:

<http://d.android.com/reference/android/location/LocationManager.html>

Android Tools: “Using Hardware Devices”:

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

Android Resources: “Common Tasks and How to Do Them in Android”:

<http://d.android.com/guide/faq/commontasks.html>

Android sample code:

<http://d.android.com/tools/samples/index.html>

Index

A

About screen, 100, 380

AbsoluteLayout class, 334

AccessibilityManager, 35

AccessibilityService APIs, 457

Accounts

Developer Console, 461, 468

Google Wallet Merchant, 461

publishing to Google Play, 468–471

third-party APIs, 45–46

Action, intent, 108

Action bars, 411–415

Action buttons, 411–413

Action overflow menu, 412–415

**ACTION_BATTERY_LOW, broadcast
intent, 111**

ActionBar class, 248

Actions

action bars

action buttons, 411–412

action overflow, 412–415

ActionBar compatibility, 415

application icon, 411

contextual action mode, 415

overview of, 411

View control, 411

from within application's content, 416

dialogs, 415

menus, 410–411

Activities

- avoiding killing, 103–104
- callbacks
 - destroying static data, 104–105
 - initializing static activity, 102
 - initializing/retrieving data, 103
 - saving state into bundle, 104
 - state/resource management, 101
 - stopping/saving/releasing data, 103
- creating, 71–72
- creating layout using XML resources, 210
- creating layouts programmatically, 211–213
- customizing behavior in back stack, 404
- definition of, 97
- editing manifest file, 117
- fragments
 - attaching/detaching, 246–247
 - defined, 105
 - MP3 music player example, 105–106
 - organizing components, 98, 105–106
- indicators
 - activity bar and circles, 202
 - ProgressBar, 199–202
 - SeekBar, 202–203
- intents
 - with action/data, 108
 - activity transitions with, 106–107
 - application navigation, 110
 - broadcasting, 111–112
 - defined, 108
 - Google application, 109
 - launching activities by class name, 107–108

- launching external activities, 108–109
- overview of, 106
- passing additional information, 109
- receiving, 111–112
- transitioning between, 106–107
- lateral navigation for, 400–401
- life cycle of, 100–102
- naming, 71–72
- navigation. *see* **Navigation**
- overview of, 99–100
- testing mimicking real-world, 426–427

Activity bars, 202**Activity circles, 202****Activity class**

- callbacks
 - method stubs for, 101
 - onCreate(), 102
 - onDestroy(), 104–105
 - onPause(), 103
 - onResume(), 103
 - onSaveInstanceState(), 104
- fragments
 - within components, 105–106
 - defining, 257–258
 - design, 249–250
- overview of, 99–100
- preferences
 - private, 282
 - shared, 282–283
- screen workflow, 244–245

Activity_main.xml layout, 210**ActivityInstrumentationTestCase2<T> class, 455****Ad revenue, 462**

Adapters

- AdapterView controls, 233
- in autocompletion, 187–188
- binding data to AdapterView, 235–236
- creating, 233
- handling selection events, 236–237
- in spinner controls, 188
- using ArrayAdapter, 234
- using CursorAdapter, 234–235
- using ListActivity, 237

AdapterView class, 233, 235–237**ADB (Android Debug Bridge) tool**

- defined, 434
- mobile development, 393
- overview of, 491

Add Imported Class QuickFix option, 84**AddOnBackStack ChangedListener() method, 254****AddToBackStack() method, 401****AddView() method, 214****AddWord() method, 322****Adobe AIR, 33****ADT (Android Developer Tools)**

- as Android Eclipse plug-in, 25
- as Android IDE, 25
- defined, 25
- developing applications faster with, 485
- resource editors, 491–493
- running first test using Android IDE, 450
- upgrading Android SDK, 42

ADT Bundle

- Android IDE and, 38, 47, 485, 547
- Android SDK in, 37–38, 43
- as development environment in this book, 5

- editing manifest file, 116
- emulator in, 505
- functions, 47
- installation of, 39
- updates, 37, 55

Alert messages, 418**AlertDialog class, 266–267, 270–271****Aliases, 166****All Apps screen**

- customizing emulator with widgets, 524–525
- entry navigation in, 400
- launching applications in emulator, 68
- running application in emulator, 76, 79

Allocation Tracker

- memory use and, 532
- monitoring memory with DDMS, 534–536

Allow mock locations, 42**Alternative resources**

- advantages of, 338
- caution when using for layout, 170
- for compatibility
 - for different orientations, 345–346
 - organizing efficiently, 345–347
 - organizing with qualifiers, 339–344
 - resolving, 338–339
 - smoothing differences between devices, 333
 - using programmatically, 345
- default vs., 141–142
- designing tablets, 348
- naming, 338–339

Amazon Appstore for Android

- distributing applications through, 27
- as distribution channel, 375
- publishing to, 479

Ancestral navigation, 401, 403

Anddev.org, 6

Android

costs, 23, 27

Developer website

ADT Bundle updates, 55

as resource, 6

SDK updates, 37, 55

Support Package updates, 260

manifest file. *see* **Manifest file**

mobile software development history

“The Brick,” 13–15

proprietary mobile platforms,
17–18

WAP, 15–17

way back when, 11–13

Open Handset Alliance

advantages of, 22

application development, 21–22

forming, 19

Google goes wireless, 18–19

manufacturers designing devices,
19–20

marketplace, 22–23

mobile operators, 20–21

platform

application framework, 34–35

Google services, 45–46

native and third-party applications,
33–34

packages, 34

programming language choices,
32–33

security and permissions, 31–33

services, 45

underlying architecture, 29–31

platform differences

costs, 26–27

developing powerful
applications, 26

familiar tools, 25–26

free and open source, 25

free market, 27–28

growing platform, 28–29

next-generation, 23–25

overview of, 23

secure application integration,
26–27

third-party APIs, 45

WAP, 15–17

**Android Application Project creation wizard,
69–73**

Android command-line tool

Android Virtual Device Manager, 508

creating AndroidManifest.xml
file, 116

defined, 501

Android Debug Bridge (ADB) tool

defined, 434

mobile development, 393

overview of, 491

**Android Developer Tools. *see* ADT (Android
Developer Tools)**

Android Developers Blog, 382

Android Device Chooser, 90–92

Android IDE

building first application

core files/directories, 73–75

creating AVD, 75

creating new project, 69–73

debugging in emulator, 80–83

debugging on hardware, 90–92

keyboard mapping, 83

launch configuration, 75–76

launching AVD Manager, 61

location-based services, 88–90

- logging, 83–84
- media, 84–87
- running in emulator, 76–79
- stepping through code, 83
- choosing source control system, 360–370
- defined, 25
- development environment set up
 - ADT, 47
 - configuring, 37–38
 - installation process, 38–39
 - other IDEs, 38
 - SDK samples, downloading, 51–52
 - SDK upgrades, 42
- emulator
 - integration with, 489
 - launching with specific AVD, 512–515
- exporting/signing package files, 465–467
- as focus of this book, 38, 485
- fragments, 335
- layouts
 - default file for, 210
 - designing, 168–170
 - testing on emulator, 211
- manifest file
 - creating in, 116
 - editing, 116–121
- organizing workspace, tips
 - closing unwanted tabs, 550
 - customizing log filters, 551
 - repositioning tabs, 548
 - searching your project, 551
 - source control services, 547
 - tasks, 551–552
 - viewing two sections of same file, 550
 - windows, 548–551
- overview of, 25–26
- resolving build errors, 556
- resources
 - accessing programmatically, 142
 - adding to project, 138
 - creating menus, 162–163
 - designing layouts, 168–170
 - different types of, 147
 - setting simple values, 143–146
- testing applications
 - creating test project, 441–447
 - running first test, 450–453, 456
 - standalone tools, 435, 529–530
 - white-box testing, 429
- testing development environment
 - adding project to workspace, 57–60
 - adding samples with SDK Manager, 56–57
 - creating AVD, 61–62
 - launch configuration, 62–65
 - overview of, 55–56
 - running application in emulator, 66–68
- writing code in Java, tips
 - autocomplete, 552
 - formatting, 553
 - Javadoc style documentation, 556
 - new classes and methods, 552
 - organizing imports, 553
 - refactoring, 554
 - renaming, 553–554
 - reorganizing, 555

Android IDE, writing code in Java, tips
(*continued*)

- using Extract Local Variable tool, 554–555

- using Extract Method tool, 555

- using QuickFix, 555–556

Android manifest file editor, 492

Android Market. *see* Google Play

Android Project Wizard, 116

Android Studio, 26, 38, 491

Android Tools Project Site, 6

Android Virtual Device. *see* AVD (Android Virtual Device)

Android Virtual Device Manager

- creating AVD within IDE, 509–510

- creating new AVD, 61–63

- functions of, 48–49

- launching emulator from, 515–516

- loading emulator, 66

- using, 508–509

AndroidManifest.xml. *see also* Manifest file

- defined, 73

- location-based service permissions, 89–90

- overview of, 115

Android.util package, 83–84

Android.widget package

- App Widgets vs., 178

- Button class, 190

- layout

- FrameLayout control, 222

- GridLayout control, 228–230

- LinearLayout control, 218

- overview of, 178

- RelativeLayout control, 219

- Space view, 230

- TableLayout control, 226

- NumberPicker, 199

- Switch control, 194

- TextView control, 179

- Toast message, 192

- user interface controls, 177

Animation

- providing visual feedback, 418

- resources

- frame-by-frame, 160–161

- overview of, 159–160

- tweened, 161–162

Apache Software License (ASL/Apache 2), 25

API Guides, 486

ApiDemos utility, 52

APIs (application programming interfaces)

- billing, 384

- in documentation, 488

- in emulator console, 525

- legacy support, 5

- level for AVD, 51

- SDK license agreement for, 44

- third-party, 45–46

- undocumented, 34

.apk file, 465

App Widgets, 178, 524–525

Appending data, 303

Apple iPhone OS, 17, 21

Application tab, IDE manifest file resource editor, 117–118

Applications. *see also* Testing

- adding shortcut to emulator, 524

- anatomy of

- activities, 99–105

- context, 98–99

- fragments, 105–106

- intents, 106–112

- overview of, 97

- review, 112–113

- services, 110–111

- terminology, 97–98

- benefits of, 22
- building first
 - core files and directories, 73–75
 - creating AVD, 75
 - creating new project, 69–73
 - debugging in emulator, 80–83
 - debugging on hardware, 90–92
 - launch configuration, 75–76
 - location-based services, 88–90
 - logging support, 83–84
 - media support, 84–87
 - overview of, 68–69
 - running in emulator, 76–79
- compatibility
 - alternative resources for. *see* **Alternative resources**
 - targeting devices, 347–350
 - user interface design for, 333–338
- configuring using manifest file. *see* **Manifest file**
- designing
 - architectures, 371
 - avoiding mistakes, 388, 393
 - best practices, 379–380, 388–389
 - billing/revenue generation, 383–384
 - code diagnostics, 391–392
 - code reviews, 391
 - coding standards, 390
 - deploying, 374–375
 - development phase, 373
 - extensibility, 371–372
 - feasibility testing, 389
 - interoperability, 372–373
 - limitations of, 370–371
 - maintenance/extensibility, 371–372
 - maintenance/support, 375–377
 - quality guidelines, 384–385
 - security, 383
 - single device bugs, 392
 - software development process, 389
 - stability/responsiveness, 381–382
 - target markets, 375
 - testing applications, 373–374
 - third-party standards, 385
 - tools, 387–388, 393
 - updates/upgrades, 385–387
 - user demands, meeting, 380
 - user interfaces, 380–381
- developing, 21–22, 26–27
- files
 - creating on file system, 310–311
 - creating/writing in default directory to, 304–306
 - creating/writing to external storage, 311–312
 - overview of, 303–304
 - reading from in default directory, 306
 - reading raw, 306–308
 - reading XML, 308–309
- frameworks, 34–35
- free markets for, 27
- marketplace, 22–23
- mobile development
 - device database management, 361–364
 - requirements, 357–360
 - risks, quality assurance, 367–368
 - risks. project assessment, 364–367
 - third-party requirements, 360
 - use case development, 360
- planning user experience. *see* **User experience**
- resources. *see* **Resources**
- sample, 51–52

Applications (*continued*)

- secure integration of, 26–27
- security and permissions, 31–32
- writing first
 - adding Android samples, 56–57
 - adding Snake project, 57–60
 - creating AVD, 61–63
 - launch configuration, 62
 - overview of, 55–56
 - running in emulator, 66–68

ApplicationTestCase, 392**Apply() batch method, 326****Apply() method, 285****Architectures**

- mobile application design, 371
- platform, 29–30

ArrayAdapter, 233–234**Arrays**

- integer, 139, 152
- mixed-type, 139
- resource, 165–166, 188–189
- string, 139, 146, 150–151

ASL/Apache 2 (Apache Software License), 25**AssertEquals() method, 450****Assertions, 450****AssetManager, 99****Assets**

- folder, 74
- retrieving application, 99

Asynchronous messaging mechanism, 98**AT&T Apps, 479****Attributes**

- <uses-sdk> tag, 124
- color state list <item>, 158
- dialog, 268–269
- <fragment tag>, 246

FrameLayout view, 222–223

layout, 215

LinearLayout view, 217–218

preferences, 287–289

RelativeLayout view, 219–221

TableLayout, 225–227

TableRow, 227

TextView, 180

ViewGroup subtypes, 215–216

Audience

- distribution model for, 459
- responsive mobile applications for, 382
- for this book, 1–2

AudioManager, 35**Authors, contacting, 7****Autocomplete, in Java, 552****AutoCompleteTextView, 186–187****AutoLinks, TextView, 180–182****Automated testing, 391–392, 428****Availability, 380****AVD (Android Virtual Device)**

- building first application, 75–76, 77–79
- calling between two emulator instances, 517–518
- configuring GPS location of emulator, 516–517
- targeting for debugging, 64–65
- working with
 - creating with custom hardware, 510–511
 - creating within IDE, 509–510
 - overview of, 507–508
 - using Android Virtual Device Manager, 508–509
- writing first application, 61–63, 66–68

B

Back button

- back navigation, 401–402
- navigating back stack, 404

Back stack, 404

Backup Service, 383, 386

Backup testing, 432

Backups, enabling, 383

Best practices

- Android Marketplace, 22
- design
 - avoiding silly mistakes, 388
 - leveraging diagnostics, 386
 - maintenance and upgrade, 385–387
 - for maximum profit, 383–384
 - meeting users' demands, 380
 - quality guidelines, 384–385
 - rules, 379–380
 - security, 383
 - stability and responsiveness, 381–382
 - third-party quality standards, 385
 - tools for, 387–388
 - user interface, 380–381
- development
 - avoiding silly mistakes, 393
 - code diagnostics, 391–392
 - code reviews, 391
 - coding standards, 390
 - feasibility testing early and often, 389
 - handling defects on single device, 392
 - overview of, 388–389
 - software development process for, 389
 - tools, 393

file management, 302

fragment-based design as, 105

layouts, 209–210

testing mobile applications

- avoiding silly mistakes, 435
- defect-tracking system, 423–424
- managing testing environment, 425–427
- maximizing coverage, 427–430
- SDK tools for, 434–435
- specialized test scenarios, 431–434
- third-party standards, 430–431
- using support libraries for, 358
- XML filenames, 140

Beta

- distribution model for, 459
- feedback when self-publishing, 479

Billing

- mobile application design, 383–384
- self-publishing and, 480
- user, 461

/bin folder, 74

BitmapDrawable object, 157–158

Black-box testing, 373, 429

Blog, Android Developers, 382

Bmgr tool, 435, 501

Bold, strings, 147

Boolean

- resources, 139, 151
- using preferences, 282–284, 286

Borderless buttons, 191

Borders, 157

Breakpoints, 83

The Brick, 13–15

Broadcast receivers, 131

Broadcasting intents, 111–112

Browser content provider, 319–321

Browsing file system, File Explorer, 536–538**Bugs. see also Debugging**

- handling on single device, 392
- leveraging application diagnostics for, 386
- tracking user reports, 376

Build

- acceptance testing, 427, 428
- errors, 556
- first application
 - core files and directories, 73–75
 - creating AVD, 75
 - creating new project, 69–73
 - debugging in emulator, 80–83
 - debugging on hardware, 90–92
 - launch configuration, 75–76
 - location-based services, 88–90
 - logging support, 83–84
 - media support, 84–87
 - overview of, 68–69
 - running in emulator, 76–79
- validations, 427

Build targets, choosing, 441, 443**Built-in content providers, 316****Built-in layout classes**

- FrameLayout, 222–223
- GridLayout, 228–230
- LinearLayout, 217–219
- overview of, 215–217
- RelativeLayout, 219–222
- TableLayout, 224–227

Bundles activity state, saving, 104**Button class, 190–192****Buttons**

- action, 411–413
- basic, 190–192
- check boxes, 193

- defined, 190
- image, 192
- radio, 194–197
- styles of, 183
- switches, 190
- toggles, 193–194
- using emulator, 505

C
Cache directory

- accessing, 304
- managing size of, 311
- methods, 305
- retrieving external cache, 312
- retrieving subdirectory, 305
- storing files, 302, 310

CalendarContact content provider, 321**Callbacks**

- attaching/detaching fragments, 246–247
- destroying static data, 104–105
- initializing static activity, 102
- initializing/retrieving data, 103
- saving state into bundle, 104
- state/resource management, 101
- stopping/saving/releasing data, 103

CallLog content provider, 318–319**Cameras, in emulator, 526****Candy bar mobile phone, 15****Certificates, application signing, 32****CharacterPickerDialog class, 266–267****CheckBox control, 190, 192****CheckBoxPreference class, 287****Child View controls**

- adding to ViewGroup programmatically, 214
- defined, 214

- FrameLayout, 222–224
- GridLayout, 228–230
- layout attributes of, 215–216
- LinearLayout, 210, 217–219
- RelativeLayout, 219–221
- TableLayout, 225–227
- ViewSwitcher, 239
- Chromecast, 29, 350**
- Chronometer control, 205–206**
- Clamshell mobile phone, 15**
- Classes**
 - AbsoluteLayout, 334
 - AccessibilityManager, 35
 - Activity. *see* **Activity class**
 - AdapterView, 233, 235–237
 - AlertDialog, 266–267, 270–271
 - AnalogClock, 206–207
 - AnimationDrawable, 160–161
 - ApplicationTestCase, 392
 - ArrayAdapter, 233–234
 - AudioManager, 35
 - AutoCompleteTextView, 186–187
 - Button, 190–192
 - CharacterPickerDialog, 266–267
 - CheckBox, 190, 193
 - CheckBoxPreference, 287
 - Chronometer, 205–206
 - ClipboardManager, 35
 - ContactsContract, 322–324
 - ContentProviderOperation, 325, 327
 - Context
 - accessing device file system, 304
 - application directories, 305–309
 - defined, 97
 - using, 99
 - creating new, 552
 - creating test within IDE, 445–447
 - CursorAdapter, 233–235
 - CursorLoader, 235, 315, 317–318, 320–321
 - DatePicker, 197–199
 - DatePickerDialog, 266
 - Debug, 433, 534
 - Dialog, 266–267
 - DialogFragment, 248, 268–275
 - DigitalClock, 206
 - DisplayMetrics, 334
 - DownloadManager, 35
 - EditText, 183–185
 - EditTextPreference, 287
 - Environment, 311
 - FieldNoteListFragment, 246
 - FieldNoteViewActivity, 250
 - FieldNoteWebViewFragment, 250
 - File, 305, 309
 - Fragment
 - Activity focus on managing, 245
 - organizing Activity components, 105
 - overview of, 98
 - referencing in application package, 246
 - specialty, 248
 - FragmentManager, 35
 - FrameLayout
 - designing compatibility, 334
 - parent layout, 217
 - using, 222–224
 - GridView
 - binding data to AdapterView, 235–236
 - defined, 233
 - monitoring for click events, 236–237
 - ImageButton, 158, 192

Classes (*continued*)

- imported, 84

- Intent

- action types in, 108
 - defined, 98
 - passing additional information, 109

- LayoutParams

- FrameLayout, 222
 - LinearLayout, 218
 - RelativeLayout, 219

- LinearLayout

- creating layouts programmatically, 211–213
 - creating using XML resources, 210
 - as parent layout, 217
 - set to size of screen, 216–217
 - using, 217–219
 - working with, 167–171

- ListActivity, 237, 247

- ListFragment

- defined, 248
 - designing fragments, 249–250
 - implementing, 250–253
 - implementing ListFragment, 250–253

- ListPreference, 287

- ListView

- data-driven containers, 233, 235–237
 - designing fragments, 249
 - implementing ListFragment, 250–253
 - managing with ListActivity, 247

- LocationManager, 35

- Log

- naming conflicts in, 83, 553
 - viewing application log data, 490

- MarginLayoutParams, 215

- MediaPlayer, 84–87

- MediaStore, 316–317, 319

- modeling, 398

- MoreAsserts, 455

- MultiAutoCompleteTextView, 186–188

- MultiSelectListPreference, 287

- OnItemClickListener, 236

- PerformanceTestCase, 392

- Preference class, 287

- PreferenceActivity class

- defined, 247–248
 - managing user preferences, 286
 - using, 289–291

- PreferenceCategory class, 287–289

- PreferenceFragment

- defined, 248
 - displaying, 289–291
 - preference headers, 291–296

- ProgressBar

- activity bar and circles, 202
 - indicating progress, 199–202
 - RatingBar, 204–205
 - responsive mobile applications, 382

- ProgressDialog, 266

- RatingBar, 204–205

- RelativeLayout, 219–222, 348

- reorganizing code in Java, 555

- SeekBar, 202–203

- Service, 98, 111

- ServiceTestCase, 392

- Settings, 322

- ShapeDrawable, 155

- SharedPreferences

- adding, updating, deleting preferences, 284–285
 - defined, 99

- for multiple activities, 282–283
 - overview of, 282
 - reacting to preference changes, 285
 - searching and reading preferences, 283
 - for single activity, 282
- SimpleFragDialogActivity, 269, 273
- SlidingDrawer, 240
- Spinner
 - defined, 51
 - editing preferences, 291
 - filtering user choices, 188–190
 - styles, 183
- StrictMode, 302
- Switch, 190, 194
- TabActivity, 248
- TabHost control, 248
- TableLayout
 - defined, 178
 - user interface design, 334
 - using, 224–227
- TableRow, 224–227
- TextClock, 206
- TextView, 180
- TimePicker, 190
- TimePickerDialog, 267
- ToggleButton, 190, 193–194
- TouchUtils, 392
- View, 177, 209–210, 214
- ViewAsserts, 455
- ViewGroup, 209–210, 214–215
- ViewManager, 35
- WebView
 - implementing
 - WebViewFragment, 254
 - loading with ListView, 249
 - organizing into fragments, 244
 - WebViewFragment hosting, 248
 - WebViewFragment
 - defined, 248
 - designing fragments, 249–250
 - implementing, 254–255
 - WindowManager, 35
- Clean states, devices, 426**
- Cleaning project, for build errors, 556**
- ClearCheck() method, 196–197**
- Clients**
 - quality assurance testing on, 368
 - testing against new remote servers, 430
- ClipboardManager, 35**
- Clocks**
 - analog, 206–207
 - digital, 206
 - text, 206
- Cloud Save, 296–297**
- Cloud services**
 - change management, 376
 - network-driven applications for, 371
 - quality assurance testing on, 368
 - transmitting private data, 383
- Code**
 - accessing resources, 142
 - Boolean resources in, 151
 - color resources in, 153
 - dimension resources in, 154
 - image resources in, 157–158
 - integer resources in, 152
 - layout resources in, 170
 - layouts in, 211–213
 - menu resources in, 163
 - mobile application development
 - diagnostics, 391–392
 - reviews, 391
 - standards, 390

Code (*continued*)

- obfuscation tools, 460
- simple drawable resources in, 156
- string resources in, 148–149
- system resources in, 172
- tweened animation sequence in, 164–165
- writing in Java. *see* **Java**

Code coverage testing, unit tests, 391–392**Color, for visual feedback, 418****Color resources**

- defining in XML, 152–153
- defining with color state lists, 158–159
- overview of, 152
- using programmatically, 153

Columns, ContactContract data, 323**Comments**

- avoiding obfuscation, 390
- creating new classes, 552
- customizing tags for, 552
- in Javadoc style, 556
- renaming, 554

Commercializing, WAP applications, 16–17**Commit() method, 289****Compatibility**

- ActionBar, 415
- alternative resources
 - organizing efficiently, 345–347
 - organizing with qualifiers, 339–344
- for orientations, 345
- overview of, 338
- resolving, 338–339
- using programmatically, 345
- best practices, 331–333
- SDK license agreement, 43
- strategies, 5

targeting

- Google Chromecast devices, 350
- Google TV devices, 348–350
- tablet devices, 347–348
- testing mobile applications, 373
- user interface design
 - fragments, 335
 - nine-patch stretchable graphics, 336
 - specific screen types, 335–336
 - Support Library, 335
 - tools, 333–334
 - working squares, 336–338

<compatible-screens> tag, manifest file, 129**Complex application content, 301–302****Compression, GL texture settings, 129****Configuration changes, 347****Configuration management systems, 369–370****Conformance testing, 432****Console, emulator**

- manipulating power settings, 523–524
- monitoring network status, 523
- other commands, 524
- overview of, 520–521
- sending GPS coordinates, 523
- simulating incoming calls, 521
- simulating SMS messages, 522

ContactsContract content provider, 322–324**Containers**

- ArrayAdapter, 234
- binding data to AdapterView, 235–236
- CursorAdapter, 234–235
- data-driven, 233
- designing compatibility, 334
- DrawerLayout, 239

- ListActivity, 237–238
 - overview of, 232–233
 - scrolling, 238
 - selection event handling, 236–237
 - switchers, 239
 - ViewPager, 239
- Content providers**
- accessing with special permissions, 319
 - Browser, 319–321
 - CalendarContact, 321
 - CallLog, 318–319
 - ContactsContract, 322–324
 - MediaStore, 316–318
 - modifying data
 - adding records, 325–326
 - deleting records, 327–328
 - overview of, 324
 - updating records, 326–327
 - overview of, 315–316
 - registering, 131
 - Settings, 322
 - third-party, 328
 - UserDictionary, 321–322
 - VoiceMail, 322
- ContentProviderOperation class, 325, 327**
- Context**
- files and directories, 99
 - overview of, 97
 - preferences, 99
 - retrieving, 98
 - retrieving assets, 99
 - retrieving resources, 99
 - terminology, 97
 - using, 98–99
- Context class**
- accessing device file system, 304
 - application directories, 305–309
 - defined, 97
 - using, 99
- Contextual links, 180–182**
- Controls**
- Hierarchy Viewer indicators, 496
 - identifying unnecessary layout, with lint, 496
 - user interface, 177–178
- Conventions used in this book, 7**
- Copying files**
- from emulator/device, 538
 - to emulator/device, 539
- Copyright infringement, 460**
- Core app quality guidelines, 384**
- Core files and directories, 73–75**
- Costs, 23, 27**
- Coverage, testing, 426–427**
- Crash reports**
- tracking user, 376
 - viewing in Developer Console after publishing, 476–477
- Crashes, device defects in, 424**
- Create Activity wizard, 71–72**
- CRM (customer relationship management) applications, 318**
- CursorAdapter class, 233–235**
- CursorLoader class**
- adding to application, 235
 - content providers
 - Browser, 320–321
 - CallLog, 318
 - overview of, 315
 - MediaStore, 317–318
- Customer relationship management (CRM) applications, 318**
- Customization method, 358–359**
- Customizing**
- defect-tracking systems, 423–424
 - dialogs, 270–271

Customizing (*continued*)

- hardware settings for AVD, 510–511
- log filters, 551

D

Dalvik Debug Monitor Server. *see* **DDMS**
(**Dalvik Debug Monitor Server**)

Dalvik VM, 31

Data

- binding to AdapterView, 235–236
- handling configuration changes, 347
- intents, 108
- managing device databases, 361–364
- minimizing calls, 382
- retaining across configuration changes, 347
- testing transmissions to/from remote server, 430
- transmission of private, 430

Database, mobile device, 360–364, 426

Data-driven containers

- adapters, 234–235
- arrays, 233–234
- binding data, 235–236
- click events, handling, 236–237
- headers/footers, 237
- lists of items, 237
- overview of, 233

DatePicker control, 197–199

DatePickerDialog class, 266

DDMS (Dalvik Debug Monitor Server)

- Allocation Tracker, 534–536
- Android IDE toolbar, 47
- application logging, 543–544
- attaching debugger to applications, 531
- configuring GPS location of emulator, 516–517

- copying files to and from device, 302
- debugging with, 490–491

Emulator Control pane

- change telephony status, 540
- overview of, 539
- sending location fix, 541
- simulate incoming SMS messages, 540–541
- simulate incoming voice calls, 540–541

File Explorer

- accessing preferences file, 285–286
- browsing file system of emulator/device, 536–538
- copying files from emulator/device, 538
- copying files to emulator/device, 539
- deleting files on emulator/device, 539

forcing crash, 83

garbage collection, 534

HPROF files, 534–535

key features, 530–531

memory profiling, 387

mobile application debugging tool, 393

monitoring heap activity, 532–533

monitoring thread activity, 532

overview of, 529

screen captures of emulator/devices, 542–543

as standalone application with IDE, 529–530

stopping process, 532

System Information pane, 541–542

as testing tool, 434

viewing network statistics, 535–537

Debug As icon, 80

Debug class, 433, 534

Debug configuration

- debugging application on hardware, 90–92
- launch configuration for project, 62, 64, 76
- launching emulator to run application, 66–68, 513–515
- naming and targeting AVD for, 64–65

Debug key, 465

Debug perspective, 47, 80–83

Debug thread, 433

Debugging. *see also* DDMS (Dalvik Debug Monitor Server)

- with ADB tool, 393, 434, 490–491
- disabling when packaging code, 464
- in emulator, 80–83
- on hardware, 39–42, 90–92
- with Hierarchy Viewer, 393, 496
- with LogCat. *see* **LogCat utility**
- resolution process, 42
- with Toast messages, 192

Default application directory, 305–306

Default resources

- alternative vs., 141–142
- including with alternative resources, 339
- naming alternative resources exactly as, 338–339

Defects, mobile application, 424

Defect-tracking systems, 423–424

Deleting

- content provider records, 327–328
- files on emulator/device, 539
- preferences, 284–285

Deployment, mobile application, 374–375

Deployment Target Selection Mode, 76

Deprecated methods, 266

Descendant navigation, 401–402

Design comps, screen layouts, 417

Design patterns, navigation

- defined, 405
- documentation for, 486
- drop-down, 406–407
- master detail flow, 405, 408–409
- navigation drawer, 406, 408–409
- tabs, 406–407
- targets, 408, 410

Design tab, Android documentation, 486–488

Detaching fragments, 246–247

Dev Tools application, emulator, 506

Develop tab, Android documentation, 486–488

Developer Console account, 461

Developer Console, Google Play

- additional options, 475
- Android documentation, 487
- Google Play Game Services APIs, 476
- pricing/distribution details, 474–475
- sign up for publisher account, 468
- translating applications, 478
- upgrading application, 477
- uploading application, 471–473
- uploading assets, 473–474
- viewing statistics, 476–477

Developer Distribution Agreement, Google Play, 461

Developer Options, 40

Developer Program Policies, Google Play, 461

Developer website

- ADT Bundle updates, 55
- emulator, 489
- as resource, 6
- SDK updates, 37, 55

Developer website (*continued*)

- signing, 467

- Support Package updates, 260

- supporting different types of screens, 336

Developers

- access to underlying hardware, 26

- Android as free and open source for, 23, 25

- familiar/inexpensive tools for, 25–26

- Google Play store registration, 32

- learning curve for, 26

- mobile, 18

- proprietary mobile platforms with programs for, 17

- SDK License Agreement for, 43

- this book for software, 1

- WAP solution for mobile and, 16

- websites for software, 6

Developers Blog, StrictMode, 433**Development environment**

- set up

- application framework, 44–45

- basic installation process, 38–39

- configuring, 39–42

- core framework, 45–46

- device debugging, 39–40

- emulator, 48–50

- IDE and ADT, 47–48

- overview of, 37

- sample applications, 51–52

- SDK and AVD Managers, 48

- SDK documentation, 44

- SDK license agreement, 42–44

- SDK problems, 42

- SDK upgrades, 42

- testing

- adding project to workspace, 57–60

- adding samples with SDK Manager, 56–57

- creating AVD, 61–62

- launch configuration, 62–65

- overview of, 55–56

- running application in emulator, 66–68

- in this book, 5–6

Device databases

- configuration management, 426

- overview of, 361–362

- storing device data, 361–363

- third-party, 364

- using device data, 363–364

- which devices to track, 361

Devices

- accessing settings, 322

- apps driving sale of, 21–22

- bugs on single, 392

- choosing for AVD, 509

- clean state determination on, 426

- compatibility, 331–332

- configurations, 425–426

- creating AVD, 62–64

- debugging, 39–40, 514

- delivering experience, 20–21

- file system interaction on, 536–539

- getting information about screen, 334

- logging defect information, 423–424

- manufacturers designing, 19–20

- mobile

- limitations, 370

- project risks, 364–367

- user interface design for, 380–381

- quality assurance testing, 367

- rooted, 426

- specifying features, 127–128

specifying supported screen sizes, 128

testing

- content provider code, 315

- on emulator vs., 428

- glossary of terms for, 424

- in preproduction, 428

- on real, 302–303

- on upgrades of, 432

Devices pane, DDMS, 530

Diagnostics

- developing code, 391–392

- leveraging application, 386

Dialer application, 517–518

Dialog class, 266–267

DialogFragments class

- custom, 270–271

- defined, 248

- lifecycle of, 268–270

- overview of, 267–268

- working with support package, 271–275

Dialogs

- alert, 266–267

- attributes, 268–269

- basic, 266–267

- character pickers, 266–267

- customizing, 270–271

- date pickers, 266–267

- dismissing, 269–270

- fragment method, 267–270

- fragments, Support Package, 270–275

- legacy method, 265

- lifecycle, 268–270

- presentation, 267

- presenting actions to users, 415

- progress, 266–267

- providing visual feedback, 418

- showing, 269

- time pickers, 267

Digital signatures

- for application updates, 477

- packaging/signing application, 465–467

Dimension resources, 153–154

Dimensions, for compatibility, 334

Directional pad (D-pad), Google TV, 349

Directories

- accessing, 99

- application, 304

- cache

 - accessing, 304

 - defined, 302

 - management of, 310–311

 - methods, 305

- core, 73–75

- default application

 - reading from files in, 306

 - writing to files in, 304–306

- external storage, 312

- File Explorer

 - browsing file system of emulator/
device, 536–538

 - copying files from emulator/
device, 538

 - copying files to emulator/
device, 539

 - deleting files on emulator/
device, 539

- monitoring, 311

- resources, 138

- setting up, 309–311

- subdirectories

 - applications, 304, 310

 - resources, 137–138

- working with, 303–304

Directory qualifiers

- organizing alternative resources, 339–344
- storing alternative resources, 344

Disk space use, device defects in, 424**Dismiss() method, dialogs, 268–269****Displaying**

- characteristics of device at runtime, 334
- data to users
 - adjusting progress, 202–203
 - clocks, 206–207
 - progress bars, 199–202
 - ratings, 204–205
 - time passage, 205
- text, 179–182

DisplayMetrics, 334**Distribute tab, Android documentation, 486–488****Distribution**

- choosing application version, 370
- documentation for open, 487
- infrastructure for published applications, 467
- methods, 459–460

Distribution tab, Developer Console, 475**Dmtracedump tool, 502****Documentation**

- Android help, 486
- AVD testing of device
 - configuration, 429
- custom action, 131
- handling bugs on single device, 392
- Javadoc style, 556
- lacking in early development, 28
- mobile application
 - maintenance, 369, 376
 - overview of, 368–369

porting, 369

test plans for quality assurance, 368–369

third party requirements, 369

reading, 44

reference for, 36

SDK, 56–57

updated features of, 22

Documents, coding standards, 390**Domain modeling, 398****Downloaded content, 301–302****DownloadManager, 35****Downloads, design documentation, 486****Downtimes, minimizing, 429****Dp units, 154****D-pad (directional pad), Google TV, 349****draw9patch tool, 387, 497–501****Drawable resources**

- defining in XML, 155–156
- defining with color state lists, 158–159
- image resources as, 156–157
- simple, 154–155
- using programmatically, 156

DrawerLayout pattern, 238**Drop-down navigation, 406–407**

E
Eclipse IDE. *see also* Android IDE

- as alternative to Android IDE, 38
- Android Studio alternative to, 26
- designing Android apps, 25–26
- development system requirements, 38
- installation process, 38–39
- using sample applications, 52

Editing

- manifest file manually, 119–121

- manifest file with Android IDE
 - application/activity settings, 117–118
 - overview of, 115–116
 - package-wide settings, 117
 - permissions, 117–118
 - test instrumentation, 119
- preferences, 284–285

EditText controls, 183–185

EditTextPreference class, 287

ElapsedRealTime() method, Chronometer, 205

Ellipsize attribute, TextView, 180

Ems attribute, TextView, 180

Emulation options, AVD, 51

Emulator

- accessing, 505
- as application design tool, 373–374, 386
- calling between two instances, 517–518
- configuring GPS location of, 88–89, 516–517
- configuring startup, 513
- console
 - network status, 523
 - other commands, 524
 - overview of, 520–521
 - power settings, 523–524
 - sending GPS coordinates, 523
 - simulating incoming calls, 521
 - simulating SMS messages, 522
- debugging in, 80–83
- enjoying, 524–525
- functions of, 48, 50
- interacting with file system, 536–539
- launching
 - from Android Virtual Device Manager, 515–516
 - overview of, 513–515

- to run application, 62–68, 76–79
 - with specific AVD, 512
- limitations of, 525–526
- messaging between two instances, 518–520
- overview of, 505–506
- performance, 512–513
- power of, 489
- startup options, 513
- testing
 - layout, 211
 - mobile applications, 373–374
 - on real device vs., 428–429, 507
 - with, 393, 434
- tips for using, 506–507
- working with AVDs
 - creating AVD, 509–510
 - creating AVD with custom hardware, 510–511
 - launching with specific AVD, 512–515
 - overview of, 507–508
 - using Android Virtual Device Manager, 508–509

Emulator Control pane, DDMS

- change telephony status, 540
- features, 531
- overview of, 539
- sending location fix, 541
- simulate incoming voice calls, 540–541

Encouraging action. see Actions

End User License Agreement (EULA), mobile, 383

Enforcing

- permissions
 - overview of, 132–133
 - using Permissions tab, 117–118

Enforcing (*continued*)

- platform requirements
 - device features, 127–128
 - input methods, 126–127
 - screen sizes, 128
- system requirements
 - maximum SDK version, 126
 - minimum SDK version, 124–125
 - overview of, 123
 - target SDK version, 125–126
 - targeting specific SDK versions, 123–124

Entities, project, 398**Entity relationship modeling, 398****Entry navigation, 400****Environment**

- development. *see* **Development environment**
- for testing
 - clean start state, 426
 - managing device configurations, 425–426
 - overview of, 55–56, 425–427
 - real-world, mimicking, 426–427

Environment class, 311–312**Error handling**

- coding standards for, 390
- debugging your Android application in emulator, 80–83
- resolving mysterious build errors, 556

Escaping, XML, 148**etc1tool command-line tool, 502****EULA (End User License Agreement), mobile, 383****Exerciser Monkey (monkey)**

- defined, 434
- testing for unexpected, 433
- testing with, 456

ExpandableListActivity, 237**ExpandableListAdapter, 237****Experience.** *see* **User experience****Exporting, and signing package file, 465–467****Extensibility, mobile design for, 371****External cache directory, retrieving, 312****External libraries, 128–129****External navigation, 404****External storage**

- accessing, 303, 311–312
- creating/writing files to, 311–312
- file management practices, 302

Extract Local Variable tool, Java, 554–555**Extract Method tool, Java, 555****Extras, Intent object, 109**

F
F6 key, 507**F8 key, 505****Facebook App Center, 375****FC (force close) issues, avoiding, 433****Feasibility testing, 389–390****Feature support**

- Google TV devices, 349
- mobile application, 377
- tablets, 348

Features, device defects in, 424**Feedback**

- application diagnostics from, 386
- on device defects, 424
- providing visual, 418
- usability studies for mobile, 430

Fierce Developer newsletter, 6**File Explorer**

- accessing preferences file, 285–286
- browsing file system of emulator/device, 536–538

- copying files from emulator/
device, 538
- copying files to emulator/device, 539
- deleting files on emulator/device, 539

Files

- accessing, 304
- AndroidManifest.xml, 73, 89–90,
115
- application data on device, 301–302
- application directories, 304
- copying, 302
- creating and writing
 - in default application directory,
304–306
 - to external storage, 311–312
- creating on file system, 310–311
- directories
 - creating and writing files to default,
304–306
 - overview of, 304
 - reading from files in, 306
 - setting up structure, 309–311
- managing, 302–303
- manifest
 - application/activity settings,
117–118
 - defined, 115
 - editing manually, 119–121
 - editing with Android IDE,
116–117
 - enforcing platform requirements,
126–128
 - enforcing system requirements,
123–126
 - external libraries, 128–129
 - Google TV settings, 349
 - identity management, 122–123
 - lesser-used settings, 129

- maximizing device
compatibility, 333
- other settings, 133
- overview of, 115–116
- package-wide settings, 117
- permissions, 117–118, 131–133
- registering activities in, 129–131
- test instrumentation, 119
- monitoring, 311
- permissions
 - accessing content providers, 319
 - overview of, 303
- reading from, 306
- reading raw, 306–308
- reading XML, 308–309
- searching project, 551
- viewing two sections of same, 550
- working with, 303–304

FileStreamOfConsciousness application, 304

Filter() method, InputFilter interface, 184–185

Filters

- creating LogCat, 543–544
- customizing log, 551
- Google Play, 121, 463–464
- input, 184–185
- intent, 130–131
- other application configuration
settings, 129

FindViewById() method, 178

Finish() method, 106

Firmware upgrades, testing, 376

First-generation mobile phones, 13–14

Footers, in ListView, 237

Force close (FC) issues, avoiding, 433

ForceError() method, 80–83

Form factors, mobile phone, 15

Format strings, 148**Formats**

- Google TV supported, 349
- image, 156
- media, 164
- resource references, 165
- string resources, 147
- writing code in Java, 553

Forums

- Android developer, 6
- XDA-Developers Android Forum, 7

Fragment classes

- organizing Activity components, 105–106
- overview of, 98
- specialty, 247–248

Fragment subclasses, 247–248**<fragment tag>, 245–246****Fragmentation, 331–332****FragmentManager**

- defined, 35, 245
- dialog implementation, 265–266
- implementing ListFragment, 254
- support package dialog fragments, 272–275

Fragments

- in Android application terminology, 98
- attaching, 246–247
- back navigation, 401–402
- compatibility, 334–335
- defining, 245–246
- designing applications
 - Activity classes, 257–258
 - layout resource files, 255–257
 - ListFragment, 250–253
 - overview of, 248–249
 - WebViewFragment, 254–255

detaching, 246–247

dialog

- custom, 270–271
 - defined, 248
 - implementation, 265–266
 - lifecycle of, 268–270
 - overview of, 267–268
 - support package, 271–275
- lifecycle, 244–245, 268–270
- managing modifications, 246
- navigating with, 404
- nested, 261
- overview of, 105–106, 243
- paging data with ViewPager, 239
- special types, 247–248
- support
 - legacy, 259
 - linking to your project, 260–261
 - new applications targeting older platforms, 259–260
 - overview of, 259
- tablets, 348
- understanding, 243–245
- user interface design tips for mobile, 380

FragmentTransaction operation, 246**Frame-by-frame animation, 159–161****FrameLayout**

- designing compatibility, 334
- as parent layout, 217
- using, 222–224

Frames, Nine-Patch graphics for, 157**Framework, application**

- exploring, 45–46
- JUnit testing for, 391–392
- overview of, 34–35

Free applications, 23, 383

Free market, 27
 Free trial version, distribution, 460

G

Game Services APIs, Google Play, 476

Games

emergence of, 17
 first-generation “time-waster,” 14

GC (garbage collection)

prompting DDMS to force, 534
 updating heap statistics after, 532

/gen folder, 58, 74

GetActionBar() method, 414

GetApplication Context() method, 98

GetAssets() method, 99

GetBoolean() method, 151

GetContentResolver(), 326

GetDimension() method, 154

GetExternalCacheDir() method, 312

GetExternalFilesDir() method, 312

GetExternalStoragePublicDirectory()
 method, 312

GetExternalStorageState() method,
 311–312

GetFragmentManager() method, 245

GetJar, publishing to, 479

GetLocation() method, 88

GetQuantityString() method, 149–150

GetResources() method, 99

GetSharedPreferences() method, 99

GetSupportActionBar() method, 415

GetSupportFragmentManager() method, 273

GetText() method

EditText, 184
 test MatchingPasswords() method, 454
 TextView, 179
 unit testing APIs/assertions, 450

Getting started, design documentation
 for, 486

GL texture compression settings, 129

Glossary, logging device defects, 424

GNU General Public License Version 2
 (GPLv2), 25

Google

APIs, 56–57

Apps domain, 478

Chromecast devices, 350

design comp tools, 417

Experience device, 367

Glass, 29

going wireless, 18–19

Nexus Android devices, 21

OHA. *see* **OHA (Open Handset
 Alliance)**

SDK license agreement for APIs, 44

third-party Android APIs, 45–46

TV devices, 348–349

Google Analytics

App Tracking SDK, 462

leveraging application diagnostics
 from, 386

Google Play

Android documentation for, 487

application diagnostics from, 386

buying and selling Android
 applications, 6

crash and bug reports, 376

customization method for project
 requirements, 358–359

developer registration, 32

developer registration fee, 23

developing Google TV devices, 349

as distribution channel

billing user, 461

following policies, 461

Google Play, as distribution channel*(continued)*

- License Verification Library, 460
- as marketing channel, 375
- overview of, 27–28
- sharing profits with, 459

downloading apps, 21

filtering

- with `<uses-feature>` tag, 121
- with `<uses-sdk>` tag, 124

Game Services SDK, 21–22

international laws for selling on, 468

packaged application

- digital signature validity period, 465
- preparing, 463–464

publishing to

- additional options, 475
- Developer Distribution Agreement, 468–469
- Game Services API, 476
- overview of, 468, 476
- pricing/distribution details, 474–475
- private channel, 478
- removing, 477–478
- return policy, 476–477
- sign up for publisher account, 468–471
- sign-in page, 468–469
- Staged Rollouts, 478
- translation, 478
- upgrades, 477
- uploading, 471–473
- uploading marketing assets, 473–474

redesign of, 21–22

services, 35

Staged Rollouts, 374

Google Plus website, 6**Google Services**, 487**Google Team Android Apps website**, 6**Google Wallet Merchant account**, 461, 468, 470**GPLv2 (GNU General Public License Version 2)**, 25**GPS coordinates**

- configuring location of emulator, 516–517
- emulator console sending, 523
- sending location fix in Emulator Control pane, 541

Graphical Layout Editor, 386**Graphical Layout mode**, UI designer, 492**Graphics**. *see also* **Nine-Patch Stretchable Graphics**

- alternative resources for different orientations, 345–346
- designing compatibility, 334
- drawable resources for, 154–155
- sizing appropriately for mobile devices, 381
- storing, 141

GridLayout, 228–230**GridView**, 233, 235–237**Groups**, permission, 133

H**Handango**, publishing to, 479**Hardware**

- access to underlying, 26
- application debugging, 90–92
- creating AVD with, 51
- creating AVD with custom, 510–511
- device compatibility, 331–332
- device debugging, 39
- testing applications, 426
- upgrades, 22

Headers

- in ListView, 237
- organizing preferences with, 322
- preference, 291–298

Heap

- inspecting with HPROF files, 534
- memory use and, 532
- monitoring with DDMS, 532–533

Height, TextView, 180**Help documentation, 486****Help screen, 100, 380****Hiding ActionBar, 413–414****Hierarchy View perspective, IDE toolbar, 47****Hierarchy Viewer**

- accurate interface design tool, 387
- inspecting application layouts, 230
- mobile development tool, 393
- as testing tool, 434
- white-box testing, 429

Home screen, emulator

- customizing, 524
- stopping application, 507

HorizontalScrollView control, 238**HPROF files, creating with DDMS, 534–535****Hprof-conv tool, 502****HTTP (HyperText Transfer Protocol), 15–17****I****Iautomator testing framework**

- defined, 434, 502
- functions of, 456
- user interface, 392
- white-box testing, 429

Icons

- customizing emulator, 524
- Google Play, 464

- packaging application, 463
- placing application and, 411
- setting application and, 122–123

Id attribute, fragments, 246**Id fields, CursorAdapter, 235****Identity**

- communicating for application, 416–417
- managing application
 - name and icon setting, 122–123
 - overview of, 122
 - versioning, 122

Image buttons, 192**Images**

- formats, 156
- Nine-Patch Stretchable Graphics, 157
- using programmatically, 157–158

ImageView layout, 170–171**Imports, organizing in Java, 553****Improving app quality, 385****In-app billing, testing, 433****<include> tags, layout, 232****Incoming calls**

- simulating in emulator console, 521
- simulating in Emulator Control pane, 540

Indicators

- activity bar and circles, 202
- adjusting progress, 202–203
- clocks, 206–207
- customizing, 203
- Hierarchy Viewer performance, 496
- progress bars, 199–202
- ratings, 204–205
- time passage, 205–206

Infringements, intellectual property protection, 460

Init() method, 198–199**Input**

- filters, constraining user, 184–185
- methods
 - enforcing platform requirements, 126–127
 - user interface design for mobile, 381
- mode of
 - designing Google TV devices, 349
 - designing tablets, 348
- providing validation for user, 418
- validation, 424

InputFilter interface, 184–185**Install Packages, 56–57****Installation, testing application, 432****Instrumentation tab, IDE manifest file resource editor, 119****Integer resources, 152****Integration**

- device defects in, 424
- points, 430
- testing application for points of, 431

Intellectual property protection, 460**Intent class**

- action types in, 108
- defined, 98
- passing additional information, 109

Intent filters, 108**Intents**

- with action/data, 108
- activity transitions with, 106–107
- application navigation, 110
- ContactsContract content provider, 322
- definition of, 98, 108
- filters, 130–131
- Google application, 109

launching activities

- by class name, 107–108
- external, 108–109
- overview of, 106
- passing additional information, 109
- receiving/broadcasting, 111–112

International laws, Google Play, 468**Internationalization**

- alternative resources for, 141–142
- testing application, 432
- testing applications, 432

Internet

- not scaling well for mobile, 15
- WAP solution for mobile, 15–17

INTERNET permission, adding media to application, 86–87**Interoperability, mobile design for, 372–373****Introduction to this book**

- changes in this edition, 3–5
- contacting authors, 7
- conventions used, 7
- development environment used, 5–6
- questions answered, 2
- structure of, 2–3
- supplementary materials available, 6
- where to find more information, 6–7
- who should read it, 1–2

INVITE_CONTACT Intent type, 322**IsFinishing() method, 105****Issue Tracker website, 42****Italic strings, 147****<item> attributes, color state list, 158****Iterative approach, mobile development, 357****J****Jarsigner command-line tool, 467**

Java

- Android apps written in, 26
- as only choice for Android SDK, 32
- writing code in
 - autocomplete, 552
 - formatting code, 553
- Javadoc style documentation, 556
- new classes and methods, 552
- organizing imports, 553
- refactoring code, 554
- renaming almost anything, 553–554
- reorganizing code, 555
- using Extract Local Variable tool, 554–555
- using Extract Method tool, 555
- using QuickFix, 555–556

Java ME (Java Micro Edition), 17**Java perspective, 47****JDK (Java Development Kit), Version 6, 37, 38****JUnit testing framework**

- analyzing results, 450–453
- assertions, 450
- running first test, 450
- unit testing with, 391–392, 436–437

K**Keyboard**

- commands in emulator, 505
- mapping in IDE/Mac OS X, 83
- type/availability, alternative resource qualifiers, 343

Keystore selection screen, 466**Keytool command-line tool, 467****Killer apps, testing for, 433–434****L****Labels, Google Play, 464****Landscape mode**

- for 10-inch tablets, 75
- alternative resources for, 339, 345–346
- displaying fragments in, 249–250, 253
- flexible layout controls for, 334
- for Google TV devices, 349
- layout files for, 255–258
- resolving resources, 141–142
- screen size/density for, 332
- switching emulator to, 507
- as tablet default, 348
- using layout resources programmatically, 170
- working square in, 336–337

Language code, alternative resource qualifiers, 341**Languages**

- testing internationalization, 432
- translating applications, 478

Last-in-first-out ordering, back stack, 404**Lateral navigation, 400–401****Launch configuration**

- configuring new Android application, 71
- creating for Snake project, 62
- creating for your project, 75–76

Layout designer, 168–170**Layout direction, alternative resource qualifiers, 341****Layout View mode, Hierarchy Viewer, 494****Layout_height attribute**

- fragments, 246
- ViewGroup, 216–217

Layout_margin attribute, ViewGroup, 216–217

Layout_width attribute

- fragments, 246
- ViewGroup, 216

LayoutParams class

- FrameLayout, 222
- LinearLayout, 218
- RelativeLayout, 219
- ViewGroup, 215

Layouts

- built-in classes
 - frames, 222–223
 - grids, 228–230
 - linear, 217–219
 - overview of, 215–217
 - relative, 219–222
 - tables, 224–227
- compatibility
 - different-size screens, 336
 - tools for, 334
- configuring in TextView, 180–181
- containers
 - ArrayAdapter, 234
 - binding data to AdapterView, 235–236
 - CursorAdapter, 234–235
 - data-driven, 233
 - DrawerLayout, 239
 - ListActivity, 237–238
 - overview of, 232–233
 - scrolling support, 238
 - selection event handling, 236–237
 - switchers, 239
 - ViewPager, 239
- creating
 - programmatically, 211–213
 - using XML resources, 209–211

- fragments, 255–257
- multiple, 230–232
- organizing, 214–215
- purposes of, 209
- repositioning tabs within perspectives, 548
- resource files
 - controls as dimension resources, 153–154
 - designing in IDE, 168–170
 - overview of, 166–168
 - resource references in, 165
 - using programmatically, 170–171
- screen design, 417
- tablet design, 348
- user interface controls, 178
- using Graphical Layout Editor, 386
- using Hierarchy Viewer. *see* **Hierarchy Viewer**
- using QuickFix, 556

Legacy API support, 5**Legacy applications**

- adding ActionBar to, 415
- fragment support for, 259
- screen compatibility mode of, 336

Legacy method, dialogs, 265**Legal issues, 23****/libs folders, 74****Licensing**

- Android apps free of, 23
- Google Play License Verification Library, 460
- OS of Android, 25
- SDK agreement, 43–44

Lifecycle

- activity, 100–102
- Dialog and DialogFragment, 268–270

- fragments
 - attaching/detaching with activities, 246–247
 - defining, 245–246
 - managing modifications, 246
 - overview of, 244–245
- invoking on emulator, 507
- performance issues, 433

LIMIT statement, CursorLoader, 321**Limitations, mobile device design, 370****LinearLayout**

- creating layouts programmatically, 211–213
- creating using XML resources, 210
- as parent layout, 217
- set to size of screen, 216–217
- using, 217–219
- working with, 167–171

Lines attribute, height of TextView, 180**Links**

- creating contextual text, 180–182
- Support Package to project, 260–261
- test project to application in IDE, 443

Lint command-line tool

- identifying unnecessary layout controls, 496
- optimizing layouts, 230, 232
- testing with, 434–435

ListActivity class, 237, 247**ListFragment class**

- defined, 248
- designing fragments, 249–250
- implementing, 250–253
- implementing ListFragment, 250–253

ListPreference class, 287**ListView**

- as data-driven containers, 233, 235–237
- designing fragments, 249

- implementing ListFragment, 250–253
- managing with ListActivity class, 247

LoadInBackground() method, content providers, 315, 317–318**Local storage, minimizing, 382****Localization**

- alternative resources for, 141–142
- testing application
 - internationalization, 432
- translating applications, 478
- user interface design tips for mobile, 381

Location-based services

- adding to application, 88–89
- configuring GPS for emulator, 516–517
- sending fix in Emulator Control pane, 541
- testing applications, 427

LocationManager, 35**Logcat tool, 502****LogCat utility**

- customizing log filters, 551
- in DDMS, 543–544
- debugging in emulator, 80
- as testing tool, 434
- viewing application log data, 84–85, 490

LogCat window, DDMS, 531**Logging**

- adding to application, 83–84
- defect information, 423–424
- diagnostic, 391
- disabling when packaging code, 464

Loupe/zoom pane, Hierarchy Viewer, 496**Lowest common denominator method**

- determining project requirements, 357–358
- hybrid approach, 359

LunarLander game, 52

LVL (License Verification Library), Google Play, 460

M

Mac OS X, stepping through code in, 83

Main UI thread

avoiding lengthy operations on, 381, 432–433

coding standards for, 390

file management practices, 302

pausing/resizing in mobile applications on, 424

performance issues, 432–433

Maintenance

mobile application

design, 371

design for easy, 385–386

documentation, 369

support, 375–377

Malware requirements, SDK license agreement, 43

Managers

Android platform service, 35

this book for project, 1

Manifest file

accessing application, 99

accessing content providers with special permissions, 319

configuring applications

application/activity settings, 117–118

editing manually, 119–121

editing with Android IDE, 116–117

overview of, 115–116

package-wide settings, 117

permissions, 117–118

test instrumentation, 119

core, 73–75

defined, 115

editing in IDE, 492

enforcing application requirements

platform, 126–128

system, 123–126

external libraries, 128–129

Google Play filters, 463–464

Google TV settings, 349

identity management

overview of, 122

setting name and icon, 122–123

versioning, 122

lesser-used settings, 129

maximizing device compatibility, 333

other settings, 133

permissions

registering application enforced, 132–133

registering required, 131–132

registering activities

intent filters, 130–131

other application components, 131

overview of, 129–130

primary entry points, 130

viewing test project, 444–445

Manifest tab, IDE manifest file resource editor, 117

Manual editing, manifest file, 119–121

Manufacturers

first-generation mobile phones, 13–14

growth of Android, 21

mobile project risk assessment, 365–366

OHA device, 19–20

proprietary platforms for handsets, 17

Maps application, 88–89

MarginLayoutParams class, 215

Marketplace

choosing distribution model, 459–460

free market, 27

mobile, 374–375

mobile operators, 20–21

no one platform has emerged
victorious, 18

uploading assets to Google Play,
473–474

where we are now, 22–23

Mascot, 23–24

Master detail flow navigation, 405, 408–409

The Matrix Phone, Nokia 8110, 15

MaxEms attribute, width of TextView, 180

Maximizing windows, in workspace, 548

Maximum SDK version, 126

MaxLines attribute, height of TextView, 180

MaxSdkVersion attribute, API level, 124

Media formats, 164

Media support, 84–87

MediaPlayer class, 84–87

MediaRouter APIs, Chromecast, 350

MediaStore content provider, 316–318

**Membership supplementation, mobile
design, 384**

Memory

avoiding leaks, 99

coding standards, 390

creating AVD, 62–64

device defects in, 424

monitoring with Allocation Tracker,
532, 534–536

monitoring with heap, 532

Menu resources

defining in XML, 162–163

using programmatically, 163

<merge> tags, layout, 232

Messaging, in emulator, 518–519

Methodologies, mobile development

iteration, 358

waterfall, 356–357

Methods

addOnBackStack

ChangeListener(), 254

addToBackStack(), 401

AddView(), 214

addWord(), 322

application file management, 304–305

apply(), 285

apply() batch, 326

assertEquals(), 450

clearCheck(), 196–197

commit(), 289

customization, 358–359

deprecated, 266

dismiss(), 268–269

elapsedRealTime(), 205

Extract Method tool, 555

filter(), 184–185

findViewById(), 178

finish(), 106

forceError(), 80–83

getActionBar(), 414

getApplication Context(), 98

getAssets(), 99

getBoolean(), 151

getDimension(), 154

getExternalCacheDir(), 312

getExternalFilesDir(), 312

getExternalStoragePublicDirectory(), 312

getExternalStorageState(), 311–312

getFragmentManager(), 245

getLocation(), 88

Methods (*continued*)

- getQuantityString(), 149–150
- getResources(), 99
- getSharedPreferences(), 99
- getSupportActionBar(), 415
- getSupportFragmentManager(), 273
- getText()
 - EditText, 184
 - test MatchingPasswords()
 - method, 454
 - TextView, 179
 - unit testing APIs/assertions, 450
- init() method, 198–199
- isFinishing(), 105
- loadInBackground(), 315, 317–318
- Lowest common denominator
 - determining project requirements, 357–358
 - hybrid approach, 359
- newDelete(), 327–328
- newInsert(), 326
- newUpdate(), 327
- onActivityCreated()
 - fragments, 247
 - implementing ListFragment, 250–253
 - implementing
 - WebViewFragment, 255
- onAttach(), 247
- onBackPressed(), 401
- onBackStackChanged(), 254
- OnCheckedChangeListener, 196–197
- onClick(), 192, 441
- onConfigurationChanged(), 347
- onCreate()
 - ancestral navigation, 403
 - fragments, 247
 - initializing static activity data, 102
 - intents, 109
 - layouts, 212
 - logging support, 84
 - media, 85
 - PreferenceActivity class, 289
- onCreateOptionsMenu(), 163
- onCreateView(), 247
- onDateChanged(), 197–198
- onDestroy()
 - avoiding killing activities, 103
 - destroying static activity data, 104
 - fragments, 247
- onDestroyView(), 247
- onDetach(), 247
- onItemClickListener(), 237
- onListItemClick(), 237
- onOptionsItemSelected(), 414–415
- onPause()
 - avoiding killing activities, 103–104
 - emulator, 507
 - fragments, 247
 - stopping/saving/releasing activity data in, 103
- onRatingChanged(), 204–205
- onResume() method
 - fragments, 247
 - initializing/retrieving activity data in, 103
- onRetain
 - NonConfigurationInstance(), 347
- onSaveInstanceState(), 104
- onStart() callback, 247
- onStop()
 - adding media, 86–87
 - avoiding killing activities, 103
 - emulator, 507
 - fragments, 247
- openFileInput(), 306–307, 309

- openFileOutput(), 305–306, 309
- playMusicFromWeb(), 85–86
- registerOnSharedPreferenceChangeListener(), 285
- reorganizing code in Java, 555
- selectAll(), 184
- sendKeys(), 454
- setBase(), 205
- setContentView()
 - creating layouts, 210
 - creating layouts programmatically, 212
 - fragments, 246
 - ListActivity, 237
 - ProgressBar, 202
- setFilters(), 185
- setMaxDate(), 197
- setMinDate(), 197
- setOnClickListener(), 192
- setSelection(), 184
- setText()
 - creating layouts programmatically, 212
 - EditText, 184
 - TextView, 179
- SharedPreferences interface, 283
- show()
 - custom dialog, 271
 - dialog lifecycle, 268–269
 - support package dialog fragments, 273
- start(), 205
- startActivity()
 - descendant navigation, 401
 - lateral navigation, 400–401
 - launching activities by class name, 107–108
 - temporary transitions, 106
 - startActivityForResult(), 404
 - tapView(), 454
 - testMatchingPasswords(), 453–455
 - testPreConditions(), 450–453
 - text input, 85
 - unregisterOnSharedPreferenceChangeListener(), 285
 - uri parsing, 85
 - writing code in Java, 552
- MinEms attribute, width of TextView, 180**
- Minimizing windows, in workspace, 548**
- Minimum SDK version, 70, 124–125**
- MinLines attribute, height of TextView, 180**
- MinSdkVersion attribute, API level, 124**
- Mistakes, avoiding**
 - in design, 388
 - in development, 393
 - in testing, 435
- Mksdcard command-line tool, 502**
- Mobile applications**
 - design best practices
 - avoiding silly mistakes, 388
 - leveraging diagnostics, 386
 - maintenance and upgrade, 385–387
 - for maximum profit, 383–384
 - meeting users' demands, 380
 - quality guidelines, 384–385
 - rules, 379–380
 - security, 383
 - stability and responsiveness, 381–382
 - third-party quality standards, 385
 - tools for, 387–388
 - user interface, 380–381
 - development best practices
 - avoiding silly mistakes, 393
 - code diagnostics, 391–392

Mobile applications, development best practices (*continued*)

- code reviews, 391
- coding standards, 390
- feasibility testing early and often, 389
- handling defects on single device, 392
- overview of, 388–389
- software development process for, 389
- tools, 393

Mobile development

- acquiring target devices, 366
- application design
 - architectures, 371
 - deploying, 374–375
 - development phase, 373
 - extensibility and maintenance, 371–372
 - interoperability, 372–373
 - limitations of, 370–371
 - supporting and maintaining, 375–377
 - target markets, 375
 - testing applications, 373–374
- best practices
 - avoiding silly mistakes, 393
 - code diagnostics, 391–392
 - code reviews, 391
 - coding standards, 390
 - feasibility testing early and often, 389
 - handling defects on single device, 392
 - overview of, 388–389
 - software development process for, 389
 - tools, 393
- choosing software methodology
 - iteration, 358
 - waterfall, 356–357

configuration management systems, 369–370

documentation

- maintenance and porting, 369
- overview of, 368–369
- test plans for quality assurance, 368–369
- third party requirements, 369

hurdles, 355

managing device database, 361–364

overview of, 355

requirements

- determining feasibility of, 366–367
- developing use cases, 360
- overview of, 357
- project, 358–359
- third-party recommendations and, 360

risks

- assessing, 366
- quality assurance, 367–368

Mobile marketplaces, 375**Mobile network code, alternative resource qualifiers, 341****Mobile operators, OHA, 20–21****Mobile software development history**

- The Brick, 13–15
- form factors, 14
- Google goes wireless, 18–19
- proprietary mobile platforms, 17–18
- WAP, 15–17
- way back when, 11–13

Mobiletuts+ website, development tutorials, 6**Mockups, application, 418****MODE_APPEND, permissions, 303****MODE_PRIVATE, permissions, 303****MODE_WORLD_READABLE, 303**

MODE_WORLD_WRITEABLE, 303

Modes, permission, 303

Modifications, managing fragments, 246

Monitoring, files and directories, 311

Monkey (Exerciser Monkey)

- defined, 434, 502
- testing for unexpected, 433
- testing with, 456

Monkeyrunner API

- automating testing, 428, 456
- defined, 434, 502

Monospace text, in this book, 7

MoreAsserts class, 455

Motorola DynaTAC 8000X, 13–14

Mouse, emulating trackball on emulator, 507

MP3 music files, 84–87

MultiAutoCompleteTextView, 186–188

Multimedia content, files for, 301–302

Multiple APK support, 358–359

Multiple user accounts, restricted profiles, 32

MultiSelectListPreference class, 287

MyFirstAndroidApp application. see Applications, building first

N

Naming conventions

- accessing resources
 - programmatically, 142
- activity for Android project, 71–72
- alternative resources, 338–339
- application, 122–123
- AVD, 61
- coding standards for, 390
- creating AVD, 509
- custom log filters, 551
- new project, 69–70

- packaging application, 463
- renaming in Java, 553–554
- resource files, 156
- storing resource files, 140–141
- test project in IDE, 442
- XML files, 140

Native application support, 33–34

Native code, 33

Native Development Kit (NDK)

- developing Google TV devices, 349
- integrating native code, 33

Navigation

- action vs., 408
- with activities and intents, 110
- alternative resource qualifiers, 343–344
- ancestral, 401, 403
- back, 401
- back stack, 404
- descendant, 401
- design patterns
 - defined, 405
 - DrawerLayout, 239
 - drop-down, 406–407
 - master detail flow, 405, 408–409
 - navigation drawer, 406, 408–409
 - tabs, 406–407
 - targets, 408, 410
- designing tablet, 348
- entry, 400
- external, 404
- with fragments, 404
- Google TV devices, 349
- lateral, 400–401
- launching tasks, 404
- planning, 404–405

Navigation drawer design pattern, 406, 408–409

NDK (Native Development Kit)

- developing Google TV devices, 349
- integrating native code, 33

Nested fragments, 261**Network statistics, viewing with DDMS, 535–537****Network status, in emulator console, 523****Network-driven applications, 371–372****Networking, toggling in emulator, 505****New Android Application settings, 70–71****New applications, fragment support, 259–260****NewDelete() method, 327–328****NewInsert() method, 326****Newsletter, Fierce Developer, 6****NewUpdate() method, 327****Next-generation platform, Android as, 23–25****Nexus devices, 21****Night mode, alternative resource qualifiers, 342****Nine-Patch Stretchable Graphics**

- creating, 157
- device compatibility, 334, 336
- image format, 156
- mobile use, 388
- tablet design, 348
- working with, 157–158, 498–500

Nokia

- 8110 “The Matrix Phone,” 15
- Snake video game on, 14

NotePad demo, 52**NumberPicker control, 199****O****Obfuscation**

- coding standards avoiding, 390
- tools for code, 460

OEM/ODM relationships, 26**OHA (Open Handset Alliance)**

- developing applications, 21–22
- device manufacturers, 19–20
- forming, 19
- Google goes wireless, 18–19
- marketplace, 22–23
- mobile operators, 20–21
- taking advantage of, 22

OnActivityCreated() callback method

- fragments, 247
- implementing ListFragment, 250–253
- implementing WebViewFragment, 255

OnAttach() callback method, 247**OnBackPressed() method, 401****OnBackStackChanged() method, 254****OnCheckedChangeListener() method, 196–197****OnClick() method**

- Button controls, 192
- PasswordMatcher, 441

OnConfigurationChanged() method, 347**OnCreate() method**

- ancestral navigation, 403
- fragments, 247
- initializing static activity data, 102
- intents, 109
- layouts, 212
- logging support, 84
- media, 85
- PreferenceActivity class, 289

OnCreateOptionsMenu() method, 163**OnCreateView() callback method, 247****OnDateChanged() method, 197–198****OnDestroy() method**

- avoiding killing activities, 103
- destroying static activity data, 104
- fragments, 247

- OnDestroyView() callback method, 247**
- OnDetach() callback method, 247**
- OnItemClickListener() method, 237**
- OnItemClickListener class, 236**
- OnListItemClick() method, 237**
- OnOptionsItemSelected() method, 414–415**
- OnPause() method**
 - avoiding killing activities, 103–104
 - emulator, 507
 - fragments, 247
 - stopping/saving/releasing activity data in, 103
- OnRatingChanged() method, 204–205**
- OnResume() method**
 - fragments, 247
 - initializing/retrieving activity data in, 103
- OnRetain NonConfigurationInstance() method, 347**
- OnSaveInstanceState() method, 104**
- OnStart() callback method, 247**
- OnStop() method**
 - adding media, 86–87
 - avoiding killing activities, 103
 - emulator, 507
 - fragments, 247
- Open Handset Alliance. see OHA (Open Handset Alliance)**
- Open Handset Alliance website, 6**
- Open platform, 22**
- OpenFileInput() method, 306–307, 309**
- OpenFileOutput() method, 305–306, 309**
- OpenGL, 331–332**
- OpenGL ES, 127**
- OpenIntents, 131**
- Open-source platform, 23, 25**
- Operators, mobile project risks, 365**
- Organize Imports command, 84**

- Orientation. see Landscape mode; Portrait mode**
- OS (operating system)**
 - for Android development, 26
 - applications as users of, 31
 - configuring for device debugging, 39–40
 - Linux, 31
 - underlying Android architecture, 29–30
- Outside billing, mobile applications, 384**
- Overriding back navigation, 401–402**

P

Packages

- changing name for project, 70
- commonly used, 34
- editing manifest file, 119
- publishing applications
 - preparing, 463–464
 - readying servers/services, 467
 - required resources, 467
 - signing, 465–467
 - steps for, 462–463
 - testing release version, 467
- in SDK, 45
- PagerAdapter, ViewPager control, 239**
- Palm OS (WebOS), 17**
- Parent view, 214**
- Partitioning, user interface. see Fragments**
- PasswordMatcher application**
 - additional tests, 453–455
 - analyzing results, 450–453
 - creating, 441–447
 - overview of, 437–441
 - running using IDE, 450
 - what tests should prove, 441
 - writing tests, 447–450

Passwords, package file, 466**Patterns.** *see* **Design patterns, navigation****Payment, Google Play, 461****Performance**

- emulator, 512–513
- emulator limitations, 526
- testing, 432–433

PerformanceTestCase class, 392**Permissions**

- adding media, 86
- application-enforced, 117–119
- architecture r, 31–32
- content provider
 - modifying data, 325–328
 - requiring special, 319, 322
- file, 303
- groups, 133
- location-based, 89–90
- packaged application, 465–467
- registering, 131–133
- user interface tips for mobile, 381

Permissions tab, IDE manifest file resource editor, 117–119**Persistent storage, 281****Personas, targeting users with, 397–398****Perspectives**

- Android IDE toolbar, 47
- repositioning tabs within, 548

Pixel density

- Google TV devices, 349
- supported screen sizes, 128

Pixel Perfect mode, Hierarchy Viewer, 494, 497**Planning application**

- navigation, 404–405
- objectives, 396–397

Platforms

- applications, 32–35
 - architecture, 29–31
 - differences in Android
 - costs, 27
 - developer learning curve, 26
 - development environment, 5–6, 25–26
 - familiar language, 25–26
 - free and open source, 25
 - free market, 27–28
 - freely available SDK, 25
 - growing platform, 28–29
 - maximizing device compatibility, 331–332
 - next-generation, 23–25
 - overview of, 23
 - powerful apps, 26
 - secure application integration, 26–27
 - downloading SDK sample apps, 57
 - emulator and versions of, 506
 - father of, 19
 - improvements, 28–29
 - proprietary mobile, 17–18
 - security and permissions, 31–32
 - services, 35
 - verifying target, 463
- PlayMusicFromWeb() method, 85–86**
- <plurals>, quantity strings, 149–150**
- PNG files.** *see* **Nine-Patch Stretchable Graphics**
- Policies**
- device support, 365
 - documentation for, 487
 - Google Play, 461

Port numbers

- calling between two emulator instances, 517–518
- messaging between two emulator instances, 518–520

Porting

- identifying opportunities for, 376
- mobile application documentation for, 369

Portrait mode

- alternative resources for, 339, 345–346
- flexible layout controls for, 334
- layout files for, 255–257
- resolving resources, 141–142
- screen size/density for, 332
- switching emulator to, 507
- as tablet default, 348
- using layout resources programmatically, 170

Positions, compatibility, 334**Power button, emulator, 507****Power settings. emulator console, 523–524****Preference class, 287****PreferenceActivity class**

- defined, 247–248
- managing user preferences, 286
- using, 289–291

PreferenceCategory class, 287–289**PreferenceFragment class**

- defined, 248
- displaying, 289–291
- preference headers, 291–296

PreferenceManager, 289**Preferences**

- accessing, 99
- adding, 284–285
- Cloud Save, 296–298
- deleting, 284–285

determining appropriateness, 281

editing, 284–285

finding data, 285–286

functionality, 282

keeping windows under control, 550–551

private, 282

reacting to changes, 285

reading, 283

remote, 297

searching, 283

shared, 282–283

storing values, 282

updating, 284–285

user

- creating resource file, 287

- headers, 291–296

- overview of, 286

- PreferenceActivity class, 289–291

PreferenceScreen class, 287, 289**Preproduction devices, testing on, 428****Presentation class, 267****Pricing application, 459****Pricing tab, Developer Console, 475****Primary entry points, intent filter, 130****Primitive resources, storing, 140–141****Privacy**

- mobile application security, 383

- mobile user demands, 380

- SDK license agreement, 43

Private channel, publishing to Google Play, 478**Private controlled testing, 374****Private group testing, 374****Private keys**

- digital signatures and, 465

- exporting/signing package file, 465–467

- signing application updates, 477

Private permissions, 303

Processes, DDMS, 531–532

Profiles, restricted, 32

Profit

- ad revenue, 462
- billing user, 461
- choosing distribution model for, 459
- mobile application design for, 383–384
- objectives, 396–397

Programming language choices, 32–33

ProgressBar

- activity bar and circles, 202
- indicating progress, 199–202
- RatingBar, 204–205
- responsive mobile applications, 382
- SeekBar, 202–203

ProgressDialog, 266

ProGuard tool, 460, 502

Proguard-project.txt file, 73

Project files, searching, 551

Project requirements, 357–358

Project.properties file, 73

Promoting applications, 487

Property pane, Hierarchy Viewer, 496

Proprietary mobile platforms, 17–18

Prototypes, 419

Publishing application

- alternative marketplaces, 479
- Android documentation, 487
- billing user, 461–462
- collecting statistics, 462
- distribution method, choosing, 459–460
- to Google Play
 - additional options, 475
 - Game Services API, 476
 - overview of, 468, 476

pricing/distribution details, 474–475

private channel, 478

removing, 477–478

return policy, 476–477

sign up, 468–471

Staged Rollouts, 478

translation, 478

upgrades, 477

uploading, 471–473

uploading marketing assets, 473–474

intellectual property protection, 460

packaging

including all required resources, 467

overview of, 462–463

preparing code, 463–464

readying servers/services, 467

signing and, 465–467

testing release version, 467

policies of Google Play, 461

self-publishing, 479–480

Python, 33

Q

Qualifiers, alternative resources, 339–344

Quality

assurance

engineers, 1

mobile project risks, 367–368

documentation for application, 487

guidelines

core applications, 384

improving applications, 385

tablet apps, 384–385

third-party standards, 385

Quantity strings, 149–150

Questions, answered in this book, 2

QuickFix feature, 84, 555–556

Quiz questions & answers

Android IDE and Eclipse tips, 557, 566

application basics, 112, 560

application resource management,
173, 560

bulletproof design and development,
394, 563–564

compatible application design, 350, 563

content providers, 328, 562

DDMS, 545, 565

development environment set up,
52, 559

development tools, 503

dialogs, displaying, 276, 562

emulator, 526, 565

files and directories, 312, 562

fragments, 262, 561

introducing Android, 36, 559

layouts, designing with, 239–240, 561

manifest file, 134, 560

preferences, 298–299, 562

publishing, 481, 564

software development process,
377, 563

testing, 564

testing applications, 457

tools, 565

user experience, planning, 420, 564

user interface building blocks, 207, 561

writing first application, 93–94, 559

R

RadioButton, 190, 194–196

RadioGroup, 190, 194–196

RatingBar, 204–205

Raw file resources

storing, 141

using programmatically, 165

Raw files

defining resources, 164–165

defining XML resources, 164

read byte by byte, 306–308

working with, 164

Reacting to preference change, 285

READ_CONTACTS permission, 322

Reading

files

in default application directory, 306

management practices, 302

raw, 306–308

preferences, 283

XML files, 308–309

Real devices

application design tools, 387

mitigating risk of testing on, 367

testing applications on, 40

testing feasibility early and often, 389

**Real-world activities. mimicking in testing,
426–427**

Receiving intents, 111–112

Records

adding content provider data, 325–326

deleting content provider data,
327–328

updating content provider data,
326–327

Recursion, 382

Refactoring code, in Java, 554

Reference tab, documentation, 486–488

References

activity bars and activity circles, 202

ADB, 491

References (*continued*)

- Android Developers Blog, 382
 - Android IDE and Eclipse tips, 557
 - Android introduction, 36
 - Android topics on book website, 6
 - application anatomy, 113
 - CalendarContract app, 321
 - Cloud Save, 297
 - contacting authors, 7
 - content providers, 329
 - custom indicators, 203
 - DDMS, 545
 - designing/developing applications, 393–394
 - developing Google TV devices, 350
 - development environment, 53
 - development tools, 503
 - dialogs, 276
 - emulator, 527
 - files and directories, 313
 - Google design comp tools, 417
 - Google Experience device, 367
 - Google Play, 460
 - Google Play international laws, 468
 - help documentation, 486
 - layout design, 239–240
 - LinearLayout, 219
 - lint command-line tool, 496
 - manifest file, 135
 - media formats, 164
 - MediaRouter APIs, 350
 - memory analysis, 535
 - multiple APK support, 359
 - NumberPicker, 199
 - OpenIntents, 131
 - permissions, 133
 - planning application navigation, 405
 - preferences, 299
 - ProGuard support, 460
 - publishing alternatives, 479
 - publishing application, 481–482
 - RelativeLayout, 222
 - resources, 173
 - retrieving application, 99
 - screen maps, 405
 - signing, 467
 - software development process, 378
 - source code for this book, 6
 - StrictMode, 433
 - Support Package, 123–124
 - third-party content providers, 328
 - useful websites, 6
 - user interface, 208
 - VoiceMailContract app, 322
 - writing applications, 94
- Referencing resources, 165–166, 171–172**
- Refresh, for build errors, 556**
- Region code, alternative resource qualifiers, 341**
- RegisterOnSharedPreferenceChangeListener() method, 285**
- Registration**
- activities
 - broadcast receivers, 131
 - content providers, 131
 - intent filters, 130–131
 - overview of, 129–130
 - primary entry points, 130
 - services, 131
 - permissions
 - enforced by application, 132–133
 - required by application, 131–132
 - publisher account to Google Play, 468
- Relative positions, compatibility, 334**
- RelativeLayout, 219–222, 348**

Release version, testing, 419, 467

Remote preferences, 297

Remote storage, 383

Removing application, from Google Play, 477–478

Renaming in Java, 553–554

Reorganizing code, in Java, 555

Reports

infringement, 460

tracking crash and bug, 376

viewing statistics, 476–477

Requirements

mobile development, 357–360

platform, 126–128

project, 357–358

SDK license agreement for malware, 43

software development, 37–38

system, 123–126

/res project directory, 74, 137

/res/drawable-*, 74

/res/layout directory, 74, 209

/res/menu directory, 74

Resolution, Google TV, 348–349

Resolving

mysterious build errors, 556

resources, 141–142

Resource editors, 491–492

Resources

accessing programmatically, 142

alternative

caution using layouts, 170

for compatibility, 338

default vs., 141–142

designing tablets, 348

for different orientations, 345–346

maximizing device

compatibility, 333

naming, 338–339

organizing efficiently, 345–347

organizing with qualifiers, 339–344

resolving, 338–339

targeting devices, 347–350

using programmatically, 345

animations

frame-by-frame, 160–161

overview of, 159–160

tweened, 161–162

Boolean

defining in XML, 151

overview of, 151

using programmatically, 151

coding standards, 390

color

defining in XML, 152–153

overview of, 152

using programmatically, 153

color state lists, 158–159

default, 141–142

definition of, 137

dimension

defining in XML, 153–154

using programmatically, 154

drawables

defining in XML, 155–156

simple, 154–155

using programmatically, 156

filename conventions, 156

Google Play filters, 464

images

Nine-Patch Stretchable
Graphics, 157

overview of, 156

using programmatically,
157–158

Resources (*continued*)

integer

- defining in XML, 152
- using programmatically, 152

layouts

- designing in IDE, 168–170
- overview of, 166–168
- using programmatically, 170–171

menus

- defining in XML, 162–163
- using programmatically, 163

raw files

- defining, 164–165
- overview of, 164
- using programmatically, 165

referencing

- overview of, 165–166
- system, 171–172

screen compatibility mode, 336

setting with IDE, 143–146

storing, 137–138

strings

- arrays, 150–151
- bold, italic, and underlined, 147
- format, 148
- formatting examples, 147
- overview of, 147
- quantity, 149–150
- using programmatically, 148–149

task management, 404

testing availability for application
release, 467

value types

- overview of, 138–140
- storing graphics and files, 141
- storing other, 141
- storing primitive, 140–141

XML files

- defining raw, 164
- overview of, 163
- storing most resources in, 492
- using programmatically, 164

Resources class, 152**Resources tab, IDE resource editor, 143–146****Responsiveness**

- device defects in, 424
- mobile design, 381–382

Restores, testing application, 432**Restricted profiles, 32****/res/values* directory, 74–75****Return policy**

- choosing distribution model, 460
- Google Play, 476–477

Revenues, stakeholder objectives, 396–397**Reverse domain name notation, 70****Reverse engineering, trade secrets, 460****Reviews, code, 391****RGB color values, storing, 152–153****Rights, SDK license agreement, 43–44****RIM Blackberry OS, 17****Ringtones**

- emergence of, 16
- as network-driven applications, 371

Risk

mobile project

- acquiring target devices, 366
- feasibility of application
requirements, 366–367
- identifying target devices, 364–366
- quality assurance, 367–368
- SDK license agreement for, 44

Rollbacks, 430**Rooted devices, 426****RSA key fingerprint, 91–92**

Rs:ResEnum application, 172
Rubin, Andy, 19
Rules, mobile application, 379–380, 388–389
Run As icon, 76
Run configuration
 creating in IDE, 514–515
 debugging in emulator, 80
 launch configuration, 62, 64–65,
 75–76
 running application in emulator, 76–79
Running first test, 450
Runtime environment, 31

S

Sample applications, SDK
 adding project to workspace, 57–60
 downloading, 57–58
 error, 58
 overview of, 51–52
 testing with, 55–56
Samsung
 growth of Galaxy S line, 21
 publishing to, 479
Sanity testing, validating, 426
Scalability
 containers, 334
 of Nine-Patch graphics, 497–501
 remote servers/services, 430
Screen size
 alternative resource qualifiers, 341
 compatibility
 maximizing device, 331–333
 Nine-Patch Stretchable
 Graphics, 336
 supporting types, 335–336
 tools for, 334
 working square principle, 336–338

customization and, 359
 Google Play restrictions, 129
 information about device, 334
 preference headers and, 296
 specifying supported, 128

Screens

aspect ratio, 342
 capturing emulator/device, 542–543
 compatibility for legacy
 applications, 336
 maps, planning navigation, 405
 orientation
 alternative resource qualifiers, 342
 Google TV devices, 349
 tablets, 348
 pixel density
 alternative resource qualifiers,
 341–342
 Google TV devices, 348–349
 resolution, Google TV devices,
 348–349
 sharing devices/applications, 29

Screenshots, 5

Scripting languages, 33

ScrollView, 238, 334

SD cards

choosing capacity for AVD, 51, 62–64
 external storage using, 311–312
 removable storage, 381

SDK (software development kit)

basic installation process, 38
 code names for Android, 24–25
 configuring development
 environment with, 37
 core framework, 45–46
 core tools, 47–50
 deprecated methods in, 266
 documentation, 44

SDK (software development kit) (*continued*)

- documentation for, 487
- enforcing system requirements
 - maximum version, 124–125, 126
 - target version, 125–126
 - targeting specific SDK versions, 123–124
- latest version, 25
- launching Hierarchy Viewer, 494–495
- layout, 178–179
- License Agreement, 43–44
- license agreement, 56–57
- Manager
 - launching, 38–39, 47–48
 - overview of, 48–49
 - upgrading SDK, 42
 - viewing documentation, 44
- packages in, 45
- problems with, 42
- sample applications, 51–52
- targeting multiple platform versions
 - within single application, 358
- testing mobile applications, 373
- third-party Android APIs, 45–46
- upgrading, 37, 42
- version 4.3 (Jelly Bean)
 - freely available, 25
 - used in this book, 5–6
- views, 177

Search menu, project files, 551**Security**

- defects in device, 424
- mobile application design, 383
- testing remote server, 430
- underlying architecture, 31–32

SeekBar, 202–203**SelectAll() method, 184****Selection event handling, 236–237****<selector> resource type, 158–159****Self-publishing applications, 479–480****Self-signing applications, 465****SendKeys() method, 454****Servers**

- managing changes for live, 376
- quality assurance testing, 368
- readying for application release, 467
- testing applications, 429–430
- testing remote, 429

Service class, 98, 111**Services**

- Android platform, 35
- definition of, 98
- Google, 35, 487
- location-based, 88–89
- overview of, 110–111
- purposes of, 111
- readying for application release, 467
- registering, 131
- testing applications, 429–430
- testing remote, 429

ServiceTestCase class, 392**SetBase() method, Chronometer, 205****SetContentView() method**

- creating layouts, 210
- creating layouts programmatically, 212
- fragments, 246
- ListActivity, 237
- ProgressBar, 202

SetFilters() method, input filters, 185**SetMaxDate() method, DatePicker, 197****SetMinDate() method, DatePicker, 197****SetOnClickListener() method, Buttons, 192****SetSelection() method, 184**

- SetText() method**
 - creating layouts programmatically, 212
 - EditText, 184
 - TextView, 179
- Settings application, emulator, 506**
- Settings class, 322**
- Settings content provider, 322**
- ShapeDrawable class, 155**
- SharedPreferences class**
 - adding, updating, deleting preferences, 284–285
 - defined, 99
 - for multiple activities, 282–283
 - overview of, 282
 - reacting to preference changes, 285
 - searching and reading preferences, 283
 - for single activity, 282
- Short Message Service. *see* SMS (Short Message Service)**
- Show() method**
 - custom dialog, 271
 - dialog lifecycle, 268–269
 - support package dialog fragments, 273
- Signing applications**
 - with debug key, 465
 - digitally with private key for publishing, 465
 - for trust relationships, 32
- Simple_fragments_ layout.xml resource file, 255–257**
- SimpleActionBar application, 411**
- SimpleAltResources application, 345–346**
- SimpleFiles application, 304**
- SimpleFragDialogActivity class, 269, 273**
- SimpleFragments application, 248**
- SimpleFragmentsActivity class, 257–258**
- SimpleLayout application, 215**
- SimpleNavigation application, 400**
- SimplePreferences application, 281**
- SimpleScrolling application, 238**
- Single payment, mobile application design, 383**
- SkeletonApp/SkeletonAppTest, 52**
- Sketches, screen layouts, 417**
- Slider mobile phones, 15**
- SlidingDrawer class, 240**
- Smart watches, 29**
- Smoke tests, 427**
- SMS (Short Message Service)**
 - in emulator console, 522
 - in Emulator Control pane, 540–541
 - payments through, 16
 - between two emulator instances, 518–520
- Snake game**
 - demo application, 52
 - Nokia, 14
 - project. *see* Applications, writing first
- SnakeDebugConfig, 64–68**
- Snapshot feature, Emulation Options, 62, 66**
- Soc.io.Mall, publishing to, 479**
- Software development kit. *see* SDK (software development kit)**
- Software development process**
 - acquiring target devices, 366
 - configuration management systems, 369–370
 - device database management, 361–364
 - documentation
 - maintenance and porting, 369
 - overview of, 368–369
 - test plans for quality assurance, 368–369
 - third party requirements, 369
 - methodologies, 356–358

Software development process (*continued*)

mobile applications

architectures, 371

deploying, 374–375

development phase, 373

development team hurdles, 355

extensibility and maintenance,
371–372

interoperability, 372–373

limitations of, 370–371

supporting and maintaining,
375–377

target markets, 375

testing applications, 373–374

overview of, 355

requirements

configuring development
environment, 37–38

determining feasibility of, 366–367

overview of, 358–360

risks

assessing, 366

quality assurance, 367–368

upgrades, 22

Software methodology

iteration, 356

waterfall, 356–357

Solitaire game, 371**Source code**

test project within IDE, 445–446

for this book, 6

Source control

IDE integration with, 547

mobile development, 369–370

resolving build errors, 556

Sp units, layouts/graphics, 154**Space view, layout, 230****Spinner control**

defined, 51

editing preferences, 291

filtering user choices, 188–190

packages supporting, 34

styles of, 183

SpinnerTest, 51**Split ActionBar, 413****Spotlight, documentation, 487****sqlite3 command-line tool**

defined, 502

mobile development tool, 393

as testing tool, 434

/src folders, 74**Stable applications, mobile design,
381–382****Stack Overflow website, Android, 6****Staged Rollouts, Google Play, 374, 478****Stakeholder objectives, planning application,
396–397****Standalone applications**

DDMS, 529–530

extensibility/maintenance of, 371–372

mobile design, 371

Standards

mobile application

coding, 390

design, 384–385

third-party testing, 430–431

Start() method, Chronometer, 205**StartActivity() method**

descendant navigation, 401

lateral navigation, 400–401

launching activities by class name,
107–108

temporary transitions, 106

StartActivityForResult() method, 404

Startup

- emulator options, 513
- responsive mobile applications, 382

State

- CheckBox control, 193
- defects in devices, 424
- device clean start, 426
- responsive mobile applications, 382
- Switch control, 194
- ToggleButton control, 193–194

Static activity data, 104**Statistics-gathering**

- application diagnostics, 386
- Google Play Developer console, 476–477
- before publishing application, 462

Stencils, design comp tool, 417**Stopping processes, DDMS, 532****Storage**

- device database, 360–363
- file management practices, 302, 311–312
- files and directories. *see* **Directories; Files**
- minimizing local, 382
- of preferences, 281
- removable user, 381
- resources, 137–138

Storyboards, UI, 418–419**Stretchable graphics. *see* **Nine-Patch******Stretchable Graphics****StrictMode, 302, 433****String resource editor, 492****String resources**

- arrays, 150–151
- bold, italic, and underlined, 147
- as format strings, 148
- formatting examples, 147

- overview of, 147
- quantity, 149–150
- setting values in IDE, 143–146
- using programmatically, 148–149

Strings.xml tab , IDE resource editor, 143–146**Styles**

- communicating identity, 416–417
- documentation for
 - Android design, 486
 - Javadoc, 556
- providing visual feedback, 418
- screen layouts, 417
- user interface
 - Button controls, 183, 191
 - EditText controls, 183
 - Spinner controls, 183

Subscription billing, mobile design, 384**Supplementary materials, 6****Support, mobile application, 375****Support Library**

- ActionBar compatibility, 415
- designing compatibility, 335–336
- increasing market size, 358

Support Package

- fragments
 - legacy applications, 259
 - overview of, 258
 - targeting older platforms in new applications, 259–260
- increasing market size, 357–358
- linking to project, 260–261
- as mobile development tool, 393
- targeting specific SDKs, 123–124

Support4Demos application, 52**Support7Demos application, 52****Support13Demos application, 53**

SupportAppNavigation, 53

SupportFragDialog application, 272–275

SupportFragDialogActivity, 273

<supports-gl-texture> tag, manifest file, 129

<supports-screens> tag

- designing compatibility, 335–336
- packaging application, 463

Switch control, 190, 194

Symbian OS, 17

Sync, testing application, 432

System Information pane, DDMS, 541–542

System requirements

- enforcing, 123
- targeting SDK versions
 - expanding range, 123–124
 - maximum, 126
 - minimum, 124–125
 - specific, 123–124
 - specifying, 125–126

System resources

- caution when using, 172
- referencing, 171–172

Systrace performance analysis tool, 435, 502

T

Tab navigation, 406–407

TabActivity class, 248

TabHost control, 248

TableLayout

- defined, 178
- user interface design, 334
- using, 224–227

TableRow, 224–227

Tablet devices

- action bar display, 413
- app quality guidelines, 384–385

- designing, developing and publishing, 347–348
- Jelly Bean, 22

Tabs, closing unwanted, 550

Tags

- adding customized comment, 552
- creating custom log filters, 551

TapView() method, 454

Targets

- compatibility
 - Google Chromecast devices, 350
 - Google TV devices, 348–350
 - tablet devices, 347–348
- devices, 364–367
- navigation, 408, 410
- platforms, 463
- SDK versions
 - maximum, 126
 - minimum, 124–125
 - specific, 123–124
 - specifying, 125–126
- users
 - entity discovery and organization, 398
 - mocking up application, 418
 - overview of, 418
 - personas, 397–398
 - prototypes, 419
 - UI storyboards, 418–419
 - use cases/use case diagrams, 398–399

TargetSdkVersion attribute, API level, 124

Tasks

- activities within, 97
- launching, 404
- organizing IDE, 551–552

TDD (Test Driven Development), 437

- Team objectives, planning application, 396**
- Technical specifications, devices, 387**
- Telephony status, Emulator Control pane, 540**
- Terminology, actions on devices, 424**
- Test instrumentation, 119**
- Test MatchingPasswords() method, 453–455**
- Test release, 374**
- Test servers, 430**
- Testing**
 - adding additional tests, 453–455
 - analyzing results, 450–453
 - in-app billing, 433
 - applications, 373–374
 - automated, 428
 - avoiding mistakes, 435
 - backups, 432
 - black-box, 429
 - for compatibility, 333, 336
 - conformance, 432
 - content provider code, 315
 - creating test project, 441–447
 - defect-tracking systems, 423–424
 - development environment
 - adding project to workspace, 57–60
 - adding samples with SDK Manager, 56–57
 - creating AVD, 61–62
 - launch configuration, 62–65
 - overview of, 55–56
 - running application in emulator, 66–68
 - device upgrades, 432
 - on emulator vs. device, 428–429
 - environment
 - clean start state, 426
 - managing device configurations, 425–426
 - overview of, 55–56, 425–427
 - real-world, mimicking, 426–427
 - firmware upgrades, 376
 - installations, 432
 - integration points, 430
 - internationalization, 432
 - killer apps, 433–434
 - PasswordMatcher
 - adding additional tests, 453–455
 - analyzing test results, 450–453
 - creating, 441–447
 - overview of, 437–441
 - running first test using IDE, 450
 - writing tests, 447–450
 - performance, 432–433
 - with personal devices, 360–361
 - physical hardware for, 426
 - quality assurance risks, 367–368
 - on real devices, 302–303
 - release build, 419
 - release version of package file, 467
 - running using IDE, 450
 - servers/services, 429–430
 - that use external storage, 312
 - third-party standards, 430
 - this book for, 1
 - tools
 - Android, 433–434
 - other automated, 455–457
 - translations, 478
 - unexpected, 433
 - unit
 - analyzing results, 450–453
 - APIs/assertions, 450
 - developing, 391–392
 - with JUnit, 436–437
 - running first test, 450

Testing (*continued*)

- upgrades, 431
- validating builds/designing smoke tests, 427
- visual appeal/usability, 430
- white-box, 429
- writing programs for, 435–436
- writing test program, 435–436

TestMatchingPasswords() method, 453**TestPreConditions() method, 450–453****Text**

- autocompletion, 186–188
- displaying with TextView
 - contextual links, 180–182
 - displaying text, 178
 - layout and sizing, 179–180
- retrieving data with text fields
 - constraining input, 184–185
 - EditText controls, 183–184

Text input method, alternative resource qualifiers, 343**TextOff attribute, ToggleButton, 193****TextOn attribute, ToggleButton, 193****TextView class, 180****TextView control**

- accessing, 178
- autocomplete, 186–187
- contextual links, 180–182
- creating layout using XML resources, 210
- creating layouts programmatically, 211–213
- displaying, 178
- layout, 168–171
- layout and sizing, 179–180
- retrieving text input with EditText, 183–184

Themes, ActionBar compatibility, 415**Third parties**

- Android SDKs, 45–46
- billing methods, 384
- content providers, 328
- device databases, 364
- device defects in
 - nonconformance, 424
- distribution model, 460
- documentation done by, 369
- documentation requested by, 369
- mobile requirements, 360
- private keys for, 465
- quality standards, 385
- stores, 28
- support, 33–34
- testing standards, 430–431

Threads. see also Main UI thread

- Debug, 433
- monitoring activity with DDMS, 532–533

Thumbing, 381**TicTacToeLib demo, 52****TicTacToeMain demo, 52****Time**

- displaying
 - AnalogClock control, 206–207
 - DigitalClock control, 206
 - TextClock control, 206
- passing of, with Chronometer, 205–206
- retrieving, with TimePicker, 199

TimePicker control, 190**TimePickerDialog class, 267****“Time-waster” games, 14****Title bar, ProgressBar in, 201****T-Mobile G1, 20****Toast messages, 192, 418**

Toggle Breakpoint, 83

ToggleButton control, 190, 193–194

Tokenizer, AutoCompleteTextView, 186–188

Tools

ADB, 434, 490

ADT plugin for Eclipse, 25

android command-line tool, 501

Android documentation, 487–489

Android Hierarchy Viewer

application design, 434

launching, 494–495

Layout View mode, 495–496

overview of, 493–494

Pixel Perfect mode, 497

user interface optimization,
496–497

Android IDE, 25–26

Android Studio, 26

application design, 386–387

AVD Manager, 48

bmgr, 435, 501

calling between two instances,
517–518

compatibility, 333–334

configuring GPS location of, 516–517

configuring startup options, 513

console

monitoring network status, 523

other commands, 524

overview of, 520–521

power settings, 523–524

sending GPS coordinates, 523

simulating incoming calls, 521

simulating SMS messages, 522

DDMS

Android IDE toolbar, 47

copying files to and from
device, 302

debugging with, 434, 490–491

File Explorer and, 285–286

forcing app to crash, 83

memory profiling with, 387

mobile application debugging
tool, 393

as testing tool, 434

dmtracedump, 502

draw9patch, 497–501

Eclipse ADT plug-in, 47

Eclipse IDE plug-in, 25

emulator

application design, 373–374, 386
calling between two instances,
517–518

configuring GPS location of,
88–89, 516–517

configuring startup, 513

console, 520–524

creating AVD, 509–510

creating AVD with custom
hardware settings, 510–511

debugging in, 80–83

enjoying, 524–525

launching from Android Virtual
Device Manager, 515–516

launching to run application,
76–79, 513–515

launching with specific AVD, 512

limitations of, 525–526

maintaining performance,
512–513

messaging between two instances,
518–520

overview of, 48, 50, 505–506

power of, 489

simulating real device, 506–507

testing on, 434

testing on device vs., 428–429

Tools, emulator (*continued*)

- using Android Virtual Device Manager, 508–509
- working with AVDs, 507–508
- enjoying, 524–525
- etc1tool, 502
- Exerciser Monkey, 434
- hprof-conv, 502
- launching from Android Virtual Device Manager, 515–516
- launching to run application, 513–515
- launching with specific AVD, 512
- limitations of, 525–526
- lint, 434
- logcat, 502
- LogCat, viewing log data, 490
- LogCat utility, 434
- maintaining performance, 512–513
- messaging between two instances, 518–520
- mksdcard, 502
- mobile applications, 393
- monkey, 502
- monkeyrunner API, 434, 502
- Nine-Patch Stretchable Graphics, 497–500
- overview of, 485, 505–506
- ProGuard tool, 502
- references
 - complete list of, 500
 - latest information, 485
 - Tools Project Site, 6
- resource editors, 490
- SDK, 25
- SDK Manager, 48
- simulating real device environment, 506–507
- sqlite3, 434, 502
- systrace, 435, 502

- testing, 433–434

- traceview, 433–434, 502

- UI designer, 490–493

- UiAutomation class, 434, 457

- uiautomator, 434, 456, 502

- working with AVDs

- creating AVD, 509–510

- creating AVD with custom hardware settings, 510–511

- overview of, 507–508

- using Android Virtual Device Manager, 508–509

- zipalign, 467, 502

Tools tab, Android documentation, 487

Top-level directories, 304

Touchscreen

- alternative resource qualifiers, 343

- tablets, 348

- user interface for mobile, 380

TouchUtils class, 392, 454, 455

Trace logs, 433

Traceview tool, 502

Trackball, emulating on emulator, 507

TrafficStats class, 535–537

Training, documentation for, 486

Transitions

- activity. *see* **Intents**

- providing visual feedback, 418

Translation applications, 478

Transmissions

- of private data, 383

- testing remote servers, 430

Trust relationships

- application signing for, 32

- private keys for, 465

TV devices, developing Google, 348–349

Tweened animation sequences, 161

U

UI designer, 492–493

UI mode, alternative resource qualifiers, 342

UI storyboards, 418–419

UI threads. see Main UI thread

UiAutomation class, 434, 457

UiAutomator testing framework

defined, 434, 502

overview of, 456

Underlined strings, 147

Undo pattern, visual feedback, 418

Undocumented APIs, 34

Unexpected, testing application for, 433

Uniform Resource Identifiers (URIs), 32

Unit tests

assertions and, 450

code diagnostics, 391–392

with JUnit, 436–437

running using IDE, 450

Unlocking emulator, 67

Unpublish action, Google Play, 477–478

**UnregisterOnSharedPreference
ChangeListener() method, 285**

Updates

ADT Bundle, 37, 55

application, on Google Play, 477

content provider records, 326–327

to fragments, 246

mobile application design for easy,
385–386

preferences, 284–285

SDK, 37

upgrades vs., 386

Upgrades

Android marketplace, 22

choosing application version
system, 370

Google Play application, 477

mobile application design for easy,
385–386

SDK, 42

testing application, 431, 432

testing firmware, 376

testing remote server, 430

updates vs., 386

version code managing, 122

Uploading

application to Google Play, 471–473

marketing assets to Google Play,
473–474

Uri parsing methods, 85

URIs (Uniform Resource Identifiers), 32

Usability

defects in device, 424

testing mobile application, 430

upgrades, 22

USB debugging

allowing, 91–92

configuring hardware for devices, 42

USB driver, debugging, 39–40

Use case diagrams, 398–399

Use cases, 360, 398–399

User experience

communicating identity,
416–417

designing screen layouts, 417

emulator, on platform versions, 506

encouraging action

action bars, 411–415

from within application's
content, 416

dialogs, 415

menus, 410–411

overview of, 408–410

User experience (*continued*)

navigation

- ancestral, 401, 403
- back, 401
- back stack, 404
- descendant, 401
- design patterns, 405–408
- entry, 400
- external, 404
- with fragments, 404
- lateral, 400–401
- launching tasks, 404
- planning application, 404–405
- reference for learning about, 400

objectives

- stakeholder, 396–397
 - team, 396
 - user, 396
- observing target users for usability, 418–419
- planning application for, 395
- providing visual feedback, 418
- targeting users
- entity discovery and organization, 398
 - personas, 397–398
 - use cases/use case diagrams, 398–399

User flows, navigation, 405**User interface**

- actions originating from, 415
- buttons
 - action, 411–413
 - basic, 190–192
 - check boxes, 193
 - CheckBox and ToggleButton, 193–194
 - defined, 190

image, 192

overview of, 190

radio, 194–197

RadioGroup and RadioButton, 194–197

styles of, 183

switches, 190

toggles, 193–194

using basic, 190–192

coding standards, 390

compatibility

fragments, 335

nine-patch stretchable graphics, 336

specific screen types, 335–336

Support Library, 335

tools, 333–334

working squares, 336–338

controls, 177–178

data-driven containers

adapters, 234–235

arrays, 233–234

binding data, 235–236

click events, handling, 236–237

headers/footers, 237

lists of items, 237

overview of, 233

dates/times/numbers, retrieving, 197–199

dialogs

alert, 266–267

attributes, 268–269

basic, 266–267

character pickers, 266–267

customizing, 270–271

date pickers, 266–267

dismissing, 269–270

- fragment method, 267–270
- fragments, Support Package, 270–275
- legacy method, 265
- lifecycle, 268–270
- presentation, 267
- presenting actions to users, 415
- progress, 266–267
- providing visual feedback, 418
- showing, 269
- time pickers, 267
- documentation, 368–369
- fragments
 - Activity classes, 257–258
 - attaching, 246–247
 - back navigation, 401–402
 - defining, 245–246
 - designing applications, 248–249
 - designing compatibility, 334–335
 - designing tablets, 348
 - detaching, 246–247
 - dialog, 248, 267–270
 - dialog, support package, 271–275
 - layout resource files, 255–257
 - legacy support, 259
 - lifecycle, 244–245
 - linking support to your project, 260–261
 - ListFragment implementation, 250–253
 - managing modifications, 246
 - navigating with, 404
 - nested, 261
 - overview of, 243
 - special types, 247–248
 - support, 259–261
 - understanding, 243–245
 - WebViewFragment implementation, 254–255
- indicators
 - activity bars and activity circles, 202
 - adjusting progress, 202–203
 - clocks, 206–207
 - customizing, 203
 - Hierarchy Viewer performance, 496
 - progress bars, 199–202
 - ratings, 204–205
 - time passage, 205–206
- layouts
 - built-in classes, 215–217
 - containers. *see* **Containers**
 - creating programmatically, 211–214
 - creating using XML resources, 209–211
 - frames, 222–224
 - grids, 228–230
 - linear, 217–219
 - multiple, 230–232
 - organizing, 214–215
 - overview of, 178
 - relative, 219–222
 - tables, 224–227
- mobile
 - design tips for devices, 380–381
 - meeeting user demands, 380
 - walking-and-chewing gum analogy, 430
- navigation
 - designing tablets, 348
 - developing Google TV devices, 349

User interface (*continued*)

Nine-Patch Stretchable Graphics

- creating, 157
- device compatibility, 336
- image format, 156
- mobile use, 388
- tablet design, 348
- working with, 498–500

optimizing contents with Hierarchy Viewer, 496

preferences

- accessing, 99
- adding, 284–285
- Cloud Save, 296–298
- creating resource file, 287
- deleting, 284–285
- determining appropriateness, 281
- editing, 284–285
- finding data, 285–286
- functionality, 282
- headers, 291–296
- PreferenceActivity class, 289–291
- private, 282
- reacting to changes, 285
- reading, 283
- remote, 297
- searching, 283
- shared, 282–283
- storing values, 282
- updating, 284–285
- user, 286

Spinner controls, 188–190

switches, 190

tablets, 348

text display

- contextual links, 180–182
- layout and sizing, 179–180
- with TextView, 179

text input

- with autocompletion, 186–188
- constraining with input filters, 184–185
- retrieving, 183–184

time

- displaying, 206–207
- displaying passage of, 205

view, 177

User objectives, applications, 396–397**User preferences**

- creating resource file, 287
- headers, 291–296
- overview of, 286
- PreferenceActivity class, 289–291

UserDictionary content provider, 321–322**<uses-configuration> tag, manifest file**

- overview of, 126–127
- packaging application, 463

<uses-feature> tag, manifest file

- filtering Google Play store, 121
- packaging application, 463
- specifying required device features, 127–128

<uses-library> tag, manifest file

- defined, 129
- packaging application, 463

<uses-permission> tag, manifest file

- accessing content providers, 319, 322
- packaging application, 463

<uses-sdk> tag, manifest file

- defined, 124
- packaging application, 463

V**Validation**

- device defects in input, 424
- with visual feedback, 418

Value types, resources, 138–141

Values

- setting with IDE, 143–146
- storing preferences as, 281–282

Variables, 554–555**Version code, application identity, 122****Version name, application identity, 122****VersionCode attribute, 370, 464, 477****Versioning**

- for application identity, 122
- enforcing system requirements
 - maximum SDK version, 126
 - minimum SDK version, 124–125
- overview of, 123
- target SDK version, 125–126
- targeting specific SDK versions, 123–124
- packaging application, 463
- scheme for, 370
- testing remote servers, 430

VersionName attribute, <manifest> tag, 464**Videos, design documentation, 486****View class**

- controls, 177
- defined, 177
- layouts using XML resources, 209–210
- ViewGroup vs., 214

View containers

- defined, 214
- DrawerLayout, 238
- GridView, 233–237
- ListView, 233–237
- Scroll View and HorizontalScrollView, 238
- using ViewGroup subclasses as, 214–215
- ViewPager, 238
- ViewSwitcher, 238

View control

- designing compatibility, 334
- multiple layouts on screen, 230–231
- placing on action bar, 411
- TableLayout, 225–227

ViewAsserts class, 455**ViewGroup, 209–210, 214–215. see also Containers****ViewManager, 35****ViewPager container, 238****Views**

- controls, 177–178
- layout control, 178–179
- user interface, 177

ViewSamples application, 178**ViewStub, 232****ViewSwitcher control, 238****Visibility, ProgressBar indicators, 202****Visual appeal/usability, testing, 430****Visual feedback, 418**

W**Walking-and-chewing gum analogy, 430****Walled garden approach, WAP portal, 17****Wallpaper**

- customizing emulator, 524
- designing, 371–372
- emergence of, 16
- as network-driven application, 371

WAP (Wireless Application Protocol), 15–17**Waterfall approach, 356****Web applications, for Android devices, 33****Web cameras, in emulator, 526****WebOS (Palm OS), 17****WebView control**

- implementing WebViewFragment, 254
- loading with ListView, 249

WebView control (*continued*)

- organizing into fragments, 244
- WebViewFragment hosting, 248

WebViewFragment class

- defined, 248
- designing fragments, 249–250
- implementing, 254–255

White-box testing, 373, 429**Widgets, 177, 524****Width, TextView, 180****WindowManager, 35****Windows, in workspace**

- configuring for device debugging, 39–40
- keeping under control, 550–551
- maximizing/minimizing, 548
- viewing side by side, 548–550

Wireframe model display, Hierarchy Viewer, 496**Wireframes, screen layouts, 417****Wireless Application Protocol (WAP), 15–17****Wireless Developer Network website, 6****WML (Wireless Markup Language), 16****Working square principle, 336–338****Workspace**

- Android IDE tips
 - closing unwanted tabs, 550
 - controlling windows, 550–551
 - customizing log filters, 551
 - maximizing/minimizing windows, 548
 - repositioning tabs, 548
 - searching project, 551
 - source control services, 547
 - tasks, 551–552
 - viewing two sections of same file, 550

- viewing windows side by side, 548–550

creating/configuring new project, 69–73

organizing into perspectives, 47

writing first project to Android IDE, 57–60

WRITE_CONTACTS permission, 322**Writing**

applications. *see* **Applications**, writing first

files

- in default application directory, 305–306

- to external storage, 311–312

- management practices, 302

software documentation, 368–369

test programs, 435–436

tests

- creating project, 441–447

- standard steps for, 447–449

- unit, 437

X**XDA-Developers Android Forum, 7****XML**

escaping, 148

filenames, 140

reading files, 308–309

resources

- Boolean files in, 151

- color files in, 152–153

- defining raw, 164

- dimension files in, 153–154

- drawable files in, 155–156

- integer files in, 152

- layout files in, 255–257

- menu files in, 162–163
- overview of, 163
- preference files in, 287–289
- stored in, 492
- using programmatically, 164

SAX support package, 34

- storing, 141
- tweened animation sequence in, 161

Z

Zipalign command-line tool

- defined, 502
- signing package files, 467