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

“*Introduction to Android Application Development* is a great resource for developers who want to understand Android app development but who have little or no experience with mobile software. This fifth edition has a bunch of great changes, from using Android Studio to understanding and implementing navigation patterns, and each chapter has quiz questions to help make sure you’re picking up the vital info that fills this book.”
—Ian G. Clifton, author of *Android User Interface Design*

“Revamped, revitalized, and refreshed! *Introduction to Android Application Development, Fifth Edition*, is a wonderful upgrade to an already impressive compendium. Common pitfalls are explained, new features are covered in depth, and the knowledge that the book is geared to cover everything from introduction of a concept to learning how to implement it into your app makes this a great choice for new developers who are ready to make the jump into Android development. Being already accustomed to the professional work and experience that Annuzzi et al., bring to the table, you will be grateful to have expert insight along with the care and instruction that developers of all skill levels can benefit from.”
—Phil Dutson, solution architect, ICON Health & Fitness

“Best technical summary of Material Design implementation I’ve seen outside the Android sample docs.”
—Ray Rischpater, software development manager, Uber

“*Introduction to Android Application Development* is well written and fulfills the requirements of developers, project managers, educators, and entrepreneurs in developing fully featured Android applications. In addition, it emphasizes quality assurance for mobile applications, teaches you how to design and plan your Android application, and teaches the software development process through a step-by-step, easy-to-understand approach. I recommend this book to anyone who wants to not just focus on developing apps, but also to apply tips and tricks and other tools for project management in their development of successful applications.”
—Bintu Harwani, founder of MCE (Microchip Computer Education)
This page intentionally left blank
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.
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
Title: Introduction to Android application development : Android essentials / Joseph Annuzzi, Jr., Lauren Darcey, Shane Conder.
Identifiers: LCCN 2015037913 | ISBN 9780134389455 (pbk. : alk. paper)
Classification: LCC QA76.76.A65 A56 2016 | DDC 005.3—dc23
LC record available at http://lccn.loc.gov/2015037913

Copyright © 2016 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. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearsoned.com/permissions/.

Some figures that appear in this book have been reproduced from or are modifications based on work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/).

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 Attribution 3.0 License. See https://developers.google.com/site-policies.

Screenshots of Google Products follow these guidelines:

The following are registered trademarks of Google:
Android™, Chrome™, Google Play™, Nexus™, Dalvik™, Google Maps™, Google+™, Google TV™, Google and the Google logo are registered trademarks of Google Inc.

ARM is a registered trademark of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

Altium® and Altium Designer® are trademarks or registered trademarks of Altium Limited or its subsidiaries.

Qualcomm and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries.

Cyanogen is a trademark of Cyanogen Inc., registered in certain countries.

CyanogenMod is a trademark of CyanogenMod, LLC, registered in the United States.

JetBrains® and IntelliJ®, are registered trademarks owned by JetBrains s.r.o.


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

First printing December 2015
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: Platform Overview**
- 1. Presenting Android 11
- 2. Setting Up for Development 31
- 3. Creating Your First Application 51

**II: Application Basics**
- 4. Understanding Application Components 77
- 5. Defining the Manifest 95
- 6. Managing Application Resources 111
- 7. Exploring Building Blocks 147
- 8. Positioning with Layouts 179
- 9. Partitioning with Fragments 213

**III: Application Design Essentials**
- 10. Architecting with Patterns 237
- 11. Appealing with Style 269
- 12. Embracing Material Design 285
- 13. Designing Compatible Applications 319

**IV: Application Development Essentials**
- 14. Using Android Preferences 345
- 15. Accessing Files and Directories 363
- 16. Saving with SQLite 377
- 17. Leveraging Content Providers 393
V: Application Delivery Essentials
18 Learning the Development Workflow 411
19 Planning the Experience 437
20 Delivering Quality Applications 449
21 Testing Your Applications 467
22 Distributing Your Applications 499

VI: Appendixes
A Tips and Tricks: Android Studio 527
B Quick-Start: Android Emulator 539
C Quick-Start: Android Device Monitor 567
D Mastery: Android SDK Tools 585
E Quick-Start: Gradle Build System 603
F Answers to Quiz Questions 623

Index 631
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 for This Book  6
Conventions Used in This Book  6
Where to Find More Information  7
Contacting the Authors  8

I: Platform Overview

1 Presenting Android  11
The Android Open Source Project (AOSP)  11
The Open Handset Alliance  12
  Google Goes Mobile First  12
  Introducing the Open Handset Alliance  12
  Joining the Open Handset Alliance  13
Manufacturers: Designing Android Devices  13
Mobile Operators: Delivering the Android Experience  14
Apps Drive Device Sales: Developing Android Applications  14
  Taking Advantage of All Android Has to Offer  15
Android: Where We Are Now  15
Android Platform Uniqueness  16
  Android: The Code Names  16
  Free and Open Source  17
  Familiar and Inexpensive Development Tools  17
  Reasonable Learning Curve for Developers  18
  Enabling Development of Powerful Applications  18
  Rich, Secure Application Integration  19
3 Creating Your First Application 51

Testing Your Development Environment 51
  Importing the BorderlessButtons Sample into Android Studio 52
  Using the Preinstalled AVD for Running Your BorderlessButtons Project 54
  Running the BorderlessButtons Application in the Android Emulator 55

Building Your First Android Application 57
  Creating and Configuring a New Android Project 57
  Understanding the Android Symbolic View and the Traditional Project View 62
  Core Files and Directories of the Android Application 62
  Running Your Android Application in the Emulator 64
  Debugging Your Android Application in the Emulator 66
  Adding Logging Support to Your Android Application 69
  Debugging Your Application on Hardware 71

Summary 73

Quiz Questions 74
Exercises 74
References and More Information 74

II: Application Basics

4 Understanding Application Components 77

Mastering Important Android Terminology 77

The Application Context 78
  Retrieving the Application Context 78
  Using the Application Context 78

Performing Application Tasks with Activities 80
  The Lifecycle of an Android Activity 80

Organizing Activity Components with Fragments 85

Managing Activity Transitions with Intents 87
  Transitioning between Activities with Intents 88

Organizing Application Navigation with Activities, Fragments, and Intents 90
5 Defining the Manifest 95
Configuring Android Applications Using the Android Manifest File 95
   Editing the Android Manifest File 96
Managing Your Application’s Identity 99
   Setting the Application Name and Icon 99
Enforcing Application System Requirements 100
   Enforcing Application Platform Requirements 100
Other Application-Configuration Settings and Filters 102
Registering Activities in the Android Manifest 103
   Designating a Primary Entry-Point Activity for Your Application Using an Intent Filter 103
   Configuring Other Intent Filters 103
Registering Other Application Components 104
Working with Permissions 105
   Registering Permissions Your Application Requires 105
   Registering Permissions Your Application Enforces 108
Exploring Other Manifest File Settings 109
Summary 109
Quiz Questions 109
Exercises 110
References and More Information 110

6 Managing Application Resources 111
What Are Resources? 111
   Storing Application Resources 111
   Resource Value Types 112
   Accessing Resources Programmatically 116
Adding Simple Resource Values
in Android Studio 116
Working with Different Types of Resources 120
   Working with String Resources 120
   Using String Resources as Format Strings 121
   Working with Quantity Strings 123
   Working with String Arrays 123
   Working with Boolean Resources 124
   Working with Integer Resources 125
   Working with Colors 126
   Working with Dimensions 126
   Using Dimension Resources Programmatically 127
   Drawable Resources 128
   Working with Images 129
   Working with Color State Lists 131
   Working with Animation 133
   Working with Menus 135
   Working with XML Files 137
   Working with Raw Files 138
   References to Resources 138
Working with Layouts 140
   Designing Layouts in Android Studio 141
   Using Layout Resources Programmatically 144
   Referencing System Resources 144
   Summary 145
Quiz Questions 146
Exercises 146
References and More Information 146

7 Exploring Building Blocks 147
   Introducing Android Views and Layouts 147
      The Android View 147
      The Android Controls 147
      The Android Layout 148
   Displaying Text to Users with TextView 148
      Configuring Layout and Sizing 149
      Creating Contextual Links in Text 150
   Retrieving Data from Users with Text Fields 152
      Retrieving Text Input Using EditText Controls 152
Contents

Constraining User Input with Input Filters 154
Helping the User with Autocompletion 155
Giving Users Choices Using Spinner Controls 157
Allowing Simple User Selections with Buttons and Switches 159
Using Basic Buttons 159
Using CheckBox and ToggleButton Controls 161
Using RadioGroup and RadioButton 163
Retrieving Dates, Times, and Numbers from Users with Pickers 166
Using Indicators to Display Progress and Activity to Users 168
Indicating Progress with ProgressBar 168
Adding Progress Indicators to the ActionBar 170
Indicating Activity with Activity Bars and Activity Circles 171
Adjusting Progress with Seek Bars 171
Other Valuable User Interface Controls 173
Displaying Rating Data with RatingBar 173
Showing Time Passage with the Chronometer 174
Displaying the Time 175
Playing Video with VideoView 175
Summary 177
Quiz Questions 177
Exercises 177
References and More Information 178

8 Positioning with Layouts 179
Creating User Interfaces in Android 179
Creating Layouts Using XML Resources 179
Creating Layouts Programmatically 181
Organizing Your User Interface 184
Using ViewGroup Subclasses for Layout Design 185
Using ViewGroup Subclasses as View Containers 185
Using Built-in Layout Classes 185
  Using LinearLayout 187
  Using RelativeLayout 189
  Using FrameLayout 193
  Using TableLayout 196
  Using GridLayout 198
  Using Multiple Layouts on a Screen 202
Using Container Control Classes 203
  Using Data-Driven Containers 204
  Adding Scrolling Support 208
  Exploring Other View Containers 209
Summary 210
Quiz Questions 210
Exercises 211
References and More Information 211

9 Partitioning with Fragments 213
  Understanding Fragments 213
    Understanding the Fragment Lifecycle 214
    Managing Fragment Modifications 216
    Working with Special Types of Fragments 217
    Designing Fragment-Based Applications 218
Using the Android Support Library Package 228
  Adding Fragment Support to Legacy Applications 229
  Using Fragments in New Applications Targeting Older Platforms 229
  Linking the Android Support Package to Your Project 230
Additional Ways to Use Fragments 231
  Behavior Fragments without a User Interface 231
  Exploring Nested Fragments 232
Summary 232
Quiz Questions 232
Exercises 233
References and More Information 233
III: Application Design Essentials

10 Architecting with Patterns 237
   Architecting Your Application’s Navigation 237
   Android Application Navigation Scenarios 237
   Launching Tasks and Navigating the Back Stack 240
   Navigating with Fragments 241
   Relationships between Screens 242
   Android Navigation Design Patterns 243

Encouraging Action 251
   Menus 251
   Action Bars 251
   Floating Action Button 256
   Actions Originating from Your Application’s Content 257
   Dialogs 258

Summary 267
Quiz Questions 267
Exercises 267
References and More Information 268

11 Appealing with Style 269
   Styling with Support 269
   Themes and Styles 269
      Defining the Default Application Themes 270
      Theme and Style Inheritance 271
   Colors 272
   Layout 273
      Merge and Include 274
      TextInputLayout 275
      FloatingActionButton 275
      Toolbar as Bottom Bar 276
   Application Branding 278
   The Results Applied 280
   Typography 280
   Summary 282
   Quiz Questions 282
   Exercises 283
   References and More Information 283
12 Embracing Material Design 285
   Understanding Material 285
   The Default Material Theme 286
   The SampleMaterial Application 286
   Implementing the SampleMaterial Application 286
   Dependencies 286
   Material Support Styles 287
   Showing the Dataset in the List 288
   Summary 317
   Quiz Questions 317
   Exercises 317
   References and More Information 318

13 Designing Compatible Applications 319
   Maximizing Application Compatibility 319
   Designing User Interfaces for Compatibility 321
       Working with Fragments 323
       Leveraging the Various Android Support Library APIs 323
       Supporting Specific Screen Types 323
       Working with Nine-Patch Stretchable Graphics 324
   Providing Alternative Application Resources 324
       Understanding How Resources Are Resolved 325
       Organizing Alternative Resources with Qualifiers 326
       Providing Resources for Different Orientations 331
       Using Alternative Resources Programmatically 333
       Organizing Application Resources Efficiently 334
   Targeting Tablets and TVs 336
       Targeting Tablet Devices 336
       Targeting TV Devices 337
   Extending Your Application to Watches and Cars 338
   Ensuring Compatibility with SafetyNet 339
   Summary 340
   Quiz Questions 340
   Exercises 341
   References and More Information 341
IV: Application Development Essentials

14 Using Android Preferences  345
   Working with Application Preferences  345
      Determining When Preferences Are Appropriate  345
      Storing Different Types of Preference Values  346
      Creating Private Preferences for Use by a Single Activity  346
      Creating Shared Preferences for Use by Multiple Activities  346
      Searching and Reading Preferences  347
      Adding, Updating, and Deleting Preferences  348
      Reacting to Preference Changes  349
   Finding Preferences Data on the File System  349
   Creating Manageable User Preferences  350
      Creating a Preference Resource File  350
      Using the PreferenceActivity Class  353
      Organizing Preferences with Headers  354
   Auto Backup for Android Applications  359
   Summary  361
   Quiz Questions  361
   Exercises  361
   References and More Information  362

15 Accessing Files and Directories  363
   Working with Application Data on a Device  363
   Practicing Good File Management  364
   Understanding Android File Permissions  365
   Working with Files and Directories  366
      Exploring the Android Application Directories  366
      Working with Other Directories and Files on the Android File System  372
   Summary  375
   Quiz Questions  375
   Exercises  375
   References and More Information  376
16 Saving with SQLite 377

SampleMaterial Upgraded with SQLite 377

Working with Databases 377
  Providing Data Access 379
  Updating the SampleMaterialActivity Class 381
  Updating the SampleMaterialAdapter Constructor 382
  Database Operations Off the Main UI Thread 382
  Creating a Card in the Database 384
  Getting All Cards 385
  Adding a New Card 386
  Updating a Card 387
  Deleting a Card 388

Summary 390
Quiz Questions 390
Exercises 391
References and More Information 391

17 Leveraging Content Providers 393

Exploring Android’s Content Providers 393
  Using the MediaStore Content Provider 394
  Using the CallLog Content Provider 397
  Using the CalendarContract Content Provider 398
  Using the UserDictionary Content Provider 399
  Using the VoicemailContract Content Provider 399
  Using the Settings Content Provider 399
  Introducing the ContactsContract Content Providers 400

Modifying Content Provider Data 402
  Adding Records 402
  Updating Records 404
  Deleting Records 405

Using Third-Party Content Providers 406

Summary 406
V: Application Delivery Essentials

18 Learning the Development Workflow 411
   An Overview of the Android Development Process 411
   Choosing a Software Methodology 412
      Understanding the Dangers of Waterfall Approaches 412
      Understanding the Value of Iteration 413
   Gathering Application Requirements 413
      Determining Project Requirements 413
      Developing Use Cases for Android Applications 416
      Incorporating Third-Party Requirements and Recommendations 417
   Managing a Device Database 417
   Assessing Project Risks 421
      Identifying Target Devices 421
      Acquiring Target Devices 422
      Determining the Feasibility of Application Requirements 423
      Understanding Quality Assurance Risks 423
   Writing Essential Project Documentation 425
      Developing Test Plans for Quality Assurance Purposes 425
      Providing Documentation Required by Third Parties 426
      Providing Documentation for Maintenance and Porting 426
   Leveraging Configuration Management Systems 426
      Choosing a Source Control System 426
      Implementing an Application Version System That Works 427
   Designing Android Applications 427
      Understanding Android Device Limitations 428
Exploring Common Android Application Architectures 428
Designing for Extensibility and Maintenance 428
Designing for Application Interoperability 430
Developing Android Applications 430
Testing Android Applications 431
Controlling the Test Release 431
Deploying Android Applications 432
Determining Target Markets 432
Supporting and Maintaining Android Applications 433
Track and Address Crashes Reported by Users 433
Testing Firmware Upgrades 433
Maintaining Adequate Application Documentation 433
Managing Live Server Changes 434
Identifying Low-Risk Porting Opportunities 434
Application Feature Selection 434
Summary 434
Quiz Questions 435
Exercises 435
References and More Information 435

19 Planning the Experience 437
Thinking about Objectives 437
User Objectives 438
Team Objectives 438
Objectives of Other Stakeholders 438
Techniques for Focusing Your Product Efforts 439
Personas 439
User Story Mapping 440
Entity Discovery and Organization 440
Planning User Interactions 441
Communicating Your Application’s Identity 442
Designing Screen Layouts 443
Sketches 443
Wireframes 443
Design Comps 444
Reacting Properly with Visual Feedback 444
Observing Target Users for Usability 445
  Mocking Up the Application 445
  Testing the Release Build 446
Summary 446
Quiz Questions 446
Exercises 447
References and More Information 447

20 Delivering Quality Applications 449
Best Practices in Delivering Quality Applications 449
  Meeting Android Users’ Demands 450
  Designing User Interfaces for Android Devices 450
  Designing Stable and Responsive Android Applications 451
  Designing Secure Android Applications 453
  Designing Android Applications for Maximum Profit 453
  Following the Android Application Quality Guidelines 454
  Leveraging Third-Party Quality Standards 456
  Designing Android Applications for Ease of Maintenance and Upgrades 456
  Leveraging Android Tools for Application Design 458
Avoiding Silly Mistakes in Android Application Design 459
Best Practices in Delivering Quality Android Applications 459
  Designing a Development Process That Works for Android Development 460
  Testing the Feasibility of Your Application Early and Often 460
  Using Coding Standards, Reviews, and Unit Tests to Improve Code Quality 461
  Handling Defects Occurring on a Single Device 463
  Leveraging Android Tools for Development 464
  Avoiding Silly Mistakes in Android Application Development 464
Summary 464
Quiz Questions 465
Contents

Exercises 465
References and More Information 465

21 Testing Your Applications 467
   Best Practices in Testing Mobile Applications 467
      Designing a Mobile Application Defect-Tracking System 467
      Managing the Testing Environment 469
      Maximizing Testing Coverage 471
      Leveraging Android SDK Tools for Android Application Testing 477
      Avoiding Silly Mistakes in Android Application Testing 479
   Android Application Testing Essentials 479
      Unit Testing with JUnit 480
      Introducing the PasswordMatcher Application 481
      Determining What Our Tests Should Prove 485
      Creating a Run Configuration for Test Code 485
      Writing the Tests 489
      Running Your First Test Using Android Studio 491
      Analyzing the Test Results 492
      Adding Additional Tests 493
   More Android Automated Testing Programs and APIs 496
   Summary 497
   Quiz Questions 497
   Exercises 498
   References and More Information 498

22 Distributing Your Applications 499
   Choosing the Right Distribution Model 499
      Protecting Your Intellectual Property 500
      Following the Policies of Google Play 501
      Billing the User 501
   Packaging Your Application for Publication 502
      Preparing Your Code for Packaging 503
      Packing and Signing Your Application 504
Contents

Testing the Release Version of Your Application Package 508
Including All Required Resources 508
Readying Your Servers or Services 508
Distributing Your Application 508

Publishing to Google Play 509
Signing Up for Publishing to Google Play 509
Uploading Your Application to Google Play 511
Uploading Application Marketing Assets 514
Configuring Pricing and Distribution Details 515
Configuring Additional Application Options 516
Managing Other Developer Console Options 516
Publishing Your Application to Google Play 516
Managing Your Application on Google Play 517

Google Play Staged Rollouts 518
Publishing to the Google Play Private Channel 518

Translating Your Application 519
Publishing Using Other Alternatives 520
Self-Publishing Your Application 520

Summary 522
Quiz Questions 522
Exercises 522
References and More Information 523

VI: Appendixes

A Tips and Tricks: Android Studio 527
Organizing Your Android Studio Workspace 527
Integrating with Source Control Services 527
Repositioning Windows within Android Studio 528
Resizing the Editor Window 528
Resizing Tool Windows 528
Viewing Editor Windows Side by Side 529
Viewing Two Sections of the Same File 529
Closing Unwanted Tabs 531
Keeping Editor Windows under Control 531
Creating Custom Log Filters 532
Acknowledgments

This book is the result of collaboration among the finest group of professionals: from the efforts of the team at Pearson Education (Addison-Wesley); 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 Android Open Source Project for their vision and expertise. Special thanks go to Mark Taub for believing in the vision for extending this book to another edition; Laura Lewin, who was the driving force behind the book—without her this book would not have become a reality; Olivia Basegio, who was instrumental in orchestrating all of the efforts among everyone involved; and Songlin Qiu for performing countless iterations combing through the manuscript to make this book ready for production. And to the technical reviewers: Ray Rischpater, who surprised us yet again with quality recommendations; Doug Jones, whose expertise uncovered needed improvements to the fine details; and Valerie Shipbaugh, who was able to provide tips on desperately needed clarification (as well as Mike Wallace, Mark Gjoel, Dan Galpin, Tony Hilleron, Ronan Schwarz, and Charles Stearns, who reviewed previous editions). For previous editions, Dan Galpin 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.
About the Authors

Joseph Annuzzi, Jr. is a code warrior, graphic artist, entrepreneur, and author. 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 APIs; tinkering with peer-to-peer, cryptography, and biometric algorithms; or creating stunningly realistic 3D renders. He is always on the lookout for disruptive Internet and mobile technologies. 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 and iOS 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 pint-sized geekling daughter. She 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, has gotten stuck in a ravine with two hungry lions in Kenya, has 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 has gotten her tongue stuck to an iceberg in Iceland (while being watched by a herd of suspicious wild reindeer). Most recently, she can be found hiking along the Appalachian Trail with her daughter and documenting the journey with Google Glass.

Shane Conder has extensive application development experience and has focused his attention on mobile and embedded development for well over a 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 tech industry and is known for his keen insights regarding mobile development platform trends. Shane received a BS in computer science from the University of California, Santa Cruz.

A self-admitted gadget freak, Shane always has the latest smartphone, tablet, or wearable. He 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 three devices with him when backpacking (“just in case”)—even where there is no coverage. Lately, his smartwatch collection has exceeded his number of wrists. Luckily, his young daughter is happy to offer her own. Such are the burdens of a daughter of engineers.
Android is a popular, free, open-source mobile platform that has taken the 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, the 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.
Introduction

- **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? How should developers take advantage of these differences?
3. How do developers use Android Studio and the Android SDK tools 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. What is material design and why does it matter?
8. How does the mobile development process differ from traditional desktop development?
9. What strategies work best for Android development?
10. What do managers, developers, and testers need to look for when planning, developing, and testing a mobile application?
11. How do mobile teams deliver quality Android applications for publishing?
12. How do mobile teams package Android applications for distribution?
13. How do mobile teams make money from Android applications?
14. And, finally, what is new in this edition of the book?

**How This Book Is Structured**

*Introduction to Android Application Development, Fifth 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: Platform Overview**
  Part I provides an introduction to Android, explaining how it differs from other mobile platforms. You become familiar with the Android SDK 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: Application Basics**
  Part II introduces the 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. 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 III: Application Design Essentials**
  Part III dives deeper into how applications are designed in Android. You learn about material design, styling, and common design patterns found among applications. You also learn how to design and plan your applications. This section is of primary interest to developers.

- **Part IV: Application Development Essentials**
  Part IV covers the features used by most Android applications, including storing persistent application data using preferences, working with files and directories, SQLite, and content providers. This section is of primary interest to developers.

- **Part V: Application Delivery Essentials**
  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 tips and tricks for Android Studio, an overview of the Android SDK tools, three helpful quick-start guides for the Android development tools—the emulator, Device Monitor, and Gradle—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 millions of Android devices (with thousands of different device models) shipping all over the world every quarter—phones, tablets, e-book readers, smartwatches, and specialty devices such as gaming consoles, TVs, 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. But don’t worry, it’s still the book readers loved the first, second, third, and fourth 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 included quiz questions to help readers ensure they understand each chapter’s content, and 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:

- The entire book has been overhauled to include coverage of the Android Studio IDE. Previous editions of this book included coverage of the Eclipse IDE. Where applicable, all content, images, and code samples have been updated for Android Studio. In addition, coverage of the latest and greatest Android tools and utilities is included.
- The chapter on defining the manifest includes coverage of the new Android 6.0 Marshmallow (API Level 23) permission model, and it provides a code sample demonstrating the new permission model.
- A brand new chapter on material design has been added and demonstrates how developers can integrate common material design features into their application, and it includes a code sample.
- A brand new chapter on working with styles has been included with tips on how to best organize styles and reuse common UI components for optimized display rendering, and it provides a code sample.
- A brand new chapter on common design patterns has been added with details on various ways to architect your application, and it offers a code sample.
- A brand new chapter on incorporating SQLite for working with persistent database-backed application data has been added, and it includes a code sample.
- An appendix providing tips and tricks for using Android Studio has been included.
- An appendix on the Gradle build system has been included to help you understand what Gradle is and why it’s important.
- The AdvancedLayouts code sample has been updated so that the GridView and ListView components make use of Fragment and ListFragment classes respectively.
- Some code samples include an ActionBar by making use of the new Toolbar, and have done so using the support library for maintaining compatibility on devices running older APIs. When necessary, application manifests have been updated to support parent-child Activity relationships that support up-navigation.
- Many code samples make use of the AppCompatActivity class and the appcompat-v7 support library.
- All chapters and appendixes 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, 8, and Mac OS X 10.9
- Android Studio 1.3.2
- Android SDK API Level 23 (referred to in this book as Android Marshmallow)
- Android SDK Tools 24.3.4
- Android SDK Platform Tools 23.0.0
- Android SDK Build Tools 23.0.0
- Android Support Repository 17 (where applicable)
- Java SE Development Kit (JDK) 7 Update 55
- Android devices: Nexus 4, 5, and 6 (phones), Nexus 7 (first- and second-generation 7-inch tablet), Nexus 9 and 10 (large tablet), including various other popular devices and form factors.

The Android platform continues to grow aggressively in market share against competing mobile platforms, such as Apple iOS, Windows Phone, and BlackBerry OS. 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 Marshmallow, brings many new features. This book covers 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 as many popular devices as possible on the market today. As of the writing of this book, approximately 9.7% of users’ devices are running a version of Android Lollipop, 5.0 or 5.1, and Android Marshmallow has yet to be released on real devices. Of course, some devices will receive upgrades, and users will purchase new Lollipop and Marshmallow 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 for This Book**

The source code that accompanies this book is available for download from our book’s website: [http://introductiontoandroid.blogspot.com/2015/08/5th-edition-book-code-samples.html](http://introductiontoandroid.blogspot.com/2015/08/5th-edition-book-code-samples.html). The code samples are organized by chapter and downloadable in zip format or accessible from the command line with Git. You’ll also find other Android topics discussed on our book’s website ([http://introductiontoandroid.blogspot.com](http://introductiontoandroid.blogspot.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 to 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.

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 mobile industry:

- Android Developer website: the Android SDK and developer reference site:
- Google Plus: Android Developers Group:
  https://plus.google.com/+AndroidDevelopers/posts
- YouTube: Android Developers and Google Design:
  https://www.youtube.com/user/androiddevelopers
  https://www.youtube.com/channel/UCiKO7be7O9eUGL94PHnAeOA
- Google Material Design:
  https://www.google.com/design/spec/material-design/introduction.html
- 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
- Android Open Source Project:
  https://source.android.com/index.html
- Open Handset Alliance: Android manufacturers, operators, and developers:
  http://openhandsetalliance.com
- Google Play: buy and sell Android applications:
  https://play.google.com/store
- tuts+: Android development tutorials:
  http://code.tutsplus.com/categories/android
- Google Sample Apps: open-source Android applications hosted on GitHub:
  https://github.com/googlesamples
- 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://fiercedeveloper.com
- XDA-Developers Android Forum:
  http://forum.xda-developers.com/android
- Developer.com: a developer-oriented site with mobile articles:
  http://developer.com
Contacting the Authors

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

- [http://introductiontoandroid.blogspot.com](http://introductiontoandroid.blogspot.com)

Or email us at:

- introtoandroid5e@gmail.com

Find Joseph Annuzzi on LinkedIn:

- Joseph Annuzzi, Jr.: [https://www.linkedin.com/in/josephannuzzi](https://www.linkedin.com/in/josephannuzzi)

Circle Joseph Annuzzi on Google+:

There are many different ways for storing your Android applications’ data. As you learned in Chapter 14, “Using Android Preferences,” and in Chapter 15, “Accessing Files and Directories,” there is definitely more than one way for accessing and storing your data. But what if you need to store structured data for your application, such as data more suited for storing in a database? That’s where SQLite comes in. In this chapter, we are going to be modifying the SampleMaterial application found in Chapter 12, “Embracing Material Design,” so that card data is stored persistently in a SQLite database on the device and will survive various lifecycle events. By the end of this chapter, you will be confident in adding a SQLite database for your application.

**SampleMaterial Upgraded with SQLite**

The SampleMaterial application found in Chapter 12, “Embracing Material Design,” shows you how to work with data in the application but fails when it comes to storing the data permanently so that it survives Android lifecycle events. When adding, updating, and deleting cards from the SampleMaterial application, and then clearing the SampleMaterial application from the Recent apps, the application is not able to remember what cards were added, updated, and deleted. So we updated the application to store the information in a SQLite database to keep track of the data permanently. Figure 16.1 shows the SampleSQLite application, which looks the same as the SampleMaterial application, but is backed by a SQLite database.

**Working with Databases**

The first thing that must be done is to define the database table that should be created for storing the cards in the database. Luckily, Android provides a helper class for defining a SQLite database table through Java code. That class is called SQLiteOpenHelper. You need to create a Java class that extends from the SQLiteOpenHelper, and this is where you can define a database name and version, and where you define the tables and columns. This is also where you create and upgrade your database. For the SampleSQLite application,
we created a CardsDBHelper class that extends from SQLiteOpenHelper, and here’s the implementation that can be found in the CardsDBHelper.java file:

```java
public class CardsDBHelper extends SQLiteOpenHelper {
    private static final String DB_NAME = "cards.db";
    private static final int DB_VERSION = 1;

    public static final String TABLE_CARDS = "CARDS";
    public static final String COLUMN_ID = "_ID";
    public static final String COLUMN_NAME = "NAME";
    public static final String COLUMN_COLOR_RESOURCE = "COLOR_RESOURCE";

    private static final String TABLE_CREATE = 
        "CREATE TABLE " + TABLE_CARDS + " (" + 
```
This class starts off by defining a few static final variables for providing a name and version number, and an appropriate table name with table column names. Further, the TABLE_CREATE variable provides the SQL statement for creating the table in the database. The CardsDBHelper constructor accepts a context and this is where the database name and version are set. The onCreate() and onUpgrade() methods either create the new table or delete an existing table, and then create a new table.

You should also notice that the table provides one column for the _ID as an INTEGER, one column for the NAME as TEXT, and one column for the COLOR_RESOURCE as an INTEGER.

**Note**
The SQLiteOpenHelper class assumes version numbers will be increasing for an upgrade. That means if you are at version 1, and want to update your database, set the version number to 2 and increase the version number incrementally for additional versions.

**Provisioning Data Access**
Now that you are able to create a database, you need a way to access the database. To do so, you will create a class that provides access to the database from the SQLiteDatabase.
class using the SQLiteOpenHelper class. This class is where we will be defining the
methods for adding, updating, deleting, and querying the database. The class for doing
this is provided in the CardsData.java file and a partial implementation can be
found here:

class CardsData {
    public static final String DEBUG_TAG = "CardsData";

    private SQLiteDatabase db;
    private SQLiteOpenHelper cardDbHelper;

    private static final String[] ALL_COLUMNS = {
        CardsDBHelper.COLUMN_ID,
        CardsDBHelper.COLUMN_NAME,
        CardsDBHelper.COLUMN_COLOR_RESOURCE
    };

    public CardsData(Context context) {
        this.cardDbHelper = new CardsDBHelper(context);
    }

    public void open() {
        db = cardDbHelper.getWritableDatabase();
    }

    public void close() {
        if (cardDbHelper != null) {
            cardDbHelper.close();
        }
    }
}

Notice the CardsData() constructor. This creates a new CardsDBHelper() object
that will allow us to access the database. The open() method is where the database is
created with the getWritableDatabase() method. The close() method is for closing
the database. It is important to close the database to release any resources obtained by
the object so that unexpected errors do not occur in your application during use. You
also want to open and close the database during your application’s particular lifecycle
events so that you are only executing database operations at the times when you have the
appropriate access.
Updating the SampleMaterialActivity Class

The onCreate() method of the SampleMaterialActivity now creates a new data access object and opens the database. Here is the updated onCreate() method:

```java
public CardsData cardsData = new CardsData(this);

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sample_material);

    names = getResources().getStringArray(R.array.names_array);
    colors = getResources().getIntArray(R.array.initial_colors);

    recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    new GetOrCreateCardsListTask().execute();

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Pair<View, String> pair = Pair.create(v.findViewById(R.id.fab),
                    TRANSITION_FAB);

            ActivityOptionsCompat options;
            Activity act = SampleMaterialActivity.this;
            options = ActivityOptionsCompat.makeSceneTransitionAnimation(act, pair);

            Intent transitionIntent = new Intent(act, TransitionAddActivity.class);
            act.startActivityForResult(transitionIntent, adapter.getItemCount(),
                    options.toBundle());
        }
    });
```
Notice the new `GetOrCreateCardsListTask().execute()` method call. We cover this implementation later in this chapter. This method queries the database for all cards or fills the database with cards if it is empty.

**Updating the `SampleMaterialAdapter` Constructor**

An update in the `SampleMaterialAdapter` class is also needed, and the constructor is shown below:

```java
public CardsData cardsData;
public SampleMaterialAdapter(Context context, ArrayList<Card> cardsList,
        CardsData cardsData) {
    this.context = context;
    this.cardsList = cardsList;
    this.cardsData = cardsData;
}
```

Notice a `CardsData` object is passed into the constructor to ensure the database is available to the `SampleMaterialAdapter` object once it is created.

**Warning**

Because database operations block the UI thread of your Android application, you should always run database operations in a background thread.

**Database Operations Off the Main UI Thread**

To make sure that the main UI thread of your Android application does not block during a potentially long-running database operation, you should run your database operations in a background thread. Here, we have implemented an `AsyncTask` for creating new cards in the database, and will subsequently update the UI only after the database operation is complete. Here is the `GetOrCreateCardsListTask` class that extends the `AsyncTask` class, which either retrieves all the cards from the database or creates them:

```java
public class GetOrCreateCardsListTask extends AsyncTask<Void, Void,
        ArrayList<Card>> {
    @Override
    protected ArrayList<Card> doInBackground(Void... params) {
        cardsData.open();
        cardsList = cardsData.getAll();
        if (cardsList.size() == 0) {
            for (int i = 0; i < 50; i++) {
```
Card card = new Card();
card.setName(names[i]);
card.setColorResource(colors[i]);
cardsList.add(card);
cardsData.create(card);
Log.d(DEBUG_TAG, "Card created with id " + card.getId() + ", name " + card.getName() + ", color " + card.getColorResource());
}
}
return cardsList;
}

@Override
protected void onPostExecute(ArrayList<Card> cards) {
    super.onPostExecute(cards);
    adapter = new SampleMaterialAdapter(SampleMaterialActivity.this,
cardsList, cardsData);
    recyclerView.setAdapter(adapter);
}

When this class is created and executed in the onCreate() method of the Activity, it overrides the doInBackground() method and creates a background task for retrieving all the cards from the database with the call to getAll(). If no items are returned, that means the database is empty and needs to be populated with entries. The for loop creates 50 Cards and each Card is added to the cardsList, and then created in the database with the call to create(). Once the background operation is complete, the onPostExecute() method, which was also overridden from the AsyncTask class, receives the cardsList result from the doInBackground() operation. It then uses the cardsList and cardsData to create a new SampleMaterialAdapter, and then adds that adapter to the recyclerView to update the UI once the entire background operation has completed.

Notice the AsyncTask class has three types defined; the first is of type Void, the second is also Void, and the third is ArrayList<Card>. These map to the Params, Progress, and Result generic types of an AsyncTask. The first Params is used as the parameter of the doInBackground() method, which are Void, and the third Result generic is used as the parameter of the onPostExecute() method. In this case, the second Void generic was not used, but would be used as the parameter for the onProgressUpdate() method of an AsyncTask.
Note

Note that you are not able to call UI operations on the doInBackground() method of an AsyncTask. Those operations need to be performed before or after the doInBackground() method, but if you need the UI to update only after the background operation has completed, you must perform those operations in the onPostExecute() method so the UI is updated appropriately.

Creating a Card in the Database

The magic happens in the call to cardsData.create(). This is where the Card is inserted into the database. Here is the create() method definition found in the CardsData class:

```java
public Card create(Card card) {
    ContentValues values = new ContentValues();
    values.put(CardsDBHelper.COLUMN_NAME, card.getName());
    values.put(CardsDBHelper.COLUMN_COLOR_RESOURCE, card.getColorResource());
    long id = db.insert(CardsDBHelper.TABLE_CARDS, null, values);
    card.setId(id);
    Log.d(DEBUG_TAG, "Insert id is " + String.valueOf(card.getId()));
    return card;
}
```

The create() method accepts a Card data object. A ContentValues object is created to temporarily store the data that will be inserted into the database in a structured format. There are two value.put() calls that map the database column to a Card attribute. The insert() method is then called on the cards table and the temporary values are passed in for insertion. An id is returned from the call to insert() and that value is then set as the id for the Card, and finally a Card object is returned. Figure 16.2 shows the logcat output of cards being inserted into the database.

![Figure 16.2 logcat output showing items inserted into the database.](image-url)
Getting All Cards

Earlier, we mentioned the `getAll()` method that queries the database for all the cards in the cards table. Here is the implementation of the `getAll()` method:

```java
public ArrayList<Card> getAll() {
    ArrayList<Card> cards = new ArrayList<>();
    Cursor cursor = null;
    try {
        cursor = db.query(CardsDBHelper.TABLE_CARDS,
                           COLUMNS, null, null, null, null, null);
        if (cursor.getCount() > 0) {
            while (cursor.moveToNext()) {
                Card card = new Card();
                card.setId(cursor.getLong(cursor
                                .getColumnIndex(CardsDBHelper.COLUMN_ID)));
                card.setName(cursor.getString(cursor
                                .getColumnIndex(CardsDBHelper.COLUMN_NAME)));
                card.setColorResource(cursor
                                .getInt(cursor.getColumnIndex(CardsDBHelper
                                .COLUMN_COLOR_RESOURCE)));
                cards.add(card);
            }
        }
    } catch (Exception e) {
        Log.d(DEBUG_TAG, "Exception raised with a value of " + e);
    } finally {
        if (cursor != null) {
            cursor.close();
        }
    }
    return cards;
}
```

A query is performed on the cards table inside a `try` statement with a call to `query()` that returns all columns for the query as a `Cursor` object. A `Cursor` allows you to access the results of the database query. First, we ensure that the `Cursor` count is greater than
zero, otherwise no results will be returned from the query. Next, we iterate through all the
cursor objects by calling the `moveToNext()` method on the cursor, and for each database
item, we create a `Card` data object from the data in the `Cursor` and set the `Cursor` data to
`Card` data. We also handle any exceptions that we may have encountered, and finally the
`Cursor` object is closed and all cards are returned.

**Adding a New Card**

You already know how to insert cards into the database because we did that to initialize
the database. So adding a new `Card` is very similar to how we initialized the database. The
`addCard()` method of the `SampleMaterialAdapter` class needs a slight modification. This
method executes `AsyncTask` to add a new card in the background. Here is the updated
implementation of the `addCard()` method creating a `CreateCardTask` and executing
the task:

```java
public void addCard(String name, int color) {
    Card card = new Card();
    card.setName(name);
    card.setColorResource(color);
    new CreateCardTask().execute(card);
}

private class CreateCardTask extends AsyncTask<Card, Void, Card> {
    @Override
    protected Card doInBackground(Card... cards) {
        cardsData.create(cards[0]);
        cardsList.add(cards[0]);
        return cards[0];
    }

    @Override
    protected void onPostExecute(Card card) {
        super.onPostExecute(card);
        ((SampleMaterialActivity) context).doSmoothScroll(getItemCount() - 1);
        notifyItemInserted(getItemCount());
        Log.d(DEBUG_TAG, "Card created with id " + card.getId() + ", name " + card.getName() + ", color " + card.getColorResource());
    }
}
```
The `doInBackground()` method makes a call to the `create()` method of the `cardsData` object, and in the `onPostExecute()` method, a call to the `doSmoothScroll()` method of the calling `Activity` is made, then the adapter is notified that a new Card has been inserted.

### Updating a Card

To update a Card, we first need a way to keep track of the position of a Card within the list. This is not the same as the database id because the id of the item in the database is not the same as the position of the item in the list. The database increments the id of a Card, so each new Card has an id one higher than the previous Card. The RecyclerView list, on the other hand, shifts positions as items are added and removed from the list.

First, let's update the `Card` data object found in the `Card.java` file and add a new `listPosition` attribute with the appropriate getter and setter methods as shown here:

```java
private int listPosition = 0;

public int getListPosition() {
    return listPosition;
}

public void setListPosition(int listPosition) {
    this.listPosition = listPosition;
}
```

Next, update the `updateCard()` method of the `SampleMaterialAdapter` class and implement an `UpdateCardTask` class that extends `AsyncTask` as follows:

```java
public void updateCard(String name, int list_position) {
    Card card = new Card();
    card.setName(name);
    card.setId(getItemId(list_position));
    card.setListPosition(list_position);
    new UpdateCardTask().execute(card);
}

private class UpdateCardTask extends AsyncTask<Card, Void, Card> {

    @Override
    protected Card doInBackground(Card... cards) {
        cardsData.update(cards[0].getId(), cards[0].getName());
        (Continues)
```
cardsList.get(cards[0].getListPosition()).setName(cards[0].getName());
return cards[0];
}

@Override
protected void onPostExecute(Card card) {
    super.onPostExecute(card);
    Log.d(DEBUG_TAG, "list_position is " + card.getListPosition());
    notifyItemChanged(card.getListPosition());
}

The UpdateCardTask calls the update() method of the cardsData object in the doInBackground() method and then updates the name of the corresponding Card in the cardsList object and returns the Card. The onPostExecute() method then notifies the adapter that the item has changed with the notifyItemChanged() method call.

Finally, the CardsData class needs to implement the update() method to update the particular Card in the database. Here is the update() method:

public void update(long id, String name) {
    String whereClause = CardsDBHelper.COLUMN_ID + "=\" + id;
    Log.d(DEBUG_TAG, "Update id is \" + String.valueOf(id));
    ContentValues values = new ContentValues();
    values.put(CardsDBHelper.COLUMN_NAME, name);
    db.update(CardsDBHelper.TABLE_CARDS, values, whereClause, null);
}

The update() method accepts id and name parameters. A whereClause is then constructed for matching the id of the Card with the appropriate id column in the database, and a new ContentValues object is created for adding the updated name for the particular Card to the appropriate name column. Finally, the update() method is executed on the database.

Deleting a Card

Now let's take a look at how to modify the deletion of cards. Remember the animateCircularDelete() method—this is where a Card was animated off the screen and deleted from the cardsList object. In the onAnimationEnd() method, construct a Card data object and pass that to the execute method of a DeleteCardTask object, which is an AsyncTask. Here are those implementations:

public void animateCircularDelete(final View view, final int list_position) {
    int centerX = view.getWidth();
int centerY = view.getHeight();
int startRadius = view.getWidth();
int endRadius = 0;
Animator animation = ViewAnimationUtils.createCircularReveal(view,
        centerX, centerY, startRadius, endRadius);

animation.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        super.onAnimationEnd(animation);
        view.setVisibility(View.INVISIBLE);
        Card card = new Card();
        card.setId(getItemId(list_position));
        card.setListPosition(list_position);
        new DeleteCardTask().execute(card);
    }
});
animation.start();

private class DeleteCardTask extends AsyncTask<Card, Void, Card> {
    @Override
    protected Card doInBackground(Card... cards) {
        cardsData.delete(cards[0].getId());
        cardsList.remove(cards[0].getListPosition());
        return cards[0];
    }

    @Override
    protected void onPostExecute(Card card) {
        super.onPostExecute(card);
        notifyItemRemoved(card.getListPosition());
    }
}
The doInBackground() method of the DeleteCardTask calls the delete() method of the cardsData object and passes in the id of Card. Then the Card is removed from the cardsList object, and in the onPostExecute() method, the adapter is notified that an item has been removed by calling the notifyItemRemoved() method and passing in the list position of the Card that has been removed.

There is one last method to implement—the delete() method of the CardsData class. Here is that method:

```java
public void delete(long cardId) {
    String whereClause = CardsDBHelper.COLUMN_ID + "= " + cardId;
    Log.d(DEBUG_TAG, "Delete position is " + String.valueOf(cardId));
    db.delete(CardsDBHelper.TABLE_CARDS, whereClause, null);
}
```

The delete() method of the CardsData class accepts an id of a Card, constructs a whereClause using that id, and then calls the delete() method on the cards table of the database, passing in the appropriate whereClause with the id of the Card to delete.

**Summary**

You now have a full implementation of a database that provides permanent storage for your application. In this chapter, you learned how to create a database. You also learned how to access the database for querying, inserting, updating, and deleting items from it. In addition, you also learned how to update the SampleMaterial application so that Card data is stored in a database. Finally, you learned how to perform your database operations off of the main UI thread by performing the operations in the background with an AsyncTask so as not to block the UI when running these blocking operations. You should now be ready to implement simple SQLite databases in your own applications.

**Quiz Questions**

1. What is the SQLiteDatabase method for creating a table?
2. What method provides access for reading and writing a database?
3. True or false: The async() method of an AsyncTask allows you to execute long-running operations off the main UI thread in the background.
4. True or false: The onAfterAsync() method of an AsyncTask allows you to execute UI methods after an AsyncTask completes.
Exercises

1. Read through the “Saving Data in SQL Databases” training in the Android documentation found here: http://d.android.com/training/basics/data-storage/databases.html.

2. Read through the “SQLiteDatabase” SDK reference to learn more about how to utilize a SQLite database here: http://d.android.com/reference/android/database/sqlite/SQLiteDatabase.html.

3. Modify the SampleSQLite application to support the deletion of all items from the database with a single database operation.

References and More Information

Android Tools: “sqlite3”:

SQLite:
 http://www.sqlite.org/

Command Line Shell For SQLite:
 http://www.sqlite.org/cli.html

Android API Guides: “Content Providers”:
 http://d.android.com/guide/topics/providers/content-providers.html

Android SDK Reference regarding the application android.database.sqlite package:

Android SDK Reference regarding the application AsyncTask class:

Android SDK Reference regarding the application ContentValues class:
 http://d.android.com/reference/android/content/ContentValues.html

Android SDK Reference regarding the application SQLiteDatabase class:
 http://d.android.com/reference/android/database/sqlite/SQLiteDatabase.html

Android SDK Reference regarding the application SQLiteOpenHelper class:

Android SDK Reference regarding the application Cursor class:
This page intentionally left blank
Index

A

AbsoluteLayout, avoiding use of, 322

Action buttons
    adding to ActionBar, 252–253
    floating action buttons, 256–257, 275–276

ActionBar class
    action overflow, 253–255
    adding action buttons, 252–253
    adding progress indicators, 170–171
    compatibility and, 255
    floating action buttons, 256–257, 275–276
    organizing application navigation, 90
    overview of, 251–252
    toolbars as action bars, 255–256

Actions
    action bar compatibility, 255
    action bars, 251–252
    action buttons, 252–253
    action overflow, 253–255
    adding primary action to applications, 297–301
    floating action buttons, 256–257, 275–276
    menus, 251
    originating from application content areas, 257–258
    overview of, 251
    Toolbar as action bars, 255–256
<activity-alias> tag, 109
Activity bars, 171
Activity circles, 171
Activity class
  adding to new project, 59–61
  attaching/detaching fragments, 216–217
  customizing back stack, 241
  defined, 77
  defining for hosting Fragment components, 227–228
  dialogs for organizing information and reacting to user events, 260
  displaying using indicator controls, 168–169
  in Fragment lifecycle, 214–215
  fragments and, 213
  handling configuration changes, 335
  instantiating LinearLayout programmatically, 181–182
  launching activity belonging to another application, 89
  launching by class name, 88
  managing transitions with intents, 87–89
  organizing application navigation, 90
  organizing with fragments, 85–87
  performing application tasks, 80–85
  private preference for use by single activity, 346
  registering in manifest file, 103–105
  shared preference for use by multiple activities, 346–347
<activity> tag, registering activities in manifest file, 103
Ad revenues, in distribution of applications, 502
Adapter class, binding data set to RecyclerView, 293–296

AdapterView controls
  ArrayAdapter control, 204–205
  CursorAdapter control, 205–206
  data-driven containers, 204
  ListView control, 206–208
  ScrollView control, 208–209
ADB (Android Debug Bridge), 591
addCard(), adding cards to database, 386–387
Adobe AIR, support for Android platform, 24
ADT (Android Developer Tools) plugin, 18
Ahead-of-time compilation (AOT), 21
AlertDialog, 257
Aliases, activity, 109
Allocation Tracker, in Device Monitor, 573–574
Alternative resources. See also Resources
  creating, 325–326
  maximizing application compatibility, 321
  organizing, 334–335
  organizing with qualifiers, 326–331
  providing for different screen orientations, 331–333
  reasons for including, 324–325
  targeting tablet devices, 336
  using, 115–116
Amazon Appstore, 20, 456
Amazon Fire OS, 26–27
AnalogClock control, 175
Analyzing test results, 492–493
Ancestral navigation, 239–241
Android application development
  high-quality. See Quality Android applications
  standard for development and design. See Material design
  workflow in. See Development workflow
Android application package (APK)
- multiples in Google Play, 414–415
- variants in Gradle, 616–618
Android Auto, extending applications to cars, 339
Android Backup Service, 458
Android Debug Bridge (ADB), 591
Android Developer Tools (ADT) plugin, 18
Android Developers Blog, 452
Android development process.
  See Development workflow
Android emulator. See Emulator
Android manifest for filtering, in Packaging applications, 503–504
Android Marshmallow
  - auto backup feature introduced in, 99
  - permission levels, 105–106
  - permissions for fingerprint authentication, 108
Android Open Source Project (AOSP)
  - Compatibility Test Suite, 339
  - Google initiative making Android source code available, 11
Android OS, introduction to
  - AOSP (Android Open Source Project), 11
  - applications, 23–26
  - architecture underlying, 21–22
  - current focus, 15–16
  - custom forks, 26–28
  - Google’s mobile first philosophy, 12
  - Open Handset Alliance, 12–15
  - reasonable learning curve for developers, 18
  - security and permissions, 22–23
  - summary, Q&A, and references, 28–29
  - uniqueness of Android platform, 16–21
Android project view, comparing with traditional Project view, 62
Android robot, 16–17
Android Runtime (ART), 21, 106–108
Android SDK
  - accessing documentation, 40
  - Android Debug Bridge and, 591
  - code diagnostics in, 462
  - configuring development environment, 31
  - design tools in, 458–459
  - development tools in, 464
  - Device Monitor and, 567, 590–591
  - documentation for, 585–588
  - downloading and installing, 32–33
  - exploring Android SDK manager, 43–45
  - exploring core features, 40–42
  - freely available, 18
  - Hierarchy Viewer, 591–595
  - introduction to, 585
  - layout editor, 591–592
  - leveraging Android emulators with, 588–589
  - license agreement in, 38–39, 449
  - logcat for viewing log data, 589–590
  - Nine-Patch Stretchable Graphics, 596–598
  - other tools in, 597–600
  - preference system, 345
  - reporting bugs, 37–38
  - summary, Q&A, and references, 600–601
  - testing essentials, 477–480, 496
  - unit tests in, 462–463
  - upgrading, 37
Android application package (APK)
  - multiples in Google Play, 414–415
  - variants in Gradle, 616–618
Android Auto, extending applications to cars, 339
Android Backup Service, 458
Android Debug Bridge (ADB), 591
Android Developer Tools (ADT) plugin, 18
Android Developers Blog, 452
Android development process.
  See Development workflow
Android emulator. See Emulator
Android manifest for filtering, in Packaging applications, 503–504
Android Marshmallow
  - auto backup feature introduced in, 99
  - permission levels, 105–106
  - permissions for fingerprint authentication, 108
Android Open Source Project (AOSP)
  - Compatibility Test Suite, 339
  - Google initiative making Android source code available, 11
Android OS, introduction to
  - AOSP (Android Open Source Project), 11
  - applications, 23–26
  - architecture underlying, 21–22
  - current focus, 15–16
  - custom forks, 26–28
  - Google’s mobile first philosophy, 12
  - Open Handset Alliance, 12–15
  - reasonable learning curve for developers, 18
  - security and permissions, 22–23
  - summary, Q&A, and references, 28–29
  - uniqueness of Android platform, 16–21
Android project view, comparing with traditional Project view, 62
Android robot, 16–17
Android Runtime (ART), 21, 106–108
Android SDK
  - accessing documentation, 40
  - Android Debug Bridge and, 591
  - code diagnostics in, 462
  - configuring development environment, 31
  - design tools in, 458–459
  - development tools in, 464
  - Device Monitor and, 567, 590–591
  - documentation for, 585–588
  - downloading and installing, 32–33
  - exploring Android SDK manager, 43–45
  - exploring core features, 40–42
  - freely available, 18
  - Hierarchy Viewer, 591–595
  - introduction to, 585
  - layout editor, 591–592
  - leveraging Android emulators with, 588–589
  - license agreement in, 38–39, 449
  - logcat for viewing log data, 589–590
  - Nine-Patch Stretchable Graphics, 596–598
  - other tools in, 597–600
  - preference system, 345
  - reporting bugs, 37–38
  - summary, Q&A, and references, 600–601
  - testing essentials, 477–480, 496
  - unit tests in, 462–463
  - upgrading, 37
Android SDK Manager

downloading Android Support Library package, 230–231
exploring, 43–45

Android Studio

adding resources to project, 112
adding simple resource values, 116–120
Autocomplete and, 533–537
build error resolution in, 537
closing unwanted tabs, 531
creating new projects, 58
custom log filters, 532
Debugger tab options, 67
designing layouts, 141–143
Device Monitor and, 567–568
editor windows, controlling, 531
editor windows, resizing, 528
editor windows, viewing side by side, 529
exploring, 42
Gradle build system and. See Gradle builds, Android Studio for
IDE for Android application development, 18
importing BorderlessButtons sample, 52–54
imports in, 533–534
IntelliJ IDEA as alternative to, 38
Intention Actions, 536–537
Java code and, 533–537
Javadoc-style documentation, 537
Method extraction, 536
new classes/methods, 533
organizing workspaces, 527
refactoring code, 535
reformatting code, 534
Rename tool in, 534
reorganizing code, 536
reporting bugs, 37
repositioning windows, 528
searching projects, 532
source control service integration, 527–528
summary, Q&A, and references, 537–538
task organization, 532–533
for Testing applications, 491–496
tips/tricks, 527
tool windows, resizing, 528–529
upgrading, 35–36
Variable extraction in, 535
viewing two sections of same file in, 529–531

Android Support Library package

adding Fragment support to legacy applications, 229
adding Fragment support to new applications targeting older platforms, 229–230
dependencies in Gradle, 606–609
designing user interfaces for compatibility, 322
dialog fragments, 264–267
leveraging for compatibility, 323
linking to project, 230–231
nested fragments added to, 232
overview of, 228
for styles, 269

Android Virtual Device Manager

creating AVDs, 543–547
creating AVDs with custom hardware settings in, 547–548
introduction to, 542–543
launching emulators, 549, 554–555

Android Virtual Devices (AVD)
calling between two emulator instances, 557–558
creating, 543–547
creating for devices you want to emulate, 55
creating with custom hardware settings, 547–548
exploring AVD manager, 43–44
GPS locations of emulators and, 555–556
introduction to, 541–542
launching emulators to run applications and, 550–554
launching emulators with, 548–549
maintaining emulator performance, 549
running BorderlessButtons project using preinstalled profile, 54
selecting project profile, 51
Android Wear API, 338
AndroidJUnitRunner, 497
AndroidManifest.xml. See Manifest file
android.util.class, 69
android.view package, 147
android.widget package, 147–148
Animation
of circular reveal, 297
referencing animation files by filenames, 114
working with, 133–135
AOSP (Android Open Source Project)
Compatibility Test Suite, 339
Google initiative making Android source code available, 11
AOT (ahead-of-time compilation), 21
Apache Software License (ASL/Apache2), 17
APIs
for adding and implementing material design, 285
Android Wear API, 338
leveraging support library for compatibility, 323
maximizing application compatibility, 319–320
supporting material Activity APIs, 291
APK (Android application package)
multiples in Google Play, 414–415
variants in Gradle, 616–618
AppCompatActivity class
for backward compatible Activity, 85
designing Fragment-based applications, 218–219
extending to support material Activity APIs, 291
Application branding, 277–279
Application components
Android terminology related to, 77–78
managing Activity transitions with intents, 87–89
organizing Activity components with fragments, 85–87
organizing application navigation, 90
overview of, 77
performing application tasks with activities, 80–85
receiving and broadcasting intents, 91–92
registering in manifest file, 104–105
retrieving and using application Context, 78–79
summary, Q&A, and references, 92–93
working with services, 90–91
Application dependencies, configuring, 615
<application> tag, 109
Applications
adding Fragment support to legacy applications, 229
adding Fragment support to new applications targeting older platforms, 229–230
architecting with patterns. See Architectural design patterns
Applications

associated with user profiles, 22
configuring using manifest file, 95–96
defining default themes, 270–271
designing Fragment-based applications, 218–219
developing. See Development workflow
distributing. See Distribution of applications
driving device sales, 14–15
exploring Android sample applications, 45–47
high-quality. See Quality Android applications
introduction to Android APIs, 23–26
maintaining. See Maintenance of applications
managing on Google Play, 517–518
maximizing compatibility, 319–321
packaging. See Packaging applications
placing application icon in ActionBar, 252
popular third-party, 41
publishing. See Publication of applications
requirements of, 413–421, 423
standard for development and design. See Material design
testing. See Testing applications
version systems for, 427

Applications, building first
adding logging support to, 69–70
Android project view and traditional Project view, 62
core files and directories, 62–64
creating/configuring new project, 57–61
debugging new application on hardware, 71–73
debugging new project in emulator, 66–69
overview of, 57
running new project in emulator, 64–66
summary, Q&A, and references, 73–74

Applications, testing development environment
importing BorderlessButtons sample into Android Studio, 52–54
overview of, 51
running BorderlessButtons application in emulator, 55–57
running BorderlessButtons project using preinstalled AVD, 54–55

ApplicationTest class, 485

Apps. See Applications

Architectural design patterns
action bar compatibility, 255
action bars, 251–252
action buttons, 252–253
action overflow, 253–255
action-related features, 251
actions originating from application content areas, 257–258
dialogs, 258
floating action buttons, 256–257
hierarchical relationships between screens, 242–244
implementing dialogs, 259
launching tasks and navigating the back stack, 240–241
master detail flows, 249–250
menus, 251
navigating with fragments, 241–242
navigation drawer, 247–249
navigation scenarios, 237–240
overview of, 237
summary, Q&A, and references, 267–268
swipe views, 244–246
Boolean resources, working with

Authentication, permissions for fingerprint authentication, 108

Auto application
   extending applications to watches and cars, 338
   quality guidelines, 455

Auto Backup, 99, 359–361

Autocomplete, 155–157, 533–537

AutoCompleteTextView, 155–156

autolink attribute, creating contextual links in text, 150–152

Automation, of testing applications, 472, 496–497

AVD. See Android Virtual Devices (AVD)

B

Back navigation, 239

Back stack, navigating, 240–241

Backups
   auto backup feature, 99, 359–361
   in testing applications, 476

Behaviors, creating reusable behavior components without a user interface, 231–232

Beta Channel, options for upgrading Android Studio, 35

Billing
   distribution of applications and, 501–502
   in-app, 477
   methods for, 453–454

Binding application data, to RecyclerView, 293–296

BitmapDrawable, working with image resources, 130

Black-box testing, 473

Bold attribute, working with string resources, 121

Boolean resources, working with, 124–125
BorderlessButtons sample app
importing into Android Studio, 52–54
running in emulator, 55–57
running using AVD, 54–55
Branding, application branding, 277–279
Breakpoints, viewing/setting, 67
Broadcasts, of intent, 91–92
Browsing files of emulators/devices, in File Explorer, 575–576
Bug reports, in development workflow, 433
Builds
errors, 537
in Gradle. See Gradle build system validation, 471–472
Built-in content providers
CalendarContract content provider, 398–399
CallLog content provider, 397–398
ContactsContract content provider, 400–402
MediaStore content provider, 394
overview of, 394
Settings content provider, 399
UserDictionary content provider, 399
Voicemail content provider, 399
Built-in layout classes
FrameLayout view, 193–195
GridLayout view, 198–202
LinearLayout view, 187–189
overview of, 185–187
RelativeLayout view, 189–193
TableLayout view, 196–198
Button objects
adding action buttons to ActionBar, 252–253
creating layout using XML resources, 186
defining for applying state list resource, 132–133
floating action buttons, 256–257, 275–276
for PasswordMatcher, 481–485
as targets, 243–244
using basic buttons, 158–161
Cache, data, 364
CalendarContract content provider, 398–399
Callback methods
attaching/detaching fragments with activities, 216–217
using Activity callbacks to manage application state and resources, 81–82
CallLog content provider
adding required permissions for accessing, 398
overview of, 397–398
Calls
simulation of incoming, 559–560, 579
between two emulator instances, 557–558
Canary Channel, options for upgrading Android Studio, 35
Card data object
adding cards to database, 384, 386–387
completing transition and revealing new card, 304–306
creating, 291–292
deleting cards, 313–317, 388–390
editing card activity, 310–313
initializing using initCards(), 292
querying (getAll) in database, 385–386
updating cards, 387–388
viewing/editing card transition, 306–310
CardView class
animating circular reveal, 297
grouping View widgets, 286
implementing ViewHolder class, 296–297
showing dataset in list, 288
types of user interface controls, 210

Cars, extending applications to, 338–339
CharacterPickerDialog, 258
CheckBox control, 161–163
Child views, view controls, 184–185
Chronometer control, 174
Circular reveal, animation of, 297
Classes. See also by individual types
built-in classes for layouts, 185–187
importing required packages, 70
modeling in planning user experiences, 440
Clean starting states, in testing applications, 470
Click
click handlers for RadioButton control, 165
organizing application navigation, 90
Client/server applications, testing, 424
Closing unwanted tabs, in Android Studio, 531
Cloud-friendly applications, testing, 424
Code
creating run configuration for testing, 485–489
delivering quality code, 461–463
Google initiative making Android source code available, 11
instructions in Android applications, 111
for packaging and distribution of applications, 503
preparing for distribution of applications, 503
step through, 67
working with Java code, 533–537
Colors
adding simple resource values, 116–120
in application branding, 277–279
attributes, 273
defining by RGB values, 288–289
styles and themes, 272–273
working with, 126
working with color state lists, 131–133
colors.xml file, 288–289
Compatibility
creating alternative resources, 325–326
designing user interfaces, 321–322
extending applications to watches and cars, 338–339
leveraging Android Support library APIs, 323
maintaining backward compatibility, 374–375
maximizing application compatibility, 319–321
Nine-Patch Stretchable Graphics and, 324
organizing alternative resources, 334–335
organizing alternative resources with qualifiers, 326–331
programmatic use of alternative resources, 333–334
providing alternative resources for different screen orientations, 331–333
reasons for including alternative resources, 324–325
SafetyNet ensuring, 339–340
screen type support, 323–324
summary, Q&A, and references, 340–342
targeting tablet devices, 336–337
targeting TV devices, 337–338
working with fragments, 323
Compatibility Test Suite, AOSP, 339
<compatible-screens> tag, 102
Configuration changes
handling, 335
retaining data across, 335
Configuration file. See Manifest file
Conformance standards, in testing applications, 476
Console, Developer, 509–512, 515–517
Console, Emulator. See Emulator console
ContactsContract content provider
adding records to, 402–404
data column classes, 402
overview of, 400
updating records, 404–405
working with, 400–402
Container class controls, in layouts
AdapterView controls, 206–207
ArrayAdapter binding array element to views, 204–205
CursorAdapter binding columns of data to views, 205–206
data-driven containers, 204
list of, 209–210
ListView controls, 207–208
overview of, 203
Content areas, actions originating from application, 257–258
Content providers
accessing when permissions required, 396
adding records to, 402–404
built-in, 394
CalendarContract content provider, 398–399
CallLog content provider, 397–398
ContactsContract content provider, 400–402
Cursor objects and, 205–206
deleting records, 405–406
exploring, 393
MediaStore content provider, 394–396
<providers> tag, 109
Settings content provider, 399
summary, Q&A, and references, 406–407
third-party, 406
updating records, 404–405
UserDictionary content provider, 399
Voicemail content provider, 399
ContentProviderOperation class
deleting records, 405–406
updating records, 404–405
Context class
defined, 77
file access methods, 366–367
openFileOutput(), 367
retrieving and using application context, 78–79
Context menu, 251
Contextual action mode, 256
Contextual links, creating in text, 150–152
Control of test releases, in development workflow, 431–432
Controls. See also by individual types, 147–148
Copying files, to/from emulators/devices, 577
Core app quality guidelines, 454–455
Coverage of tests, 471–477
Crash reports, in development workflow, 433
CRM (customer relationship management), 397–398
CursorAdapter control, binding columns of data to views, 205–206
Developers, reasonable learning curve for

Debug class, 476

Debugging
- configuring hardware debugging, 34–35
- configuring operating system for device debugging, 34
- in Device Monitor, 569
- new application on hardware, 71–73
- new project in emulator, 66–69
- reporting bugs, 37–38

Default resources
- defined, 324
- using, 115–116

Defect-tracking systems, 467–469

Defects, redefining for mobile applications, 468–469

Deleting
- cards, 313–317, 388–390
- files from emulators/devices, 577–578
- preferences, 348–349
- records, 405–406

Dependencies, adding to application, 286–287

Deploying Android applications, 432

Descendant navigation, 238–239

Design
- architecting with patterns. See Architectural design patterns
- designing Android applications in development workflow, 427–430
- of developmental process, 460
- material design. See Material design
- planning user experiences, 444
- tools for, 458–459

Dev Channel, options for upgrading Android Studio, 35

Developer Console, 477, 509–512, 515–517

Developers, reasonable learning curve for, 18

CursorLoader class, performing cursor queries, 393

Custom dialogs, 263–264

Custom log filters, 532

Customer relationship management (CRM), 397–398

Cyanogen OS, 27

Data. See also Resources
- retaining across configuration changes, 335
- storing application data on devices, 363–364

Data access object, creating, 381–382

Data-driven containers, 204

Data types
- supported as preference-setting values, 346
- utilities for handling different file types, 366
- value types, 112–114

Databases. See also SQLite
- accessing, 379–380
- adding cards, 384, 386–387
- creating, 377–379
- deleting cards, 388–390
- in development workflow, 417–418
- operations off main UI thread, 383–384
- querying (getAll) cards, 385–386
- storing device data in, 419–420
- third-party, 420–421
- tracking devices in, 419
- updating cards, 387–388
- using device data from, 420

DatePicker control, 166–168

DatePickerDialog, 257

Dates, retrieving using pickers, 166–168
Development, setting up for

accessing Android SDK documentation, 40
configuring development environment, 31–34
configuring hardware debugging, 34–35
configuring operating system for device debugging, 34
exploring Android emulator, 44–46
exploring Android sample applications, 45–47
exploring Android SDK and AVD managers, 43–45
exploring Android Studio, 42
exploring core features of Android SDK core, 40–42
IntelliJ IDEA as alternative to Android Studio, 38
overview of, 31
reporting Android SDK bugs, 37–38
reporting Android Studio bugs, 37
summary, Q&A, and references, 48–49
understanding Android SDK license agreement, 38–39
upgrading Android SDK, 37
upgrading Android Studio, 35–36

Development workflow

application requirements, 413–421
application version systems, 427
architecture of Android devices, 428
bug reports, 433
configuration management systems, 426–427
controlling test releases, 431–432
crash reports, 433
customization method, 414–416
deploying Android applications, 432
designing Android applications, 427–430
devolving Android applications, 430
device databases, 417–421
documentation, 425–426, 433
exercises for, 435
extensibility, 428–429
feasibility of application requirements, 423
Google Play Staged Rollouts, 432
interoperability of applications, 430
introduction to, 411
iterative process, 413
limitations of Android devices, 428
live server changes, 434
low-risk porting opportunities, 434
lowest common denominator method, 413–414, 416
maintaining Android applications, 433–434
maintenance, designing for, 428–429
maintenance, documentation for, 426
maximizing application compatibility, 319–320
overview of, 411–412
porting documentation in, 426
private controlled testing, 431
private group testing, 431–432
project risk assessment, 421–432
quality assurance, risk assessment for, 423–424
quality assurance, test plans for, 425–426
releases of devices, 422
retirement (sunset) of devices, 422
software methodologies, 412–413
source control systems, 426–427
summary, Q&A, and references, 434–436
supporting Android applications, 433–434
target devices, acquiring, 422–423
target devices, identifying, 421–422
target markets, 432
testing applications, 424, 431–432
testing firmware upgrades, 433
third-party documentation
requirements, 426
third-party requirements/
recommendations, 417
third-party testing facilities, 426
use cases in, 416
user interface documentation, 425
waterfall approaches to, 412–413
Device configuration, in testing applications, 469–470
Device databases. See also Databases
in development workflow, 417–418
selecting devices for tracking, 419
storing data, 419–420
third-party, 420–421
using data from, 420
Device Monitor
Allocation Tracker, 573–574
debuggers in, 569
debugging applications, 590–591
Emulator Control pane, 578–579
File Explorer pane, 575–578
Garbage Collection, 571–572
heap activity, 570–571
HPROF files, 572–573
interacting with processes, 569
key features of, 568–569
LogCat tool, 581–582
Network Statistics, 574–575
Quick-Start, 567
screen captures of emulators/device
screens, 580–581
as stand-alone application, 567–568
stopping processes, 570
summary, Q&A, and references, 582–583
System Information pane, 580
thread activity, 570
using with Android Studio, 567–568
Devices. See also Hardware
debugging applications on, 71–73
extending applications to watches and
cars, 338–339
maximizing application compatibility,
320–321
specifying device features with
manifest file, 101–102
storing application data on, 363–364
support for adoptable storage, 372
targeting tablet devices, 336–337
targeting TV devices, 337–338
Diagnostics, in quality Android applications,
456–457
Dialog class
implementing dialogs, 259
overview of, 258
summary, Q&A, and references, 267–268
types of dialogs, 257, 259–260
working with Android Support
package dialog fragments, 264–267
working with custom dialogs,
263–264
working with dialogs and dialog
fragments, 260–263
DialogFragment class
lifecycle of, 261
managing dialogs for use with user
interface, 261
working with special types of
fragments, 218
Digital signatures, in distribution of
applications, 504–508
Dimensions

adding simple resource values in
Android Studio, 116–120
designing user interfaces for
compatibility, 322
working with, 126–128

Directories

core, 62–64
creating and writing to files in default
application directory, 367–368
default resource directories, 112
exploring, 366–367
reading from files in default application
directory, 369
retrieving and using application
Context, 79
viewing, 61
working with other directories and
files on Android file system, 372–373

DisplayMetrics utility, getting information
about screen, 322

Distribution of applications

ad revenues in, 502
Android manifest for filtering in,
503–504
billing users, 501–502
code preparation for, 503
details in Google Play, 515
digital signatures in, 504–508
disabling debugging/logging for, 504
distribution channels and, 508–509
file packages, 504–508
Google Play policies, 501
Google Play requirements, 504
intellectual property protection in, 500
introduction to, 499
models for, 499–502
name/icons of applications in, 503
packaging applications for publication
and, 502–509
permissions, verifying, 504
publishing to alternative
marketplaces, 520
required resources in, 508
self-publishing applications for,
520–521
servers/services in, preparing, 508
summary, Q&A, and references,
522–523
target platforms, verifying, 503
testing release versions in, 508
translating applications for, 519
versioning applications, 503

Dividers, for visual appeal, 279–280

Documentation

accessing Android SDK
documentation, 40
for Android SDK tools, 585–588
in development workflow, 425–426
Javadoc-style documentation in
Android Studio, 537

Domain modeling, 440

Dot (.) character, shortcut in package
specification, 98

Drawable resources

adding simple resource values in
Android Studio, 116–120
defining circle-shape, 270–271
defining circle-shape, 270–271
working with, 128–129
<drawable> tag, 128

drawerLayout control, 210

drawers, organizing application
navigation, 90
E

E-mail apps, implementing service for, 91

Eclipse

ADT facilitating Android development
with, 18
ADT plugin no longer supported, 32, 527
Android Support Library for use with, 231
choosing source control system, 426

**Editor windows**
controlling, 531
resizing, 528
viewing side by side, 529

**Editors/editing**
card activity, 310–313
card transitions, 306–310
layout editor, 591–592
manifest file, 96–99
preference editor, 348
video editing, 91

**EditText boxes**, 481–485, 490

**EditText controls**
constraining user input with input filters, 154–155
retrieving text input, 152–154

**Education guidelines**, 455
ems, measuring width of TextView, 150

**Emulator. See also Android Virtual Device Manager; Android Virtual Devices (AVD)**
Android SDK tools and, 588–589
calling between two instances in, 557–558
debugging new project, 66–69
exploring, 44–46
GPS locations of, 555–556
limitations of, 563–564
messaging between two instances, 558–559
personalizing, 563
Quick-Start for, 539
running BorderlessButtons application in, 55–57
running new project in, 64–66
simulating reality, 539–541
summary, Q&A, and references, 565–566
testing applications, 472–473

**Emulator console**
manipulating power settings, 562–563
monitoring network status, 562
sending GPS coordinates, 561
simulating incoming call, 559–560
simulating SMS message, 560–561

**Emulator Control pane**, 579
introduction to, 578
location fixes in, 579
simulating incoming SMS messages, 579
telephony status in, 578–579

**Encryption**, 453
End User License Agreement (EULA), 453
Entity discovery/organization, 440–441
Entity relationship modeling, 440
Entry navigation, 238

**Environment class**, accessing external storage, 374

**Environments**, managing test environments, 469–471

**Errors**, debugging new project in emulator, 66–69

**Espresso**, 497

**EULA (End User License Agreement)**, 453

**Events**, handling selection events, 206–207

**Exerciser Monkey**, 497

**Experience, user. See Planning user experience**

**Extensibility**, in development workflow, 428–429

**External navigation**, 240

**External storage**, 374
Feasibility

of application requirements, 423

testing, 460

Features

specifying device features with manifest file, 101–102
targeting tablet devices, 336
targeting TV devices, 338

File Explorer pane

browsing files of emulators/devices, 575–576
copying files to/from emulators/devices, 577
deleting files from emulators/devices, 577–578
directories in, 576
finding preferences data, 349–350
introduction to, 575

File objects, 372–373

Files

core, 62–64
creating and writing to external storage, 374
creating and writing to in default application directory, 367–368
file package in distribution, 504–508
finding preferences data in file system, 349–350
good management practices, 364–365
in Gradle, 603–604
maintaining backward compatibility, 374–375
permissions, 365
reading from files in default application directory, 369
reading raw files byte by byte, 369–370
reading XML files, 370–371

referencing animation files by filenames, 114
retrieving and using application Context, 79
storing, 111–112, 114
storing application data on devices, 363–364
summary, Q&A, and references, 375–376
support for adoptable storage devices, 372
viewing, 61
working with, 366
working with other directories and files on Android file system, 372–373

Filters

application filtering options, 102
configuring intent filters, 103–104
constraining user input with input filters, 154–155
packaging applications and, 503–504

Fingerprint, permissions for fingerprint authentication, 108

Fire OS (Amazon), 26–27

Firmware upgrades, 433

Fixed tabs, navigation design patterns, 246–247

Flavors, Gradle builds, 611–614

Floating action buttons

adding primary action to material application, 297–301
adding to main layout, 289–290
overview of, 256–257
styles and themes and, 275–276

Fonts, typography, 280–282

Forks, custom uses of Android, 26–28

Format strings, 121

Fragment class

adding Fragment support to legacy applications, 229
adding `Fragment` support to new applications targeting older platforms, 229–230

Android Support Library package and, 228

attaching/detaching with activities, 216–217

creating reusable behavior components without a user interface, 231–232

defining, 215–216

defining `Activity` classes for hosting `Fragment` components, 227–228

defining layout resource files for, 225–226

designing compatible applications, 323

designing `Fragment`-based applications, 218–219

designing user interfaces for compatibility, 322

dialog fragments in Support package, 264–267

implementing `ListFragment`, 219–223

implementing `WebViewFragment`, 223–225

lifecycle of, 214–215

linking Android Support package to project, 230–231

`ListFragment` control, 207–208

managing modifications, 216

master detail flows, 249–250

navigation with, 241–242

nested, 232

organizing `Activity` components with, 85–87

organizing application navigation, 90

overview of, 78

special types of, 217–218

summary, Q&A, and references, 232–234

targeting tablet devices, 336

understanding, 213–214

working with dialog fragments, 260–263

Fragmentation issues. See also Compatibility, 319

`FragmentTransaction`, 216

Frame-by-frame animation, 133–134

`FrameLayout` view

attributes, 195

using, 193–195

Games

Google Play Game Services, 15, 516

implementing service for, 91

typical activities in, 80

Gaps, for visual appeal, 279–280

Garbage Collection, 571–572

GCM (Google Cloud Messaging), 516

`getAll()`, querying cards in database, 385–386

`getQuantityString()`, working with quantity strings, 123

`getResources()`, retrieving application resources, 79, 121–122

Global Positioning System (GPS), locations of emulators, 555–556

GMS (Google Mobile Services), 13

GNU General Public License Version 2 (GPLv2), 17

Goals, of testing applications, 485

Google

list of Google services, 26

mobile first philosophy, 12

role Open Handset Alliance, 13

Google Cloud Messaging (GCM), 516

Google Experience devices, 423

Google Maps, 555–556

Google Mobile Services (GMS), 13
Google Play

Android markets, 20
application filtering options, 102
configuring application options, 516
Developer Console in, 509–512, 515–517
Developer Distribution Agreement, 476, 501, 510
developers registering with, 23
distribution details in, 515
in distribution of Android apps, 14–15
distribution policies of, 501
distribution requirements of, 504
Game Services in, 516
for low cost development, 19
managing applications on, 517–518
packaging requirements of, 504
pricing details in, 515
Private Channel in, 518–519
publishing applications to, 509, 516
removing applications from, 518
return policy, 517
SafetyNet service, 339
signing up for, 509–511
Staged Rollouts in, 432, 518
targeting TV devices, 338
uploading application marketing assets to, 514
uploading applications to, 511–513
Google Wallet Merchant accounts, 501, 510–511
GPLv2 (GNU General Public License Version 2), 17
GPS (Global Positioning System), locations of emulators, 555–556
Gradle build system
adding dependencies to application, 286–287
files in, 603–604
module settings in, 606–609
project settings in, 604–605
Quick-Start for, generally, 603
summary, Q&A, and references, 620–621
support library dependencies in, 606–609
wrappers in, 606–609
Gradle builds, Android Studio for
adding library dependencies, 615–616
APK variants, 616–618
configuring Android properties in, 609–611
configuring application dependencies, 615
configuring build flavors in, 611–614
configuring build types in, 614–615
introduction to, 603, 609
running tasks in, 618–620
signing options in, 611
syncing projects, 609
Graphics
Nine-Patch Stretchable Graphics, 130
referencing by filenames, 114
storing, 114
storing resources files, 112
using alternative resources, 115
working with image resources, 129
GridLayout view
attributes, 200
using, 198–202
GridView controls, 204
Guidelines for Android applications, 454–456

H
HAL (hardware abstraction layer), 21
Hardware
configuring debugging, 34–35
debugging new application (Nexus 4 example), 71–73
manifest file enforcing application platform requirements, 100
maximizing compatibility, 320–321
open source, 27–28
upgrading, 15
`hasSystemFeature()`, specifying device features, 101–102
Headers, organizing preferences, 354–359
Heap activity, in *Device Monitor*, 570–571
Hierarchies, screen relationships, 242–244
Hierarchy Viewer
  introduction to, 592
  launching, 593
  Layout View mode in, 592–594
  Pixel Perfect mode in, 592, 595
  user interface optimization in, 594–595
HPROF files, in *Device Monitor*, 572–573
HTML
  HTML-style attributes, 122
  working with layouts, 140

IARC (International Age Rating Coalition), 516
Icons
  placing in *ActionBar*, 252
  setting application icon, 99
  working with image resources, 129
Identity, managing application identity, 99
Identity of applications, in planning user experience, 442–443
IDEs (integrated development environments).
  See also *Android Studio*
  alternatives to *Android Studio*, 38
  for application development, 18
  configuring development environment, 32
Images
  formats, 130
  *MediaStore* content provider, 394
  storing on devices, 363
  working with image resources, 129–131
Imports
  in *Android Studio*, 533–534
  BorderlessButtons sample, 52–54
  Optimize Imports command, 70
  `<include>` statement, including layout within other layouts, 274–275
Indicator controls
  adding progress indicators to *ActionBar*, 170–171
  adjusting progress with seek bars, 171–172
  displaying progress and activity using, 168–169
  indicating activity with activity bars and activity circles, 171
Inheritance, styles and themes, 271–272
`initCards()`, initializing *Card* data object, 292
Input filters, constraining user input with, 154–155
Input methods, specifying with manifest file, 100–101
Input mode
  targeting tablet devices, 336
  targeting TV devices, 337
InputFilter objects, 154–155
Installations, in testing applications, 476
`<instrumentation>` tag, unit tests, 109
Integer resources, 125–126
Integration points, in testing applications, 474–475
Intellectual property protection, 500
IntelliJ IDEA
as alternative to Android Studio, 38
Android Studio based on, 18, 32, 527

Intent class
configuring intent filters, 103–104
creating intents with action and data, 88–89
defined, 78
designating primary entry point using intent filter, 103
managing Activity transitions with, 87–88
organizing application navigation, 90
passing information using, 89–90
receiving and broadcasting, 91–92

Intention Actions feature, 536–537

International Age Rating Coalition (IARC), 516
Internationalization, in testing applications, 475
Interoperability of applications, 430
Interprocess communication (IPC), 91

Introduction, to book
authors’ contact information, 8
changes to this edition, 3–5
conventions used in book, 6–7
development environments used in book, 5–6
how book is structured, 2–3
intended audience, 1–2
key issues addressed, 2
list of useful websites, 7
supplementary materials, 6

IPC (interprocess communication), 91
Issue Tracker website, reporting bugs to, 37–38

Italic attribute, working with string resources, 121

itemView
implementing ViewHolder class, 296–297
OnClickListener of, 306

J
Java
Android Studio and, 533–537
Autocomplete, 533–537
developing Android applications with, 24, 533–537
documentation, 537
file utilities, 366
imports, 533–534
Intention Actions, 536–537
Method extraction, 536
new classes/methods, 533
refactoring code, 535
reformatting code, 534
Rename tool and, 534
reorganizing code, 536
Variable extraction, 535

JavaDoc-style documentation, in Android Studio, 537

java.io
reading raw files byte by byte, 369–370
utilities for handling different file types, 366

JDK (Java Development Kit), 31

JetBrains
IntelliJ IDEA. See IntelliJ IDEA
reporting bugs via, 37

JUnit tests. See also Unit tests, 462–463, 480–481
defining layout files for fragments, 225–226

designing screens for compatibility, 324

designing with Android Studio, 141–143

FrameLayout view, 193–195

GridLayout view, 198–202

LinearLayout view, 187–189

list of view containers, 209–210

ListView controls, 207–208

organizing your user interface, 184–185

RelativeLayout view, 189–193

scrolling support added to, 208–209

styles and themes, 273–274

summary, Q&A, and references, 210–212

TableLayout view, 196–198

using layout resources programmatically, 144

using multiple layouts on a screen, 202–203

working with, 140–141

LBS (Location-based services), 416, 555–556

Legacy applications

adding ActionBar to, 255

adding Fragment support to, 229

Libraries. See also Android Support Library package

adding library dependencies, 615–616

adding support libraries to application, 286–287

License Verification Library (LVL), 500

Licenses

Android, 17

understanding license agreement, 38–39
Lifecycle
- of an activity, 80–82
- of fragments, 214–215

LinearLayout view
- attributes, 188–189
- creating layout using XML resources, 186
- defining layout for transitions, 299–301
- instantiating programmatically, 181–184 using, 187–189

Lines, measuring height of TextView, 150

Links, creating contextual links in text, 150–152

Linux OSs
- Android running on, 21
- configuring operating system for device debugging, 34
- SELinux (Security-Enhanced Linux), 22–23

ListFragment class
- designing Fragment-based applications, 218–219
- implementing, 219–223
- using ListView control with, 207–208
- working with special types of fragments, 217

ListView control
- binding data to AdapterView, 206
- types of data-driven containers, 204
- using with ListFragment, 207–208

Live server changes, 434

Location-based services (LBS), 416, 555–556

Location fixes, in Emulator Control, 579

Logcat utility
- Device Monitor, 581–582
- filtering logging messages, 70
- logging application information, 67
- for viewing log data, 589–590

Logging
- adding logging support to application, 69–70
- application information, 67
- defect information, 467–468
- filtering logging messages, 70
- in quality Android applications, 462–463
- viewing log data, 589–590

Logos, 115

Low-risk porting opportunities, 434

LVL (License Verification Library), 500

M

Mac OS X, configuring operating system for device debugging, 34

Maintenance of applications
- designing for, 428–429
- in development process, 433–434
- documentation for, 426, 433
- ease of, 456

Maintenance of emulator performance, 549

Maker Movement, Android versions, 27–28

Managers
- exploring Android SDK and AVD managers, 43–44
- handling state of system services, 25–26

Manifest file
- configuring Android applications using, 95–96
- editing, 96–99
- enforcing system requirements, 100–102
- list of configurable features, 109
- managing application identity, 99
- registering activities in, 103–105
- registering hardware and software requirements, 321
Multimedia, storing application data on devices

registering permissions enforced by application, 108–109
registering permissions required by application, 105–106
requesting permissions at runtime, 106–108
summary, Q&A, and references, 109–110
targeting TV devices, 337
Manufacturers, members of Open Handset Alliance, 13–14
Markets/marketing. See also Google Play revenue models and, 19
uploading application marketing assets to Google Play, 514
Master detail flows, navigation design patterns, 90, 249–250
Material design
adding dependencies to application, 286–287
adding primary action, 297–301
animating circular reveal, 297
binding application data to RecyclerView, 293–296
completing transition and revealing new card, 304–306
creating/initializing Card data object, 291–292
creating layout, 289–290
default themes, 286
defining colors, 288–289
defining string resources, 289
deleting cards, 313–317
editing card activity, 310–313
extending AppCompatActivity class, 291
implementing TransitionActivity class, 302–304
implementing ViewHolder class, 296–297
styling applications, 287–288
summary, Q&A, and references, 317–318
understanding, 285–286
viewing/editing card transition, 306–310
Material theme support library, 287
MediaController object, 177
MediaStore content provider, 394–396
Menus
organizing application navigation, 90
styling, 280
types of, 251
working with, 135–137
<merge> tag, including layout within other layouts, 274–275
Metadata, 109
Method extraction, in Android Studio, 536
Mimicking real-world activities, in testing applications, 470–471
Mobile applications
Google’s mobile first philosophy, 12
testing applications, 467
Mobile operators, members of Open Handset Alliance, 14
Mockups, 445–446
MODE_APPEND, file permissions, 365
MODE_PRIVATE, file permissions, 365
Module settings, in Gradle, 606–609
Monitoring network status, 562
monkey program (UI/Application Exerciser Monkey), 496
monkeyrunner test tool, 472, 496
MultiAutoCompleteTextView, 155–157
Multimedia, storing application data on devices, 363
N scenarios, navigation, 237–240

Names
distribution of applications and, 503
setting application name, 99

Navigation
with fragments, 241–242
hierarchical relationships between screens, 242–244
launching tasks and navigating the back stack, 240–241
master detail flows, 249–250
n scenarios, 237–240
navigation drawer, 247–249
organizing application navigation, 90
swipe views, 244–246
tabs, 246–247
targeting tablet devices, 336
targeting TV devices, 337
targets, 243–244
Navigation drawer, 247–249
Nested fragments, 232
Nested layouts, 202
Network Statistics, in Device Monitor, 574–575
Networks, downloading content from, 363
New classes/methods, in Android Studio, 533
New Project creation wizard, 58
News apps, implementing service for, 91
Nexus
debugging new application on, 71–73
exploring Android emulator, 46
Google’s Android offering, 14
Nine-Patch Stretchable Graphics
Android SDK tools, 596–598
screen compatibility and, 324
working with, 130
Numbers, retrieving using pickers, 166–168

OHA (Open Handset Alliance)
members, 13–15
overview of, 12–13
onActivityResult(), implementing material activity, 304
OnClickListener
adding to floating action button, 298–299
deleting cards, 313–317
viewing/editing card transition, 306
onCreate()
initializing static Activity data, 82
SampleMaterialActivity class, 381
TransitionEditActivity class, 310
onDestroy(), destroying static Activity data, 84–85
onPause(), stopping, saving, releasing Activity data, 83
onResume(), initializing and retrieving Activity data, 83
onSaveInstanceState(), saving Activity state into a bundle, 84
onStart(), confirming Activity features, 83
openFileOupput(), 367
OpenGL ES, 101–102
Optimize Imports command, 70
Options menu, 251
Orientation of screen
providing alternative resources for different orientations, 331–333
targeting tablet devices, 336
targeting TV devices, 337
OS (operating system), configuring for device debugging, 34
Overscan margins, targeting TV devices, 337
Planning user experience

class modeling, 440
design comps, 444
determining objectives, 437–439
domain modeling, 440
domain discovery/organization, 440–441
domain relationship modeling, 440
exercises for, 447
focusing on objectives, 439–442
identity of applications, 442–443
introduction to, 437
mockups, 445–446
personas, 439–440

Performance, in testing applications, 476

Permission groups, 108

<permission> tag, for fingerprint authentication, 109

Permissions
accessing CallLog content provider, 398
accessing content providers that require, 396
accessing Settings content provider, 399
applications enforcing own, 23
file permissions, 365
registering enforced, 108–109
registering required, 105–106
requesting at runtime, 106–108
verifying, 504

Personalizing emulators, 563

Personas, 439–440

Pickers, retrieving dates, times, and numbers, 166–168

Pixel density, specifying, 102

Pixel Perfect mode, in Hierarchy Viewer, 592, 595

Planning user experience

Parent views, view controls, 184–185

Partitioning. See Fragment class

PasswordMatcher application
adding additional tests, 493–496
analyzing test results, 492–493
Android Studio for, 491–496
assertions, 491
goals in testing of, 485
introduction to, 481–485
run configurations of code for, 485–489
unit-testing APIs for, 491
writing tests for, 489–491

Packages
commonly used, 25
file packages in, 504–508
important, 41
naming, 99
shortcut in package specification, 98
views and controls, 147

Packaging applications
code preparation for, 503
digital signatures, 504–508
enabling debugging/logging for, 504
distribution channels and, 508–509
file packages in, 504–508
filtering, 503–504
Google Play requirements, 504
introduction to, 502–503
name/icons of applications in, 503
permissions verification, 504
preparing servers/services, 508
required resources, 508
target platforms, 503
testing release versions, 508
versioning applications, 503

Packaging applications
commonly used, 25
file packages in, 504–508
important, 41
naming, 99
shortcut in package specification, 98
views and controls, 147

Packaging applications
code preparation for, 503
digital signatures, 504–508
disabling debugging/logging for, 504
distribution channels and, 508–509
file packages in, 504–508
filtering, 503–504
Google Play requirements, 504
introduction to, 502–503
name/icons of applications in, 503
permissions verification, 504
preparing servers/services, 508
required resources, 508
target platforms, 503
testing release versions, 508
versioning applications, 503

Parent views, view controls, 184–185

Partitioning. See Fragment class

PasswordMatcher application
adding additional tests, 493–496
analyzing test results, 492–493
Android Studio for, 491–496
assertions, 491
goals in testing of, 485
introduction to, 481–485
run configurations of code for, 485–489
unit-testing APIs for, 491
writing tests for, 489–491

PCB (printed circuit board), 27–28

Performance, in testing applications, 476

Permission groups, 108

<permission> tag, for fingerprint authentication, 109

Permissions
accessing CallLog content provider, 398
accessing content providers that require, 396
accessing Settings content provider, 399
applications enforcing own, 23
file permissions, 365
registering enforced, 108–109
registering required, 105–106
requesting at runtime, 106–108
verifying, 504

Personalizing emulators, 563

Personas, 439–440

Pickers, retrieving dates, times, and numbers, 166–168

Pixel density, specifying, 102

Pixel Perfect mode, in Hierarchy Viewer, 592, 595

Planning user experience

class modeling, 440
design comps, 444
determining objectives, 437–439
domain modeling, 440
domain discovery/organization, 440–441
domain relationship modeling, 440
exercises for, 447
focusing on objectives, 439–442
identity of applications, 442–443
introduction to, 437
mockups, 445–446
personas, 439–440
prototypes, 446
screen layouts, 443–444
screen maps, 441–442
sketches, 443
stakeholder objectives, 438–439
summary, Q&A, and references, 446–447
target users, feedback from, 445–446
team objectives, 438
testing release builds, 446
UI storyboards, 445
user flows, 441
user objectives, 438
user story mapping, 440
visual feedback, 444–445
wireframes, 443–444
<plurals>, working with quantity strings, 123
PNG graphics, 130
Pop-up menu, 251
Porting, documentation for, 426
Portrait mode, providing alternative resources for different orientations, 332–333
Power settings, 562–563
PreferenceActivity class
creating manageable user preferences, 350
using, 353–354
PreferenceFragment class, 217
Preferences
accessing application preferences, 79
adding, updating, and deleting, 348–349
Auto Backup, 359–361
creating manageable user preferences, 350
creating preference resource file, 350–352
creating private and shared preferences, 346–347
determining when to use, 345
finding preferences data in file system, 349–350
organizing with headers, 354–359
PreferenceActivity class, 353–354
reacting to changes in, 349
retrieving and using application Context, 79
searching and reading, 347
storing preference values, 346
summary, Q&A, and references, 361–362
Pricing & Distribution tab, 515
Primitive types, storing, 114
Printed circuit board (PCB), 27–28
Private controlled testing, in development workflow, 431
Private data, 453
Private group testing, in development workflow, 431–432
Private preferences, 346–347
Product Details section, 514
Profit maximization, 453–454
Programmatic approach
accessing Boolean resources, 125
accessing resources, 116
accessing string resources, 121–122
to layouts, 144, 181–184
to menu resources, 136–137
to raw files, 138
to tweened animation, 135
use of alternative resources, 333–334
using Boolean resources, 125
using color resources, 126
using dimension resources, 127
Qualifiers

- list of important alternative resource qualifiers, 328–331
- organizing alternative resources, 326–327

Quality Android applications

- avoiding mistakes in, 459
- best practices, 449, 459–460
- code quality in delivery of, 461–463
- design tools for, 458–459
- designing development process, 460
- diagnostics, 456–457
- feasibility testing, 460
- guidelines for, 454–456
- introduction to, 449
- logging, 462–463
- maintenance, 456
- private data in, 453
- profit maximizing and, 453–454
- responsiveness of applications, 451–452
- security of applications, 453
- stability of applications, 451–452
- summary, Q&A, and references, 465
- third-party standards for, 456
- unit tests, 461–463
- upgrades and updates, 456–458
- user demands, 450
- user interface design, 450–451

Quality assurance

- in project risk assessment, 423–424
- test plans for, 425–426

Quantity strings
RadioButton control, 163–165
RadioGroup control, 163–165
RatingBar control, 172–174
Raw data, storing, 111
Raw files
  defining Raw XML resources, 137
  reading byte by byte, 369–370
  storing, 114
  working with, 138
READ_EXTERNAL_STORAGE permission, 396
Reading, preferences, 347
Real-world activity simulation, 470–471
Records, content provider
  adding, 402–404
  deleting, 405–406
  updating, 404–405
RecyclerView
  adapter for binding data set to, 293–296
  adding to main layout for application, 289–290
  implementing ViewHolder class, 296–297
  populating list of names into, 286
  types of user interface controls, 209–210
Refactoring code, in Android Studio, 535
Reformatting code, in Android Studio, 534
RelativeLayout view
  attributes, 191–193
  creating, 289–290
  using, 189–191
Releases of devices, in project risk assessment, 422
Rename tool, in Android Studio, 534
Reorganizing code, in Android Studio, 536
Reporting bugs, 37–38
Repositioning windows, within Android Studio, 528
Required resources, in distribution of applications, 508
res directory, 112
res/layout directory
  defining layout files for fragments, 225–226
  XML resources in, 179
res/value directory
  colors subdirectory, 272
  string array resources in, 123
  string resources in, 120
  themes and styles subdirectories, 270, 287–288
Resolution, designing user interfaces for compatibility, 321–322
Resource files (XML)
  creating preference resource file, 350–352
  defining layout resource files for fragments, 225–226
  storing, 111–112
  working with XML resource files, 137
Resources
  accessing programmatically, 116
  adding simple resource values in Android Studio, 116–120
  alternative. See Alternative resources
default, 115–116, 324
  referencing, 138–140, 144–145
  retrieving application resources, 79
  storing, 111–114
  summary, Q&A, and references, 145–146
  using Activity callbacks to manage application state and resources, 81–82
using default and alternative resources, 115–116
what they are, 111
working with animation, 133–135
working with Boolean resources, 124–125
working with color state lists, 131–133
working with colors, 126
working with dimensions, 126–128
working with drawable resources, 128–129
working with images, 129–131
working with integer resources, 125–126
working with layouts, 140–144
working with menus, 135–137
working with quantity strings, 123
working with raw files, 138
working with string arrays, 123–124
working with string resources, 120–122
working with XML files, 137
Responsiveness of applications, 451–452
Return policy, Google Play, 517
Reuse, creating reusable behavior components without a user interface, 231–232
RGB color values
defining color resources, 126, 288–289
styles and themes, 272
Risk assessment. See Project risk assessment
Rjava class file, accessing resources programmatically, 116–117
RSA key, 71–73
Rubin, Andy, 13
Run/Debug Configurations, 485–489, 550–553

Runtime
Android apps running as separate process, 21
requesting permissions at, 106–108

S
SafetyNet, ensuring compatibility, 339–340
SAX utilities, for working with XML files, 370
Screen compatibility mode, 324
Screen layouts, in planning user experiences, 443–444
Screen maps, in planning user experiences, 441–442
Screens
capturing emulators/device screens, 580–581
compatibility support, 323–324
designing user interfaces, 322
hierarchical relationships between, 242–244
maximizing application compatibility, 320
providing alternative resources for different screen orientations, 331–333
specifying sizes, 102
targeting tablet devices, 336
targeting TV devices, 337
using multiple layouts on a, 202–203

Scripting languages, developing Android applications with, 24
Scrollable tabs, navigation design patterns, 246–247
ScrollView, adding to layout, 208–209

SD cards
creating and writing files to, 374
maintaining backward compatibility, 374–375
support for adoptable storage devices, 372
SDK (software development kit). See Android SDK

Searches
preferences, 347
project search, 532

Security
Android platform and, 22–23
of applications, 453

Security-Enhanced Linux (SELinux), 22–23

SeekBar control, adjusting progress with seek bars, 171–172

Selections, handling selection events, 206–207

<selector> resource type, 131

Self-publishing applications, for distribution, 520–521

SELinux (Security-Enhanced Linux), 22–23

Servers
packaging and distributing applications, 508
testing applications, 473–474

Service class
defined, 78
working with services, 90–91

Set up, for development. See Development, setting up for

Settings content provider, 399

Shared preferences
creating, 346–347
reacting to changes in, 349

SharedPreferences interface
methods, 347–348
overview of, 346
reading, 347

Signing options, in Gradle, 611

Signing up, for Google Play, 509–511

Simulation of reality
in emulators, 539–541
in testing applications, 470–471

Size attributes, TextView control, 149–150

Sketches, 443

Smartwatch, 15

Smoke test design, 471–472

SMS messages
Emulator console and, 560–561
Emulator Control and, 579
sending between two emulator instances, 558–559

Social networking app, implementing service for, 91

Software
manifest file enforcing application platform requirements, 100
methodologies for, 412–413
upgrading, 15

Software development kit (SDK). See Android SDK

Source control
integrating services for, 527–528
systems for, 426–427

Spinner control, 157–158

SQLite. See also Databases
accessing databases, 379–380
adding card to database, 384, 386–387
creating databases, 377–379
creating new data access object, 381–382
database operations off main UI thread, 383–384
deleting cards, 388–390
querying (getAsAll) cards in database, 385–386
summary, Q&A, and references, 390–391
updating cards, 387–388

SQLiteOpenHelper, 377, 380

Stability of applications, 451–452

Stable Channel, options for upgrading
Android Studio, 35
Stack
- Activity, 80–81
  launching tasks and navigating back stack, 240–241

Stakeholder objectives, 438–439

Stand-alone applications, Device Monitor, 567–568

startActivity(), managing Activity transitions with intents, 87–89

State, callbacks for managing application state, 81–82

Stopping processes, in Device Monitor, 570

Storage
- creating and writing files to external storage, 374
- device data in databases, 419–420
- limited capacity of Android devices, 364
- of preference values, 346
- of resources, 111–114
- support for adoptable storage devices, 372

Store Listing tab, 514

Stretchable graphics. See Nine-Patch Stretchable Graphics

StrictMode class, 476

String resources
- accessing programmatically, 116
- defining <string-array>, 289
- using programmatically, 121–122
- working with, 120–121
- working with quantity strings, 123
- working with string arrays, 123–124

strings.xml file, 288–289

Styles
- application branding, 277–279
- applying to application, 287–288
- colors, 272–273
- compared with themes, 269–270
- defining default application themes, 270–271
- divider and gap use, 279–280
- for FloatingActionButton, 275–276
- inheritance, 271–272
- layout, 273–274
- menu styles, 280
- merge and include, 274–275
- summary, Q&A, and references, 282–284
- support libraries for, 269
- TextInputLayout, 275
- toolbar use as bottom bar, 276–277
- typography, 280–282

Subclasses, ViewGroup, 185

Sunset of devices, in project risk assessment, 422

Support Library. See Android Support Library package

<supports-gl-texture> tag, 102
<supports-screens> tag, 102

Swipe
- navigation design patterns, 244–246
- organizing application navigation, 90

SwipeRefreshLayout, 209

Syncing projects, in Gradle, 609

System Information, 580

System requirements, 100–102

System resources, 144–145

TabLayout, adding, 246–247

TABLE_CREATE, SQL statement, 379

TableLayout view
- attributes, 197
- using, 196–198

TableRow view, 197
Tablet devices

- app quality guidelines, 455
- targeting, 336–337

Tabs, navigation design patterns, 246–247

Targets

- acquiring target devices, 422–423
- identifying target devices, 421–422
- market targets, 432
- navigation design patterns, 243–244
- user target, 445–446
- verifying target platforms, 503

Tasks

- launching, 240–241
- organizing, 532–533

TDD (Test-Driven Development), 481

Team objectives, 438

Telephony status, in Emulator Control, 578–579

Test-Driven Development (TDD), 481

Testing applications, 467

- adding additional tests, 493–496
- analyzing results, 492–493
- Android SDK tools for, 477–480
- Android Studio for, 491–496
- AndroidJUnitRunner for, 497
- assertions in, 491
- automated testing programs/APIs, 496–497
- automation of, 472
- backups in, 476
- best practices, 467
- black-box testing, 473
- build validation, 471–472
- clean starting states, 470
- conformance standards, 476
- defect-tracking systems, 467–469
- in development workflow, 431–432
- device configuration, 469–470
- on emulators, 472–473
- Espresso for, 497
- exercises for, 498
- goals of, 485
- in-app billing, 477
- installation, 476
- integration points, 474–475
- internationalization of, 475
- introduction to, 467
- JUnit for, 480–481
- “killer apps,” 477
- logging defect information, 467–468
- managing environments, 469–471
- maximizing testing coverage, 471–477
- mimicking real-world activities, 470–471
- mobile application defects, 468–469
- monkey for, 496
- monkeyrunner for, 496
- PasswordMatcher for, 481–485
- performance, 476
- on real devices, 365
- release builds, 446
- risk assessment and, 424
- run configurations of code for, 485–489
- servers and services, 473–474
- smoke test design, 471–472
- specializing scenarios in, 474
- summary, Q&A, and references, 497–498
- UiAutomation, 497
- unexpected scenarios, 477
- unit tests, 480–481, 491
- upgrades, 475
- visual appeal/usability, 474
- white-box testing, 473
- writing tests for, 489–491
Testing release versions
  in distribution of applications, 508
  in packaging applications, 508
testMatchingPasswords() method, 493–494
testPreConditions() method, 490–493

Text
  configuring text layout and size, 149–150
  creating contextual links in, 150–152
  displaying with TextView, 148–149
  editors. See EditText controls
elements of string resources, 120
  retrieving text input using EditText control, 152–154

TextClock control, displaying time, 175

TextInputLayout, styles and themes and, 275

TextView control
  configuring text layout and size, 149–150
  creating layout using XML resources, 180
  defining layout for transitions, 300
  displaying text with, 148–149
  instantiating LinearLayout programmatically, 181–183
  in PasswordMatcher, 483–485
  writing tests for PasswordMatcher application, 489–491

Themes
  <application> tag attributes, 109
  compared with styles, 269–270
  default theme in material design, 286–287
  defining default, 270–271
  inheritance, 271–272

Third-party
  applications, 24
  content providers, 406
  device databases, 420–421
  documentation requirements, 426
  firmware, 469
  standards, 456
  testing facilities, 426

Threads
  database operations off main UI thread, 383–384
  in Device Monitor, 570

Time
  displaying, 175
  retrieving using pickers, 166–168
  showing passage of time with
  Chronometer, 174

TimePicker control, 167–168

TimePickerDialog, 258

TODO windows, 528, 532

ToggleButton control, 161–163

Tokenizer, assigning to
  MultiAutoCompleteTextView, 156–157

Tool windows, resizing, 528–529

Toolbar
  as action bar, 255–256
  adding progress indicators to, 170–171
  application branding and, 277–279
  as bottom bar, 276–277
  defining themes, 270
  types of user interface control, 209–210

Tracking devices, 419

TrafficStats class, 574

TransitionActivity class
  completing transition and revealing new card, 304–306
  editing card activity, 310–313
  implementing, 302–304
viewing/editing card transition, 306–310

**TransitionAddActivity class**
defining layout for, 299
implementing, 302

**TransitionEditActivity class**
onCreate(), 310
viewing/editing card transition, 307

**Translating applications, for distribution of applications, 519**

**Trust relationships, application signing for, 23**

**TV**
app quality guidelines, 455
targeting TV devices, 337–338

**Tweened animation, 134–135**

**Typography, styles, 280–282**

**U**

**UI/Application Exerciser Monkey, 496**

**UI storyboards, 445**

**UiAutomation, 497**

**uiautomator, 497**

**Underlining attribute, working with string resources, 121**

**Unit tests**
APIs for, 491
delivering quality Android applications, 461–463
<instrumentation> tag, 109
testing applications, 480–481

**Updates**
ease of, 457–458
preferences, 348–349

**Upgrades**
Android SDK, 37
Android Studio, 35–36
current Android focus, 15
ease of maintenance in, 456–458
firmware, 433
testing, 475

**Uploads**
application marketing assets to Google Play, 514
applications to Google Play, 511–513

**USB, connecting Android device to computer, 71–73**

**User interface (UI)**
adding progress indicators to ActionBar, 170–171
adjusting progress with seek bars, 171–172
autocomplete feature, 155–157
buttons, 158–161
CheckBox and ToggleButton controls, 161–163
configuring text layout and size, 149–150
constraining user input with input filters, 154–155
creating contextual links in text, 150–152
creating in Android, 179
creating reusable behavior components without a user interface, 231–232
current Android focus, 16
database operations off main UI thread, 383–384
design of, 450–451
designing for compatibility, 321–322
displaying progress and activity using indicator controls, 168–169
displaying rating data with RatingBar, 172–174
displaying text with TextView, 148–149
displaying time, 175
documentation for, 425
in Hierarchy Viewer, 594–595
indicating activity with activity bars
and activity circles, 171
navigating Android system UI, 237
organizing, 184–185
playing video with VideoView, 175–177
RadioGroup and RadioButton
controls, 163–165
retrieving dates, times, and numbers
using pickers, 166–168
retrieving text input using EditText
control, 152–154
showing passage of time with
Chronometer, 174
Spinner control for user choices,
157–158
summary, Q&A, and references,
177–178
targeting tablet devices, 336
types of user interface controls,
209–210
UI elements, 257
views and layouts for, 147–148
User story mapping, 440
UserDictionary content provider, 399
Users
Android apps associated with user
profiles, 22
creating manageable preferences, 350
meeting objectives of, 438
meeting quality demands of, 450
members of Open Handset Alliance, 14
multiple users with restricted profiles, 23
planning user experience. See Planning
user experience
<uses-configuration> tag, 100–101
<uses-features> tag, 101–102
<uses-permissions> tag, 105–106, 396

V

Value types
resource types, 112–113
storing primitive types, 114

Variable extraction, in Android Studio, 535

Version systems
development workflow and, 427
distribution of applications and, 503

Video
implementing service for video
editing, 91
MediaStore content provider, 394
playing with VideoView, 175–177
storing application data on devices, 363

VideoView control, 175–177

View container controls
AdapterView controls, 206–207
ArrayAdapter control, 204–205
CursorAdapter control, 205–206
data-driven containers, 204
ListView control, 207–208
using container control classes, 203

View controls
adding to ActionBar, 252
designing user interfaces for
compatibility, 322
parent and child views, 184–185
ViewGroup classes. See ViewGroup
classes
ViewGroup containers, 203

ViewGroup classes
built-in classes for layouts, 185–187
FrameLayout view, 193–195
GridLayout view, 198–202
LinearLayout view, 187–189
organizing user interface, 184–185
RelativeLayout view, 180, 189–193
subclasses used in layout design, 185
TableLayout view, 195–198
TextView control, 180
types of, 179
ViewGroup containers, 203
ViewHolder class, 296–297
ViewPager control, 210
Views
android.view package, 147–148
displaying text with TextView, 148–149
VideoView, 175–177
Visual appeal/usability, in testing applications, 474
Visual feedback, in planning user experience, 444–445
Voicemail content provider, 399

W
Watches, extending applications to, 338–339
Wear applications
extending applications to watches and cars, 338
quality guidelines, 455
Weather apps, implementing service for, 91
Web applications, 24
WebViewFragment class
implementing, 223–225
working with special types of fragments, 218
Welcome to Android Studio dialog, 58
White-box testing, 473
Widgets
grouping View widgets, 286
TextInputLayout, 275

Windows OSs, configuring for device debugging, 34
Wireframes, 443–444
Workflow. See Development workflow
Workspaces, organizing, 527
Wrappers, in Gradle, 606–609
WRITE_SECURE_SETTINGS permission, 399
WRITE_SETTINGS permission, 399
Writing tests, 489–491

X
XML
colors.xml file, 272
creating layout using XML resources, 179–181
defining Boolean resources, 125
defining color resources, 126
defining dimension resources, 127
defining drawable resources, 128
defining fragments, 215–216
defining menu resources, 135–136
defining tweened animation sequences, 134–135
editing manifest file, 96
manifest file as, 95
reading XML files, 370–371
resource files. See Resource files (XML)
resources stored as, 111–112
storing XML files, 114
utilities, 371
working with layouts, 140
XML Pull Parser, 370